

# Geodätisch-geophysikalische Arbeiten in der Schweiz

(Fortsetzung der Publikationsreihe  
«Astronomisch-geodätische Arbeiten in der Schweiz»)

herausgegeben von der

Schweizerischen Geodätischen Kommission  
(Organ der Akademie der Naturwissenschaften Schweiz)

**Einhundertfünfter Band**  
**Volume 105**

**On the adjustment, calibration  
and orientation  
of drone photogrammetry and  
laser-scanning**

Emmanuel Cledat

2020

Adresse der Schweizerischen Geodätischen Kommission:

ETH Zürich  
Institut für Geodäsie und Photogrammetrie  
Eidg. Technische Hochschule Zürich  
8093 Zürich  
Switzerland

Internet: <http://www.sgc.ethz.ch>

ISBN 978-3-908440-52-9

Redaktion des 105. Bandes:  
Dr. E. Cledat, J. Müller-Gantenbein, Prof. Dr. M. Rothacher  
Druck: Print-Atelier ADAG, Zürich



## VORWORT

Drohnen haben zu einer Revolution in einer Vielzahl von Bereichen geführt. Dazu gehören unter anderem die Kartografie und das grossflächige Monitoring. Sie stellen ein flexibles und kostengünstiges Mittel dar, um Beobachtungen aus der Vogelperspektive durchzuführen. Einerseits bleiben dabei die bewährten Prinzipien der Georeferenzierung von optischen Daten unverändert, da diese unabhängig vom Fahrzeug sind, auf dem die Instrumente installiert sind. Andererseits ist die Qualität der Messungen durch die reduzierte Instrumentengrösse begrenzt. Dies gilt sowohl für die optischen Sensoren als auch für die Navigationsinstrumente, wobei sich die Präzision der letzteren auf die Orientierung der ersteren auswirkt. Diese Dissertation stellt mehrere neuartige Methoden vor, um diese Einschränkungen anzugehen. Diese Methoden verbessern den Nutzen von Drohnen in der Präzisionskartografie qualitativ, sowohl in einer natürlichen als auch in einer bebauten Umgebung.

Die hier vorgestellten Forschungsarbeiten bauen Brücken zwischen den mathematischen/theoretischen Aspekten der Photogrammetrie und deren Anwendungen. Der Autor präsentiert zuerst die generellen Modelle für optische Sensoren (Kamera, LiDAR) und Navigationsinstrumente (GNSS, IMU) und deren rigorose Kombination durch eine generalisierte Optimierung. Diese Theorie wird später bei der Untersuchung verschiedener anspruchsvoller Orientierungs-, Kalibrierungs- und Kartierungsszenarien verfeinert. Zu den Szenarien gehören unter anderem die hochpräzise Korridor-Kartierung (Empfindlichkeit gegenüber der Kamerakalibrierung), die Nahbereichskartierung in einer gebirgigen Umgebung (Empfindlichkeit gegenüber einer reduzierten GNSS-Positionierungsgenauigkeit) und die kollaborative Kartierung mit terrestrischen Fahrzeugen und Drohnen (Empfindlichkeit gegenüber der Massstabsbestimmung bei einem kinematischen Objekt) oder mit einem Drohnentandem mit grossem vertikalen Abstand, wobei die Position der unteren Drohne von der oberen mit optischen Mitteln bestimmt wird.

Die Dissertation besteht aus konsolidierten wissenschaftlichen Publikationen, denen ein langes Einführungskapitel vorangestellt ist, das die Beobachtungsmodelle beschreibt, die im Rahmen des Optimierungsverfahrens implementiert wurden. Zwei Kapitel wurden in hochrangigen Fachzeitschriften veröffentlicht, die anderen in begutachteten Konferenzbeiträgen. Für das zweite dieser Kapitel wurde Herrn Cleat der "Best Young Author Award 2020" der ISPRS (International Society for Photogrammetry and Remote Sensing) verliehen.

**Prof. Dr. Jan Skaloud**  
EPF Lausanne  
Dissertationsleiter

**Prof. Dr. Markus Rothacher**  
ETH Zürich  
Präsident der SGK

## PREFACE

Les drones ont révolutionné de nombreux domaines, comme par exemple celui de la cartographie à grande échelle. Cette nouvelle technologie est une méthode flexible et abordable pour atteindre le point de vue des oiseaux. Les principes généraux du géoréférencement des images restent inchangés : le drone n'est qu'un moyen d'embarquer des capteurs. Cependant, il impose une limite de taille et de poids, et donc de qualité aux capteurs embarqués, que ce soient les capteurs imageurs, ou les capteurs de navigation qui participent à déterminer l'orientation des capteur imageurs. Cette thèse s'attaque à ce défi en proposant plusieurs méthodes innovantes permettant une amélioration qualitative pour l'utilisation de drone au service de la cartographie de haute précision dans des environnements naturels et construits.

La recherche synthétisée ici jette des ponts entre les aspects théoriques/mathématiques de la photogrammétrie et ses applications. Les modèles généraux décrivant les capteurs imageurs (appareil photos et LIDAR) et les instruments de navigation (GNSS et centrales inertielles) permettent de combiner rigoureusement les données des uns avec les données des autres via une optimisation générale. Cette théorie est ensuite déclinée pour être adapté à différents cas pratiques, tels que la cartographie de haute précision de corridor (sensible à la calibration de l'appareil photographique), la cartographie proche du sol en environnement montagneux (sensible à la baisse de la qualité du positionnement GNSS), la cartographie collaborative d'un tandem terrestre/aérien (sensible à la détermination de l'échelle d'une cible mobile) et la cartographie collaborative entre un duo de drone ayant une forte différence d'altitude pour lequel la position du drone volant en bas est déterminé visuellement par le drone volant au-dessus.

Cette thèse prend la forme de publications scientifiques étoffées, précédées par un long chapitre introductif décrivant les modèles d'observations implémentés et leur optimisation. Deux de ces chapitres sont publiés dans des journaux scientifiques de haut niveau, les autres dans des annales de conférences après examens par les pairs. Mr. Cledat a notamment reçu le prix du meilleur jeune auteur 2020 de la société de l'ISPRS (International Society for Photogrammetry and Remote Sensing) pour la publication basée sur le second chapitre de sa thèse.

**Prof. Dr. Jan Skaloud**  
EPF Lausanne  
Directeur de thèse

**Prof. Dr. Markus Rothacher**  
ETH Zürich  
Président de la CGS

## FOREWORD

Drones are disruptive in many domains, including in those of mapping and monitoring at large scales. They represent a new, flexible and cost-effective means to obtain a bird's-eye view perspective for such purposes. On one side, using a drone as a payload platform does not change the well-established principles of geo-referencing optical data as they are principally independent of the vehicle holding the instruments; on the other hand, the quality of on-board devices is limited by the reduced size that is required for use with drones. This applies to both navigation and optical sensors which in turn affects the orientation of the later. This thesis addresses such handicaps by proposing several novel methodologies that qualitatively improve the utility of drones in the service of environmental observations and high-accuracy drone mapping in natural and built-up environments.

The presented research bridges the gaps between the mathematical/theoretical aspects of photogrammetry and their applications. The author presents first the development of models for optical sensors (cameras, LiDAR) and navigation instruments (GNSS, IMU) and their rigorous combination via a general optimization framework. This theory is then refined when investigating different challenging orientation-calibration-mapping scenarios such as: high-precision corridor mapping (sensitivity to camera calibration), close-range mapping in mountain environments (sensitivity to suboptimal and temporal quality of GNSS positioning), collaborative mapping between terrestrial vehicles and drones (sensitivity to scale determination on kinematic target) and between a tandem of drones with important vertical separation where the position of the lower drone is determined optically from the upper one.

The dissertation takes the form of consolidated scientific publications that are preceded by a long introductory chapter describing the observation models implemented within the optimization procedure. Two chapters are published in highly ranked journals, and others in peer-reviewed proceedings. Among them is the second chapter for which Mr. Cledat received Best Young Author Award 2020 from the International Society for Photogrammetry and Remote Sensing (ISPRS).

**Prof. Dr. Jan Skaloud**  
EPF Lausanne  
Thesis director

**Prof. Dr. Markus Rothacher**  
ETH Zürich  
President of SGK



# Abstract

Centimetre level precision mapping is essential for many applications such as land-use, infrastructure inspection, cultural heritage preservation, and construction site monitoring. However, the acquisition and its preparation (in particular the setting of a ground control point network (GCPs)) are still expensive or even impossible in cluttered or dangerous areas. The recent development of UAVs together with the miniaturization of the sensors is a promising evolution for reducing costs and expand opportunities.

The sensors embedded on the drone: GNSS antenna, IMU, camera and (optional) LIDAR are light and often low-cost. The low quality of their raw measurements must be counterbalanced by their rigorous modeling in order to obtain accurate final results: if we cannot expect the sensors to be error-free, one must model these in order to correct them. This is achieved by in-situ calibration or on a dedicated calibration field, together with a rigorous fusion of the raw data acquired by the different sensors with the so-called bundle-adjustment method.

This thesis proposes several models to describe the behavior of the sensors, in order to hybridize them rigorously in the bundle-adjustment. Consistent datasets have been acquired on the field specifically to assess the relevance of both the sensor models and their hybridizing in complex photogrammetric processing.

The contribution of this thesis could be divided into two main categories. On one hand, this thesis suggests tools and recommendation to improve directly the procedures achieved by end-users using current UAV-mapping commercial solutions (in particular for the GCPs placement, for the choice of the camera calibration and model and for the flight-plan). On the other hand, this thesis put forward exotic methods (methods considered as exotic at the time of the writing of the thesis) such as Photo-LIDAR hybridizing and collaborative mapping achieved by a terrestrial-aerial tandem (a terrestrial vehicle holding a LIDAR, GNSS, imaging and inertial sensors followed by a drone conceived to proceed to airborne photogrammetry) or an aerial-aerial tandem (two drones flying in formation to proceed to airborne photogram-

---

metry).

The contribution of this thesis will permit to reduce costs, to improve the quality of mapping products and to enlarge the possibilities of mapping: in particular, map cluttered or inaccessible zones which are nowadays considered as difficult or even impossible to map.

Key words: Photogrammetry, computer-vision, data-fusion, optimization, Bundle-Adjustment, UAV, GNSS, IMU, Camera, LIDAR

# Résumé

Une cartographie de précision centimétrique est un médium essentiel pour de nombreuses applications tels que l'aménagement du territoire, l'inspection d'ouvrages d'arts, la préservation des monument archéologiques ou historiques et le suivit de chantier. Cependant, la préparation du relevé cartographique (en particulier, la mise en place de points d'appuis au sol : GCPs) et l'acquisition des données est couteuse, voire impossible dans des zones encaissées ou dangereuses. Le développement récent des drones, accompagné de la miniaturisation des capteurs est une évolution prometteuse pour réduire les coûts et élargir le champ des possibilités.

Les capteurs embarqués sur les drones : antenne GNSS, centrale inertielle, appareil photo et parfois LIDAR sont limité en poids et souvent peu-onéreux. La qualité limitée des mesures brutes des capteurs doit être compensé par une modélisation rigoureuse de ceux-ci pour obtenir des résultats finaux précis : si on ne peut pas espérer que ces capteurs soient exempts de défauts, il convient de les modéliser pour les corriger. Cela est réalisé grâce à des calibrations sur site, ou sur un champ de calibration dédié, ainsi qu'une fusion rigoureuse des données brutes acquises par les différents capteurs à l'aide d'une méthode appelée ajustement en bloc.

Cette thèse propose plusieurs modèles pour décrire les capteurs, afin de les hybrider de manière rigoureuse dans un ajustement en bloc. De nombreuses données ont été acquise sur le terrain spécialement pour évaluer la pertinence des modèles choisis pour modéliser les capteurs et la méthode d'hybridation dans leur processus photogrammétrique complexe.

Les contributions de cette thèse sont de deux ordres. D'une part, cette thèse propose des outils, des recommandations et des méthodes utilisables directement par des géomètres utilisant une solution commerciale de drone conçu pour les relevées photogrammétriques. En particulier pour le placement des GCPs, pour le choix de la méthode de calibration de la caméra et de son modèle, pour le choix du plan de vol, etc. D'autre parts, cette thèse étudie des concepts de relevé considérés aujourd'hui comme exotiques tels que l'hybridation Photogrammétrique-

---

Lasergrammétrie et la cartographie collaborative opérée par un tandem terrestre-aérien (une voiture munie de capteurs LIDAR, GNSS, inertiels et d'appareil photo suivit d'un drone équipé pour un relevé photogrammétrique) ou un tandem aérien-aérien (deux drones volant en formation équipé pour un relevé photogrammétrique).

Les contributions de cette thèse permettront de réduire les coûts, d'améliorer la qualité des produits cartographiques et d'ouvrir le champ des possibles, notamment à des zones considérées aujourd'hui comme difficile ou impossible à cartographier de manière efficace parce que trop encaissées ou trop difficilement accessibles.

Mots clef : Photogrammétrique, vision par ordinateur, fusion de données, optimisation, Ajustement en bloc, drone, GNSS, centrale inertielle, appareil photographique, LIDAR.



# Contents

<b>Abstract (English/Français)</b>	<b>i</b>
<b>List of figures</b>	<b>xi</b>
<b>List of tables</b>	<b>xvii</b>
<b>I Method description and analysis</b>	<b>9</b>
<b>1 Introduction to Photogrammetry with Adjustment Methods and Lie-Groups</b>	<b>11</b>
1.1 The photogrammetric Rosetta-Stone . . . . .	11
1.1.1 Literature review of photogrammetry notations . . . . .	12
1.1.2 The physics of a camera: the pinhole camera model . . . . .	12
1.1.3 Exterior Orientation . . . . .	15
1.1.4 Interior Orientation: principal distance and principal point . . . . .	16
1.1.5 The Photogrammetric Rosetta-Stone: an Overview . . . . .	17
1.1.6 Interior Orientation: Camera modeling and parametrization . . . . .	21
1.2 Other observation models . . . . .	33
1.2.1 Normal camera . . . . .	33
1.2.2 Camera rig . . . . .	33
1.2.3 LIDAR . . . . .	34
1.2.4 Spherical photos . . . . .	34
1.2.5 GCP . . . . .	36
1.2.6 GNSS antenna . . . . .	36
1.2.7 Orientation measurements . . . . .	37
1.2.8 Time-synchronisation between the sensors . . . . .	39
1.2.9 Concatenation of observation and parameters . . . . .	40
1.2.10 Covariance matrix of the parameters . . . . .	41
1.3 Expressing optimization problem . . . . .	42
1.4 Solving the optimization problem . . . . .	49
1.4.1 Gradient descent with small steps . . . . .	52

## Contents

---

1.4.2	Gradient descent with exact search . . . . .	53
1.4.3	Gauss-Newton Algorithm . . . . .	54
1.5	Lie-Group tutorial . . . . .	57
1.5.1	Motivation . . . . .	57
1.5.2	The Lie-Group of 2D rotations . . . . .	59
1.5.3	Skew-Symmetric matrix definition and exponential . . . . .	64
1.5.4	The $\mathbb{S}\mathbb{O}_3$ Lie-Group of 3D rotations . . . . .	69
1.5.5	Comparison of the $\mathbb{S}\mathbb{O}_2$ and the $\mathbb{S}\mathbb{O}_3$ Lie-Group . . . . .	74
1.5.6	Derivatives of rotation matrix logarithms . . . . .	76
1.6	Benchmarking Euler-Angles vs Lie-Groups . . . . .	80
1.6.1	Resistance to bad initialization . . . . .	81
1.6.2	Convergence rate . . . . .	82
1.6.3	Pose covariance matrix . . . . .	85
1.6.4	Conclusion for the comparison of the use of Euler-Angle and Lie-Groups in Bundle Adjustment . . . . .	85
<b>2</b>	<b>Toward Camera Calibration models and methods</b>	<b>87</b>
2.1	Introduction . . . . .	88
2.2	Camera Models . . . . .	90
2.2.1	Brown distortion model . . . . .	91
2.2.2	Orthogonal polynomials . . . . .	92
2.3	Camera Calibration . . . . .	92
2.3.1	Orthogonal Polynomials over-parameterization . . . . .	94
2.3.2	Parameters significance . . . . .	94
2.4	Re-estimating the Interior Orientation . . . . .	95
2.4.1	Leading Lead . . . . .	95
2.4.2	<i>a posteriori</i> covariance APC . . . . .	95
2.4.3	<i>a posteriori</i> covariance inflated APCI . . . . .	96
2.5	Experimental evaluation . . . . .	96
2.6	Discussion . . . . .	100
2.6.1	Camera calibration strategy . . . . .	100
2.6.2	Camera models . . . . .	111
2.6.3	Re-calibration . . . . .	113
2.7	Conclusions . . . . .	114
<b>3</b>	<b>Mapping quality prediction for RTK micro-drones operating in complex environ- ment</b>	<b>119</b>
3.1	Introduction . . . . .	120
3.1.1	Motivation . . . . .	120
3.1.2	Challenges . . . . .	122

3.1.3	Conventional approach . . . . .	122
3.1.4	Proposition . . . . .	123
3.2	Methodology . . . . .	124
3.2.1	Concept . . . . .	124
3.2.2	Basic relation . . . . .	126
3.2.3	Flight plan . . . . .	126
3.2.4	Camera model . . . . .	127
3.2.5	Ground control . . . . .	127
3.2.6	Tie-points . . . . .	128
3.2.7	Aerial position control . . . . .	130
3.2.8	Precision map . . . . .	132
3.3	Precision analysis . . . . .	134
3.3.1	Local . . . . .	134
3.3.2	Global . . . . .	134
3.3.3	Placement of ground control . . . . .	137
3.3.4	Evaluation of check-point misclosures . . . . .	137
3.4	Experimental validation . . . . .	140
3.4.1	Test sites and equipment . . . . .	141
3.4.2	Aerial position accuracy . . . . .	141
3.4.3	Tie-point density . . . . .	143
3.4.4	Local analysis . . . . .	144
3.4.5	Global analysis . . . . .	146
3.4.6	New GCP placement . . . . .	147
3.5	Conclusions and perspectives . . . . .	150
3.6	Appendix: Interpolation of estimated precision . . . . .	151
 <b>II Photo-LIDAR Fusion</b>		 <b>155</b>
 <b>4 Fusion of photo with airborne Laser Scanning</b>		 <b>157</b>
4.1	Introduction . . . . .	158
4.2	Review of optical correspondences . . . . .	160
4.2.1	Tie-points: Link image-image . . . . .	160
4.2.2	Coplanarity: Link LIDAR plane-LIDAR plane . . . . .	160
4.2.3	Point to Patch: Link LIDAR point-LIDAR pointcloud . . . . .	160
4.2.4	Homologous point: Link LIDAR point-LIDAR point . . . . .	160
4.2.5	Tie-Point to LIDAR point: Link Photo-LIDAR . . . . .	161
4.2.6	Tie-Point to LIDAR point cloud: Link Photo-LIDAR . . . . .	161
4.3	Methods and models . . . . .	162
4.3.1	Parametrizing variables . . . . .	162

## Contents

---

4.3.2	Camera model: collinearity equation . . . . .	164
4.3.3	Aerial control . . . . .	164
4.3.4	LIDAR model . . . . .	165
4.3.5	GCPs . . . . .	166
4.3.6	Bundle-Adjustment . . . . .	166
4.4	Results . . . . .	167
4.5	Conclusion . . . . .	171
4.6	Future outlook . . . . .	172
4.7	Appendix: Precision assessment of plane intersection method for Image-LIDAR matching . . . . .	173
<b>III Collaborative mapping</b>		<b>177</b>
<b>5</b>	<b>Mapping GNSS restricted environments with a drone tandem and indirect position control</b>	<b>179</b>
5.1	Introduction . . . . .	180
5.2	Indirect Position Control . . . . .	183
5.3	Technical Feasibility . . . . .	186
5.3.1	Tie-points Matched in Both D1 and D2 Nadir Images . . . . .	186
5.3.2	Visual Tracking and localizing of D1 from D2 . . . . .	186
5.4	Mapping Accuracy Prediction . . . . .	187
5.5	Conclusions . . . . .	191
<b>6</b>	<b>Compensating over- and underexposure in optical target pose determination</b>	<b>193</b>
6.1	Introduction . . . . .	194
6.2	Overview of the image processing pipeline . . . . .	196
6.3	Edge Formation Model . . . . .	197
6.4	Subpixel Edge Position . . . . .	199
6.5	Camera Pose Determination . . . . .	201
6.6	Correcting for the exposure bias . . . . .	204
6.7	Experimental Evaluation . . . . .	206
6.8	Conclusions . . . . .	211
<b>IV Appendix</b>		<b>221</b>
<b>A</b>	<b>3D Visualization toolbox for easy display of complex data from Matlab or Python</b>	<b>223</b>
A.1	User Manual for MATLAB . . . . .	225
A.1.1	ruby_create . . . . .	225
A.1.2	ruby_close . . . . .	225

A.1.3	Input in SketchUp	226
A.1.4	ruby_point	227
A.1.5	ruby_line	229
A.1.6	ruby_axis	230
A.1.7	ruby_pose	231
A.1.8	ruby_ellipsoid	233
A.1.9	ruby_plane	235
A.1.10	ruby_tin	236
A.1.11	ruby_antenna	238
A.1.12	ruby_arrow	239
A.2	User Manual for Python	241
A.2.1	ruby_create	241
A.2.2	ruby_close	241
A.2.3	Input in SketchUp	242
A.2.4	ruby_point	243
A.2.5	ruby_line	245
A.2.6	ruby_axis	246
A.2.7	ruby_pose	247
A.2.8	ruby_ellipsoid	250
A.2.9	ruby_plane	252
A.2.10	ruby_tin	253
A.2.11	ruby_theodolite	255
A.2.12	ruby_antenna	255
A.2.13	ruby_resection	256
A.2.14	ruby_arrow	257
<b>B</b>	<b>Bundle-Adjustment Algorithm: Code description</b>	<b>259</b>
B.1	Quick Start: import file and process it	259
B.1.1	Input file	259
B.2	Tuning parameters	263
B.3	Output and visualisation	266
B.3.1	Convergence rate	266
B.3.2	Covariances matrices	266
B.3.3	GCP residuals	266
B.4	Code and Algorithm description	266
B.4.1	Important variables	267
B.4.2	Update step	267
B.4.3	Observation models and derivations	267
B.4.4	Management of the position of the sub-matrix-block	272
B.4.5	Management of the construction of the sparse matrix	274

**Contents**

---

**Bibliography**

**294**

# List of Figures

1	Examples of mapping products . . . . .	2
2	Mapping of a canyon with photogrammetry (a and b) and with LIDAR (c). Overlap between two images (in a and b) is represented by diagonal stripes. . . . .	5
3	Illustration of terrestrial-aerial and aerial-aerial tandems . . . . .	6
1.1	Camera Obscura Principle (left) and Perspective principle from an engraving by Albrecht Dürer (right). (For concordance with next images, the engraving have been mirrored horizontally) . . . . .	12
1.2	Pinhole Camera Model . . . . .	13
1.3	Pinhole Camera Model with virtual plane front of the perspective point . . . . .	14
1.4	Illustration of the $B_1$ scaling parameter and the $B_2$ skewing parameter (special case where the principal distance $c$ corresponds exactly to half the height) . . . . .	24
1.5	True error of a trajectory sample computed from a consumer-grade IMU . . . . .	38
1.6	Computation of a sub-block of $\Sigma_{xx}$ . . . . .	41
1.7	Pseudo-distributions constructed by the $\rho$ function . . . . .	43
1.8	Typical examples for the $\rho$ function . . . . .	46
1.9	$\rho(v^2)$ . . . . .	46
1.10	Block covariance matrix . . . . .	48
1.11	An example of Multilateration problem: the triangles are the beacon-points, the lines are the distance measurements, and the circle represent fix distances to the beacon points. The red measurement from the blue beacon-point is probably a blunder . . . . .	50
1.12	Traduction of the multilateration problem into an optimisation problem . . . . .	51
1.13	Gradient descent with small steps. The iterative process on the left of the figure converges to the global minima (in blue), whereas the iterative process on the right converges to a local minima . . . . .	52
1.14	One step of the gradient descent with exact search . . . . .	53
1.15	Gradient descent with an exact search. The iterative process on the left of the figure converges to the global minima (in blue), whereas the iterative process on the right converges to a local minima . . . . .	54

## List of Figures

---

1.16	One step of the Gauss-Newton algorithm. The paraboloid is tangent on the initial guess (right part of the picture). The minimum of this paraboloid is represented by a little ball (bottom of the picture). The 'planimetric' position of this ball will be used as start for the next iteration (second ball above the first one) . . . . .	55
1.17	Gauss-Newton algorithm. The iterative process on the front of the figure converges to the local minima (in blue), whereas the iterative process on the back of the picture converges to a local minima . . . . .	56
1.18	Rotation $z$ applied to $\clubsuit$ . . . . .	59
1.19	Composition of rotations . . . . .	60
1.20	Tangent space of $\mathbb{S}\mathbb{O}_2$ . . . . .	62
1.21	axes adapted to compute exponential of the skew symmetric matrix . . . . .	65
1.22	A rotation (represented by the curved arrow) have several logarithms (black dots) . . . . .	68
1.23	Simulated data-set 1: Classical Aerial photogrammetry block . . . . .	81
1.24	Simulated data-set 2: Close range photogrammetry inspired from [91] and [162] . . . . .	82
2.1	Calibration flight CF1 . . . . .	99
2.2	Calibration flight CF2 . . . . .	100
2.3	Test Flight PF . . . . .	101
2.4	IO correlation matrix for Brown10 in CF1. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	102
2.5	IO correlation matrix for Brown15 in CF1. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	103
2.6	IO correlation matrix for Brown18 in CF1. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	104
2.7	IO correlation matrix for Poly15 in CF1. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	105
2.8	IO correlation matrix for Poly18 in CF1. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	106
2.9	IO correlation matrix for Brown10 in CF2. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	107
2.10	IO correlation matrix for Brown15 in CF2. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	108
2.11	IO correlation matrix for Brown18 in CF2. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	109
2.12	IO correlation matrix for Poly15 in CF2. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	110
2.13	IO correlation matrix for Poly18 in CF2. Black to white colors highlight low and high correlation coefficients, respectively. . . . .	111



2.14	Decomposition of the departure from collinearity vector, $\vec{v}_{P_i}$ into its radial ( $v_\rho$ ) and orthoradial ( $v_\theta$ ) components for a tie-point observed at image coordinates $z_{P_i}$ . . . . .	112
2.15	Distortions of Figure 2.16 corresponds to points close to the diagonal (inside a tolerance) whose distortion $\vec{v}$ have been projected into $\vec{v}_\parallel$ on the diagonal. . . . .	113
2.16	Radial component of the observed departures from collinearity, $\vec{v}_{P_i}$ (red dots), and as implied by the estimated IO model, $\vec{v}_M$ (blue curve), for points on the diagonal of the image plane. . . . .	114
2.17	Radial component of $\vec{v}_M$ as a function of image coordinates. Black corresponds to +10 pixels and red to -10 pixels. Note the negative distortions for the lower-right corner of the images while the distortion on the other corners is positive. This leads to relatively high values for the tangential distortions. . . . .	115
2.18	Orthoradial component of $\vec{v}_M$ as a function of image coordinates. Black corresponds to +2.5 pixels and red to -2.5 pixels . . . . .	116
3.1	Drone mapping in mountainous environment with prediction of satellite visibility and tie-point uncertainty. . . . .	120
3.2	Sky-view (azimuth - zenith coordinates) showing satellite geometry. . . . .	121
3.3	Conventional (left) and proposed (right) workflow from flight planning to survey execution with quality assessment. . . . .	123
3.4	Workflow for predicting mapping precision. . . . .	125
3.5	Optimal and non-optimal tie-point matching . . . . .	128
3.6	Empirical GNSS precision positioning as function of number of visible satellites and co-factor elements. . . . .	131
3.7	Visualization of predicted mapping precision for bad (left) and good (right) distribution of GCPs. . . . .	133
3.8	First four modes of simulated network with sub-optimally placed GCPs (in red). Each weakness is up-scaled by a factor of 100 for visualization. . . . .	136
3.9	Prediction of GNSS positioning precision in each image for two different mission start times at Z1. Color legend as in Fig. 3.6. . . . .	138
3.10	Comparison of true missclosure with accuracy prediction . . . . .	138
3.11	Projection of misclosure vector on principal axis of covariance matrix in two dimensions. . . . .	140
3.12	Actual GNSS positioning quality status obtained in post-processing (PPK) for two flights realized according to plan depicted in Fig. 3.9. . . . .	140
3.13	Comparison of GNSS positioning with the reference computed via indirect orientation with all GCPs. Black & red dashed-lines correspond respectively to $1\sigma$ & $2.57\sigma$ . . . . .	142
3.14	3D model of Z2 created using commercial software (Pix4D). Approximated scale added manually. . . . .	144

## List of Figures

---

3.15 Comparison of GNSS positioning with the reference computed via indirect orientation with all GCPs. Black & red dashed-lines correspond respectively to $1\sigma$ & $2.57\sigma$ . . . . .	146
3.16 Global spectral analysis of mapping precision: predicted (blue) and actual (red) misclosure as a function of the Eigen-Value index of . . . . .	147
3.17 New GCP placement proposal (green point) with respect to other GCPs (red points). . . . .	148
4.1 Description of the complete trajectory based on camera pose parameters only. No photos are taken at time $t$ . $\Gamma_t$ is not a parameter but could be computed from $\Gamma_i$ and $\Gamma_j$ via interpolation of double differences . . . . .	163
4.2 Test-site. Red triangles represent the GCPs, blue spheres represent <i>links</i> between Photogrammetry and LIDAR i.e. point on the ground measured both by LIDAR and on photos . . . . .	167
4.3 Point-Cloud section centered on a powerline. <i>Ground truth</i> point cloud (blue) has been computed using the Tactical Grade IMU trajectory. The point-cloud resulting from fusion of photogrammetry and LIDAR (yellow) uses consumer grade IMU and should be compared to the point-cloud using the same IMU without fusion (pink). . . . .	169
4.4 Difference between the generated point-cloud and the <i>Ground Truth</i> in East, North and Up produced using the trajectory of the consumer Grade IMU without (first row) and with (second row) fusion with photogrammetry. . . . .	170
4.5 Histogram of improvement ratio for all acquired points. 99 % of the values are represented in blue. . . . .	171
4.6 Line as a <i>link</i> between Photo and LIDAR data . . . . .	172
5.1 Rockfall protection structures and bridges in a 300 m deep gorge (Viamala, Thusis, Switzerland), where the GNSS reception is absent or unreliable for autonomous UAV guidance. . . . .	181
5.2 Schematic representation of the proposed method. Red shading represents field of view of the cameras embedded on D1 and D2 drone, blue lines represent image measurements, black dotted lines represent phase GNSS observation. . . . .	182
5.3 Planimetric position of tie-points. The black line are the UAV flight path. Yellow dots are seen by both N-S and E-W flight lines, blue dots only from N-S or E-W flight lines. . . . .	187
5.4 Contour lines of the canyon every five meters in height. . . . .	188

5.5	Tie-point precision as computed in <i>Case 3</i> . Poses are represented by pyramids (showing the field of view). In orange, error ellipsoids of D2 tie-points, in purple, error ellipsoids of D1 tie-points, in blue, error ellipsoids of D1 side camera tie-points. All ellipsoids are up-scaled by a factor 100. See the white double arrow for scale (10 m for the environment, 10 cm for the ellipsoids). . . . .	190
6.1	Two images of an ArUco target taken from the same camera pose with different exposure times, the underexposed image is on the left and the overexposed one on the right. The brightness of the underexposed image has been enhanced. The difference between the apparent location of the top and bottom edge is highlighted by red and blue lines. . . . .	195
6.2	The proposed target design. Overexposing the target results in a reduction of the apparent thickness of the black ring (purple arrow), while increasing the camera distance leads to a uniform scaling of the whole target. . . . .	196
6.3	A 3D edge point $P$ maps to its unknown projection $P'$ on the image plane via $\Pi(\cdot)$ . $\vec{n}$ is defined as orthogonal to the edge at $P'$ , while $\vec{m}$ is parallel to it. . . .	198
6.4	Pixel intensity of a black/white transition along $\vec{n}$ . The intensity $i$ is proportional to the amount of light exposing the sensor. $\tilde{i}$ is the measured pixel intensity (subject to overexposure) and $\hat{i}$ is the estimated from $\tilde{i}$ . The estimated edge position $\hat{n}_{P'}$ differs from the true value $n_{P'}$ by a few hundredths of pixels if the image exposure is ideal. However, in overexposed images the difference can reach several pixels. . . . .	200
6.5	Exposure bias as a function of the overexposure ratio for different values of $\sigma$ . . .	201
6.6	Geometric elements involved in the pose determination problem. The image plane is depicted in red. The pose of the camera reference frame in relation to the target frame is given by $T$ and $R$ . An object point $P_i$ lying on one of the two concentric circles of the target with angle $\theta_i$ projects to $\hat{P}'_i$ on the image plane. . . . .	202
6.7	Pixel intensity along the diameter of the target for an underexposed (top) and an overexposed (bottom) image. The brightness of the top image has been enhanced. In the underexposed image, the black/white transition is shifted by a few pixels towards the black area, which is highlighted by red and blue lines in the plots on the right. . . . .	205
6.8	Reprojection error for the considered edge points $\hat{P}'_i$ in an image without (on the left) and with (on the right) exposure compensation. . . . .	206
6.9	Schematic representation of the set-up to obtain the reference positions of the target and the camera. The total-stations network in $B$ , $B'$ and $B''$ measures the position of the camera ( $A$ ), the position of several ArUco targets ( $C$ ), and the position and orientation of the proposed target ( $D$ ). The camera $A$ is oriented from the total-stations and from image observation of the ArUco targets, $C$ . . .	207

## List of Figures

---

6.10	Target distance as a function of exposure time obtained with different algorithms with respect to the reference. It is possible to see that the distance obtained with exposure compensation, as in Section 6.6, is practically independent with respect to exposure time. . . . .	208
6.11	Distance error as a function of the true target distance for different pose determination algorithms. . . . .	209
6.12	Calculation flow. At the top, we see the input, the intermediates steps, and the output; at the bottom, the formulae used to calculate the parameters. . . . .	213
6.13	Schematic representation of the camera with two targets that the total-stations must measure. . . . .	217
A.1	Example of figure created with the toolbox . . . . .	223
A.2	Different type of points . . . . .	227
A.3	Display of a point-cloud of a church . . . . .	228
A.4	Example of line drawn by the library . . . . .	230
A.5	Axis definition . . . . .	231
A.6	Camera pose . . . . .	232
A.7	Ellipsoids . . . . .	233
A.8	Example of polygone drawn by the function <code>ruby_plane</code> . . . . .	235
A.9	Example of Digital Elevation Model created by <code>ruby_tin</code> with a gradient texture overlaid . . . . .	237
A.10	Example of single arrow . . . . .	239
A.11	Different type of points . . . . .	243
A.12	Example of line drawn by the library . . . . .	245
A.13	Axis definition . . . . .	247
A.14	Camera pose . . . . .	248
A.15	Ellipsoids . . . . .	250
A.16	Example of polygone drawn by the function <code>ruby_plane</code> . . . . .	252
A.17	Example of Digital Elevation Model created by <code>ruby_tin</code> with a gradient texture overlaid . . . . .	254
A.18	Example of single arrow . . . . .	257
B.1	Click File/export/export cameras... . . . .	260
B.2	Choose Blocks Exchange .xml . . . . .	260
B.3	GCP/CP Graphical User Interface. Green: Images position, Red: GCPs, Blue: CPs	262
B.5	Matrix A structure . . . . .	273
B.6	Adding sub-matrix in matrix A . . . . .	275
B.7	Adding sub-diagonal-matrix in matrix A . . . . .	276
B.4	Link between obs and terrain variable . . . . .	277

# List of Tables

1	Combinations of LIDAR sensors (first row) or photogrammetry (second row) on different kind of platforms . . . . .	3
1.1	Conversion of classical vs unitless coefficients . . . . .	23
1.2	Correspondences between Brown parameters and Orthogonal polynomials ones	33
1.3	Cost function definition . . . . .	47
1.4	Derivation of left and right multiplication update-step . . . . .	74
1.5	Derivatives of absolute and relative orientation . . . . .	80
1.6	Simulated data-set characteristics . . . . .	83
1.7	Benchmarking of Euler and Lie-Groups based Algorithms on two simulated data-set . . . . .	84
1.8	Standard deviation and correlation matrices outputted by Euler-Angles and Lie-Groups based algorithm in a situation of gimbal lock . . . . .	85
2.1	Calibration and production flights details. . . . .	97
2.2	Statistics of the checkpoints error for PF. Units are mm. . . . .	98
2.3	Significance of the IO parameters i.e. absolute value of the parameter divided by its standard deviation. . . . .	117
3.1	Texture categories with mean tie-points density and precision. . . . .	129
3.2	Tested scenarios in Z1 (top) and Z2 (bottom). . . . .	145
3.3	Improvement in mapping quality with new GCP placement. . . . .	149
3.4	Benchmarking of improvement in mapping quality with new GCP placement via exhaustive search. . . . .	149
3.5	Comparison of three covariance interpolation methods. . . . .	154
4.1	Comparison Photogrammetry/LIDAR . . . . .	159
4.2	Differents type of <i>links</i> for Photogrammetry, Laser-Scanning and Photo-LIDAR fusion . . . . .	162
4.3	4 examples of photo-LIDAR matches . . . . .	174
4.4	Performance evaluation . . . . .	175

## List of Tables

---

5.1 Accuracy prediction of the tie-points representing the canyon floor, and the canyon slopes (unit: mm) . . . . .	189
6.1 Geometric mean of the relative error for every position. . . . .	210
B.1 Structure of camera Interior Orientation model . . . . .	263
B.2 Notations . . . . .	268

# Introduction

## Context

Cartographic products such as maps and 3D models have shown countless successful applications for a tremendous number of industrial and civil applications. The (expressed or implicit) needs of the final user must be studied in order to adapt the cartographic product characteristics, and thus the method to produce it. The important product characteristics are the map extents (size of the described area), the spatial resolution (Ground Sampling Distance for a raster, sample interval for a point-cloud, level of detail for a CAD model), the precision and accuracy (how close elements of the map coordinate are from the reality<sup>1</sup>), the completeness (no element should be omitted), etc. Typical cartographic products are listed below.

- Orthophotos: rectified photo free from deformation and possibly superimposed on a map. [152].
- DEM (Digital Elevation Models) and DSM (Digital Surface Models). Differences between DSM at two different epochs permit to monitor changes such as erosion, accretion or snowpack thickness.
- Point-cloud, from which horizontal & vertical sections could be extracted.
- CAD textured 3D models [85].

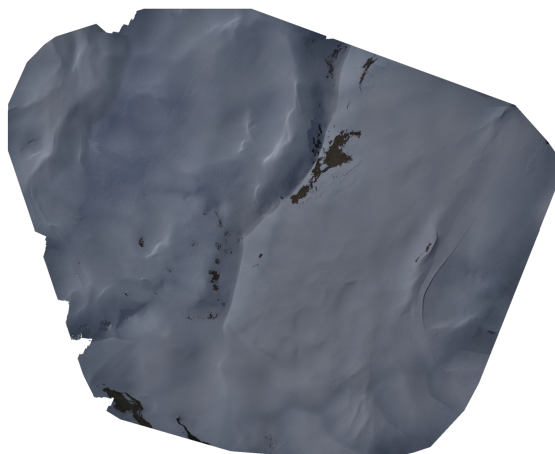
Two dimensional products such as orthophoto could be a basis for the creation of maps for whose elements are linked to a database that permits to proceed to combined semantic/geographic processing. This link between 2D objects and elements of a database is the fundamental of GIS (Geographic Information System). So as GIS, BIM (Building Information Modelling) is a CAD 3D model whose elements are linked to a database.

---

<sup>1</sup>Rigorous definition of precision and accuracy could be found in [51]

## List of Tables

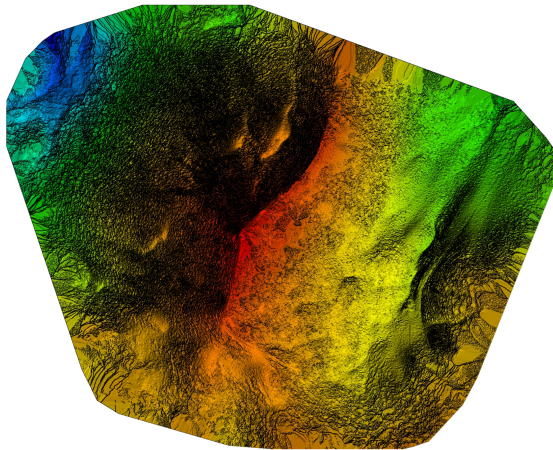
---



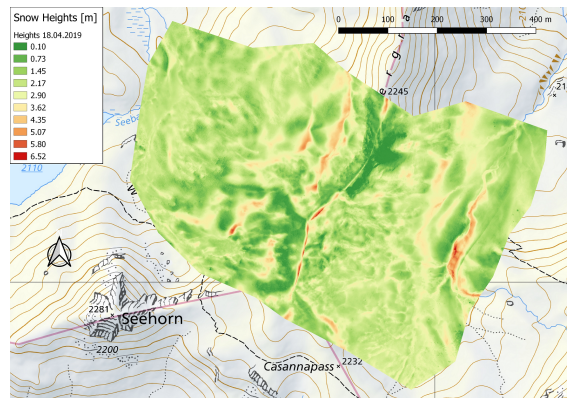
Ortho-photo



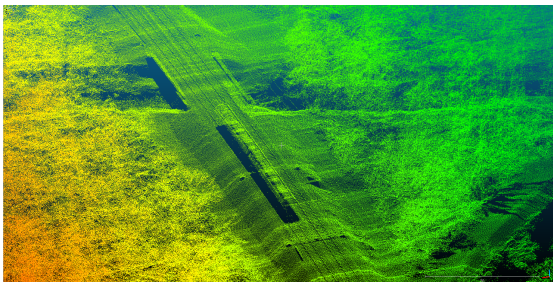
textured CAD model



DTM



Snow pack thickness



Point-cloud



Point-cloud section

Figure 1. Examples of mapping products

This Thesis will focus on airborne mapping acquisition with light payload ( $< 5\text{ kg}$ ): several sensors are mounted together on a flying platform (e.g. kite, fixed wing or multi-copter Unmanned Aerial Vehicle [UAV]) and the acquired data from the sensors are fused together.





Table 1. Combinations of LIDAR sensors (first row) or photogrammetry (second row) on different kind of platforms

Such a method permits to cover an area of typically less than  $1 \text{ km}^2$ , with a decimeter or centimeter-level precision.

The sensors used for airborne mapping acquisition are cameras, Light Detection and Ranging system (LIDAR), Global Navigation Satellite System (GNSS) antenna and Inertial Measurement Unit (IMU). A camera is a passive sensor that acquires images at the pose exposition time. A LIDAR is an active sensor that uses laser beams to measure the distance to the measured points. GNSS permits the computation of the absolute coordinates of the GNSS antenna phase center. An IMU is composed of 3 gyroscopes and 3 accelerometers measuring respectively angular velocity and specific force. Such measurements are traditionally fused with GNSS measurements (with methods such as forward Kalman filtering, and eventually backward filtering and smoothing) to compute the IMU trajectory, i.e. its position and orientation through time. Chapter 1 will present methods to fuse rigorously the raw data from these four sensors, in order to create a base for correct mapping products.

**Research objectives and thesis outline**

The goal of this research is to reduce the cost of State Of The Art (SOTA) airborne mapping acquisition while increasing its accuracy and its range of applications. This set of objectives was considered through three parts. The first one describes, models and simulates the pho-

togrammetry process in order to optimize the choices a surveyor may face while proceeding to a classical aerial photogrammetric survey such as camera calibration, flight-plan and GCP position. The aim of the second one is to improve the photogrammetric process by adding a LIDAR sensor on the same platform and operate rigorous hybridisation between the embedded sensors. Finally, the last part will split the observation acquisition to different platforms in a collaborative mapping. This thesis is also followed by the implementation of a Bundle Adjustment: a data-fusion algorithm to create 3D models out of the sensor observations, and to assess its quality. Chapter 1 presents the goal and the underlying theory of this algorithm. Chapter 2 tests this algorithm for real photogrammetry applications to assess the models proposed in Chapter 1. Chapter 3 couples this algorithm with a simulator of photogrammetric mapping to predict the precision of a planned survey in an environment where GNSS positioning quality varies temporally and spatially. Chapter 4 permits the algorithm to handle LIDAR observations. Finally, Chapter 5 will use this algorithm together with the simulator described in Chapter 3 to assess the feasibility and the potentiality of a new mapping concept: cooperative mapping with a drone duo.

## Detailed outline

### Part I

Chapter 1 presents fundamentals of photogrammetry, as well as the fusion with other sensors: GNSS, LIDAR, and IMU. It gives the needed theory for all the following chapters. Chapter 2 compares several camera calibration methods and models proposed in Chapter 1 through their applications on a difficult scenario: a photogrammetric corridor mapping with UAV. Such a study permits choosing the most appropriate camera model and more importantly the most appropriate method to improve mapping precision. Chapter 3 presents a software that permits to simulate photogrammetric, GNSS and IMU measurements in order to predict the mapping precision. This software permits to optimize the position of GCPs, the flight-time, and the Flight-plan to reduce costs while assessing the precision.

### Part II

Some objects or areas may be occluded when mapped via photogrammetry or LIDAR. However, the typology of the occlusion differs from one method to the other. For LIDAR, a necessary condition for a point to be measured is to be in line-of-sight with the LIDAR sensor. In photogrammetry, this condition is more difficult to reach since a point must be in line-of-sight with at least two poses. The so called *stereo-occlusion* of photogrammetry are thus more frequent than with LIDAR. In Figure 2.a for example, there is no overlap between the two photo. The overlap could be improved by reducing the base-line i.e. bringing the two poses

closer. However, close poses lead to a bad configuration for intersection calculation. The point positioning in the bottom of a canyon or terrain depression (Figure 2.b) will be less accurate in altimetry due to a lower angle of intersection. In these conditions the observations with a LIDAR (Figure 2.c) will be less affected. However, the accuracy of LIDAR is directly dependent on the accuracy of external orientation provided by INS/GNSS integration. This motivates for the hybridizing of these two methods as described in Chapter 4. In this chapter, LIDAR will be fused with the three other sensors embedded (and rigidly fixed) on the platform: the GNSS antenna, the IMU and the camera. The theory introduced in Chapter 1 is extended and the described method of fusion is compared with the conventional approach when each optical sensor is treated separately.

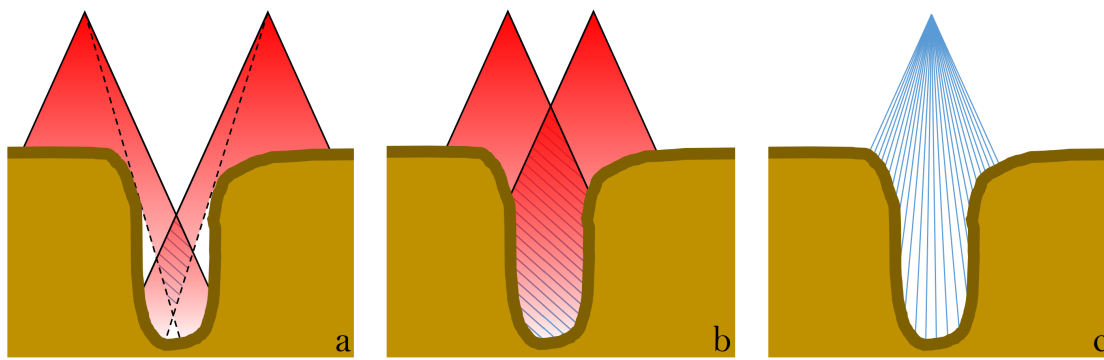


Figure 2. Mapping of a canyon with photogrammetry (a and b) and with LIDAR (c). Overlap between two images (in a and b) is represented by diagonal stripes.

### Part III

The previous parts consider mapping techniques performed by a single mapping system (e.g. a single UAV). Here, the potential of collaborative mapping is studied in Chapter 5 in aerial-aerial configuration and in Chapter 6 in terrestrial-aerial configuration. Chapter 5 describes the algorithms and mapping benefits when tandem of UAV is flying with vertical separation while the position of the lower drone is determined from the upper drone via optical means (Figure 3 right). Chapter 6 presents a solution for precise scale determination in so called kinematic-ground control point used within a terrestrial-aerial tandem called MapKite (Figure 3 left).

The graph on the next page presents the links between the different chapters of the thesis, as well as other publications/presentations published/presented in the scope of the Ph.D. The technical & scientific outcomes of this thesis contribute to various projects of the Geodetic Engineering Laboratory (TOPO-EPFL) such as PEACE4UAV, mapKITE and DoDo.



Figure 3. Illustration of terrestrial-aerial and aerial-aerial tandems

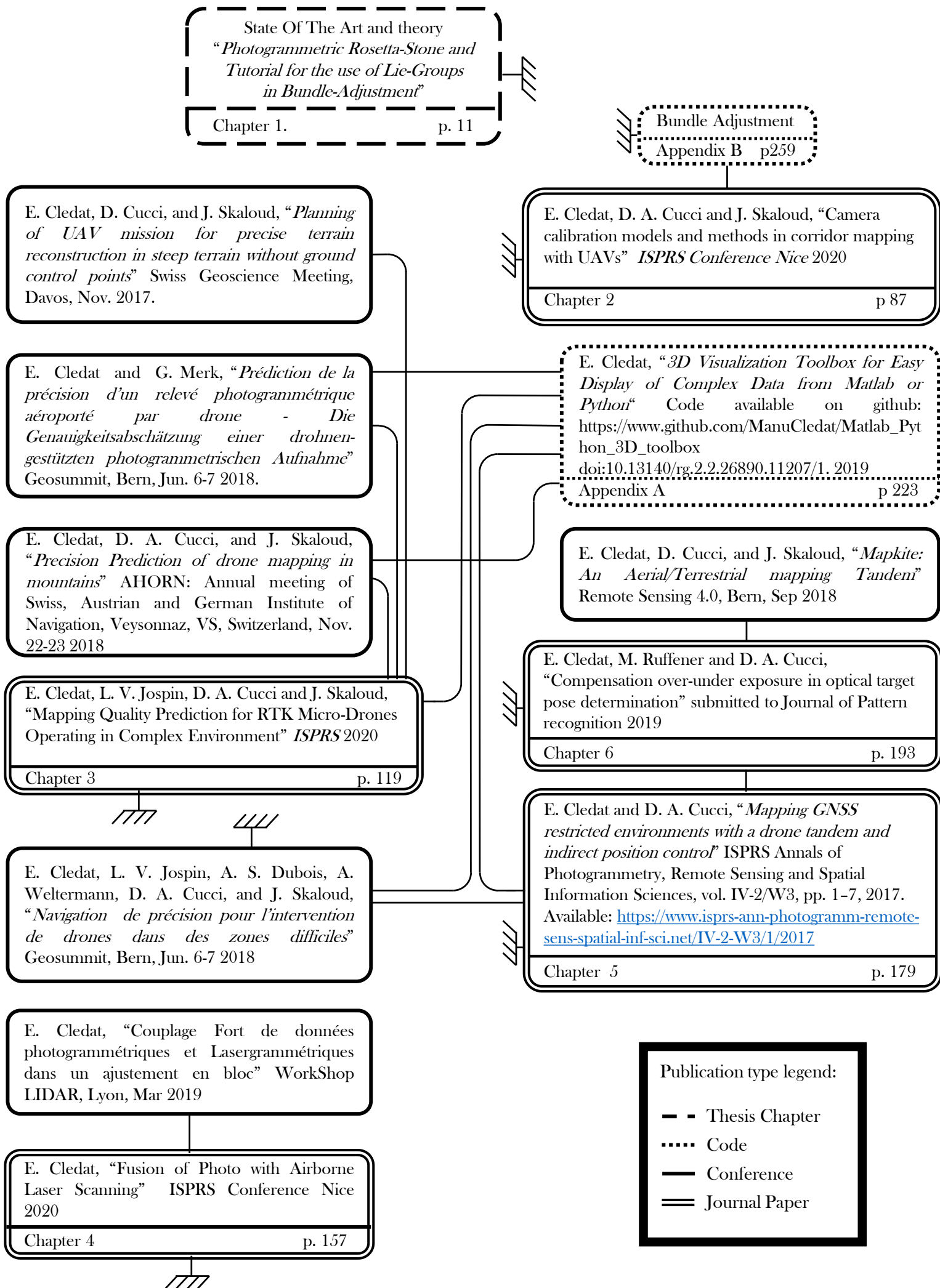


Figure 4: Graph of connections between the Ph.D Publications.



# **Method description and analysis Part I**





# 1 Introduction to Photogrammetry with Adjustment Methods and Lie-Groups

## 1.1 The photogrammetric Rosetta-Stone

The Rosetta-Stone is a  $\sim 1m$  tall stone from 196BC on which is written a decree in three languages: Ancient Egyptian using hieroglyphic script, Egyptian using Demotic script and Ancient Greek. It is famous for having permitted Jean-François Champollion to do a transliteration of the Egyptian scripts.

The aim of the chapter is to be the Rosetta-Stone of Photogrammetry. It will do a comparison of the notations and reasonings of traditional photogrammetry, modern photogrammetry, and Computer Vision. This document is intended for both expert in one field which wants to easily get into the other and for students which would like to learn photogrammetry using the notation that suit them best.

Photogrammetry is a method which has its source in the mid  $XIX^e$  century with cartographers and geodesist such as Dominique François Jean Arago and Aimé Laussedat. The long history of photogrammetry explains the evolution of the notation used in this field. Computer vision has a more recent history since it began in the late 60s with the idea of giving robots the ability to see. At that time, this problem was thought to be at the level of difficulty of a summer student's project [121]. The tools and discoveries done by Computer Vision experts are mainly complementary with respect to photogrammetry since the expectations of Photogrammetry is to achieve the best possible geometric precision and to assess its quality whereas Computer Vision tends to automatize this task. These two fields are driven by two main communities, where some scientists perceive this difference as a competition. Finally, the main outcomes of research in this field were synthesized in a book with a provocative title: *Photogrammetric Computer Vision* [51].

### 1.1.1 Literature review of photogrammetry notations

The presented synthesis stems from following work in Traditional photogrammetry: [6], [127], [12] Modern Photogrammetry: [138], [44], [18], [36] and Computer Vision: [110] [43] [164]. Photogrammetric Rosetta-Stone already exists. For example: [64] and [76] do a photogrammetric review in which both Traditional photogrammetry and Computer Vision are presented. However, this document will present the exact translation for the three notations.

### 1.1.2 The physics of a camera: the pinhole camera model

The most simple camera model is known as the pinhole camera model. This principle was discovered in the 16th century under the appellation *Camera obscura* (see Figure 1.1). Let a dark room whose only aperture is a single point (or at least small compared to the size of the room). An image of the world outside the *Camera obscura* will form on the wall back to this hole: the image plane. This *Camera obscura* is a large scale model of a pinhole camera model.



Figure 1.1. Camera Obscura Principle (left) and Perspective principle from an engraving by Albrecht Dürer (right). (For concordance with next images, the engraving have been mirrored horizontally)

For a pinhole camera model (or a *Camera obscura*), the aperture, modeled by a point is denoted as the perspective center. The distance between this perspective center and the image plane is known as the principal distance  $c$ , and its orthogonal projection on the image plane is the principal point  $pp$  (Figure 1.2).

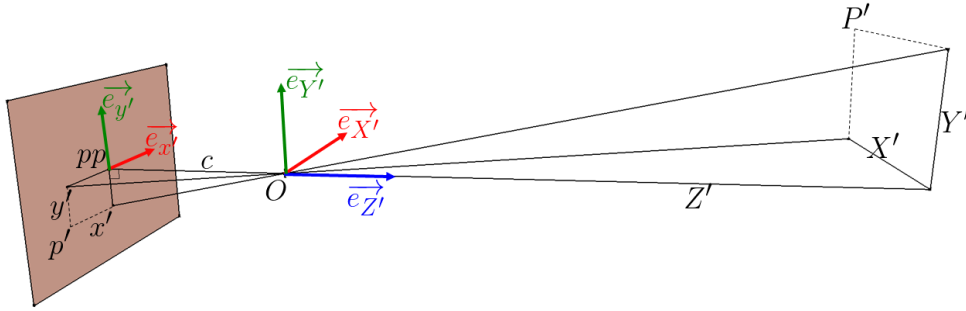


Figure 1.2. Pinhole Camera Model

The coordinates of a projection of a point from the *world* permit to compute the coordinates of its projection on the image plane thanks to the similar triangle theorem. Let a 2D Cartesian frame  $x'y'$  be associated to the image plane, and centered on the principal point. For convenience, the  $x'$  and  $y'$  axes are parallel to the borders of the image plane (usually a rectangle). Let a 3D Cartesian frame  $X'Y'Z'$  be associated to the camera and centered on the perspective point. The  $X'$  and  $Y'$  axes are collinear respectively to the  $x'$  and  $y'$  axes of the image plane. The  $Z'$  axes is perpendicular to the two others such that  $X'Y'Z'$  is a right-handed Cartesian frame. The  $Z'$  axes defined a line called the 'view axes'. A point  $P'$  from the *world*, whose coordinate are  $X' Y' Z'$  will project on the image plane to a point<sup>1</sup>  $p'$  whose coordinates are  $x' y'$ . The similar triangle theorem gives the following relationship –known as the collinearity equations- between  $X' Y' Z'$  and  $x' y'$  (see column *Traditional Photogrammetry* of table 1.1.5).

$$\begin{cases} x' &= -c \frac{X'}{Z'} \\ y' &= -c \frac{Y'}{Z'} \end{cases} \quad (1.1)$$

Note that the image is inverted with respect to the object (see Figure 1.1). This leads to a minus (-) sign in the collinearity equations. Moreover, the unit of  $x'$  and  $y'$  are given by the unit of  $c$ . Typically,  $c$  is expressed in *mm* so  $x'$  and  $y'$  are also expressed in *mm*.  $x'$  and  $y'$  could also be expressed in *pixels* if the image plane is a digital sensor (e.g. CCD or CMOS),  $c$  should be thus express in *pixels*. For convenience, a virtual plane is defined front of the perspective center, at a distance of 1. The formed image will first be upright (not inverted), and the coordinates  $\tilde{x}$  and  $\tilde{y}$  of the projected points  $\tilde{p}$  will be unit-less. Thus, points are first

<sup>1</sup>These points were used to be recognized and matched on analogic photos by experimented operators using optico-mechanical stereo-comparators. The informatic revolution permits to automatize the process of recognizing the same object point in several images with algorithms detections and matching algorithms (SIFT, SURF, ORB, KAZE, LDAHash, etc. see [106], [137], [120], [150], [88]).

projected on this virtual plane with a unitary principal distance before being scaled by the actual principal distance. The projection of a point (from the *world*) to this virtual plane is given by the following application (see column *Modern Photogrammetry* of tabular 1.1.5).

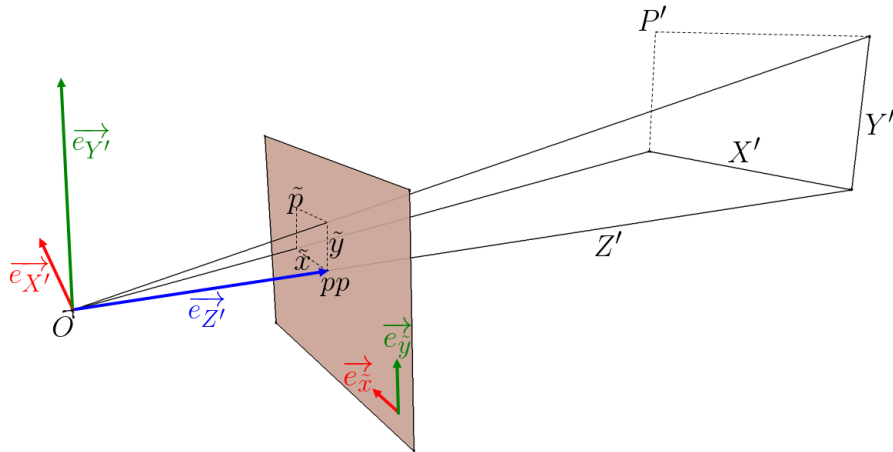


Figure 1.3. Pinhole Camera Model with virtual plane front of the perspective point

$$\pi: \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \mapsto \frac{1}{Z'} \begin{bmatrix} X' \\ Y' \end{bmatrix} \quad (1.2)$$

Computer Vision notation use homogeneous coordinates formalism. In homogeneous coordinates, a vector (such as  $[X'; Y'; Z']$ ) is intended to be *normalized* with the  $\pi$  function (to the vector  $[X'/Z'; Y'/Z']$ , or to a vector with a last unitary coordinate:  $[X'/Z'; Y'/Z'; Z'/Z']$  with a variant  $\tilde{\pi}$  of the projection function). To keep a third coordinate in the projected vector (and thus to prevent a vector to be projected), a last unitary coordinate is added to the  $[X'; Y'; Z']$ , thus denoting a homogeneous coordinate:  $[X'; Y'; Z'; 1]$ . The next section will present how this last unitary coordinate will be used to compute 3D transformation. When the vector needs to

be projected, the last unitary coordinate could be removed thanks to the *projection matrix*  $\tilde{\Pi}$ .

$$\begin{bmatrix} X'/Z' \\ Y'/Z' \\ 1 \end{bmatrix} = \tilde{\pi} \left( \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\tilde{\Pi}} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \right) \quad (1.3)$$

In this section, we have seen three different notations to express the pinhole camera model. Relationships between a 3D point expressed in a frame attached to the camera  $[X'; Y'; Z']$  and its image on a virtual plane has been established. However, two important steps are missing for characterizing the image formation model. In one hand, the camera axes are (in general) not parallel to the ones of the mapping coordinate system which raise the need of determining the so called *exterior orientation* (section 1.1.3). On the other hand, lens effect must be modeled, together with the determination of the image formation model in an operation called the *interior orientation* (sections 1.1.4 and 1.1.6).

### 1.1.3 Exterior Orientation

The aim of photogrammetry is to derive 3D coordinates of the observed object from measurements in images taken of these objects. Determination of the so called pose (position and orientation) of the camera is fundamental for this task. A *world* frame, or better a mapping frame is defined for the object to be mapped, in which all the objects coordinates will be expressed. This section will focus on the frame transformation between the mapping frame and the sensor frame.

Consider a point whose 3D-coordinates in *world* frame are  $P^W = [X_P; Y_P; Z_P]$ . The 3D-coordinates of this same point in sensor frame are  $P^c = [X'; Y'; Z']$ . The formulae to transform the coordinates from sensor frame to *world* frame is simply:  $P^W = T^W + R_c^W P^c$  where  $T^W = [X_0; Y_0; Z_0]$  is the 3D-position of the perspective center in *world* frame and  $R_c^W$  is the rotation matrix between sensor frame and *world* frame.

Expanding the frame transformation  $P^W = T^W + R_c^W P^c$  leads to the following equation system.

$$\begin{cases} X_P = X_o + r_{11}X' + r_{12}Y' + r_{13}Z' \\ Y_P = Y_o + r_{21}X' + r_{22}Y' + r_{23}Z' \\ Z_P = Z_o + r_{31}X' + r_{32}Y' + r_{33}Z' \end{cases} \quad (1.4)$$

We have seen in the previous section that the Computer Vision community uses homogeneous coordinate which bears certain advantages. It consists in adding a fourth unitary coordinate to 3D vectors (or a third unitary coordinate to 2D vectors). This unitary coordinate simplifies handling rotations and translation. Indeed, the operation of the translation and the rotation will be encapsulated to a single transformation matrix as below (where  $\mathbf{0}$  represent the  $1 \times 3$  null vector:  $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ ).

$$\begin{bmatrix} P^W \\ 1 \end{bmatrix} = \begin{bmatrix} R_c^W P^c + T^W \\ 1 \end{bmatrix} = \begin{bmatrix} R_c^W & T^W \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^c \\ 1 \end{bmatrix} \quad (1.5)$$

Detailing 1.5 bellow helps understanding the homogeneous coordinate formalism used in Computer Vision.

$$\begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & X_o \\ r_{21} & r_{22} & r_{23} & Y_o \\ r_{31} & r_{32} & r_{33} & Z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \quad (1.6)$$

Note that after operation such translations and rotations, the fourth coordinate must remain unitary. This implies the last row of the transformation matrix (containing the translation and the rotation) to be  $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ .

#### **1.1.4 Interior Orientation: principal distance and principal point**

The coordinates  $x'$  and  $y'$  are express in image unit ( $[mm]$  or  $[pix]$ ) in a frame centered around the principal point  $pp$ . This principal point is not necessarily exactly at the center of the image, and is not a convenient origin for image measurements. The goal of the internal orientation is to convert the image measurements i.e. the coordinate of distinguishable points on the image

## 1.1. The photogrammetric Rosetta-Stone

$\ell_x, \ell_y$  to  $x'$  and  $y'$ . For old metric argentic cameras, the frame origin for image observation were defined to be on the center of the image. Since the position of argentic film or the argentic glass could have been not exactly known with respect to the lens, the so called *fiducial marker* were used as crosses on the camera corners that were impressed on the image while the photo was taken. The origin and the axes of the frame were defined with respect to these *fiducial marker*. The basic conversion of  $\ell_x, \ell_y$  to  $x', y'$  is a simple translation.

$$\begin{cases} \ell_x = x' + pp_x \\ \ell_y = y' + pp_y \end{cases} \quad (1.7)$$

If the origin is the center of the image, the coordinates of the principal point:  $pp_x$  and  $pp_y$  are usually very small (typically few pixels, or even fraction of pixels). However if the origin of the image is defined to be on a corner (as in *Computer Vision*),  $pp_x$  and  $pp_y$  values are close to half of the height and half of the width respectively.

With numeric photos, it is more convenient to define the origin of the picture as the center of the upper-left pixel, the first axes going down and the second pointing right. This frame is left-handed, leading to a second inversion of the image. Equation 1.7 turn to 1.8 by multiplying the unitless values of  $\tilde{x}$  and  $\tilde{y}$  by the principal distance  $c$ .

$$\begin{cases} \ell_x = c \cdot \tilde{x} + pp_x \\ \ell_y = c \cdot \tilde{y} + pp_y \end{cases} \quad (1.8)$$

This last equation could be expressed with homogeneous coordinate formalism by introducing the so called calibration matrix  $K$ .

$$\begin{bmatrix} \ell_x \\ \ell_y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} c & 0 & pp_x \\ 0 & c & pp_y \\ 0 & 0 & 1 \end{bmatrix}}_K \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} \quad (1.9)$$

### 1.1.5 The Photogrammetric Rosetta-Stone: an Overview

The full collinearity equation is obtained by linking the external orientation (3D transformation) to the internal orientation (2D transformation) via the pin-hole projection function.

If the calculation is exactly the same between *traditional photogrammetry* and *modern photogrammetry* (the formalism is just slightly different) the ones of Computer Vision is different.

The pose determination defined in photogrammetry to describe the external orientation is the position  $T^W$  and the orientation  $R_c^W$  of the camera with respect to the *world*, whereas the Computer Vision formalism tends to use the position  $T^c$  and the orientation  $R_W^c$  of the *world* with respect to the camera. A possible explanation of this paradigm is the inheritance from robotics where the robot is the studied object. The *world* in which the robot evolve is modeled in the frame of the robot to plan its future actions. However, this is not suitable for precise mapping purposes because this leads to artificial correlations between  $T^c$  and  $R_W^c$ . Moreover, it leads to difficulties in the interpretation of the correlations between the  $T^c$  and the  $R_W^c$  of different camera poses.

The unitless image coordinates could be computed by projecting (with the projection matrix  $\tilde{\Pi}$  and the projection function  $\pi$ ) the 3D point in camera frame. The computed unitless coordinates could be translated to image observation by the  $K$  matrix as in equation 1.9.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \begin{bmatrix} X'/Z' \\ Y'/Z' \\ 1 \end{bmatrix} = \underbrace{\tilde{\pi}}_{\tilde{\Pi}} \left( \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_W^c & T^c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^W \\ 1 \end{bmatrix} \right) \quad (1.10)$$

Equation 1.9 could be fused together with 1.10 because  $\pi$  and  $K$  are commutative<sup>2</sup>. In contrary

<sup>2</sup>The commutativity of the function  $\pi$  and the matrix  $K$  require to redefine  $\tilde{\pi}$  as below in order to satisfy the dimension of the studied vectors.

$$\tilde{\pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \mapsto \frac{1}{Z'} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \quad (1.11)$$

The null elements of the matrix  $K$  permit the commutation between  $\tilde{\pi}$  and  $K$ .

$$\underbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}}_K \tilde{\pi} \left( \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \right) = \tilde{\pi} \left( \underbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}}_K \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \right) \quad (1.12)$$

Note that the only difference between the function  $\pi$  and the function  $\tilde{\pi}$  is the last unitary coordinate outputted by  $\tilde{\pi}$



## 1.1. The photogrammetric Rosetta-Stone

to the common use in homogeneous formalism, the  $K$  matrix is multiply by a vector whose last coordinate is not unitary. The coordinates of the principal point  $pp_x$  and  $pp_y$  are thus multiplied by  $Z'$ , added to  $c \cdot X'$  and  $c \cdot Y'$  before being divided again by  $Z'$ . This complex mathematical operation that is applied to the coordinates of the principal point does not reflect any physical phenomena and will lead to complexification in the next section.

$$\begin{bmatrix} \ell_x \\ \ell_y \\ 1 \end{bmatrix} = \tilde{\pi} \left( \underbrace{\begin{bmatrix} c & 0 & pp_x \\ 0 & c & pp_y \\ 0 & 0 & 1 \end{bmatrix}}_K \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\tilde{\Pi}} \begin{bmatrix} R_W^c & T^c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^W \\ 1 \end{bmatrix} \right) \quad (1.13)$$

Next table summarizes notations and reasoning of the three fields: *Traditional Photogrammetry*, *Modern Photogrammetry*<sup>3</sup> and *Computer Vision*

<sup>3</sup>The notations used in [18] correspond to the *modern photogrammetry* with the same willingness for clarity and conciseness. The following Table retranscribes the notation of [18] and links them to the notations presented in this document for the *modern photogrammetry*.

<i>modern photogrammetry</i>	[18]
$\dots + pp$	$T_2(\dots, pp)$
$c \cdot \dots$	$S(c, \dots)$
$\pi(\dots)$	$H(\dots)$
$R^T \dots$	$L(R^T, \dots)$
$\dots - T$	$T_3(\dots, -T)$
$f \cdot \pi(R^T(P - T)) + pp$	$T_2(S(c, H(L(R^T, T_3(P, -T))))), pp)$

	Traditionnal Photogrammetry	Modern Photogrammetry	Computer Vision
Sensor $\rightarrow$ World	$\underbrace{\begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix}}_{P^W} = \underbrace{\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix}}_{T^W} + \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{R_c^W} \underbrace{\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}}_{P^c}$	$P^W = T^W + R_c^W P^c$	$\begin{bmatrix} P^W \\ 1 \end{bmatrix} = \begin{bmatrix} R_c^W & T^W \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^c \\ 1 \end{bmatrix}$
World $\rightarrow$ Sensor	$\begin{cases} X' = r_{11}(X_P - X_o) + r_{21}(Y_P - Y_o) + r_{31}(Z_P - Z_o) \\ Y' = r_{12}(X_P - X_o) + r_{22}(Y_P - Y_o) + r_{32}(Z_P - Z_o) \\ Z' = r_{13}(X_P - X_o) + r_{23}(Y_P - Y_o) + r_{33}(Z_P - Z_o) \end{cases}$	$P^c = R_c^{W^T} (P^W - T^W)$	$\begin{bmatrix} P^c \\ 1 \end{bmatrix} = \begin{bmatrix} R_W^c & T^c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^W \\ 1 \end{bmatrix}$ $\begin{cases} R_W^c = R_c^{W^T} \\ T^c = -R_c^{W^T} T^W \end{cases}$
Pin-Hole camera	$\begin{cases} \frac{X'}{Z'} \\ \frac{Y'}{Z'} \end{cases}$	$\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ $\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \mapsto \frac{1}{Z'} \begin{bmatrix} X' \\ Y' \end{bmatrix}$	
Ideal camera World $\rightarrow$ Image	$\begin{cases} \ell_x = c \frac{r_{11}(X_P - X_o) + r_{21}(Y_P - Y_o) + r_{31}(Z_P - Z_o)}{r_{13}(X_P - X_o) + r_{23}(Y_P - Y_o) + r_{33}(Z_P - Z_o)} + pp_x \\ \ell_y = c \frac{r_{12}(X_P - X_o) + r_{22}(Y_P - Y_o) + r_{32}(Z_P - Z_o)}{r_{13}(X_P - X_o) + r_{23}(Y_P - Y_o) + r_{33}(Z_P - Z_o)} + pp_y \end{cases}$	$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = c \cdot \pi \left( R_c^{W^T} (P^W - T^W) \right) + pp$	$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = \pi \left( K \tilde{\Pi} \begin{bmatrix} R_W^c & T^c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} P^W \\ 1 \end{bmatrix} \right)$ $K = \begin{bmatrix} c & 0 & pp_x \\ 0 & c & pp_y \\ 0 & 0 & 1 \end{bmatrix}$

### 1.1.6 Interior Orientation: Camera modeling and parametrization

Up to this point, the camera have been modeled by the pin-hole camera model. The pine-hole of real cameras is realized by a (simple or compound) lens. A simple lens could define a single piece of transparent material (usually glass) with cylindrical symmetry and different curvature on each side. The axes of symmetry of the simple lens is simply called the axes. However, in most optical devices, several lenses are assembled together on the same axes. The concatenation of lenses could be modeled by a single lens (according to the characteristics of the concatenated lenses) and is called a compound lens<sup>4</sup>

A lens is designed (accordingly to the Snell-Descartes law) such that a beam of light-rays coming from infinity (i.e. parallel one with the other) converging to a single point. Light-rays parallels to the lens axes converge to the focal point. The focal length  $f$  is the distance between the focal point and the center of the lens. The focal plane is the plane normal to the lens axes at a distance  $f$  from the center of the lens. Theoretically, a beam of parallel light rays (not necessarily parallel to the lens axes) converges to a single point situated on the focal plane.

If a camera observes a distant object (typically, more than 5 m for a wide angle lens and more than 20 m for a telephoto lens), it should be focused at infinity. This is achieved by making the focal plane coincide with the image plane. Under this condition, the view axis coincides with the optical axis of the lens and the focal length  $f$  and the principal distance  $c$  are equivalent. We would like here to disambiguate a common mistake in the literature. The focal length and the lens axes are intrinsic property of the lens whereas the principal distance and the camera axes of view is defined by the mount of the lens and of the image plane inside the camera. In particular, the focal length do not always correspond to the principal distance.

If the camera observes a close object, the focus could be done on the object to acquire a clear image (without blur)<sup>5</sup> 6. It means that the lens will be translated along the camera axes toward the image plane. The lens axes and the camera axes will still be collinear, but the principal distance will be smaller than the focal length.

Important departures from collinearity must be modelled. For instance, the imperfection of the lens itself, the possible misalignment of the lens axes with the camera axes, the possible default of planarity of the sensor (argentic glass, argentic film, CCD or CMOS sensor) representing the image plane, and the possible imperfections of the digitalisation of argentic images or of the CCD or CMOS sensor. Two distortions models chosen from the literature are presented here: the Brown model, and orthogonal polynomials such as Ebner functions.

---

<sup>4</sup>The French translation for simple lens is *une lentille* whereas a compound lens is called *un objectif*.

<sup>5</sup>In practice, in photogrammetry, the focus is very often at infinity for two mains reasons. First, the camera are calibrated on this position and fixed on this position in order to have stable and well known calibration parameters. Second, blur images could be exploited under certain conditions and could even leads to better results than focused images (A rigorous definition of the blur based on the Shannon point-spread function is used in Chapter

### Brown model

These imperfections could be modeled by the Brown model<sup>6</sup> The publication usually cited when referring to the Brown model is a more recent one [24] focusing on the calibration of these parameters. In equation 1.14, the – sign have been omitted as in [22, 23, 24] under the assumption the image frame is left-handed. described below (we recall that  $x'$  and  $y'$  are expressed in  $[mm]$  or in  $[pix]$ ).

$$\begin{cases} \ell_x &= x'(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + \dots) + (P_1 (r^2 + 2x'^2) + 2P_2 x' y') (1 + P_3 r^2 + \dots) + pp_x \\ \ell_y &= y'(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + \dots) + (2P_1 x' y' + P_2 (r^2 + 2y'^2)) (1 + P_3 r^2 + \dots) + pp_y \end{cases} \quad (1.14)$$

The (theoretic) cylindrical symmetry of the lens is represented by the polar coordinates with:  $r^2 = x'^2 + y'^2$ .

In the modern photogrammetry formulation and in the Computer Vision formulation, points are first projected on a virtual plane with a unitary principal distance before being scaled by the actual principal distance. Recent adaptations of Brown model are applied on the unitless coordinates  $\tilde{x}$  and  $\tilde{y}$  (the radius is redefined:  $\tilde{r}^2 = \tilde{x}^2 + \tilde{y}^2$ ). Conceptually, it represents better the image formation process since the lens distortions are first considered, and then the projection on the camera image plane is considered.

$$\begin{cases} \ell_x &= c\tilde{x}(1 + \tilde{K}_1 \tilde{r}^2 + \tilde{K}_2 \tilde{r}^4 + \tilde{K}_3 \tilde{r}^6 + \dots) + c(P_1 (\tilde{r}^2 + 2\tilde{x}^2) + 2P_2 \tilde{x}\tilde{y}) (1 + P_3 \tilde{r}^2 + \dots) + pp_x \\ \ell_y &= c\tilde{y}(1 + \tilde{K}_1 \tilde{r}^2 + \tilde{K}_2 \tilde{r}^4 + \tilde{K}_3 \tilde{r}^6 + \dots) + c(2P_1 \tilde{x}\tilde{y} + P_2 (\tilde{r}^2 + 2\tilde{y}^2)) (1 + P_3 \tilde{r}^2 + \dots) + pp_y \end{cases} \quad (1.15)$$

The classical Brown model (equation 1.14) have a numerical drawback. If  $x'$  and  $y'$  are expressed in  $[\mu m]$  or in  $[pix]$ ,  $r^2$ ,  $r^4$ ,  $r^6$  will be very large (order of magnitude  $10^{12}$  for  $r^6$  if the order of magnitude of  $x'$  or  $y'$  is  $10^3$  (which is usual when using  $[pix]$ ). This lead to very small values for the coefficients  $K_1$ ,  $K_2$ ,  $K_3$ ,  $P_1$ ,  $P_2$  ...

Moreover, the units of these coefficients are clumsy: if  $x'$  and  $y'$  are expressed in  $[pix]$ ,  $P_1$  and

---

<sup>6</sup>In the first publication of Brown [22], the radial distortions were described as in equation 1.14, but the tangential distortion, which are said to represent the decentering of the lens axes with respect to the camera axes were described analytically, with decentering angles, leading to complex formalism. A second publication [23] simplifies this expression by introducing the parameters  $P_1$ ,  $P_2$ ,  $P_3$  to get all the terms of equation 1.14. Note that the terms  $P_3$  and beyond are the coefficients of a polynomial in  $r^2$ , but they scale the vector field of the tangential distortions described by  $P_1$  and  $P_2$ .

## 1.1. The photogrammetric Rosetta-Stone

$P_2$  are in  $[pix^{-1}]$ ,  $K_1, P_3$  are in  $[pix^{-2}]$ ,  $K_2, P_4$  are in  $[pix^{-4}]$  and so on. The unitless-based Brown model solve this issue using unitless coefficients. Both sequences  $\{K_1, K_2, K_3, \dots\}$  and  $\{\tilde{K}_1, \tilde{K}_2, \tilde{K}_3, \dots\}$  are decreasing, but the decrease rate of the second is less dramatic, leading to fewer numerical issues.

The following conversion table 1.1 compares classical and unitless coefficients. The effect of these different parameters are graphically represented by the quiver-plots on the next pages.

$\tilde{K}_1 = \frac{1}{c^2} K_1$	$\tilde{P}_1 = \frac{1}{c^2} P_1$
$\tilde{K}_2 = \frac{1}{c^4} K_2$	$\tilde{P}_2 = \frac{1}{c^2} P_2$
$\tilde{K}_3 = \frac{1}{c^6} K_3$	$\tilde{P}_3 = \frac{1}{c^2} P_3$

Table 1.1. Conversion of classical vs unitless coefficients

For the sake of clarity, the unitless distortion model is encapsulated in the function  $\xi_1$ .

$$\begin{aligned}
 \xi_1: \quad \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\
 \tilde{p} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} &\mapsto (1 + \tilde{K}_1 \tilde{r}^2 + \tilde{K}_2 \tilde{r}^4 + \tilde{K}_3 \tilde{r}^6 + \dots) \tilde{p} + \begin{bmatrix} (\tilde{P}_1 (\tilde{r}^2 + 2\tilde{x}^2) + 2\tilde{P}_2 \tilde{x}\tilde{y}) \\ (2\tilde{P}_1 \tilde{x}\tilde{y} + \tilde{P}_2 (\tilde{r}^2 + 2\tilde{y}^2)) \end{bmatrix} (1 + \tilde{P}_3 \tilde{r}^2 + \dots) \\
 &\text{where} \quad \tilde{r}^2 = \tilde{x}^2 + \tilde{y}^2
 \end{aligned} \tag{1.16}$$

The modern-photogrammetry expression for the image formation model could thus be extended by the distortions.

$$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = c \cdot \xi_1 \left( \pi \left( R_c^{W^T} (P^W - T^W) \right) \right) + pp \tag{1.17}$$

However, this last model do not account for possible difference of scaling in  $x$  and  $y$ , nor possible skewing (illustrated on Figure 1.4). The complete Interior-Orientation camera model

could thus be represented by the function  $\xi = \xi_2 \circ \xi_1$  where  $\xi_2$  models the position of the lens with respect to the image plane, and the possible deformation of the image plane.  $B_1$  accounts for possible difference of scaling between  $x$  and  $y$  (i.e. non square pixel), and  $B_2$  accounts for possible skewing of the image (i.e. non rectangular pixels).

$$\xi_2: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} c + B_1 & B_2 \\ 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} pp_x \\ pp_y \end{bmatrix} \quad (1.18)$$

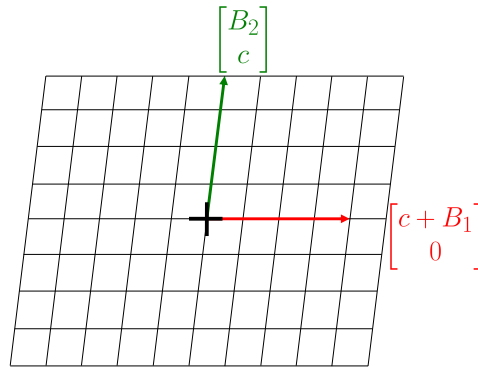


Figure 1.4. Illustration of the  $B_1$  scaling parameter and the  $B_2$  skewing parameter (special case where the principal distance  $c$  corresponds exactly to half the height)

Finally, Equation 1.17 could simply be re-written as Equation 1.19.

$$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = \xi \left( \pi \left( R_c^{WT} (P^W - T^W) \right) \right) \quad (1.19)$$

### Orthogonal Polynomials

The Brown model is based on a physical modeling of the lens and the camera. [15] reasons for employing models based on mathematical considerations rather than on a physical modelling of the sensors. A basis of polynomials is built to be orthogonal to a given scalar product, in order to avoid correlations. Each coordinate is independent from the other and computed

## 1.1. The photogrammetric Rosetta-Stone

with a second order degree polynomials from the coordinates given by the  $\pi$  function:  $\tilde{x}$  and  $\tilde{y}$ .

$$\left\{ \begin{array}{l} \ell_x = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 \end{bmatrix} \\ \ell_y = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 \end{bmatrix}^T \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 \end{bmatrix} \end{array} \right. \quad (1.20)$$

For numerical reasons, the quadratic terms  $\tilde{x}^2$  and  $\tilde{y}^2$  could be balanced respectively with the values  $b_x$  and  $b_y$  which could be chosen as half the image width and half the image height in unit-less coordinates (i.e. width and height divided by twice the nominal principal distance).

$$\left\{ \begin{array}{l} \ell_x = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 - \frac{2}{3}b_y^2 \end{bmatrix} \\ \ell_y = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 - \frac{2}{3}b_y^2 \end{bmatrix} \end{array} \right. \quad (1.21)$$

Some parameters of the full 18 parameters (of double second degree polynomial) are correlated with external-orientation. For this reason, [15] suggests to introduce the following constraints between the parameters in order to remove this correlation.

$$\left\{ \begin{array}{l} b_{13} + 2 \cdot a_{22} = 0 \\ a_{31} + 2 \cdot b_{22} = 0 \\ a_{12} - b_{21} = 0 \end{array} \right. \quad (1.22)$$

These constraints lead to the following modification in the orthogonal polynomial.

$$\begin{cases} \ell_x = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} a_{11} & b_{21} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ -2 \cdot b_{22} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 - \frac{2}{3}b_y^2 \end{bmatrix} \\ \ell_y = \begin{bmatrix} 1 \\ \tilde{x} \\ \tilde{x}^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} b_{11} & b_{12} & -2 \cdot a_{22} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} 1 \\ \tilde{y} \\ \tilde{y}^2 - \frac{2}{3}b_y^2 \end{bmatrix} \end{cases} \quad (1.23)$$

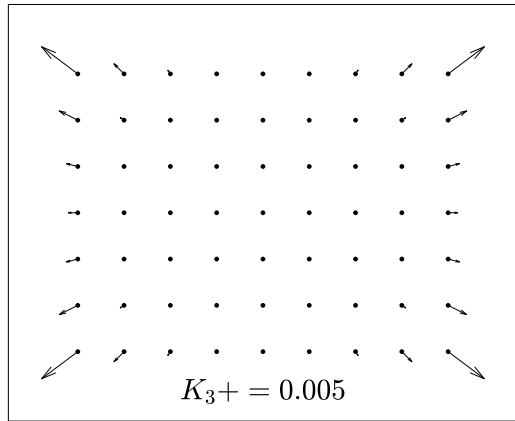
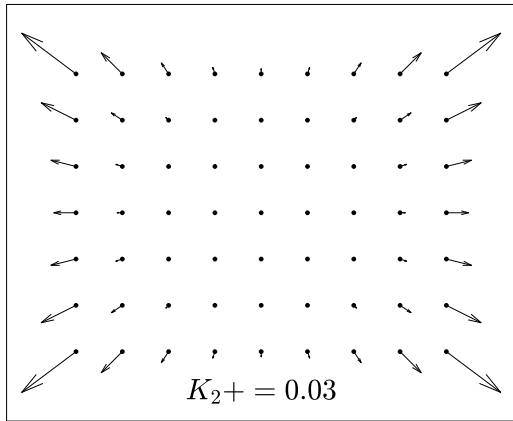
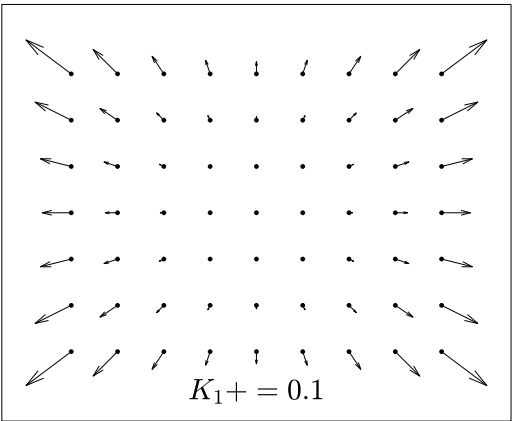
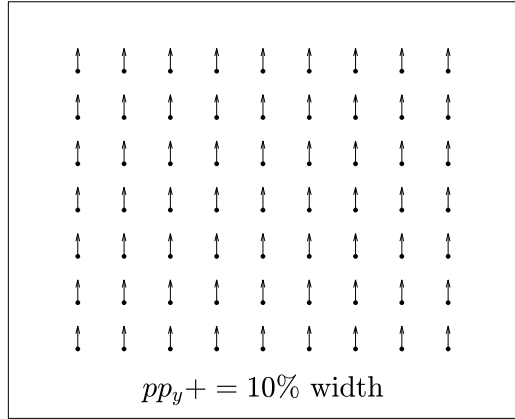
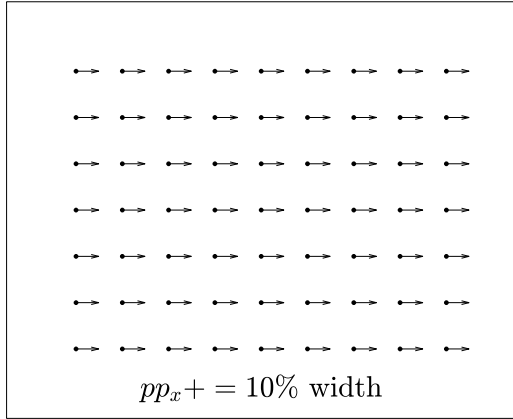
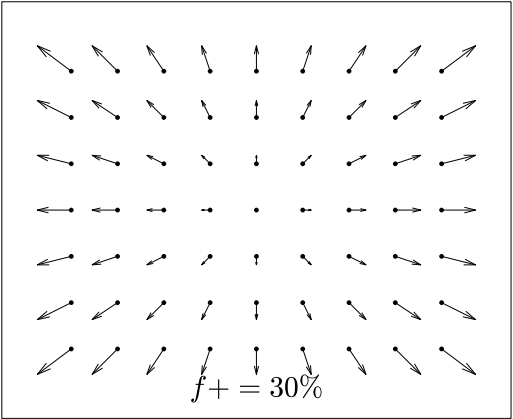
[15] suggests as well to introduce constraints between  $a_{11}$ ,  $b_{11}$ ,  $a_{21}$  and  $b_{12}$  as in Equation 1.24, however we propose to let these parameters free since they account for possible variation in principal point and principal distance.

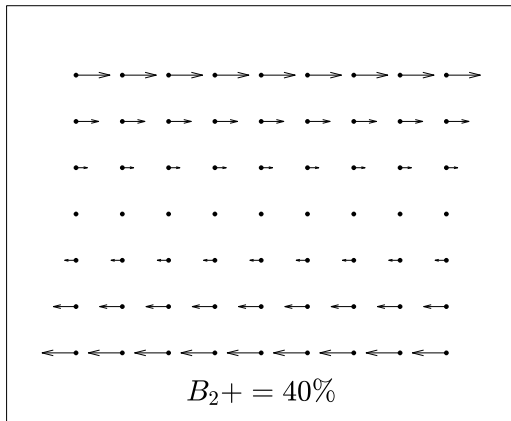
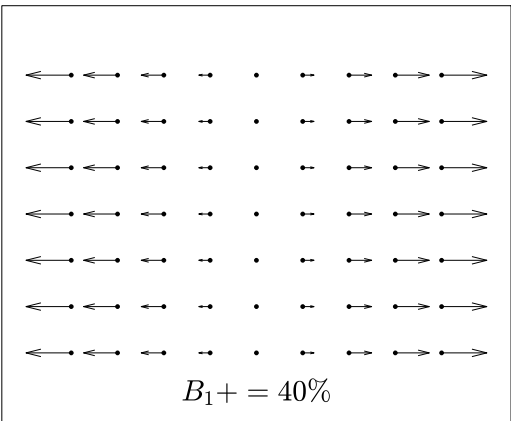
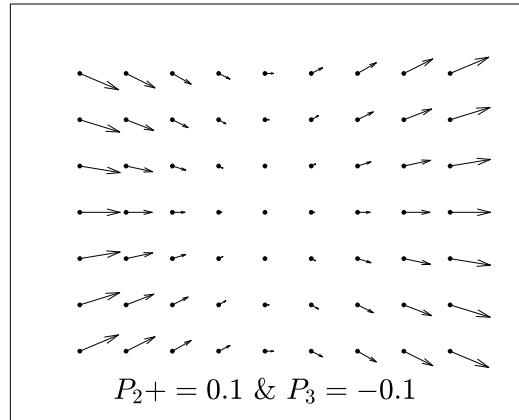
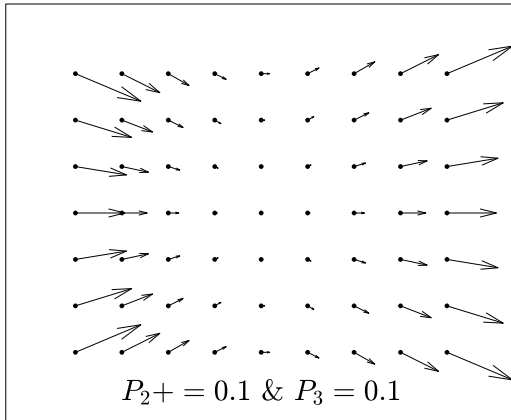
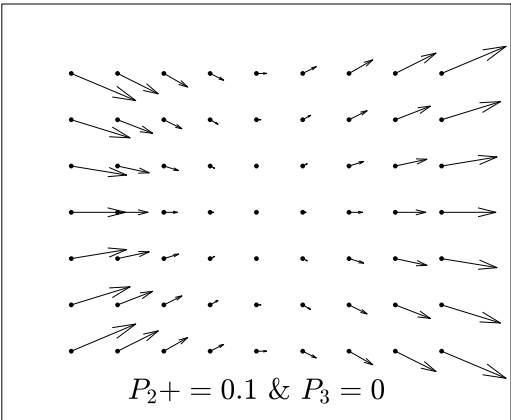
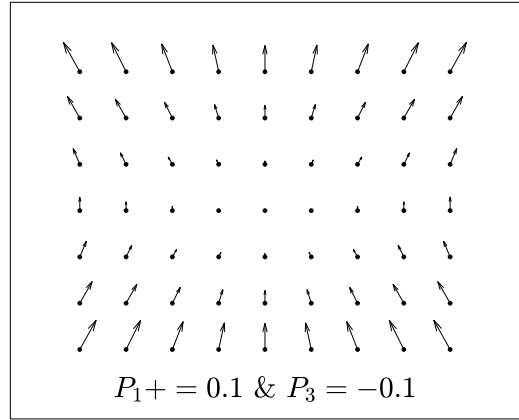
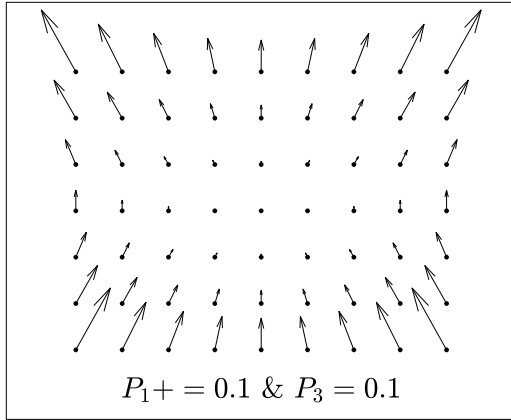
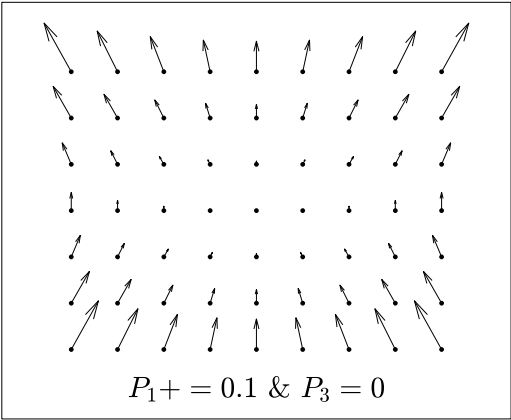
$$\begin{cases} a_{11} = b_{11} = 0 \\ a_{21} + b_{12} = 0 \end{cases} \quad (1.24)$$

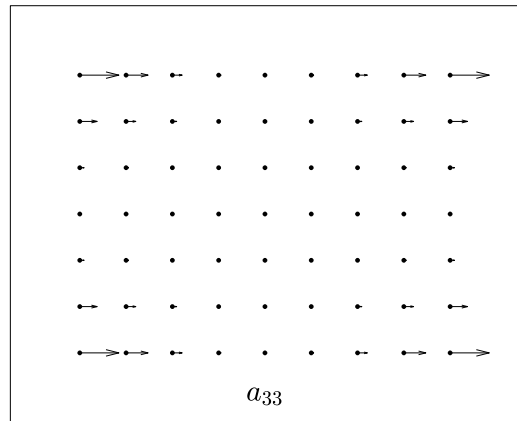
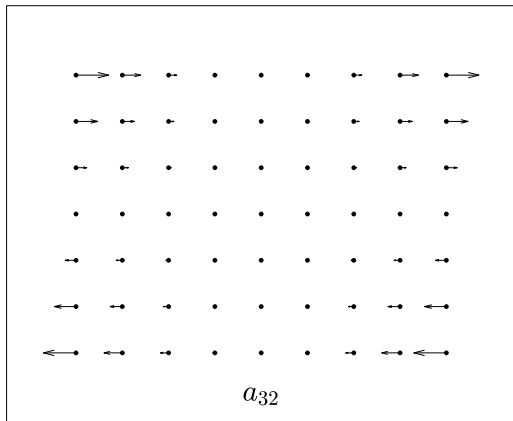
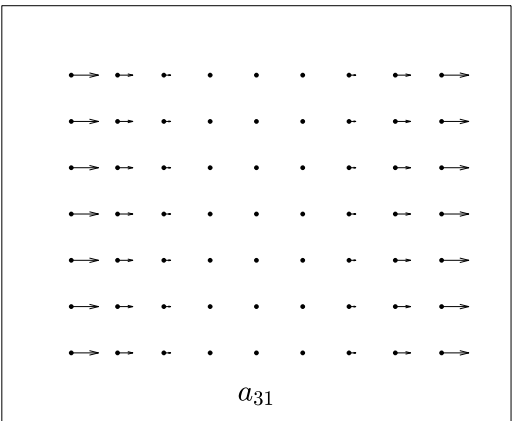
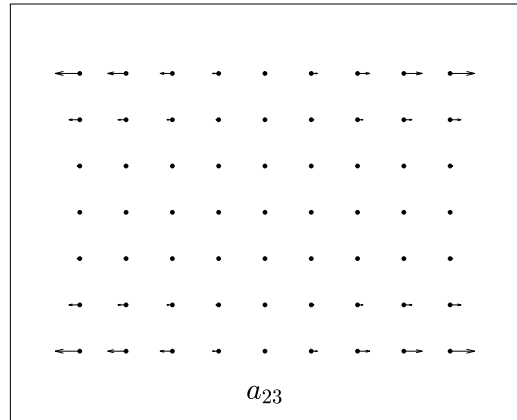
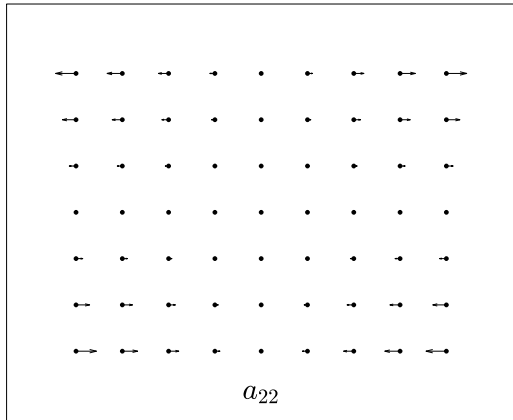
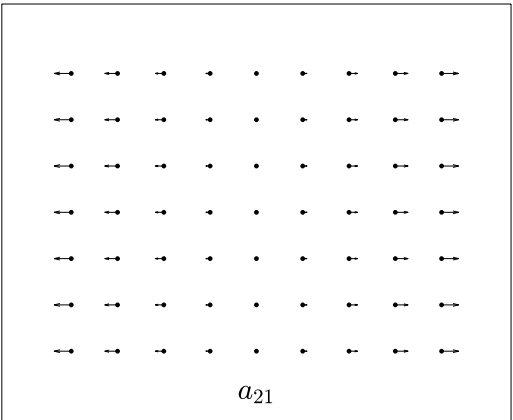
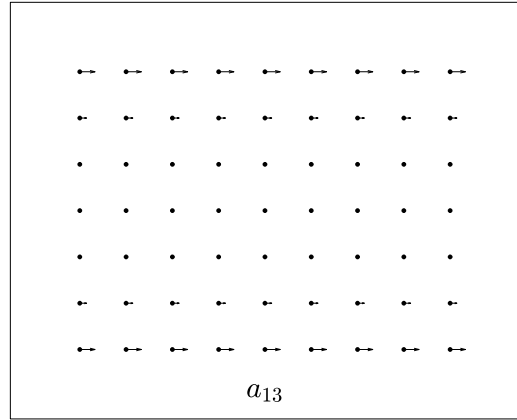
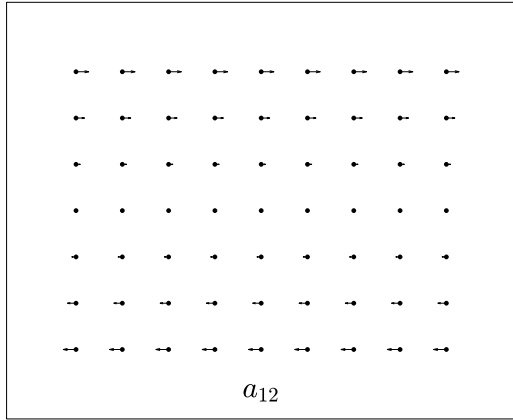
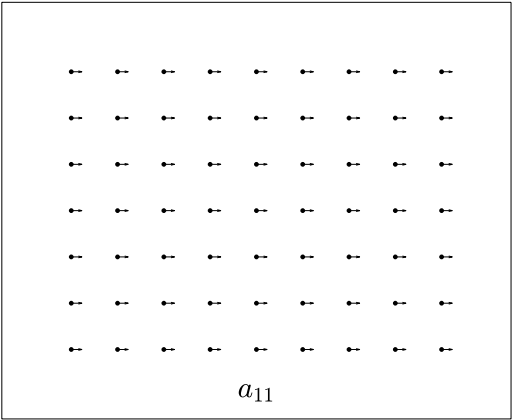
Rigorous relationship between parameters from Brown model and orthogonal polynomials could not be achieved, since the polynomial order is different from the physical model. The only possible translation could occur in the special case when there is no distortions, thus, most parameters (which are not in the following Table 1.2) are null.

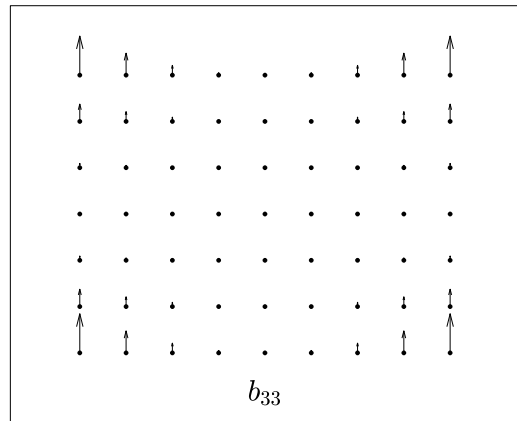
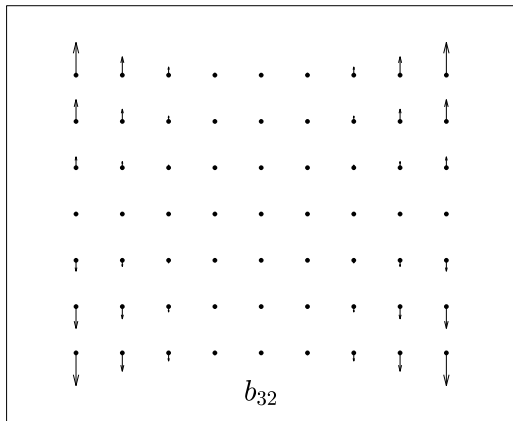
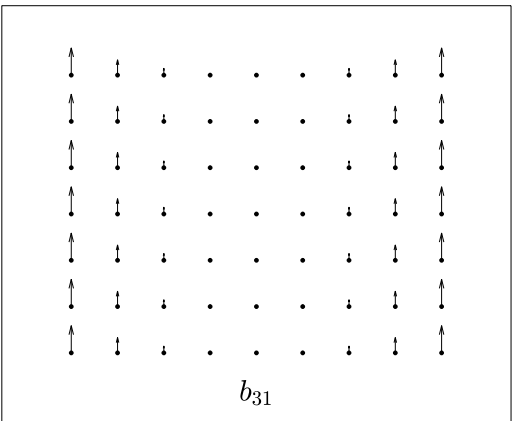
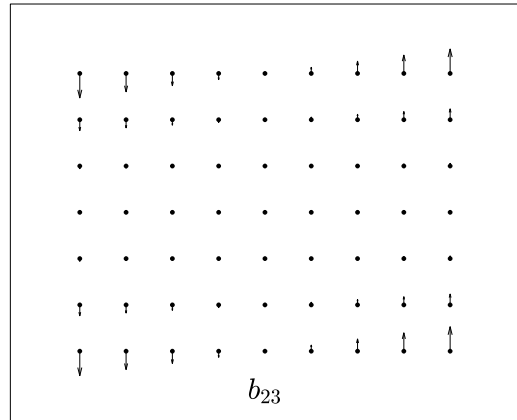
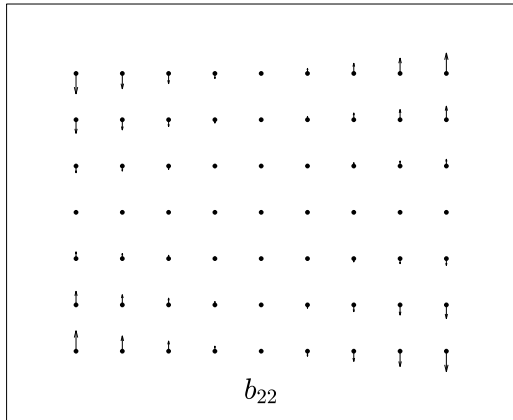
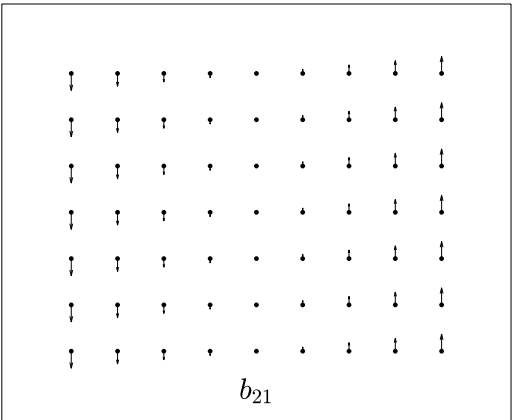
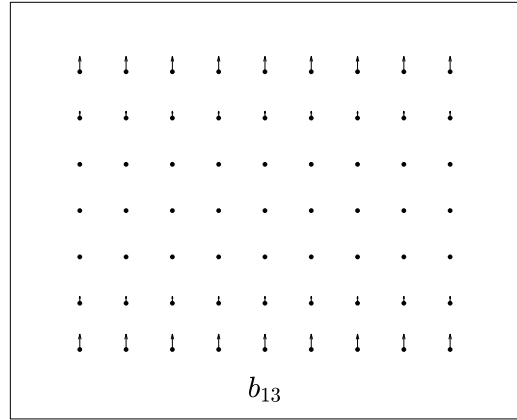
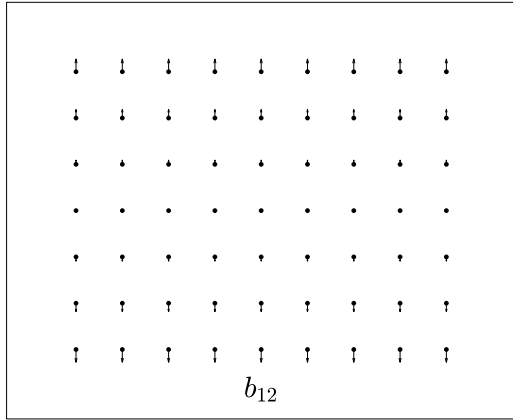
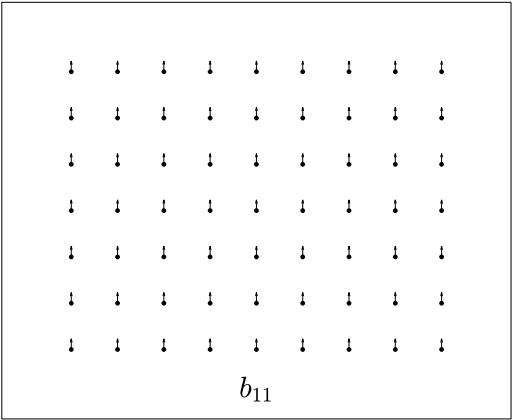
Similarly to the Brown model, orthogonal polynomials (with or without constraints) could be encapsulated in a function  $\xi$  and be used in Equation 1.19. These two camera calibration models are studied and compared in Chapter 2 for a particular high-quality UAV camera. The following quiver-plots represents the effect of each parameters of Brown and orthogonal polynomials respectively.

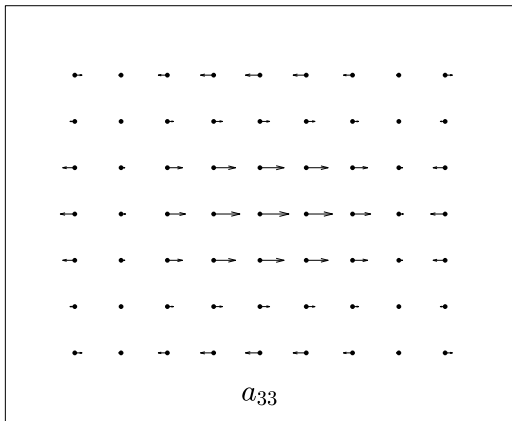
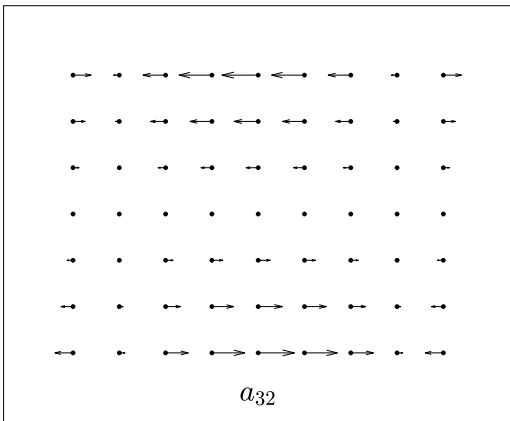
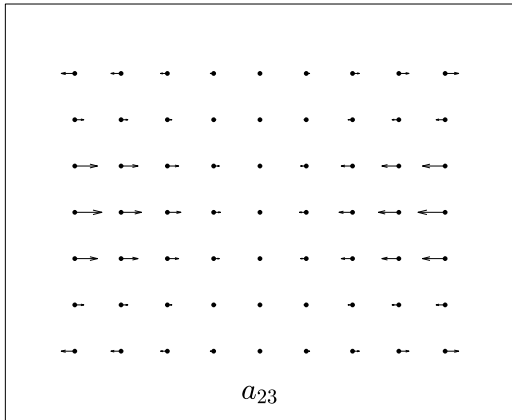
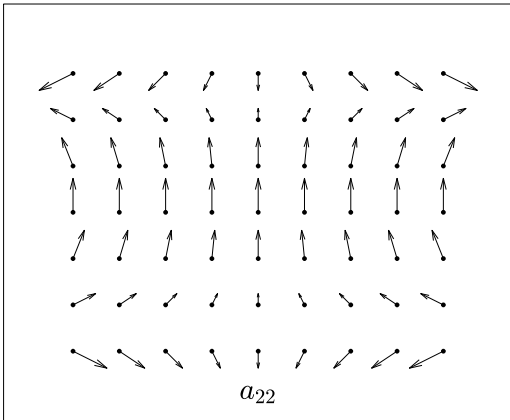
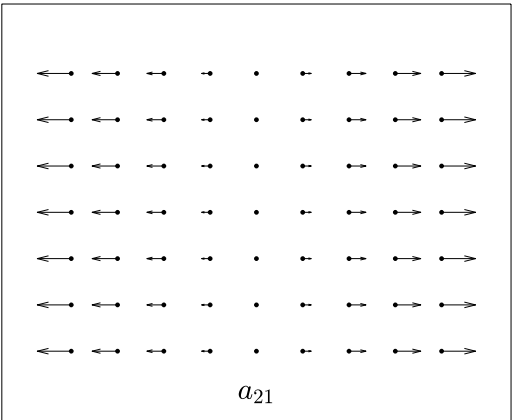
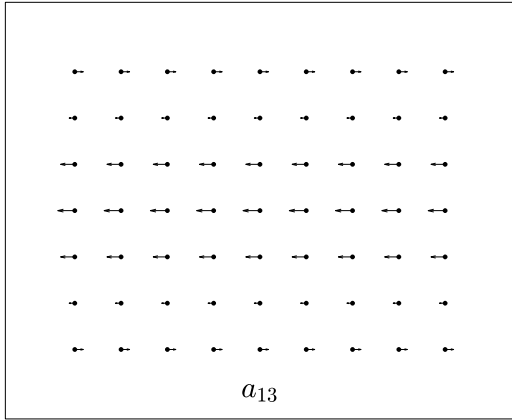
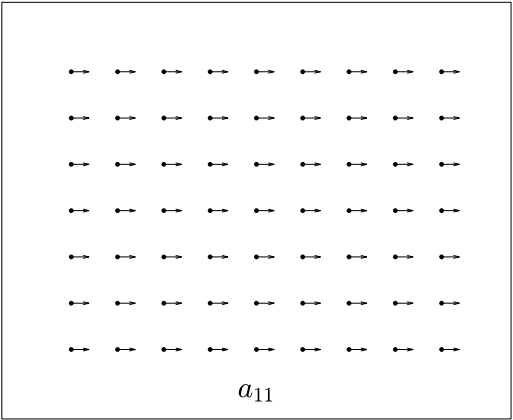


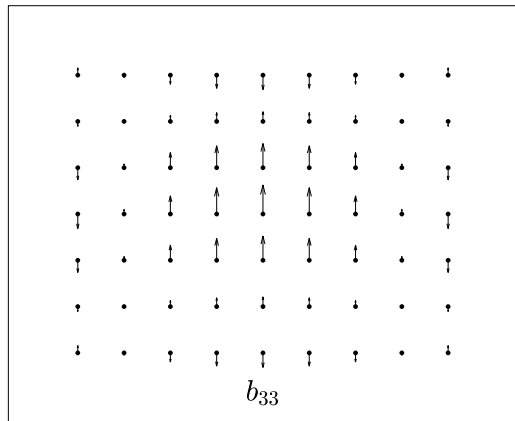
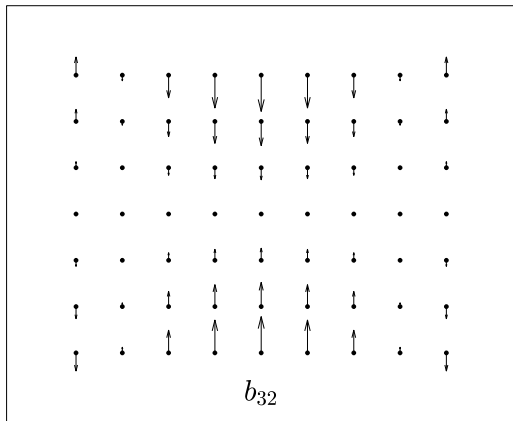
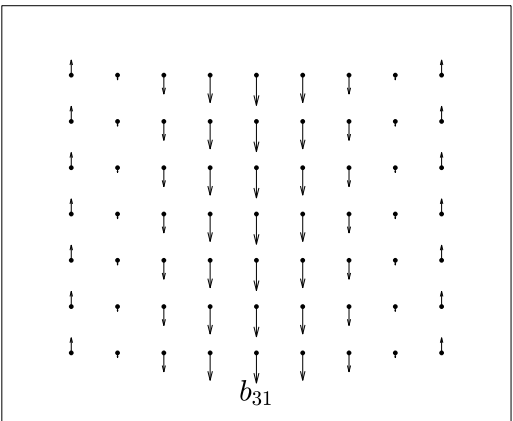
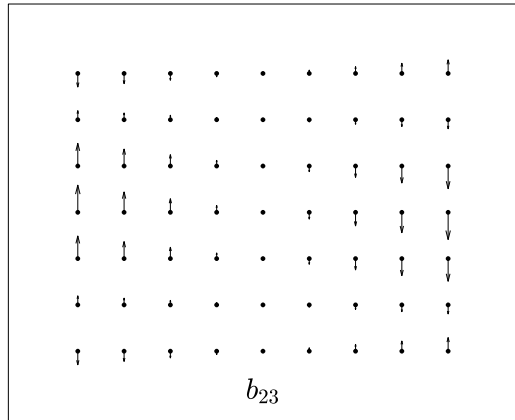
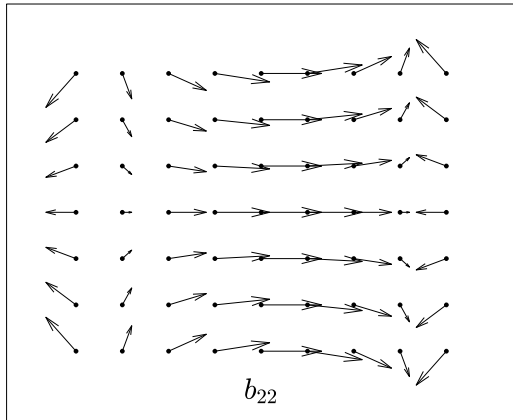
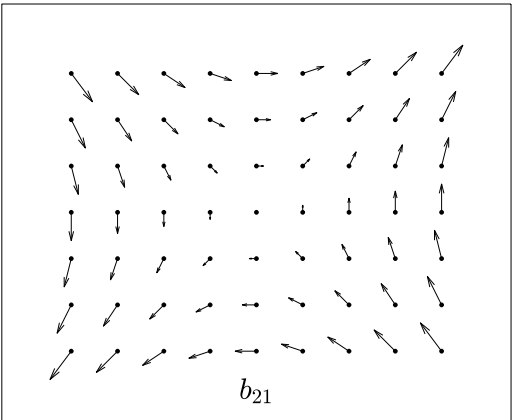
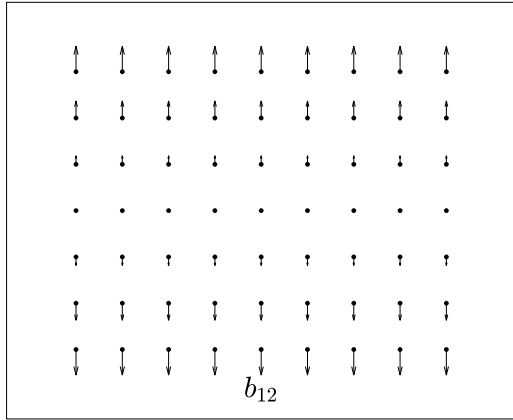
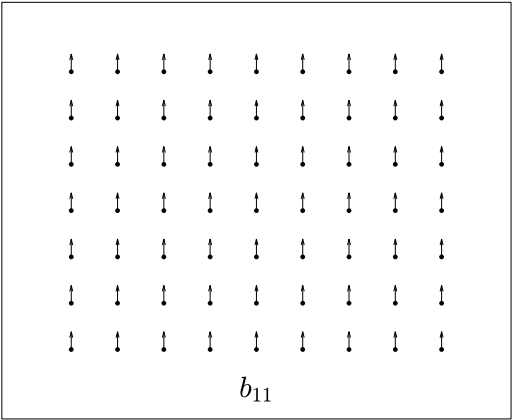












Brown	Orthogonal polynomials
$pp_x$	$a_{11}$
$pp_y$	$b_{11}$
$c + B_1$	$a_{21}$
$c$	$b_{12}$
$B_2$	$b_{12}$

Table 1.2. Correspondences between Brown parameters and Orthogonal polynomials ones

## 1.2 Other observation models

### 1.2.1 Normal camera

A re-written Equation 1.19 below models both IO (Interior Orientation) and EO (External orientation) of one camera  $A$  (embedded on the platform).

$$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = \xi_A \left( \pi \left( R_A^{WT} (P^W - T_A^W) \right) \right) \quad (1.25)$$

$P^W$  is the considered point on the ground,  $R_A^W$  is the rotation matrix from the camera  $A$  to the *world* frame, and  $T_A^W$  is the position of the camera perspective center.  $\xi_A$  models the Internal Orientation of camera  $A$ .

### 1.2.2 Camera rig

If several cameras are mounted jointly (i.e rigidly fixed with respect to each other's), the common method is to consider the first camera ( $A$ ) as the reference for the others. It means that all others sensors embedded on the same platform are georeferenced with respect to this first camera. For example, a second camera  $B$  is referenced with respect to the reference camera  $A$  by its lever-arm  $T_A^B$  (vector joining the perspective center of camera  $A$  to the perspective center of camera  $B$  in the frame of camera  $A$ ) and its boresight matrix  $R_A^B$  from camera  $A$  frame to camera  $B$  frame).

The lever-arm and boresight matrix definition 1.26 must be inputted in the camera  $B$  projection

model 1.27 to create the camera model in the frame of the reference camera  $A$  1.28.

$$\begin{cases} T_B &= T_A + R_A^W T_A^B \\ R_B^W &= R_A^W R_B^A \end{cases} \quad (1.26)$$

$$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = \xi_B \left( \pi \left( R_B^{W^T} (P^W - T_B^W) \right) \right) \quad (1.27)$$

$$\begin{bmatrix} \ell_x \\ \ell_y \end{bmatrix} = \xi_B \left( \pi \left( R_A^B \left( R_A^{W^T} (P^W - T_A^W) - T_A^B \right) \right) \right) \quad (1.28)$$

The method presented in equation 1.26 permits to convert reference frame from one sensor to the other. Hence it can be applied to all other sensors embedded on the same platform described in 1.2.3, 1.2.4, 1.2.7, etc.

### 1.2.3 LIDAR

Contrarily to image coordinates that represent a direction only, LIDAR measures both the direction of the point (in sensor frame), and the distance. Thus, the observation vector  $\ell_L$  of a point<sup>7</sup> measured by the LIDAR contains the three coordinates in the sensor frame. Note that since the frequency of LIDAR acquisition is substantially higher than the camera acquisition rate, the timestamp of the LIDAR point acquisition does not correspond to any photo taken by the camera. Methods to express the position  $T_L^W$  and the orientation  $R_L^W$  of the LIDAR at the time of measurement acquisition will be discuss in Chapter 4. The relation between LIDAR observation after application of sensor IO reads:

$$P^W = T_L^W + R_L^W \ell_L \quad (1.29)$$

### 1.2.4 Spherical photos

A spherical image is an image which represents all directions from the acquisition point of view [164]. Usually, these images are visualized as if the observer was inside a bubble on which

---

<sup>7</sup>Center or pear of returned energy spread over a certain region called foot-print (usually few *cm*)



the image is projected. The most famous spherical image bank is Google Street-view. Such images could be acquired by a panoptic camera (set of multiple cameras rigidly fixed one to the others). If the raw images are available, and if the Internal Orientation of each camera, as well as the lever-arm and boresight matrix from each camera to a chosen master camera are known (or could be calibrated), Equation 1.28 could be applied.

However, the raw images from panoptic cameras are often transformed to spherical images for practical reason e.g., simplification of photogrammetric problem, diffusion to end-users. In particular, the images from Google Street-view are only available as spherical images.

The observed longitude  $\lambda$  and latitude  $\varphi$  of a point on the photo could be represented on the unit-sphere.

$$\ell = \begin{bmatrix} \cos(\lambda) \cos(\varphi) \\ \sin(\lambda) \cos(\varphi) \\ \sin(\varphi) \end{bmatrix} \quad (1.30)$$

The projection function  $\pi$  defined by equation 1.2 was adapted to classical cameras since it projects a 3D point on a theoretic plane whose distance from the perspective center is unitary. modeling spherical cameras require re-defining a projection function  $\tilde{\pi}_s$  to project 3D point on a theoretic unit-sphere ( $\|\bullet\|$  is the Euclidian norm for 3d vectors).

$$\tilde{\pi}_s : \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \rightarrow \left\| \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \right\|^{-1} \cdot \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \quad (1.31)$$

The spherical camera model whose perspective center is  $T$  and orientation is  $R$  project the 3D point  $P^W$  on the observation point on the unit-sphere. This formulation permits to handle easily spherical images in a rigorous way since all points will be considered equally (with the same weight, see 1.3) independently to their latitude  $\varphi$ .

$$\ell = \tilde{\pi} (R^T (P^W - T)) \quad (1.32)$$

### 1.2.5 GCP

Ground Control Points (GCPs) are points on the object to be surveyed (e.g. the field, or the ground) which are signalized to be well visible and recognizable on the photos and which have been measured with external method (e.g. theodolite, terrestrial GNSS, etc.). These points could be considered as *special* tie-points since they are seen (and recognized) in the images taken by the cameras. The observation model of a direct measurement of the GCP is trivial since the observation  $\ell_{GCP}$  of the point must be equal (up to instrument imprecision) to the coordinate of the point in *world* frame  $P_{GCP}^W$ .

$$\ell_{GCP} = P_{GCP}^W \quad (1.33)$$

Check Points (CPs) are well recognizable points on the objects, however, Contrarily to GCPs, the direct measurement in *world* frame is NOT used as input to photogrammetric computation, but ONLY at the end of the photogrammetric process to compare their position measured on the field  $\ell_{CP}$  to the position computed<sup>8</sup> via photogrammetry  $P_{CP}^W$ .

$$\ell_{CP} \stackrel{?}{=} P_{CP}^W \quad (1.34)$$

The difference between  $\ell_{CP}$  and  $P_{CP}^W$  is called the Check Point missclosure (the appellation Check Point residual could also be found in the literature). It is an indicator of the local accuracy assessed at these specific points.

### 1.2.6 GNSS antenna

The embedded GNSS receiver measures the absolute position of the antenna phase center thanks to measurements from GNSS satellites signals. Complete description of GNSS measurement system can be found in [20], [19] and [77]. The given position of the antenna phase

---

<sup>8</sup>There are two approaches to compute the Check Point position via photogrammetry. The first approach is to consider it in the computation as a Tie-Point. However, the signalization of these special tie-points (target, chess-board) often permits to reach a better precision than the other tie-points. The use of Check Point to measure the accuracy leads to a modification of this accuracy (a better accuracy). Therefore, a second approach (used in this thesis) consists in removing the Check Point image observation from Bundle Adjustment (to prevent them helping too much 3D reconstruction), compute the camera IO and EO with the other measurements, and compute the Check Point positions only at the end through light-ray intersection.

center is related to the reference camera perspective center thanks to the lever-arm  $T_A^{GNSS}$ .

$$\ell_{GNSS} = T + R T_A^{GNSS} \quad (1.35)$$

The benefit of GNSS measurements to close-range airborne photogrammetry were for instance studied in [145], [132] and [130]. The lever-arm could be determined either by calibration (considering it to be unknown during Bundle Adjustment), or by theodolites measurement with the method described in [130].

### 1.2.7 Orientation measurements

The use of GNSS/IMU observations permit to derive orientations as described in [145]. These orientations could be inputted in absolute or relative terms to best describe the behavior of the sensors and the measurement process.

#### Absolute orientation

IMU measurements permit to derive the trajectory of the IMU<sup>9</sup> sensor when coupled with GNSS or other sensors via Kalman filtering and smoothing. At the time of the pose, the given orientation of the IMU  $R_{IMU}^W$  is related to the orientation of the principal camera  $R_A^W$  by the boresight matrix  $R_A^{IMU}$ , as described and assessed in [131].

$$R_A^W = R_{IMU}^W R_A^{IMU} \quad (1.36)$$

#### Relative orientation

The use of low-cost small IMUs may lead to large systematic orientation biases. This leads to time-dependent errors in the trajectory. Figure 1.5 presents the difference in orientation between the trajectory computed from a consumer-grade IMU and a navigation grade IMU<sup>10</sup>, thus, the presented values could be considered as true-errors. The error at a given moment is highly correlated with the error just before or after, and the correlation between two error decreases with the time difference separating these.

<sup>9</sup>IMU position center may be used also as position observation in similar manner as Equation 1.35 while the advantage of smoothing GNSS positions is to follow better high dynamic

<sup>10</sup>The navigation grade IMU is a IXblue AIRINS described in [72] and the consumer grade IMU is a NavChip ISNC01 described in [31]. These sensors are used in the experimental part of Chapter 4.

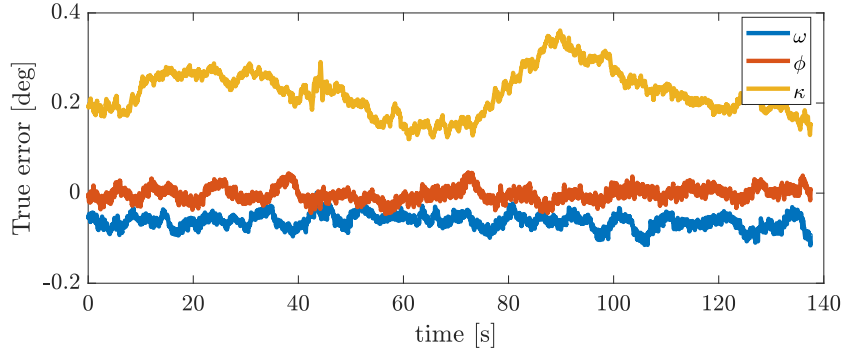


Figure 1.5. True error of a trajectory sample computed from a consumer-grade IMU

This motivates the computation of relative orientations between one pose (index 1) and the next one (index 2). Two relative orientations computations could be found in the literature. [14], [16] and [131] propose the relative orientations 1.37 while [95] proposes the relative 1.38.

$$\Delta = R_{IMU_1}^W R_{IMU_2}^W{}^T \quad (1.37)$$

$$\Delta = R_{IMU_1}^W{}^T R_{IMU_2}^W \quad (1.38)$$

Appart eliminating the same orientation bias between two camera poses captured few second appart, the asset of 1.37 relative orientation is to simplify the system calibration, i.e. the boresight-matrix do not need to be inputted nor need to be determined explicitly.

$$\Delta = R_{IMU_1}^W R_{IMU_2}^W{}^T = R_{A_1}^W \underbrace{R_A^{IMU^T} R_A^{IMU}}_{I_3} R_{A_2}^W{}^T \quad (1.39)$$

To compare the efficiency of the approaches described by equation 1.37 and 1.38, both approaches have been implemented. However, the comparison is unfair since 1.38 needs explicit computation of the boresight-matrix  $R_A^{IMU}$  whereas 1.37 do not need it. This is why, a second *mis-alignment* matrix have been introduced to permit to compute the same number of parameters using equation 1.37 or 1.38. This matrix was chosen to be the rotation matrix

between *world* frame and auxiliary navigation frame. In practice, it is a non-sense to consider this matrix to be unknown, but it permits to proceed to a more fair comparison between both approaches.

$$R_A^i = R_W^i R_{IMU}^W R_A^{IMU} \quad (1.40)$$

With this new *miss-alignment* matrix  $R_W^i$ , the relatives orientations 1.37 and 1.38 became respectively 1.41 and 1.42.

$$\Delta = R_{IMU_1}^W R_{IMU_2}^W{}^T = R_W^i{}^T R_{A_1}^i \underbrace{R_A^{IMU T} R_A^{IMU}}_{I_3} R_{A_2}^i{}^T R_W^i \quad (1.41)$$

$$\Delta = R_{IMU_1}^W{}^T R_{IMU_2}^W = R_A^{IMU} R_{A_1}^i{}^T \underbrace{R_W^i R_W^i{}^T}_{I_3} R_{A_2}^i R_A^{IMU T} \quad (1.42)$$

The comparison of the implementation of 1.41 (which need to compute  $R_W^i$  explicitly) with the implementation of 1.42 (which need to compute  $R_A^{IMU}$  explicitly) have been done on a corridor mapping scenario with 3 GCPs and 18 Check Points, without GNSS inputs. No significant performances difference have been observed (Check Point residuals given by the implementation using 1.41 are few percent smaller than the one given by the implementation using 1.42). This experiment motivate the choice for the approach represented by equation 1.37 since the simplification of the boresight-matrix  $R_A^{IMU}$  makes it simpler to implement.

### 1.2.8 Time-synchronisation between the sensors

The assembly of several sensors requires stable rigid mounting as well as common and stable time frame for all observations. Thus, the timestamp of the camera exposure needs to be recorded for each photo (in practice, an electric pulse is emitted in relation of the exposure that is time-stamped by navigation sensor such as GNSS receiver). However, some delay may exist between the timestamp (electric pulse arrival) and the actual time of the measurement. This delay could be considered as the sum of a constant time (deterministic) and a random noise (stochastic). The deterministic part of this delay could be either measured with high precision clock or estimated (i.e. considered as unknown) in the Bundle Adjustment [128].

### 1.2.9 Concatenation of observation and parameters

The set of all inputs observations described in this section (acquired by the sensors: camera, LIDAR, embedded IMU, embedded GNSS, ground GNSS) are concatenated together<sup>11</sup> in a single vector  $\ell$  called observation vector. The size of a single observation taken separately from the other is generally 2 or 3 (e.g. an image point observation  $[\ell_x; \ell_y]$  have length 2, and a GNSS positioning  $[\ell_x; \ell_y; \ell_z]$  have length 3), however, the size of the full vector of observation  $\ell$  is denoted  $n$  (number of observation) and could have an order of magnitude of  $10^6$ .

Similarly to the input observations, the unknown parameters that permit to compute such observations (i.e. the position of the tie-points  $P^W$ , position of the camera  $T^W$ , orientation of the camera  $R^W$ , lever-arms, boresight matrix, etc.) are concatenated in a single vector  $\mathbf{x}$  called parameter vector. The size of a single parameter taken separately from the other is generally 3 (e.g. a point position, a camera position or a lever-arm vector), however, the size of the full vector of parameters  $\mathbf{x}$  is denoted  $u$  (number of parameters) and is generally 2 or 3 times lower than the one of  $\ell$ . Formally, the rotation matrices cannot be concatenated into the  $\mathbf{x}$  vector. Section 1.5 provides the mathematical theory to handle the rotations properly in a Bundle Adjustment.

The observation models (equations from 1.25 to 1.33) for each measurements are themselves concatenated in a single function  $f: \mathbb{R}^u \rightarrow \mathbb{R}^n$ . Such a function takes as argument the parameters  $\mathbf{x}$ , and output a simulation of the observation  $\ell$ . More rigorously, the function  $f$  must be build such that if  $f$  is applied to the perfect theoretic parameters  $\check{\mathbf{x}}$ , it must output perfect theoretic measurements  $\check{\ell}$ .

$$f(\check{\mathbf{x}}) = \check{\ell} \tag{1.43}$$

Adjustment algorithms usually starts from an approximated value of the parameters denoted  $\overset{\circ}{\mathbf{x}}$ , which leads to the approximated values of the observations thanks to the function  $f$ .

$$f(\overset{\circ}{\mathbf{x}}) = \overset{\circ}{\ell} \tag{1.44}$$

The goal of the adjustment will be to choose the so called *compensated* parameters  $\hat{\mathbf{x}}$  such that

---

<sup>11</sup>The notation  $\ell$  and  $\mathbf{x}$  refers to large size (order of magnitude of  $10^6$ ) vectors issued from concatenation of low size vectors  $\ell$  (usually 2 or 3 elements).

the corresponding *compensated* observations  $\hat{\ell}$  are *as close as possible* to  $\check{\ell}$ .

$$\mathbf{f}(\hat{\mathbf{x}}) = \hat{\ell} \tag{1.45}$$

The goal of Section 1.3 is to define rigorously the notion of *as close as possible*, while that of Section 1.4 deals with the computation of  $\hat{\mathbf{x}}$ . There, we will need to compute the Jacobian matrix  $A$  of  $\mathbf{f}$  with respect to  $\mathbf{x}$ .

$$A = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \tag{1.46}$$

### 1.2.10 Covariance matrix of the parameters

Once the values of parameters  $\hat{\mathbf{x}}$  is determined, one has to determine the variance of these parameters. If the observations  $\ell$  follow a multivariate normal distribution, their precision could be fully represented by their covariance matrix  $\Sigma_{\ell\ell}$ . Error propagation permits to compute the covariance matrix of the compensated parameters  $\hat{\mathbf{x}}$ .

$$\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = (A^T \Sigma_{\ell\ell}^{-1} A)^{-1} \tag{1.47}$$

The computation of the full inverse of  $A^T \Sigma_{\ell\ell}^{-1} A$  can be time and memory consuming. Moreover, the full  $\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$  is not necessary useful: usually, only some block of the full matrix are useful for practical applications (Figure 1.6). Thus, [136] and [84] propose a method to compute only the needed parts  $\Sigma_{sub}$  of  $\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$  without compute the full inverse.

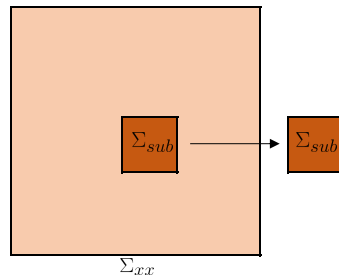


Figure 1.6. Computation of a sub-block of  $\Sigma_{xx}$

### 1.3 Expressing optimization problem

Sections 1.1 and 1.2 describe the so called observation model. They relate the parameters describing the model (mapped points on the ground coordinates, position and orientation of the camera, interior orientation of the camera, lever-arm and boresight-matrix, etc.) to the theoretic values of the observations acquired by the sensors, which will be considered as input-data in an adjustment software. An adjustment software solves an inverse-problem i.e. tries to recover the parameters values  $\mathbf{x}$  (concatenation of all necessary parameters) from the observation vector  $\ell$  (concatenation of all sensors measurements). However, due to inherent imperfection of the sensors and measuring process, the observations are not perfect. We define the *effective* observations  $\ell$  as the direct output of the sensors, and the *adjusted* observation  $\hat{\ell}$  as the best possible choice of modified observation that fit the model described in Sections 1.1 and 1.2. The residuals are defined as the difference between the *effective* observations and the *adjusted* observation: equation 1.48. Another appellation for  $v$  is OMC (Observed Minus Calculated).

$$\mathbf{v} = \ell - \hat{\ell} \quad (1.48)$$

The above mentioned *best possible choice* of parameters corresponds to reducing the residuals  $v_i$  to be as small as their probability let them to be. More rigorously, it translates in finding the parameters  $\hat{\mathbf{x}}$  maximizing the probability density of  $v_i$  depending of  $\mathbf{x}$ .

$$\begin{aligned} \hat{\mathbf{x}} &= \underset{\mathbf{x}}{\operatorname{argmax}} && p(\mathbf{v}|\mathbf{x}) \\ &\text{subject to} && \mathbf{v} = \ell - \mathbf{f}(\mathbf{x}) \end{aligned} \quad (1.49)$$

The full probability density of the whole set of  $v_i$  is obtained by computing the probability of the intersection of the set of assertion  $v_i \approx 0$ , where  $\approx$  could be defined rigorously as *equal, up to a given tolerance*. The intersection of set translates as product for probability.

$$P\left(\bigcap_i v_i \approx 0\right) = \prod_i P(v_i \approx 0) \quad (1.50)$$

By choosing different tolerances for the definition of  $\approx$ , the equation for probability 1.50 could



### 1.3. Expressing optimization problem

be translated to density of probability 1.51.

$$p\left(\bigcap_i v_i | \mathbf{x}\right) = \prod_i p(v_i | \mathbf{x}) \quad (1.51)$$

The most commonly used probability distribution is the normal distribution 1.52, represented by the 'least square' curve of Figure 1.7. It follows the hypothesis that the error is the sum of infinitesimal errors that could occur either positively or negatively. The distribution of an unbiased residual  $v_i$  is characterized by its standard deviation  $\sigma_i$  expressing the dispersion of the values.

$$p(v_i) = \frac{1}{\sqrt{2\pi} \sigma_i} e^{-v_i^2/2\sigma_i^2} \quad (1.52)$$

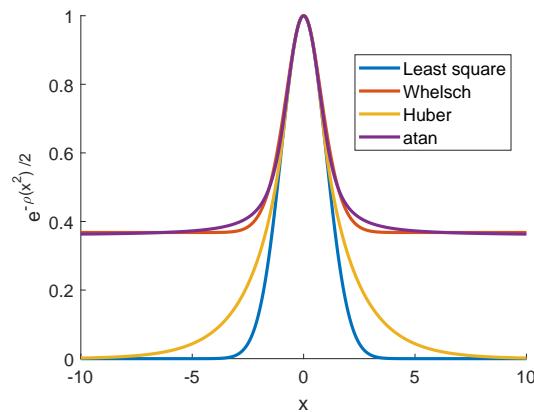


Figure 1.7. Pseudo-distributions constructed by the  $\rho$  function

Equation 1.51 could be re-written with normal distribution.

$$p(\mathbf{v}) = p\left(\bigcap_i v_i | \mathbf{x}\right) = \prod_i \frac{1}{\sqrt{2\pi} \sigma_i} e^{-v_i^2/2\sigma_i^2} = \frac{1}{(2\pi)^{\frac{n}{2}} \prod_i \sigma_i} \exp\left(-\frac{1}{2} \sum_i \frac{v_i^2}{\sigma_i^2}\right) \quad (1.53)$$

For each observation, its weight  $p_i$  is defined as the inverse of its squared sigma<sup>12</sup>.

$$p_i = \frac{1}{\sigma_i^2} \quad (1.54)$$

Equation 1.53 could be simplified by introducing the cost function  $\Omega$ .

$$\Omega = \sum_i p_i v_i^2 \quad (1.55)$$

The goal of the adjustment is to maximize the probability density 1.56 that the chosen parameters satisfies the given observations. This could be achieved by minimizing the cost function  $\Omega$ . Indeed, 1.53 is a decreasing function which depends on  $\Omega$  only (all other terms are constants i.e. have *a priori* known values).

$$p(\mathbf{v}) = \frac{1}{(2\pi)^{\frac{n}{2}} \prod_i \sigma_i} \exp\left(-\frac{1}{2}\Omega\right) \quad (1.56)$$

The above defined cost function  $\Omega$  is known as the uncorrelated weighted least-square criterion, and its minimisation is known as uncorrelated weighted least-square. It is the most simple of the ones described on Table 1.3 and yet the most commonly used. However, such definition of the objective function  $\Omega$  is not robust: a blunder on one observation will lead to a high squared residual in  $\Omega$ . A minimization algorithm whose aim would be to minimize  $\Omega$  will tends to underestimate the residuals associated to the blunders i.e. to not discard the blunders.

The goal of employing robust estimators is to take into account possible blunders and to minimize their impact on parameters. This could be achieved by modifying the hypothesis of normal distribution 1.52 followed by the residuals  $v_i$ . One method to take account blunders in normal distribution is to consider the possibility of a blunder with a probability  $\epsilon$ . The probability that the residual of an observation follows a normal distribution is thus  $1 - \epsilon$ , as in

---

<sup>12</sup>For numerical and practical reasons, it is also recommended to scale all the weight with a common factor, as described in [104].

equation 1.57.

$$p(v_i) = (1 - \epsilon_i) \frac{1}{\sqrt{2\pi} \sigma_i} e^{-v_i^2/2\sigma_i^2} + \epsilon_i \mathfrak{U}(v_i) \quad (1.57)$$

$\epsilon_i$  is the probability that the observation  $i$  whose residual is  $v_i$  is a blunder.  $\mathfrak{U}$  is the uniform distribution followed by the residual in case of blunder (the range of this uniform distribution should be chosen accordingly to the context of the measurement, e.g. the maximum size of measurement error that could occur).

$$\mathfrak{U}(v_i) = \begin{cases} \frac{1}{2 v_{max}} & \text{if } v_i \in [-v_{max}, v_{max}] \\ 0 & \text{else} \end{cases} \quad (1.58)$$

The  $\rho$  function is defined such that the probability distribution could be expressed as equation 1.59.

$$p(v_i) = \frac{1}{\sqrt{2\pi} \sigma_i} e^{-\frac{1}{2} \rho_i(v_i^2/\sigma_i^2)} \quad (1.59)$$

Equation 1.57 leads to the  $\rho$  function defined as 1.60. In the particular case of a normal distribution ( $\epsilon_i = 0$ ), this function  $\rho_i$  is the identity.

$$\rho_i : x \mapsto -2 \log \left( (1 - \epsilon_i) e^{-\frac{1}{2}x} + \epsilon_i \sqrt{2\pi} \sigma_i \mathfrak{U}(v_i) \right) \quad (1.60)$$

Applying the same reasoning as 1.53 leads to a modified version of the cost function  $\Omega$ . Again, an identity  $\rho$  function leads to the classical uncorrelated weighted least square cost function.

$$\Omega = \sum_i \rho_i(p_i v_i^2) \quad (1.61)$$

Such a  $\rho$  function defines the *Whelsch* minimization function. The definition of other  $\rho$  functions (e.g. Huber, or atan as on Figure 1.8) generates other robust estimators. The  $\rho$

function must be increasing (usually concave) function such that  $\rho(0) = 0$ . For the sake of consistency, the presented  $\rho$  function are also normalized<sup>13</sup> such that  $\rho(x) \approx_0 x$ .

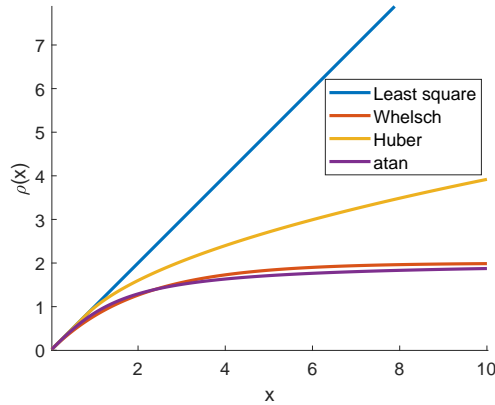


Figure 1.8. Typical examples for the  $\rho$  function

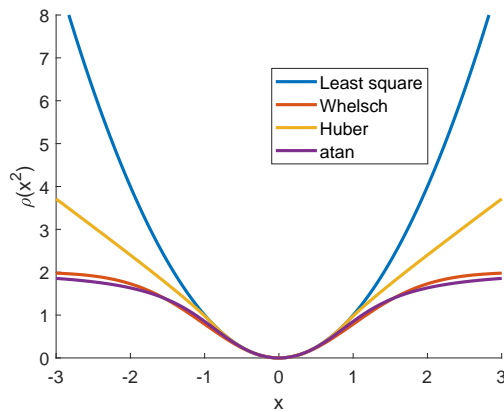


Figure 1.9.  $\rho(v^2)$

The increase of Whelsch, Huber and atan  $\rho$  function is less important than the identity (representing Least square) cost function, leading to a robust estimator. Figure 1.9 shows a single term inputted in the sum the function inputted in the global cost function 1.61. Quadratic terms (constituting least square estimators) give an over-rated importance to blunders, whereas robust estimators (e.g. Whelsch, Huber, atan) reduce the weight of blunders.

The introduction of the  $\rho$  function permits to robustify least square method with respect to

<sup>13</sup>In particular, the function represented on Figure 1.7, 1.8 and 1.9 is not the  $\rho$  function defined by equation 1.60 but  $\tilde{\rho}(x) = \alpha (\rho(x) - \rho(0))$  where  $\alpha$  have been chosen such that  $\tilde{\rho}(x) \approx_0 x$ .

### 1.3. Expressing optimization problem

$\Omega$	non robust	robust
non correlated	$\sum_i p_i v_i^2$	$\sum_i \rho_i (p_i v_i^2)$
correlated	$\mathbf{v}^T P \mathbf{v}$	$\sum_i \rho_i (\mathbf{v}_i^T P_i \mathbf{v}_i)$

Table 1.3. Cost function definition

Blunders (second column of table 1.3). However, this do not take into account for possible correlations between the inputs.

The following input observation vector  $\ell$  is described by its covariance matrix  $\Sigma$  (the notion of *is described by* is denoted by the symbol  $\sim$ ). The diagonal element of this matrix are the variances of the elements of  $\ell$ . This matrix must be symmetric and semi-definite to be a covariance matrix, i.e. it satisfies  $\Sigma = \Sigma^T$ , and all its eigenvalues are strictly positives. Thus, it define a scalar product on  $\mathbb{R}^n$ .

$$\begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \\ \vdots \\ \ell_n \end{bmatrix} \sim \underbrace{\begin{bmatrix} \sigma_1^2 & \sigma_{21}^2 & \sigma_{31}^2 & \cdots & \sigma_{n1}^2 \\ \sigma_{21}^2 & \sigma_2^2 & \sigma_{32}^2 & \cdots & \sigma_{n2}^2 \\ \sigma_{31}^2 & \sigma_{32}^2 & \sigma_3^2 & \cdots & \sigma_{n3}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \cdots & \sigma_n^2 \end{bmatrix}}_{\Sigma} \quad (1.62)$$

The coefficient of correlation between two terms could be computed as  $\frac{\sigma_{ij}}{\sigma_i \sigma_j}$ . It could be shown that if the matrix  $\Sigma$  is symmetric positive definite, the correlation coefficients belongs to the interval  $[-1, 1]$ . Note that the reciprocal is not true: not all set of coefficients between -1 and 1 permits to build a proper positive definite covariance matrix. The correlations between

inputs leads to the following modification to the normal density function 1.52.

$$p(\mathbf{v}) = \frac{1}{\sqrt{2\pi}^n \sqrt{|\Sigma|}} e^{-\frac{1}{2} \mathbf{v}^T \Sigma^{-1} \mathbf{v}} \quad (1.63)$$

Similarly to equation 1.54, the weight matrix  $P$  is defined as the inverse of the covariance matrix  $\Sigma$ .

$$P = \Sigma^{-1} \quad (1.64)$$

Maximising the probability density 1.63 leads to minimising the following cost function accounting for the possible correlations between the inputs:  $\Omega = \mathbf{v}^T P \mathbf{v}$  (see Table 1.3).

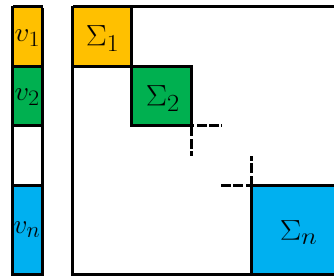


Figure 1.10. Block covariance matrix

This last cost function considers for correlation between observations but is not robust. There are several ways to robustify this expression [33]. The one presented here is adapted if the observation vector could be sliced into several independent sub-vectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ . The covariance matrix of such observation vector is thus block-diagonal (Figure 1.10). If there is a blunder on a vector, it does not have implication on possible blunders on another vector. However, a blunder on a component on one vector leads to a very high probability that every component of this same vector is wrong as well. These hypothesis leads to the following cost function.

$$\Omega = \sum_i \rho_i(\mathbf{v}_i^T P_i \mathbf{v}_i) \quad (1.65)$$

Table 1.3 summarizes the proposed cost functions for robust and non-robust estimation of correlated and non-correlated inputs. The cost-function of the right column are a generalization of the one on the left column, and the cost function of the second line is a generalization of the one of the first line.

- $\sum_i p_i v_i^2$  is a particular case of  $\sum_i \rho_i(p_i v_i^2)$  when the  $\rho$  function is taken as identity, i.e. when considering independent normal distribution.
- $\sum_i p_i v_i^2$  is a particular case of  $\mathbf{v}^T P \mathbf{v}$  when the input variables are independent, or decorrelated. Their covariance matrix  $\Sigma$  is diagonal, and thus the weight matrix  $P$  is diagonal as well.
- $\sum_i \rho_i(p_i v_i^2)$  is a particular case of  $\sum_i \rho_i(\mathbf{v}_i^T P_i \mathbf{v}_i)$  when the observation vector, and thus the residuals are sliced in vector with only one component each.
- At the opposite of the previous generalization,  $\mathbf{v}^T P \mathbf{v}$  is a particular case of  $\sum_i \rho_i(\mathbf{v}_i^T P_i \mathbf{v}_i)$  when the vector  $\mathbf{v}$  is sliced into only one vector. Thus, minimizing  $\mathbf{v}^T P \mathbf{v}$  is the same as minimizing  $\rho(\mathbf{v}^T P \mathbf{v})$  because  $\rho$  is an increasing function.

## 1.4 Solving the optimization problem

In the previous section, we have seen how to express an over-determined problem as an optimization problem. In this section, we will present different methods to solve such problem. For the sake of illustration, all presented methods will be applied on a very simple geodetic problem: the multilateration.

The principle of the multilateration has been known since antiquity (see the Hipparque map [70]). The rigorous theory to solve the problem were given simultaneously by Gauss and Legendre at the end of XVIII<sup>e</sup> century. The simplicity of this method makes it still up to date [47]. The goal of this chapter is to provide illustrative example of optimizations methods that could be applied to a complex problem such as photogrammetry.

The principle of multilateration is quite simple. The goal is to determine the 2D planimetric coordinates of an unknown point on a planimetric surface e.g. on the ground. This could be done by using points on this plane which 2d coordinates are well known, called survey mark, or beacon points. The distances between such known points and the unknown point permit to determine the coordinates of the unknown point. A distance measurement between a single beacon point and the unknown point is not sufficient to compute the position of the unknown point which could be at any point on the circle centered on the known beacon point and with a radius of the measured distance. In the general cases (not in a singular case), two distances

permit to determine the position of the unknown point, up to an ambiguity: since there is (generally) two intersection between two circles, the unknown point as an equal probability to be at any of these intersections. Usually, several (three or more) distances are measured to determine the coordinates of the unknown points. Figure 1.11 presents an example of multilateration problem.

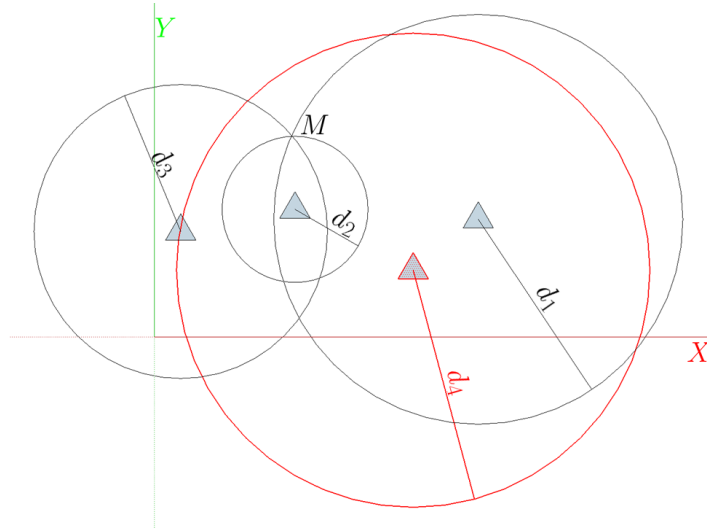


Figure 1.11. An example of Multilateration problem: the triangles are the beacon-points, the lines are the distance measurements, and the circle represent fix distances to the beacon points. The red measurement from the blue beacon-point is probably a blunder

If  $\hat{X}_M, \hat{Y}_M$  are the compensated 2D coordinates of the unknown point, the distance from the first beacon point could be calculated with Euclidian norm.

$$\hat{d}_1 = \sqrt{(\hat{X}_M - X_1)^2 + (\hat{Y}_M - Y_1)^2} \quad (1.66)$$

The observation of this distance is  $\ell_{d_1}$  or simply  $d_1$ .

$$d_1 - v_1 = \sqrt{(\hat{X}_M - X_1)^2 + (\hat{Y}_M - Y_1)^2} \quad (1.67)$$

The residual of such observation could thus be computed as the difference between the actual



measurement, and the computed one.

$$v_1 = d_1 - \sqrt{(\hat{X}_M - X_1)^2 + (\hat{Y}_M - Y_1)^2} \quad (1.68)$$

The goal is to minimize a cost function built from the residuals which will permit to share the error between the different residuals, i.e. do some compromises.

$$\Omega = \sum \rho(v_i^2) \quad (1.69)$$

The presented plots show the cost function when the  $\rho$  function is identity. The  $XY$  coordinates represent the possible position of the point, and the  $Z$  represents the cost  $\Omega$  associated to different  $XY$  couples of coordinate.

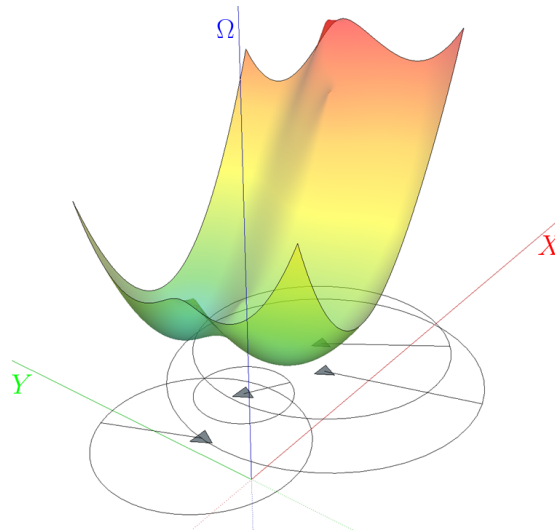


Figure 1.12. Traduction of the multilateration problem into an optimisation problem

The solution of the compensation is given by finding the  $XY$  coordinates corresponding to the lowest point of the curved surface. Finding the minimum by computing  $\Omega$  for all possible coordinate couple  $XY$  is nicknamed *Brute-Force* algorithm. In practice, it is intractable to compute  $\Omega$  for all possible coordinate couples  $XY$ , and thus, it is impossible to plot nor visualize the curved surface (in  $n$  dimensions). In order to find the minimum of the cost function  $\Omega$ , countless minimization algorithms have been proposed in the literature, for which the complexity (number of necessary operation) is neglectable compare to a brute-force

algorithm. The most common and powerful of these minimization algorithms are described in [96]. The methods, the principles and the limits are presented in a didactic way. This section will illustrate these methods on this simple example of the multilateration.

### 1.4.1 Gradient descent with small steps

The most intuitive method to find the minimum point of such a curve is the gradient descent with small steps. It is an iterative algorithm which starts from an initial value. At each iteration, the direction of the highest slope is determined by gradient calculation of the cost function  $\Omega$ . Then a small step is performed in the direction: the next value is given by the translation of the previous position by the direction of the highest slope. Intuitively, it corresponds to the way of a drop of water without inertia on the slope. Its trajectory will follow at each time the highest slope.

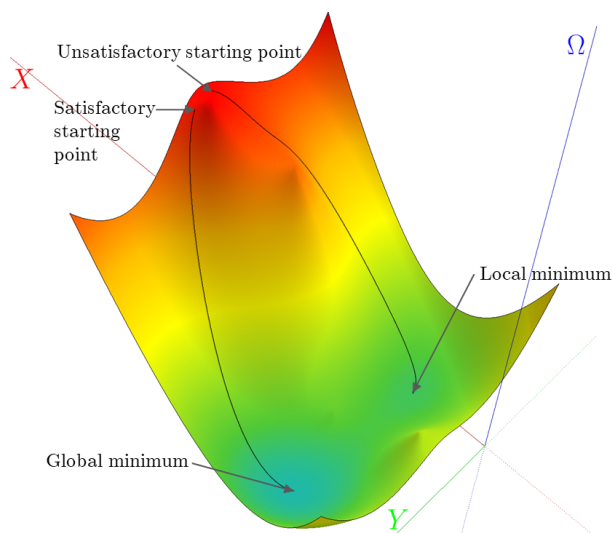


Figure 1.13. Gradient descent with small steps. The iterative process on the left of the figure converges to the global minima (in blue), whereas the iterative process on the right converges to a local minima

Note that the choice of the initial value is important for success of the algorithm to converge to the global minimum. Figure 1.13 presents two runs of the gradient-descent algorithm with small steps. The first one (left) starts from a high point and converges to the global minimum. The second one (right) starts from another point and converges to a local minima. This shows the importance of choosing an initial guess in the pull-in region of the global minimum i.e. close enough from the final solution.

### 1.4.2 Gradient descent with exact search

The first presented gradient descent need to compute the gradient of the function  $\omega$  at each iteration (typically several hundred/thousand are needed). The goal of the other methods is to reduce the number of iterations, and thus to improve the efficiency of each of them.

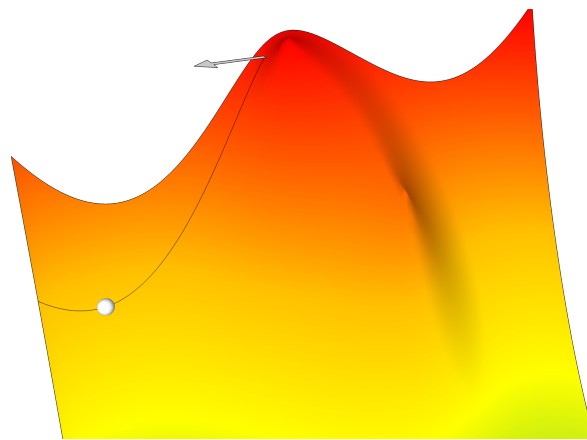


Figure 1.14. One step of the gradient descent with exact search

In the gradient descent method with exact search, each iteration begins with the gradient calculation of  $\Omega$  at the current point. This permits to determine the direction of the steepest slope (given by the arrow on Figure 1.14). The *exact search* refers to the minimum calculation of the minimum on the half line starting to the current point and in the direction of the steepest slope. This minimum search is performed by dichotomy algorithm. Thus, at each iteration, the gradient is computed once, and the  $\Omega$  function is computed several times in order to find the minimum along the search-line. The next iteration starts on this minimum.

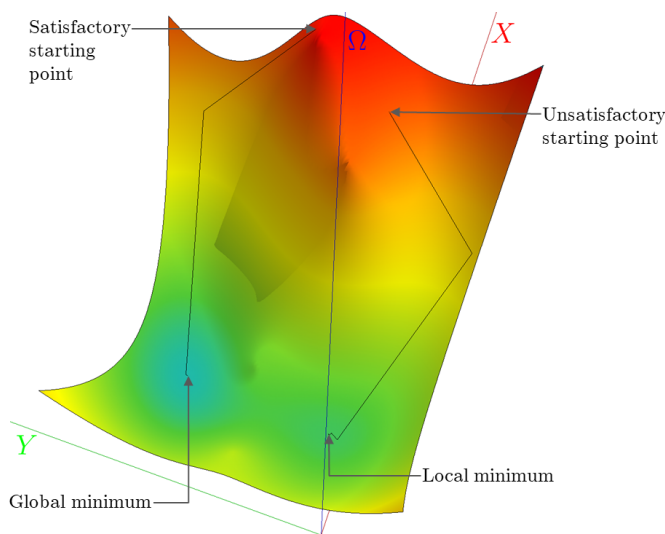


Figure 1.15. Gradient descent with an exact search. The iterative process on the left of the figure converges to the global minima (in blue), whereas the iterative process on the right converges to a local minima

Similarly to the previous method, the convergence success depends on the initial value. Figure 1.15 presents two runs of the Gradient descent with exact search algorithm. One (left) converges to the global minimum while the other (right) converges to a local minimum. The number of iterations required is lower than the previous method (around 10), but the time needed for each iteration is higher (around  $10\times$  more).

### 1.4.3 Gauss-Newton Algorithm

The gradient descent with exact search requires less iteration than the gradient descent with small steps. However, each iteration needs several computations of the  $\Omega$  function to find the minimum along the studied line. The goal of Gauss-Newton algorithm is to perform fast and efficient iterations to reach the minimum of the function in a small number of iterations, in a limited time.

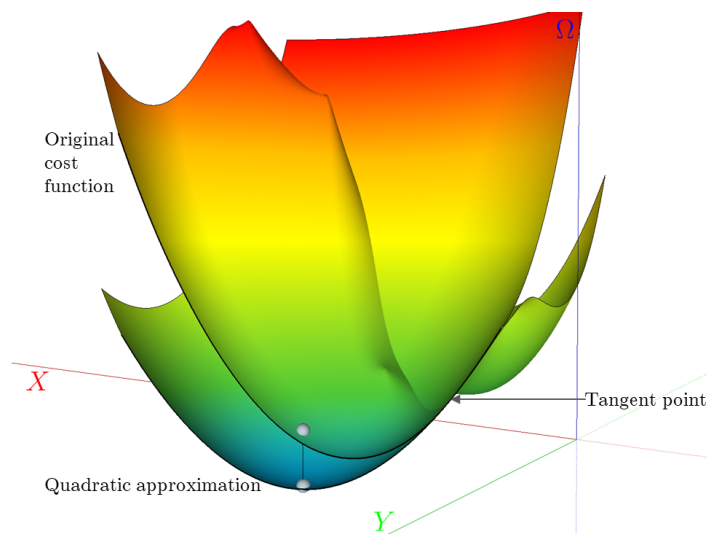


Figure 1.16. One step of the Gauss-Newton algorithm. The paraboloid is tangent on the initial guess (right part of the picture). The minimum of this paraboloid is represented by a little ball (bottom of the picture). The 'planimetric' position of this ball will be used as start for the next iteration (second ball above the first one)

Each iteration starts from an initial value. Not only the gradient but also the Hessian (second derivatives) matrix of  $\Omega$  are computed on the studied point. This permits to approximate the  $\Omega$  function by a quadratic function whose representation is a paraboloid (a bowl). Figure 1.16 represents the  $\Omega$  function, approximated by a paraboloid on the initial point: both curves are tangent to this point<sup>14</sup>. The minimum of the paraboloid (lowest ball on Figure 1.16) gives the result of the iteration.

<sup>14</sup>The approximation of the surface representing  $\Omega$  must be a paraboloid with a single minimum (an upright bowl). This is achieved if the Hessian of  $\Omega$  is symmetric and definite semi-positive. A default of this property could lead to the failure of the Gauss-Newton algorithm.

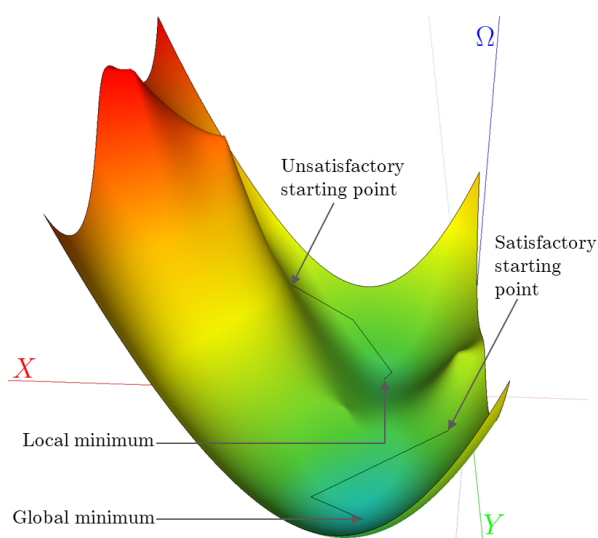


Figure 1.17. Gauss-Newton algorithm. The iterative process on the front of the figure converges to the local minima (in blue), whereas the iterative process on the back of the picture converges to a local minima

The number of iterations needed to reach the minimum of the  $\Omega$  function is even less than with the other methods. However, the choice of the initial point is even more important than with previous methods. Figure 1.17 show two solutions, one starting from an unsatisfactory initial guess and converging to a local minimum (in the back (upper part) of the figure), the second starting from a satisfactory initial value and leads to a convergence to the global minimum.

Gauss-Newton approach is very fast compared to gradient-descent algorithm. However, its pull-in region for success is usually smaller than the one of gradient-descent. [96] describes several SOTA algorithms (Levenberg Marquardt, trust region DogLeg) whose principle is to mix the two approaches to improve the robustness of Gauss-Newton algorithm to bad initial values while limiting the needed computation time.

Gradient-descent algorithm, Gauss-Newton algorithm, Levenberg Marquardt algorithm and trust region DogLeg algorithm needs computation of the derivatives (gradient and Hessian) of the cost-function  $\Omega$  with respect to the parameters. These derivatives needs the calculation of the derivatives of the observation models with respect to the parameters.

## 1.5 Lie-Group tutorial

### 1.5.1 Motivation

The goal of this tutorial is to handle the rotations correctly in Bundle Adjustment via the Lie-Groups theory. The intention is not to do a mathematical course about Lie-Groups (see [55] for the theoretic basis or [45] and [149] for an application to Bundle Adjustment), but to present a strict basis to build the theory from which Bundle Adjustment benefits. For the purposes of optimisation in Bundle Adjustment, derivatives of the observations equations with respect to the parameters must be computed. For example, the derivatives of collinearity equation 1.70 (see section 1.1) must be expressed with respect to  $P_{tp}^W$ ,  $P_c^W$  and  $R_W^c$ . The derivation with respect to the ground coordinate  $P_{tp}^W$  and  $P_c^W$  are direct, since they belong in an Euclidian space. However, the derivation with respect to  $R_W^c$  needs special mathematic concept.

$$\ell^c = \xi \left( \pi \left( R_W^c \left( P_{tp}^W - P_c^W \right) \right) \right) \quad (1.70)$$

At each iteration of the process, the minimization algorithm (e.g. Gauss-Newton algorithm, Levenberg-Marquardt algorithm or Dog-Leg algorithm) gives a vector of the increments of the parameters  $\delta x$ . Each component of  $\delta x$  represent a correction for a single parameter. For example, the correction of the 3D vector  $P_{tp}^W$  is the 3D incremental vector  $\delta P_{tp}^W$ . At each iteration, each approximated value  $\overset{\circ}{x}$  is updated to a value  $\hat{x}$  of the parameter that is closer to the final solution. The resulting value after an iteration will became the approximated value  $\overset{\circ}{x}$  of the next iteration. If  $x$  belongs to an Euclidian space (like  $P_{tp}^W$  and  $P_c^W$ ), the update step is given as 1.71.

$$\hat{P}_{tp}^W = \overset{\circ}{P}_{tp}^W + \delta P_{tp}^W \quad (1.71)$$

Thus, such set of 3D coordinate  $P_{tp}^W$  is said to be *parametrized* by  $P_{tp}^W$  because this  $P_{tp}^W$  is a sub-part of the long parameters vector  $\mathbf{x}$  described in 1.4, and used in the following equation 1.72. The equation 1.71 is a sub-part of equation 1.72.

$$\hat{\mathbf{x}} = \overset{\circ}{\mathbf{x}} + \delta \mathbf{x} \quad (1.72)$$

The most famous representation for rotation is the set of Euler angles:  $\{\omega, \varphi, \chi\}$ . The rotation is said to be parametrized by these three angles because these angles are included in the full parameter vector  $\mathbf{x}$ . The translation from Euler angle to rotation matrix is given by the

formulae 1.73 (other rotation sequences are possible to define, but the following reasoning hold for each of them).

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\omega) & -\sin(\omega) \\ 0 & \sin(\omega) & \cos(\omega) \end{bmatrix} \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \cos(\chi) & -\sin(\chi) & 0 \\ \sin(\chi) & \cos(\chi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1.73)$$

Naturally the update step for these Euler angles seems to be as follow.

$$\begin{cases} \hat{\omega} = \overset{\circ}{\omega} + \delta\omega \\ \hat{\phi} = \overset{\circ}{\phi} + \delta\phi \\ \hat{\chi} = \overset{\circ}{\chi} + \delta\chi \end{cases} \quad (1.74)$$

This parameterization has three main drawbacks. First, these three parameters are not *orthogonal*, leading to artificial correlations between these angles. This parameterization became even singular when  $\varphi \approx \frac{\pi}{2}$ . This phenomenon is well known as Gimbal-lock and leads to singularity since  $\omega$  became inseparable from  $\chi$ . Finally, this parameterization leads to highly complex derivatives for handling orientation observations.

Other parameterization do exist i.e. different sequences of Euler angles, quaternions, or the full rotation matrix parameterization. If the rotation is over-parametrized (i.e., if there is more than 3 parameters to describe a single rotation) constraints must be added to the parameters. For example, for quaternions, the norm must be unitary, and if the rotation is described by the nine elements of the matrix, 6 constraints must be added (the vectors composing the matrix must be unitary and their scalar product must be null. Moreover, the determinant of such matrix must be positive, and thus unitary).

An usual method consists in doing the derivation with respect to these parameters. Countless publications do a comparison of these rotation parameterization. However, up to the knowledge of the authors, only one ([5]) do a rigorous comparison of these behavior when used in the Bundle Adjustment. This publication do not show a significant difference of one parameterization with respect to the others.

Regardless the choice of rotational parameters, these do not belong to an Euclidian space. The Lie-Group theory permits to avoid the *parametrisation* of the rotation matrix: the parameter vector  $\mathbf{x}$  is no longer explicitly built since it is not possible to include a rotation matrix directly in the vector  $\mathbf{x}$  but the theory permits proceeding with defining rigorous derivatives. The



increment vector  $\delta \mathbf{x}$  is the only vector of equation 1.72 that will be explicitly built. The corresponding increments in  $\delta \mathbf{x}$  of the rotations will be the  $\omega$  values described in equation 1.75 and 1.76. When the Lie-Group theory is applied to Bundle Adjustment, the *update-step* 1.74 is substituted either by the one described by the formula 1.75 (called left multiplication update-step) or the one described by the formula 1.76 (called right multiplication update-step). The goal of this document is to motivate such a choice, and to describe how the Bundle Adjustment is modified accordingly.

$$\hat{R} = \exp([\omega]_{\times}) \overset{\circ}{R} \tag{1.75}$$

$$\hat{R} = \overset{\circ}{R} \exp([\omega]_{\times}) \tag{1.76}$$

### 1.5.2 The Lie-Group of 2D rotations

#### Lie-group

The concepts will be introduced in 2D and will be generalised in 3D in the section 1.5.4. The section 1.5.5 summarizes the essential reasoning via a comparison between 2D and 3D. First, we motivate the appellation *Lie-Group*, and particularly, the appellation *Group*.

Rotations in 2D could be represented via the set of complex numbers with a unitary norm. Such a complex number is represented by an arrow in Figure 1.18. The action of such complex number  $z$  on the object  $\clubsuit$  is the rotation around the origin by the angle that is the argument of this complex number. The result of such operation could be noted  $z \cdot \clubsuit$ .

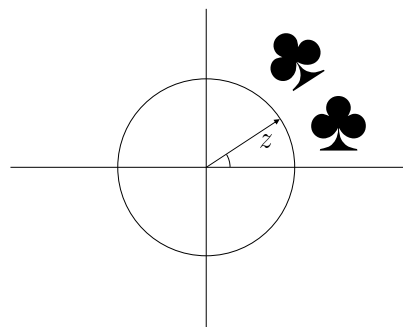


Figure 1.18. Rotation  $z$  applied to  $\clubsuit$

The set of 2D rotation is noted  $\mathbb{S}\mathbb{O}_2$ . In the following, a group structure will be associated to this set.

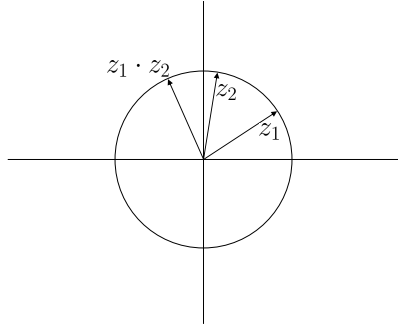


Figure 1.19. Composition of rotations

If  $z_1$  and  $z_2$  represent rotations, the rotation  $z_2$  could be applied to an object  $\clubsuit$ , then the rotation  $z_1$  could be applied to the result of this previous rotation. The two rotations  $z_1$  and  $z_2$  applied successively to  $\clubsuit$  could be described as a single rotation. Combining these two rotations  $z_1$  and  $z_2$  in one is called a *composition*:  $z_1 \circ z_2$ . This operation turns for complex numbers to multiplication:  $z_1 \circ z_2 = z_1 \cdot z_2$ . See Equation 1.77, and Figure 1.19.

$$z_1 \cdot (z_2 \cdot \clubsuit) = (z_1 \cdot z_2) \cdot \clubsuit \quad (1.77)$$

The rules of operation are called the *group law* of the set. The following equations show that the set  $\mathbb{S}\mathbb{O}_2$  with the operation *composition* follows the four axioms defining a *group*: *Closure*, *Associativity*, *Existence of Identity element* and *Inverse transformation for each element of the group*.

*Closure*:

$$\forall z_1, z_2 \in \mathbb{S}\mathbb{O}_2, z_1 \cdot z_2 \in \mathbb{S}\mathbb{O}_2$$

*Associativity*:

$$\forall z_1, z_2, z_3 \in \mathbb{S}\mathbb{O}_2, (z_1 \cdot z_2) \cdot z_3 = z_1 \cdot (z_2 \cdot z_3)$$

*Existence of Identity element*: for the set  $\mathbb{S}\mathbb{O}_2$ : 1.

$$\forall z \in \mathbb{S}\mathbb{O}_2, 1 \cdot z = z \cdot 1 = z$$

Each element of the group is invertible

$$\forall z \in \mathbb{S}\mathbb{O}_2, z \cdot \bar{z} = 1$$

Note that in the 2D case, the composition is also *Commutative* (This property will not hold in the 3D case).

*Commutativity* (only in 2D)

$$\forall z_1, z_2 \in \mathbb{S}\mathbb{O}_2, z_1 \cdot z_2 = z_2 \cdot z_1$$

These axioms permit to apply rotations to objects, but not to compute the derivatives. In other terms, they could permit to be used in the collinearity equation 1.70, but not to compute any derivatives of such equation. For a given rotation represented by  $z$ , an Euclidean space, tangent to the rotation group could be build. This space is called the *tangent space* of the group at the point  $z$ . In the case of  $\mathbb{S}\mathbb{O}_2$ , this tangent space is a line in the 2D space tangent to the unitary circle. The following derivation permits to reach this property.

The differentiation  $d(\bullet)$  of the fourth axiom of a group:  $z \cdot \bar{z} = 1$  leads to the following property  $d(z\bar{z}) = dz \bar{z} + z d\bar{z} = 0$ .

$$dz \bar{z} = -\overline{dz \bar{z}} \tag{1.78}$$

The complex number  $dz \bar{z}$  is equal to the opposite of it's conjugate. Its real part is thus null, and therefore, it is a purely imaginary number.

$$\exists d\theta \in \mathbb{R}, d\theta i = dz \bar{z} \tag{1.79}$$

Finally, it leads to Equation 1.80.

$$d\theta i z = dz \tag{1.80}$$

The differential of  $z$  is  $dz$ . Figure 1.20 presents  $z$  and  $dz$ . Since  $z$  lies on a circle, its differential is given by a vector directed by the direction of the tangent of the circle at  $z$  (Figure 1.20). This direction is perpendicular to  $z$ , which is  $i z$ . The norm is  $d\theta$ .

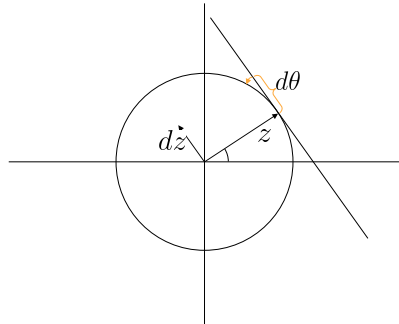


Figure 1.20. Tangent space of  $\mathbb{S}\mathbb{O}_2$

### Differentiation in the scope of Bundle Adjustment

From Equation 1.80 could be computed the derivative of the 2D rotation represented by  $z$ .

$$\frac{dz}{d\theta} = i z \tag{1.81}$$

Note that this equation gives a very straightforward way to show that the derivative of the function  $\sin$  is the function  $\cos$ , and that the derivative of  $\cos$  is  $-\sin$ .

Since a rotation is meaningless without an object to rotate (such as  $\clubsuit$ ), we provide the derivative of this rotated object.

$$\frac{d(z\clubsuit)}{d\theta} = i z \clubsuit \tag{1.82}$$

For the particular case of 2D rotation, it was possible to compute the derivative of  $z$  independently from the rotated object  $\clubsuit$ . The section 1.5.4 will show that in the scope of 3D rotation, it is not longer possible to differentiate the rotation independently from the object rotated by this rotation.

### Update-step in the scope of Bundle Adjustment

Equation (1.80) which describes the tangent space of 2D rotations could be seen as a differential equation whose solution is given by (1.83) were the exponential function is defined as  $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ . This series converges to a function that is equal to its own derivative, with a

value of 1 at 0.

$$\tilde{z} = z e^{i\theta} \tag{1.83}$$

In this equation,  $z$  is constant, and is the tangent point.  $\tilde{z}$  is the solution of the differential equation, function of  $\theta$ . The constant was chosen such that  $\tilde{z}$  coincides with  $z$  when  $\theta = 0$ . The interpretation of the meaning of  $\tilde{z}$  will be enlightened by the study of the properties of the exponential of a Purely Imaginary Number  $e^{i\theta}$ .

- $e^0 = 1$
- $\forall \theta \in \mathbb{R}, |e^{i\theta}| = 1$
- $\forall \theta_1, \theta_2 \in \mathbb{R}, e^{i(\theta_1 + \theta_2)} = e^{i\theta_1} \cdot e^{i\theta_2}$
- $\theta \mapsto e^{i\theta}$  is  $2\pi$  periodic.

These properties are considered as basics properties, and leads to the following important property:  $e^{i\theta}$  is the complex number representing the rotation around the origin of the angle  $\theta$  expressed in radian. This property is equivalent to the Euler's formula.

$$e^{i\theta} = \cos(\theta) + i \sin(\theta) \tag{1.84}$$

The *cos* and *sin* function are defined thanks to this formula, as respectively the real and imaginary part of  $e^{i\theta}$ . (Note that the proof of these basics properties from the definition of the exponential function are not straightforward, and are often under-estimated in textbooks).

If a 2D rotation is to be used in a Bundle Adjustment, the *update-step* would be an adaptation of the formula 1.83 re-written below, where  $\overset{\circ}{z}$  is the approximate value of the rotation, around which the derivation is computed (as in equation 1.82),  $\hat{z}$  is the updated value, and  $\theta$  is the value found in the increment vector given by the optimization algorithm e.g. Gauss-Newton algorithm.

$$\hat{z} = \exp(i\theta) \overset{\circ}{z} \tag{1.85}$$

### 1.5.3 Skew-Symmetric matrix definition and exponential

#### Skew-Symmetric matrix definition

Section 1.5.2, provided basic theory about the structure of the Lie-group of the rotation in a 2D plane. Before generalizing to 3D, we need to introduce additional mathematical background, starting with the cross product  $\times$  between two vectors  $\omega$  and  $v$ .

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \omega_y v_z - \omega_z v_y \\ \omega_z v_x - \omega_x v_z \\ \omega_x v_y - \omega_y v_x \end{bmatrix} \quad (1.86)$$

Let  $[\omega]_{\times}$  be the matrix satisfying  $[\omega]_{\times} v = \omega \times v$ . This matrix  $[\omega]_{\times}$  belongs to the set of  $3 \times 3$  skew symmetric matrix denoted  $\mathfrak{so}_3$  ( $\mathfrak{so}_3$  is the tangent-space of the set of the rotations  $\mathbb{SO}_3$  [55]).

$$[\omega]_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (1.87)$$

#### Skew-Symmetric matrix exponential

In the following, we will compute the exponential of such matrix. By definition, the exponential of a square matrix  $M$  is defined by  $\exp(M) = \sum_{n=0}^{\infty} \frac{1}{n!} M^n$  where  $M^0 = I$ .

Computing the exponential of the matrix  $[\omega]_{\times}$  is equivalent to compute the exponential of the linear application 1.88 it represents.

$$\begin{array}{ccc} \mathbb{R}^3 & \rightarrow & \mathbb{R}^3 \\ v & \mapsto & \omega \times v \end{array} \quad (1.88)$$

The exponential of the function 1.88 is 1.89.

$$v \mapsto \frac{1}{0!} I_3 v + \frac{1}{1!} \omega \times v + \frac{1}{2!} \omega \times (\omega \times v) + \frac{1}{3!} \omega \times (\omega \times (\omega \times v)) + \frac{1}{4!} \omega \times (\omega \times (\omega \times (\omega \times v))) + \dots \quad (1.89)$$

If  $\omega$  is null, the exponential is the identity. For handling the general case, let  $\vec{e}_3 = \frac{\omega}{\|\omega\|}$ , and  $\vec{e}_1$  be a unitary vector perpendicular to  $\vec{e}_3$  such that  $v$  belongs to the plane generated by  $\vec{e}_1$   $\vec{e}_3$ , and  $\vec{e}_2$  a unitary vector such that  $\vec{e}_1$ ,  $\vec{e}_2$ ,  $\vec{e}_3$  form a right handed orthonormal basis.

Let  $v_1$ ,  $v_2$  and  $v_3$  be the coordinates of  $v$  in this base, as expressed by the definition in equation 1.90 and Figure 1.21.  $v_2$  is thus null.

$$\begin{cases} v_1 = v \cdot \vec{e}_1 \\ v_2 = v \cdot \vec{e}_2 = 0 \\ v_3 = v \cdot \vec{e}_3 \end{cases} \quad (1.90)$$

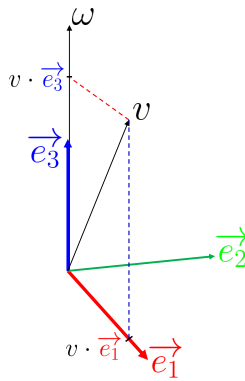


Figure 1.21. axes adapted to compute exponential of the skew symmetric matrix

This basis permits to express the elements of equation 1.89 as below.

$$\begin{cases} v & = v_1 \vec{e}_1 + v_3 \vec{e}_3 \\ \omega \times v & = \|\omega\| v_1 \vec{e}_2 \\ \omega \times (\omega \times v) & = -\|\omega\|^2 v_1 \vec{e}_1 \\ \omega \times (\omega \times (\omega \times v)) & = -\|\omega\|^3 v_1 \vec{e}_2 \\ \omega \times (\omega \times (\omega \times (\omega \times v))) & = \|\omega\|^4 v_1 \vec{e}_1 \\ \dots & = \dots \end{cases} \quad (1.91)$$

The function described by equation 1.89 could thus be written as below.

$$v \mapsto v_3 \vec{e}_3 + v_1 \vec{e}_1 + \|\omega\| v_1 \vec{e}_2 - \frac{1}{2!} \|\omega\|^2 v_1 \vec{e}_1 - \frac{1}{3!} \|\omega\|^3 v_1 \vec{e}_2 + \frac{1}{4!} \|\omega\|^4 v_1 \vec{e}_1 + \dots \quad (1.92)$$

Using the definition of  $\sin$  and  $\cos$  function given by the Equation 1.84, it is possible to show that the exponential of the linear function 1.88 is a right-hand rotation around the axes  $\omega$  by an angle  $\|\omega\|$  expressed in radians.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto v_3 \vec{e}_3 + v_1 \cos(\|\omega\|) \vec{e}_1 + v_1 \sin(\|\omega\|) \vec{e}_2 \quad (1.93)$$

The definition 1.87 and its exponential 1.93 will be used in the following section.

### Skew-Symmetric matrix deconstruction

The equation 1.87 defines the construction of a skew-symmetric matrix  $[\omega]_{\times}$  from a 3D vector  $\omega$ . The operator  $[\bullet]_{\times}$  is a function from  $\mathbb{R}_3$  to  $\mathfrak{so}_3$ . Naturally, it is possible to deconstruct a skew symmetric matrix to get back to the 3D vector via the operator  $[\bullet]^{\vee}$ .

$$\begin{aligned} [\bullet]^{\vee} : \mathfrak{so}_3 &\rightarrow \mathbb{R}_3 \\ M &\mapsto \begin{bmatrix} M_{3,2} \\ M_{1,3} \\ M_{2,1} \end{bmatrix} \end{aligned} \quad (1.94)$$

In practice, the implementation of this function requires to check the skew-symmetry of the input matrix, and to handle the possible small variations of the input matrix from a perfect skew-symmetric matrix. A  $3 \times 3$  matrix  $M$  could be considered to be skew-symmetric if the Frobenius norm of  $M + M^T$  is smaller than a given tolerance.

$$\begin{aligned} [\bullet]^{\vee} : \mathbb{R}_{3 \times 3} &\rightarrow \mathbb{R}_3 \\ M &\mapsto \begin{matrix} \text{error} & \text{if } M \notin \mathfrak{so}_3 \\ \frac{1}{2} \begin{bmatrix} M_{3,2} - M_{2,3} \\ M_{1,3} - M_{3,1} \\ M_{2,1} - M_{1,2} \end{bmatrix} & \text{else} \end{matrix} \end{aligned} \quad (1.95)$$



### Logarithm of Lie-Groups elements

The exponential of a skew symmetric matrix have been proven in 1.5.3 to be a rotation matrix. More precisely, the exponential of the skew symmetric matrix  $[\omega]_{\times}$  is the right-hand rotation around the axes  $\omega$  by an angle  $\|\omega\|$  expressed in radians. The goal of this paragraph is to define the reverse function. First, we assume the existence of such inverse function as below.

$$\forall R \in \mathbb{SO}_3, \exists v \in \mathbb{R}^3, R = \exp([\nu]_{\times}) \quad (1.96)$$

Note that this  $\nu$  is not unique. For a given unitary  $\hat{\nu}$  vector (whose norm is 1), the following function is  $2\pi$  periodic.

$$\begin{aligned} \mathbb{R} &\rightarrow \mathbb{SO}_3 \\ t &\mapsto \exp([t \hat{\nu}]_{\times}) \end{aligned} \quad (1.97)$$

It follows that for a given rotation matrix  $R$ , there is multiple candidate for  $\nu$  such that  $R = \exp([\nu]_{\times})$ . For a given candidate  $\nu$ , we define  $\hat{\nu}$  to be a unitary vector collinear to  $\nu$  (this vector could be any vector if  $\nu$  is null). For any relative integer  $k$ ,  $\nu + 2k\pi \hat{\nu}$  is also a candidate i.e.

$$R = \exp([\nu]_{\times}) \Rightarrow \forall k \in \mathbb{Z}, R = \exp([\nu + 2k\pi \hat{\nu}]_{\times}) \quad (1.98)$$

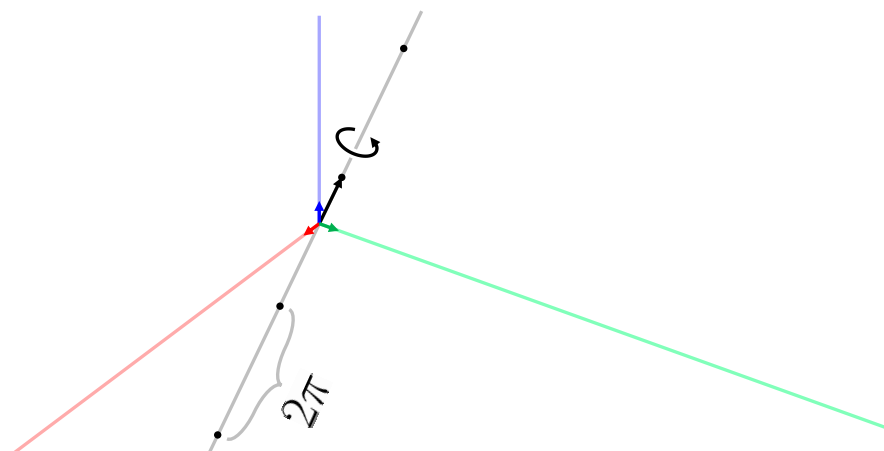


Figure 1.22. A rotation (represented by the curved arrow) have several logarithms (black dots)

This set of possible solution is represented by Figure 1.22.

We call  $\log$  the inverse function of the exponential of the skew symmetric matrix. This function is a *multivalued function* defined as below.

$$\forall R \in \mathbb{S}\mathbb{O}_3, R = \exp([\log(R)]_{\times}) \quad (1.99)$$

Two analytical expression of this log function have been found in the literature. The goal of the following is to describe them, study their drawbacks, and propose another expression. The expression given in [68] is reproduced below.

$$\log(R) = \tilde{v} \frac{\arcsin(\|\tilde{v}\|)}{\|\tilde{v}\|} \quad \text{where } \tilde{v} = \frac{1}{2} [R - R^T]^{\vee} \quad (1.100)$$

This expression presents a singularity when  $\tilde{v}$  is small, i.e. for small rotation.

A second expression, given in [45], [149] and [107] is reproduced below.

$$\log(R) = \begin{cases} \frac{1}{2} [R - R^T]^\vee & \text{for } d \rightarrow \pm 1 \\ \frac{\arccos(d)}{2\sqrt{1-d^2}} [R - R^T]^\vee & \text{for } d \in ]-1, 1[ \end{cases} \quad \text{where } d = \frac{1}{2} (\text{trace}(R) - 1) \quad (1.101)$$

The two different cases permit to avoid the singularity, but it needs to define a threshold to distinguish when does  $d$  belongs to  $] - 1; 1[$ , and when does it belongs to  $\pm 1$ .

To avoid any singularities, we suggest to come back to the definition of the inverse function of a skew-symmetric matrix. The first step is to compute the eigenvalue of  $R$ . Since  $R$  is a rotation matrix, at least one eigenvalue is unitary. Without loss of generality, we assume it to be the first one (we define the first eigenvalue  $\lambda_1$  as the one closer to 1). Let  $V_1$  be the eigenvector associated to this eigenvalue.  $R$  is a rotation matrix around the axes  $V_1$ . We assume that the norm of  $V_1$  is unitary (most implementation of eigenvector decomposition output normalized eigenvectors). The norm of  $\log(R)$  must be equal to the angle of rotation, which is the argument  $arg$  of one of the complex eigenvalue  $\lambda_2$  or  $\lambda_3$ . Our method consists in computing  $arg(\lambda_2)V_1$  and  $arg(\lambda_3)V_1$  and taking the result  $V$  such that  $exp([V]_\times)$  is the closest from  $R$ . Even if it seems to be brute-force from an algorithmic point of view, it is free from numerical singularities.

### Logarithm of matrix product

The logarithm of the product of two rotation matrices is not necessarily the sum of their logarithms. This property is true when these two matrices commute, which is the case when one of them is the identity, or when they share the same axes of rotation. In practice, we consider the property to be valid when the two matrices are close to the identity, or when their axes of rotation are close.

$$\forall R_1, R_2 \in \mathbb{S}\mathbb{O}_3, R_1 \cdot R_2 = R_2 \cdot R_1 \Rightarrow \log(R_1 \cdot R_2) = \log(R_1) + \log(R_2) \quad (1.102)$$

### 1.5.4 The $\mathbb{S}\mathbb{O}_3$ Lie-Group of 3D rotations

The additional theory of 1.5.3 will permit to transfer the results about the  $\mathbb{S}\mathbb{O}_2$  lie-group of section 1.5.2 to the  $\mathbb{S}\mathbb{O}_3$  lie-group in order to compute derivatives of 3D rotation.

### $\mathbb{S}\mathbb{O}_3$ Lie-group

The set of matrices, with the multiplication law, form a group because they satisfy the axioms of a group. We focus on the group of 3D rotation matrices, which are the  $3 \times 3$  matrices satisfying  $RR^T = R^T R = I_3$  and  $\det(R) = 1$ . The proof that  $\mathbb{S}\mathbb{O}_3$  is a group could be done by showing that  $\mathbb{S}\mathbb{O}_3$  is a sub-group of the matrix group.

Similarly to the section 1.5.2, the differentiation of the fundamental property of rotation matrix  $RR^T = I$  permits to differentiate a rotation matrix.

$$d(RR^T) = dR R^T + R dR^T = 0 \quad (1.103)$$

Similarly to equation 1.78, we get a relation between  $dR R^T$  and its transpose.

$$dR R^T = -R dR^T = -(dR R^T)^T \quad (1.104)$$

The matrix  $dR R^T$  is equal to the opposite of its transpose.  $dR R^T$  is thus an (infinitesimal) skew symmetric matrix. The section 1.5.3 showed how to build such a skew symmetric matrix.

$$\exists \omega \in \mathbb{R}^3, [\omega]_{\times} = dR R^T \quad (1.105)$$

### Differentiation in the scope of Bundle Adjustment

We have introduced sufficient amount of theory to compute the derivative of equation 1.70 with respect to the rotation matrix. Indeed, we need to compute the derivative of  $R v$ , where  $v$  is a constant  $3 \times 1$  vector (i.e. the vector  $v$  does not depend on the rotation matrix itself, as the derivation with respect to this vector is done independently from the derivation with respect to the rotation matrix). For convenience, we note  $\bar{v} = R v$ .

First, we compute the differential  $d(R v)$  of the product  $R v$ .

$$d(R v) = dR v = [d\omega]_{\times} R v = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \bar{v} \quad (1.106)$$

$$d(R v) = \omega_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \bar{v} + \omega_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \bar{v} + \omega_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{v} \quad (1.107)$$

$$d(R v) = \omega_x \begin{bmatrix} 0 \\ -\bar{v}_z \\ \bar{v}_y \end{bmatrix} + \omega_y \begin{bmatrix} \bar{v}_z \\ 0 \\ -\bar{v}_x \end{bmatrix} + \omega_z \begin{bmatrix} -\bar{v}_y \\ \bar{v}_x \\ 0 \end{bmatrix} = -[\bar{v}]_{\times} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (1.108)$$

The vector  $\omega$  is composed by infinitesimal values. The notation  $\frac{d(Rv)}{\omega}$  stands for the Jacobian matrix of  $R v$  with respect to the rotation matrix. The result is  $-[\bar{v}]_{\times}$ .

In some case, we need to compute the derivative of  $R^T v$ . First, we compute the differential of  $R^T v$ .

$$d(R^T v) = d(R^T) v = R^T [-d\omega]_{\times} v = -R^T \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} v \quad (1.109)$$

$$d(R^T v) = -R^T \left( \omega_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} v + \omega_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} v + \omega_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} v \right) \quad (1.110)$$

$$d(R^T v) = -R^T \left( \omega_x \begin{bmatrix} 0 \\ -v_z \\ v_y \end{bmatrix} + \omega_y \begin{bmatrix} v_z \\ 0 \\ -v_x \end{bmatrix} + \omega_z \begin{bmatrix} -v_y \\ v_x \\ 0 \end{bmatrix} \right) = R^T [v]_{\times} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (1.111)$$

The vector  $\omega$  is composed by infinitesimal values. The notation  $\frac{d(R^T v)}{\omega}$  stands for the Jacobian matrix of  $R^T v$  with respect to the rotation matrix. The result is  $R^T [v]_{\times}$ .

Since the Jacobian matrix of  $R v$  or  $R^T v$  is computed with respect to the tangent space of the space of the rotation in which belongs the vector  $\omega$ . Such Jacobian is a sub-part of the Jacobian  $A$  defined by equation 1.46. The optimization algorithm described in 1.4 output at each iteration an increment. This increment must be added to the approximate value if it belongs to an Euclidian space. The next section describes how to use this increment  $\omega$  to correct an approximate rotation matrix  $\overset{\circ}{R}$  in order to get a compensated matrix  $\hat{R}$ .

#### Update-step in the scope of Bundle Adjustment

This differential equation  $[d\omega]_{\times} \cdot R = dR$  could be solved as follows.

$$\tilde{R} = \exp([\omega]_{\times}) R \quad (1.112)$$

As in the equation 1.83,  $R$  is constant, and is the tangent point.  $\tilde{R}$  is the solution of the differential equation, function of  $\omega$ . The constant was chosen such that  $\tilde{R}$  coincide with  $R$  when  $\omega$  is the null vector. Since  $\exp([\omega]_{\times})$  is a rotation matrix (as shown in 1.5.3),  $\tilde{R}$  is a rotation matrix as well.

This equation is used as an *update step* for Bundle Adjustment where  $\overset{\circ}{R}$  is the approximated rotation (around which derivatives are taken),  $\hat{R}$  is the updated rotation, and  $\omega$  is given by the increment vector given by the optimization program (Gauss-Newton for example).

$$\hat{R} = \exp([\omega]_{\times}) \overset{\circ}{R} \quad (1.113)$$

**Update-step by right multiplication**

In the previous paragraphs, we have defined the update-step process with a left multiplication.

$$\hat{R} = \exp([\omega]_{\times}) \overset{\circ}{R} \tag{1.114}$$

This flows from the differential of the property:  $RR^T = I$  in equation 1.103. However, the reasoning from 1.103 to 1.113 could be applied on the fundamental property:  $R^T R = I$ .

This leads to another update step characterised by a right multiplication 1.115 as used in [68].

$$\hat{R} = \overset{\circ}{R} \exp([\omega]_{\times}) \tag{1.115}$$

From the choice of the update step follows a different result for the calculus of the rotation derivative.

$$d(R v) = dR v = R [\omega]_{\times} v = R \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} v \tag{1.116}$$

$$d(R v) = R \left( \omega_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} v + \omega_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} v + \omega_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} v \right) \tag{1.117}$$

$$d(R v) = R \left( \omega_x \begin{bmatrix} 0 \\ -v_z \\ v_y \end{bmatrix} + \omega_y \begin{bmatrix} v_z \\ 0 \\ -v_x \end{bmatrix} + \omega_z \begin{bmatrix} -v_y \\ v_x \\ 0 \end{bmatrix} \right) = -R [v]_{\times} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{1.118}$$

The vector  $\omega$  is composed by infinitesimal values. The notation  $\frac{d(Rv)}{\omega}$  stands for the Jacobian matrix of  $Rv$  with respect to the rotation matrix. The result is  $-R [v]_{\times}$ .

The derivative of  $R^T v$  could be computed with the same method. As an important result of this chapter, the following table presents the derivation of 3D rotation as a function of the definition of the update-step.

	$\hat{R} = \exp([\omega]_{\times}) \overset{\circ}{R}$	$\hat{R} = \overset{\circ}{R} \exp([\omega]_{\times})$
$\frac{\partial Rv}{\partial \omega}$	$-[Rv]_{\times}$	$-R[v]_{\times}$
$\frac{\partial R^T v}{\partial \omega}$	$R^T [v]_{\times}$	$[R^T v]_{\times}$

Table 1.4. Derivation of left and right multiplication update-step

### 1.5.5 Comparison of the $\mathbb{S}\mathbb{O}_2$ and the $\mathbb{S}\mathbb{O}_3$ Lie-Group

The aim of the following table is to give an overview of the construction and the usage of  $\mathbb{S}\mathbb{O}_3$  Lie-Group described in the previous section via a comparison with the  $\mathbb{S}\mathbb{O}_2$  Lie-Group.



### 1.5. Lie-Group tutorial

	2D: $\mathbb{S}\mathbb{O}_2$	3D: $\mathbb{S}\mathbb{O}_3$	
rotation operator	$z$	$R$	
Constraint of rotation	$z \cdot \bar{z} = 1$	$RR^T = R^T R = I_3$ & $\det(R) = 1$	
Differentiation	$dz \bar{z} = -\overline{dz \bar{z}}$	$dR R^T = -(dR R^T)^T$	$R^T dR = -(R^T dR)^T$
Element of the tangent space	$i d\theta$	$\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$	
Derivation of the rotation	$\frac{d(z\clubsuit)}{d\theta} = i z \clubsuit$	$\frac{d(Rv)}{\omega} = -[Rv]_\times$	$\frac{d(Rv)}{\omega} = -R[v]_\times$
Exponential function	$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!}$ where $z^0 = 1$	$\exp(M) = \sum_{n=0}^{\infty} \frac{1}{n!} M^n$ where $M^0 = I$	
Signification of the exponential function	$e^{i\theta}$ : rotation around the origin of the angle $\theta$ in the anti-clockwise direction	$\exp([\omega]_\times)$ : rotation around axes $\omega$ of the angle $\ \omega\ $ right hand directed.	
Update-step of Bundle Adjustment	$\hat{z} = \exp(i\theta) \overset{\circ}{z}$	$\hat{R} = \exp([\omega]_\times) \overset{\circ}{R}$	$\hat{R} = \overset{\circ}{R} \exp([\omega]_\times)$

### 1.5.6 Derivatives of rotation matrix logarithms

#### Motivation and theory

The theory presented until this point permits to compute derivatives for observations models such as image observation and GNSS observation. However, it does not permit to introduce observation from GNSS/IMU integration data which act as direct or indirect observation of attitude.

The following section is an introduction to the direct observation of rotation via the most simple rotation adjustment problem: the rotation averaging [68]. This rotation averaging will be itself introduced by the most simple adjustment problem: real values averaging.

The average of the  $n$  input observation  $\ell_1, \ell_2, \dots, \ell_n$  is  $x = (\ell_1 + \ell_2 + \dots + \ell_n) / n$ . This result results from searching a value  $x$  which is as close as possible to each  $\ell_i$ .

$$\begin{cases} \ell_1 \approx x \\ \ell_2 \approx x \\ \vdots \quad \vdots \quad \vdots \\ \ell_n \approx x \end{cases} \quad (1.119)$$

The lack of rigor in the choice of the  $\approx$  symbol in the previous equation is on purpose. The aim of the following is to add this missing rigor. Let  $v_i$  be the residual of the observation  $\ell_i$  such that  $\ell_i - v_i$  is the corrected value of  $\ell_i$ .

$$\underbrace{\begin{bmatrix} \ell_1 \\ \ell_2 \\ \vdots \\ \ell_n \end{bmatrix}}_{\ell} - \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{\mathbf{A}} x \quad (1.120)$$

The assumption that the observations  $\ell_i$  are unbiased, uncorrelated, have the same precision and follows a normal distribution leads to minimize the quadratic form  $\mathbf{v}^T \mathbf{v}$  where  $\mathbf{v}$  is the aggregated vector of the  $v_i$ . This minimization leads to the following formulation, which is

equivalent to the classical expression of the average.

$$x = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \boldsymbol{\ell} \quad (1.121)$$

The matrix rotation averaging will be built with the same logic. The aim of rotation averaging is to find the rotation  $R$  that is the closest to all the input observations  $\ell_{R_1}, \ell_{R_2}, \dots, \ell_{R_n}$ .

$$\begin{cases} \ell_{R_1} \approx R \\ \ell_{R_2} \approx R \\ \vdots \quad \vdots \quad \vdots \\ \ell_{R_n} \approx R \end{cases} \quad (1.122)$$

The definition of the symbol  $\approx$  for rotation matrices is not as straightforward as for reals numbers. It could be assumed that two rotation matrices are approximately equal if the angle of difference between the two is small, and follows a normal distribution (other definitions of  $\approx$  could be found in [68]). The angle  $\theta_i$  between  $R$  and  $\ell_{R_i}$  is given by the following expression (see Section 1.5.3).

$$\theta_i = \|\log(R^T \ell_{R_i})\| \quad (1.123)$$

The angles  $\theta_i$  between the input matrix observations  $\ell_{R_i}$  and  $R$  could be assumed to be uncorrelated, and follow a Gaussian noise with the same (small) standard deviation. This leads to the minimization of the quadratic function  $\sum \theta_i^2$ . The concatenation of the lie-logarithms of the matrices difference forms the misclosure vector  $\mathbf{v}$ .

$$\mathbf{v} = \begin{bmatrix} \log(R^T \ell_{R_1}) \\ \log(R^T \ell_{R_2}) \\ \vdots \\ \log(R^T \ell_{R_n}) \end{bmatrix} \quad (1.124)$$

The above-mentioned quadratic function  $\sum \theta_i^2$  is equal to the squared norm of the misclosure

vector  $\mathbf{v}$ .

$$\sum \theta_i^2 = \|\mathbf{v}\|^2 \quad (1.125)$$

This  $\mathbf{v}$  is defined with the final value of  $R$ :  $\hat{R}$ . The rotation averaging problem is non-linear and could be solved via an iterative process such as Gauss-Newton. At a given iteration, an approximated value  $\overset{\circ}{R}$  is available. To be compatible to the algorithm provided in [68], we use the right multiplication update step.

$$\hat{R} = \overset{\circ}{R} \exp([\omega]_{\times}) \quad (1.126)$$

Since both  $\exp([\omega]_{\times})$  and  $R^T \ell_{R_i}$  should be close to the identity, they commute (see section 1.5.3), and could be expressed as a sum of their logarithms.

$$\log(\hat{R}^T \ell_{R_i}) = -\omega + \log(\overset{\circ}{R}^T \ell_{R_i}) \quad (1.127)$$

The aggregation of the input from all rotation matrices could be expressed in matrix form.

$$\underbrace{\begin{bmatrix} \log(\overset{\circ}{R}^T \ell_{R_1}) \\ \log(\overset{\circ}{R}^T \ell_{R_2}) \\ \vdots \\ \log(\overset{\circ}{R}^T \ell_{R_n}) \end{bmatrix}}_{\overset{\circ}{\mathbf{v}}} - \underbrace{\begin{bmatrix} \log(\hat{R}^T \ell_{R_1}) \\ \log(\hat{R}^T \ell_{R_2}) \\ \vdots \\ \log(\hat{R}^T \ell_{R_n}) \end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} I_3 \\ I_3 \\ \vdots \\ I_3 \end{bmatrix}}_{\mathbf{A}} \omega \quad (1.128)$$

Minimizing the squared norm of the residuals vector  $\mathbf{v}$  via Gauss-Newton algorithm leads at each step to compute  $\omega$  via the following formula.

$$\omega = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \overset{\circ}{\mathbf{v}} \quad (1.129)$$

Then, the update step is performed thanks to the right multiplication update formulae: 1.126.

This rotation averaging algorithm was taken from [68]. This same reasoning could be applied starting from the left multiplication update step, and will leads to a different algorithm which will provide exactly the same result.

### Practical application

In practice, an orientation observation could be written in the form  $\ell_R^T R = I_3$  where  $R$  is a unknown matrix to be adjusted in the Bundle Adjustment, and  $\ell_R^T$  is another rotation matrix containing input observation.

The right multiplication update step leads to replace  $\hat{R}$  by  $\overset{\circ}{R} \exp([\omega]_{\times})$ .

$$\ell_R^T \overset{\circ}{R} \exp([\omega]_{\times}) = I_3 \quad (1.130)$$

Knowing that the exponential of the opposite is equal to the inverse of the exponential, we could have an expression for  $\omega$ .

$$\omega = -\log(\ell_R^T \overset{\circ}{R}) \quad (1.131)$$

This leads to the following derivative that could be used in the Jacobian matrix.

$$\frac{\partial \log(\ell_R^T \overset{\circ}{R})}{\partial \omega} = -I_3 \quad (1.132)$$

This same reasoning could be applied with  $R^T$  and with the left multiplication update. The rotation observation that we need to apply ( $R \ell_R^T = I_3$ ,  $\ell_R^T R = I_3$ ,  $\ell_R R^T = I_3$  or  $R^T \ell_R = I_3$ ) leads the choice of update-step used in the Bundle Adjustment (right multiplication or left multiplication). Note that in the same Bundle Adjustment, both type of update-step could be used, the derivatives must be adapted accordingly, as summarized in Table 1.5.

This same method could also be applied to add relatives orientation observation  $\ell_{R_{ij}}$  such that  $R_i \ell_{R_{ij}} = R_j$  or  $\ell_{R_{ij}} R_i = R_j$  (where  $R_i$  and  $R_j$  are rotation matrices to be determined) as it is presented in [29].

	$\hat{R} = \exp([\omega]_{\times}) \mathring{R}$	$\hat{R} = \mathring{R} \exp([\omega]_{\times})$
Absolute orientation	$R \ell_R^T = I_3$ $\frac{\partial \log(\mathring{R} \ell_R^T)}{\partial \omega} = -I_3$	$\ell_R^T R = I_3$ $\frac{\partial \log(\ell_R^T \mathring{R})}{\partial \omega} = -I_3$
	$\ell_R R^T = I_3$ $\frac{\partial \log(\ell_R \mathring{R}^T)}{\partial \omega} = I_3$	$R^T \ell_R = I_3$ $\frac{\partial \log(\mathring{R}^T \ell_R)}{\partial \omega} = I_3$
Relative orientation	$R_i \ell_{R_{ij}} R_j^T = I_3$ $\frac{\partial \log(\mathring{R}_i \ell_{R_{ij}} \mathring{R}_j^T)}{\partial \omega_i} = -I_3$ $\frac{\partial \log(\mathring{R}_i \ell_{R_{ij}} \mathring{R}_j^T)}{\partial \omega_j} = I_3$	$R_j^T \ell_{R_{ij}} R_i = I_3$ $\frac{\partial \log(\mathring{R}_j^T \ell_{R_{ij}} \mathring{R}_i)}{\partial \omega_i} = -I_3$ $\frac{\partial \log(\mathring{R}_j^T \ell_{R_{ij}} \mathring{R}_i)}{\partial \omega_j} = I_3$

Table 1.5. Derivatives of absolute and relative orientation

## 1.6 Benchmarking Euler-Angles vs Lie-Groups

The previous sections presents the Lie-group theory for rotation handling, and its application in Bundle Adjustment. The use of Lie-groups in Bundle Adjustment is more rigorous and leads to simpler derivatives than Euler-Angles. The aim of this section is to assess the performances of Lie-groups compared to the ones of Euler-Angles.

The performances of these two methods were tested on two simulated data-sets. The first-one is a classic small block of aerial photogrammetry (Figure 1.23) while the second one is a close range survey of a 3D grid (Figure 1.24) with various orientations that could leads to gimbal lock of Euler-Angles. Table 1.6 summarizes the characteristics of both data-sets<sup>15</sup>. The two

<sup>15</sup>In the Aerial photogrammetry block, the tie-points observations are uniformly distributed on the images. In the close-range photogrammetry scenario, the images observations span the whole image (i.e. the width and height of the close-range photogrammetry scenario corresponds to the area of the images where images observations are

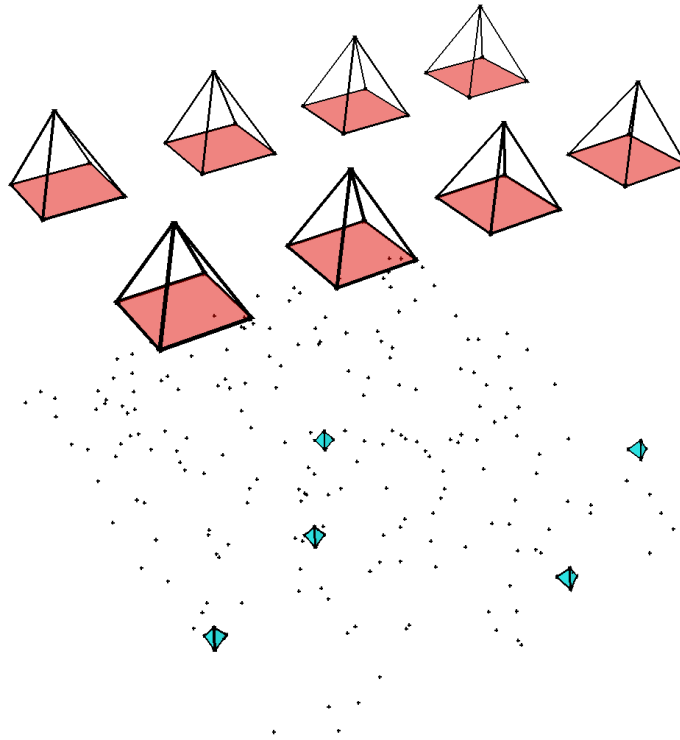


Figure 1.23. Simulated data-set 1: Classical Aerial photogrammetry block

methods are tested on three criteria: the resistance to bad initial values, the convergence rate, and the singularity of outputted covariance matrices.

### 1.6.1 Resistance to bad initialization

The most important feature expected from a Bundle Adjustment algorithm is to converge to the correct solution i.e. to the global minima. This requires to start from relatively good initial values, or that the algorithm is robust to bad initial values. The goal of this study is to assess the robustness of the algorithm to bad initial values.

Two Bundle Adjustment algorithms were implemented with the Gauss-Newton algorithm: one using Euler angles and the other using Lie-Group. Both algorithms were tested with different level of noise on the initial values added on position and orientation. For each noise level (represented on the abscissa for the noise on position and on ordinate for the noise on the orientation, table 1.7), ten set of initial values were generated. Both algorithm were tested with the inputted noisy initial guess. The results were compared to the ground truth to determine if the algorithm converges to the right global minima. On total, 48 002 runs of the algorithms

located.)

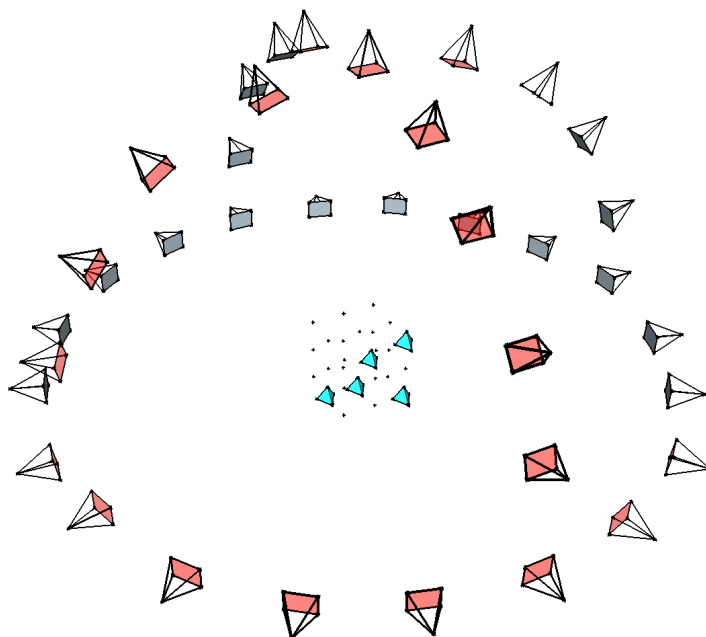


Figure 1.24. Simulated data-set 2: Close range photogrammetry inspired from [91] and [162]

were performed (30 position noise levels  $\times$  40 orientation noise levels  $\times$  10 simulations at a given level of noise  $\times$  2 data-set  $\times$  2 algorithms + two runs to determine the ground-truth). The success rate is represented in gray-scale on the map of the first two lines of Table 1.7.

For the first data-set considering a classical photogrammetric project (first column of table 1.7) both algorithms performs with the same performances. The success rate is 20% for both algorithms (this number needs to be considered with precautions since it is completely dependent on the chosen level of noise on the initial value). The case where one algorithm succeed while the other fails are very rare (less than 2% of the cases of at least one algorithm success).

The second data-set offers a way more difficult situation (second column of table 1.7), especially for the Euler-Angle based algorithm. The success rate is 25% for the Lie-Group based algorithm while only 6.8% for the Euler-Angle one. We do not reported a single case where the Euler-angle algorithm success while the Lie-group one fails.

### 1.6.2 Convergence rate

The previous paragraph studied if an algorithm converges or not. The goal of this paragraph is to study the convergence of an algorithm when this-one converges to a global minima.



## 1.6. Benchmarking Euler-Angles vs Lie-Groups

	Data-Set 1 Aerial photogrammetry block	Data-Set 2 Close range photogrammetry
nb images	8	37
nb of strips	2	3
longitudinal overlap	60%	100%
lateral overlap	40%	100%
nb GCPs	5	5
nb Tie-Points	200	26
mean image observation (i.e. nb of points per images)	75	26
mean dist. to object	100 <i>m</i>	10 <i>m</i>
Principal Distance [ <i>pix</i> ]	5000	5000
Width [ <i>pix</i> ]	5000	1400
Height [ <i>pix</i> ]	5000	1150
GSD	2 <i>cm</i>	2 <i>mm</i>
noise on image observation	0.5 <i>pix</i>	0.5 <i>pix</i>
Internal Orientation	Fix	Fix
Aerial control	No	No
Other sensors	No	No

Table 1.6. Simulated data-set characteristics

Gauss-Newton algorithm is iterative. The convergence rate of iterative algorithms could be visualized on semi-logarithmic plots (third line of table 1.7). The curves display for each iteration the distance to the final value given by the last iteration. The evolution of the position is the mean value of the distance between the current and the final camera positions, and the evolution of the orientation is the mean angle between the final camera orientations and the one at the current iteration<sup>16</sup>.

For the first data-set (the classical photogrammetric block), the convergence rate is very similar between the two algorithms, both for the position and the orientation. However, for the second data-set (close range 3D grid survey), the Lie-group based algorithm converges faster to the solution. More than two iteration are needed to the Euler-angle algorithm to outperform the Lie-group one.

<sup>16</sup>The reference value at the end of the convergence were chosen to be the one given by the Euler-Angles based algorithm to ensure not to underestimate its performances. The difference at the last iteration is thus 0 for the Euler-Angles based algorithm, which is not representable on a logarithmic plot.

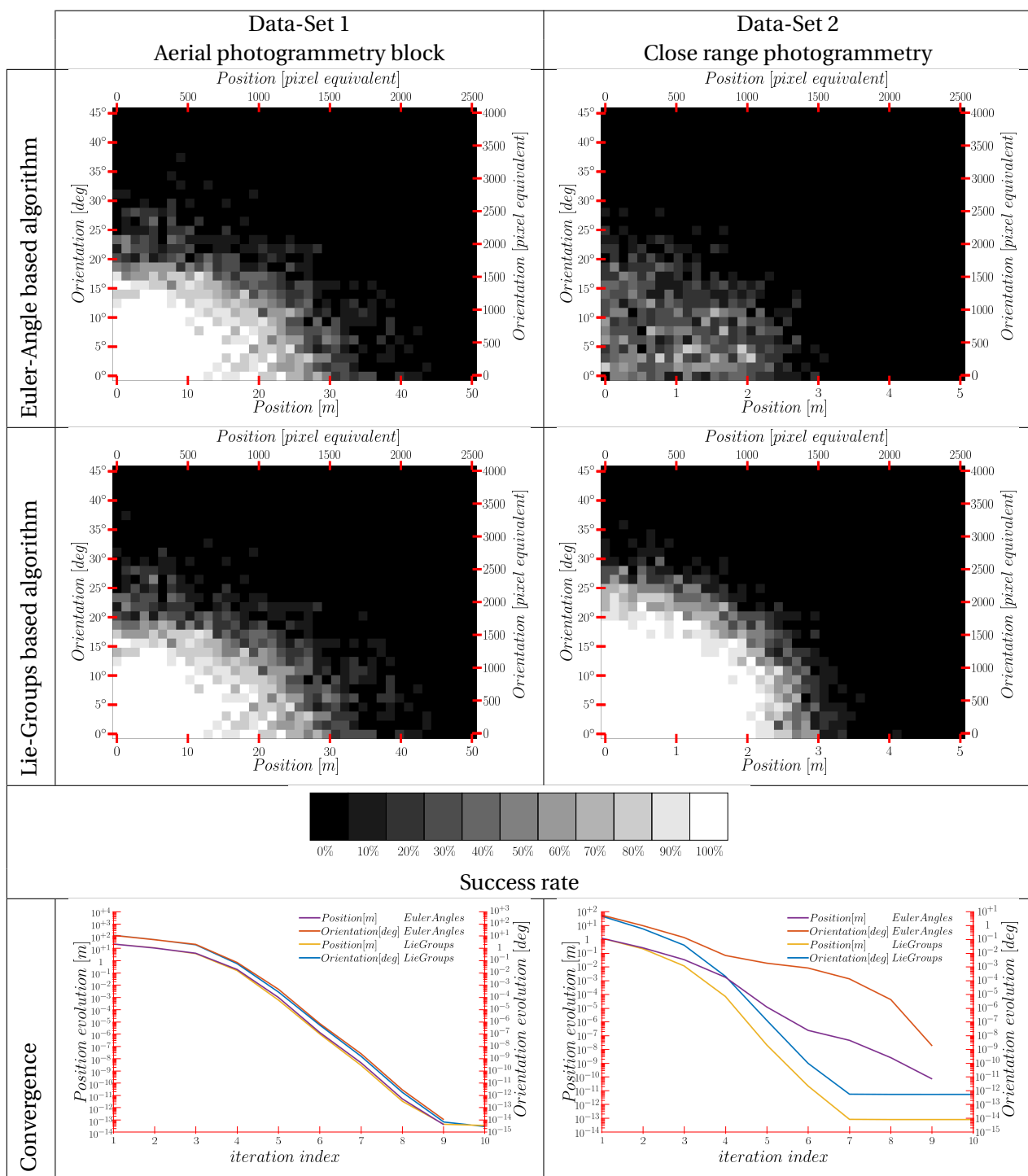


Table 1.7. Benchmarking of Euler and Lie-Groups based Algorithms on two simulated data-set

## 1.6. Benchmarking Euler-Angles vs Lie-Groups

### 1.6.3 Pose covariance matrix

At the end of the iterative process, if both algorithms converges to the global minima, they give exactly the same values for the position and orientation of the camera poses, and for the position of the points on the ground<sup>17</sup>. However, the way they express the precision of the outputted results is different.

The following *a posteriori*  $\sigma$  and correlation matrices represents the expected precision on a given camera pose of the second data-set (close range 3D grid survey).

		x	y	z	$\omega$	$\psi$	$\chi$
x	3.9 mm	100	3	-2	-4	6	4
y	5.3 mm	3	100	-22	-99	27	99
z	5.0 mm	-2	-22	100	17	-99	-17
$\omega$	227 °	-4	-99	17	100	-22	-100
$\psi$	0.03 °	6	27	-99	-22	100	22
$\chi$	227 °	4	99	-17	-100	22	100
$\sigma$		$\rho$					

Euler Angle  
standard deviation and correlation matrix

		x	y	z	$\omega_x$	$\omega_y$	$\omega_z$
x	3.9 mm	100	3	-2	-1	6	4
y	5.3 mm	3	100	-22	-24	22	99
z	5.0 mm	-2	-22	100	11	-99	-21
$\omega_x$	0.46 °	-1	-24	11	100	-13	-19
$\omega_y$	0.48 °	6	22	-99	-13	100	22
$\omega_z$	0.50 °	4	99	-21	-19	22	100
$\sigma$		$\rho$					

Lie-Group  
standard deviation and correlation matrix

Table 1.8. Standard deviation and correlation matrices outputted by Euler-Angles and Lie-Groups based algorithm in a situation of gimbal lock

The covariance matrices (Table 1.8) are the same for the position of the image, however, they are different for the orientation. The covariance matrix of the Euler angles representing the orientation of the camera is singular, whereas the covariance matrix of the values on the tangent space around the rotation is not. The standard deviation of the angles  $\omega$  and  $\chi$  are not meaningful since there are more than  $180^\circ$  i.e. the angles are not determined. The covariance matrix outputted by the Lie-Group based algorithm is not singular. The  $\omega_x$ ,  $\omega_y$  and  $\omega_z$  denote here the components of the rotation increment value used in the update step 1.114. Thus,  $\omega_x$ ,  $\omega_y$  and  $\omega_z$  represent small rotations around axis respectively X, Y and Z of the local frame, which makes the covariance matrix easier to interpret than with Euler-Angles.

### 1.6.4 Conclusion for the comparison of the use of Euler-Angle and Lie-Groups in Bundle Adjustment

This study shows that for classical photogrammetry scenario such as a block of aerial photos, both Euler-Angle and Lie-Groups based algorithms performs similarly. However, in difficult scenarios that could lead to gimbal lock of the Euler-Angles sequence, the Lie-Group approach outperforms the parametrisation via Euler-Angles significantly in terms of i) larger convergence region, ii) faster convergence iii) stability - absence of singularity.

<sup>17</sup>Up to numerical rounding errors, here  $10^{-13}m$  for the position and  $10^{-12}deg$  for the orientation

## **Conclusion**

This chapter gave the needed theory for the implementation of sensor-fusion algorithms that will be proposed and evaluated in the subsequent chapters.

It also has compared several propositions of sensor models. E.g. 1.2.7 described different possibilities to input orientation, and 1.6 evaluated two approaches when handling rotation in the Bundle Adjustment.

The comparison of the different camera models deserves a separate complete section 1.1.6 would go beyond the scope of this first chapter, and thus deserve a full chapter: Chapter 2.

## 2 Toward Camera Calibration models and methods

In Chapter 1, several camera models have been proposed. This chapter compares these camera models together with the calibration procedure to determine the parameters and the method to best use these parameters in application for difficult scenario such as corridor mapping. This chapter is originated from the following preprint.

E. Cledat, D. A. Cucci and J. Skaloud. Camera calibration models and methods in corridor mapping with UAVs *ISPRS Conference Nice, 2020*

The idea was first suggested by J. Skaloud, the hardware assembling and the data acquisition involved all three authors, the programming of the software and the experimental procedure were achieved by E. Cledat and for this specific chapter, the redaction was mainly conducted by D. A. Cucci.

### **Abstract**

Camera calibration refers to the modeling of the relationship between the coordinates of object points and their projections on the image plane. This is usually done by parametric models that describe the physical properties of the lens systems and camera assemblies, such as the camera principal distance, the principal point, and various types of optical distortions. In photogrammetry, accurate knowledge of the parameters of such models, often referred to as Interior Orientation (IO), is of ultimate importance. In this work, we target advanced corridor mapping applications with UAVs. In this scenario, the camera calibration is not completely observable due to the unfavorable geometry of the flight trajectory (e.g., no cross flight lines available and a single altitude) and needs to be determined beforehand. Further challenges are introduced by the limited mechanical stability of UAV-grade cameras. This may cause slight variations in the IO that need to be recovered while processing production flights. We review and compare two well known camera models, the Brown-Conrady and the Ebner's

self-calibration functions, in 36 calibration setups and provide a discussion of the results, where sub ground sampling distance accuracy in the checkpoints was achieved for some, but not all, configurations.

### 2.1 Introduction

Camera calibration, or the camera Interior Orientation (IO), is an essential prerequisite to determine precise and accurate three dimensional, metric, information from images. It models the relationship between the coordinates of object points in the camera reference frame and the corresponding image coordinates. The knowledge of such relationship allows, among others, to recover three dimensional object coordinates from multiple views by means of a bundle adjustment algorithm [158]. The IO is specific to each particular assembly of camera and lenses and may change in time [86].

A wide range of digital cameras commonly employed in photogrammetry are equipped with lenses designed to obey the perspective projection law (i.e., the collinearity equations). However, this is seldom true in reality because of at least four effects related to imperfections in their construction [52]: i) symmetric radial distortion, ii) decentering distortion, iii) image plane unflatness and iv) in-plane image distortion. These can cause non negligible deviations from collinearity. At least two families of models have been proposed to account for this: Brown-Conrady [24] (abbreviated to Brown in the following) and Ebner's self calibration functions [46]. These models are characterized by a variable number of parameters that need to be determined, through a process called camera calibration [133]. An important issue is the quality of the determination of these variables which describe the behavior of the camera, and especially, their correlations. The two sources of these correlations could be classified into to categories. 1) Structural through model definition, e.g. the radial parameters of the Brown model are naturally correlated. 2) Through the calibration process, e.g. the lack of scale variation due to the use of a 2D target field leads to correlations between the principal distance and the principal point.

Camera calibration is typically approached by acquiring a large set of images (typically more than 50) of an object (such as a chessboard or a calibration field composed of several Ground Controls Points: GCPs) in order to describe accurately the camera behavior. The parameters of the chosen camera model (e.g., Brown) are then estimated to best fit the observations. A very well known approach in computer vision relies on the use of easily recognisable targets (e.g., checkerboards) whose metric dimensions are known. Several established tools are available to determine Brown's distortion coefficients in this setup, e.g., [21], however, these often yield correlated estimates for the calibration parameters. One reason is that all the known points lie on the same plane. In laboratory calibration setups, the stand-off distance is typically much smaller with respect to target applications, which leads to different values of distortion

coefficients [92]. Therefore, photogrammetrists typically prefer to perform camera calibration in conditions that are similar with respect to the target application. For example, in aerial photogrammetry, dedicated flights are performed over a field equipped with several GCPs: the orientation of the camera, along with the IO, are recovered during the bundle adjustment. This approach also relies on additional points identified automatically (tie-points) whose object space coordinates are not known.

Further issues related to camera calibration arise in the emerging field of aerial photogrammetry based on Unmanned Aerial Vehicles (UAVs), which are becoming an essential tool for surveyors, engineers and scientists [39]. Here, consumer-grade cameras and lenses are employed due to payload size, weight and cost limitations of UAVs. The mechanical stability of such sensors is uncertain: for example, the  $\mu\text{m}$  level positioning and alignment of the lens might change when vibrations, bumps during landing, temperature variations, etc. occur during the operation of a UAV. This implies that the IO determined from one calibration flight may not be directly applicable, as it is, in subsequent production flights [41]. However, in general it is not possible to determine the IO from scratch in production flights as i) the geometry of certain mapping missions, e.g., for corridor mapping, is such that the IO is not fully observable, ii) practitioners strive to reduce the number of GCPs since they are time and cost intensive. This calls for sound methods to exploit the IO from dedicated calibration flights but at the same time recover minor variations due to the mechanical instability of consumer-grade cameras.

In this work we focus on long corridor mapping missions with UAVs where no GCPs are available. Here, a good *a priori* knowledge of IO is essential since the difficult configuration (long corridor with small height variations: the maximum height difference of the terrain is around  $10\text{m}$ ) does not permit a satisfactory self-calibration of the camera. We consider five instances of models from the Brown's and Ebner's families, differing by the number of IO parameters, which we determine using two different calibration setups. Next, we employ these IOs in a 2 km long corridor mapping flight, first using only values determined during calibration, and then using three different strategies to estimate corrections for camera mechanical instabilities during the bundle adjustment. Practically unbiased Check Points (CPs) errors and sub-Ground Sampling Distance (GSD) root mean squared error (RMS) were achieved, in certain cases, approaching the accuracy to which CPs are known. We've also found combinations of models and calibration strategies which do not provide satisfactory results, for which we investigate the causes and provide an extensive discussion.

This work is organised as follows: in Section 2.2 we review the Brown and Ebner's calibration models in a unified formulation. In Section 2.3 and 2.4 we review the calibration state of the art in Integrated Sensor Orientation (ISO) and we present three strategies to re-calibrate the IO during production flights. In the last sections we present the results of a rigorous experimental

evaluation of the different models and strategies presented in a real-world corridor mapping application.

### 2.2 Camera Models

In this section we review two approaches to model the departures from collinearity typical of many narrow angle cameras and lenses. The first is the physical oriented approach, the Brown function, while the second, more numerically oriented, is given by the Ebner self-calibration functions, and later extensions. Whereas the principles behind those are well known, several slightly different formulations have been presented in the literature. Thus, we report those used in this study in the following analysis.

We define the projection function  $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$  as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}, \quad (2.1)$$

where  $[X, Y, Z]^T$  are the object coordinates of a point with respect to the camera reference frame (in meters) and  $[x, y]^T$  is the corresponding (unit-less) projection on a plane at unitary distance from the optical center of the camera. The well known pinhole camera model is found by multiplying  $\pi$  by the principal distance of the camera, e.g., in pixels. Many lens systems are designed to best follow this relation.

More complex camera models, able to account for different kinds of distortion effects introduced by real lens systems, build on  $\pi$  as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \xi\left(\pi([X, Y, Z]^T)\right), \quad (2.2)$$

with  $\xi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . Image coordinates  $[x', y']^T$  are typically expressed in pixels. The function  $\xi$  involves a set of parameters which are generally referred to as Additional Parameters (APs), or camera Interior Orientation (IO).



### 2.2.1 Brown distortion model

In Brown's distortion model [24], the function  $\xi(\cdot)$  is defined as the composition of two other functions  $\xi = \xi_2 \circ \xi_1$ : the output of the function  $\xi_1$  is used as input of the function  $\xi_2$ .

$$\xi_2 : \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} f + B_1 & B_2 \\ 0 & f \end{bmatrix} \begin{bmatrix} x'' \\ y'' \end{bmatrix} + \begin{bmatrix} pp_x \\ pp_y \end{bmatrix}, \quad (2.3)$$

$$\xi_1 : \begin{bmatrix} x'' \\ y'' \end{bmatrix} = (1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + \dots) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} (P_1 (r^2 + 2x^2) + 2P_2 xy) \\ (2P_1 xy + P_2 (r^2 + 2y^2)) \end{bmatrix} (1 + P_3 r^2 + \dots). \quad (2.4)$$

where:

1.  $f$  and  $[pp_x, pp_y]^T$  are the principal distance and the principal point (in pixels),
2.  $B_1$  accounts for non-uniform scaling and  $B_2$  for skewing along the axis of the imaging sensor,
3.  $K_i$  and  $P_i$  are the radial and tangential distortion coefficients,
4. “...” stands for an arbitrary number of additional terms in the polynomial expansion in  $r^2$ ,
5.  $r^2 = x^2 + y^2$ .

This formulation is very well known and corresponds to the one implemented in established photogrammetry software, such as Agisoft Metashape, or open-source computer vision libraries, such as OpenCV.

In this work we consider three specific instances of the Brown model, referred to as Brown10, Brown15 and Brown18 in the following, which differ in the total number of parameters employed. Whereas  $f$ ,  $pp_x$ ,  $pp_y$ ,  $B_1$  and  $B_2$  are always considered, the models differ in the order of the polynomial expansions for the radial and tangential distortion (with respect to  $r^2$ ):

1. Brown10:  $K_i, i \in [1, \dots, 3], P_j, j \in [1, \dots, 2]$ ,
2. Brown15:  $K_i, i \in [1, \dots, 6], P_j, j \in [1, \dots, 4]$ ,
3. Brown18:  $K_i, i \in [1, \dots, 8], P_j, j \in [1, \dots, 5]$ .

The number of coefficients in 2.(Brown15) and 3.(Brown18) exceeds the numbers of coefficients that are commonly used by photogrammetrists. The number of coefficients has been

chosen to match the one of orthogonal polynomial model in order to proceed to a fair comparison in the experimental evaluation. The state-of-the-art approach would consist in keeping only the coefficients which values surpasses its confidence level. However, in this study, the goal is to keep all the parameters, and use them in the direct application in the test flight together with their full covariance matrix.

### 2.2.2 Orthogonal polynomials

A second family of models for camera distortions was introduced in [15] building on a typical example of Orthogonal polynomials: the Ebner's self-calibration functions [46] (extensions of Ebner's Orthogonal polynomials such as Grün polynomials:[65] are also referred in [15] but increase the number of parameters). We refer the reader to the original publications for the derivations. The function  $\xi$  reads as follows:

$$\xi : \begin{matrix} x' = \begin{bmatrix} 1 \\ x \\ x^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 - \frac{2}{3}b_y^2 \end{bmatrix} \\ y' = \begin{bmatrix} 1 \\ x \\ x^2 - \frac{2}{3}b_x^2 \end{bmatrix}^T \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 - \frac{2}{3}b_y^2 \end{bmatrix} \end{matrix}, \quad (2.5)$$

where  $b_x$  and  $b_y$  are the width and the height of the image (in pixels) divided by twice the nominal focal length (in pixels). This model is fully defined by the set of 18 parameters  $a_{ij}$  and  $b_{ij}$  and will be referred to as OrthoPoly18 in the following.

Brown's and Orthogonal Polynomial models are intrinsically different and it is not possible to give a closed form expression that would map parameters from one model to the other, except for the following:

1.  $a_{11} \rightarrow pp_x$ ,
2.  $b_{11} \rightarrow pp_y$ ,
3.  $a_{21} \rightarrow f + B_1$ ,
4.  $b_{12} \rightarrow f$ .

## 2.3 Camera Calibration

In order to properly compensate for the distortions introduced by the lens system, one has to first choose which model is best suited for the camera in use, and second determine the

values for the parameters defining the model (the IO, or the APs). The first step is often left to user experience, even though modern photogrammetry software is able to automatically select the most significant parameters to be considered within a model family (e.g., choose between Brown10 and Brown15). The second step is called calibration. If the calibration is performed without the use of an external instrument to characterize the sensor, we speak of *self*-calibration. The methods are well known, and we are going to summarize them in the following.

Camera calibration via aerial photogrammetry is commonly achieved by establishing a dense network of accurately geo-referenced ground control points (GCPs) over a *calibration field*. Dedicated flights are executed to collect several images in a favourable geometrical configuration (e.g., cross flight lines, different elevations and a mix of nadir/oblique views). The images are then oriented by means of the bundle adjustment algorithm, which determines the camera exterior orientation (EO), the parameters of the distortion model (IO), and possibly their precision and reliability. This approach is commonly referred to as Indirect Sensor Orientation (InSO).

In InSO, it is sometimes difficult to achieve optimal de-correlation between the camera EO and IO, and within the IO itself, even when the calibration have been performed with several camera orientations and flight heights. This is the case when the GCPs lie on the same plane, such as when the computer vision approach to camera calibration, based on checkerboards, is employed. In this case the parameters corresponding to the focal length ( $f$  in BrownXX and  $a_{21}$ ,  $b_{12}$  in OrthoPolyXX) remain correlated with the camera position.

If the aerial mapping platform is equipped with a survey grade GNSS receiver, and (optionally) an IMU, the additional information made available by these sensors can be introduced in the bundle adjustment. This approach is called Integrated Sensor Orientation (ISO), see for example see [127] for a comprehensive description focused on UAVs.

In ISO, GNSS and inertial observations are fused together by means of a Kalman filter/smoothing in a pre-processing step. This calculates positions and orientations for the camera that can be used as prior information in the bundle adjustment (aerial control, in photogrammetry jargon). More modern approaches consider a single step where both image and raw inertial/GNSS observation are adjusted together [43]. The availability of the extra information from GNSS and inertial sensors helps to de-correlate the camera model parameters from the camera exterior orientation.

In UAVs, only MEMS IMUs can be employed due to space and take-off weight limitations. These are substantially less accurate with respect to tactical or navigation grade IMUs commonly employed in airborne photogrammetry. Thus, the GNSS/inertial solution might be biased, or not sufficiently accurate. In this case, a very effective approach is to use relative,

instead of absolute, position/orientation control in the bundle adjustment, as proposed in [16] and further investigated for the UAV scenario in [131]. This technique exploits the fact that the GNSS/inertial solution may not be sufficiently accurate, but is locally precise, and thus it can be effectively employed to constrain the *relative* change between subsequent image orientations. Furthermore, relative orientation control eliminates the need to determine the camera boresight with respect to the IMU. We refer the reader to the original publications for the details.

### 2.3.1 Orthogonal Polynomials over-parameterization

In InSO, OrthoPoly18 suffers from over-parameterization: some of the parameters defining the model fully correlate with the exterior orientation of the camera. This means that if no absolute aerial orientation control is available, unreliable estimates or worse, numerical instabilities and singularities will be obtained during the bundle adjustment.

To address this issue, in [15], the authors introduced six constraints on the parameters  $a_{ij}$ ,  $b_{ij}$ :

$$\left\{ \begin{array}{l} a_{11} = b_{11} \\ a_{21} = -b_{12} \\ b_{13} = -2a_{22} \\ a_{31} = -2b_{22} \\ a_{12} = b_{21} \end{array} \right. . \quad (2.6)$$

If these are used to simplify Equation 2.5, the original 12 parameter Ebner's self-calibration functions are obtained.

In this work we've chosen to omit the first two constraints in Equation 2.6, since they correspond to  $f$  and  $pp$  in BrownXX. These are important for modeling consumer-grade cameras where the lens system can not be considered fully geometrically stable in time, or in large camera systems where they depend on environmental factors such as temperature and air pressure. Simplifying Equation 2.5 with the remaining three constraints gives OrthoPoly15, which requires at least absolute aerial position control during calibration.

### 2.3.2 Parameters significance

The state of the art method to select the appropriate number of parameters, and thus to prevent over-parametrisation is to compute the significance of each parameters. The significance of the parameter  $x$  is quantified by the ratio  $|x|/\sigma_x$ . By convention we consider a parameter to be significant when this ratio is above 1. Therefore, the probability that its sign is wrong (+

instead of – or conversely) is lower than 16%.

Table 2.3 presents the significance of the IO parameters calibrated in the section 2.5. Surprisingly, the significance of the parameters, and especially of the radial and tangential polynomials coefficients do not decrease with the degree of the monomials which constitute the polynomial. The goal of this study is to assess the quality of IO parameters in use in application. The difference of parameters significance showed in Table 2.3 are not reflected on the quality of the use of these parameters for another flight: Table 2.2.

## 2.4 Re-estimating the Interior Orientation

In UAVs, consumer grade digital cameras are commonly employed. These are not built for photogrammetry applications and can not be considered mechanically and geometrically stable. In particular, the alignment of the lens with the imaging sensor is subject to change due to external stresses such as vibration or temperature change. The lens radial distortions are in general considered to be more stable, but can still depend on environmental factors. This means that the IO may slightly change with time and thus be different between calibration and production flights.

To account for this, IO is often re-calibrated during production flights. This means that the bundle adjustment algorithm is initialized with the IO determined from calibration flights, and estimated corrections which account for camera instabilities are applied while processing subsequent production flights. Indeed, particular care needs to be taken as the geometry of production flights may not guarantee the observability of the full IO (e.g., in corridor mapping). In this work we consider three strategies which are discussed in the following.

### 2.4.1 Leading Lead

The *leading* option consists in estimating the *leading* parameters  $f$ ,  $pp_x$  and  $pp_y$  in BrownXX or the corresponding parameters  $a_{11}$ ,  $a_{21}$ ,  $b_{11}$  and  $b_{12}$  in OrthoPolyXX, while the remaining parameters are kept fixed. This option and naming is inspired by the drone mapping software such as Pix4D Mapper or Agisoft Metashape.

### 2.4.2 *a posteriori* covariance APC

Let  $\Theta$  be the vector of the interior orientation parameters, e.g., for BrownXX:

$$\Theta = [f, pp_x, pp_y, B_1, B_2, \dots] \quad (2.7)$$

The *a posteriori* covariance matrix of this vector,  $\Sigma_{\Theta}$ , is available as an additional output from the bundle adjustment run on the dedicated calibration datasets. The square root of the diagonal of  $\Sigma_{\Theta}$  is typically reported as the error bound for the IO vector. On the off-diagonal terms, often neglected yet significant to experienced practitioners, we find the covariance between two parameters. Non-complete observability of some parameters is spotted by computing the correlation matrix from  $\Sigma_{\Theta}$  and looking for off-diagonal terms which are close to one in absolute value.

We can exploit the prior information on the IO available from the calibration flight, including the residual correlations between the single parameters, for the re-estimation of the IO. During the bundle adjustment for the production flight we re-estimate the entire IO vector, where we include an extra observation equation as follows, where  $\Theta_0$  have been determined in a calibration flight.

$$\Theta - \Theta_0 = 0, \tag{2.8}$$

$\Sigma_{\Theta}^{-1}$  weights the pseudo-observations  $\Theta_0$ .

### 2.4.3 *a posteriori* covariance inflated APCI

While  $\Sigma_{\Theta}$  is the best estimate of uncertainty for the camera IO just after the calibration flight, the use of this information in production flights does not account for possible changes in IO due to camera instability.

We thus propose to scale up the diagonal elements corresponding to the leading parameters defined in Section 2.4.1. The degree to which each parameter is up-scaled is determined by the amount that they are expected to change with time.

## 2.5 Experimental evaluation

In this work we address challenging corridor mapping applications (Figure 2.3) where we seek to achieve a ground accuracy that is better than the GSD using no ground control points. In these cases, it is essential to determine a reliable IO beforehand since the geometry of such production flights does not allow for self-calibration. Indeed, in corridor mapping missions we typically do not observe at multiple elevations and/or orthogonal flight-lines. Furthermore, in our case no ground control point is available. Another important element is the correct choice of the camera model, which needs to be able to compensate for lens distortion while minimizing the number of parameters employed.

For all the flights, we have employed an aerial photogrammetry payload tailored for a small

## 2.5. Experimental evaluation

	CF1	CF2	PF
UAV type	Copter	Fixed wing	Fixed wing
Aerial control	No	Yes	Yes
Embedded GNSS antenna receiver precision (X,Y,Z) [mm]		4,4,11	4,4,11
Relative orientation precision		$0.015^\circ / \sqrt{s}$	$0.015^\circ / \sqrt{s}$
Geometry	Close range	Block	Corridor
Images	75	440	290
Flight lines		26	4
Flight levels	2	2	2
Long. overlap [%]	$\approx 100$	65	70
Lat. overlap [%]	$\approx 100$	45	70
Mean depth [m]	16.3	157	117
Min depth [m]	6.9	111	84
Max depth [m]	22.7	546	186
mean GSD [mm]	3	30	20
Tie-points	2,565	22,955	23,813
# GCPs	17	21	0
# CPs	1	4	24
GCPs accuracy (XYZ) [mm]	2, 2, 2	10, 10, 15	10, 10, 15
GCPs accuracy (xy) [pixels]	0.1	0.2	0.2

Table 2.1. Calibration and production flights details.

			CF1			CF2		
			E	N	h	E	N	h
Brown10	Fix	mean	56	-8	185	22	-5	1
		max	95	38	243	58	28	46
		RMS	60	21	186	29	12	17
	Lead	mean	2	1	48	4	-1	15
		max	40	28	96	38	23	60
		RMS	22	15	52	20	11	23
	APC	mean	0	1	46	15	-3	3
		max	37	24	92	51	26	48
		RMS	20	12	49	25	12	17
APCI	mean	2	0	31	7	-1	9	
	max	36	24	77	42	23	53	
	RMS	20	12	36	21	11	19	
Brown15	Fix	mean	55	-9	173	23	-6	2
		max	94	37	229	60	28	46
		RMS	59	20	174	31	12	17
	Lead	mean	3	0	43	5	-1	14
		max	41	25	90	40	22	57
		RMS	22	14	47	20	11	22
	APC	mean	1	0	45	17	-4	3
		max	36	24	90	53	26	47
		RMS	20	12	48	26	12	17
APCI	mean	3	0	29	8	-2	10	
	max	38	23	74	43	23	53	
	RMS	20	12	34	21	11	20	
Brown18	Fix	mean	55	-9	172	23	-6	2
		max	94	37	229	60	28	46
		RMS	59	20	173	30	12	17
	Lead	mean	3	0	42	5	-1	14
		max	41	26	90	40	22	57
		RMS	22	15	47	20	11	22
	APC	mean	1	0	44	17	-4	3
		max	36	24	90	53	26	47
		RMS	20	12	48	26	12	17
APCI	mean	3	0	28	8	-2	11	
	max	37	23	74	43	23	54	
	RMS	20	12	34	21	11	20	
OrthoPoly15	Fix	mean	-274	41	-850	32	-8	-6
		max	388	136	924	65	50	49
		RMS	284	80	852	36	24	24
	Lead	mean	-6	-4	-114	13	-6	-115
		max	78	98	182	48	54	158
		RMS	40	39	119	23	25	117
	APC	mean	11	-11	-255	23	-7	-12
		max	45	78	308	55	49	54
		RMS	24	33	256	28	24	25
APCI	mean	3	-5	-138	10	-4	-21	
	max	34	70	189	41	49	63	
	RMS	20	30	141	19	24	30	
OrthoPoly18	Fix	mean	Not applicable			32	-8	-6
		max	Not applicable			65	50	49
		RMS	Not applicable			36	24	24
	Lead	mean	Not applicable			15	-9	-145
		max	Not applicable			54	61	186
		RMS	Not applicable			26	29	146
	APC	mean	Not applicable			23	-7	-11
		max	Not applicable			56	49	53
		RMS	Not applicable			29	24	25
APCI	mean	Not applicable			10	-4	-20	
	max	Not applicable			41	48	62	
	RMS	Not applicable			19	24	30	

Table 2.2. Statistics of the checkpoints error for PF. Units are mm.



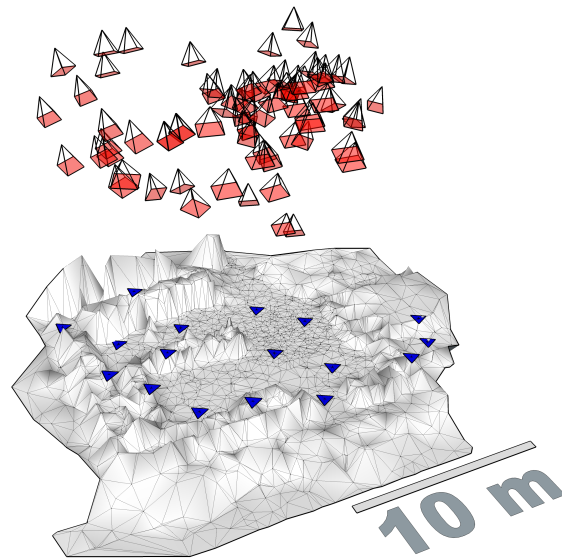


Figure 2.1. Calibration flight CF1

fixed wing UAV [130]. This payload is composed of: i) a custom, 20 Mpx camera for aerial photogrammetry developed by IGN, France [99] with Zeiss Biogon 35 mm lens, ii) a Gecko4Nav redundant IMU board with two Intersense NavChip MEMs IMUs [31], iii) a Topcon B110 GPS/GLONASS L1/L2 receiver.

We determine the IO of the camera during the bundle adjustment as described in Section 2.3. Two different strategies are considered:

1. CF1. A set of close-range images of a calibration field densely covered with mm-accurate GCPs (surveyed with theodolites) is oriented without the use of any aerial control (Indirect Sensor Orientation: InSO). The images are taken with converging geometry and from distances between 8 and 12 m. (Figure 2.1)
2. CF2. A block mapping mission is flown over a large area equipped with several cm-accurate GCPs (surveyed by post-processed GNSS). The images are oriented with aerial control (Integrated Sensor orientation: ISO with absolute position and relative orientation control). Two flight altitudes are considered, i.e., 120 and 150 m AGL. (Figure 2.2)

See Table 2.1 for a detailed description of the considered flights.

For both CF1 and CF2, we consider five choices for the camera models: i) Brown10, ii) Brown15, iii) Brown18, iv) OrthoPoly15 and v) OrthoPoly18. See Section 2.2 and Section 2.3.1.

We test the IO determined with each combination of calibration strategy and camera model

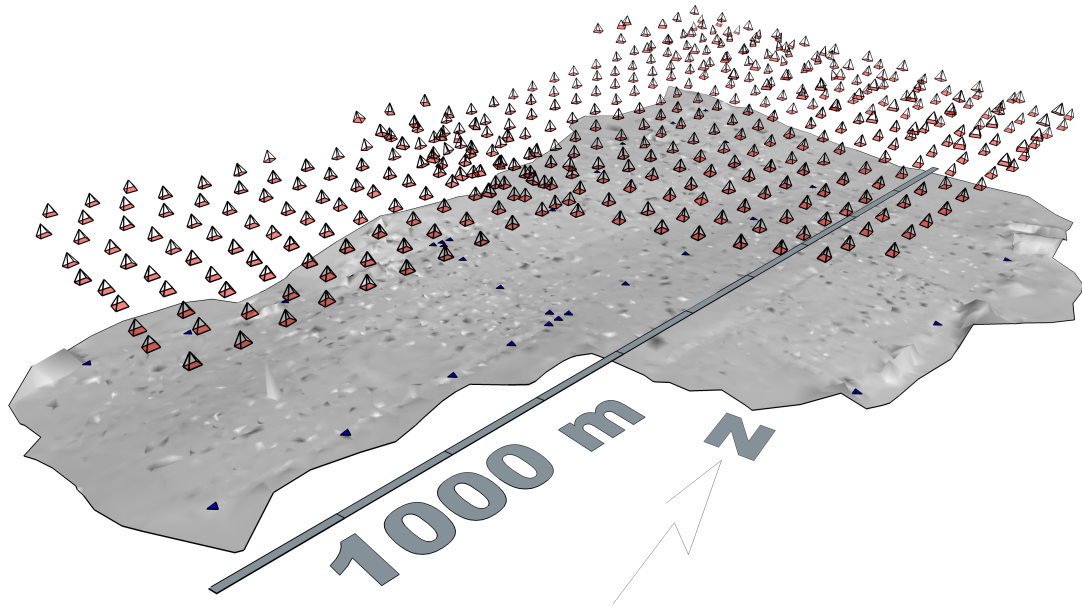


Figure 2.2. Calibration flight CF2

on a challenging 2 km N-S oriented corridor mapping mission (Figure 2.3), referred to as the Production Flight (PF) in the following, see again Table 2.1. This can be done by either keeping the IO fixed as it has been determined from the calibration flights (Fix, in the following), or by correcting for estimated camera instabilities in the bundle adjustment starting from the known initial values, according to any of the strategies presented in Section 2.4, i.e., Lead, APC, APCI.

For each of the 5 (camera models)  $\times$  4 (re-calibration strategies)  $\times$  2 (calibration setups) – 4 (because OrthoPoly18 cannot be applied in CF1 where no aerial control is available) = 36 experiments, we compute the statistics of the CPs error, in terms of mean, maximum and root mean squared error (RMS). See Table 2.2. To ease the interpretation, the cells are color coded according to the RMS, where white and red are associated with the lowest and highest values, respectively.

## 2.6 Discussion

### 2.6.1 Camera calibration strategy

The IO parameters determined in CF2 perform better than those determined in CF1. In fact, when the first ones are directly applied in PF with no correction, (Fix), they always yield better results.

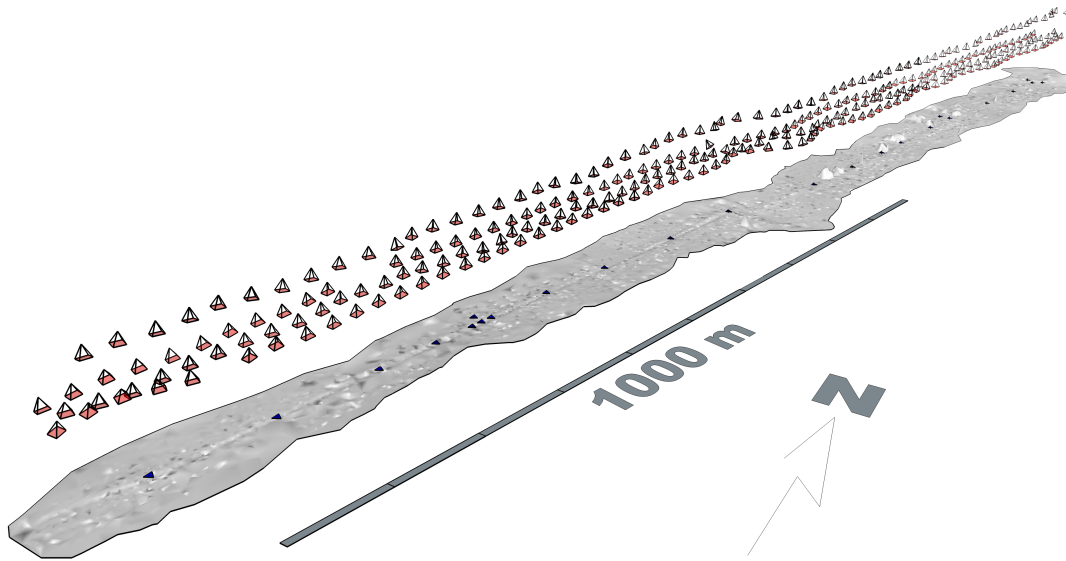


Figure 2.3. Test Flight PF

The Figures from 2.4 to 2.13 report the correlation matrix for the IO parameters as determined from CF1 and CF2<sup>1</sup>. The matrix for CF1 shows substantially higher correlations between parameters as compared to that of CF2, notably for the *lead* parameters. Note that almost complete correlation exists between the principal distance and the  $y$  component of the principal point ( $\rho_{f,pp_y} = -0.97$ ). This means that the values for the IO parameters could not be fully resolved, and that one could manipulate those to a relatively large extent (e.g., along the direction of the first eigenvector of the covariance matrix) and obtain similar values for the image observation residuals.

<sup>1</sup>The correlation matrices of the parameters given by the different IO model with the two calibration flight are organised as follow.

	CF1	CF2
Brown10	Figure 2.4	Figure 2.9
Brown15	Figure 2.5	Figure 2.10
Brown18	Figure 2.6	Figure 2.11
Poly15	Figure 2.7	Figure 2.12
Poly18	Figure 2.8	Figure 2.13

	f	ppx	ppy	R1	R2	R3	T1	T2	B1	B2
1.26e+00	1.00	-0.28	-0.97	0.28	-0.02	0.01	-0.65	-0.12	0.73	-0.20
1.46e-01	-0.28	1.00	0.28	-0.11	0.03	-0.03	0.21	0.87	-0.22	0.52
6.56e-01	-0.97	0.28	1.00	-0.29	0.04	-0.03	0.68	0.11	-0.81	0.23
1.88e-04	0.28	-0.11	-0.29	1.00	-0.90	0.85	-0.41	-0.08	0.07	-0.02
1.32e-03	-0.02	0.03	0.04	-0.90	1.00	-0.99	0.07	0.01	-0.02	0.01
3.03e-03	0.01	-0.03	-0.03	0.85	-0.99	1.00	-0.06	-0.01	0.01	-0.01
1.78e-05	-0.65	0.21	0.68	-0.41	0.07	-0.06	1.00	0.15	-0.18	0.05
7.69e-06	-0.12	0.87	0.11	-0.08	0.01	-0.01	0.15	1.00	-0.03	0.22
6.71e-02	0.73	-0.22	-0.81	0.07	-0.02	0.01	-0.18	-0.03	1.00	-0.29
2.26e-02	-0.20	0.52	0.23	-0.02	0.01	-0.01	0.05	0.22	-0.29	1.00

Figure 2.4. IO correlation matrix for Brown10 in CF1. Black to white colors highlight low and high correlation coefficients, respectively.

We argue that the reason for this is that aerial control in CF2 brings extremely valuable information to de-correlate IO parameters. This is an argument for the use of aerial control in camera calibration flights.

	f	ppx	ppy	R1	R2	R3	R4	R5	R6	T1	T2	T3	T4	B1	B2
1.25e+00	1.00	-0.26	-0.96	-0.06	0.11	-0.11	0.11	-0.10	0.10	-0.52	-0.06	-0.01	0.03	0.73	-0.20
1.47e-01	-0.26	1.00	0.25	-0.01	-0.01	0.02	-0.02	0.02	-0.02	0.02	0.66	0.16	-0.11	-0.21	0.51
6.48e-01	-0.96	0.25	1.00	-0.00	-0.05	0.05	-0.05	0.05	-0.05	0.57	0.00	-0.04	0.02	-0.81	0.23
1.14e-03	-0.06	-0.01	-0.00	1.00	-0.98	0.94	-0.89	0.85	-0.81	-0.03	-0.00	0.01	-0.01	-0.01	0.01
2.87e-02	0.11	-0.01	-0.05	-0.98	1.00	-0.99	0.96	-0.93	0.90	-0.02	-0.00	-0.01	0.02	0.03	-0.02
3.32e-01	-0.11	0.02	0.05	0.94	-0.99	1.00	-0.99	0.98	-0.95	0.02	0.01	0.01	-0.02	-0.04	0.03
1.93e+00	0.11	-0.02	-0.05	-0.89	0.96	-0.99	1.00	-0.99	0.98	-0.02	-0.01	-0.02	0.02	0.04	-0.04
5.44e+00	-0.10	0.02	0.05	0.85	-0.93	0.98	-0.99	1.00	-1.00	0.02	0.01	0.02	-0.02	-0.04	0.04
5.95e+00	0.10	-0.02	-0.05	-0.81	0.90	-0.95	0.98	-1.00	1.00	-0.02	-0.01	-0.02	0.03	0.04	-0.04
2.05e-05	-0.52	0.02	0.57	-0.03	-0.02	0.02	-0.02	0.02	-0.02	1.00	-0.42	-0.54	0.47	-0.14	0.04
1.24e-05	-0.06	0.66	0.00	-0.00	-0.00	0.01	-0.01	0.01	-0.01	-0.42	1.00	0.76	-0.67	-0.01	0.13
1.06e-01	-0.01	0.16	-0.04	0.01	-0.01	0.01	-0.02	0.02	-0.02	-0.54	0.76	1.00	-0.97	-0.01	-0.02
2.81e-01	0.03	-0.11	0.02	-0.01	0.02	-0.02	0.02	-0.02	0.03	0.47	-0.67	-0.97	1.00	0.01	0.03
6.67e-02	0.73	-0.21	-0.81	-0.01	0.03	-0.04	0.04	-0.04	0.04	-0.14	-0.01	-0.01	0.01	1.00	-0.29
2.26e-02	-0.20	0.51	0.23	0.01	-0.02	0.03	-0.04	0.04	-0.04	0.04	0.13	-0.02	0.03	-0.29	1.00

Figure 2.5. IO correlation matrix for Brown15 in CF1. Black to white colors highlight low and high correlation coefficients, respectively.

## Chapter 2. Toward Camera Calibration models and methods

	f	ppx	ppy	R1	R2	R3	R4	R5	R6	R7	R8	T1	T2	T3	T4	T5	B1	B2
1.26e+00	1.00	-0.26	-0.96	-0.08	0.08	-0.07	0.05	-0.04	0.03	-0.02	0.02	-0.39	-0.04	-0.00	0.00	0.00	0.73	-0.20
1.47e-01	-0.26	1.00	0.24	0.00	-0.01	0.01	-0.01	0.01	-0.01	0.01	-0.00	-0.10	0.54	0.21	-0.19	0.18	-0.20	0.51
6.48e-01	-0.96	0.24	1.00	-0.02	0.01	-0.02	0.03	-0.03	0.04	-0.04	0.05	0.46	-0.03	-0.06	0.05	-0.05	-0.81	0.23
2.78e-03	-0.08	0.00	-0.02	1.00	-0.98	0.94	-0.90	0.86	-0.82	0.79	-0.75	-0.02	0.01	0.01	-0.01	0.01	0.00	-0.00
1.13e-01	0.08	-0.01	0.01	-0.98	1.00	-0.99	0.97	-0.94	0.91	-0.88	0.85	0.01	-0.01	-0.00	0.00	-0.00	-0.00	0.00
2.23e+00	-0.07	0.01	-0.02	0.94	-0.99	1.00	-0.99	0.98	-0.96	0.93	-0.91	-0.01	0.00	-0.00	0.00	-0.00	0.01	-0.00
2.43e+01	0.05	-0.01	0.03	-0.90	0.97	-0.99	1.00	-1.00	0.98	-0.97	0.95	0.02	0.00	0.01	-0.01	0.01	-0.01	0.01
1.53e+02	-0.04	0.01	-0.03	0.86	-0.94	0.98	-1.00	1.00	-1.00	0.99	-0.98	-0.02	-0.00	-0.01	0.01	-0.01	0.02	-0.01
5.58e+02	0.03	-0.01	0.04	-0.82	0.91	-0.96	0.98	-1.00	1.00	-1.00	0.99	0.02	0.00	0.01	-0.01	0.01	-0.02	0.01
1.08e+03	-0.02	0.01	-0.04	0.79	-0.88	0.93	-0.97	0.99	-1.00	1.00	-1.00	-0.02	-0.01	-0.01	0.01	-0.01	0.02	-0.01
8.70e+02	0.02	-0.00	0.05	-0.75	0.85	-0.91	0.95	-0.98	0.99	-1.00	1.00	0.02	0.01	0.02	-0.02	0.01	-0.02	0.01
2.45e-05	-0.39	-0.10	0.46	-0.02	0.01	-0.01	0.02	-0.02	0.02	-0.02	0.02	1.00	-0.72	-0.76	0.71	-0.67	-0.11	0.03
1.74e-05	-0.04	0.54	-0.03	0.01	-0.01	0.00	0.00	-0.00	0.00	-0.01	0.01	-0.72	1.00	0.89	-0.83	0.78	-0.00	0.08
4.03e-01	-0.00	0.21	-0.06	0.01	-0.00	-0.00	0.01	-0.01	0.01	-0.01	0.02	-0.76	0.89	1.00	-0.98	0.95	0.00	0.00
2.45e+00	0.00	-0.19	0.05	-0.01	0.00	0.00	-0.01	0.01	-0.01	0.01	-0.02	0.71	-0.83	-0.98	1.00	-0.99	-0.00	-0.00
4.82e+00	0.00	0.18	-0.05	0.01	-0.00	-0.00	0.01	-0.01	0.01	-0.01	0.01	-0.67	0.78	0.95	-0.99	1.00	0.01	0.01
6.67e-02	0.73	-0.20	-0.81	0.00	-0.00	0.01	-0.01	0.02	-0.02	0.02	-0.02	-0.11	-0.00	0.00	-0.00	0.01	1.00	-0.29
2.25e-02	-0.20	0.51	0.23	-0.00	0.00	-0.00	0.01	-0.01	0.01	-0.01	0.01	0.03	0.08	0.00	-0.00	0.01	-0.29	1.00

Figure 2.6. IO correlation matrix for Brown18 in CF1. Black to white colors highlight low and high correlation coefficients, respectively.

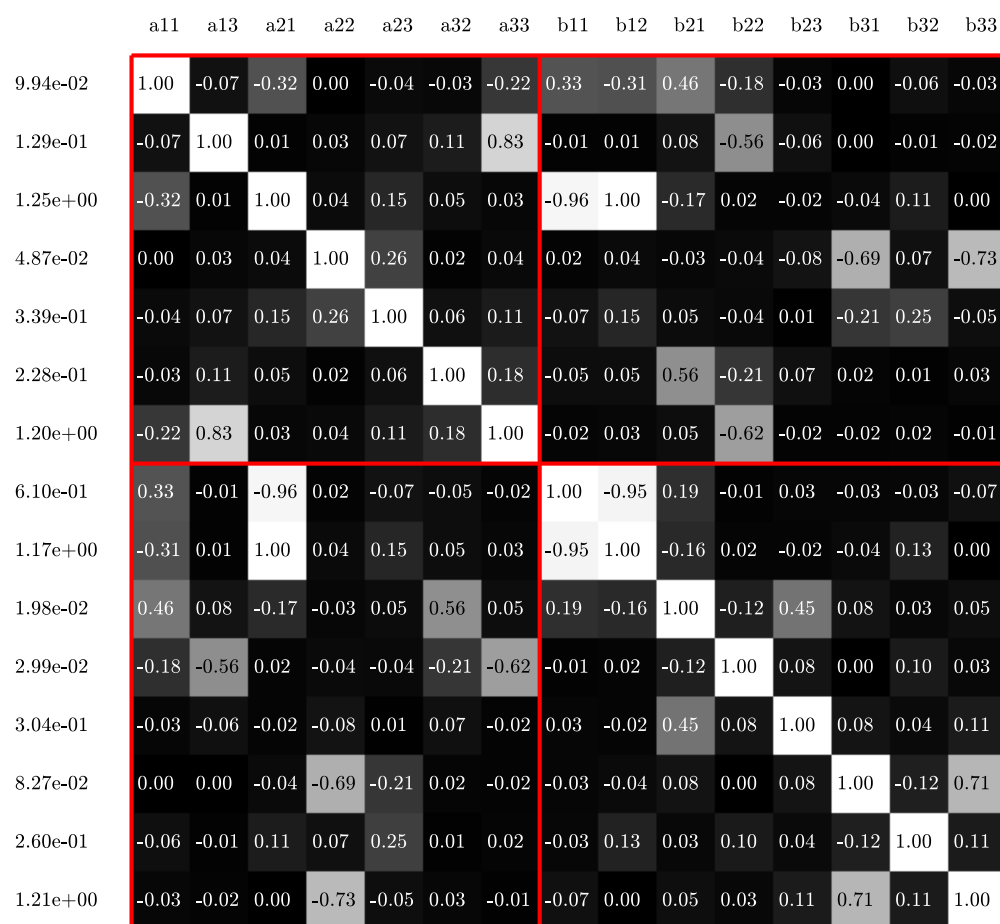


Figure 2.7. IO correlation matrix for Poly15 in CF1. Black to white colors highlight low and high correlation coefficients, respectively.

## Chapter 2. Toward Camera Calibration models and methods

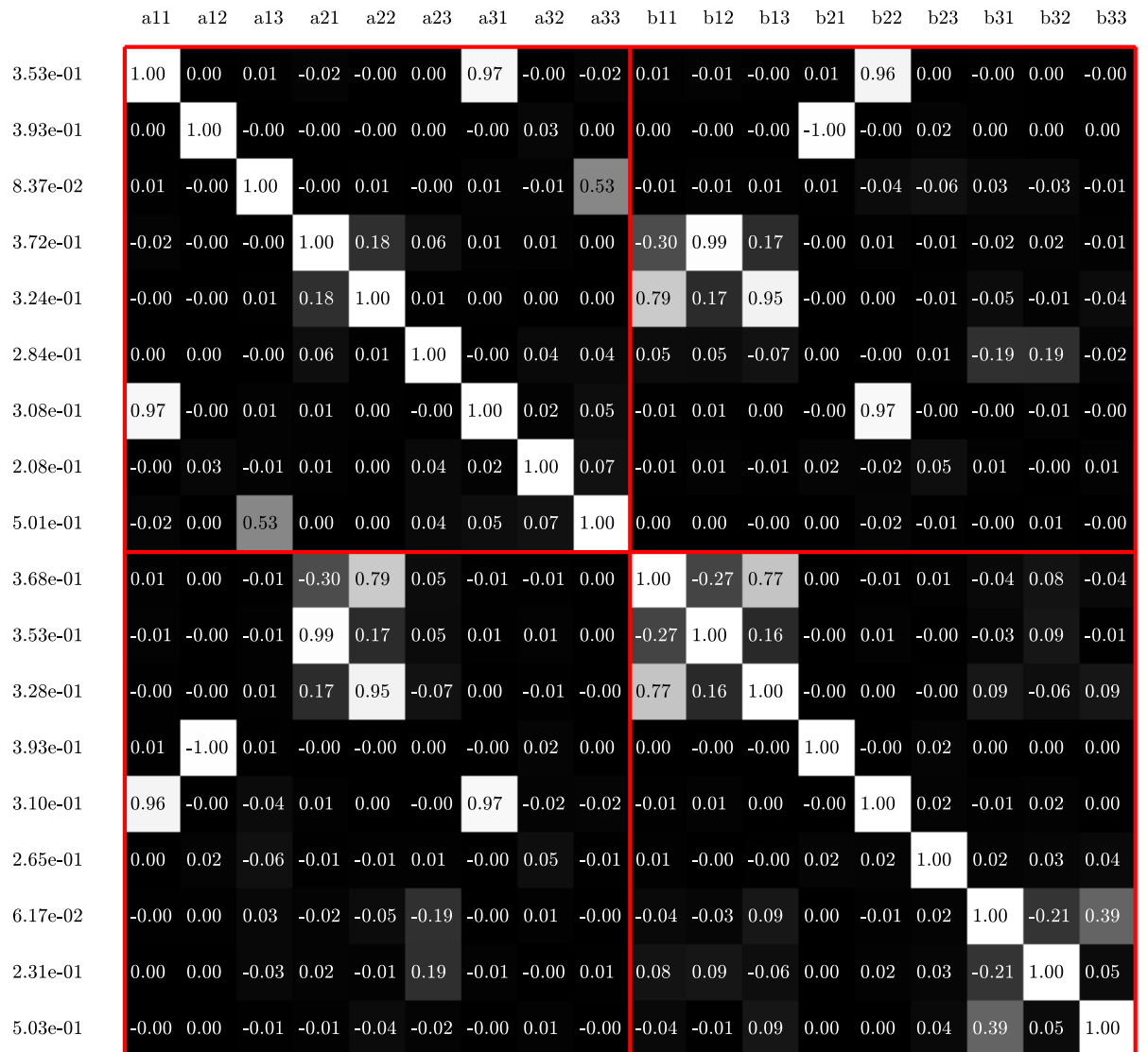


Figure 2.8. IO correlation matrix for Poly18 in CF1. Black to white colors highlight low and high correlation coefficients, respectively.



	f	ppx	ppy	R1	R2	R3	T1	T2	B1	B2
2.08e-01	1.00	-0.01	0.10	-0.38	0.35	-0.32	-0.00	-0.00	-0.09	0.01
2.34e-01	-0.01	1.00	-0.02	0.00	-0.01	0.01	-0.01	0.90	0.02	-0.02
1.86e-01	0.10	-0.02	1.00	0.01	-0.00	0.00	0.75	-0.01	0.01	0.01
3.85e-04	-0.38	0.00	0.01	1.00	-0.97	0.93	0.03	0.01	0.03	0.01
2.86e-03	0.35	-0.01	-0.00	-0.97	1.00	-0.99	-0.01	-0.01	-0.06	-0.01
6.35e-03	-0.32	0.01	0.00	0.93	-0.99	1.00	0.01	0.01	0.06	0.01
1.12e-05	-0.00	-0.01	0.75	0.03	-0.01	0.01	1.00	0.00	-0.13	0.01
1.48e-05	-0.00	0.90	-0.01	0.01	-0.01	0.01	0.00	1.00	0.00	0.06
3.56e-02	-0.09	0.02	0.01	0.03	-0.06	0.06	-0.13	0.00	1.00	0.01
3.54e-02	0.01	-0.02	0.01	0.01	-0.01	0.01	0.01	0.06	0.01	1.00

Figure 2.9. IO correlation matrix for Brown10 in CF2. Black to white colors highlight low and high correlation coefficients, respectively.

## Chapter 2. Toward Camera Calibration models and methods

	f	ppx	ppy	R1	R2	R3	R4	R5	R6	T1	T2	T3	T4	B1	B2
2.89e-01	1.00	-0.00	0.06	-0.71	0.65	-0.59	0.55	-0.51	0.48	-0.00	0.00	-0.00	0.01	-0.04	0.01
2.42e-01	-0.00	1.00	-0.17	-0.00	-0.00	0.00	-0.00	0.00	-0.00	-0.30	0.68	0.26	-0.20	0.03	-0.02
2.00e-01	0.06	-0.17	1.00	0.00	-0.00	-0.00	0.00	-0.00	0.00	0.62	-0.37	-0.35	0.28	0.01	-0.00
2.68e-03	-0.71	-0.00	0.00	1.00	-0.98	0.94	-0.89	0.85	-0.81	-0.00	0.01	0.02	-0.03	-0.03	-0.01
6.54e-02	0.65	-0.00	-0.00	-0.98	1.00	-0.99	0.96	-0.94	0.90	0.02	-0.02	-0.03	0.04	0.03	0.01
7.37e-01	-0.59	0.00	-0.00	0.94	-0.99	1.00	-0.99	0.98	-0.96	-0.02	0.02	0.04	-0.05	-0.03	-0.01
4.18e+00	0.55	-0.00	0.00	-0.89	0.96	-0.99	1.00	-1.00	0.98	0.03	-0.03	-0.05	0.06	0.03	0.01
1.16e+01	-0.51	0.00	-0.00	0.85	-0.94	0.98	-1.00	1.00	-1.00	-0.04	0.03	0.06	-0.07	-0.02	-0.01
1.24e+01	0.48	-0.00	0.00	-0.81	0.90	-0.96	0.98	-1.00	1.00	0.04	-0.04	-0.07	0.08	0.02	0.01
2.68e-05	-0.00	-0.30	0.62	-0.00	0.02	-0.02	0.03	-0.04	0.04	1.00	-0.82	-0.89	0.80	-0.06	-0.02
2.67e-05	0.00	0.68	-0.37	0.01	-0.02	0.02	-0.03	0.03	-0.04	-0.82	1.00	0.82	-0.73	0.02	0.06
2.49e-01	-0.00	0.26	-0.35	0.02	-0.03	0.04	-0.05	0.06	-0.07	-0.89	0.82	1.00	-0.97	0.01	0.03
6.39e-01	0.01	-0.20	0.28	-0.03	0.04	-0.05	0.06	-0.07	0.08	0.80	-0.73	-0.97	1.00	-0.01	-0.03
3.57e-02	-0.04	0.03	0.01	-0.03	0.03	-0.03	0.03	-0.02	0.02	-0.06	0.02	0.01	-0.01	1.00	0.01
3.54e-02	0.01	-0.02	-0.00	-0.01	0.01	-0.01	0.01	-0.01	0.01	-0.02	0.06	0.03	-0.03	0.01	1.00

Figure 2.10. IO correlation matrix for Brown15 in CF2. Black to white colors highlight low and high correlation coefficients, respectively.

	f	ppx	ppy	R1	R2	R3	R4	R5	R6	R7	R8	T1	T2	T3	T4	T5	B1	B2
3.86e-01	1.00	0.00	0.05	-0.82	0.75	-0.69	0.64	-0.60	0.56	-0.53	0.50	-0.01	0.01	0.01	-0.01	0.01	-0.04	0.01
2.46e-01	0.00	1.00	-0.20	-0.01	0.01	-0.01	0.01	-0.01	0.01	-0.01	0.01	-0.33	0.59	0.26	-0.21	0.18	0.03	-0.02
2.04e-01	0.05	-0.20	1.00	-0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	0.00	0.54	-0.38	-0.30	0.23	-0.19	0.01	0.00
6.56e-03	-0.82	-0.01	-0.00	1.00	-0.98	0.94	-0.90	0.86	-0.82	0.79	-0.76	0.01	-0.01	-0.01	0.01	-0.02	-0.00	0.00
2.62e-01	0.75	0.01	0.00	-0.98	1.00	-0.99	0.97	-0.94	0.91	-0.88	0.85	-0.01	0.01	0.01	-0.02	0.02	-0.00	-0.00
5.13e+00	-0.69	-0.01	-0.00	0.94	-0.99	1.00	-0.99	0.98	-0.96	0.94	-0.91	0.01	-0.01	-0.02	0.02	-0.03	0.01	0.00
5.55e+01	0.64	0.01	0.00	-0.90	0.97	-0.99	1.00	-1.00	0.98	-0.97	0.95	-0.01	0.01	0.02	-0.03	0.03	-0.01	-0.00
3.48e+02	-0.60	-0.01	-0.00	0.86	-0.94	0.98	-1.00	1.00	-1.00	0.99	-0.98	0.02	-0.02	-0.02	0.03	-0.04	0.01	0.00
1.26e+03	0.56	0.01	0.00	-0.82	0.91	-0.96	0.98	-1.00	1.00	-1.00	0.99	-0.02	0.02	0.03	-0.04	0.04	-0.01	-0.00
2.43e+03	-0.53	-0.01	-0.00	0.79	-0.88	0.94	-0.97	0.99	-1.00	1.00	-1.00	0.02	-0.02	-0.03	0.04	-0.05	0.01	0.00
1.94e+03	0.50	0.01	0.00	-0.76	0.85	-0.91	0.95	-0.98	0.99	-1.00	1.00	-0.03	0.03	0.04	-0.05	0.06	-0.01	-0.00
4.19e-05	-0.01	-0.33	0.54	0.01	-0.01	0.01	-0.01	0.02	-0.02	0.02	-0.03	1.00	-0.92	-0.91	0.83	-0.76	-0.04	-0.01
3.99e-05	0.01	0.59	-0.38	-0.01	0.01	-0.01	0.01	-0.02	0.02	-0.02	0.03	-0.92	1.00	0.88	-0.80	0.73	0.01	0.03
6.88e-01	0.01	0.26	-0.30	-0.01	0.01	-0.02	0.02	-0.02	0.03	-0.03	0.04	-0.91	0.88	1.00	-0.98	0.94	0.01	0.00
3.99e+00	-0.01	-0.21	0.23	0.01	-0.02	0.02	-0.03	0.03	-0.04	0.04	-0.05	0.83	-0.80	-0.98	1.00	-0.99	-0.00	0.00
7.62e+00	0.01	0.18	-0.19	-0.02	0.02	-0.03	0.03	-0.04	0.04	-0.05	0.06	-0.76	0.73	0.94	-0.99	1.00	0.00	-0.01
3.57e-02	-0.04	0.03	0.01	-0.00	-0.00	0.01	-0.01	0.01	-0.01	0.01	-0.01	-0.04	0.01	0.01	-0.00	0.00	1.00	0.01
3.54e-02	0.01	-0.02	0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	-0.01	0.03	0.00	0.00	-0.01	0.01	1.00

Figure 2.11. IO correlation matrix for Brown18 in CF2. Black to white colors highlight low and high correlation coefficients, respectively.

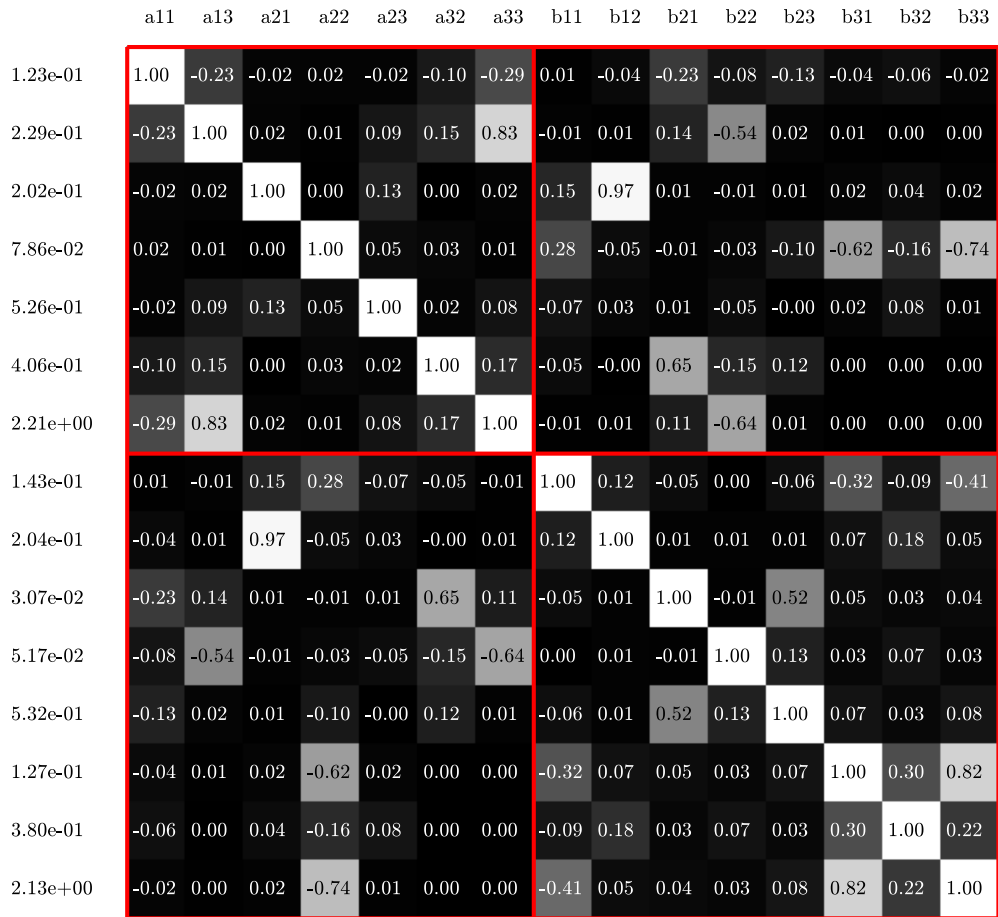


Figure 2.12. IO correlation matrix for Poly15 in CF2. Black to white colors highlight low and high correlation coefficients, respectively.

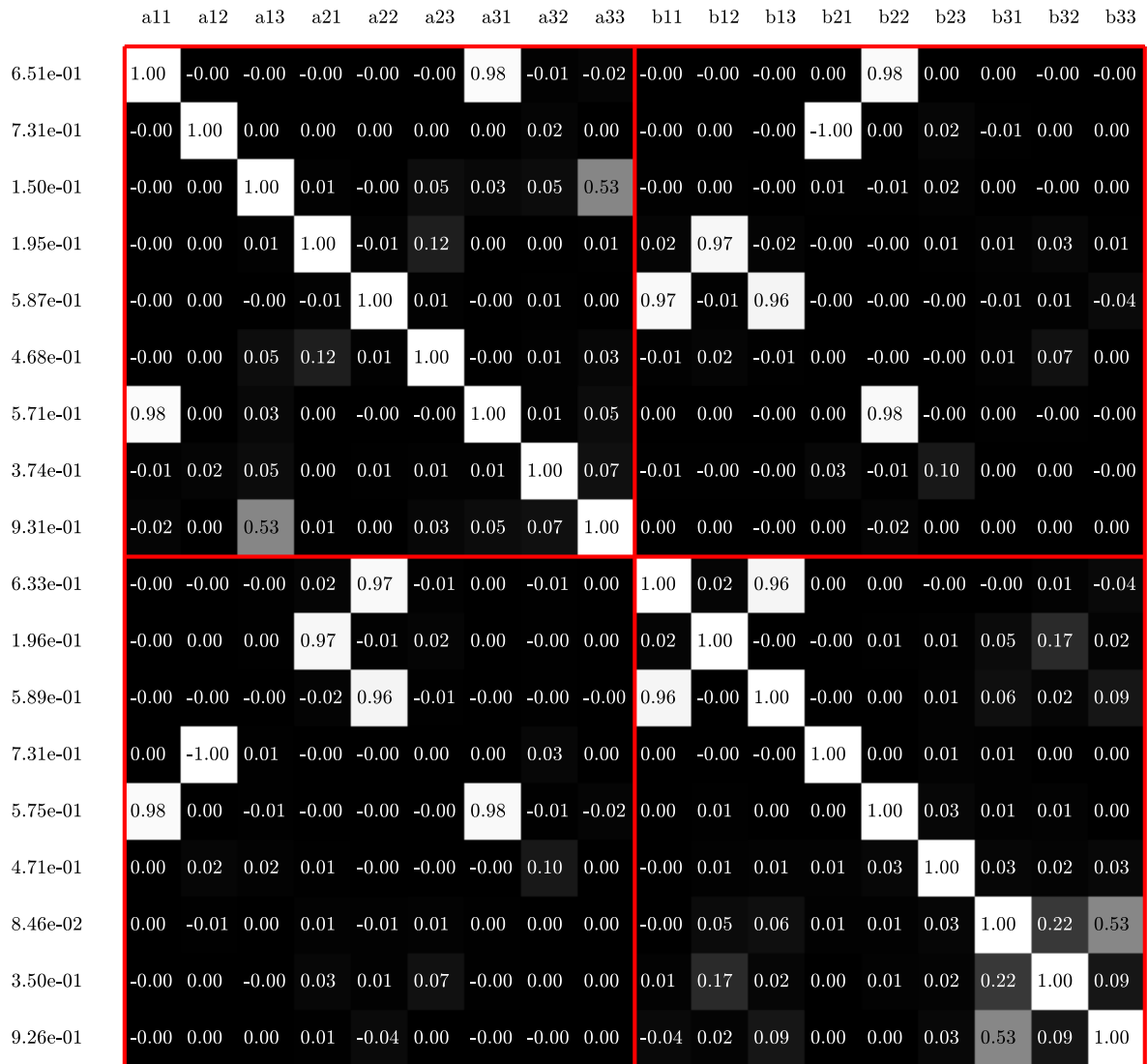


Figure 2.13. IO correlation matrix for Poly18 in CF2. Black to white colors highlight low and high correlation coefficients, respectively.

## 2.6.2 Camera models

We observe that the physical models BrownXX outperform OrthoPolyXX in the considered case. Indeed, some aspects of the distortions specific to the considered lens system could not be compensated using any of the OrthoPolyXX models.

To show this, we attempt to compare the departures from collinearity as observed from tie-points and as implied by the estimated IO model. For each tie-point  $P_i$  we define a vector  $\vec{v}_{P_i}$  as the difference between its image projection implied by the pinhole camera model and the actual tie-point image coordinates:

$$\vec{v}_{P_i} = f\pi(P_i) + \begin{bmatrix} pp_x \\ pp_y \end{bmatrix} - z_{P_i}, \quad (2.9)$$

where  $P_i$ ,  $f$ ,  $pp_x$  and  $pp_y$  are taken from the output of the bundle adjustment for PF with `OrthoPoly18`. A vector  $\vec{v}_M$ , i.e., the corrections implied by a given IO model, can be defined analogously for any point of the image plane.

$\vec{v}$  can be decomposed in two components, radial, in the direction of the principal point, and orthoradial<sup>2</sup> The orthoradial direction (green vector of Figure 2.14) is different from the tangential distortion (see the vector field of tangential coefficients displayed on the two first lines of page 28), i.e., orthogonal to the first, see Figure 2.14.

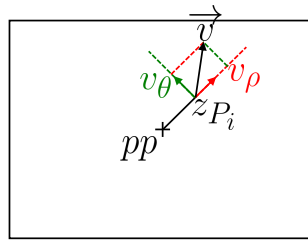


Figure 2.14. Decomposition of the departure from collinearity vector,  $\vec{v}_{P_i}$  into its radial ( $v_\rho$ ) and orthoradial ( $v_\theta$ ) components for a tie-point observed at image coordinates  $z_{P_i}$ .

The `OrthoPolyXX` are not designed along a radial structure. For representing the radial distortions the same way for `BrownXX` and `OrthoPolyXX`, we have chosen to plot the component of  $\vec{v}$  along the diagonal for the tie-point observation close to such diagonal (Figure 2.15), for all the considered IO models. Thus, Figure 2.16 presents the distortions all along the diagonal. The middle of the plot (of abscissa 0) corresponds to the principal point. It is evident that `BrownXX` fits the observations better than `OrthoPolyXX`. Note the overfit in `Brown18` in the extremities of Figure 2.16. In Figure 2.17 and Figure 2.18 we have plotted the radial and the orthoradial components of  $\vec{v}_M$ , respectively, as a function of the image coordinates. We can observe that no clear radially symmetric correction is implied by `OrthoPolyXX`, see especially Figure 2.17 and 2.18.

These results may hold for the specific lens at hand, a high quality Zeiss Biogon, and may not

<sup>2</sup>The orthoradial direction is defined for a given 2D point as perpendicular to the radial direction, and pointing in the counter-clockwise direction with respect to the origin (here the principal point).

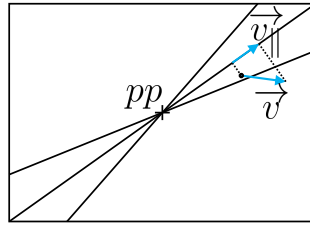


Figure 2.15. Distortions of Figure 2.16 corresponds to points close to the diagonal (inside a tolerance) whose distortion  $\vec{v}$  have been projected into  $\vec{v}_{\parallel}$  on the diagonal.

be generally applicable. However, the numerical models based on Ebner's self-calibration functions were developed in the seventies when aerial photogrammetry was performed by means of film cameras and additional corrections due to film out of plane deformations were important. This is why numerically inspired models were developed. We suggest that this may no longer be appropriate with modern digital cameras. In fact, with digital cameras, even consumer grade, the distortion effects are dominated by a component that is radially symmetrical. For this, no explicit term is present in Ebner's functions.

### 2.6.3 Re-calibration

With respect to the re-calibration strategy, we argue the following:

1. In general, it is not optimal to directly employ the IO as determined in calibration flights (Fix), at least with consumer grade cameras, since it is well known that the leading IO parameters may slightly but significantly change due to camera mechanical instability.
2. The well known strategy of re-estimating the leading IO parameters (Lead) improves results with respect to Fix in all the considered experiments and yields some of the best results.
3. APC and APCI further improve upon Lead and such improvement is more marked when the available IO parameters from calibration are correlated, as in CF1. Unfortunately, these strategies are not implemented in commercial aerial photogrammetry software.
4. It is not clear whether APCI is better than APC. We note that APCI has one tuning parameter which affects how much the diagonal components of  $\Sigma_{\Theta}$  corresponding to the leading IO parameters are inflated. This is related to how much those are expected to change with time and is an empirical parameter that relies upon user experience and the specific camera being utilized.

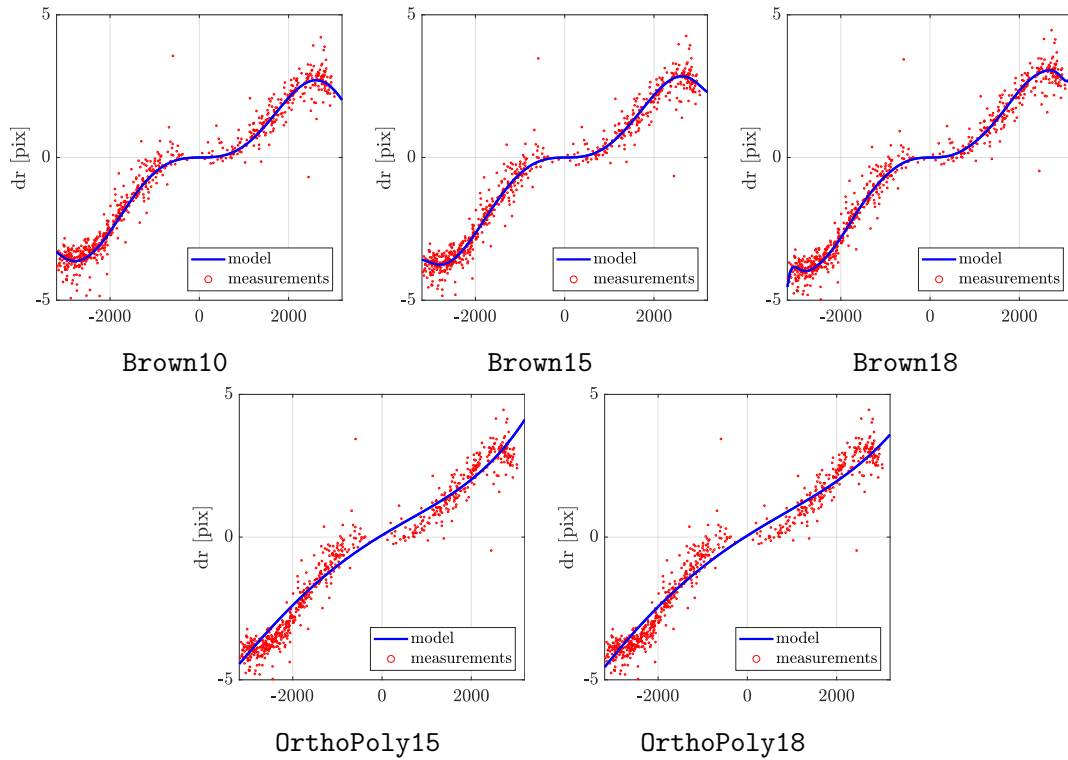


Figure 2.16. Radial component of the observed departures from collinearity,  $\vec{v}_{P_i}$  (red dots), and as implied by the estimated IO model,  $\vec{v}_M$  (blue curve), for points on the diagonal of the image plane.

Finally, we note that no matter what re-calibration strategy we employ, better results are obtained when IO from CF2 are employed. This is related to the fact that the geometry of corridor mapping missions enables correction for imperfect IO, even when the full *a priori* uncertainty  $\Sigma_\Theta$  is employed.

## 2.7 Conclusions

In this work we have investigated different calibration strategies and IO models targeting corridor mapping applications. In this scenario, *a priori* knowledge of the IO is essential. We've seen that certain choices may lead to poor results and should be avoided, while others allowed us to obtain sub-GSD residuals at checkpoints using no GCP.

Our results can be summarized in the following guidelines useful to UAV practitioners who are targeting delicate mapping missions in which redundancy and ground control are limited:

1. prefer calibration flights in similar configurations (e.g., altitude) with respect to produc-



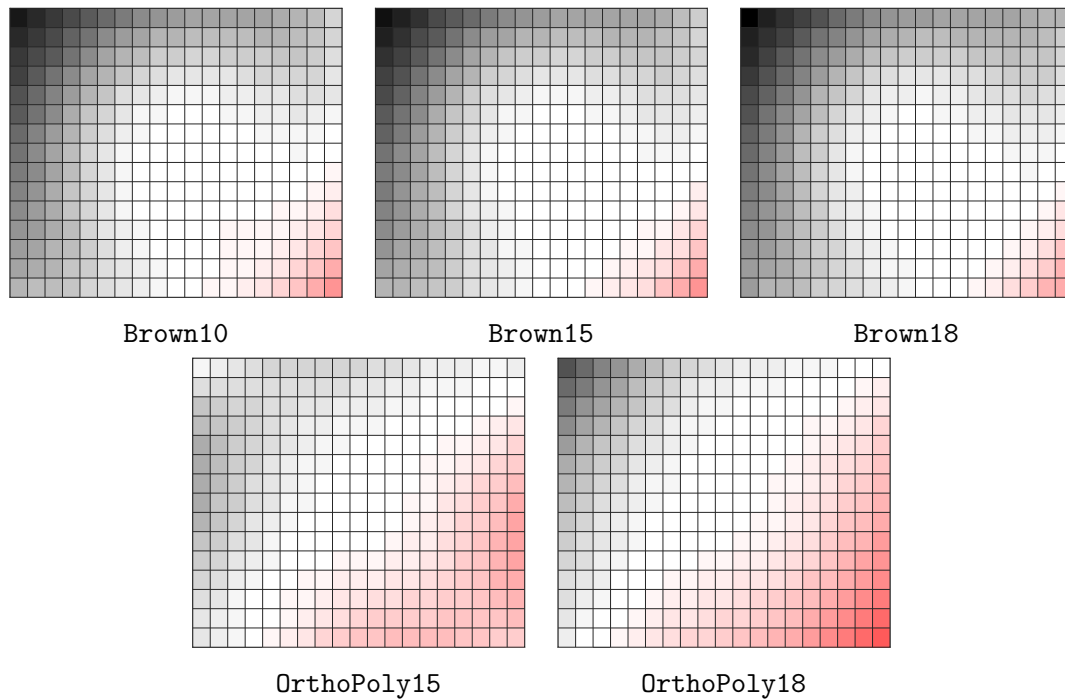


Figure 2.17. Radial component of  $\vec{v}_M$  as a function of image coordinates. Black corresponds to +10 pixels and red to -10 pixels. Note the negative distortions for the lower-right corner of the images while the distortion on the other corners is positive. This leads to relatively high values for the tangential distortions.

tion flights,

2. use calibration at scale where addition of aerial control (in terms of precision) is significant<sup>3</sup>,
3. allow some form of re-estimation of the camera calibration during the bundle adjustment of production flights (unfortunately only Lead is available in commercial bundle adjustment software),
4. strive to reduce the number of calibration parameters, as long as they are sufficient for modeling the lens at hand,
5. prefer the Brown family model over Ebner's when the effect of the radial and tangential distortions induced by the lens characteristics are predominant over the potential sensors deformations.

<sup>3</sup>integrated sensor orientation implies using aerial control

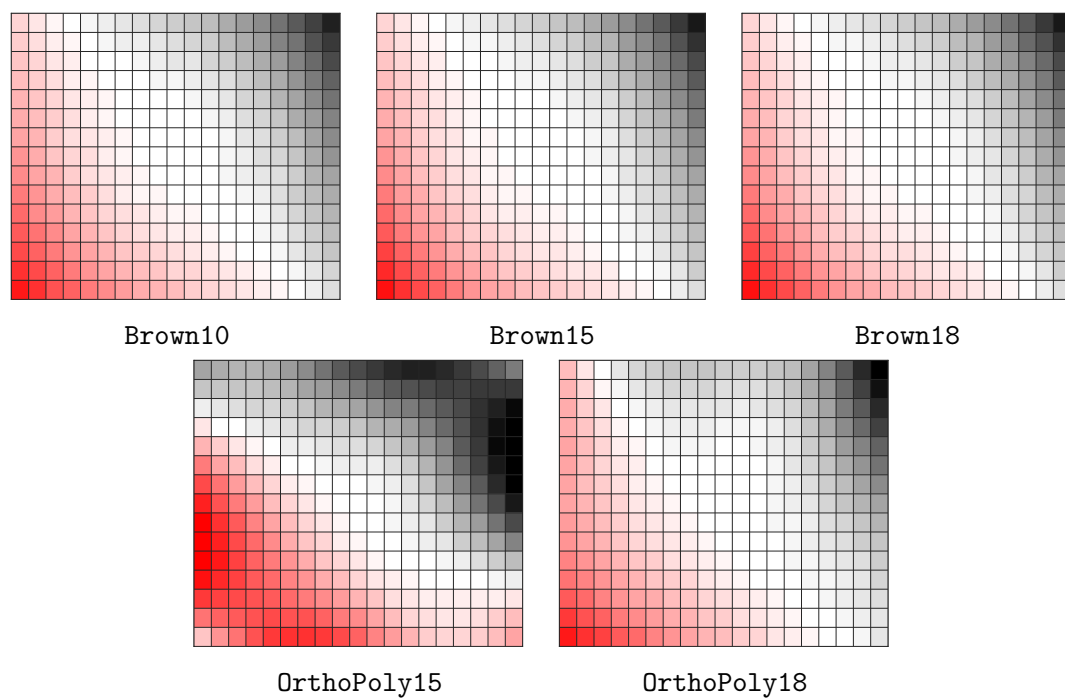


Figure 2.18. Orthoradial component of  $\vec{v}_M$  as a function of image coordinates. Black corresponds to +2.5 pixels and red to -2.5 pixels

CF1				CF2					
<b>Brown</b>				<b>Brown</b>					
	<b>10</b>	<b>15</b>	<b>18</b>		<b>10</b>	<b>15</b>	<b>18</b>		
<i>f</i>	4466.14	4480.92	4461.34	<i>f</i>	27017.67	19423.34	14510.04		
<i>pp<sub>x</sub></i>	17722.62	17662.23	17612.82	<i>pp<sub>x</sub></i>	11050.68	10720.81	10519.72		
<i>pp<sub>y</sub></i>	3004.72	3043.28	3042.51	<i>pp<sub>y</sub></i>	10627.55	9836.34	9644.59		
<i>R<sub>1</sub></i>	87.26	19.34	9.32	<i>R<sub>1</sub></i>	41.59	7.00	4.02		
<i>R<sub>2</sub></i>	45.75	6.11	3.13	<i>R<sub>2</sub></i>	22.04	1.85	1.44		
<i>R<sub>3</sub></i>	19.81	3.19	2.18	<i>R<sub>3</sub></i>	10.49	0.79	0.86		
<i>R<sub>4</sub></i>		2.12	1.96	<i>R<sub>4</sub></i>		0.53	0.54		
<i>R<sub>5</sub></i>		1.37	1.91	<i>R<sub>5</sub></i>		0.38	0.29		
<i>R<sub>6</sub></i>		0.75	1.91	<i>R<sub>6</sub></i>		0.24	0.06		
<i>R<sub>7</sub></i>			1.93	<i>R<sub>7</sub></i>			0.13		
<i>R<sub>8</sub></i>			1.95	<i>R<sub>8</sub></i>			0.30		
<i>T<sub>1</sub></i>	57.22	47.14	34.89	<i>T<sub>1</sub></i>	92.49	33.89	22.08		
<i>T<sub>2</sub></i>	112.15	65.00	40.86	<i>T<sub>2</sub></i>	63.74	30.61	20.94		
<i>T<sub>3</sub></i>		0.29	6.11	<i>T<sub>3</sub></i>		2.71	0.48		
<i>T<sub>4</sub></i>		3.43	5.98	<i>T<sub>4</sub></i>		1.07	0.33		
<i>T<sub>5</sub></i>			6.49	<i>T<sub>5</sub></i>			0.47		
<i>B<sub>1</sub></i>	3.55	3.66	3.71	<i>B<sub>1</sub></i>	1.01	1.01	1.03		
<i>B<sub>2</sub></i>	23.75	24.21	24.20	<i>B<sub>2</sub></i>	5.64	5.70	5.71		
<b>Orthogonal polynomials</b>				<b>Orthogonal polynomials</b>					
	<b>15</b>		<b>18</b>			<b>15</b>		<b>18</b>	
<i>a<sub>11</sub></i>	26177.95		<i>a<sub>11</sub></i>	7357.17	<i>a<sub>11</sub></i>	21058.64		<i>a<sub>11</sub></i>	3998.91
<i>a<sub>13</sub></i>	32.76		<i>a<sub>12</sub></i>	9.25	<i>a<sub>13</sub></i>	23.40		<i>a<sub>12</sub></i>	5.61
<i>a<sub>21</sub></i>	4462.58		<i>a<sub>13</sub></i>	50.39	<i>a<sub>21</sub></i>	27724.69		<i>a<sub>13</sub></i>	35.77
<i>a<sub>22</sub></i>	75.05		<i>a<sub>21</sub></i>	14948.14	<i>a<sub>22</sub></i>	26.36		<i>a<sub>21</sub></i>	28810.06
<i>a<sub>23</sub></i>	25.95		<i>a<sub>22</sub></i>	17.24	<i>a<sub>23</sub></i>	15.05		<i>a<sub>22</sub></i>	6.10
<i>a<sub>32</sub></i>	0.26		<i>a<sub>23</sub></i>	30.92	<i>a<sub>32</sub></i>	0.71		<i>a<sub>23</sub></i>	16.92
<i>a<sub>33</sub></i>	2.03		<i>a<sub>31</sub></i>	21.30	<i>a<sub>33</sub></i>	0.90		<i>a<sub>31</sub></i>	0.61
<i>b<sub>11</sub></i>	3234.50		<i>a<sub>32</sub></i>	0.40	<i>b<sub>11</sub></i>	13729.60		<i>a<sub>32</sub></i>	0.79
<i>b<sub>12</sub></i>	4738.63		<i>a<sub>33</sub></i>	4.87	<i>b<sub>12</sub></i>	27481.38		<i>a<sub>33</sub></i>	2.11
<i>b<sub>21</sub></i>	18.46		<i>b<sub>11</sub></i>	5350.73	<i>b<sub>21</sub></i>	5.46		<i>b<sub>11</sub></i>	3094.15
<i>b<sub>22</sub></i>	50.98		<i>b<sub>12</sub></i>	15775.33	<i>b<sub>22</sub></i>	37.50		<i>b<sub>12</sub></i>	28580.41
<i>b<sub>23</sub></i>	0.98		<i>b<sub>13</sub></i>	16.40	<i>b<sub>23</sub></i>	2.79		<i>b<sub>13</sub></i>	4.49
<i>b<sub>31</sub></i>	134.93		<i>b<sub>21</sub></i>	7.41	<i>b<sub>31</sub></i>	54.98		<i>b<sub>21</sub></i>	6.08
<i>b<sub>32</sub></i>	9.81		<i>b<sub>22</sub></i>	6.45	<i>b<sub>32</sub></i>	33.30		<i>b<sub>22</sub></i>	9.53
<i>b<sub>33</sub></i>	7.94		<i>b<sub>23</sub></i>	1.02	<i>b<sub>33</sub></i>	11.28		<i>b<sub>23</sub></i>	3.20
			<i>b<sub>31</sub></i>	180.98				<i>b<sub>31</sub></i>	82.22
			<i>b<sub>32</sub></i>	11.13				<i>b<sub>32</sub></i>	36.10
			<i>b<sub>33</sub></i>	19.08				<i>b<sub>33</sub></i>	25.92

Table 2.3. Significance of the IO parameters i.e. absolute value of the parameter divided by its standard deviation.



# 3 Mapping quality prediction for RTK micro-drones operating in complex environment

The Bundle-Adjustment algorithm based on the theory presented in Chapter 1, and applied to real application in chapter 2 is here coupled to a custom photogrammetric simulator to predict the precision of a mapping procedure. This chapter is originated from the following preprint.

E. Cledat, L. V. Jospin, D. A. Cucci and J. Skaloud. Mapping Quality Prediction for RTK Micro-Drones Operating in Complex Environment *ISPRS journal*, 2020

The idea originated from D. A. Cucci and J. Skaloud was first implemented in *Matlab* by E. Cledat and then ported in *C++* by L. V. Jospin who also suggested the use of *log* interpolation and LERM. The data acquisition was achieved by E. Cledat, D. A. Cucci and J. Skaloud, and the validation procedure was mainly achieved by E. Cledat.

## Abstract

Drone mapping with GNSS-assisted photogrammetry is a highly efficient method for surveying small- or medium-sized areas. However, the mapping quality is not intuitively predictable, particularly in complex environments (with steep and cluttered terrain), in which the quality of the real-time kinematic (RTK) or post-processed kinematic (PPK) positioning varies. We present a method to predict the mapping quality from the information that is available prior to the flight, such as the flight plan, expected flight time, approximate digital terrain model, prevailing surface texture, and embedded sensor characteristics. After detailing the important considerations, we also present the concept of global precision within the context of minimal and efficient ground control point placement in a complex terrain. Finally, we validate the proposed methodology by means of rigorous statistical testing against numerous experiments conducted under different mapping conditions.

## 3.1 Introduction

### 3.1.1 Motivation

Unmanned aerial vehicles (UAVs) are becoming an important tool for surveyors, engineers, and scientists as the number of available cost-effective and easy-to-use systems is rapidly increasing. These platforms offer an attractive alternative to mapping small areas with centimeter-level resolution. Numerous successful applications have been reported: repetitive surveys of buildings, civil engineering structures or construction sites, as well as land monitoring and precision farming [39].

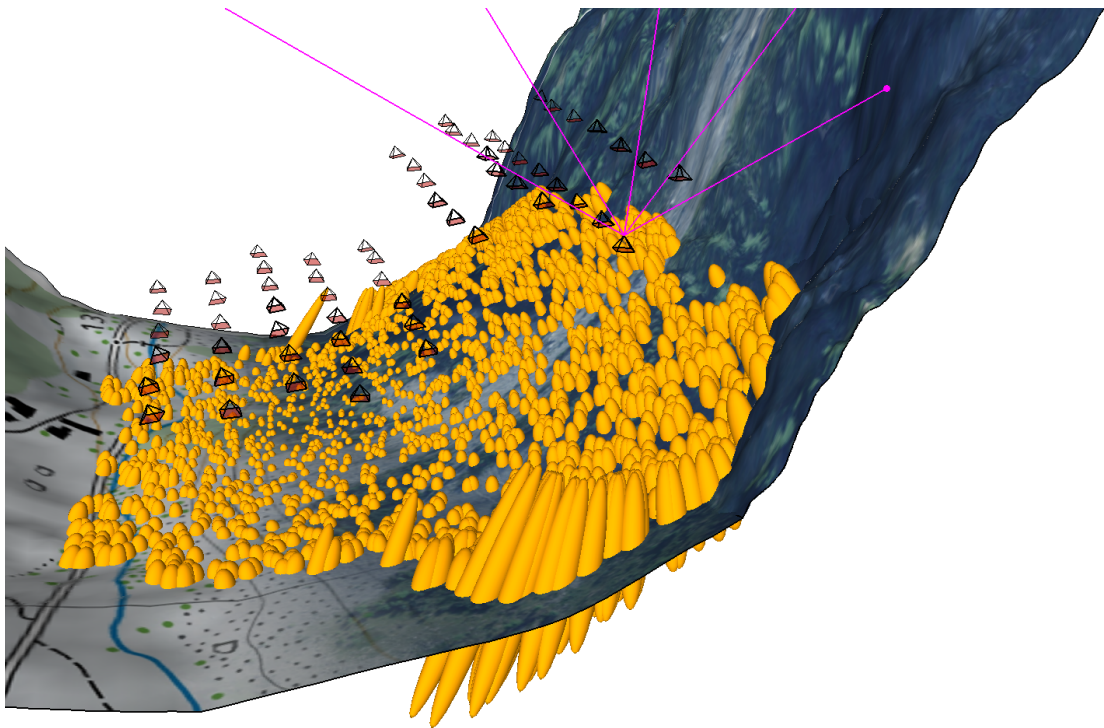


Figure 3.1. Drone mapping in mountainous environment with prediction of satellite visibility and tie-point uncertainty.

The aerial images that are acquired in this manner need to be geo-referenced accurately and precisely, either to localize specific features of interest in the images, such as cracks or corrosion spots in concrete structures, or for three-dimensional (3D) modeling. The typical airborne photogrammetric acquisition of a drone is depicted in Fig. 3.1, in which the flight plan is conceived to follow the terrain contours. Aerial triangulation, in which photographs are linked through image observations of so-called *tie-points*, is a common method. The 3D model data in object space are obtained using multiple ground control points (GCPs) [158]. These points limit model distortions owing to error accumulation and are also used for quality

control.

The process of initializing a dense and uniform ground control network is notoriously time and cost intensive (as physical signalization and separate surveying are required). Moreover, the process is sometimes complicated by the terrain, severely limiting the response time and the application of UAVs in difficult environments and/or under strict accuracy requirements (Fig. 3.1 - variation in ground precision). This is why initially research [129] and later industry [141] and [102] focused on using airborne photogrammetry methods for UAVs, with the aim of removing or mitigating the need for GCPs. In assisted aerial triangulation, ground control is replaced with aerial control: a survey-grade GNSS receiver can provide centimeter-level accuracy positions for each camera station, provided that the reference and rover receivers share sufficient satellites (Fig. 3.1 - signal obstruction) in an appropriate geometric configuration (as in the right part of Fig. 3.2) and that the carrier-phase signal is of sufficient quality. This remains challenging onboard small UAVs [153]. This technique is very popular at present in real-time kinematic (RTK) and post-processed kinematic (PPK)-enabled UAVs. Moreover, survey-grade GNSS receivers are now available on the market [142] and [102]. If an inertial measurement unit of sufficient quality is also available; for example, as in [105], orientation control can be employed, which may be required in difficult mapping scenarios such as corridor mapping [130]. In recent years, variations of such techniques have been developed to deal with specific peculiarities of UAV-based aerial photogrammetry, including that of relative position/orientation control [16] and [131] and raw observation, and multi-sensor adjustment [43].

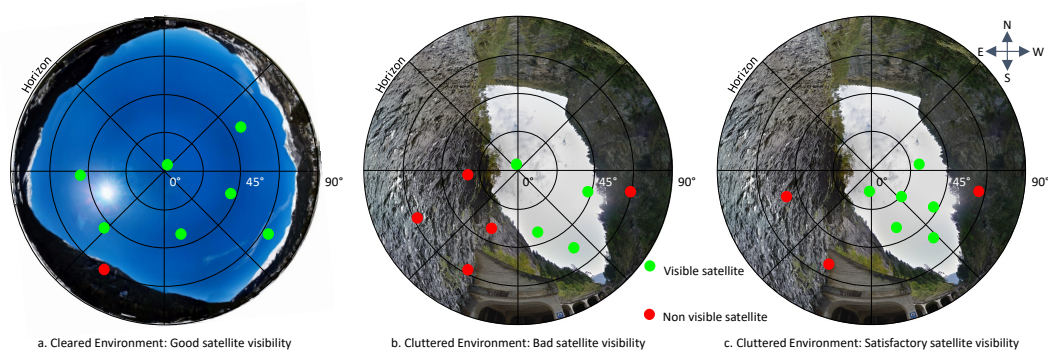


Figure 3.2. Sky-view (azimuth - zenith coordinates) showing satellite geometry.

### 3.1.2 Challenges

Despite the extensive range of recent possibilities offered by both hardware and post-processing methodologies, the survey design remains crucial for the efficient deployment of UAVs: the operator needs to select factors such as suitable sensors, the configuration and number of GCPs, and the flight plan and its time schedule so that the accuracy requirements are fulfilled, while minimizing the survey time and cost as well as the chances of repeating all or part of the survey owing to unsatisfactory results.

In difficult mapping scenarios, such as corridor mapping or cluttered environments, in which high variations in the topography cause the image overlap and sampling distance to vary substantially, it becomes difficult, even for expert operators, to predict the quality of the final mapping products. For example, Fig. 3.1 shows how the 3D object point precision varies depending on the individual performance of directly observed trajectory information, individual image overlap and terrain texture. A direct consequence thereof is that either too many or too few GCPs are installed, leading in the first case to an unnecessary cost increase, or in the second case, requiring a repeat of all or part of the mission.

Another source of uncertainty in the survey outcomes arises from the quality of the aerial position control, which in the first instance affects drone guidance and in the second instance influences the image orientation quality. The latter aspect is partly driven by the signal-to-noise ratio (SNR) of the carrier-phase GNSS signal on two or more frequencies, which is determined by the drone hardware components and their assembly. Furthermore, the satellite constellation available at the time of the flight affects both the capability of obtaining a fixed GNSS position throughout the entire trajectory (operational safety), and the ability to resolve and maintain the double-difference carrier-phase ambiguities that are essential for observing the camera-coordinates with centimeter-level precision correctly.

The importance of GNSS constellation planning is well recognized among surveyors. However, it is typically performed for representative positions with arbitrary elevation masks, and the terrain is not necessarily taken into account (Figs. 3.1 and 3.2). Furthermore, the fact that the satellite occlusions vary along the drone trajectory and as a function of time is neglected. Figure 3.9 presents a situation in a cluttered environment in which the GNSS signal quality varies with time within the same flight.

### 3.1.3 Conventional approach

The commonly used method for UAV photogrammetry is to fly according to a flight plan and to post-process the recorded data to produce surface-related products such as 3D models and ortho-images. The images are first oriented either approximately by using navigation sensors or accurately using the computationally intensive process of tie-point identification



and matching. All inputs, namely the image coordinates, object coordinates, and possibly the camera position/attitude (GNSS + INS) are combined with appropriate weights within a bundle adjustment (BA), from which the mapping quality factors are derived. If the quality is unsatisfactory, the process needs to be restarted to acquire additional images, improve the aerial control or add more GCPs (a new target if new photographs are acquired or the measurement of existing identifiable targets).

The simulation of the geometry for the 3D reconstruction problem has already been explored, although it is not necessarily part of drone mission planning. For example, in [124], a method was described to simulate the geometry of a photogrammetric survey. In [25], noise was added to the simulated image measurements, and 3D reconstruction was executed and compared with the ground truth to quantify the expected survey precision. In this work, the reconstruction was executed only once, while in [94] and [160], a Monte Carlo approach was used, exploiting several runs of the reconstruction step to enable estimation of the covariance matrix of the expected precision. In [116] and [118], the expected precision covariance matrices were obtained via linear covariance propagation. However, the inclusion of aerial control adds another level of complexity to this process, especially if the spatio-temporal variations in the expected aerial control quality in obstructed environments are considered.

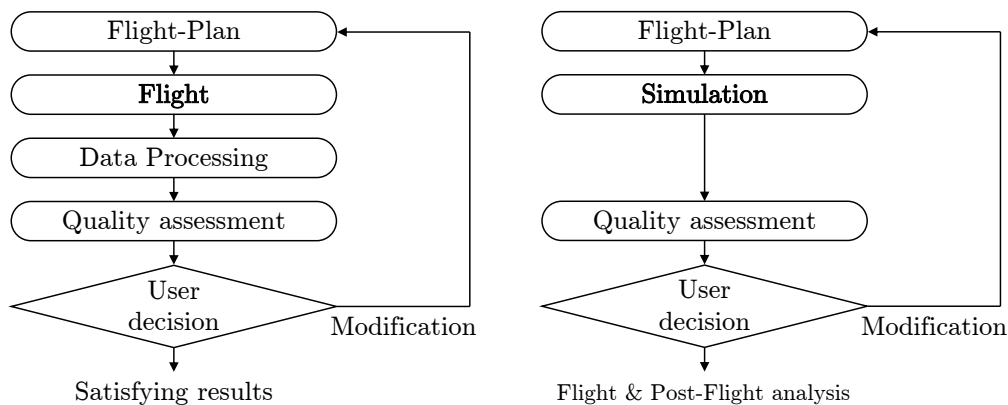


Figure 3.3. Conventional (left) and proposed (right) workflow from flight planning to survey execution with quality assessment.

### 3.1.4 Proposition

In this study, we consider the fact that satellite visibility changes while the drone is moving during the mission planning phase (that is, both the satellites and the portion of visible sky are moving), and predict the capability of the drone guidance (sustaining the minimum number

## **Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment**

---

of visible satellites) as well as the success of the kinematic ambiguity resolution in real-time (RTK) or post-processing (PPK). Thereafter, we combine this information with either planned or realized (immediately after landing) image overlap, the prevailing texture of the scene, and the distribution of the GCPs to predict an accuracy map over the surveyed area that may be inspected by the operator via interactive visualization. The mission parameters can subsequently be altered to improve the estimated accuracy. If the predicted accuracy does not meet the operator requirements, stability analysis of the entire BA network is performed to determine the lowest-accuracy locations and to propose the optimal placement of additional GCP(s). Similar analyses are performed following the flight to display the predicted accuracy map with the actual camera perspective center position/attitude and its quality prior to processing the actual images.

The remainder of this paper is organized as follows: In Section 3.2, we present the methodology for simulating the required inputs. We detail the relationship between the geometry of visible satellites and the probability of successful ambiguity resolution, which is used to derive the aerial position uncertainties for each camera station and time. These are encapsulated into one parameter that serves as guidance for proposing the optimal mission time of the day. Moreover, we describe the additional inputs related to the probabilistic tie-point distribution based on the scene texture. Thereafter, we present the local and global quality factors that are useful for precision evaluation. Subsequently, in Section 3.3, we present an analysis of the network for its principal weaknesses and demonstrate the most efficient improvement in the mapping precision by placing additional GCP(s) within the indicated area(s). Finally, in Section 3.4, we present experiments conducted in mountainous environments to validate the proposed methodology.

## **3.2 Methodology**

### **3.2.1 Concept**

As indicated in the left part of Fig. 3.3, the conventional loop of data acquisition, post-flight processing, and derivation of the quality control at the end of the survey process is not ideal, particularly in complex scenarios in which the aerial position control quality is likely to vary with time. Therefore, we aim to extend such tools with the spatio-temporal elements of the aerial control in an obstructed environment in combination with probabilistic measures for tie-point placement to simulate the acquisition process realistically and derive the expected quality prior to the flight. Overall, we aim to shorten the interaction loop before obtaining acceptable results regarding the conventional approach, in which the quality estimators are based on real data, possibly leading to repeated missions (left vs. right organogram of Fig. 3.3).

As schematically depicted in Fig. 3.4, the proposed method is based on the following inputs:

### 3.2. Methodology

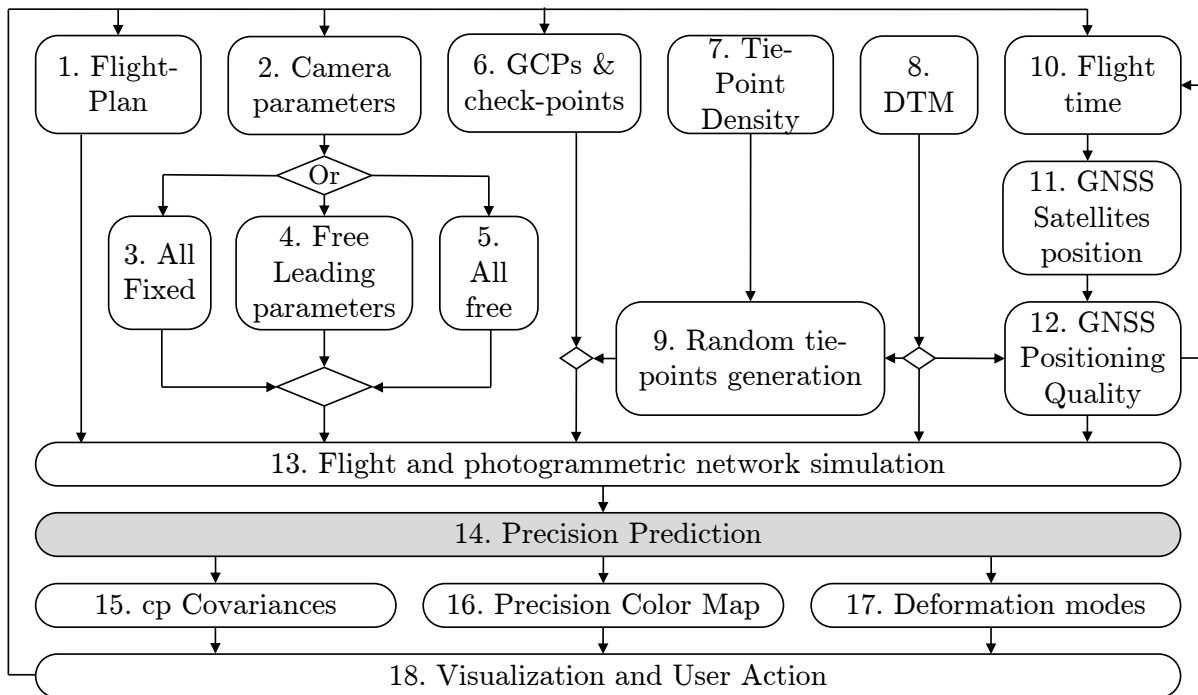


Figure 3.4. Workflow for predicting mapping precision.

i) the flight plan covering the area with the aimed overlap and resolution represented by the mean ground sampling distance; ii) the foreseen flight date and time; iii) a coarse digital terrain model (DTM)<sup>1</sup>; and iv) the nominal camera interior orientation and whether this needs to be re-calibrated [52]. Based on these elements, simulated image and aerial control observations are generated, along with their expected precision (which varies temporally and spatially for aerial position control), and a full adjustment is run. This forms a basis for formulating realistic predictions of the mapping quality that can be expected from the actual survey at a specific date and time of the day, with details over the entire area. On this basis, the user can interactively modify the flight plan (for example, by selecting a different time, payload, flight line, and overlap) and ground control network (for example, by adding GCPs or changing their placements) until the quality requirements are fulfilled, thereby maximizing the probability of a successful data collection campaign and minimizing the costs thereof.

If the predicted quality is higher than the requirements, the operating procedure could be simplified, thereby enabling a reduction in the cost, prior to executing the flight and related survey. If the quality is unsatisfactory and improvement is sought through additional GCPs, further analysis of the network is performed to identify its weakest characteristics and areas in

<sup>1</sup>Coarse resolution DTMs are available globally; for example, see [111]. In most developed countries, higher-resolution DTMs are provided by the respective topographic agencies

which the placement of GCPs would be the most effective.

### 3.2.2 Basic relation

The basic relation for 3D object restitution in photogrammetry is the collinearity in Eq. 3.1, which is in its most simple form based on the pinhole camera model. It relates the 3D coordinates of a tie-point  $P_t^m$  in a mapping frame ( $m$ ) to its image coordinates  $x^c, y^c$  (measured from the principal point on a theoretical focal plane, the distance of which to the camera perspective center  $P_c$  is 1) via a scale factor  $\lambda$ , rotation matrix  $R_c^m$ , and the 3D position of the camera perspective center in the mapping frame  $P_c^m$ :

$$\exists \lambda \in \mathbb{R}, P_t^m = P_c^m + \lambda R_c^m \begin{bmatrix} x^c \\ y^c \\ 1. \end{bmatrix} \quad (3.1)$$

Additional parameters modeling departures from collinearity need to be assumed for real cameras, as in Section 3.2.4. The observations in relation to different elements of Eq. 3.1 are described in the following Sections 3.2.3 and 3.2.8.

### 3.2.3 Flight plan

The flight preparation is achieved by a mission planner such as [117] and [56]. The principle of such software is to assist the user in designing the trajectory that the drone is supposed to fly according to certain criteria. It requires an approximate DTM such as [111] to achieve the desired parameters; for example, the scale (ground sampling distance) and lateral overlap, while ensuring that the drone remains below the maximum height fixed by the regulations.

Apart from providing the input for the drone guidance, the mission planner supplies the rough values of the camera poses for the simulation (Fig. 3.4). The positioning precision of the RTK/PPK is assumed to exhibit time-varying characteristics and is further detailed in Section 3.2.7.

### 3.2.4 Camera model

$$\begin{bmatrix} \ell_x + v_x \\ \ell_y + v_y \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + p_d \cdot \left( \underbrace{\left(1 + K_1 r^2 + K_2 r^4 + K_3 r^6\right)}_{\text{radial distortions}} \begin{bmatrix} x^c \\ y^c \end{bmatrix} + \underbrace{\begin{bmatrix} P_1 (r^2 + 2x^c y^c) + 2P_2 x^c y^c \\ 2P_1 x^c y^c + P_2 (r^2 + 2y^c y^c) \end{bmatrix}}_{\text{tangential distortions}} \right) \quad (3.2)$$

The camera model relates the coordinates  $(x^c, y^c)$  in Eq. 3.1 to the observed pixels  $(\ell_x, \ell_y)$  in a real image. It includes the sensor position in the camera frame and additional parameters related to departure from collinearity. A commonly used physical model is that of Brown [51], which is defined in Eq. 3.2.

where  $(p_x, p_y)$  is the position of the principal point,  $p_d$  is the principal distance,  $(K_{1,2,3})$  and  $(P_{1,2})$  are the radial and tangential distortions parameters and  $r^2 = x^{c2} + y^{c2}$ . The camera parameter values can be obtained from a previous project; alternatively, the approximate magnitude of  $p_d$  from the sensor datasheet can be used. The parameter uncertainty and need for re-calibration will influence the mapping precision. A reasonable selection considers the extreme cases (in which all parameters are either free or fixed: Fig. 3.4, steps 3/5), as well as an intermediate scenario (Fig. 3.4, step 4). The latter reflects the case in which the lens-related parameters may be temporarily stable (particularly on a rigidly mounted prime lens used at a similar temperature), while the values of  $p_x$ ,  $p_y$ , and  $p_d$  may vary slightly among flights on different days and need to be readjusted.

### 3.2.5 Ground control

In the absence of aerial control, at least three GCPs are required for absolute orientation in the mapping frame, whereas more may be necessary to improve the mapping quality. Their position and placement are typically part of the project design, for the purpose of which their approximate locations can be obtained by clicking on the digital map employed in the flight planning (Fig. 3.4, step 6). Alternatively, existing coordinates can be loaded in a file together with their accuracy values, which is dependent on surveying technology (such as GNSS and leveling) to be considered as the observations:

$$\ell_{GCP} + v_{GCP} = P_{GCP}^m \quad (3.3)$$

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

The answer to the influence of the GCPs on the mapping accuracy in terms of the number, quality, and distribution is part of the analysis presented in Section 3.3.2.

#### 3.2.6 Tie-points

We consider the tie-point as a general term for an image feature, the center of which is observed in at least two images by tie-point detection and a matching algorithm. Although the quality and quantity of the automated tie-point detection and matching vary slightly with the employed algorithms [106] and [137], the surface texture is the most important factor. Indeed, the tie-point density on surfaces with a homogeneous texture (for example, water, fresh snow, and certain types of vegetation) may be low or even zero, whereas it may be extremely high in texture-rich environments (such as built-up areas).

On a textured surface, the tie-point matching could be inadequate for two reasons. *Duplication* is when a same object-point is recognized as two distinct, but very close image point in a photo, resulting to two distinct matching (blue and red in 3.5) with different other photos. Tie-points duplication could reduce the quality of the final results. First, the point cloud will be noisier, because there will be two distinct points instead of only one. Second, it reduces the redundancy, since there is more unknown tie-points to compute, and the quality of the intersections is degraded. Wrong matches corresponds to a matching of two distinct points in the real world (see Figure 3.5). They are usually detected to be blunders by photogrammetric software. The corresponding tie-points observations are thus removed from the functional model. However, if this observation remains, it could propagate onto the whole photogrammetric network, leading to inaccurate results.

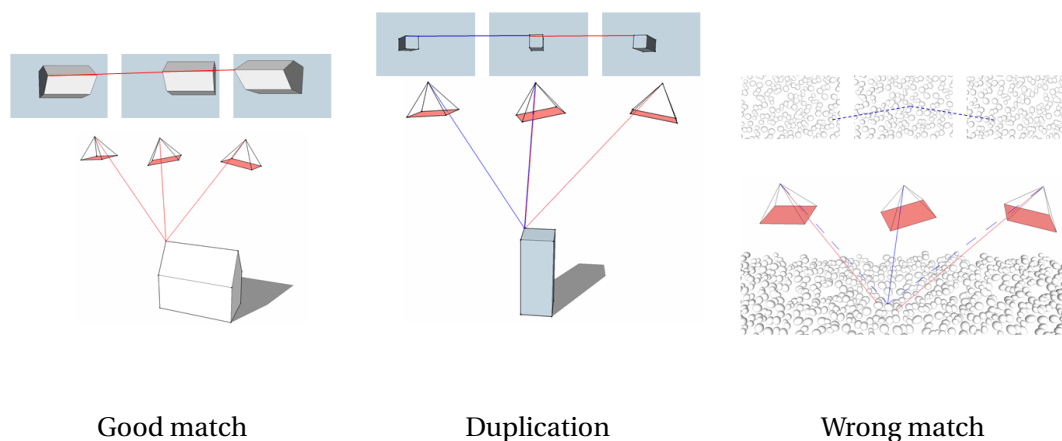


Figure 3.5. Optimal and non-optimal tie-point matching

As the real texture is unknown at the prediction stage, the outcome of such a process can only

### 3.2. Methodology

be probabilistic. To determine such a probabilistic measure, we evaluated several flights at a 100 – 200 m height above ground level, in which we considered several texture categories and calculated their mean tie-point densities, as displayed in Table 3.1. Thereafter, we used the image residuals in these categories to estimate the mean variance of the image observations empirically.




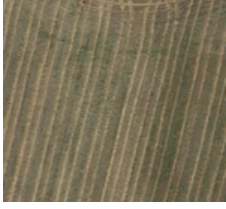

		#tie-points / megapix	$\sigma$ tie-point [pix]
Water		0	$\infty$
powder snow		0-8	4
crop		40	2
bare ground		200	1
built-up		1000	0.8

Table 3.1. Texture categories with mean tie-points density and precision.

It is reasonable to assume that the selection of the prevailing texture is guided by *a priori* knowledge regarding the terrain. Thereafter, the corresponding empirical values of the image tie-point density in Table 3.1 are converted into the tie-point density on the ground via flight parameters to generate tie-points on the approximate digital surface (Fig. 3.4, step 8) using a stochastic process (Fig. 3.4, step 9). These tie-points are subsequently merged with the other points (GCPs; check-points) that are manually entered by the user (Fig. 3.4, step 6) to generate image observations. The influence of the tie-point distribution and quality is discussed in Section 3.4.3.

#### 3.2.7 Aerial position control

The relation between the GNSS-derived position and camera position is given by the following equation, where  $\ell_A$  is the 3D position given by the GNSS antenna,  $P_c^m$  is the camera position in the mapping frame,  $R_c^m$  is the camera orientation, and  $\vec{a}^c$  is the lever arm between the center of the GNSS antenna phase and the camera perspective center.

$$\ell_A + v_A = P_c^m + R_c^m \vec{a}^c \quad (3.4)$$

In manned airborne missions, *shift* parameters (or *drift* parameters) could be added to the above equation to *absorb* effects due to incorrect determination of ambiguities within a block or per flight-line. In drone mapping the shorter distance to the base improves the reliability of ambiguity determination due to differential atmosphere, and thus improve greatly the accuracy. However, wrongly determined ambiguities could still happen if the geometric configuration is weak, and that — due to proximity of obstacles — could happen anytime within a flight-line. For these reasons, the modelling by shift (drift) parameters is no longer appropriate and it is better to plan the flight execution so that acceptable observation conditions are maintained throughout the whole trajectory. As mentioned previously, apart from the signal strength (SNR) in the code and carrier-phase observations, the precision of the GNSS positioning is mainly dependent on (i) the number of observed satellites and (ii) the satellite-to-receiver geometry. These geometrical factors can be analyzed in advance from the mission plan and coarse digital elevation model (DEM) over a time-span that is specified by the user (Fig. 3.4, step 10). For this purpose, the most recent information regarding the GNSS approximate satellite position is obtained from the almanac [66] or ephemerides. The time of each photo is predicted by the mission plan with respect to the specified mission start, considering its planned position and the nominal UAV speed. The positions of the GNSS satellites are computed from the almanac at each planned camera location and time (Fig. 3.4, step 11), while a ray-tracing algorithm allows for determining whether this satellite is in the



line of sight; that is, not obstructed by the terrain [115].

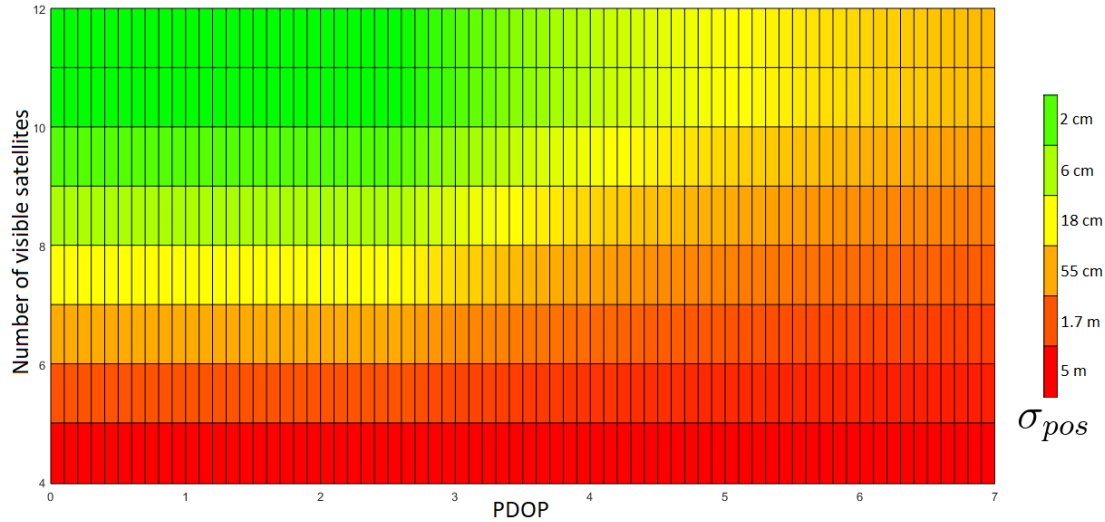


Figure 3.6. Empirical GNSS precision positioning as function of number of visible satellites and co-factor elements.

Once the visible satellite configuration is determined, the positioning covariance matrix is predicted by the calculation of the co-factor matrix scaled by the ranging precision (Fig. 3.4, step 12). The latter depends on the foreseen positioning mode used in Eq. 3.4: with carrier-phase differential corrections or without; that is, standalone positioning. The former can be achieved either during the flight (RTK) or thereafter (PPK). Although the noise level of the PPK is generally lower than that of the RTK, as discussed later (Section 3.4.5), the crucial aspect affecting the ranging is the capacity of resolving the carrier-phase ambiguities. This is related to the size of the ambiguity search space and its geometrical shape, which is proportional to the principal elements of the GNSS covariance matrix or its compound metric, known as the position dilution of precision (PDOP) [156]. We suggest employing the empirical probabilistic function proposed by [139], based on experience with many helicopter flights close to the terrain in a mountainous environment, to inflate the previously derived covariance matrix by means of such a function. This is illustrated in Fig. 3.6 as a function of the PDOP and number of observed satellites.

The color scheme of the probable positioning quality in each photograph is displayed in the flight plan, as illustrated in Fig. 3.9, to guide the feasibility of the foreseen flying time intuitively. If the result is not satisfactory (more red than green over a specific area), the flight time can be modified (by means of back-looping from Fig. 3.4, step 12 to Fig. 3.4, step 10) to analyze another interval.

### 3.2.8 Precision map

The precision map is based on the predicted covariances of the individual tie-points in the object space. These are obtained by co-variance propagation of the previously described observation equations with the simulated measurements. This problem is part of the simultaneous optimization of the camera orientation and 3D location of the ground points from the image observations and aerial ground control; that is, the BA. In BA, the residuals associated with each observation are minimized as a function of the parameters by means of weighted nonlinear least-squares methods; for example, as described in [158] and [51] or for specific aerial control of UAVs in [131].

Firstly, a ray-tracing algorithm is applied to construct an adjacency list that links the camera orientations and tie-points in view in its image.

This set of observations, together with the terrestrial position (Section 3.2.5) and aerial position (Section 3.2.7) permits the photogrammetric network to be simulated (Fig. 3.4, step 13), with the weight matrix defined as the inverse of the covariance matrix of the observation vector  $\Sigma_{\ell\ell}$ . Assuming that the observation errors are zero mean, Gaussian distributed, and independent, a single standard deviation for each observation or class of observations can be specified. The parameters are the camera poses (position and attitude), tie-point positions, and intrinsic camera parameters (for example, principal distance, principal point, and/or lens distortion parameters).

These parameters are concatenated in the state vector  $\mathbf{x}$ . Let  $\mathbf{f}$  be the function describing the observation model; that is, the function such that  $\ell = \mathbf{f}(\mathbf{x})$ . All of the collinearity equations of all tie-point measurements (Section 3.2.4), aerial position observations (Section 3.2.7), and terrestrial observations Sec. 3.2.5) are enclosed in this  $\mathbf{f}$  function. Algorithms that are used to solve least-squares problems (such as Gauss–Newton, Levenberg–Marquardt or the dog-leg algorithm) require the Jacobian matrix of  $\mathbf{f}$  with respect to  $\mathbf{x}$ . This matrix, which is known as the *design matrix*, is denoted by  $\mathbf{A}$  in the following. The Lie group theory [149] is used to handle the derivations of the rotation matrices describing the camera orientation.

In the scope of covariance propagation, the values of the parameters that are known *a priori* are used to construct  $\mathbf{A}$ . Then, the covariance matrices of the parameters  $\Sigma_{xx}$  (Fig. 3.4, step 14), including those of the tie-points, are obtained by inverting the approximated Hessian matrix  $\mathbf{H}$ :

$$\Sigma_{xx} = \mathbf{H}^{-1} = (\mathbf{A}^T \Sigma_{\ell\ell}^{-1} \mathbf{A})^{-1}. \quad (3.5)$$

The inversion of the Hessian matrix is computationally demanding in terms of both memory and time, and it is therefore generally avoided in BA. As we are interested only in certain blocks close to the diagonal, the methods described in [136] or [84] are preferable, as they allow for computation of only the required elements.

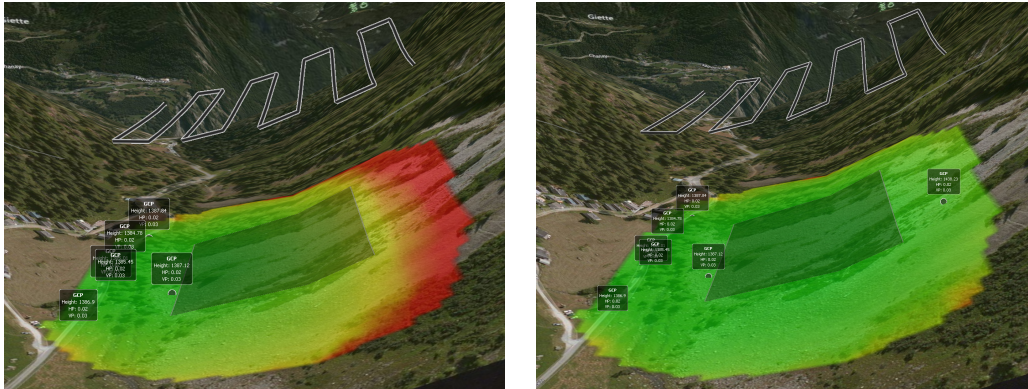


Figure 3.7. Visualization of predicted mapping precision for bad (left) and good (right) distribution of GCPs.

The extracted covariance matrices of the tie-points can be used to depict the 3D uncertainty, as illustrated in Fig. 3.1, but such a representation is less suitable for visualizing a point-cloud containing all tie-points. Therefore, we propose propagating the uncertainty from the tie-points (which can be represented by ellipsoids) to a mesh surface (Fig. 3.4, step 16), as illustrated in Fig. 3.7. The rigorous projection of 3D precision onto a surface is a non-trivial mathematical operation. Indeed, a simple approach using (for example, bilinear) interpolation of the covariance matrix elements could lead to values that differ from the truth by a factor of three, as indicated in Table 3.5 of 3.6. A preferable approach is to perform error propagation from the tie-points used as vertices in the digital surface model, as pursued in [146]. However, this strategy possibly underestimates the error in the face centers, and is therefore dependent on the DTM resolution. A numerical comparison employing this method indicated that the possible inaccuracy was still considerable (refer to Table 3.5). Although a certain degree of imprecision owing to projection in the visualization is inevitable, using a method that considers the correct manifold of the covariance matrices, such as the log-Euclidean Riemannian metric (LERM) [8], can maintain it at a tolerable level (for example,  $\sim 5\%$ ). The details of such methodology are presented in 3.6, while the visualization results are depicted in Fig. 3.7.

### **3.3 Precision analysis**

The influences on the mapping precision are multiple. One aspect is related to errors in the observations (image, terrestrial, and aerial); another aspect is related to the prior knowledge (camera intrinsic uncertainty) and global geometry (such as the number of parameters, observation redundancy, and corridor/block configuration). To detect the principal weaknesses and suggest an improvement, we propose first analyzing the situation locally at the scale of individual tie-points, and later at the level of the entire network.

#### **3.3.1 Local**

The predicted precision of a tie-point is provided by the covariance matrix of this tie-point, which corresponds to the appropriate lines and columns extracted from  $\Sigma_{xx}$ . This covariance is influenced by numerous factors, including the precision of the image observations, number of images in which this tie-point is observed and the precision of their aerial control, the precision of the camera intrinsic parameters, and the location of the tie-points in the images.

The local analysis does not consider the relationship between tie-points, and in its simplest form only inspects the block diagonals of  $\Sigma_{xx}$ , as employed in the precision map creation (Section 3.2.8). However, the values of the off-diagonal may be significant; for example, in a steep terrain mapped with a nadir-oriented camera. In such a situation, it is recommended either to i) perform principal component analysis of the covariance matrix and display its most significant component(s), or ii) rotate the covariance matrix to align it with the terrain principal slope and aspect when displaying the diagonal terms.

#### **3.3.2 Global**

Larger-scale analysis is useful for identifying the principal geometrical weaknesses of the entire photogrammetric network and when searching for their effective mitigation. We first review its derivation from the estimated uncertainty.

Let  $\Sigma_{xx}$  be a covariance matrix of the vector  $\mathbf{x}$  of  $n$  random variables, and  $V$  be an  $n$ -dimensional unitary vector. The variance of  $V^T \mathbf{x}$  is  $V^T \Sigma_{xx} V$  (this theorem is trivial when  $V$  is one vector of the canonical base). If  $\Sigma_{xx}$  is the covariance matrix of the parameters in our system, the maximum weakness of this system is in the direction  $V$ , where  $V^T \mathbf{x}$  has its maximum variance; thus, when  $V$  is the eigenvector associated with the maximum eigenvalue of  $\Sigma_{xx}$ .

$$\Sigma_{xx} V = \lambda V \tag{3.6}$$

### 3.3. Precision analysis

Analogously to the analysis of structural stiffness in civil engineering<sup>2</sup>, this  $V$  is associated with the first vibration mode of a beam structure; that is, the vibration mode for which the structure is the most likely to oscillate.

Note that a similar analysis is used in geodetic networks [10] and [13] and forms part of modern adjustment software [93] and [28], including the registration of dense networks of terrestrial laser scanning [71].

Let  $\lambda_{max}$  be the largest eigenvalue of  $\Sigma_{xx}$  and  $V_{max}$  be its associated eigenvector. Thus,  $V_{max}$  is also the eigenvector associated with the minimum eigenvalue of the Hessian  $H$ , which is the inverse of  $\lambda_{max}$ .

$$HV = \frac{1}{\lambda} V \quad (3.7)$$

The eigenvalues of  $\Sigma_{xx}$  are sorted in descending order from largest to smallest:  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_i \geq \dots \geq \lambda_u$ . Thereafter, a weakness mode  $m_i$  can be defined as  $m_i = \pm \sqrt{\lambda_i} V_i$  for  $i = 1, \dots, u$ .

The application of such analysis to a simulated photogrammetric network over a flat and rectangular area with sub-optimally placed GCPs (close to one corner) is depicted in Fig. 3.8 for the first four modes. The middle part of each sub-plot presents the undistorted situation, whereas the other two parts depict the case with  $\pm \sqrt{\lambda_i} V_i$ . The modes 5 and beyond are insignificant.

---

<sup>2</sup>One could make an analogy between Geodetic or Photogrammetric Networks and Beam Structures in civil engineering. See [82] and [103] (page 104). In a beam-structure, a beam act as a spring, and tries to reach it's relaxed position, while in a geodetic or a photogrammetric network, the compensated lectures tries to be as close as possible from their actual lectures, to keep the residuals as low as possible. The stiffness of the springs is comparable to the observations weight. The degree of freedom of the structure (as it is defined in [26] 4.2.8, 4.2.9) corresponds to the opposite of the total redundancy of the geodetic or the photogrammetric network. In civil engineering, a major problem are resonance effects, as it is shown by the very well-known collapsing of the Tacoma Narrows Bridge [11]. These resonances are due to global weakness of the whole structure. Spectral analysis is a powerful tool for the analysis of both behavior of civil engineering structures face to excitation and Geodetic or Photogrammetric Networks.

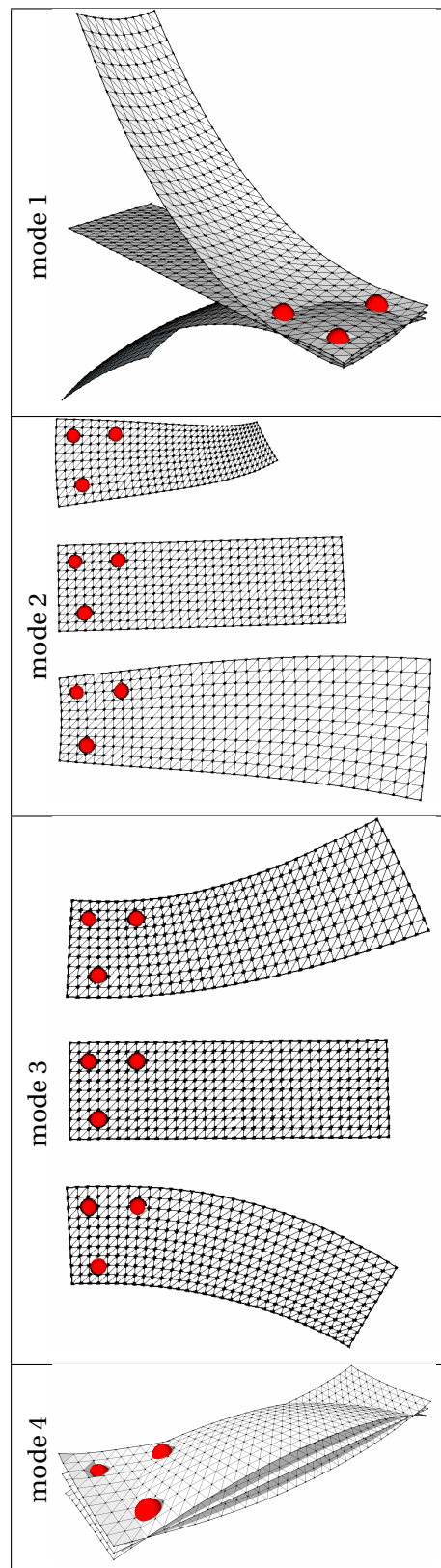


Figure 3.8. First four modes of simulated network with sub-optimally placed GCPs (in red). Each weakness is up-scaled by a factor of 100 for visualization.

#### 3.3.3 Placement of ground control

The previously described analysis may be used as guidance for improving the mapping precision. This can be achieved in several manners; for example, by employing sensors of higher quality, pre-calibrating several intrinsic camera parameters, adding certain GCPs or modifying the flight plan (such as increasing the lateral and inline overlap, and adding flight lines with different orientations and heights).

Whereas modifying the flight plan is a relatively rapid operation, the addition of new GCPs may be considerably more demanding in terms of time and resources. For this reason, the subsequent discussion focuses on the optimization of the GCP locations.

Intuitively, new GCPs are placed at the weakest points of the network. Thus, the local analysis is not an ideal indicator, as the precision of the tie-points may vary significantly within a neighborhood. Moreover, the tie-points close to the mapping area border are usually visible only in certain images, which is one factor that makes them less precise. In such a situation, the errors committed in the (few) observations of the GCP image coordinates may cause larger systematic deformation.

Therefore, the selection of the GCP locations is based on global analysis. In the previously mentioned analogy with a beam structure, in which the nodes represent the camera poses and the beams represent the connections between tie-points in the object and image space, the placement of a new node (GCP) should have a maximum effect on increasing the structure resistance. In our case, it should minimize its deformation owing to the accumulation of random errors.

Therefore, the suggested method consists of studying the eigenvector  $V_1$  associated with the highest eigenvalue of  $\Sigma_{xx}$ . The tie-point with the highest displacement by this eigenvector is detected and is selected as the optimal location to add a new GCP. In Fig. 3.8, this corresponds to adding a GCP at the point for which the deformation from the first excitation mode is maximal, yet avoiding the areas close to the border of the area of interest, and selecting only points that are visible in at least three to four images. The details are explained as part of the experimental validation (Section 3.4.6).

#### 3.3.4 Evaluation of check-point misclosures

Check-points are often used for the assessment of 3D models. However, when the check-points exhibit varying precision in different directions, which may be the case for coordinates determined by static carrier-phase GNSS on mountain slopes, their correct employment requires new considerations.

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

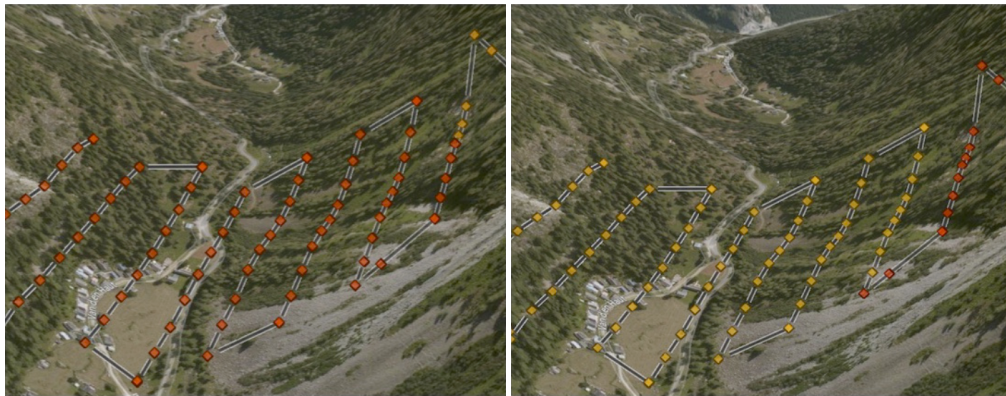


Figure 3.9. Prediction of GNSS positioning precision in each image for two different mission start times at Z1. Color legend as in Fig. 3.6.

Let  $v$  be the check-point misclosure; that is, the difference between the GNSS-determined and photogrammetry-determined coordinates. This misclosure is represented by the vector on Fig. 3.10.

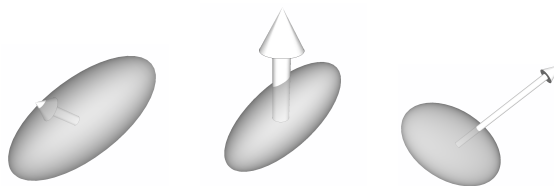


Figure 3.10. Comparison of true missclosure with accuracy prediction

However, the previously presented method only predicts the covariance matrix  $\Sigma_{cp}$  of such a check-point for which the misclosure is  $v$ . Therefore, it is challenging to compare meaningfully the predicted covariance  $\Sigma_{cp}$ , representing the probability density, with punctual realization of the check-point misclosures  $v$ .

Let  $\Sigma_{dm}$  be the covariance matrix of the direct measurement (likely obtained by static carrier-phase differential GNSS<sup>3</sup>).

The misclosure is a simple vector difference with compound covariance  $\Sigma_{dif} = \Sigma_{dm} + \Sigma_{cp}$ .

The combination of the misclosures with their covariances in the square of the Mahalanobis norm  $v^T \Sigma_{dif}^{-1} v$  theoretically follows the  $\chi^2$  distribution when the predicted precisions agree

<sup>3</sup>The positions of the GCPs were measured by terrestrial GNSS with particular care: each point was measured for at least 10 min, leading to a standard deviation of  $\sim 2$  cm in planimetry and  $\sim 3$  cm in altimetry. The covariance matrix of the ground measurement is  $\Sigma_{dm} = \text{diag}((2 \text{ cm})^2 \quad (2 \text{ cm})^2 \quad (3 \text{ cm})^2)$ .



with the observed residuals.

However, if  $\Sigma_{cp}$  is underestimated (the prediction is too optimistic), the square Mahalanobis norm will be high, and if  $\Sigma_{cp}$  is overestimated (the prediction is too pessimistic), the square Mahalanobis norm will be low.

Let  $\Lambda$  be the diagonal matrix containing the eigenvalues of  $\Sigma_{dif}^{-1}$  and  $V$  be the matrix of the associated eigenvectors, which is an orthonormal basis.

$$\Sigma_{dif}^{-1} = V \Lambda V^T \quad (3.8)$$

The standardized misclosures, which are defined as  $\tilde{v} = \sqrt{\Lambda} V^T v$ , are constructed. These are unitless, with unitary standard deviation and uncorrelated coordinates. Note that the squared Mahalanobis norm of  $v$  is equal to the squared Euclidian norm of  $\tilde{v}$ . Thereafter, it is possible to aggregate these  $\tilde{v}$ , either for creating a histogram (for example, bottom plot of Fig. 3.15) or for proceeding with  $\chi^2$  tests.

For the latter, we formulated the hypothesis that the aggregated normalized misclosures follow a standard normal distribution. This hypothesis was not rejected by the one-sample Kolmogorov–Smirnov test [100] at the 5% significance level for 19 of the 26 studied cases. The majority of the problematic (rejected) cases were related to scenarios employing RTK-derived aerial control, indicating possible position biases owing to incorrect determination of (several) double-difference carrier-phase ambiguities in several flight lines.

To proceed with the paradigm of global analysis, as presented in Section 3.3.2, we need to define a new check-point misclosure vector  $v$  as the concatenation of all individual check-point misclosures within one experiment. The notation  $v$  denotes the  $(3 \cdot n) \times 1$  vector of the set of  $n$  check-points. Let  $\Sigma_{cp}$  be the theoretical covariance matrix of this  $v$ , which corresponds to the proper subset of rows and columns of the matrix  $\Sigma_{xx}$ . The principle of the subsequent analysis is to compare  $v$  to the spectral decomposition of its theoretical covariance matrix  $\Sigma_{cp}$ . Let  $\lambda_i$  be the  $i^{th}$  eigenvalue of  $\Sigma_{cp}$  when the eigenvalues are sorted in descending order and  $V_i$  be the associated (normalized) eigenvector. The eigenvalue can be recovered from  $\Sigma_{cp}$  as a result of the following relation:

$$V_i^T \Sigma_{cp} V_i = \lambda_i. \quad (3.9)$$

Subsequently, we firstly have  $\lambda_i$ , namely the predicted variance along the eigenvector  $V_i$ ,

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

and secondly, the projection of the misclosures  $v$  on this eigenvector is  $v \cdot V_i$ . Note that the square of this projection could be interpreted as an *a posteriori* variance  $\hat{\lambda}_i$ , which is evaluated by substituting  $\Sigma_{cp}$  with the *a posteriori* variance of  $v$ :  $\hat{\Sigma}_{cp} = v v^T$ . Figure 3.11 depicts this principle in two dimensions. The true residual  $v$  (in red) is projected onto the eigenvectors  $V_1$  and  $V_2$  of its theoretical covariance matrix  $\Sigma_{cp}$ . The use of spectral analysis in the different experimental cases (Section 3.4) compares the actual realization of the error along an axis ( $v \cdot V_1$  or  $v \cdot V_2$ ) to the error prediction ( $\sqrt{\lambda_1}$  or  $\sqrt{\lambda_2}$ ).

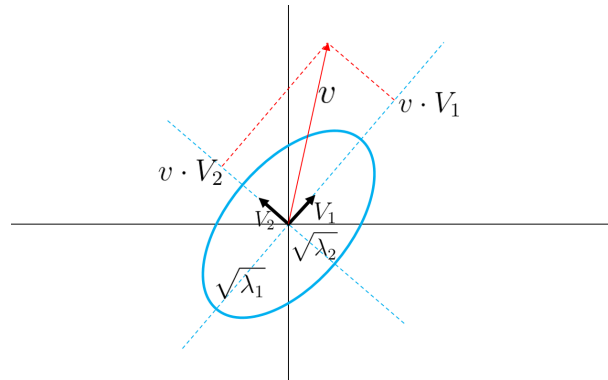


Figure 3.11. Projection of misclosure vector on principal axis of covariance matrix in two dimensions.

### 3.4 Experimental validation

The previously described methodology enables the prediction of the precision of the 3D model generated by the photogrammetric method with GNSS aerial position control under different spatial and temporal scenarios. The purpose of this section is to study the method with real data and validate it in various use cases so that the prediction corresponds to reality.

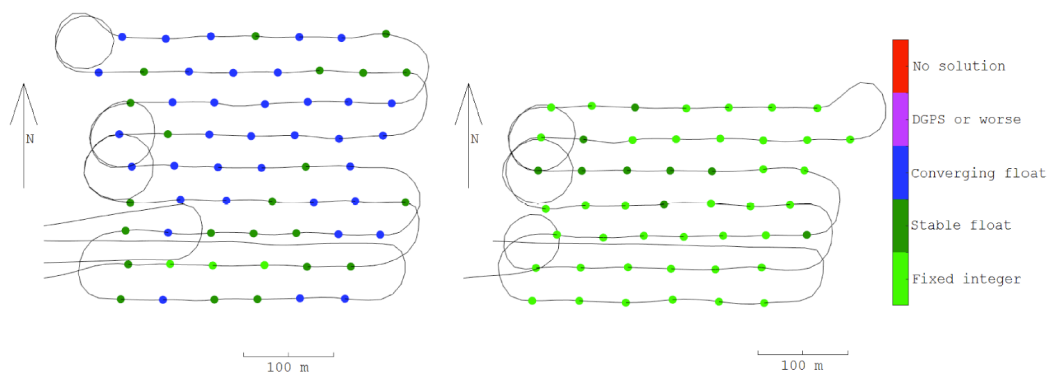


Figure 3.12. Actual GNSS positioning quality status obtained in post-processing (PPK) for two flights realized according to plan depicted in Fig. 3.9.

### 3.4.1 Test sites and equipment

For the experimental validation, we used small, lightweight (~ 1.2 kg) fixed-wing drones equipped with dual-frequency, dual-constellation (GPS & GLONASS) receivers, and small cameras with a ~ 28 – 35 mm equivalent focal length. Two zones of mountainous terrain, located in the western part of Switzerland, were used for the evaluation:

- *Z1*: A narrow valley that is surrounded by high mountain ridges in all directions but East. Numerous houses are located along the bottom of the valley. During winter 2015 to 2016 an exceptionally large avalanche fell down from the South ridge and destroyed part of the forest, the road, and several buildings. This zone first had to be mapped without GCPs (owing to the remaining risk), but repeated mapping later in summer allowed for the placement of 13 GCPs over a large part of the mapped area. The employed planes were an eBee RTK and an eBee+ from senseFly. The former carried a Cannon Power Shot S110 camera (CMOS sensor, 4000 × 3000 pixels) configured for the raw image format.
- *Z2*: A valley with a SE–NE fall line, where the geometry of the GNSS constellation (the number and positions of visible satellites) also varies considerably during the day. The terrain surface is mainly composed of grass and stones. The signalization and surveying of 20 GCPs with the determination of their coordinates by static carrier-phase differential positioning was possible. The employed plane was an eBee+ carrying a S.O.D.A. camera with an integrated lens.

The altitude variations of both areas were approximately 150 to 200 m for both terrains. More than five flights were conducted in each zone with different geometrical configurations and GNSS positioning qualities. These variations are explained in the following comparisons.

### 3.4.2 Aerial position accuracy

The precision of the GNSS quality was predicted for the camera positions in *Z1* using a flight plan evaluated at two different takeoff times, corresponding to “sub-optimal” and “close-to-optimal” satellite visibility conditions. This evaluation employed the satellite almanac and coarse DEM, the results of which are displayed in Fig. 3.9 using the color code of Fig. 3.6.

The actual 3D positioning quality obtained in PPK is illustrated in Fig. 3.12 in terms of an internal characteristic factor varying from 1 to 5 (best to worst). In agreement with the prediction, this indicates the decimeter level and centimeter level for the “sub-optimal” and “optimal” flight times, respectively. Note that, although the 2<sup>nd</sup> flight exhibited better satellite configuration over most of the zone, it was shortened towards the South to maintain aircraft navigation safety, as critical satellite visibility was predicted at the final line.

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

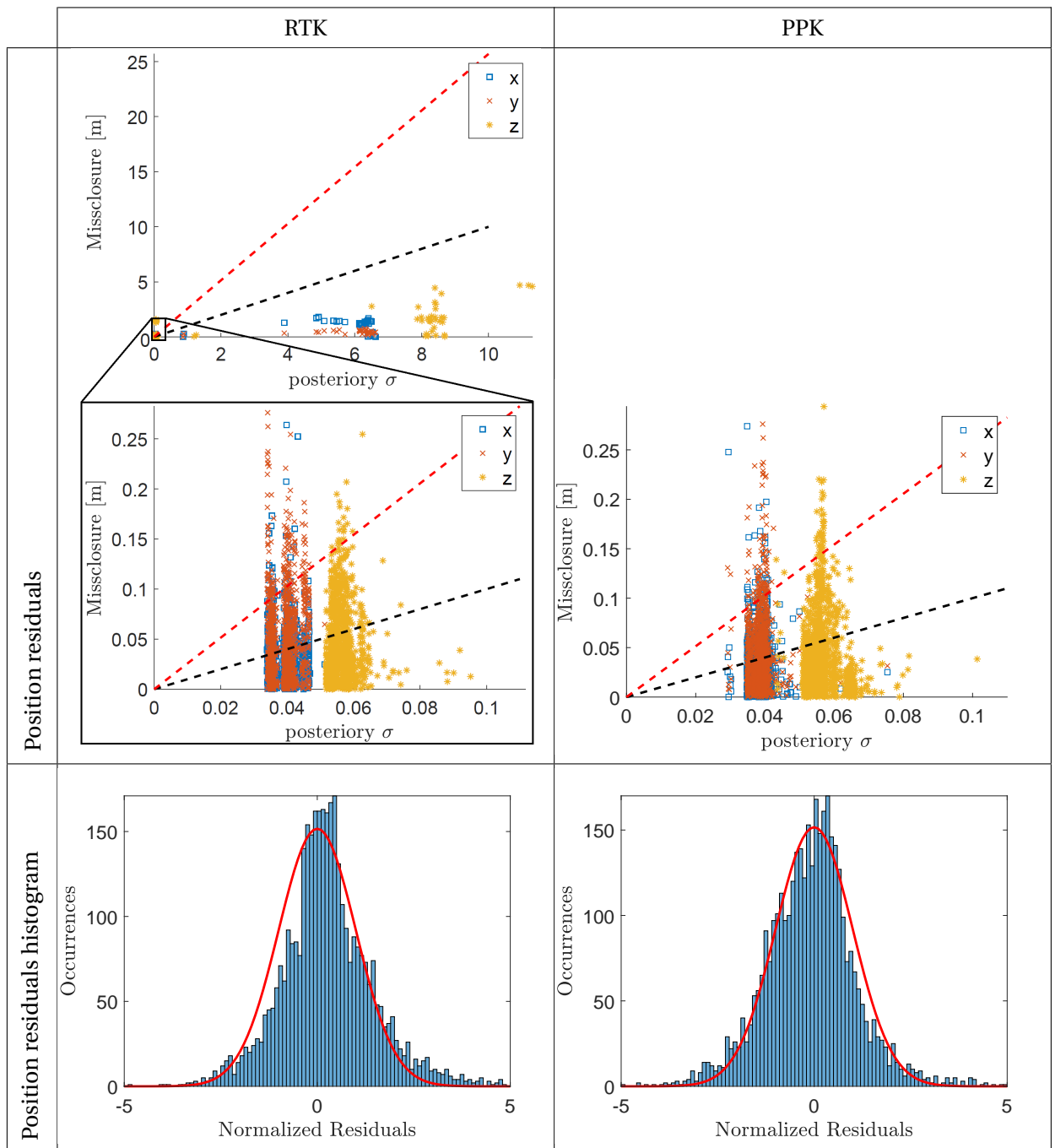


Figure 3.13. Comparison of GNSS positioning with the reference computed via indirect orientation with all GCPs. Black & red dashed-lines correspond respectively to  $1\sigma$  &  $2.57\sigma$ .

Further analysis involving other flights in the same zone compared the differences between

the photograph positions obtained by GNSS (RTK/PPK) with those of the indirect orientation using all GCPs. These airborne position misclosures are presented in the plots of Fig. 3.13. Note that the vertical component was generally worse than the horizontal one, as GNSS positioning is better in planimetry than in altimetry.

The correspondences between the residuals and posterior precision were quite strong for PPK, considering that the quality of the camera position derived via indirect positioning (the given precision of the camera pose center from the indirect orientation was considered in this statistical analysis) was not necessarily superior to that of GNSS and remained correlated with the other parameters (such as the attitude & interior orientation). This may explain certain residuals exceeding the normal probability levels.

Although such excess seems to be less present for the RTK positioning in the upper left part of Figure 3.13, note that the precision of this positioning mode was lower than that of PPK by a factor of more than 10 in numerous photographs. Indeed the RTK excess is very similar to that of PPK for *a posteriori*  $\sigma < 0.1\text{ m}$  as shown in the magnified plot. This suggests that caution should be exercised when employing RTK technology for achieving centimeter-level positioning in the airborne environment with frequent discontinuities in satellite tracking. Indeed, using PPK increases the time span for which the ambiguities are fixed as well as the likelihood of their correctness.

#### 3.4.3 Tie-point density

It is interesting to note that, with the exception of (close to) uniform texture (for example, dense and high vegetation; water), the predicted mapping precision appeared not to be very sensitive with respect to the number of tie-points, as observed empirically together with its expected precision. For example, an increase or reduction by a factor of two of the tie-point density led to a difference of several percent in the predicted mapping precision. This follows from Gruber's rule, which states that only five points are sufficient for the relative orientation of two poses of a calibrated camera [113]. The number of points is higher for small, non-metric cameras, but several tens of well-distributed points are sufficient. Thus, the benefit of the added points exhibits asymptotic behavior<sup>4</sup>. Although the selected inter-alpine environment of zones Z1 and Z2 included textural changes, from scree and grass to low and high vegetation, the observed variations in the point densities did not affect the orientation process. However, in certain areas with high vegetation, surface restitution was not possible (for example, the black areas within the internal part of Fig. 3.14).

---

<sup>4</sup>The paragraph 2.4.4.3.2. of [78] states that there is no improvement of the aerotriangulation results with the number of tie-points if this number have reached 20.

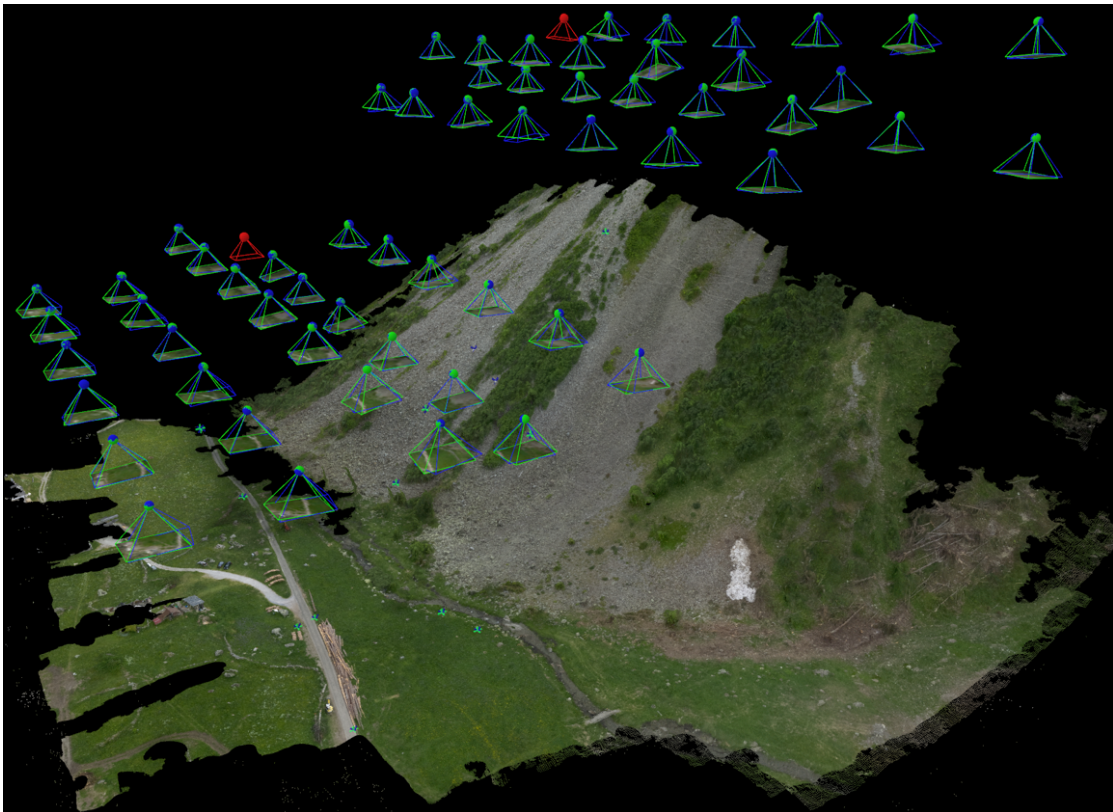


Figure 3.14. 3D model of Z2 created using commercial software (Pix4D). Approximated scale added manually.

#### 3.4.4 Local analysis

This subsection studies the mapping prediction of different flight configurations in both zones, namely *Z1* and *Z2*. In total, 26 scenarios, as described in Table 3.2, were derived from flights conducted above these two areas. In the *corridor* configuration, only two parallel flight lines were considered, where the so-called *mono-block* represents the “lawnmower sweeping” pattern with flying height adaptation per flight line, as illustrated in Fig. 3.14. A *stair plan* is a particular flight plan that staggers several smaller blocks. While the flying altitude remains constant within each block, it changes in a step among them. This plan is useful when mapping a complicated terrain shape, such as that of *Z2* (that is, a narrow and steeply climbing valley).

In total, 13 control points were determined in *Z1* and 20 were determined in *Z2*, which enabled different GCP/CP ratios and configurations to be considered. In the case of *no GCP*, all signalized points were considered as CPs. In the so-called *bad GCP* configuration, the barycenter of the GCPs differed substantially from the barycenter of the mapped area, while the remaining points were considered as CPs. However, the *good GCP* configuration considered

### 3.4. Experimental validation

the spread of GCPs over the entire terrain. To ensure that the evaluation was meaningful, the number of CPs was at least  $\geq 5$ .

For the aerial control, standard GNSS positioning ( $\sim$  meter-level accuracy) was considered in the standalone case, whereas carrier-phase corrections were employed in the RTK and PPK, with the aim of obtaining centimeter-level precision.

Zone 1 (Z1)		1	2	3	4	5	6	7	8
Flight plan	corridor	■	■	■	■	■	■	■	■
	mono block								
	Stair plan								
GCP configuration	No								
	Bad config	■	■			■	■		
	Good config			■	■			■	■
Aerial control	Standalone	■		■		■		■	
	RTK								
	PPK		■		■		■		■

Zone 2 (Z2)		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Flight plan	corridor	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
	mono block																		
	Stair plan																		
GCP configuration	No	■	■					■	■										
	Bad config			■	■														
	Good config					■	■				■	■						■	■
Aerial control	Standalone		■		■				■		■			■		■		■	
	RTK	■						■					■						
	PPK		■		■		■		■		■		■		■		■		■

Table 3.2. Tested scenarios in Z1 (top) and Z2 (bottom).

In each of these 26 scenarios, at every check-point, we compared the predicted accuracy of  $x$ ,  $y$ , and  $z$  with the actual values of  $|v_x|$ ,  $|v_y|$ , and  $|v_z|$ . The precision prediction could be represented in a color scale, as in in Fig. 3.7 for cases 5 and 7 of the bad and good GCP configurations, respectively. The plots on the first column of Fig. 3.15 present the results of a single experiment, namely case 13 of Z2, whereas the second column aggregates the values from every studied case. In the top plot of each of these figures, the predicted precision is represented by the abscissa axis, while the actual values of  $|v_x|$ ,  $|v_y|$ , and  $|v_z|$  are represented by the ordinate axis for the three coordinates. The solid lines represent the  $1\sigma$  and  $2.57\sigma$  bounds.

In the specific case 13 (Fig. 3.15 first column) as well as in all aggregated cases (Fig. 3.15 second column), 64% of the check-point misclosure coordinates belongs to  $[-\sigma, +\sigma]$  (where  $\sigma$  is the prediction precision of the check-points, represented by the black dotted line), 35% belongs to

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

$[-2.57\sigma, -\sigma[ \cup ]\sigma, 2.57\sigma]$ , and 0.76% belongs to  $]-\infty, -2.57\sigma[ \cup ]2.57\sigma, \infty[$ .

This demonstrates that the conducted experiments respected the basic properties of a normal distribution. However, global analysis needed to be performed to confirm such an assumption by means of additional statistical testing.

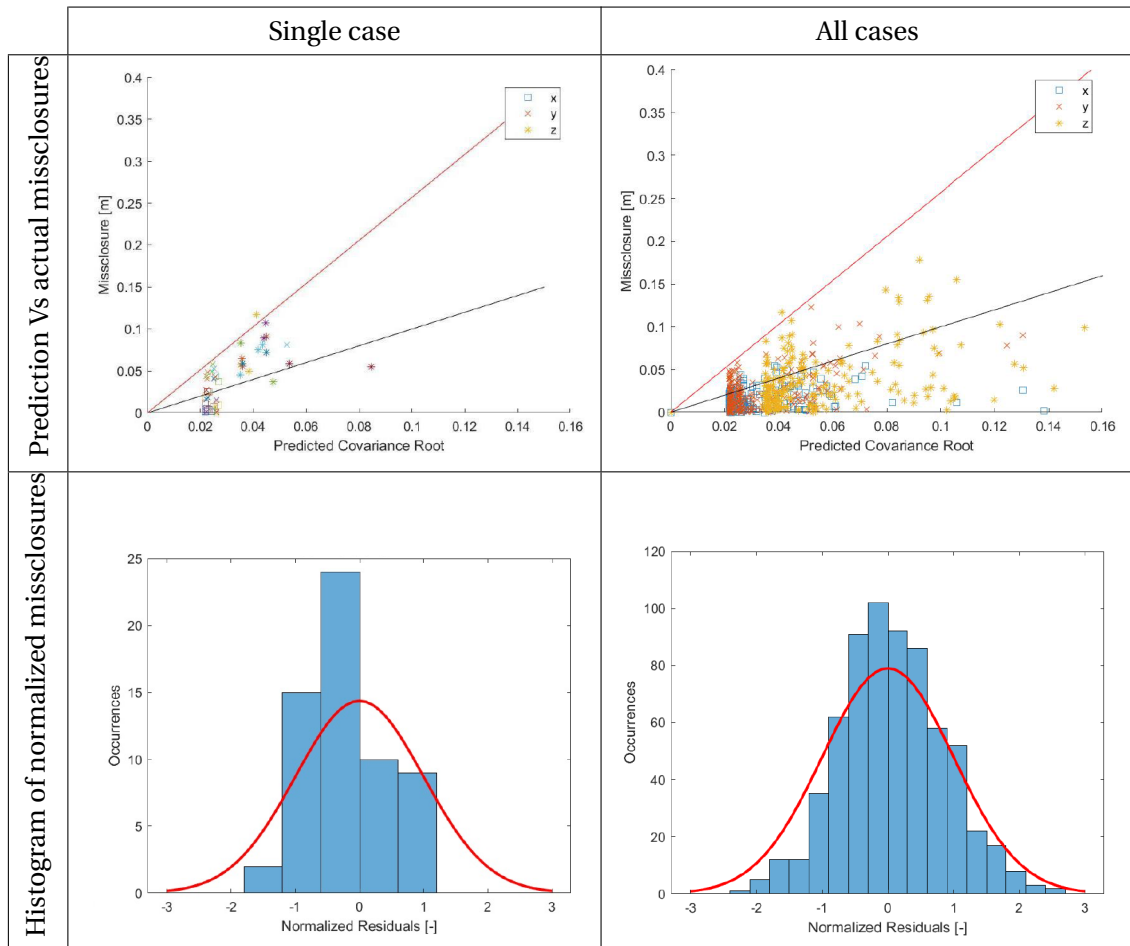


Figure 3.15. Comparison of GNSS positioning with the reference computed via indirect orientation with all GCPs. Black & red dashed-lines correspond respectively to  $1\sigma$  &  $2.57\sigma$ .

#### 3.4.5 Global analysis

The comparison for case 1 performed in  $Z1$  is detailed in Fig. 3.16. The prediction (indicated by the blue line) is the standard deviation evaluated along the eigenvector of the covariance matrix  $\Sigma_{cp}$ , computed as the square root of  $\Sigma_{cp}$  eigenvalues. The realization (indicated by the red line) is the projection of  $v$  onto the eigenvectors:  $v \cdot V_i$ .



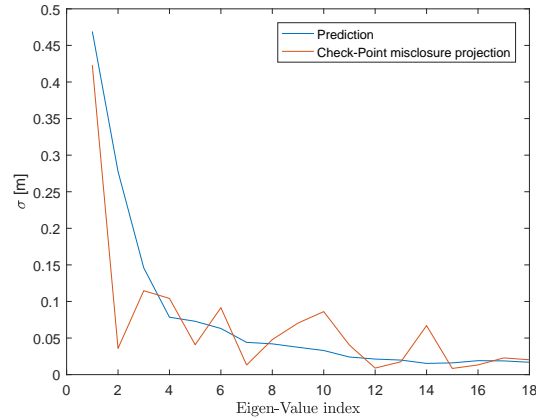


Figure 3.16. Global spectral analysis of mapping precision: predicted (blue) and actual (red) misclosure as a function of the Eigen-Value index of  $\Sigma_{cp}$

The predicted trend of decreasing the eigenvalues of  $\Sigma_{cp}$  was followed by the projection of the observed misclosures onto the corresponding eigenvectors. In terms of the deformation theory presented in Section 3.3.2, the misclosures were a linear combination of different deformation modes (schematically represented in Fig. 3.8), and in the studied case, the obtained coefficients were of the same order of magnitude as those calculated by means of the prediction.

As a similar agreement was observed in the other studied configurations, the observations were coherent with the predicted precision.

#### 3.4.6 New GCP placement

For the practical demonstration of the method relating to the optimal placement of the next GCPs (as presented in Section 3.3.3), case 5 in ZI was used, where images of an inclined terrain in block configuration were oriented via three (sub-optimally placed) GCPs (Fig. 3.17).

In this case, we were searching for the best place to situate an additional GCP. For this purpose, we employed the following quality indicators ( $a$ ,  $b$ ,  $c$ , and  $d$ ) and investigated different methods for the tie-point precision aggregation.  $a$  assess the tie-point with maximum 3D standard-deviation.  $b$  is the root mean squared (RMS) of all tie-points standard-deviation,  $c$  is the mean of tie-points 3D standard-deviation and  $d$  is the largest eigen-value of the full covariance matrix of the tie-points (and thus the highest mode).

$$a = \max_{TP_i} \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \quad (3.10)$$

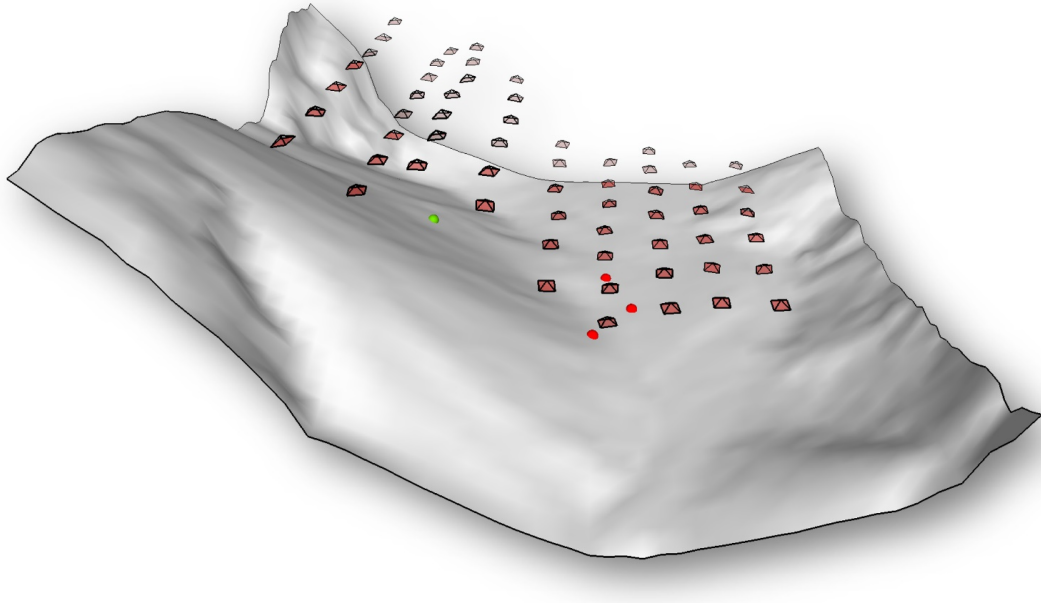


Figure 3.17. New GCP placement proposal (green point) with respect to other GCPs (red points).

$$b = \sqrt{\frac{1}{n_{TP}} \sum_{TP_i} (\sigma_x^2 + \sigma_y^2 + \sigma_z^2)} \quad (3.11)$$

$$c = \frac{1}{n_{TP}} \sum_{TP_i} \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \quad (3.12)$$

$$d = \sqrt{\max \lambda_{\Sigma_{xx}}} \quad (3.13)$$

We evaluated the quality indicators  $a$ ,  $b$ ,  $c$ , and  $d$  (3.10) to (3.13) with the three GCPs and placed them in the 1<sup>st</sup> column of Table 3.3.

Firstly, we searched for a tie-point with the maximum incertitude (indicator  $a$ ) and added the 4<sup>th</sup> GCP exactly at this location. The coordinates (with respect to the tie-point centroid) of this new GCP are indicated in the first two lines of Table 3.3 and the updated quality indicators are displayed in the column “4 GCP ←  $f(a)$ ”. As we will observe later, this strategy of selecting an adequate position for the 4<sup>th</sup> GCP was not appropriate. Secondly, we found a tie-point with

### 3.4. Experimental validation

	3 GCP	4 GCP $\leftarrow f(a)$	4 GCP $\leftarrow f(d)$
$X_{GCP(4)}(m)$	$\emptyset$	130	-56
$Y_{GPC(4)}(m)$	$\emptyset$	-32	-103
$a(mm)$	292	312	334
$b(mm)$	139	103	67
$c(mm)$	123	93	60
$d(mm)$	4179	2898	1241

Table 3.3. Improvement in mapping quality with new GCP placement.

	$a \rightarrow min$	$b \rightarrow min$	$c \rightarrow min$	$d \rightarrow min$
$X_{GCP(4)}(m)$	-16	-88	-88	-76
$Y_{GPC(4)}(m)$	-68	-148	-152	-120
$a(mm)$	<b>251</b>	313	324	325
$b(mm)$	80	<b>63</b>	64	69
$c(mm)$	74	58	<b>57</b>	60
$d(mm)$	1701	1160	1104	<b>1096</b>

Table 3.4. Benchmarking of improvement in mapping quality with new GCP placement via exhaustive search.

the maximum value in the eigenvector associated with the maximum eigenvalue of the global  $\Sigma_{xx}$ , and we placed the 4<sup>th</sup> GCP at this point. The coordinates of this point and the quality indicators computed using this new GCP are presented in the final column of Table 3.3.

Thereafter, we performed a type of benchmarking to validate the suggested methods. We created a  $4 \times 4$  m grid over the mapped surface and placed the 4<sup>th</sup> GCP at each node of this grid, evaluating the abovementioned quality indicators every time. We selected the grid place by minimizing  $a$ ,  $b$ ,  $c$ , and  $d$  separately and its coordinates are displayed in Table 3.4.

It can be observed from this analysis that, even if the quality indicator  $a$  is the most intuitive and straightforward, it should not be considered owing to its sensitivity to the random distribution of the tie-points. Hence, we suggest focusing on the quality indicators  $b$ ,  $c$ , and  $d$ . To compare the quality indicators computed with three GCPs and the same quality factor computed with four GCPs, we could define the following improvement factor:

$$\frac{\square_{4\ GCPs} - \square_{3\ GCPs}}{\square_{min} - \square_{3\ GCPs}}, \quad (3.14)$$

### Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment

---

where the *empty white square symbol*:  $\square$  could be replaced by any of the quality indicators  $a$ ,  $b$ ,  $c$  or  $d$ .  $\square_{3\text{ GCPs}}$  is the indicator calculated for three GCPs, whereas  $\square_{min}$  is the indicator calculated for the optimal position of the new GCP (in bold in Table 3.4). This ratio is defined as null if the new GCP is not usable; for example, if it does not appear in at least two images, and 100% if the GCP placement is at the best selection, as evaluated by the (time-consuming) grid search.

For any quality indicators  $b$ ,  $c$  or  $d$ , this improvement factor is approximately 45% for the second column of Table 3.3; that is, if a new GCP is placed on the tie-point with the maximum standard deviation. However, this improvement factor is approximately 95% for the third column of Table 3.3; that is, if 4<sup>th</sup> GCP is placed on the highest value of the eigenvector associated with the highest eigenvalue of  $\Sigma_{xx}$ . The quality evaluation that was performed using this exhaustive search demonstrated practically that the method for placing a new GCP, as described in Section 3.3.3, is nearly optimal (at a 95% level) with respect to the considered global mapping quality indicator  $b$ ,  $c$  or  $d$ .

## 3.5 Conclusions and perspectives

We have proposed a method for assessing the quality of drone mapping in a complex natural environment prior to the mapping being performed. The method is based on a flight plan and consists of simulating all of the observables required in the real mapping adjustment, considering a particular time and date, the observations, and the geometry of the photogrammetric network, using *a priori* knowledge regarding the terrain, drone payload, and GNSS satellites. Compared to the traditional approach, its application avoids the need for mission repetition frequency, which increases with the terrain complexity (Fig. 3.3 left vs. right).

The comparison of the quality prediction with the actual mapping accuracy in various geometrical configurations as well as the quality of the airborne GNSS positioning in the mountain environment exhibited satisfactory agreement in the sense that the statistical testing did not reject the hypothesis (at 5% significance) of the observed misclosures fitting the prediction.

Practical experience also demonstrated that the PPK approach is preferable over the RTK technology in an environment in which frequent occlusion of satellite signals occurs owing to either the drone motion or its surroundings. For the sake of navigation safety, the method of predicting the satellite positioning quality is worth considering not only at camera stations, but also (possibly with different criteria) over the entire drone trajectory, including the takeoff and landing zones.

The signalization and surveying of GCPs requires substantial effort in drone mapping, particularly in a complex terrain. Therefore, a method for the most effective placement of new

---

### 3.6. Appendix: Interpolation of estimated precision

GCPs was designed and evaluated. When actual realization of GCPs in the optimal location is not practical (for example, when the environment is too steep or is not accessible), the user may alter other geometrical or temporal parameters of the flight, and eventually change the equipment to maximize the likelihood that the executed mission will satisfy the desired precision criteria. Therefore, it is important that the prediction is not only correct, but also is executed rapidly and can offer intuitive interpretation. In our experience, the adopted metric of covariance interpolation and visualization satisfies this requirement.

In the future, we plan to develop an algorithm that can provide improved determination of the tie-point density and tie-point quality based on the (*a priori*) ground texture. This algorithm could make use of modern computer vision and machine learning. Moreover, the precision prediction could be extended to optimize the entire flight plan and overall GCP placement.

### 3.6 Appendix: Interpolation of estimated precision

#### Bilinear

The first (and naive) method for computing the precision of any point on a surface is to proceed to bilinear interpolation of the covariance matrix.

Let  $P_a$ ,  $P_b$ , and  $P_c$  be three vertices of a given triangle of the Delaunay triangulation and let  $P_d$  be a point inside this triangle. Then, there exist  $\beta$  and  $\gamma$  in  $[0, 1]$  with  $\beta + \gamma < 1$ , such that  $P_d = (1 - \beta - \gamma)P_a + \beta P_b + \gamma P_c$ .

The bilinear interpolation leads to the following:

$$\Sigma_{dd} = (1 - \beta - \gamma)\Sigma_{aa} + \beta \Sigma_{bb} + \gamma \Sigma_{cc}. \quad (3.15)$$

However, it is not mathematically correct to interpolate covariance matrices element wise.

#### Error propagation

[146] suggested performing error propagation from the three observed vertices of the triangle to the point  $P_d$ . Below is a modified version of the presented error propagation, taking into account the eventual correlations between the different points in a rigorous, fully 3D calculation.

$$\Sigma_{dd} = \begin{bmatrix} (1-\beta-\gamma)I_3 & \beta I_3 & \gamma I_3 \end{bmatrix} \cdot \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} & \Sigma_{ac} \\ \Sigma_{ab} & \Sigma_{bb} & \Sigma_{bc} \\ \Sigma_{ac} & \Sigma_{bc} & \Sigma_{cc} \end{bmatrix} \begin{bmatrix} (1-\beta-\gamma)I_3 \\ \beta I_3 \\ \gamma I_3 \end{bmatrix} \quad (3.16)$$

This algorithm was designed for estimating the precision of a grid cell center from a relatively regular and dense laser-scanning pattern. It is less suitable for randomly (and more sparsely) distributed tie-points, because it may overestimate the precision towards the middle of the triangles (when  $P_d$  does not necessarily lie in the perfect plane defined by  $P_a$ ,  $P_b$ , and  $P_c$ ).

### Log interpolation

As the covariance matrices are not directly associated with the conventional metric of Euclidean space, it is improper to proceed with linear combinations of their elements as previously suggested, although this appears to be practical. However, the *logarithms* of such matrices are part of Euclidean space [8]. Let the log of a covariance matrix be the matrix in which the eigenvalues are substituted by their logarithms. If  $\Sigma = V \Lambda V^T$ , where  $\Lambda$  is the diagonal matrix of the eigenvalues, and  $V$  is the matrix of the eigenvectors, we can define  $\log(\Lambda)$  as the diagonal matrix containing the logarithms of the diagonal elements of  $\Lambda$ , and  $\log(\Sigma) = V \log(\Lambda) V^T$ . Moreover, it is possible to demonstrate that  $\exp(\log(\Sigma)) = \Sigma$  using the power series definition of the exponential function. Therefore, the suggested interpolation of the covariance matrix logarithm is expressed as follows:

$$\Sigma_{dd} = \exp((1-\beta-\gamma)\log(\Sigma_{aa}) + \beta\log(\Sigma_{bb}) + \gamma\log(\Sigma_{cc})). \quad (3.17)$$

### Comparison

To assess the suitability of the methods for obtaining  $\Sigma_{dd}$ , we execute the algorithm that evaluates the covariance matrices of the tie-points. Thereafter, we consider three precision maps for display purposes, which are created by the three methods described above.

We run the code a second time to obtain different tie-point configurations, each with its predicted covariance matrix. This second prediction enables validation of the interpolation method selected for the first prediction. For each tie-point of the second prediction, the

### 3.6. Appendix: Interpolation of estimated precision

---

covariance matrix will be compared to the value provided by the interpolated map based on the first prediction. There are several manners in which the  $3 \times 3$  covariance matrices can be compared. One of these is the relative LERM [8]. The relative LERM of the matrices  $\Sigma_1$  and  $\Sigma_2$  is expressed by the formula below, where  $\|\bullet\|_F$  is the Frobenius norm (the square root of the sum of the squared elements of the matrix).

$$\frac{\|\log(\Sigma_1) - \log(\Sigma_2)\|_F}{\|\log(\Sigma_1)\|_F} \quad (3.18)$$

It is possible to compute the relative horizontal error and relative vertical error, which are defined by the following two equations.

$$\frac{\sigma_{1xy} - \sigma_{2xy}}{\sigma_{1xy}} \quad (3.19)$$

$$\frac{\sigma_{1z} - \sigma_{2z}}{\sigma_{1z}} \quad (3.20)$$

To aggregate the statistics computed for all tie-points, we compute the maximum difference, the mean of these differences, and the square root of the mean of the squared differences in Table 3.5. In both criteria presented in this table, it can be observed that the logarithmic interpolation is the most appropriate method for our problem, as it allows for interpolating the covariance matrix at any position, *a fortiori*, with the smallest error.

**Chapter 3. Mapping quality prediction for RTK micro-drones operating in complex environment**

---

		Bilin.	Error prop.	Log bilin.
$max \bullet$	relative LERM	41%	37%	29%
	relative $\sigma_{horiz.}$	185%	127%	81%
	relative $\sigma_{vert.}$	207%	128%	100%
$\frac{1}{n} \Sigma \bullet$	relative LERM	1%	13%	1%
	relative $\sigma_{horiz.}$	2%	29%	1%
	relative $\sigma_{vert.}$	2%	29%	2%
$\sqrt{\frac{1}{n} \Sigma \bullet^2}$	relative LERM	3%	14%	2%
	relative $\sigma_{horiz.}$	9%	31%	4%
	relative $\sigma_{vert.}$	11%	31%	6%

Table 3.5. Comparison of three covariance interpolation methods.



# Photo-LIDAR Fusion **Part II**



## 4 Fusion of photo with airborne Laser Scanning

Chapter 1 provided the background theory for photogrammetry and introduced LIDAR measurements. Chapter 2 and 3 exploited the Bundle-Adjustment in different configurations to simulate and process images, but not LIDAR observations. This chapter -originated from the following manuscript- will present a method to handle LIDAR data in the Bundle Adjustment.

E. Cledat and J. Skaloud, Fusion of Photo with Airborne Laser Scanning *ISPRS Annals Nice*, 2020

### Abstract

Photogrammetry and Laser-Scanning are usually considered as complementary. Integration of these two observation methods has the potential to blend their individual advantages. The resulting benefit is likely to be higher in drone airborne mapping, which payload capacity (and thus the quality of the embedded IMU) is limited. Thus, the trajectory computed by the IMU is subject to important *time-dependent* errors: even if the global attitude is less adequate, it is self-coherent locally. For this reason, we propose a *close* integration of Photogrammetry with Laser-Scanning based on the correction of time-dependent error of the trajectory with the help of the image observations acquired by the camera. Apart from the trajectory, this hybridization requires optical correspondences between image and Laser measurements. Such full set of input data is rigorously fused together in a Bundle-Adjustment in order to better determine the trajectory, and thus the resulting point-cloud. The presented theory was practically evaluated in an airborne case against a reference solution.

### 4.1 Introduction

Photogrammetry and Laser-Scanning have been successfully used for countless mapping applications such as land usage analyses, cultural heritage site preservation, civil engineering infrastructure inspections, and so on. The exponential development of small drones<sup>1</sup> over the last decade has allowed for the transposition of plane and helicopter airborne photogrammetry into drone airborne photogrammetry, with the advantages of mapping small and inaccessible areas at a reduced cost [122], [129]. More recently, the miniaturization of LIDAR sensors has allowed them to also be embedded in micro-UAVs (e.g. Velodyne Puck [163] and Riegl Vux [135]).

The two mapping methods, Photogrammetry and Laser-Scanning can be seen as complementary (Table 4.1). This complementarity opens potential for the method of fusion to improve the final mapping product, in aspects as geometric precision, radiometric precision and exhaustiveness<sup>2</sup>. The complementarity could also be advantageous in further analyses, such as segmentation, classification and object recognition. In particular, several decades of experience in image processing ([54], [53], [3], [4], [154], [2]) has enabled the possibility to automatically recognize and segment geometrics features such as polygons. The precise and rigorous knowledge of the images IO and EO with respect to the points acquired by the LIDAR sensor could open a wide range of new possibilities in segmenting the point-cloud thanks to the images, and thus help the 3D CAD model reconstruction.

Data from several sensors are said to be *loosely coupled* when substantial pre-processing is performed separately for each sensor before integration. Data are said to be *closely coupled* when the data are fused together at an earlier stage. In the scope of Photo-LIDAR fusion, [67] proposes a *loosely coupled* integration as the traditional photogrammetric processing chain is independent from the LIDAR one; the point-cloud generated via photogrammetry and the one from LIDAR are merged only after each is created separately. Photo-LIDAR *closely coupled* data integration is possible, however, and permits the use of the benefits from one method to help the other, and vice-versa. In particular, LIDAR measurements are taken one after the other, where each measurement represents a single point of an object and is considered independent from the next point, as the sensor moved between the subsequent measurements. Conversely, photos taken by global shutter cameras of photogrammetry give a coherent representation of [a sub-set of] an object at a given time. Direct geometric relationships can then be established between different points visible on the same image. This global coherence of any single image can give coherence both to the reconstructed 3D model and to the trajectory of the platform.

---

<sup>1</sup>MAV: Micro Aerial Vehicles, or micro-drone, usually < 3 – 5 kg.

<sup>2</sup>Exhaustiveness stands for the minimalization of holes in the model due to occlusion. See [97] for a didactic explanation of the concept of occlusion in Laser scanning and stereo-occlusion in photogrammetry. See [40] for a visualization of the acquired points both by dense-matching photogrammetry and Laser scanning.

Photogrammetry (Passive sensor)	Laser scanning (Active sensor)
+ Sufficient for indirect orientation	– Direct Sensor orientation dependent
+ Precise in planimetry	
– Less precise in altimetry	+ Precise in altimetry
– Needs overlap to create 3D: (sensitive to object occlusion)	+ Direct measurement of 3D points (less sensitive to object occlusion)
– Sensible to illumination	+ Insensible to illumination
– Sensible to object texture	+ Insensible to object texture
– Problematic in high vegetation	+ Penetrates tree canopy

Table 4.1. Comparison Photogrammetry/LIDAR

Forms of *closely coupled* integration are proposed in [97], [61], [60], [59] and [58]. In addition to these approaches, there is a possibility to adapt Bundle-Adjustment to interfere image observations (raw data of photogrammetry) with LIDAR raw observation. This requires establishing *links* between raw observations of both acquisition types. The principle and the weakness of *links* used in the works cited above will be discussed in section 4.2 together with the proposition of a stronger type of *link*.

The motivations for *closely* integration photogrammetry with LIDAR are as follows. a) IMU adapted for micro-UAV are small & low-cost, and generally not sufficiently precise for direct orientation of the platform in the scope of laser scanning. b) However, the trajectory is locally coherent i.e. the relative orientation of temporally close values is precise. c) Thus, the local *shape* of the trajectory described under point b) could be used under two well-oriented photos.

The paper contributions are organised as follow. First, different forms of optical correspondences are reviewed in terms of proposed sensor hybridization, where the goal is to improve the global redundancy and accuracy. Second, a rigorous interpolation is presented between subsequent photos and inertial-derived relative orientation. This will allow, along with model description, the rigorous merge of the whole set of observations into the Bundle Adjustment: (Section 4.3). This method will then be tested on low quality IMU data and compared to high-quality results: (Section 4.4).

### 4.2 Review of optical correspondences

#### 4.2.1 Tie-points: Link image-image

The prerequisite to Photogrammetry is the ability to recognize similar points (or other types of geometric features) between images representing the same real object. The coordinates of these points on the images are the *image observations*. The matching between a point observation on one photo and a corresponding point observation on another photo makes this point a *tie-point* (first column of table 4.2). This *link* between the observations is needed to produce the 3D position of the point itself, EO and IO<sup>3</sup>.

#### 4.2.2 Coplanarity: Link LIDAR plane-LIDAR plane

In the scope of airborne LIDAR, the first link between observation sets acquired at different times is the *plane to plane link* [144] and [69]. There, the goal was to link planes representing the same building roof from different flight-lines to calibrate the boresight-matrix and possibly other parameter in LIDAR and IMU sensors (column 2 of table 4.2). The weakness of constraining two planes to be coplanar is that one could slip on the other, hence, planes of different slope and aspect needs to be present.

#### 4.2.3 Point to Patch: Link LIDAR point-LIDAR pointcloud

The plane to plane method described in sec. 4.2.2 has been generalized in [79] to generic surfaces. The established correspondence aims to constraint a point acquired by the laser scanner during one flight line (blue point of column 3 of Table 4.2) to a surface defined by several points acquired during another flight-line (red point-cloud of column 3 of Table 4.2).

#### 4.2.4 Homologous point: Link LIDAR point-LIDAR point

Two LIDAR points taken at different moments could also be matched (represented by a double arrow in the third column of table 4.2). The naïve method of selecting them manually in a point cloud is not rigorous since the object surfaces sampling is somewhat random due to the nature of acquisition.

To input a match between two LIDAR points that are geometrically close but temporally spaced, the detected difference vector must be considered relative to LIDAR EO at the time of

---

<sup>3</sup>The External Orientation (EO) of a camera refers to its position  $T$  and orientation  $R$  (by extension, the position and orientation of the platform). The Internal Orientation of a camera refers to the function  $\zeta$  needed to compute the theoretical image observation from viewing-ray vector (by extension, all parameters describing the sensors embedded on the same platform, and their mounting one with each-other: lever-arm & boresight matrices).

measurement of one of the two points.

The following methods (1-4) permit the matching of LIDAR points.

1. [73] suggests to first rasterize the pre-processed LIDAR point-cloud, then apply raster-based point detection and matching algorithm such as SIFT.
2. Target could be placed by the operator on the ground, and be detected within point-cloud.
3. Cloud-to-cloud registration algorithms such as ICP can be applied on two small point-cloud samples representing the same object, in order to find a correspondence. Adapted versions of the ICP algorithm have been used in [59] and [58] for data adjustment.
4. [62] proposes performing the cloud sample registration with a neural network.

### 4.2.5 Tie-Point to LIDAR point: Link Photo-LIDAR

Close photogrammetric-laser scanning integration needs *links* between photo and LIDAR data. This link could be established between a point acquired by the LIDAR and the corresponding 2D point in an image. The methods previously described for homologous point registration can be adapted, in particular for the use of rasterized point-cloud [83].

Another method to match a 3D point with its corresponding 2D point on a photo is by using building roof vertex corners. This shape is characterized by the exact intersection of three planes (usually oblique). In the pre-processed point-cloud, the three planes could be fit with robust algorithms and then intersected. In the image, the 2D point is the over-intersection of three lines.

### 4.2.6 Tie-Point to LIDAR point cloud: Link Photo-LIDAR

[61] proposes to match tie-points (or points from photos computed by dense matching) to the LIDAR point-cloud. The proposed *link* aims to constrain the tie-point to be on the surface described by the LIDAR point-cloud (for clarity in the following explanation, we will consider this surface to be close to horizontal). This constraint acts perpendicularly to this LIDAR point cloud surface, i.e. vertically (represented by a double arrow in the last column of Table 4.2). The tie-point could thus *slip* horizontally on this surface.

Due to the intersection geometry, a tie-point in airborne photogrammetry is generally more precise in planimetry than in altimetry. This implies that altimetry could easily be modified by other information, such as the constraint of lying on a (close to) horizontal surface. The

## Chapter 4. Fusion of photo with airborne Laser Scanning

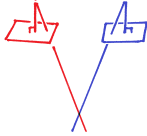
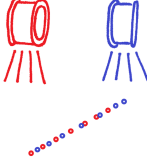
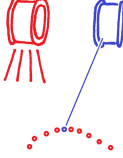
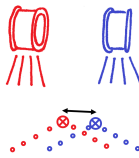
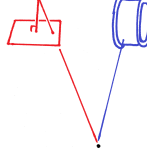
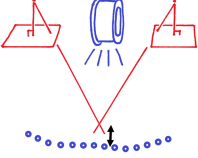
4.2.1 Image- Image	4.2.2 Plane-Plane	4.2.3 Point-Patch	4.2.4 Point-Point	4.2.5 2D Point-3D Point	4.2.6 Tie-Point - Point-cloud
					
SIFT, SURE, ORB, KAZE, etc.	(robust) plane-fit	Point to closest surface	<ol style="list-style-type: none"> <li>0. <del>Manual point selection</del></li> <li>1. rasterize LIDAR point-cloud and apply SIFT, SURE, etc.</li> <li>2. target</li> <li>3. ICP</li> <li>3. Plane intersection</li> <li>4. Neural Network</li> </ol>		Dense matching

Table 4.2. Different types of *links* for Photogrammetry, Laser-Scanning and Photo-LIDAR fusion

corollary of this property indicates that this constraint does not bring much information. The image-point to LIDAR-point *link* 4.2.5 is thus preferable for photo-LIDAR fusion because it contains more information.

### 4.3 Methods and models

The *links* described in section 4.2 will be used as additional constraints in a Bundle-Adjustment in order to determine the most probable values of the so-called *parameters*, describing the mapping process. The choice of the *parameters* is an extremely important part of Bundle-Adjustment design (4.3.1) together with observation models 4.3.2, 4.3.3, 4.3.4 and 4.3.5 is the relation between the measurements (or observations) to the parameters.

#### 4.3.1 Parametrizing variables

The first parameters to be considered are the 3D points on the ground, as well as other terrain-object modelling parameters. For example, the primitive geometrical feature of point  $P$  on Figure 4.1).

The trajectory could be described by the position and the orientation of the IMU at each



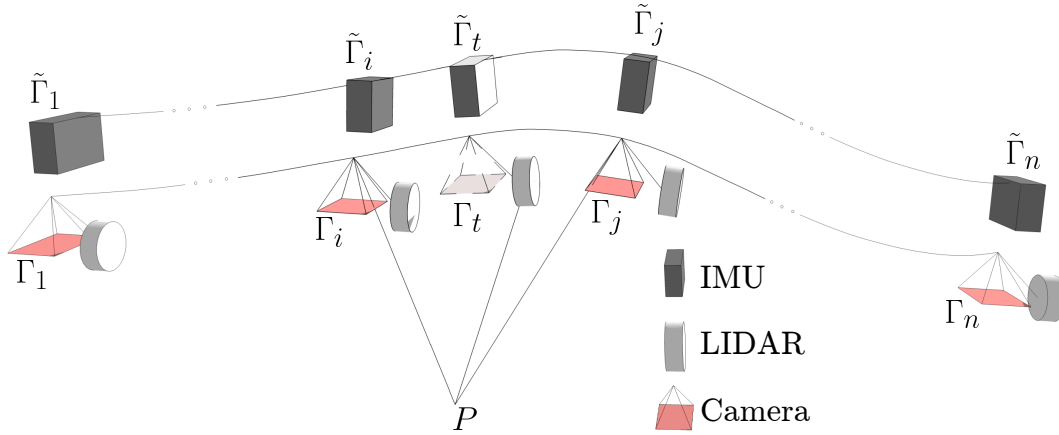


Figure 4.1. Description of the complete trajectory based on camera pose parameters only. No photos are taken at time  $t$ .  $\Gamma_t$  is not a parameter but could be computed from  $\Gamma_i$  and  $\Gamma_j$  via interpolation of double differences

IMU measurement. However, this will lead to a tremendously high number of parameters to determine, and thus to a high complexity. We propose to parametrize the trajectory only by using the position and the orientation of the camera while a photo is taken.

The position  $T$  (camera perspective center in local frame) and the orientation  $R$  (from camera to local frame) of the camera are aggregated into the matrix  $\Gamma \in \mathbb{SE}_3$  using homogeneous formalism (Equation 4.1).

$$\Gamma = \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

The entire trajectory has been pre-processed using the INS/GNSS integration (top of Figure 4.1). This pre-processed trajectory  $\tilde{\Gamma}$  is subject to time dependent errors that must be corrected in the Bundle-Adjustment. This pre-processed trajectory will act as a measurement between two successive poses by virtue of relative orientations (Sec. 4.3.3), and allows the determination of the position and the orientation  $\Gamma_t$  of the camera at any time  $t$  between two poses. This  $\Gamma_t$  is related to the position and the orientation of the LIDAR sensor by the lever-arm and boresight matrix encapsulated in the  $\mathbb{SE}_3$  matrix  $\Gamma_{bs}$ .

### 4.3.2 Camera model: collinearity equation

The camera model is based on a corrected pinhole camera model: a tie-point  $P$  projects<sup>4</sup> to the image observation  $\ell_{tp}$  on a photo.  $\Gamma$  describes the position and orientation of the camera at the time of the photo.

$$\ell_{tp} = \xi \left( \pi \left( \hat{\Pi} \Gamma \begin{bmatrix} P \\ 1 \end{bmatrix} \right) \right) \quad (4.4)$$

### 4.3.3 Aerial control

The position measurement acquired by the embedded GNSS antenna (or INS/GNSS integration point) must be translated to the camera with the lever-arm  $\vec{a}$  (from GNSS antenna phase center or IMU-centre respectively to camera perspective center in camera frame).

$$\ell_{GNSS} = \hat{\Pi} \Gamma^{-1} \begin{bmatrix} \vec{a} \\ 1 \end{bmatrix} \quad (4.5)$$

The IMU position and orientation  $\tilde{\Gamma}$  computed by the pre-processed IMU trajectory can also be input as measurements by virtue of relative measurements [131].  $\tilde{\Gamma}_i$  and  $\tilde{\Gamma}_j$  correspond to IMU pre-processed trajectory for two consecutive poses  $i$  and  $j$ , whose camera pose parameters are  $\Gamma_i$  and  $\Gamma_j$ .

---

<sup>4</sup>The collinearity equation 4.4 needs a so-called *projection matrix*  $\hat{\Pi}$ , whose aim is to remove the last unitary coordinate used by homogeneous coordinates formalism.

$$\hat{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.2)$$

The projection function  $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$  models the pinhole camera model while the function  $\xi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  models the camera interior orientation: effect of the principal distance, principal point, skewing parameters, radial and tangential distortions.

$$\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \mapsto \frac{1}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \quad (4.3)$$

Note that this method of relative orientation permits the removal of the boresight matrix between the camera and the IMU.

$$\tilde{\Gamma}_i^{-1} \tilde{\Gamma}_j = \Gamma_i^{-1} \Gamma_j \quad (4.6)$$

The relative observations are weighted accordingly to their precision computed with [131].

#### 4.3.4 LIDAR model

While images are acquired at discrete moments (low frequency  $< 1 \text{ Hz}$ ), LIDAR measurements are acquired continuously (high frequency  $> 20 \text{ kHz}$ ). The measurement  $\ell_{LIDAR}$  acquired at time  $t$  must be associated with the pose determination  $\Gamma_t$ . The pre-computed IMU position/orientation determination from IMU measurements  $\tilde{\Gamma}_t$  must be corrected with the knowledge of the camera pose preceding (of index  $i$ ) and succeeding (of index  $j$ ) the LIDAR observation. The ratio  $\tau \in [0, 1]$  quantifies the time difference between the lidar measurement event and the image  $i$ . It is null when the lidar measurement time coincides with photo  $i$  and one when the lidar measurement time coincides with photo  $j$ .

$$\tau = \frac{t - t_i}{t_j - t_i} \quad (4.7)$$

This ratio  $\tau$  permits the interpolation<sup>5</sup> of the double-differences of relative orientations com-

<sup>5</sup>The interpolation of  $\mathbb{SE}_3$  matrices are performed in the tangent-space of  $\mathbb{SE}_3$  denoted  $\mathfrak{se}_3$ . The  $\left[ \cdot \right]_{\otimes}$  operator transforms a  $\mathbb{R}^6$  vector into an element of the tangent-space  $\mathfrak{se}_3$ .

$$\left[ \begin{array}{c} t \\ \omega \end{array} \right]_{\otimes} = \left[ \begin{array}{c} t_x \\ t_y \\ t_z \\ \omega_x \\ \omega_y \\ \omega_z \end{array} \right]_{\otimes} = \left[ \begin{array}{cc} [\omega]_{\times} & t \\ \mathbf{0}_{1,3} & 0 \end{array} \right] = \left[ \begin{array}{cccc} 0 & -\omega_z & \omega_y & t_x \\ \omega_z & 0 & -\omega_x & t_y \\ -\omega_y & \omega_x & 0 & t_z \\ 0 & 0 & 0 & 0 \end{array} \right] \quad (4.8)$$

The function  $\log$  is defined as one of the reciprocal function of  $\expm \left( \left[ \cdot \right]_{\otimes} \right)$  where  $\expm$  is the function exponential for square matrices:  $\expm(M) = \sum_{n \in \mathbb{N}} M^n$  with  $M^0 = I$ . A method to compute  $\log$  is proposed in [149]

puted from the camera EO and IMU measurements.

$$\Gamma_t = \Gamma_i \cdot \expm\left(\tau \left[ \log\left(\Gamma_i^{-1} \Gamma_j \tilde{\Gamma}_j^{-1} \tilde{\Gamma}_i\right) \right]_{\times}\right) \tilde{\Gamma}_i^{-1} \tilde{\Gamma}_t \quad (4.10)$$

The lidar measurement  $\ell_{LIDAR}$  (position of the point in the LIDAR sensor frame) could be expressed in the frame of the camera due to the boresight/lever-arm matrix  $\Gamma_{bs}$  from the camera to the LIDAR sensor. This  $\Gamma_{bs}$  could be known from previous calibration, or re-determined in the Bundle-Adjustment.

$$\ell_{LIDAR} = \tilde{\Pi} \Gamma_{bs} \Gamma_t \begin{bmatrix} P \\ 1 \end{bmatrix} \quad (4.11)$$

This approach is inspired by [60] and [59], but is more rigorous as the 6 components of the trajectory (3 position and 3 orientation) are considered together (using lie-group formalism and theory).

#### 4.3.5 GCPs

The GCP observation model is trivial as it relates directly the parameters  $P$  of a given point on the ground to a direct measurement of this same point with terrestrial independent methods (terrestrial GNSS, tacheometry, etc.).

$$\ell_{GCP} = P \quad (4.12)$$

#### 4.3.6 Bundle-Adjustment

The real measurements related to the observation models presented in 4.3.2, 4.3.3, 4.3.4 and 4.3.5 are subjects to inherent errors due to the sensor and measurement processes.

---


$$\forall \Gamma \in \mathbb{SE}_3, \expm\left(\left[\log(\Gamma)\right]_{\otimes}\right) = \Gamma \quad (4.9)$$

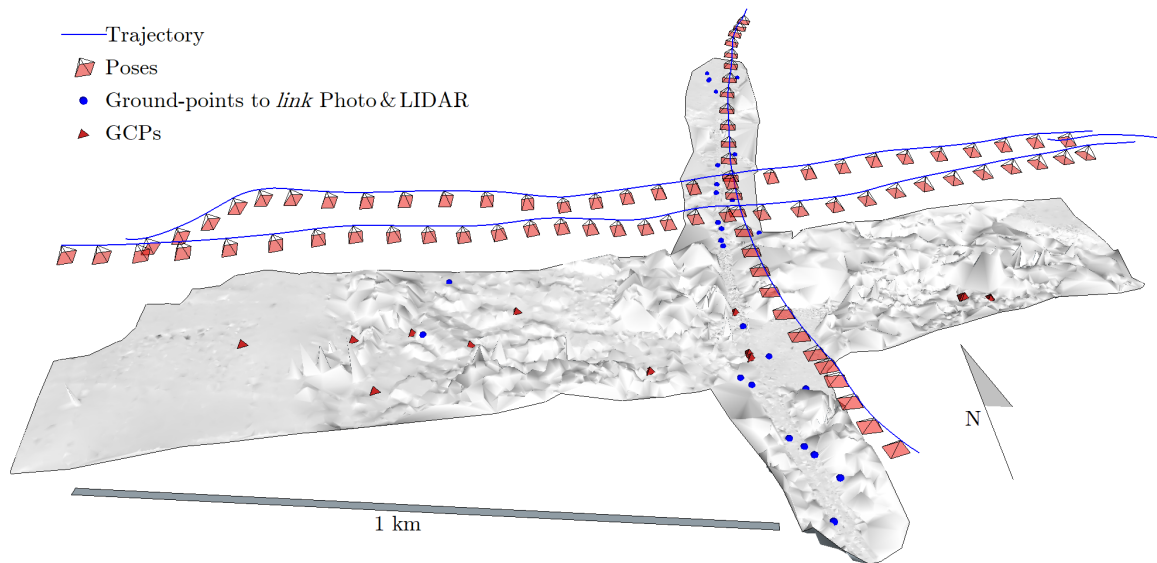


Figure 4.2. Test-site. Red triangles represent the GCPs, blue spheres represent *links* between Photogrammetry and LIDAR i.e. point on the ground measured both by LIDAR and on photos

The observations are going to be confronted with each other in an optimization process called *Bundle-Adjustment* in order to determine the most probable value of the parameters. In the scope of this work, the least-square criterion has been used, but other criterions (more robust ones for example) could be used.

## 4.4 Results

A typical application for the proposed method of LIDAR/Photo data fusion is airborne mapping. The following test on orientation improvement with respect to low-cost INS-GNSS attitude determination has been set up in order to study the benefit of photo-LIDAR links. We propose the following practical benchmark. We embark high performance aerial control and navigation sensors on the same platform as low-cost IMU (UAV-LIDAR) sensors and fly them on a helicopter at altitude and speed mimicking UAV flight. Then, we study the gain on orientation with respect to the reference. We hypothesize that there is more room for progress for LIDAR sensor miniaturization than for IMU. This justifies the use of the same LIDAR sensor for experimenting and ground-truthing, while two different IMU will be used.

The chosen test site is situated in a hilly area at the corner of five towns: Bremblens, Romanel-sur-Morges, Aclens, Vufflens-la-Ville and Bussigny (VD, Switzerland). It presents a variety of terrain types: bare ground, crops, forest, and industrial buildings, and it is also crossed by roads, train railways and a high-tension power line.

## Chapter 4. Fusion of photo with airborne Laser Scanning

---

The N-S extension of the mapping area is 1.6 *km* while the E-W extension is 1.7 *km* (Figure 4.2).

The sensors were embedded on a helicopter as described in [32]. The flight velocity was about 13 *m/s* (min: 8 *m/s*, max: 18 *m/s*) which is compatible with the velocity of a quadcopter or a fix wing drone. The flight height was between 250 *m* and 300 *m* above ground. The following subset sensors was used in this study.

- Camera: Phase One frame digcam IQ180 with a sensor of  $10328 \times 7760$  *pix*. The field of view is  $52^\circ \times 44^\circ$  [123].
- Laserscanner: Riegl VQ480U, using a rotating polygon mirror technology. The datasheet characteristics are 25 *mm* of accuracy with a Laser Beam Footprint of 9 *mm* at 300 *m* [134].
- IMU: IXblue AIRINS navigation grade IMU. The datasheet characteristics are  $0.01^\circ/hr$  for the Drift, and  $0.005^\circ/\sqrt{hr}$  for the noise [72].
- GNSS receiver: Javad TRE-G3T dual frequency and muti constellation<sup>6</sup> receiver.
- IMU: a low-cost MEMS-IMU (NavChip V1/2011, Thales) mounted on a gecko board [81]. The datasheet characteristics for the drift in run bias stability is  $15^\circ/hr$  while switch-on bias variation is not specified. The noise is estimated to be  $0.3^\circ/\sqrt{hr}$ .

---

<sup>6</sup>GPS, GLONASS and Galileo were used, although other constellations are possible in a firmware option

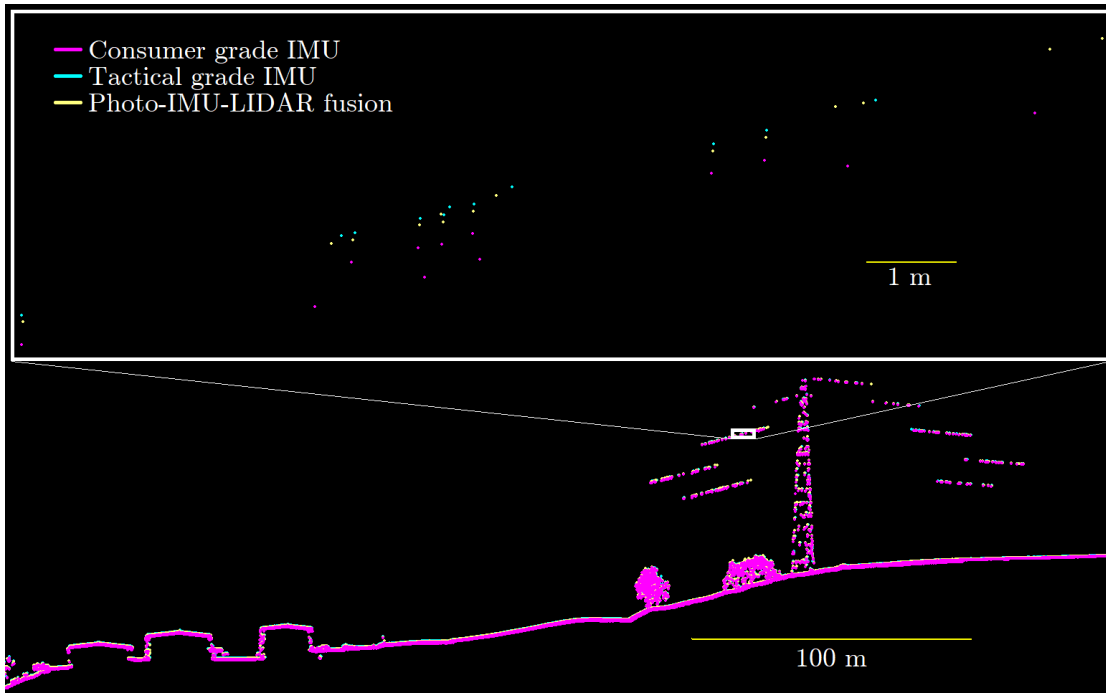


Figure 4.3. Point-Cloud section centered on a powerline. *Ground truth* point cloud (blue) has been computed using the Tactical Grade IMU trajectory. The point-cloud resulting from fusion of photogrammetry and LIDAR (yellow) uses consumer grade IMU and should be compared to the point-cloud using the same IMU without fusion (pink).

*Ground Truth* point-cloud (in blue on Figure 4.3) has been computed from LIDAR data using LIEO [143] and the trajectory given by the navigation grade IMU (AIRINS) together with GNSS measurements. The same method has been used to compute the point-cloud that would have been produced with the consumer grade IMU (in pink on Figure 4.3).

This generated point-cloud originated directly from the INS/GNSS trajectory, is subject to time-dependence errors as analysed in [32] and [161]. Figure 4.3 represents the point-cloud acquired on an overlap zone: the same object has been measured in two different flight-lines (thus, at two different moments in time). The *ground truth* point-cloud is coherent (blue), while the point-cloud generated with the consumer Grade IMU shows that the same object duplicated (pink).

The reference point-cloud data permits the simulation of 27 *links* between LIDAR measurements and image observations (corresponding on the terrain to the blue dots on Figure 4.2). In a practical application, these *links* would have been determined based on the methods described in 4.2.5. Observations from the camera, the LIDAR, GNSS and consumer grade IMU have been processed together in a Bundle-Adjustment reinforced with these *links* as

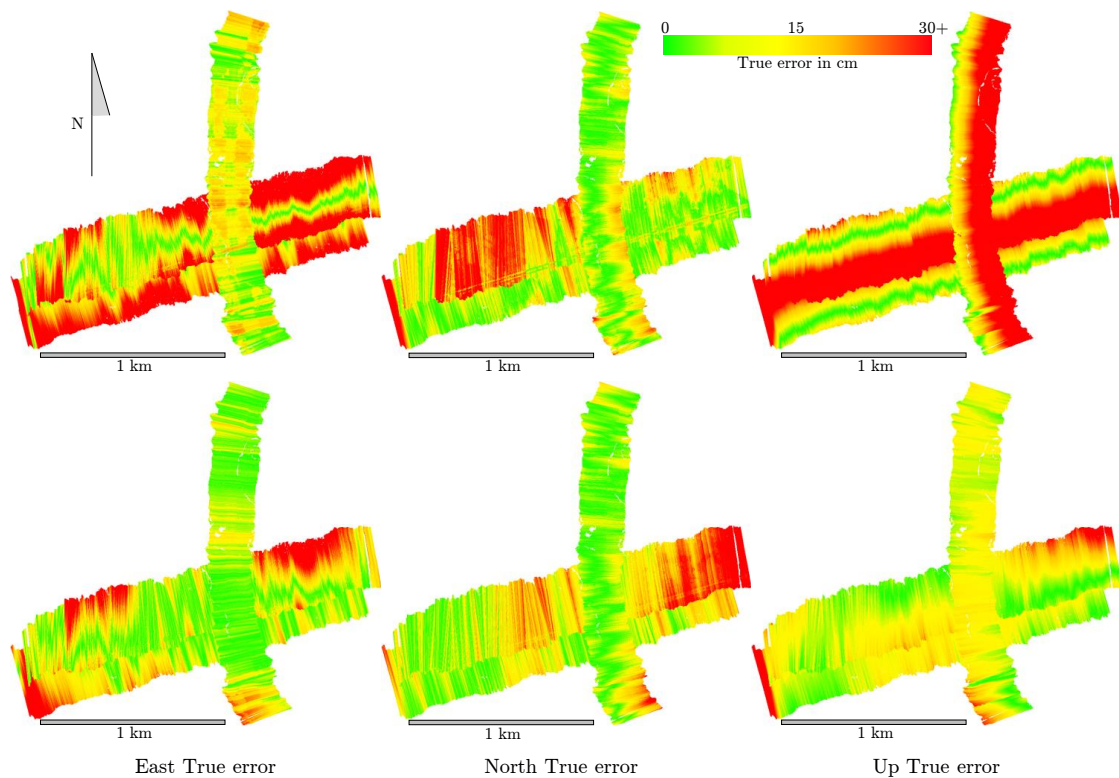


Figure 4.4. Difference between the generated point-cloud and the *Ground Truth* in East, North and Up produced using the trajectory of the consumer Grade IMU without (first row) and with (second row) fusion with photogrammetry.

additional observations and 11 GCPs. The resulting point-cloud (in yellow on Figure 4.3) is self-coherent, unlike the one generated without fusion of photos (in pink on Figure 4.3). It has been compared to the *Ground Truth*. Figure 4.4 displays the absolute errors in point-cloud due to orientation errors of low-cost IMU (1 uncalibrated unit) after INS/GNSS fusion. The adjusted point-cloud (after fusion with photo) is displayed in the lower portion of the same figure. It can be seen that on the overlapped areas, the orientation errors were mitigated, so the vertical (and also the planimetric) errors are reduced below 15 *cm*. The errors in resulting DTM are likely to be even lower (< 10 *cm*) due to the effect of averaging.

The benefit of the fusion with photogrammetry is quantified by comparing the distance between the *Ground Truth* and the produced point-cloud, both without (first row of Figure 4.4) and with (second row of Figure 4.4) photogrammetric fusion. The improvement ratio of the fusion can be defined as the ratio between the *true-error* of the results of the fusion and the *true-error* of the results without fusion (where the *true-error* is computed as the distance from *Ground Truth*). This ratio is equal to 1 if the fusion brings no improvement, is below 1 if the



fusion worsens the results, and is above 1 if the fusion mitigates the orientation errors. The improvement ratio varies with the location on the terrain. For example, on the area shown on Figure 4.3, the improvement ratio is approximately 5. Figure 4.5 is the histogram of the ratio for each LIDAR points of the survey. The quantile at 0.5 % of the ratios is  $1/2$  and the quantile at 99.5 % is 7. Thus, 99% of the ratios belong to the interval  $[1/2 - 7]$ . The ratio is inferior to 1 only for less than 7 % of the points. Indeed, these points for which data-fusion worsens the results belongs to areas which are far from any Photo-LIDAR *link*. The geometric mean of all ratios is 2. This shows that globally, the fusion between photogrammetry and LIDAR greatly improves the precision of the final point-cloud. Further analysis has been proceeded with the trajectory computed from a SIMU (Synthetic IMU) composed from 4 calibrated IMU as described and analyzed in [32] and [161]. The improvement ration of the fusion of SIMU with LIDAR and photo is inferior (1.75) to the improvement ration of a single IMU with LIDAR and photo. The benefit of photo-LIDAR fusion decreases as the quality of the IMU increases.

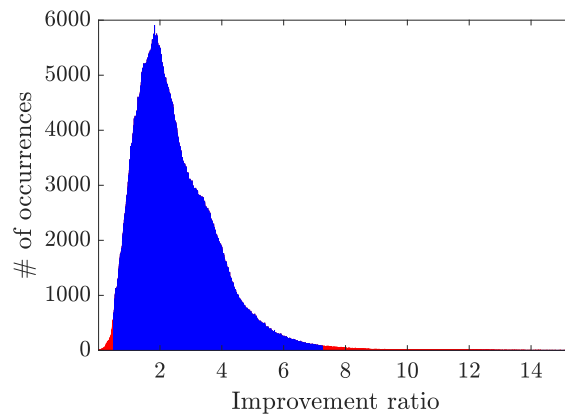


Figure 4.5. Histogram of improvement ratio for all acquired points. 99 % of the values are represented in blue.

## 4.5 Conclusion

We have proposed a new method for GNSS, IMU, LIDAR and photo data fusion utilising strong correspondences between photo and LIDAR as additional observations in a Bundle-Adjustment. The additional information provided by photogrammetry permits the improvement of the absolute trajectory determination over the period of time between photos during which relative IMU observation are self-coherent, and thus builds a point-cloud with smaller geometrical deformations. Experiments are performed on real data, but with simulated correspondences between photo and LIDAR observation (*links*), showed that even a small number of these *links* can globally improve the absolute 3D point-cloud precision by a factor of 2. With the long awaited miniaturization of LIDAR sensors, the proposed method could have an

important impact on drone LIDAR mapping by reducing IMU volume and weight, improving the geolocalization and thus increasing the area that can be mapped.

### 4.6 Future outlook

This study raises the need for accurate automated *links* between photo and LIDAR data. It focuses on points but the presented fusion method could be extended with different *links* between Photo and LIDAR based on other geometrical such as lines. Indeed, a line could be detected either in a pre-processed point-cloud (via plane intersection) or with images (as in Figure 4.6 from [34]). [83], [30] and [155] give the theory for line-based photogrammetry while [126] proposes a method of image registration with respect to point-cloud of a particular type of line: Contour Cues. A preliminary study of line-based Photo-LIDAR fusion has been presented in [34].

The presented approach is based on a pre-processing of IMU data to generate a trajectory. It aims to correct the time dependent trajectory errors up to a certain extent. The integration of IMU data could be more closely achieved from Photos and LIDAR data with the help of Dynamic Network [43].

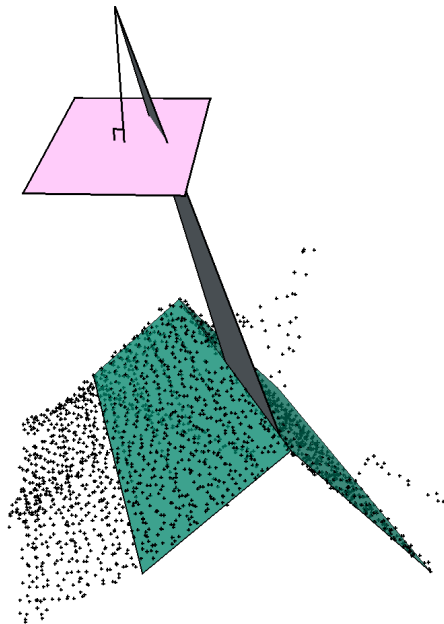


Figure 4.6. Line as a *link* between Photo and LIDAR data

Finally, this method could be applied using different camera mounts (for example, an oblique camera or an horizontal camera could help determining the azimuth of the system), different types of camera (fish-eye camera) on different platforms (terrestrial mobile mapping handle

by robots or humans).

#### 4.7 Appendix: Precision assessment of plane intersection method for Image-LIDAR matching

Points determined via plane-intersection are well defined, in particular if the intersection is favorable (when the plane intersection is close to a right-angle). The aim of this annex study is to assess the precision of such method to georeference images on a LIDAR point-cloud. The point cloud representing the city of Fribourg (FR, Switzerland) has been acquired by LIDAR with GNSS and tactical grade IMU embedded on a helicopter. In this study, the generated point-cloud is considered as a ground-truth. The aim of this study is to georeference the 536 photos taken by a camera embedded on the copter without any priors. The covered area is  $\approx 2.2 \text{ km}^2$  with an average Ground Sampling Distance (GSD) of  $3.7 \text{ cm}$ . For this task, 66 GCPs have been created from the LIDAR point-cloud. The first method to create these GCPs was manual (interesting points have been selected by an experimented user in a point-cloud visualizer software, first column of Table 4.3). The second method is based on plane intersection (third column of Table 4.3). The planes have been fit with L1 norm minimization, and then intersected. If more than 3 planes intersect (maximum 4), the intersection point have been determined by least-square estimation.

## Chapter 4. Fusion of photo with airborne Laser Scanning

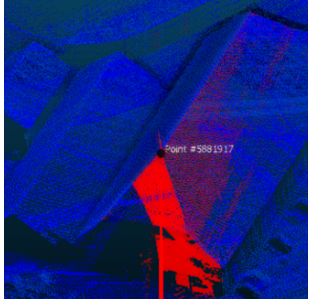

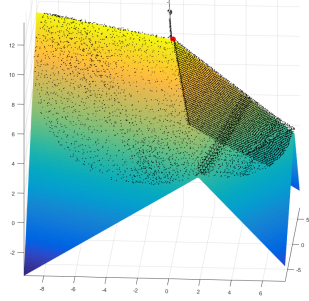
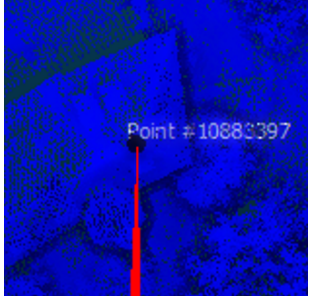

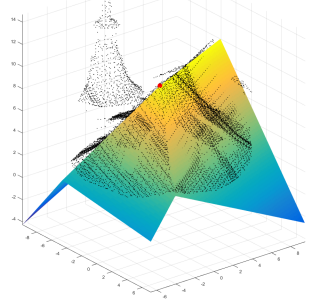
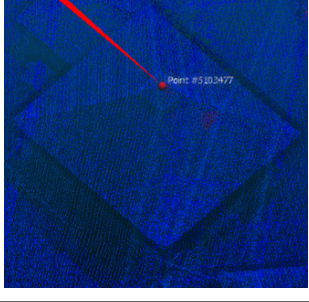
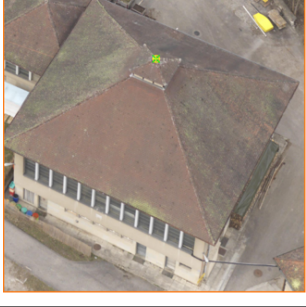
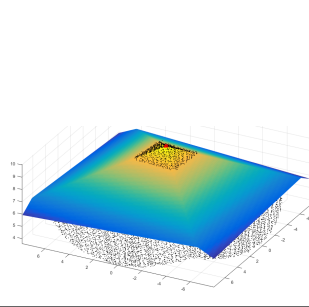
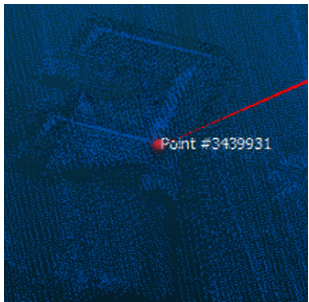
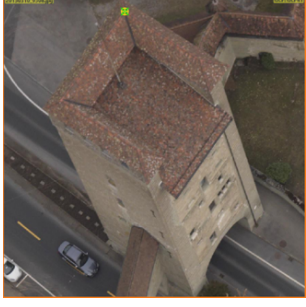
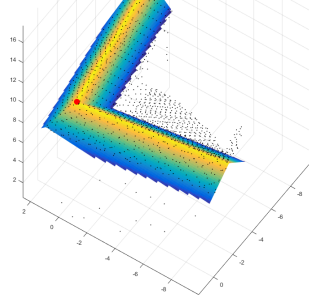
Manual selection	Point on photo	Plane (over-)intersection
		
		
		
		

Table 4.3. 4 examples of photo-LIDAR matches

After photogrammetric processing, the GCPs residuals are computed as the difference between the 3D coordinate extracted from the point-cloud and their adjusted values. Their histograms

#### 4.7. Appendix: Precision assessment of plane intersection method for Image-LIDAR matching

are displayed on Table 4.4 (first row). In order to improve the results, the GCPs generating the highest residuals have been removed. Only the best 87% have been kept (second row of Table 4.4).

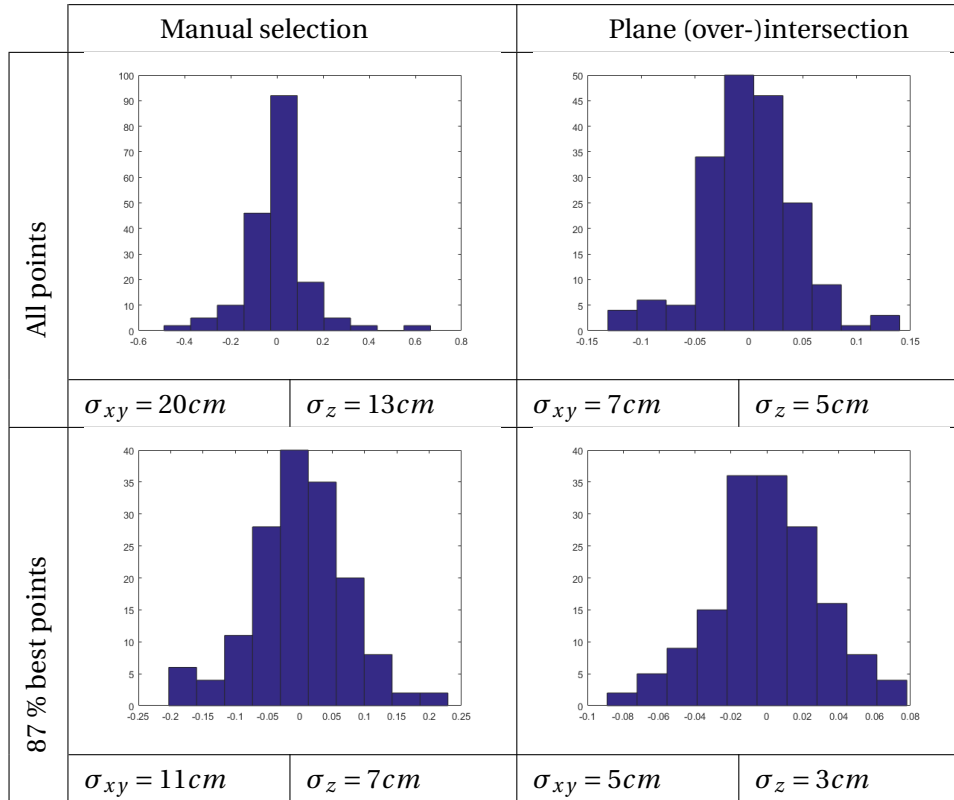


Table 4.4. Performance evaluation

The suppression of the 13% worse points improve the precision only by a factor of 1.7 whereas a rigorous determination of the GCPs coordinates by plane intersection permits an improvement of the precision by a factor of 2.5.



# Collaborative mapping **Part III**





## 5 Mapping GNSS restricted environments with a drone tandem and indirect position control

The first two parts of this thesis study photogrammetric and LIDAR methods with sensors embedded on a single platform. The aim of this last part is to consider collaborative methods with two platforms. This chapter in particular focus on two UAVs forming an aerial-aerial mapping tandem. This chapter is originated from the following publication.

E. Cledat and D. A. Cucci. Mapping GNSS restricted environments with a drone tandem and indirect position control. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-2/W3, pp. 1–7, 2017

The contribution of E. Cledat was mainly to suggest the concept and to adapt the photogrammetric simulation software presented in Chapter 3 to predict the expected precision while the contribution of D. A. Cucci was to prove the technical feasibility of this idea.

This original publication permits to launch the DoDo project: DrOne Duo for mapping GNSS denied area, founded by InnoSeed ENAC grant and leads to the results described in [75] and [148].

### Abstract

The problem of autonomously mapping highly cluttered environments, such as urban and natural canyons, is intractable with the current UAV technology. The reason lies in the absence or unreliability of GNSS signals due to partial sky occlusion or multi-path effects. High quality carrier-phase observations are also required in efficient mapping paradigms, such as Assisted Aerial Triangulation, to achieve high ground accuracy without the need of dense networks of ground control points. In this work we consider a drone tandem in which the first drone flies outside the canyon, where GNSS constellation is ideal, visually tracks the second drone and provides an indirect position control for it. This enables both autonomous guidance

and accurate mapping of GNSS restricted environments without the need of ground control points. We address the technical feasibility of this concept considering preliminary real-world experiments in comparable conditions and we perform a mapping accuracy prediction based on a simulation scenario.

### 5.1 Introduction

Unmanned Aerial Vehicles (UAVs) are becoming an important tool for surveyors, engineers and scientists as the number of cost-effective and easy-to-use systems is increasing rapidly [39]. These platforms nowadays offer an alternative to conventional airborne mapping every time small or cluttered areas have to be mapped with centimeter level resolution. Many successful applications have been reported, such as in repetitive surveys of buildings, civil engineering structures or construction sites, land monitoring and precision farming.

One important limit of current UAV technology is the dependency on GNSS coverage. Indeed, mapping missions are typically planned offline defining a set of waypoints in terms of absolute coordinates; the autopilot then closes the position control loops employing the position observations from a GNSS receiver (manual control is possible, but the quality of the overlap between the photos is more difficult to ensure). We cite the *eBee Plus* platform [142], from senseFly Ltd, a market leader in drones for professional applications, for which its ground control segment does not allow to take off if the GNSS reception is degraded. While certain platforms could also be flown in manual mode, the actual improvement in mapping productivity comes with a high degree of platform autonomy, as less qualified personnel is required and the scale of the operation can be wider.

The dependency on the GNSS reception limits the applicability of UAV based mapping in many interesting scenarios, such as natural and urban canyons, in which the sky is in large part occluded by natural or artificial structures. In these situations the quality of the constellation geometry is poor and severe multi-path effects can occur, introducing shifts in the position fix that could result in crashes, making GNSS based navigation extremely risky. In the worst case it is even impossible to compute the position fix. Examples of such sites, which require regular inspection for assessment, safety and renovation planning, are mountain roads, bridges, rock-fall protection galleries, dams, see Figure 5.1.

One very active research topic in UAVs and, more in general, in robotics regards the development of visual-only or visual/inertial navigation systems which would allow to guide autonomous platforms in an unknown environment without the dependency on the GNSS coverage. Despite the number of promising solutions published in scientific venues, see for instance [50], the technology readiness level of such systems is still rather low, and no such general system is implemented in commercial products. One reason is that it's practically



Figure 5.1. Rockfall protection structures and bridges in a 300 m deep gorge (Viamala, Thusis, Switzerland), where the GNSS reception is absent or unreliable for autonomous UAV guidance.

impossible to formulate guarantees about the performances of such navigation systems.

Even if such GNSS-independent navigation systems were available and well performing in arbitrary environmental conditions, high quality GNSS carrier-phase measurements are still required to perform high accuracy photogrammetry. Indeed, the far most common approach to image orientation in UAVs, Aerial Triangulation (AT), also referred as Indirect Sensor Orientation (ISO), is solely based on image observations, yet the process of establishing a dense network of ground control points (GCPs) is required to ensure global orientation and 3D pointing accuracy, especially in cluttered scenarios where a satisfactory overlap between the photos is sometime difficult to achieve. The process of establishing ground control is extremely time and money expensive in absence of GNSS coverage, as conventional topographic methods based on total stations have to be put in place. Second, the topology of such scenarios can make the accessibility of certain areas very impractical and even dangerous for the operators, see again Figure 5.1.

It is a well known fact that the requirements on GCPs can be eliminated in image-block scenarios if precise absolute or relative aerial control is introduced in the bundle adjustment (even if this aerial control is not operating for some parts of the flight), in the so called Assisted Aerial Triangulation (AAT) fashion [130, 105, 48]. Indeed, the recent evolution of GNSS antenna technology enabled the usage of multi-frequency and multi-constellation GNSS receivers on board of commercial MAVs [102, 142] and integrate the derived “geo-tags” (i.e., aerial position

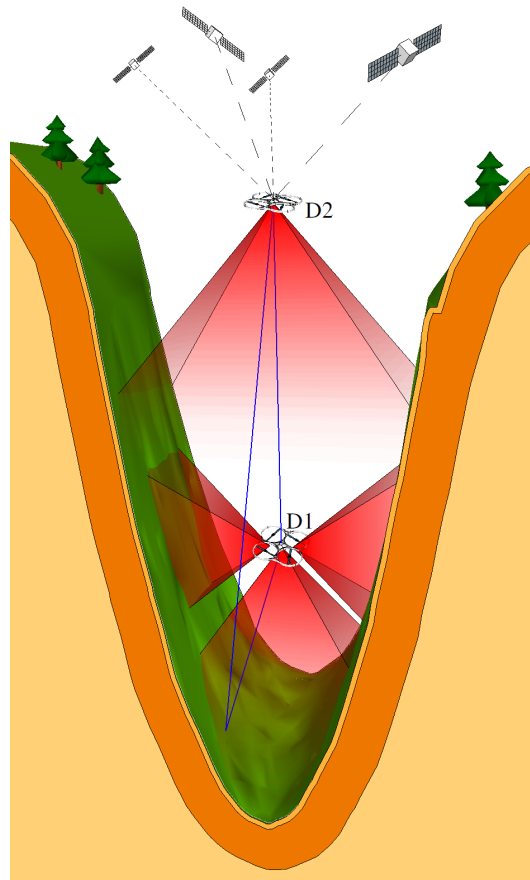


Figure 5.2. Schematic representation of the proposed method. Red shading represents field of view of the cameras embedded on D1 and D2 drone, blue lines represent image measurements, black dotted lines represent phase GNSS observation.

control) within the established processing software, e.g., [151].

In this work we propose a novel mapping concept, based on two UAVs, that enables the autonomous acquisition of aerial images in cluttered environments where the GNSS reception is degraded, such as deep gorges, natural and urban canyons. The first drone flies above the canyon where the GNSS reception is good. The second drone autonomously flies in the gorge employing position observations provided by the first drone. These are determined in real-time by tracking multiple optical signaling devices (e.g., high power LEDs) mounted on the second drone. Via the concept of indirect position control, the proposed mechanism also allows to georeference the aerial images taken by the second drone, and thus enables accurate mapping without the need of establishing dense networks of ground control points.

The idea of cooperative mapping is not new in the literature, yet it is often focused on strategies to divide the work and perform it in parallel [9, 87]. Cooperative localization instead consists

in having a tight link between the mapping robots that permits them to achieve a shared notion of each one's position. In [159] three terrestrial robots are equipped with cameras and an optical target and move in a so-called "Leap-Frog" pattern: one robot is moving while the other two are staying stationary, then, the role of the robots is exchanged. This path permits to build a triangulation network similar to the ones used for mapping entire countries with theodolites in the nineteenth century [90]. This cooperative principle is used for terrestrial robots, for example in [98] where olfactory sensors (air quality sensors) are embedded on the robots, for underwater vehicles [101] and for a team of UAVs [63]. In this last case, if the precision of the positioning is not satisfactory, one UAV could land, and act as a fixed beacon. [125] raises the problem of the complexity of dealing with a numerous team of cooperative robots.

Recently [166] introduced a hierarchy between the robots. Certain robots (called leaders) have better localization capabilities and higher quality sensors and can assist the robots which do detailed mapping (child robots) in localization. Such hierarchy exist also in the mapKITE project<sup>1</sup>, where tactical grade navigation instruments are placed on a terrestrial vehicle, along with an optical target. This target permits to track the moving terrestrial vehicle from an UAV and to enhance its aerial mapping accuracy [42, 108].

In this work we build on cooperative localization ideas and propose a solution to replace GNSS signal both in real-time, for guidance and in post-processing, for accurate mapping without ground control points. After presenting in detail the concept, in Section 5.2, we will discuss how the main technical difficulties could be tackled based on real world preliminary experiences. In Section 5.4 we will present the results of mapping accuracy predictions using different flavours of indirect position control in a conventional bundle adjustment scenario. We conclude the paper with some remarks and hints towards the real implementation.

## 5.2 Indirect Position Control

In this work we propose a novel mapping system suited for operations in cluttered outdoor environments where natural or artificial structures occlude the line-of-sight to GNSS satellites. The system is based on two UAVs, refer to Figure 5.2. The first one, from now on referred as D1, performs the actual mapping mission, acquiring high resolution nadir and possibly side aerial images. D2 carries high accuracy navigation sensors. It follows D1 and it provides position observations for D1 in real-time. D2 also captures nadir images to be used in post-processing along with the ones acquired by D1. A detailed description follows.

D2 flies in line of sight with respect to D1, typically, but not necessarily, above it. D2 flies high enough such that no environmental structure occludes the sky and the GNSS constellation

---

<sup>1</sup>"mapKITE: EGNOS-GPS/Galileo-based high-resolution terrestrial-aerial sensing system".

## Chapter 5. Mapping GNSS restricted environments with a drone tandem and indirect position control

---

is ideal. The payload of D2 includes a high grade INS/GNSS navigation system, such as, for instance, the SPAN-IGM-A1 [114]. Such systems nowadays weight around 0.5 kg and they are suitable for rotary-wing UAVs. The position and the orientation of D2 are thus available with high precision in real-time (RTK GNSS can be employed, but it is not necessary). The payload of D2 also includes a high resolution machine vision camera to acquire nadir images, store them, but also make them available to be processed by an on-board companion computer.

Multiple high power LEDs are mounted in a known, asymmetric, 3D pattern on the upper part of the D1 frame. These LEDs are visible from very high distance in camera images, as we will show later on, and are robustly identifiable with simple image processing algorithms. As the 3D LED pattern is known, the relative position and orientation of D2 with respect to D1 can be determined solving the Perspective-n-Point problem [167]. For this, the intrinsic camera calibration parameters must be known, yet, as we will discuss later on, the quality of such calibration is not determinant for the real time processing.

Once the relative position of D2 with respect to D1 is known, the absolute position of D1 can also be determined in real time: we compose the absolute position and orientation of D2 given by the INS/GNSS navigation system with the relative information from the visual tracking system. The solution is then transmitted to D1 which uses it as a position observation in the autopilot navigation filter, as if it was computed by a conventional GNSS receiver. This is what we call *indirect position control*.

Once an absolute position fix is available, D1 can perform waypoint based navigation, and thus execute a conventional mapping mission autonomously. Such a mission can be planned beforehand by means of a 3D mission planning software, such as [56]. D1 is equipped with conventional nadir camera suited for UAVs, such as the Sony NEX-5, as in [145]. Whereas the nadir camera is required, as it will become clear in the following, a side camera can be optionally installed in case the user wants to map facades or slopes, see again Figure 5.2. A low-cost IMU can also be installed on D1 and it provides *relative* attitude control in post-processing, as in [16], as long as some robustness in case of temporary loss of position fixes from D2.

In order for this concept to work, D2 has to follow D1, such that D1 is always in line-of-sight. This is critical as if the line-of-sight is lost, also the position fix for D1 is lost, possibly leading to accidents. The simplest strategy is such that D2 generates for itself a stream of waypoints always on the vertical of D1. D2 could also send commands to D1 to control the execution of the mission plan, such as pause it, or abort, in case for instance line-of-sight is at danger or speed is too high.

Once the mapping mission has been performed, data has to be post processed in order to obtain the final mapping products. In the following we propose a post-processing strategy

that can be performed with the currently available commercial software.

As a first step, the INS/GNSS raw data from D2 is fused by means of an offline Kalman smoother, such as the one available in commercial INS/GNSS processing software, as POSPac [7]. This gives centimeter level position (GNSS raw observations are processed in carrier-phase differential mode) and orientation for D2, the quality of which depends on the available IMU.

Next, the two streams of nadir images, from D1 and D2, are processed together for automatic tie-point detection. There will be thus two kind of matches: i) features that are matched only between images belonging to the same stream (i.e., only seen by the D1 *or* D2), and, ii) features that are matched in both streams, or, in other words, features that are identified at least in an image from D1 *and* in an image from D2. Matches of type ii) are the ones that allow to transfer the global position control between D1 and D2, which we call off-line *indirect position control*.

Image observations from D1 and D2, and absolute position and orientation control for the D2 ones, obtained from INS/GNSS (we assume that images from D2 are time-tagged via the GNSS receiver) are then combined in a conventional bundle-adjustment software capable of Assisted Aerial Triangulation (AAT). This step yields the nadir mapping products.

As we will discuss in Section 5.4, there are cases in which a limited number of common tie-points is available between D1 and D2 images. In this case, the precise image positions of the signaling devices fixed on D1, in D2 images, can be also introduced in the bundle-adjustment, as extra collinearity observations. Also, relative orientation control obtained pre-processing D1's IMU should be considered, as described in Chapter 1, which may require custom adjustment software.

Once the positions and the orientations for the D1 nadir camera are known, they can be used as position and orientation control for the D1 oblique cameras, once the proper boresight and lever-arm have been applied. This allows to run the conventional Assisted Aerial Triangulation (AAT) pipeline for these images as well. Nadir and side images can also be processed together for increased accuracy, provided that the bundle-adjustment software can handle boresights and lever-arm between different cameras.

The proposed mechanism allows to perform autonomous mapping missions in environments that are intractable with the currently available technology. We will discuss certain critical, yet technical details in the next section. The proposed adjustment scheme also allows to obtain accurate georeferenced mapping products even in the absence of absolute position control for D1. In Section 5.4 we will discuss different adjustment scenarios and we will derive conclusions regarding the precision that can be expected for both mapping products.

### **5.3 Technical Feasibility**

Here we discuss possible issues and point towards technological solutions that have worked in the past in similar scenarios.

#### **5.3.1 Tie-points Matched in Both D1 and D2 Nadir Images**

As presented in Section 5.2, indirect position control from D2 to D1 is obtained when the same environmental feature is seen from both UAVs' nadir camera. As D2 alone can accurately georeference world features seen in its own images via AAT, these points can act as ground control points for D1, if they are also seen in D1's nadir images. Thus, the key for indirect position control is that enough image points are correctly matched between D1 and D2 nadir images.

To confirm that such matches are possible and indeed common, even though images are captured from different elevations and orientations, we examine the tie-points extracted with Pix4D mapper in a standard, UAV based, photogrammetric flight over a rural area. See Figure 5.3. North-South flight lines are flown at an elevation of 150 m, while East-West ones at 190 m. The average GSD was 4.55 cm. A total of 1885 usable tie-points were extracted, out of which 1746 (92.63%) were seen from both elevation, while only 139 (7.37%) were matched in one image stream only. The density was 130 tie-points per hectare.

From Figure 5.3 it is possible to see that common tie-points are approximately uniformly distributed in the considered area (the red dashed polygon) and that there is no area in which these points are missing. We recognise that the considered flight depicts a nearly-optimal case, and that the elevation difference between crossing flight line may not reflect the one needed in the environments considered in this work. In the following we will consider a much lower percentage of common tie-points and we will show how the proposed method can work in much more degraded scenarios.

#### **5.3.2 Visual Tracking and localizing of D1 from D2**

[75] describes an algorithm for tracking the 8 LEDs in the image of the camera embedded on D2, and for determining the position of D1 in the camera frame of D2. The LEDs of D1 are mounted on an asymmetrical pattern such that the distances between the LEDs could reach 65 cm. The distance between the two drones were up to 100 m, leading to a GSD of the camera embedded to D2 of about 4 cm. The planimetric precision or the relative positioning between D1 and D2 was 13 cm while the altimetric precision of this relative positioning was 65 cm.



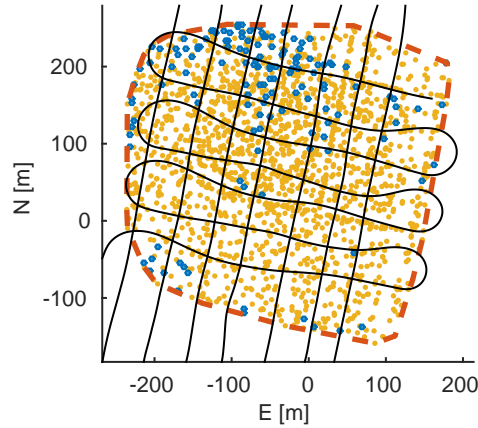


Figure 5.3. Planimetric position of tie-points. The black line are the UAV flight path. Yellow dots are seen by both N-S and E-W flight lines, blue dots only from N-S or E-W flight lines.

## 5.4 Mapping Accuracy Prediction

In this section we formulate predictions on the mapping quality achievable with the proposed method based on a simulated scenario.

We are interested in the precision of the tie-points 3D positions obtained in a conventional bundle-adjustment scenario. The parameters describing the photogrammetric network are the absolute poses of each drones (position and orientation), and the 3D position of each tie-points. These parameters are concatenated together to form the state vector  $\mathbf{x}$ . The observations are: i) position and orientation control obtained from the D2 INS/GNSS navigation system (post-processed in tightly coupled, carrier-phase differential mode), ii) image observations of the tie-points in both D1 and D2 images, iii) (optionally) and image observation of the D1 LEDs in D2 nadir images. These observations are concatenated together to form the observation vector  $\ell$ . It is possible to build a function  $\mathbf{f}$  which could simulate  $\ell$  knowing  $\mathbf{x}$ :  $\ell = \mathbf{f}(\mathbf{x})$ . The design matrix  $A$  is defined as the Jacobian matrix of  $\mathbf{f}$  with respect to the state vector  $\mathbf{x}$ , see Equation 5.1. The observation models are well known, e.g., see [131].

$$A = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \quad (5.1)$$

The covariance matrix  $\Sigma_{\mathbf{x}\mathbf{x}}$  of the parameters vector is obtained from the design matrix  $A$  and the observations covariance  $\Sigma_{\ell\ell}$

$$\Sigma_{\mathbf{x}\mathbf{x}} = (A^T \Sigma_{\ell\ell}^{-1} A)^{-1} \quad (5.2)$$

The predicted tie-point precision is obtained from the proper diagonal blocks of  $\Sigma_{\mathbf{x}\mathbf{x}}$ .

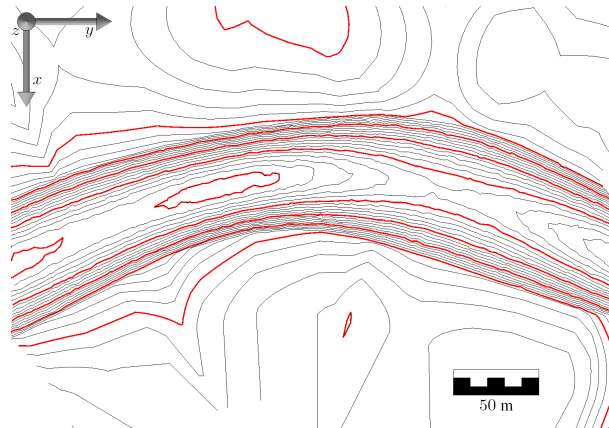


Figure 5.4. Contour lines of the canyon every five meters in height.

For this study case we consider an irregular, 350 m long canyon, up to 70 m wide and 100 m deep. See Figure 5.4 for the isolines.

Both D1 and D2 cameras have a 16 Mp sensors ( $4912 \times 3264$  pixels), and a focal length of 16 mm ( $\approx 3300$  pix). Thus, the vertical field of view is  $73^\circ$ , and the horizontal one is  $53^\circ$ . In optimal configuration, the precision of a tie-point observation is assumed to be a third of a pixel (Chapter 1.9. of [78]) and the one of a LED observation could be considered as a tenth of a pixel ([75]). However, since the conditions of mapping for urban or natural canyons are not fully controlled, we choose to input conservative values (factor 3) for the expected precisions of the image observations of tie-points and LEDs. The standard deviation of the position control for D2 is 2 cm in planimetry and 3 cm in elevation, which is compatible with GNSS carrier-phase differential processing. For the position control, we considered a standard deviation of  $0.012^\circ$  for roll and pitch, and  $0.074^\circ$  for heading, as reported for the SPAN-IGM-A1 [114].

D2 flies between 110 m and 115 m above the canyon floor<sup>2</sup>, its ground sampling distance is around 33 mm on the floor of the canyon, and the footprint of the image is around 110 m (considered in the direction of the canyon). The forward overlap is around 90 %. D1 flies between 36 m and 42 m above the canyon floor. The ground sampling distance of the nadir camera is around 11 mm on the floor of the canyon, the footprint of these images is around 38 m (considered in the direction of the canyon). The longitudinal (i.e., in the direction of the canyon) distance between two poses remain 10 m, but the drone does also lateral displacements (i.e., perpendicular of the direction of the canyon). The overlap between two successive images is up to 70 %. Two sides cameras are also embedded on D1. These cameras are equivalent to the nadir one, and are rotated by  $90^\circ$ . The distance from the canyon slopes oscillates between 10 m and 35 m, so, the GSD varies from 3 mm to 11 mm and the average

<sup>2</sup>In a real site such as Via-Mala or dense cities, we expect a satisfactory GNSS satellite visibility for a drone at 100 m above the (natural or urban) canyon floor.

## 5.4. Mapping Accuracy Prediction

		Study case				
		<i>SOTA case</i>	<i>Case 1</i>	<i>Case 2</i>	<i>Case 3</i>	<i>Case 4</i>
<i>D1</i>	$\sigma_x$	9	11	14	23	10
	$\sigma_y$	9	11	15	14	10
	$\sigma_z$	22	24	30	29	25
	nb. pts.	299	255	481	538	543
<i>D2</i>	$\sigma_x$	27	29	34	29	27
	$\sigma_y$	16	17	21	19	17
	$\sigma_z$	36	38	45	47	39
	nb. pts.	248	254	492	529	534
<i>D12</i>	$\sigma_x$	9	11	12		
	$\sigma_y$	9	12	12		
	$\sigma_z$	22	26	27		
	nb. pts.	520	539	15	0	0
<i>Side</i>	$\sigma_x$	32	32	42	34	31
	$\sigma_y$	13	14	20	15	13
	$\sigma_z$	15	18	27	26	18
	nb. pts.	151	169	292	307	320

Table 5.1. Accuracy prediction of the tie-points representing the canyon floor, and the canyon slopes (unit: mm)

overlap of the oblique images is around 40 %.

The simulation results are summarized in Table 5.1. The lines *D1*, *D2*, *D12* and *Side* give the precision and the number of, respectively, the tie-points visible by D1 nadir camera, D2, and both.  $\sigma_x$  is the precision along x direction: perpendicular to the direction of the canyon,  $\sigma_y$  is the precision along y direction: in the direction of the canyon,  $\sigma_z$  is the precision along z direction.

The classical approach for airborne UAV photogrammetry would have been to have only one UAV flying inside the canyon and equipped with INS/GNSS navigation system and one or multiple cameras. This approach can not work due to the degraded GNSS constellation. Nevertheless, we can pretend that high quality GNSS observations were available and consider such case as a reference. (column *SOTA case* of Table 5.1). This case will act as a reference case for comparing others cases.

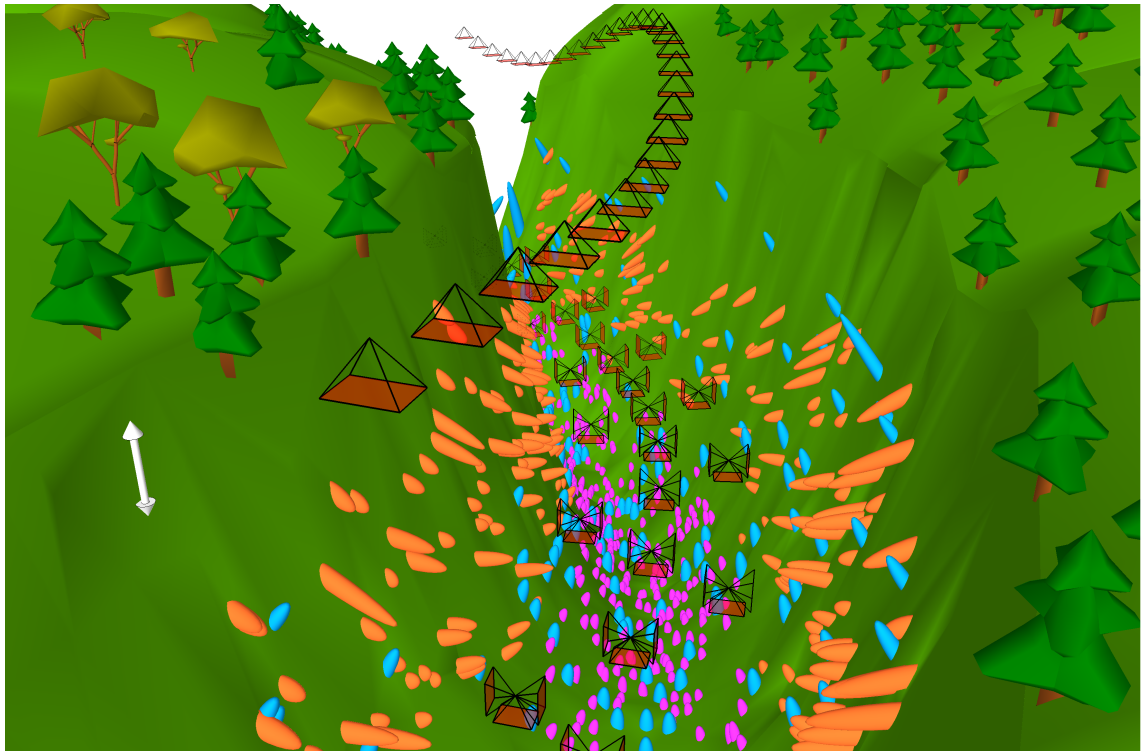


Figure 5.5. Tie-point precision as computed in *Case 3*. Poses are represented by pyramids (showing the field of view). In orange, error ellipsoids of D2 tie-points, in purple, error ellipsoids of D1 tie-points, in blue, error ellipsoids of D1 side camera tie-points. All ellipsoids are up-scaled by a factor 100. See the white double arrow for scale (10 m for the environment, 10 cm for the ellipsoids).

We consider four different adjustment scenarios. In the first case (*Case 1*) several tie-points are visible both by the upper drone, and by the lower one (line D12 of table 5.1). Most of these tie-points are visible in at least two images of D2. It is thus possible to determine their position thanks to D2, and they could act as GCPs for D1. The precision of D1 tie-points matches the one of the *SOTA case*, meaning that the position and orientation control for D1 is fully replaced by the indirect approach in this work. In highly cluttered environment, like urban or natural canyon, the number of common tie-points visible both by D1 and D2 could be lower than in *Case 1*. The lower the number of common tie-points is, the higher the standard deviation of the tie-points is. The extreme case arises when there are less than 3 commons tie-points: the system becomes unsolvable. The *Case 2*, is a middle case, between *Case 1* and this unsolvable case.

In *Case 3*, all the common tie-points are removed, see Figure 5.5. To make the system solvable again, we introduce the image observations of the LEDs. These observations permit to substitute all the common tie-points measurements between D1 and D2. The results are comparable to the ones of *case 1*, for the tie-points we are interested in: the tie-points visible by nadir and

side cameras of D1. This shows the importance of LED observations, which could substitute to hundreds of common tie-points between D1 and D2 in difficult scenarios. Such observations are always available in post processing, as D2 has to maintain D1 in the line-of-sight and uses the LEDs to provide the real-time position fix. However, the  $x$  precision of the tie-points taken by the nadir camera of D1, and the  $z$  precision of the tie-points taken by the side camera is worse than in *Case 1*. This is due to bad determination of the roll angle of D1.

A final case is also considered in which we add another type of observation, more difficult to achieve in practice, that is, D2 position in D1 images, as if LEDs were also placed on the bottom of D2. The roll and pitch angle becomes more observable as these observation have the effect of introducing position control with tens of meters of lever-arm (position control is available for D2), and thus constraining also the D1 orientation. The results are comparable to the *SOTA case* (except for the altitude whose precision is slightly worse).

## 5.5 Conclusions

This paper has presented a new technique for mapping highly cluttered environment like natural or urban canyon. The principle is to have a cooperative mapping between two drones, one flying high enough to receive GNSS signals, and localize the other one, flying in the cluttered environment.

The visual link between the two drones has shown its importance first for guidance purposes (to permit to guide the lower drone), second, for post-processing photogrammetric data. This visual link permits to reach an accuracy comparable with the one it is possible to reach in non GNSS-denied scenario.

In this work we have neglected all the important aspects related to intrinsic camera calibration and boresights and lever-arms determination. We considered the cameras, the lever arm and the boresight matrix to be perfectly calibrated. However, we argue that the intrinsic camera calibration is also observable in the combined adjustment of D1 and D2 images, and that lever-arm and boresights can be calibrated in dedicated flights as it is common in single drone UAV-based photogrammetry. The only non-trivial lever-arms are the ones which relates D1 camera to the LEDs. However, this can be determined with millimeter level accuracy with careful UAV fabrication.

We argue that the technological challenges behind the actual implementation of this methods have been addressed in previous, related, experiments. The next step is the validation of the concept in real-world applications.



## 6 Compensating over- and underexposure in optical target pose determination

The chapter 5 presents an aerial-aerial mapping tandem involving two drones. This chapter will present a subsystem for the terrestrial-aerial collaborative mapping project entitled *mapKITE*, and described in [108]. Such a method permits to map the same object from two viewpoints: side and above, and to embed heavy equipment on the terrestrial vehicle. This chapter will focus on the visual link between the aerial vehicle and the terrestrial one and is originated from the following preprint.

E. Cledat, M. Ruffener and D. A. Cucci. Compensation over-under exposure in optical target pose Determination *submitted to Journal of Pattern recognition*, 2019

The contribution of D. A. Cucci was mainly to determine the target boundaries  $\hat{P}_i'$  while the one of E. Cledat was to use them to determine the position and the orientation of the camera (which is intended to be embedded on the aerial vehicle) that see the target. The experimental validation was performed by all three authors.

### Abstract

Optical coded targets allow to determine the relative pose of a camera, on a metric scale, from one image only. Furthermore, they are easily and efficiently detected, opening to a wide range of applications in robotics and computer vision. In this work we describe the effect of pixel saturation and non-ideal lens Point Spread Function, causing the apparent position of the corners and the edges of the target to change as a function of the camera exposure time. This effect, which we call exposure bias, is frequent in over- or underexposed images and introduces a systematic error in the estimated camera pose.

We propose an algorithm that is able to estimate and correct for the exposure bias exploiting specific geometric features of a common target design based on concentric circles. Through

rigorous laboratory experiments carried out in a highly controlled environment, we demonstrate that the proposed algorithm is seven times more precise and three times more accurate in the target distance estimation than the algorithms available in the literature.

### 6.1 Introduction

Optical coded targets generally consist of a set of high-contrast geometric features (such as lines, squares or circles), in which a code is embedded to exclude false matches and distinguish multiple targets in a scene. Knowing the physical dimensions of the target and the camera calibration allows us to determine from a single image the metric pose of a target relative to the camera. Several target designs and detection algorithms have been proposed in the literature, such as ARToolKit [165], AprilTags [119], and ArUco [57], which have been successfully employed in multiple applications, such as camera calibration, photogrammetry, augmented reality, machine vision and robotics in general.

Target detection algorithms establish correspondences between the geometric features in object space and their projections on the image plane. From these correspondences, the pose of the camera relative to the target can be inferred. For instance, the main target feature in ArUco is a black square on a white background; its four corners are localized by a sub-pixel corner detector and the camera pose is determined by solving the Perspective- $n$ -Points (PnP) problem [167].

The accuracy of the camera pose depends on how precisely the critical target features can be measured on the image. A major source of bias is the exposure time. Indeed, the apparent position of edges and corners may be different in long-exposed images because of the combined effect of a non-ideal lens Point Spread Function (PSF) and pixel saturation. In Figure 6.1, we compare two images of an ArUco target taken with two different exposure times: the apparent size of the target is different in the two images and the target on the right seems to be farther away from the camera. We refer to this effect as *exposure bias*, i.e., the systematic error in the estimated camera pose that is a function of the exposure time.

Since saturation in over- or underexposed images leads to a loss of information, it seems difficult to recover the true position of an image feature, such as a corner or an edge, just by using the intensity of pixels in its vicinity. Therefore, image exposure has to be optimized to avoid saturation. However, this is difficult in many cases because of unpredictable light conditions, variable scene illumination as functions of time and space, and reflections on the target surface. Moreover, auto-exposure algorithms are typically implemented at sensor level to optimize either the overall scene exposure or the exposure of a predefined area, which does not guarantee an optimal exposure of the target.



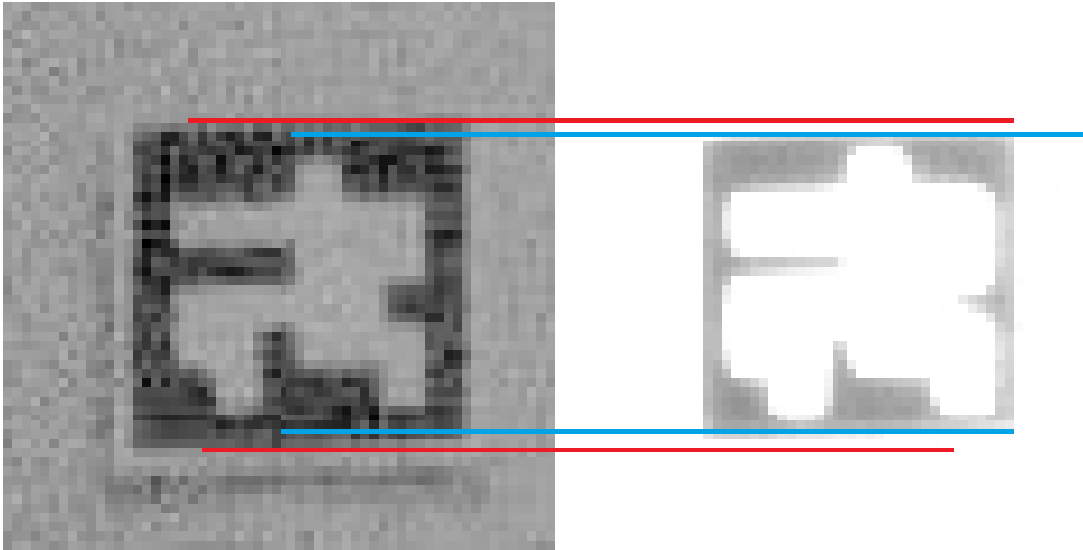


Figure 6.1. Two images of an ArUco target taken from the same camera pose with different exposure times, the underexposed image is on the left and the overexposed one on the right. The brightness of the underexposed image has been enhanced. The difference between the apparent location of the top and bottom edge is highlighted by red and blue lines.

In this work, we introduce a new optical target design and its corresponding pose determination algorithm that are not affected by either the exposure time or the illumination. The proposed target design is shown in Figure 6.2 and it consists of a black ring over a white background; inside the ring, there are small white dots in which a code is embedded. The same pattern is repeated in a fractal fashion to allow for multi-scale detection (e.g., from very far or very near). In this design, the over- or underexposure affects the apparent thickness of the black ring. However, this effect is *orthogonal* to the one that we obtain by moving the camera farther or closer to the target, leading to a uniform scaling of the whole target and preserving the proportions of every feature (see the purple and red arrows in Figure 6.2). Hence, we can design an algorithm that exploits this effect to estimate and correct the exposure bias in contour positions.

Many studies have investigated how to determine the camera pose from concentric circular features. For instance, in [74], a method for camera calibration based on concentric circles was presented, and another formulation was proposed in [1]. The geometric and algebraic constraints related to the projection of concentric circles were discussed in [80]. In [42], the authors addressed the issue of how to accurately locate the center of two concentric circles based on their projections (non-concentric ellipses). However, all these works assume that the apparent location of the circles on the image plane is not biased by the exposure. Here, we propose a new algorithm for camera pose determination, derived from the well-known Bundle Adjustment algorithm [158], that exploits the specific geometry of the target to estimate and

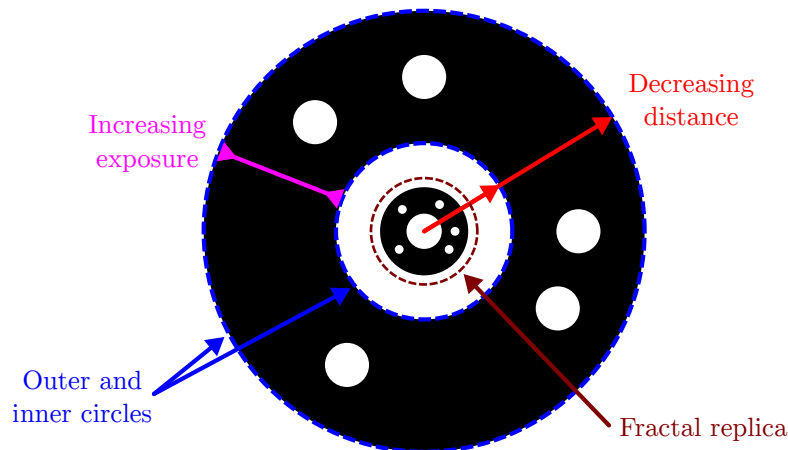


Figure 6.2. The proposed target design. Overexposing the target results in a reduction of the apparent thickness of the black ring (purple arrow), while increasing the camera distance leads to a uniform scaling of the whole target.

compensate for any exposure bias in the localization of edges.

This work is organized as follows. In Section 6.2, we introduce the image processing pipeline for our target design. Each step of the pipeline will be detailed in later sections: in Section 6.3 and Section 6.4 we develop a method to determine with sub-pixel precision the apparent image location of the two concentric circles of the target, taking into account a non-ideal lens PSF. In Section 6.5, we determine the camera pose by using the position of a set of edge points and assuming that those are the projections of two concentric circles of known diameter. In Section 6.6, we modify such method to compensate for the exposure bias, which we link to the deviations in the apparent thickness of the black ring. In Section 6.7, we validate our exposure compensation algorithm through acquired ground truth data, proving its illumination invariance and the *orthogonality* between the distance effect and the exposure bias effect.

## 6.2 Overview of the image processing pipeline

In this section, we summarize the steps required to determine the position and the orientation of the proposed target relative to the camera.

1. The target is located in the image and its main features, the inner and the outer circle, are coarsely identified. First, a pixel-level edge detection algorithm (e.g., Canny [27]) is applied, then the elliptical contours are clustered in concentric couples. For each couple, the distinctive target code is searched (see again Figure 6.2), and the two contours for

which the code test succeeds are selected as the outer and inner target circle. This step is described in detail in a previous work of the authors [42].

2. Two ellipses are fitted on the selected contours as in [49]. Then, by using the determined parameters of the ellipses, a set of normal directions  $\vec{n}_i$  is analytically computed for a set of equally spaced points  $P'_{0,i}$  on the ellipses.
3. The apparent position of the edge along each  $\vec{n}_i$  is refined by employing the edge formation model and the sub-pixel edge detector introduced in Section 6.3 and 6.4. The result is a set of refined edge points  $\hat{P}'_i$ .
4. The position and the orientation of the camera relative to the target are computed from  $\hat{P}'_i$ , without or with exposure compensation, as discussed in Section 6.5 and 6.6.

### 6.3 Edge Formation Model

In this section, we derive a closed-form model for the light intensity in the vicinity of black-/white transitions (i.e., edges) of the target, as measured from the imaging sensor. This model will be used in the next section to construct a rigorous sub-pixel edge detector.

Consider an arbitrary 3D point  $P$  on a black/white edge of the target and refer to Figure 6.3. Each point on the target maps to a point on the image plane via a function  $\Pi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ , so that  $P' = \Pi(P)$ .  $\Pi(\cdot)$  accounts for the camera pose relative to the target, the lens distortion, and the projection onto the image plane. Let now  $\vec{n}$  be a unitary vector orthogonal to the edge at  $P'$  and directed towards the white side of the edge. We define a reference frame with axes  $\vec{n}$  and  $\vec{m}$ , such that  $\vec{n} \perp \vec{m}$ , and with origin close to  $P'$ .

If no significant blur is introduced by the lens system (i.e., the PSF is ideal), the black/white transition occurs sharply on the image plane. More precisely, in the vicinity of  $P'$ , the light intensity  $i$  at coordinates  $(n, m)$  on the image plane is independent of  $m$  and it is given by

$$i(m, n) = i(n) = a\mathbf{1}(n - n_{P'}) + b, \quad (6.1)$$

where:  $a$  and  $b$  are two constants such that  $b$  is the light intensity of black areas and  $(b + a)$  is the one of white areas,  $n_{P'}$  is the  $n$  coordinate of  $P'$ , and  $\mathbf{1}(\cdot)$  is the unit step function.

However, black/white transitions are never sharp, due to the pixel sampling<sup>1</sup> and the lens property. Thus, the light intensity at each position of the image plane can be obtained by

<sup>1</sup>The pixel value of the pixels that receive the image of the black/white boundary is the average of the black and white values (weighted by their surface) under the hypothesis that the PSF sigma is negligible compared to the pixel size. See paragraph 1.9.3.1.1. of [78].

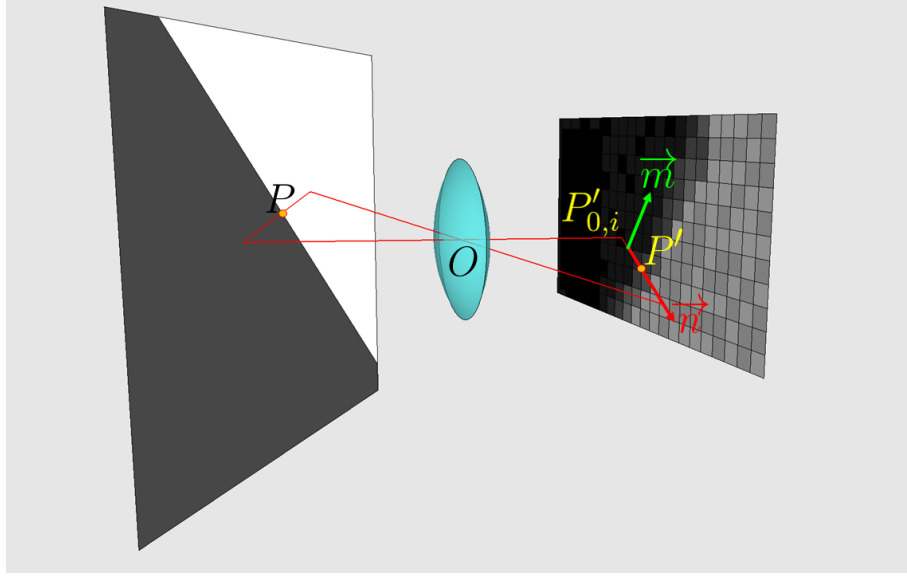


Figure 6.3. A 3D edge point  $P$  maps to its unknown projection  $P'$  on the image plane via  $\Pi(\cdot)$ .  $\vec{n}$  is defined as orthogonal to the edge at  $P'$ , while  $\vec{m}$  is parallel to it.

the convolution of the intensity function  $i(m, n)$  with the lens PSF. We have taken into consideration only zero-mean Gaussian PSFs as they are widely used to model lens systems of common cameras. These PSFs are characterized by a two-dimensional covariance matrix  $\Sigma$  that encodes the spread of a point light source on the image plane due to a non-ideal lens system. The light intensity mediated by the lens system is given by:

$$\begin{aligned}
 id(m, n) &= i(n, m) * psf(m, n) = i(n) * psf(m, n) = \\
 &= \iint_{(\tau_m, \tau_n) \in \mathbb{R}^2} i(n - \tau_n) psf(\tau_m, \tau_n) d\tau_m d\tau_n = \\
 &= \int_{\tau_n \in \mathbb{R}} i(n - \tau_n) \phi(\tau_n; \sigma) d\tau_n, \tag{6.2}
 \end{aligned}$$

where  $\phi(\cdot, \sigma)$  is the one-dimensional Gaussian probability density function with variance  $\sigma^2 = \Sigma_{nn}$ . Here, we have regarded the light intensity  $i(m, n)$  as independent of  $m$  in a local neighbourhood of  $P'$  because  $\vec{m}$  is parallel to the edge, and we have considered the integral in  $d\tau_m$  as corresponding to the marginalization of  $m$ .

By further expanding Equation 6.2, we obtain:

$$\begin{aligned} id(n) &= i(n) * \text{pfs}(n) = \int_{-\infty}^{\infty} (a\mathbf{1}(n - n_{P'} - \tau) + b) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\tau^2}{2\sigma^2}} d\tau \\ &= a \int_{-\infty}^{n-n_{P'}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\tau^2}{2\sigma^2}} d\tau + b = \frac{a}{2} \left[ 1 + \text{erf}\left(\frac{n - n_{P'}}{\sqrt{2}\sigma}\right) \right] + b. \end{aligned} \quad (6.3)$$

Equation 6.3 gives a closed-form expression of the expected light intensity at continuous coordinates on the image plane. It involves four parameters: i)  $n_{P'}$ , the actual edge position; ii and iii)  $a$  and  $b$ , which encode the light intensity associated with black and white areas; iv)  $\sigma$ , the component of the lens PSF in the  $\vec{n}$  direction.

In the next section, we show how we can obtain a sub-pixel estimate of the image coordinates of  $P'$  by fitting the model in Equation 6.3 to the pixel intensities in the image.

## 6.4 Subpixel Edge Position

In the following section, we consider the problem of determining the sub-pixel location of  $P'$  given that an initial guess,  $P'_0$ , and the vector orthogonal to the edge and passing through  $P'_0$ ,  $\vec{n}$ , are known. These are obtained in previous image processing steps. For example, in [42], the authors determine the pixel-level edge points of circular target features by using the Canny edge detector. Next, an ellipse is fitted on those as in [49]; from the ellipse,  $\vec{n}$  can be obtained in closed form for each edge point. We also show how the introduced sub-pixel edge detector is biased if any of the pixels in the vicinity of the edge point is saturated, e.g., because of over- or underexposure.

The sub-pixel edge position  $P'$  on  $\vec{n}$  is obtained by fitting the edge model in Equation 6.3 to the measured pixel intensities along  $\vec{n}$  via a non-linear least-squares optimization. First, we sample the image along  $\vec{n}$  with one pixel spacing, i.e.,  $n_j \in [-N, -N+1, \dots, N]$ , where  $N$  is the maximum distance from the query point for which we assume Equation 6.3 to hold (e.g.,  $N = 3$ ). Note that the initial guess for the position of  $P'$ ,  $P'_0$  lies at  $n = 0$ . For each  $n_j$ , the image intensity  $\tilde{i}(n_j)$  is obtained interpolating the image pixels, e.g., by means of bilinear interpolation. Given  $\tilde{i}(n_j)$ , an estimate for the location of the edge point along  $\vec{n}$ ,  $\hat{n}_{P'}$ , is calculated by solving the following least-squares optimization problem:

$$[\hat{n}_{P'}, \hat{a}, \hat{b}, \hat{\sigma}] = \underset{n_{P'}, a, b, \sigma}{\text{argmin}} \sum_{j=-N}^N (i(n_j) - \tilde{i}(n_j))^2. \quad (6.4)$$

This is a non-linear curve-fitting problem that can be solved with the Levenberg-Marquardt algorithm if the initial guess of every unknown is sufficiently close. An effective initialization

## Chapter 6. Compensating over- and underexposure in optical target pose determination

is given by  $n_{P'} = 0$ ,  $a = \tilde{i}(n_N) - \tilde{i}(n_{-N})$ ,  $b = \tilde{i}(n_{-N})$ , and  $\sigma = 1$ . The curve fitting problem also yields estimates for  $a$  and  $b$  (the intensity of black and white areas in the image plane). Note that  $\sigma$  is the component of the PSF in the direction of  $\vec{n}$  and that PSF is stable across multiple frames and consistent for different areas of the image.

The estimation problem in Equation 6.4 is solved for each edge point  $P'_{0,i}$ , obtaining a set of refined estimates of the edge location along each  $\vec{n}_i$ . These 1D coordinates are transformed back in the image reference frame and build the set of image observations  $\hat{P}'_i$  that we will employ in later sections to estimate the camera pose relative to the target.

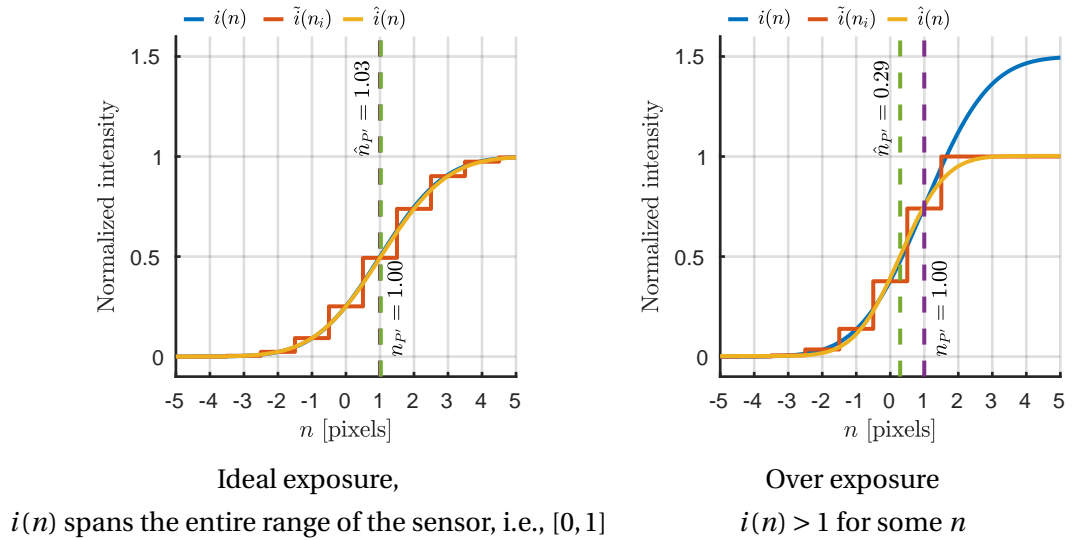


Figure 6.4. Pixel intensity of a black/white transition along  $\vec{n}$ . The intensity  $i$  is proportional to the amount of light exposing the sensor.  $\tilde{i}$  is the measured pixel intensity (subject to overexposure) and  $\hat{i}$  is the estimated from  $\tilde{i}$ . The estimated edge position  $\hat{n}_{P'}$  differs from the true value  $n_{P'}$  by a few hundredths of pixels if the image exposure is ideal. However, in overexposed images the difference can reach several pixels.

We now show that the estimate  $\hat{n}_{P'}$  is biased if some image intensities  $\tilde{i}(n_j)$  are saturated. This is common in over- or underexposed images. Figure 6.4 depicts the pixel intensity in an image with an ideal exposure (Figure 6.4a) and in an overexposed image (Figure 6.4b). In the ideal exposure case, no saturation occurs and the continuous intensity function  $i(n)$  (in blue) can be accurately recovered from the samples  $\tilde{i}(n_j)$  (in red). In this case,  $\hat{n}_{P'} = n_{P'}$ . On the other hand, in the overexposed image,  $i(n)$  is saturated for  $j > 1$ . Thus, the estimated intensity function  $\hat{i}(n)$  (in yellow) differs from  $i(n)$  in the whole saturated area, resulting in  $\hat{n}_{P'} \neq n_{P'}$ . Note that the estimated intensity function is close to the true intensity in the non-saturated area (i.e., for  $n < 2$ ).

Figure 6.5 shows a numerical study of the bias in sub-pixel edge position as function of  $\sigma$  and

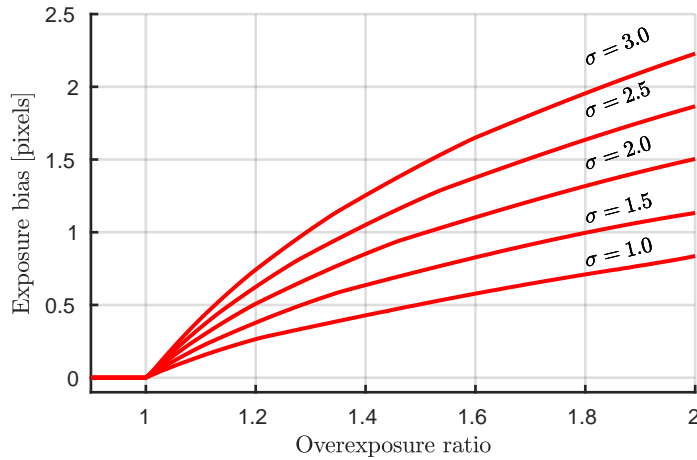


Figure 6.5. Exposure bias as a function of the overexposure ratio for different values of  $\sigma$ .

overexposure. The overexposure is parameterized as the ratio between the maximum of the intensity function  $i(n)$  and the saturation level. The exposure bias is severe in blurry images (high values of  $\sigma$ ) but it is significant also in reasonably sharp images ( $\sigma \approx 1$ ). Similar results hold for the underexposure case.

To give a measure of the error in the camera pose determination due to the exposure bias, we introduce an error  $\Delta d$  when we determine the target distance  $d$  by comparing its known size  $2r$  with the size measured on the image plane  $2r'$  (which is subject to the exposure bias  $\gamma$ ). When a target is parallel to the image plane, this error is given by  $\Delta d = -\frac{d\gamma}{r'+\gamma} \approx -d\frac{\gamma}{r'}$ .  $\Delta d$  is proportional to the distance from the target, and it is approximately proportional to the relative exposure bias (i.e., exposure bias normalized by the target radius). For example, a 5 cm target imaged from 3 m, with focal length  $f = 2900$  px, has a diameter of 48 px in the image. In this case, an exposure bias of one pixel leads to an error of 13 cm ( $\approx 5\%$ ) in the distance estimation.

In the next sections, we will explain how specific features of our target design (Figure 6.2) allow estimating and correcting for the exposure bias.

## 6.5 Camera Pose Determination

We present a new rigorous approach to determine the camera pose starting from a set of edge points being the projection of the two concentric circles composing the target (see Figure 6.2). In Section 6.6, we will extend this approach to estimate and compensate for the exposure bias.

After applying the sub-pixel detection algorithm introduced in Section 6.4, a set of edge points  $\hat{P}'_i$  is available. These are the projections of object points lying on two concentric circles of

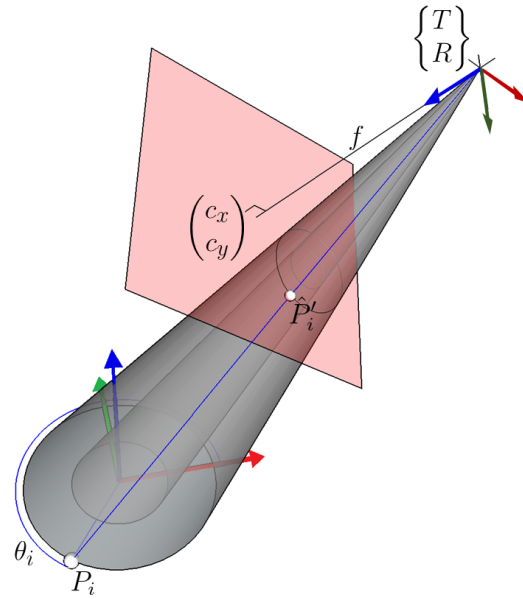


Figure 6.6. Geometric elements involved in the pose determination problem. The image plane is depicted in red. The pose of the camera reference frame in relation to the target frame is given by  $T$  and  $R$ . An object point  $P_i$  lying on one of the two concentric circles of the target with angle  $\theta_i$  projects to  $\hat{P}'_i$  on the image plane.

known radius. The camera pose is determined by minimizing the reprojection error associated with each of these points with an approach similar to Bundle Adjustment.

The edge points are given in homogeneous coordinates  $\hat{P}'_i = [x_{\hat{P}'_i}, y_{\hat{P}'_i}, 1]^T$  and they will be used as input for the algorithm presented in this section. First,  $\hat{P}'_i$  is transformed into a unit-less vector  $\tilde{P}'_i$  by employing the camera calibration matrix:

$$\tilde{P}'_i = \begin{pmatrix} f & 0 & pp_x \\ 0 & f & pp_y \\ 0 & 0 & 1 \end{pmatrix}^{-1} \hat{P}'_i, \quad (6.5)$$

where  $f$  is the principal distance of the camera, and  $pp_x$  and  $pp_y$  are the  $x$  and  $y$  coordinate of the principal point, respectively. Equation 6.5 can also be modified to account for lens distortion, as described in [24]. Note that the intrinsic camera calibration must be known *a priori* because the focal length  $f$  is correlated with the depth of the target, and they cannot be simultaneously determined with the approach presented in this work.

Let the target reference frame be placed at the center of the two concentric circles of the target with the  $Z$  axis normal to the target plane and pointing away from it. Let  $T$  be the position



of the optical center of the camera in this frame, and  $R$  the rotation from the camera to this frame (Figure 6.6). Since  $\tilde{P}'_i$  can be the projection of a point belonging either to the outer or to the inner circle of the target, we define the function  $r(i) : \mathbb{N} \rightarrow \mathbb{R}_+$  that provides the radius of the circle to which  $\tilde{P}'_i$  belongs to. It is worth noting that this function allows us to generalize the approach presented below to an arbitrary number of concentric circles.

We know from perspective geometry that, for any object point  $P$  and its projection onto the image plane  $\tilde{P}'$ , given in unit-less homogeneous coordinates, it holds that

$$\exists \lambda \in \mathbb{R} \mid T + \lambda R \tilde{P}' = P. \quad (6.6)$$

$\lambda$  is generally obtained from the third component of Equation 6.6 and eliminated.

For any  $\tilde{P}'_i$ , we know that the corresponding object point lies on a circle of radius  $r(i)$  on the plane  $Z = 0$ . Therefore we have that:

$$\exists \lambda_i \in \mathbb{R}, \theta_i \in [0, 2\pi) \mid T + \lambda R \tilde{P}'_i = \begin{bmatrix} r(i) \cos \theta_i \\ r(i) \sin \theta_i \\ 0 \end{bmatrix}. \quad (6.7)$$

$\theta_i$  encodes, for each edge points, its corresponding position on the target circle in object space: it is unknown at this stage since it is not possible, by using only local features of the target, to establish a univocal 2D-to-3D correspondence between object points on the target circles and their projections on the image plane. We eliminate  $\lambda$  and  $\theta_i$  from Equation 6.7, as discussed in detail in Appendix, obtaining a 1D condition for each point  $\tilde{P}'_i$ .

We can formulate Equation 6.7 for each edge point  $\tilde{P}'_i$ . This gives us a system of equations that can be solved for the desired  $R$  and  $T$  (e.g., in least-squares sense), provided that enough edge points are available on both circles. However, since we have not established an unambiguous 2D-to-3D correspondences, such system is underconstrained. Indeed, Equation 6.7 is still satisfied if we multiply the left side of the equality by an arbitrary rotation matrix  $\Omega_Z$  around the  $Z$  axis. This issue can be solved by constraining the optical center of the camera to the  $Y = 0$  plane. This is achieved as follows: let

$$\begin{cases} \tilde{T} &= \Omega_Z T \\ \tilde{R} &= \Omega_Z R \end{cases}, \quad (6.8)$$

such that  $\tilde{T} = [\tilde{X}, 0, \tilde{Z}]$ . This requires that

$$\Omega_Z = \begin{bmatrix} \frac{T_X}{\sqrt{T_X^2 + T_Y^2}} & \frac{T_Y}{\sqrt{T_X^2 + T_Y^2}} & 0 \\ -\frac{T_Y}{\sqrt{T_X^2 + T_Y^2}} & \frac{T_X}{\sqrt{T_X^2 + T_Y^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.9)$$

We can now replace  $T$  and  $R$  with  $\tilde{T}$  and  $\tilde{R}$  in Equation 6.7 and rearrange it in the form

$$f_i(\tilde{T}, \tilde{R}, \tilde{P}'_i) = 0. \quad (6.10)$$

This gives us one condition  $f_i$  per observation  $\tilde{P}'_i$ , providing a system of equations that can be solved for  $\tilde{T}$  and  $\tilde{R}$ , e.g., in the least-squares sense, with the Gauss-Newton or Levenberg-Marquardt algorithm. As  $\tilde{R}$  belongs to the special orthonormal group  $SO(3)$ , we rely on the manifold encapsulation technique described in [149] to deal with non-Euclidean unknowns in least-squares estimation.

As already mentioned, the target can rotate around the  $Z$  axis without changing the geometry of the problem (refer to Figure 6.6). This is because we cannot establish unique 3D-to-2D correspondences between object points on the target circles and their image projections. Indeed,  $\theta_i$  is not known in Equation 6.7. To solve this issue, we constrain the camera to the  $Y = 0$  plane. However, this is not a limiting problem since the position of the target relative to the camera,  $-\tilde{R}^T \tilde{T}$ , can still be determined.

In this section, we have described a method to determine the pose of the camera relative to the target by using a set of edge points being the projection of two concentric circles with known radius. In the next section, we will extend this method to take into account the exposure bias, i.e., the condition in which the location of the edge points on the image plane is affected by over- or underexposure.

## 6.6 Correcting for the exposure bias

In the previous section, we have seen how to determine the position and the orientation of the camera relative to the proposed target from a set of sub-pixel edge locations obtained as described in Section 6.4. However, we also saw that these measurements can be biased if the exposure time is not ideal (see Figure 6.4). Indeed, in case of over- or underexposure, the apparent position of  $\hat{P}'_i$  can be displaced along the line normal to the edge  $\vec{n}_i$ : When the image is overexposed, such displacement is towards black areas and vice versa for underexposed images. This effect is shown in Figure 6.7, which compares the pixel intensities of two pictures

## 6.6. Correcting for the exposure bias

taken with different exposure times from the same camera position and orientation.

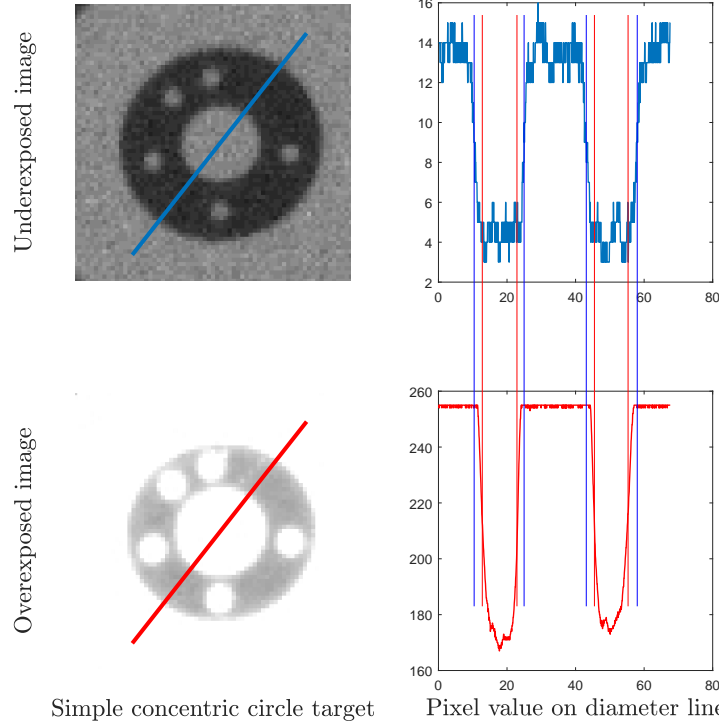


Figure 6.7. Pixel intensity along the diameter of the target for an underexposed (top) and an overexposed (bottom) image. The brightness of the top image has been enhanced. In the underexposed image, the black/white transition is shifted by a few pixels towards the black area, which is highlighted by red and blue lines in the plots on the right.

Recall that, as defined in Section 6.3,  $\vec{n}_i$  is a unitary vector on the image plane, perpendicular to the edge, and directed towards the white side of the edge. Let  $\gamma_i \in \mathbb{R}$  be the shift of  $\tilde{P}'_i$  along the vector  $\vec{n}_i$ . If the target has a uniform illumination, we can assume  $\gamma_i = \gamma, \forall i$ . The new parameter  $\gamma$  can be estimated along the geometric parameters  $\tilde{T}$  and  $\tilde{R}$ . In order to do this, we substitute  $\tilde{P}'_i$  in Equation 6.7 with an observation corrected for the exposure:  $\tilde{P}'_i + \gamma \vec{n}_i$ . Hence:

$$\exists \lambda_i \in \mathbb{R}, \theta_i \in [0, 2\pi) \mid T + \lambda R(\tilde{P}'_i + \gamma \vec{n}_i) = \begin{bmatrix} r(i) \cos \theta_i \\ r(i) \sin \theta_i \\ 0 \end{bmatrix}. \quad (6.11)$$

The three conditions of this system can be simplified to one condition in the form  $f_i(\tilde{T}, \tilde{R}, \gamma, \tilde{P}'_i) = 0$  for each edge measurement  $\tilde{P}'_i$  by eliminating  $\lambda$  and  $\theta_i$ , as discussed in the Appendix.

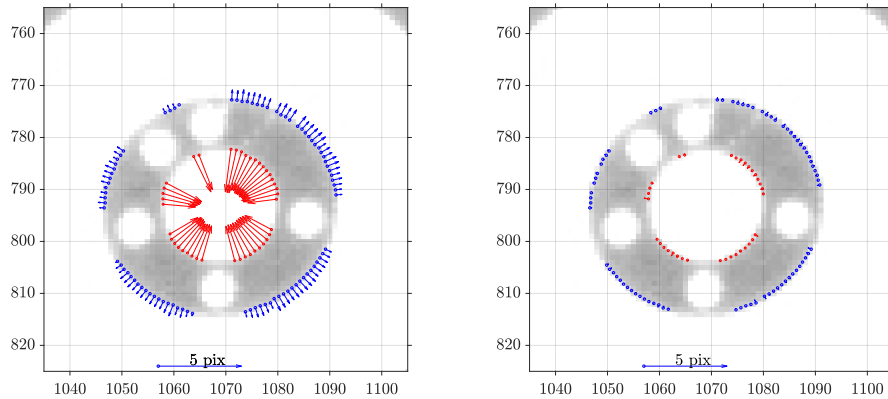


Figure 6.8. Reprojection error for the considered edge points  $\hat{P}'_i$  in an image without (on the left) and with (on the right) exposure compensation.

Figure 6.8 shows the residuals associated with each  $f_i$  in an overexposed image with (on the right) and without (on the left) correction for exposure bias. In the case of overexposure, the thickness of the black ring is smaller than expected, which translates in systematic residuals that cannot be compensated by any assignment of  $\tilde{R}$  and  $\tilde{T}$  (the residuals on the inner circle points are bigger as those points are outnumbered by the ones on the outer circle). On the other hand, such systematic effect in the residuals is completely captured once the estimation of the  $\gamma$  parameter is enabled. The impact of this on the accuracy of the camera pose determination with respect to the target will be investigated in the next section.

## 6.7 Experimental Evaluation

The goal of our method is to determine the position of a camera relative to a target compensating for any possible bias caused by over- or underexposure. We validated it in a highly controlled environment: refer to Figure 6.9. Both the camera  $A$  and the target  $D$  were fixed on a rigid tripod, and a set of photos with different exposure times was taken to simulate normal or difficult light conditions. The same set-up was repeated for 20 different positions of the target, and, for each of them, the relative position of the camera was determined i) by means of total-stations to compute a  $mm$  level precision ground truth positioning, and ii) with several Computer-Vision algorithms, including the ones presented in this paper. The results are then compared focusing on the estimated distance from the camera to the target, since the effects of over- and underexposure mainly translates in a bias in the apparent target distance. The detailed ground truth-ing procedure be found in the Appendix.

The concentric circle target  $D$  was printed on a  $50 \times 50$  cm white plastic rigid board (the target design is shown in Figure 6.2), and the total-stations measured four crosses printed on the

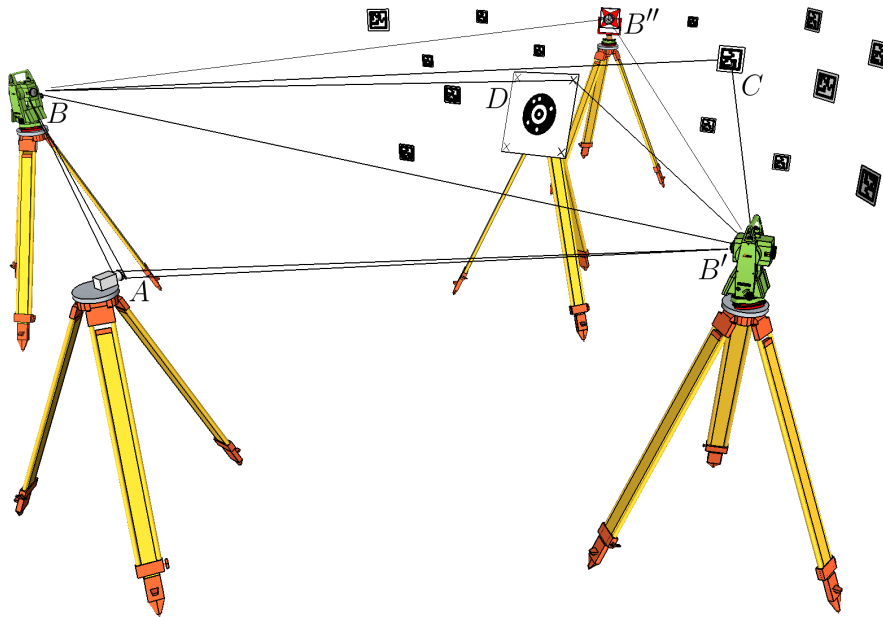


Figure 6.9. Schematic representation of the set-up to obtain the reference positions of the target and the camera. The total-stations network in  $B$ ,  $B'$  and  $B''$  measures the position of the camera ( $A$ ), the position of several ArUco targets ( $C$ ), and the position and orientation of the proposed target ( $D$ ). The camera  $A$  is oriented from the total-stations and from image observation of the ArUco targets,  $C$ .

four corners of the board (Figure 6.9).

This experimental set-up allows computing the position of the camera  $A$  relative to the target  $D$ . By using the images taken by the camera  $A$ , we validated the method presented in Sections 6.5 and 6.6 and determined the camera position. Since the pose determination depends on the exposure time, a set of 117 photos was taken with different exposure times ranging from 10 ms to 300 ms (such high exposure times depends on the rather dark indoor setup used for the experiment). We applied four computer-vision algorithms to determine the position of  $A$  relative to  $D$ , considering both the outer target (diameter: 25cm) and the inner target (diameter: 5cm). We first applied the PnP algorithm [109],[89], [] by using the centroids of the five white dots composing the target code (Figure 6.2); second, we used an algorithm based on separated ellipse fitting [42]; third, we run the joined ellipse adjustment described in Section 6.5; lastly, we corrected the joined ellipse adjustment for the exposure bias as described in section 6.6. Figure 6.10 shows the computed target distance with the four different algorithms for a given camera position and for for each exposure time.

First, when comparing the obtained results with the reference distance, we notice that the errors associated with the inner target are about five times bigger than the errors for the outer target. This is expected, as the inner target is five times smaller than the outer one. Second,

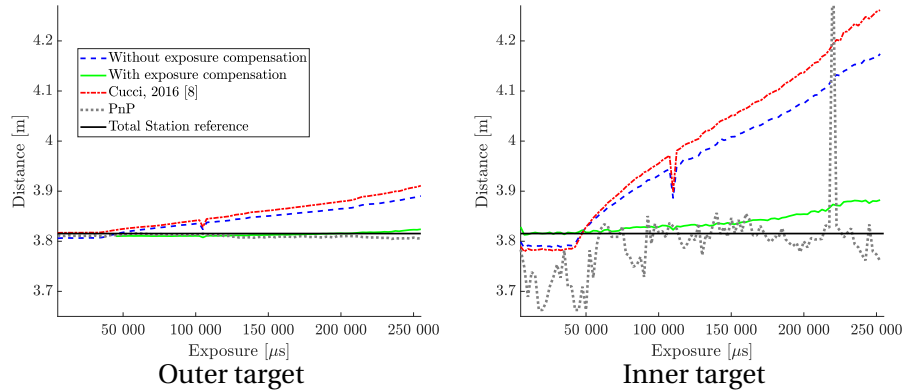


Figure 6.10. Target distance as a function of exposure time obtained with different algorithms with respect to the reference. It is possible to see that the distance obtained with exposure compensation, as in Section 6.6, is practically independent with respect to exposure time.

we notice that when the image is not overexposed (e.g., exposure time less than 50 ms, no algorithm exhibits a bias which can be correlated with the exposure time. However, if the image was overexposed (e.g., exposure time greater than 50 ms), all the algorithms without exposure compensation, with the exception of PnP, i.e., ([42] and Section 6.5) show a deviation from the reference distance that increased with the exposure time. On the other hand, when the exposure bias  $\gamma$  was computed together with the other parameters (Section 6.6), the results became independent from the exposure. Regarding the PnP algorithm, in this particular case it is not affected by exposure time as the position of the white dots, used as an input for this algorithm, does not change as a function of the exposure. This would not be the case, for example, in the case corner points are used, as in ArUcO. As a side note, as only five points are available, the distance obtained from PnP is more noisy if compared with the other algorithm employing several points on the two circles.

The same experiment was repeated for different positions of the target  $D$ . The position of  $A$ ,  $B$ ,  $B'$ ,  $B''$  and  $C$  remained unchanged while we modified the position of  $D$  20. Two statistical indicators were calculated to assess the quality of the results: the Standard deviation and the RMS of true error (Figure 6.11). The Standard deviation is determined by comparing the results of each exposure time to their mean  $\bar{d}$ ; it describes the dispersion of the computed values but it does not assess the influence of a potential bias. To address this issue, we also computed the RMS of true error based on the comparison between the obtained results and their ground truth value  $\check{d}$ .

The summary statistics are shown in Table 6.1. The errors in Figure 6.11 are normalized by

## 6.7. Experimental Evaluation

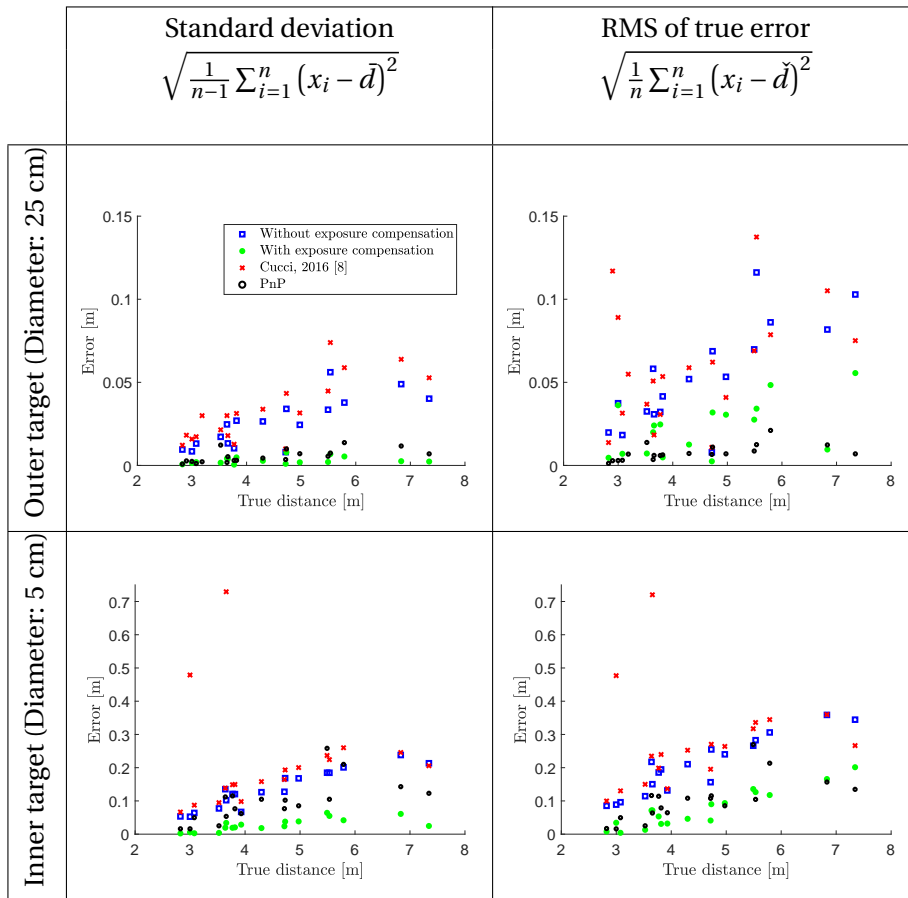


Figure 6.11. Distance error as a function of the true target distance for different pose determination algorithms.

		Geometric mean of:	
		relative standard deviation $\frac{\sqrt{\frac{1}{n-1} \sum_{i=1}^n (d_i - \bar{d})^2}}{\bar{d}}$	relative RMS of true error $\frac{\sqrt{\frac{1}{n} \sum_{i=1}^n (d_i - \check{d})^2}}{\check{d}}$
Outer target	Without exposure compensation	0.49 %	1.01 %
	With exposure compensation	0.05 %	0.37 %
	Cucci, 2016 [8]	0.66 %	1.17 %
	PnP	0.10 %	0.15 %
Inner target	Without exposure compensation	2.79 %	4.30 %
	With exposure compensation	0.44 %	1.16 %
	Cucci, 2016 [8]	4.16 %	5.73 %
	PnP	1.76 %	1.86 %

Table 6.1. Geometric mean of the relative error for every position.

the distance to the target and aggregated by geometric mean. These results show that the exposure compensation is more than seven times more precise, and more than three times more accurate than ellipse fitting without exposure compensation. However, since the position of the white dots is not affected by the exposure, the distance calculated by the proposed method is only about two times more precise than the one assessed by the PnP, and it is approximately as accurate.

We validated the method described in Section 6.6 also by assessing the independence between the geometric parameters and the exposure bias parameter  $\gamma$  by calculating the correlation factor between the computed distance  $d$  and  $\gamma$ . The covariance matrix of  $\tilde{X}$ ,  $\tilde{Z}$ , and  $\gamma$  is computed after solving the least-squares estimation problem formulated in Section 6.6. The covariance matrix of  $d$  and  $\gamma$  is determined by variance propagation as follows:

$$\begin{bmatrix} \sigma_d^2 & \sigma_{d\gamma} \\ \sigma_{d\gamma} & \sigma_\gamma^2 \end{bmatrix} = F \begin{bmatrix} \sigma_{\tilde{X}}^2 & \sigma_{\tilde{X}\tilde{Z}} & \sigma_{\tilde{X}\gamma} \\ \sigma_{\tilde{X}\tilde{Z}} & \sigma_{\tilde{Z}}^2 & \sigma_{\tilde{Z}\gamma} \\ \sigma_{\tilde{X}\gamma} & \sigma_{\tilde{Z}\gamma} & \sigma_\gamma^2 \end{bmatrix} F^T, \quad (6.12)$$

where the  $F$  matrix is given below:

$$F = \begin{bmatrix} \frac{\tilde{X}}{\sqrt{\tilde{X}^2 + \tilde{Z}^2}} & \frac{\tilde{Z}}{\sqrt{\tilde{X}^2 + \tilde{Z}^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.13)$$

The correlation between  $d$  and  $\gamma$  was computed by normalizing the non-diagonal term of



the covariance matrix between  $d$  and  $\gamma$  (given by 6.12) by the square root of their variances:  $\sigma_{d\gamma}/(\sigma_d \cdot \sigma_\gamma)$ . The typical value was around 5%, proving the orthogonality between the determination of the distance from the camera to the target and the determination of the exposure effect on the position of the ellipse edge (as suggested by Figure 6.2).

## 6.8 Conclusions

We have introduced a new method to rigorously fit the image of a concentric circle target to determine the pose of the camera that sees it. Moreover, we have addressed the problem of the exposure that displaces black and white edges, moving them towards the white direction and leading to systematic effects. Then, we have proposed a way to estimate this displacement and reduce the error.

We performed experimental evaluations of the algorithm in controlled laboratory conditions: We measured the reference target position with total-stations with millimeter-accuracy and captured several images with different exposure times; we then compared the measured distance with the distances calculated by our method at every exposure time. Our method led to highly accurate estimates of the target position with a mean error of 0.37% of the distance on the 25 cm diameter target, the accuracy of the measurements was also not affected by changes in the illumination. We showed the illumination invariance property of the novel algorithm for a single target position by plotting the target distance estimated with the algorithm in [42] and the illumination invariant one. Finally, we observed that the  $\gamma$  parameter is basically uncorrelated ( $< 5\%$ ) with the target position parameters.

## Appendix: Derivatives of $f$ with respect to parameters and observations

The goal of this appendix is to simplify Equation 6.7 and Equation 6.11 in order to clarify the least-square optimization problem. For convenience, we set a new vector  $\tilde{P}_i$  that is equal to  $\tilde{R} \tilde{P}'_i$  when there is no exposure compensation (as in Section 6.5), or to  $\tilde{R}(\tilde{P}'_i + \gamma \tilde{n}_i)$  when there is exposure compensation (as in Section 6.6). Thus, both Equation 6.7 and Equation 6.11 can be simplified as below (where the vector  $\tilde{P}_{XY_i}$  is composed of the two first components of  $\tilde{P}_i$ , and the vector  $\tilde{P}_{Z_i}$  of its last component):

$$\begin{cases} \left\| \begin{pmatrix} \tilde{X} \\ 0 \end{pmatrix} + \lambda_i \tilde{P}_{XY_i} \right\| = r(i) \\ \tilde{Z} + \lambda_i \tilde{P}_{Z_i} = 0 \end{cases} \quad (6.14)$$

## Chapter 6. Compensating over- and underexposure in optical target pose determination

This system of two equations can be solved by removing  $\lambda_i$ . Therefore, for each measurement point, the condition described in Equation 6.15 is imposed:

$$\left\| \begin{pmatrix} \tilde{X} \\ 0 \end{pmatrix} - \frac{\tilde{Z}}{\tilde{P}_{Z_i}} \tilde{P}_{XY_i} \right\|^2 - r(i)^2 = 0. \quad (6.15)$$

To solve the non-linear least-squares optimization problem, we need to compute the derivatives of the conditions with respect to the parameter vectors and the observation vectors.

For convenience, we introduce the notations  $\bar{\bar{P}}$ ,  $\Psi$  and  $\aleph$ :

$$\bar{\bar{P}}_i = \frac{\tilde{P}_{XY_i}}{\tilde{P}_{Z_i}}, \quad (6.16)$$

$$\Psi_i = \begin{pmatrix} \tilde{X} \\ 0 \end{pmatrix} - \tilde{Z} \bar{\bar{P}}_i, \quad (6.17)$$

$$\aleph_i = \|\Psi_i\|^2. \quad (6.18)$$

Hence, Equation 6.15 can be re-written as:

$$\left\| \underbrace{\begin{pmatrix} \tilde{X} \\ 0 \end{pmatrix} - \tilde{Z} \underbrace{\frac{\tilde{P}_{XY_i}}{\tilde{P}_{Z_i}}}_{\bar{\bar{P}}_i}}_{\Psi_i} \right\|^2 - r(i)^2 = 0. \quad (6.19)$$

Figure 6.12 sums up the calculation flow to compute a condition such as 6.15.

Figure 6.12 helps to compute the derivatives of 6.15 with respect to the parameters  $\frac{\partial \aleph}{\partial \tilde{X}}$ ,  $\frac{\partial \aleph}{\partial \tilde{Z}}$ ,  $\frac{\partial \aleph}{\partial \tilde{R}}$

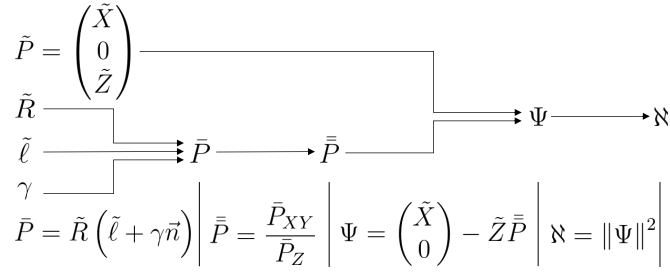


Figure 6.12. Calculation flow. At the top, we see the input, the intermediates steps, and the output; at the bottom, the formulae used to calculate the parameters.

or to the observations  $\frac{\partial \aleph}{\partial \tilde{P}'}$  (here, we have intentionally misused a derivative with respect to a rotation matrix. The rigorous approach is described below).

First, we introduce the computation of the Jacobian matrix of the condition with respect to the observations:

$$\frac{\partial \aleph}{\partial \tilde{P}'} = \frac{\partial \aleph}{\partial \Psi} \frac{\partial \Psi}{\partial \bar{P}} \frac{\partial \bar{P}}{\partial \tilde{P}} \frac{\partial \tilde{P}}{\partial \tilde{P}'}. \tag{6.20}$$

The four (elementary) matrices to compute are derived below:

$$\frac{\partial \aleph}{\partial \Psi} = 2 \left( \Psi_X \quad \Psi_Y \right), \tag{6.21}$$

$$\frac{\partial \Psi}{\partial \bar{P}} = -\tilde{Z} I_2, \tag{6.22}$$

$$\frac{\partial \bar{P}}{\partial \tilde{P}} = \frac{1}{\tilde{P}_Z} \begin{pmatrix} 1 & 0 & -\frac{\tilde{P}_X}{\tilde{P}_Z} \\ 0 & 1 & -\frac{\tilde{P}_Y}{\tilde{P}_Z} \end{pmatrix}, \tag{6.23}$$

$$\frac{\partial \tilde{P}}{\partial \tilde{P}'} = \tilde{R}. \tag{6.24}$$

## Chapter 6. Compensating over- and underexposure in optical target pose determination

Similarly to the previous calculation, it is possible to compute the derivative with respect to  $\tilde{X}$ :

$$\frac{\partial \mathfrak{N}}{\partial \tilde{X}} = \frac{\partial \mathfrak{N}}{\partial \Psi} \frac{\partial \Psi}{\partial \tilde{X}}. \quad (6.25)$$

A new matrix needs to be computed:

$$\frac{\partial \Psi}{\partial \tilde{X}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \quad (6.26)$$

Similarly to the previous calculation, it is possible to compute the derivative with respect to  $\tilde{Z}$ :

$$\frac{\partial \mathfrak{N}}{\partial \tilde{Z}} = \frac{\partial \mathfrak{N}}{\partial \Psi} \frac{\partial \Psi}{\partial \tilde{Z}}, \quad (6.27)$$

and a new matrix needs to be calculated:

$$\frac{\partial \Psi}{\partial \tilde{Z}} = -\bar{P}. \quad (6.28)$$

The Jacobian matrix with respect to  $R$  must use methods to properly handle the rotations, such as the theory of Lie-groups. Here we give a short explanation of how this theory can be applied to bundle adjustment.

The position of the camera is described by two parameters:  $\tilde{X}$  and  $\tilde{Z}$ . These parameters *belong to* an Euclidean space, and, for this reason, we can use the standard update-step:

$$\underbrace{\begin{pmatrix} \hat{X} \\ \hat{Z} \end{pmatrix}}_{Adjusted} = \underbrace{\begin{pmatrix} \tilde{X} \\ \tilde{Z} \end{pmatrix}}_{Approximated} + \begin{pmatrix} \delta \tilde{X} \\ \delta \tilde{Z} \end{pmatrix}. \quad (6.29)$$

However, the rotations do not *belong to* an Euclidean space but to the  $SO(3)$  rotation group.

Therefore, we need to change their update-step:

$$\underbrace{\hat{R}}_{Adjusted} = \exp([\omega]_{\times}) \underbrace{\tilde{R}}_{Approximated}, \quad (6.30)$$

where  $\omega$  does not *belong to* the non-Euclidean rotation group but to the Euclidean tangent plane of  $R$  (see [149]). The derivatives are computed with respect to  $\omega$ . The exponential function allows converting a value from the tangent plane into the Lie-Group.  $[\omega]_{\times}$  is the Skew-symmetric matrix built from  $\omega$ . We can now compute the Jacobian matrix of the condition with respect to the rotation:

$$\frac{\partial \mathfrak{N}}{\partial \omega} = \frac{\partial \mathfrak{N}}{\partial \Psi} \frac{\partial \Psi}{\partial \bar{P}} \frac{\partial \bar{P}}{\partial \tilde{P}} \frac{\partial \tilde{P}}{\partial \omega}. \quad (6.31)$$

For any 3D vector  $x$ , if  $\omega$  and  $R$  are linked by the Equation 6.30, the following formula can be introduced [149]:

$$\frac{\partial (Rx)}{\partial \omega} = - [Rx]_{\times}, \quad (6.32)$$

which can be applied to the Equation 6.31, giving:

$$\frac{\partial \tilde{P}}{\partial \omega} = - [\tilde{P}]_{\times}. \quad (6.33)$$

The Jacobian matrices computed above allow running a bundle adjustment with the five geometric parameters of our initial question (two parameters for the camera position, and three for its rotation). If we want to take into account the exposure compensation (i.e., if the parameter  $\gamma$  is not fixed to 0), we also need to compute the Jacobian matrix with respect to this parameter:

$$\frac{\partial \mathfrak{N}}{\partial \gamma} = \frac{\partial \mathfrak{N}}{\partial \Psi} \frac{\partial \Psi}{\partial \bar{P}} \frac{\partial \bar{P}}{\partial \tilde{P}} \frac{\partial \tilde{P}}{\partial \gamma}. \quad (6.34)$$

## Chapter 6. Compensating over- and underexposure in optical target pose determination

A new matrix needs to be computed:

$$\frac{\partial \bar{P}}{\partial \gamma} = \tilde{R} \tilde{n}. \quad (6.35)$$

The Jacobian calculations described in this appendix must be determined for each condition, i.e., for each measurement. Then, it is possible to build the matrices  $A$  and  $B$ , which are the Jacobian matrices of the function  $f$  with respect to parameters and observations, respectively (where  $f$  is the concatenation of all conditions  $f_i$  defined by the left part of equation 6.15). For each iteration of the Gauss-Newton or the Levenberg-Marquardt algorithm, we can formulate and solve the following linearised equation:

$$\underbrace{\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathfrak{N}_i}{\partial \tilde{X}} & \frac{\partial \mathfrak{N}_i}{\partial \tilde{Z}} & \frac{\partial \mathfrak{N}_i}{\partial \omega_X} & \frac{\partial \mathfrak{N}_i}{\partial \omega_Y} & \frac{\partial \mathfrak{N}_i}{\partial \omega_Z} & \frac{\partial \mathfrak{N}_i}{\partial \gamma} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_A \underbrace{\begin{bmatrix} \delta \tilde{X} \\ \delta \tilde{Z} \\ \omega_X \\ \omega_Y \\ \omega_Z \\ \delta \gamma \end{bmatrix}}_{\delta x} + \underbrace{\begin{bmatrix} \ddots & & 0 \\ & \frac{\partial \mathfrak{N}_i}{\partial \tilde{P}'_i} & \\ 0 & & \ddots \end{bmatrix}}_B \underbrace{\begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}}_v + \underbrace{\begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}}_w = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}. \quad (6.36)$$

The misclosures  $w_i$  are computed from the observations  $\tilde{P}'_i$  thanks to the condition  $f_i$ :

$$w_i = f_i(\tilde{T}, \tilde{R}, \tilde{P}'_i, \gamma). \quad (6.37)$$

This problem is solved in a least-square sense (the L2 norm of the residuals vector  $v$  is minimized) by using minimization algorithms, such as the Gauss-Newton or the Levenberg-Marquardt method. At each step of the algorithm, the update-step is given by Equation 6.29 for the position, and by Equation 6.30 for the orientation. Moreover, data-snooping is performed to remove conditions that have high normalized misclosure.

## Appendix: Ground truth determination with total-stations

The set-up of the validation experiment is shown in Figure 6.9. A network of total-stations and prisms ( $B$ ,  $B'$ ,  $B''$ ) measures the position of the objects in the experiment (i.e., the camera  $A$  and the target  $D$ ). The two total-stations  $B$  and  $B'$  were aligned by using the autocollimation method, in which both total-stations are focused to infinity, they are pointing at each other, and each total-station is aligned on the reticle of the other. The distance between the two total-stations was determined with a laser after replacing one of the total-stations with a prism. An additional prism,  $B''$ , was placed at the back of the set-up to orient the total-stations. We regularly checked the position of this prism to test the stability of the set-up. At the end of the experiment, after replacing the total-stations  $B$  and  $B'$  with two prisms, the prism in  $B''$  was in turn replaced by a total-station to measure the position of  $B$  and  $B'$ . These measurements allowed us to build a strong geodetic network and to set a laboratory frame of reference.

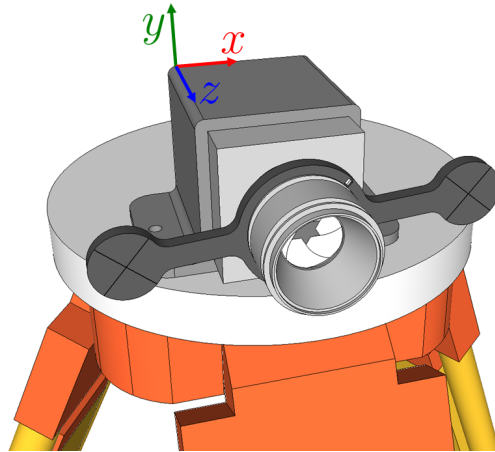


Figure 6.13. Schematic representation of the camera with two targets that the total-stations must measure.

The camera  $A$  was a Ximea with a 16mm TAMRON-Optics lens fixed on a geodetic tripod to ensure stability. The camera was calibrated in-situ with the method described in [21]. The position of the perspective center of the camera was measured by intersection from two total-stations ( $B$  and  $B'$ ). Since the center of perspective of the camera is a virtual point inside the lens, it is not possible to directly aim at it with a theodolite; therefore, we built a set of two targets around the lens (see Figure 6.13). The 3D positions of the crosses (to the left and to the right of the camera) were determined by intersecting the total-stations. We assumed that the perspective center of the camera was in the middle of the segment joining these two crosses. The two targets were made with a Computer Numerical Control Machine with an estimated precision of a tenth of a millimeter. When the targets were mounted on the lens, we considered our setting having a millimeter-level precision in both directions parallel to the CMOS sensor

## **Chapter 6. Compensating over- and underexposure in optical target pose determination**

---

plane (x and y axis). The set of two targets was placed around the diaphragm setting ring. Since the actual position of the perspective center might not be inside of the diaphragm, the position measured by the total-station could differ from the actual one by a few millimeters in the direction of the optical axis (z axis). We then compared the two positions with a resection method.

We placed dozens of ArUco targets at the back of the set-up (C) and we determined with the total-stations their 3D position in the laboratory frame. The ArUco targets were detected in the photos by using ArUco [57]. A resection calculation, which included the camera calibration details (weighted by the provided standard deviations), allowed determining the position of the perspective center of the camera, its orientation relative to the laboratory frame, and a better determination of the camera calibration parameters. This position was then compared with the one obtained by the theodolite, and it showed a difference of 1.6 mm in the image plane (x,y axis) and of 9mm in the optical axis (z axis). The final internal and external camera parameters were determined with an adjustment that included the measurements of the ArUco targets, the position of the perspective center measured with the theodolite, and the previous camera calibrations weighted by the provided standard deviations.

The concentric circle target was printed on a  $50 \times 50 \times 1$  cm plastic board with four reference crosses printed on the corners. We measured the crosses with the total-stations and we then corrected our measurements with a conditional adjustment (i.e., imposing the conditions that the distance between the center of the cross is known and that the edges of the cross form the four corner of a square). These adjusted measurements allowed determining the position and the orientation of the target in the laboratory frame.

Lastly, we computed the ground truth pose and orientation of the camera relative to the target from the absolute camera and target positions and orientations that were calculated in the laboratory frame.



# Conclusion and Perspectives

## Summary of the thesis contributions

The basic theory needed in the scope of this thesis is given in Chapter 1 in which Lie-Groups are shown to be a powerful formalism in use in Bundle-Adjustment, noteworthy because it eases the input of absolute and relative orientation.

In the first part I are given directions for the GCP placement, for camera calibration procedure and modelling as well as flight-planning optimization. The propositions of part I are simple advices to improve classical photogrammetry. They have been shown to improve the final 3D accuracy while reducing costs.

The others two parts II and III show that more exotic surveying methods such as LIDAR-Photo hybridizing and collaborative mapping could improve the quality of the final mapping product, and extend the possibilities of photogrammetry.

## Perspective

### Diffusion of methods to end-users

Some of the methods described in this thesis could be easily implemented in existent professional software.

- The accuracy prediction method described in Chapter 3 have been implemented in C++ and integrated as an *add-on* of the eMotion Flight-planner and mission handling Software. The release of a version of eMotion with this *add-on* to final customers could permit to share this method.
- The method to input the camera parameters with their given full covariance matrix (described in Chapter 2) is very simple since the observation model is trivial. It could be implemented in photogrammetric software, together with a user-friendly manager

## **Chapter 6. Compensating over- and underexposure in optical target pose determination**

---

to monitor the evolution of the camera parameter and its uncertainty for the different mapping projects achieved by the user.

- The method presented in Chapter 6 could be used in the mapKITE processing chain to improve their photogrammetric results.

### **Mid-term technology transfer objective**

The fusion of LIDAR with photogrammetry proposed in Chapter 4 has promising applications. Mapping projects that are currently achieved with copters could be handled with drones. This method inspires further research to improve the fusion. The technology transfer could be achieved via three possible business models.

1. Surveyor firm using a custom drone and custom software to directly sell mapping product to customers. This activity could be achieved by creating a new company or by joining an existing company such as Altametris.
2. Start-up to commercialize the drone and the software to be sold to surveyor firms.
3. Partnership between a research laboratory and 1) a drone manufacturer such as senseFly, Wingtra or Microdrones for the synchronization of the sensors, 2) a photogrammetric software developer company such as Pix4D, Agisoft LLC, SimActive or GEOWN for the implementation of the method.

### **Research perspectives**

The feasibility study described in Chapter 5 motivates the outset of the DoDo project (DrOne DuO project financed by the InnoSeed ENAC grant) which confirms its technical feasibility.

In the scope of this project, [75] proposes a method to recognize and localize the Lower-Drone with respect to the Upper-One which give a  $dm$  accuracy with a height difference up to 100  $m$ . [148] describes the full operating system to fly both drones together. Further research is however needed to complete an autonomous mapping survey in the targeted scenario for this technology: highly cluttered environment such as narrow gorges.

Finally, the concept of collaborative drone mapping could greatly take benefit from the LIDAR-Photogrammetric hybridization method proposed in Chapter 4. An active sensor embedded on the lower drone could be useful to map the bottom of a narrow gorge in the penumbra while being georeferenced by the upper drone.

# **Appendix Part IV**



# A 3D Visualization toolbox for easy display of complex data from Matlab or Python

Available at: [https://github.com/ManuCledat/Matlab\\_Python\\_3D\\_toolbox](https://github.com/ManuCledat/Matlab_Python_3D_toolbox)

This 3D Visualization toolbox permits to create visualization for debugging purposes, data analysis and results rendering. It permits to export usual 3D object used in research in a CAD software. This eases the navigation in the 3D model, its modification and its rendering (e.g. background, shadows, fog could be easily modified). Finally, it permits to visualize, by a simple click the meta-data of the objects, inputted by the user (i.e. object name or number) and calculated by the CAD software, as traditional GIS functionalities.

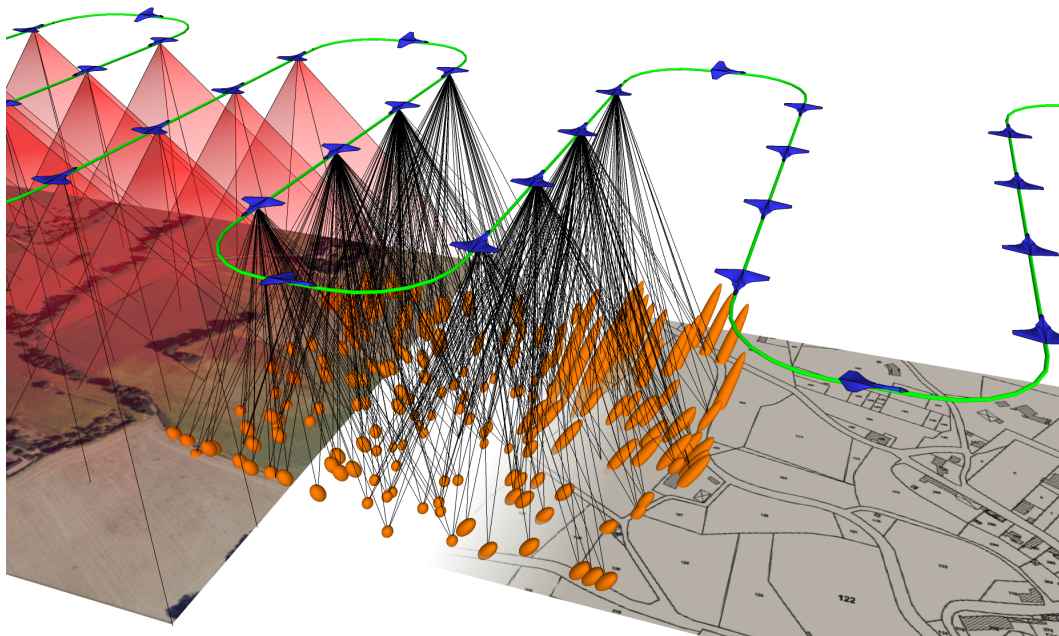


Figure A.1. Example of figure created with the toolbox

## **Introduction**

This toolbox is implemented both in Matlab and in Python. It permits to export 3D data (such as results of research data) in the free software sketchUp [147]. This CAD software where chosen for its user-friendliness and its diversity of use. This software could be used for various applications, such as mechanical engineering, architecture, spatial design, and environmental modeling. Moreover, it is usually chosen for educational purposes as a first CAD software to learn (here, a course designed for 14-15 years-old students [157]). Previous versions of our toolbox (not released) were used for geodetic engineering publications: [33, 127, 71, 35, 36, 37, 38].

## A.1 User Manual for MATLAB

### A.1.1 ruby\_create

#### Description:

Creates and initializes a ruby file and returns a struct which contains id, path and name of the file. Default file is 'script\_ruby\_3d.rb' in the local folder. File must be closed by `ruby_close`.

#### Function Declaration:

```
1 file = ruby_create()
2 file = ruby_create(nameOrPath)
3 file = ruby_create(nameOrPath, name)
```

- `nameOrPath` (optional): 'script\_ruby\_3d.rb'(default) | string  
It is either the name of the file or the path of the file.
- `name` (optional): string  
Name of the file

#### Examples:

```
1 % Create file in local folder with name 'script_ruby_3d.rb'.
2 file = ruby_create();
3
4 % Specify name, .rb appended, 'my_model.rb'.
5 file = ruby_create('my_model');
6
7 % Alternative specifying full path. .rb is appended, 'C:\Users\my_model.rb'.
8 file = ruby_create('C:\Users', 'my_model');
```

### A.1.2 ruby\_close

#### Description:

Completes and closes the ruby file.

#### Function Declaration:

```
1 ruby_close(file)
```

- `file`: file struct  
File structure created with `ruby_create`

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

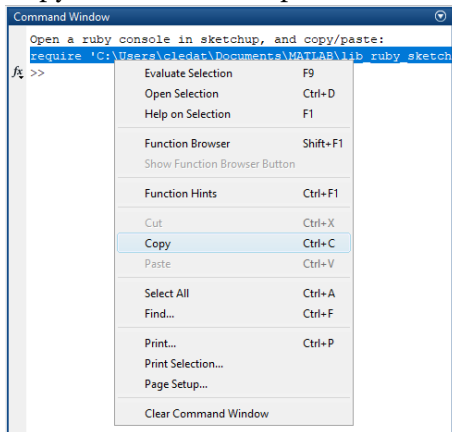
### Examples:

```
1 ruby_close(file);
```

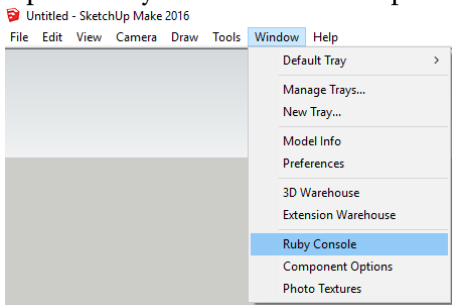
### A.1.3 Input in SketchUp

If the file is closed correctly, it could be inputted in SketchUp as follow.

Copy the command printed on Matlab console.

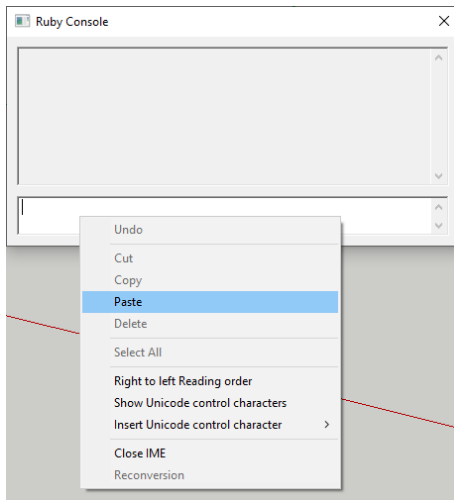


### Open a ruby console in SketchUp



Paste the command line from Matlab





### A.1.4 ruby\_point

#### Description:

Creates isolated points.

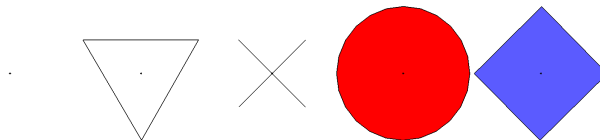


Figure A.2. Different type of points

This function also permits to display point-clouds.

#### Function Declaration:

```
1 ruby_point(file, XYZ, issymbolic, 'symbol', symbol, 'color' , color, ...  
2           'name', name)
```

- `file`: file struct

File structure created with `ruby_create`.

- `xyz`: (Nx3) matrix

Matrix where N is the number of points. Each row corresponds to a point and the columns correspond to three coordinate axis.



Figure A.3. Display of a point-cloud of a church

- `issymbolic` (optional): 0 (default) | 1 | column vector containing 0 and 1 values.  
If it is a numeric value, all points are treated equally in terms of symbols. If it is 0, there is no symbol. If it is 1, a symbol is used for each and every point.  
If it is a column vector, the number of rows should be the same as the number of rows in XYZ matrix.
- `symbol` (optional): 'triangle'(default) | 'cross'| 'circle'| 'square'  
It defines how points are represented in 3D plane by assigning a particular symbol.
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
'n'(default): none  
'w': white  
'r': red  
'o': orange  
'y': yellow  
'g': green

'b': blue  
 'p': pink  
 'k': black

- name (optional): string | column vector containing strings.

If it is a column vector, the number of rows should be the same as the number of rows in XYZ matrix.

### Examples:

```

1 % Construct some points (Here [2,2,2] and [3,3,3])
2 ruby_point(file, [2, 2, 2; 3, 2, 3]);
3
4 % To represent all points with a symbol, set the third argument to 1
5 ruby_point(file, [2, 3, 2; 3, 3, 3], 1);
6
7 % Symbol representation can be activated point by point
8 ruby_point(file, [2, 4, 2; 3, 4, 3], [1; 0]);
9
10 % The symbol is selected with a name-value pair. All points have the same
11 % symbol
12 ruby_point(file, [2, 5, 2; 3, 5, 3], 1, 'symbol', 'cross');
13
14 % With red circle
15 ruby_point(file, [2, 6, 2; 3, 6, 3], 1, 'symbol', 'circle', 'color', 'r');
16
17 % With blue square
18 ruby_point(file, [2, 7, 2; 3, 7, 3], 1, 'symbol', 'square',...
19   'color', [0, 0, 255]);
20
21 % Points can be named globally or individually using a single string or an
22 % array of strings.
23 ruby_point(file, [2, 2, 2; 3, 3, 3], 1, 'name', string('Pts'));
24 ruby_point(file, [2, 2, 2; 3, 3, 3], 1, 'name',...
25   [string('pt1'); string('pt2')]);

```

### A.1.5 ruby\_line

#### Description:

Draws a line or multiple lines.

#### Function Declaration:

```
1 ruby_line(file, XYZ, 'name', name);
```

- file: file struct

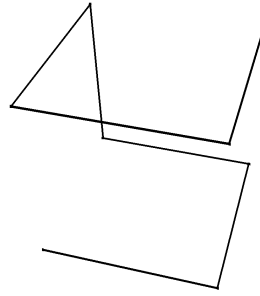


Figure A.4. Example of line drawn by the library

File structure created with `ruby_create`.

- `xyz`: (Nx3) matrix

Matrix where N is the number of points and n should be at least 2. Each row corresponds to a point and the columns correspond to three coordinate axis. The points are connected by a line one by one according to the given order. In other words, first point is connected with the second point, second point is connected with third point and so on.

- `name` (optional): string | column vector containing strings

If it is a column vector, the number of rows should be the same as the number of rows in XYZ matrix.

### Examples:

```
1 % Draw stylized cube with lines.
2 ruby_line(file, [0 0 0; 1 0 0; 1 1 0; 0 1 0; 0 1 1; 0 0 1; 1 0 1; 1 1 1]);
3
4 % Give the drawn line a name.
5 ruby_line(file, [0, 0, 5; 5, 0, 5], 'name', string('Line1'));
```

### A.1.6 `ruby_axis`

#### Description:

Creates a reference coordinate axis defined by the orientation matrix and center point.

#### Function Declaration:

```
1 ruby_axis(file, origin, R, 'name', name)
```

- `file`: file struct

File structure created with `ruby_create`.

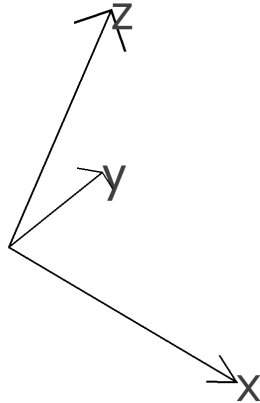


Figure A.5. Axis definition

- `origin`: (1x3) matrix  
It contains x, y, z coordinates of the new origin
- `R`: (3x3) matrix  
Orthogonal orientation matrix.
- `name` (optional): string

**Examples:**

```

1 R = [1, 0, 0; 0, cos(1), sin(1); 0, -sin(1), cos(1)];
2
3 % Create coordinate system at 4,4,4.
4 ruby_axis(file, [4, 4, 4], R);
5
6 % Create coordinate system at 4,5,4 named Origin2
7 ruby_axis(file, [4, 5, 4], R, 'name', string('Origin2'));

```

**A.1.7 ruby\_pose****Description:**

Constructs a pose. The photo is taken in the -Z direction.

**Function Declaration:**

```

1 ruby_pose(file, P, R, 'focal', focal, 'width', width, ...
2           'height', height, 'color', color, 'name', name)

```

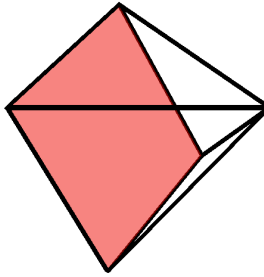


Figure A.6. Camera pose

- `file`: file struct  
File structure created with `ruby_create`.
- `p`: (1x3) matrix  
It contains x, y, z coordinates of the center of the projection of the pose.
- `R`: (3x3) matrix  
It is an orthogonal matrix.
- `focal` (optional): 0.2 (default) | Numeric value  
It is in meters.
- `width` (optional): 0.1 (default) | Numeric value  
It is in meters.
- `height` (optional): 0.1 (default) | Numeric value  
It is in meters.
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
'n'(default): none  
'w': white  
'r': red  
'o': orange  
'y': yellow  
'g': green  
'b': blue

'p': pink

'k': black

- name (optional): string.

### Examples:

```

1 R = [1, 0, 0; 0, cos(1), sin(1); 0, -sin(1), cos(1)];
2
3 % Basic default pose.
4 ruby_pose(file, [5,5,5], R);
5
6 % Set custom focal length and image size.
7 ruby_pose(file, [6,5,5], R, 'focal', 0.1, 'width', 0.3, 'height', 0.2);
8
9 % Labeled.
10 ruby_pose(file, [7,5,5], R, 'focal', 0.1, 'width', 0.3, ...
11     'height', 0.2, 'name', string('f0.1w0.3h0.2'));
12
13 % Set the image plane from undefined to blue.
14 ruby_pose(file, [8,5,5], R, 'focal', 0.1, 'width', 0.3, ...
15     'height', 0.2, 'color', 'b');
16
17 % Set the image plane from undefined to red with hint of blue.
18 ruby_pose(file, [9,5,5], R, 'focal', 0.1, 'width', 0.3, ...
19     'height', 0.2, 'color', [255, 0, 63]);

```

### A.1.8 ruby\_ellipsoid

#### Description:

Creates an ellipsoid.

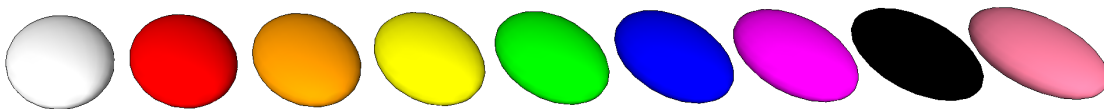


Figure A.7. Ellipsoids

An ellipsoid is a surface defined by a center point  $P$  and a covariance matrix (also called tensor)  $K$ .

$$v^T K v = 1 \tag{A.1}$$

$v$  is the vector between a point of the 3D space and the center of the ellipsoid:  $P$ .

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

### Function Declaration:

```
1 ruby_ellipsoid(file, P, K, 'color', color, ...
2   'name', name)
```

- `file`: file struct  
File structure created with `ruby_create`.
- `P`: (1x3) matrix  
It contains x, y, z coordinates of the center P of the ellipsoid.
- `K`: (3x3) matrix  
It contains the variance-covariance matrix of the ellipsoid.
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
`'n'` (default): none  
`'w'`: white  
`'r'`: red  
`'o'`: orange  
`'y'`: yellow  
`'g'`: green  
`'b'`: blue  
`'p'`: pink  
`'k'`: black
- `name` (optional): string.

### Examples:

```
1 K = [1.666 -0.424 0.244; -0.424 2.961 -0.558; 0.244 -0.558 4.121];
2
3 % Default ellipse
4 ruby_ellipsoid(file, [0,0,5], K);
5
6 % With preset color
7 ruby_ellipsoid(file, [0,0,10], K, 'color', 'r');
8
9 % With RGB color
10 ruby_ellipsoid(file, [0,0,15], K, 'color', [248,123,156]);
11
12 % With label
13 ruby_ellipsoid(file, [0,0, 20], K, 'name', string('Pos1'));
```



### A.1.9 ruby\_plane

#### Description:

Constructs a polygone.

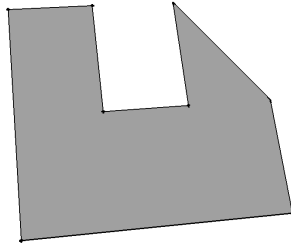


Figure A.8. Example of polygone drawn by the function `ruby_plane`

#### Function Declaration:

```
1 ruby_plane(file, XYZ, 'texture', texture, 'color', color, 'name', name)
```

- `file`: file struct  
File structure created with `ruby_create`.
- `XYZ`: (Nx3) matrix  
It is a numeric matrix consisting of points that define a plane. Points must not be colinear, but must be coplanar.
- `texture` (optional): " (default)  
An image file path with extension `'.png'`, `'.jpg'` or `'.jpeg'`
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
'n' (default): none  
'w': white  
'r': red  
'o': orange  
'y': yellow  
'g': green  
'b': blue

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

'p': pink

'k': black

- name (optional): string

### Examples:

```
1 % Inclined plane
2 ruby_plane(file_id, [ 5 , 0 , 0 ;...
3                     5 , 2 , 2 ;...
4                     6 , 2 , 2 ;...
5                     6 , 1 , 1 ;...
6                     7 , 1 , 1 ;...
7                     7 , 2 , 2 ;...
8                     8 , 1 , 1 ;...
9                     8 , 0 , 0 ])
10
11 % Triangle with texture and name.
12 ruby_plane(file, [0, 0, 3; 0, 3, 0; 2, 0, 0],...
13             'texture', '/images/rainbow.jpeg', 'name', string('myplane'));
14
15 % Rectangle with red color.
16 ruby_plane(file, [0, 1, 3; 0, 1, 5; 1, 2, 5; 1, 2, 3], 'color', 'r');
17
18 % Triangle with blue color.
19 ruby_plane(file, [0, 4, 3; 0, 7, 0; 2, 4, 0], 'color', [0, 0, 255]);
```

### A.1.10 ruby\_tin

#### Description:

Constructs a triangulated irregular network (tin).

#### Function Declaration:

```
1 ruby_tin(file, XYZ , triangles, 'texture', texture, 'color', color, ...
2         'name', name)
```

- file: file struct

File structure created with `ruby_create`.

- xyz: (Nx3) matrix

It is a numeric matrix consists of position of points. N is the number of points.

- triangles: (Mx3) matrix

It is a numeric matrix which consists of indexes of triangle points. M is the number of triangles. The triangles are best created using the `delaunay` function.

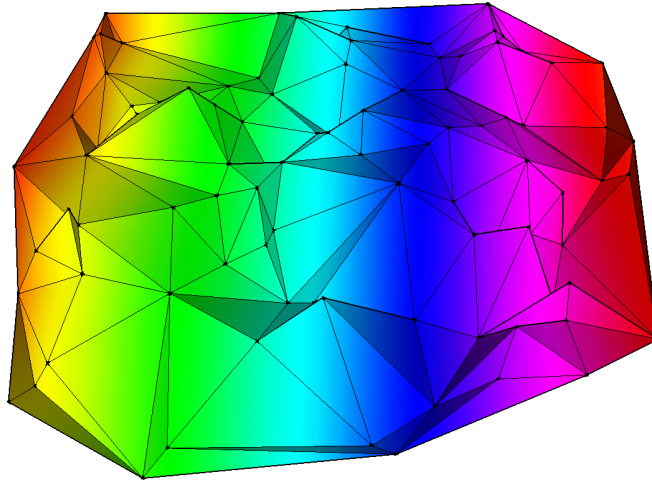


Figure A.9. Example of Digital Elevation Model created by `ruby_tin` with a gradient texture overlaid

- `texture` (optional): " (default)  
An image file path with extension `' .png', ' .jpg'` or `' .jpeg'`
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
`'n'` (default): none  
`'w'`: white  
`'r'`: red  
`'o'`: orange  
`'y'`: yellow  
`'g'`: green  
`'b'`: blue  
`'p'`: pink  
`'k'`: black
- `name` (optional): string.

#### Examples:

```
1 Points = [ 10 * rand(100,1) + 10 , 10*rand(100,1) , randn(100,1) ];  
2 triangles = delaunay(Points(:, 1:2));  
3
```

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

```
4 ruby_tin(file, Points, triangles, 'texture', '/images/rainbow.jpeg',...
5         'name', string('Elevation'));
```

### A.1.11 ruby\_antenna

#### Description:

Constructs an antenna or multiple antennas.

#### Function Declaration:

```
1 ruby_antenna(file, P, 'color', color, 'name', name)
```

- `file`: file struct  
File structure created with `ruby_create`.
- `P`: (N×3) matrix  
A numeric matrix consisting of antenna positions. N is the number of antennas.
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
'r'(default): red  
'w': white  
'o': orange  
'y': yellow  
'g': green  
'b': blue  
'p': pink  
'k': black
- `name` (optional): string | column vector containing strings

#### Examples:

```
1 % Create default (red) antenna at 2,2,2
2 ruby_antenna(file, [2, 2, 2]);
3
4 % Green Antenna
5 ruby_antenna(file, [2, 2, 3], 'color', 'g');
```

```

6
7 %Blue Antenna
8 ruby_antenna(file, [2, 2, 4], 'color', [0, 0, 255]);
9
10 % Add global antenna name
11 ruby_antenna(file, [2, 2, 5], 'name', string('Ant1'));

```

### A.1.12 ruby\_arrow

#### Description:

Constructs an arrow or quiver plot.

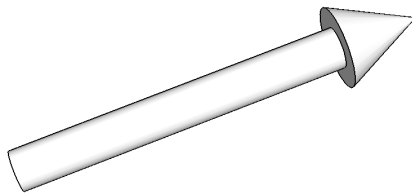


Figure A.10. Example of single arrow

#### Function Declaration:

```
1 ruby_arrow(file, P, v, 'color', color, 'name', name)
```

- `file`: file struct  
File structure created with `ruby_create`.
- `P`: (3x1) matrix  
Is a vector with the coordinates.
- `v`: (3x1) matrix  
Is a vector representing the direction of the arrow.
- `color` (optional): char | [R, G, B]  
Either one of the following colors or a set of RGB values.  
  - 'n' (default): none
  - 'w': white
  - 'r': red
  - 'o': orange

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

'y': yellow

'g': green

'b': blue

'p': pink

'k': black

- name (optional): string

### Examples:

```
1 % Draw arrow at 5,5,5 pointing along x axis
2 ruby_arrow(file, [5; 5; 5], [1; 0; 0]);
3
4 % Red arrow
5 ruby_arrow(file, [5; 5; 5], [0; 1; 0], 'color', 'r');
6
7 % Labeled blue arrow
8 ruby_arrow(file, [5; 5; 5], [0; 0; 1], 'color', [0, 0, 255], ...
9         'name', string('Arrow1'));
```

## A.2 User Manual for Python

### A.2.1 ruby\_create

**Description:**

Creates and initializes a ruby file. Returns file object `file`.

**Function Declaration:**

```
1 file = ruby_create(name_or_path = 'script_ruby_sketchup.rb')
```

- `nameOrPath` (optional): string  
File name of output file. `.rb` file extension is mandatory.

**Examples:**

```
1 # Open 'script_ruby_sketchup.rb' in local folder.
2 file = ruby_create()
3
4 # Open 'script3d.rb' in local folder.
5 file = ruby_create('script3d.rb')
6
7 # Open '/Users/myUser/Documents/script3d.rb'.
8 file = ruby_create('/Users/myUser/Documents/script3d.rb')
```

### A.2.2 ruby\_close

**Description:**

Completes and closes the ruby file.

**Function Declaration:**

```
1 ruby_close(file)
```

- `file`: file struct  
File structure created with `ruby_create`.

**Examples:**

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

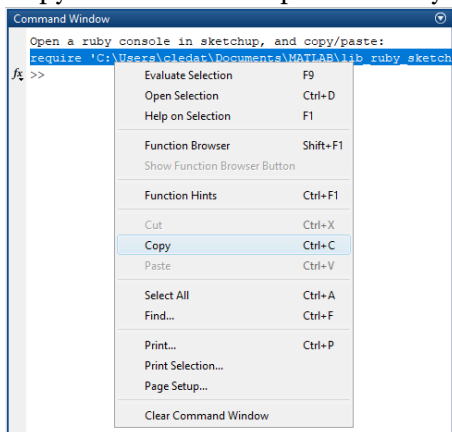
---

```
1 ruby_close(file);
```

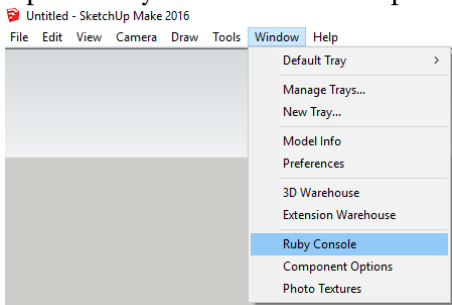
### A.2.3 Input in SketchUp

If the file is closed correctly, it could be inputted in SketchUp as follow.

Copy the command printed on Python console.

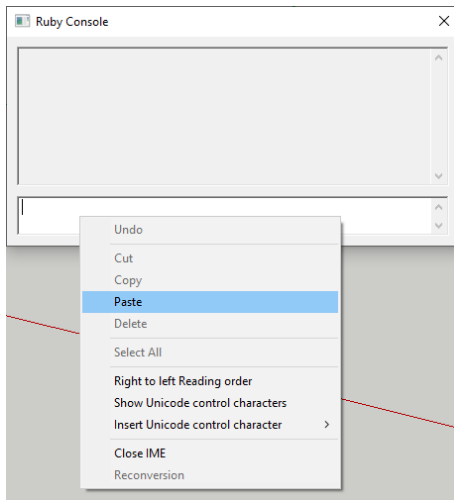


### Open a ruby console in SketchUp



Paste the command line from Python console





### A.2.4 ruby\_point

#### Description:

Creates isolated points.

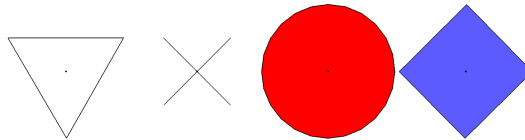


Figure A.11. Different type of points

#### Function Declaration:

```
1 ruby_point(file, XYZ, issymbolic = 0, symbol = 'triangle',  
  color = 'n', name = '');
```

- file: file struct

File structure created with `ruby_create`.

- XYZ: (Nx3) numpy.array

N is the number of points. Each row corresponds to a point and the columns correspond to the three coordinate axis.

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

- `issymbolic` (optional): `int` | `(Nx1) numpy.array`  
Single integer or column vector containing 0 and 1 values.  
If it is a numeric value, all points are treated equally in terms of symbols. If it is 0, there is no symbol. If it is 1, a symbol is used for each and every point.  
If it is a column vector, the number of rows should be the same as the number of rows in XYZ matrix.
- `symbol` (optional): `'triangle'`(default) | `'cross'` | `'circle'` | `'square'`  
It defines how points are represented in 3D plane by assigning a particular symbol.
- `color` (optional): `char`  
One of the following colors.  
`'n'`(default): none  
`'w'`: white  
`'r'`: red  
`'o'`: orange  
`'y'`: yellow  
`'g'`: green  
`'b'`: blue  
`'p'`: pink  
`'k'`: black
- `name` (optional): `string` | `(Nx1) numpy.array of strings`  
If it is a column vector, each point is given a separate name. Otherwise all points share the same name.

### Examples:

```
1 # Draws a single point
2 ruby_point(file, np.array([[ -5, 2, 2 ]]))
3
4 # Draws 2 points with default symbol(triangle) and default
   color(white)
5 ruby_point(file, np.array([[ -4, 2, 2 ], [ -3, 2, 2 ]]),
   issymbolic = 1)
6
```

```

7 # Draws 2 points with default symbol(triangle) and color
  black
8 ruby_point(file, np.array([[ -2, 2, 2], [ -1, 2, 2]]),
  issymbolic = 1,
9     color = 'k')
10
11 # Draws 2 points with color red
12 # One with symbol(square) and other without symbols
13 # Both share the same name ('points')
14 ruby_point(file, np.array([[2, 2, 2], [3, 2, 2]]),
15     issymbolic = np.array([[1], [0]]), symbol = 'square',
16     color = 'r',
17     name = "points")
18
19 # Draws 2 points with color green
20 # One without symbols and other with symbol(circle)
21 # They are named 'point1' and 'point2' respectively
22 ruby_point(file, np.array([[4, 2, 2], [5, 2, 2]]),
23     issymbolic = np.array([[0], [1]]), symbol = 'circle',
24     color = 'g',
25     name = np.array([["point1"], ["point2"]]))

```

### A.2.5 ruby\_line

#### Description:

Draws a line or polyline.

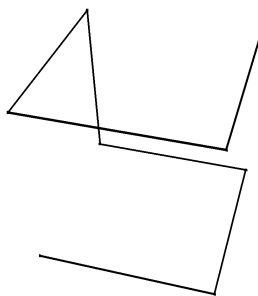


Figure A.12. Example of line drawn by the library

#### Function Declaration:

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

```
1 ruby_line(file, XYZ, name = 'name');
```

- file: file struct

File structure created with `ruby_create`.

- XYZ: (Nx3) numpy.array N is the number of points and should be at least 2. Each row corresponds to a point and the columns correspond to three coordinate axis. The points are connected by a line one by one according to the given order. In other words, the first point is connected with the second point, the second point is connected with the third point and so on.

- name (optional): string | ((N-1)x1) numpy.array of strings

If it is a column vector, each segment is given a separate name. Otherwise all segments share the same name.

### Examples:

```
1 # Draws 3 connected lines
2 ruby_line(file, np.array([[0, 3, 2], [0, 3, 4], [0, 5, 4],
3   [0, 5, 2]]))
4 # Draws 2 connected lines with the same name ('lines')
5 ruby_line(file, np.array([[0, 3, 4], [2, 3, 4], [2, 3, 2]]),
6   name = "lines")
7 # Draws 2 connected lines with different names ('line1', '
8   line2')
9 ruby_line(file, np.array([[0, 5, 4], [2, 5, 4], [2, 5, 2]]),
10   name = np.array(["line1", "line2"]))
```

### A.2.6 ruby\_axis

#### Description:

Creates a reference coordinate axis by changing the orientation and center point of the original axis.

#### Function Declaration:

```
1 ruby_axis(file, P, R, name = '')
```

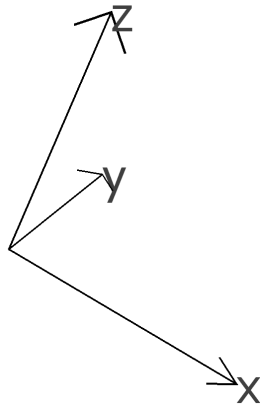


Figure A.13. Axis definition

- `file`: file struct  
File structure created with `ruby_create`.
- `P`: (1x3) `numpy.array`  
Contains `x`, `y`, `z` coordinates of the axis origin.
- `R`: (3x3) `numpy.array`  
Orthogonal rotation matrix representing the orientation of the axis.
- `name` (optional): string

**Examples:**

```

1 # Orthogonal orientation matrix
2 R = numpy.array([[1, 0, 0], [0, cos(1), sin(1)], [0, -sin(1),
   cos(1)]])
3 # Draws an axis with name ('reference_axis')
4 ruby_axis(file, np.array([[4, 4, 4]]), R, name = "
   reference_axis")

```

**A.2.7 ruby\_pose****Description:**

Constructs a pose.

**Function Declaration:**

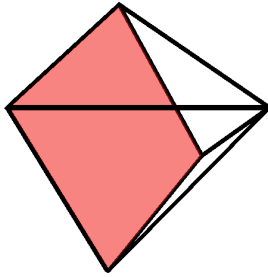


Figure A.14. Camera pose

```
1 ruby_pose(file, P, R, focal = 0.2, width = 0.1, height = 0.1,  
    color = 'n', name = '')
```

- **file**: file struct  
File structure created with `ruby_create`.
- **P**: (1x3) `numpy.array`  
Contains x, y, z coordinates of the center of the projection of the pose.
- **R**: (3x3) `numpy.array`  
Orthogonal rotation matrix representing the orientation of the pose.
- **focal** (optional): float  
In meters.
- **width** (optional): float  
In meters.
- **height** (optional): float  
In meters.
- **color** (optional): char  
One of the following colors.  
‘n’(default): none  
‘w’: white  
‘r’: red  
‘o’: orange  
‘y’: yellow

'g': green

'b': blue

'p': pink

'k': black

- name (optional): string

### Examples:

```

1 # Orthogonal Matrix
2 R = np.array([[1, 0, 0], [0, np.cos(1), np.sin(1)], [0, -np.
   sin(1), np.cos(1)]])
3
4 # Draws a pose with default focal length (0.2), width (0.1),
   height (0.1)
5 # and color (weight) options
6 ruby_pose(file, np.array([[5, 5, 5]]), R)
7
8 # Draws a pose with a name('pose1') and focal length (0.1)
9 ruby_pose(file, np.array([[6, 5, 5]]), R, focal = 0.1, name
   = "pose1")
10
11 # Draws a pose with the given focal length (0.6) and color (
   orange)
12 ruby_pose(file, np.array([[7, 5, 5]]), R, focal = 0.6, color
   = 'o')
13
14 # Draws a pose with the given focal length (0.8) and width
   (0.3)
15 ruby_pose(file, np.array([[8, 5, 5]]), R, focal = 0.8, width
   = 0.3, color = 'p')
16
17 # Draws a pose with the given focal length (1), width (0.3),
   height (0.2)
18 # and color (red)
19 ruby_pose(file, np.array([[9, 5, 5]]), R, focal = 1, width =
   0.3, height = 0.2,
20 color = 'r')
```

### A.2.8 ruby\_ellipsoid

**Description:**

Creates an ellipsoid.

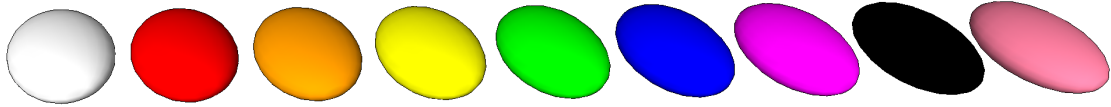


Figure A.15. Ellipsoids

An ellipsoid is a surface defined by a center point  $P$  and a covariance matrix (also called tensor)  $K$ .

$$v^T K v = 1 \tag{A.2}$$

$v$  is the vector between a point of the 3D space and the center of the ellipsoid:  $P$ .

**Function Declaration:**

```
1 ruby_ellipsoid(file, P, K, texture = '', color = 'n', name = '')
```

- `file`: file struct  
File structure created with `ruby_create`.
- `P`: (1x3) `numpy.array`  
Contains `x`, `y`, `z` coordinates of the center of the ellipsoid.
- `K`: (3x3) `numpy.array`  
Contains the variance-covariance matrix of the ellipsoid.
- `texture` (optional): string  
Image file path with extension `‘.png’, ‘jpg’` or `‘.jpeg’`.
- `color` (optional): char  
One of the following colors.  
`‘n’` (default): none



'w': white  
'r': red  
'o': orange  
'y': yellow  
'g': green  
'b': blue  
'p': pink  
'k': black

- name (optional): string

### Examples:

```
1 # Variance-covariance matrix
2 R = np.array([[1, 0, 0],
3              [0, np.cos(1), np.sin(1)],
4              [0, -np.sin(1), np.cos(1)]])
5 K = np.linalg.inv(R) * np.array([[1, 0, 0], [0, 4, 0], [0, 0,
6              9]]) * R
7
8 # Draws an ellipsoid with color(red) given variance-
9   covariance matrix R
10 ruby_ellipsoid(file, np.array([[0, 0, 10]]), K, color = 'r')
11
12 # Draws an ellipsoid with texture given variance-covariance
13   matrix R
14 ruby_ellipsoid(file, np.array([[0, 0, 20]]), K, texture = '/
15   images/color.jpg')
16
17 # Draws an ellipsoid with color(yellow) and name('error
18   ellipsoid')
19 # given variance-covariance matrix K
20 ruby_ellipsoid(file, np.array([[0, 0, 30]]), K, color = 'y',
21   name = "error ellipsoid")
```

### **A.2.9 ruby\_plane**

**Description:**

Constructs a polygone.

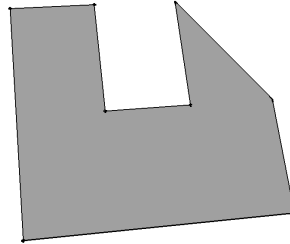


Figure A.16. Example of polygone drawn by the function `ruby_plane`

**Function Declaration:**

```
1 ruby_plane(file, XYZ, texture = '', color = 'n', name = '')
```

- `file`: file struct  
File structure created with `ruby_create`.
- `XYZ`: (Nx3) numpy.array  
Numeric matrix that consists of N points that define a plane, points must not be colinear, but must be coplanar. N must be larger than 3.
- `texture` (optional): string  
Image file path with extension `'.png'`, `'jpg'` or `'.jpeg'`.
- `color` (optional): char  
One of the following colors.  
`'n'` (default): none  
`'w'`: white  
`'r'`: red  
`'o'`: orange  
`'y'`: yellow  
`'g'`: green  
`'b'`: blue

'p': pink  
 'k': black

- name (optional): string

### Examples:

```

1  % Inclined plane
2  ruby_plane(file_id, [ 5 , 0 , 0 ;...
3                      5 , 2 , 2 ;...
4                      6 , 2 , 2 ;...
5                      6 , 1 , 1 ;...
6                      7 , 1 , 1 ;...
7                      7 , 2 , 2 ;...
8                      8 , 1 , 1 ;...
9                      8 , 0 , 0 ])
10
11 # Draws a blue plane with the label 'myplane'
12 ruby_plane(file, np.array([[0, 1, 3], [0, 1, 5], [1, 2, 5],
13                          [1, 2, 3]]),
14           color = 'b', name = 'myplane')
```

### A.2.10 ruby\_tin

**Description:** Constructs a triangulated irregular network (tin).

#### Function Declaration:

```

1  ruby_tin(file, XYZ , triangles, texture = '', color = 'n',
2          name = '')
```

- file: file struct  
File structure created with ruby\_create.
- XYZ: (Nx3) numpy.array  
Numeric matrix of point coordinates. N is the number of points.
- triangles: (Mx3) numpy.array  
Numeric matrix of point indexes as returned by delaunay.simplices. M is th number of triangles.

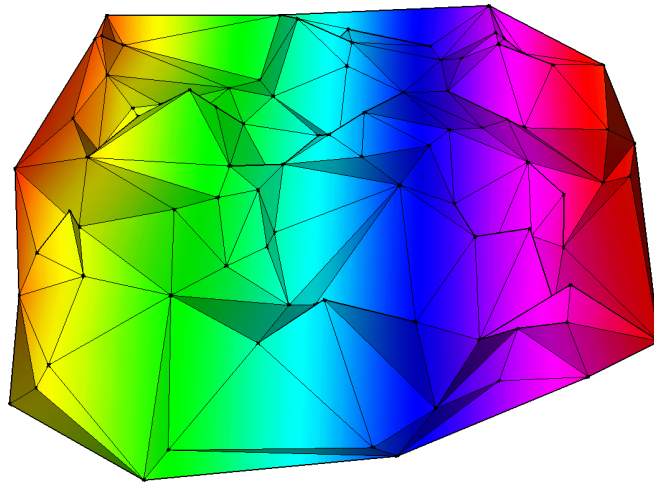


Figure A.17. Example of Digital Elevation Model created by `ruby_tin` with a gradient texture overlaid

- `texture` (optional): string  
Image file path with extension `'.png'`, `'jpg'` or `'.jpeg'`.
- `color` (optional): char  
One of the following colors.  
`'n'` (default): none  
`'w'`: white  
`'r'`: red  
`'o'`: orange  
`'y'`: yellow  
`'g'`: green  
`'b'`: blue  
`'p'`: pink  
`'k'`: black
- `name` (optional): string

### Examples:

```
1 # Create dataset
2 p1 = 10 * np.random.rand(100, 1) + 10
```

```
3 p2 = 10 * np.random.rand(100, 1)
4 p3 = np.random.rand(100, 1)
5
6 points = np.concatenate((p1, p2), axis = 1)
7 points = np.concatenate((points, p3), axis = 1)
8
9 # Extract triangles
10 triangles = Delaunay(points[:, 0:2])
11
12 # Create tin with texture
13 ruby_tin(file, points, triangles.simplices.copy(),
14         texture = '/images/rainbow.jpeg')
```

### A.2.11 ruby\_theodolite

#### Description:

Constructs a theodolite.

#### Function Declaration:

```
1 ruby\_theodolite(file, P, name = '')
```

- file: file struct  
File structure created with `ruby_create`.
- P: (1x3) numpy.array  
Numeric matrix consisting of theodolite coordinates.
- name (optional): string

#### Examples:

```
1 ruby_theodolite(file, numpy.array([[1, 2, 0]]), name = "
  theodolite")
```

### A.2.12 ruby\_antenna

#### Description:

Constructs an antenna or multiple antennas.

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

### Function Declaration:

```
1 ruby\_antenna(file, P, name = '')
```

- **file:** file struct  
File structure created with `ruby_create`.
- **P:** (Nx3) `numpy.array`  
Numeric matrix consisting of antenna positions. N is the number of antennas.
- **name (optional):** string | (Nx1) `numpy.array` of strings  
If it is a column vector, each antenna is given a separate name. Otherwise all antennas share the same name.

### Examples:

```
1 ruby_antenna(file, numpy.array([[5, 5, 3]]), name = "antenna")
```

### A.2.13 `ruby_resection`

#### Description:

Constructs a resection object.

#### Function Declaration:

```
1 ruby_resection(file, P, pos_antennas, name = '')
```

- **file:** file struct  
File structure created with `ruby_create`.
- **P:** (1x3) `numpy.array`  
Numeric matrix consisting of theodolite coordinates.
- **pos\_antennas:** (Nx3) `numpy.array`  
Numeric matrix consisting of antenna positions. N is the number of antennas.
- **name (optional):** string

### Examples:

```
1 ruby_resection(file, [5, 5, 3], [[0, -10, 0], [0, -2, 0],  
   [-3, 0, 0]],  
2     name = "resection")
```

### A.2.14 ruby\_arrow

#### Description:

Draws an arrow / quiver plot.

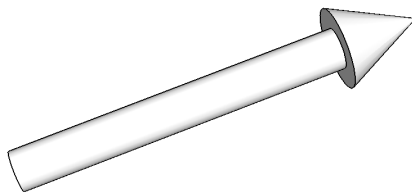


Figure A.18. Example of single arrow

#### Function Declaration:

```
1 ruby_arrow(file, P, v, color = 'n', name = '')
```

- **file:** file struct  
File structure created with `ruby_create`.
- **P:** (3x1) `numpy.array`  
Coordinate vector.
- **v:** (3x1) `numpy.array`  
Orientation vector.
- **color (optional):** char  
One of the following colors.  
'n'(default): none  
'w': white  
'r': red  
'o': orange

## Appendix A. 3D Visualization toolbox for easy display of complex data from Matlab or Python

---

'y': yellow  
'g': green  
'b': blue  
'p': pink  
'k': black

- name (optional): string

### Examples:

```
1 # Draws an arrow at 5,5,5, pointing along x axis
2 ruby_arrow(file, np.array([[5], [5], [5]]), np.array([[1],
3               [0], [0]]))
4 # Draws a red arrow
5 ruby_arrow(file, np.array([[5], [5], [5]]), np.array([[0],
6               [1], [0]]), \
7               color = 'r')
8 # Draws a green arrow with label
9 ruby_arrow(file, np.array([[5], [5], [5]]), np.array([[0],
10              [0], [1]]), \
11              color = 'g', name = "arrow3")
```

### Acknowledgements

The author would like to thank in particular Beat Geissmann for his technical contribution in formatting both Matlab and Python codes. Moreover, we would like to thank Pinar Ezgi Çöl and Simon Lütolf for their help, and the supervisors: Jan Skaloud, Jean-François Hullo and Bertrand Merminod who encouraged for the development of this toolbox.



# B Bundle-Adjustment Algorithm: Code description

This appendix presents a custom Bundle-Adjustment algorithm developed during this PhD. The global theory is presented in Chapter 1, then, a benchmarking using a wide range of different parameters is performed in Chapter 2. Chapter 3 uses a C++ version of this algorithm together with a photogrammetric network simulator. Chapter 4 and 5 use further extensions of this algorithm.

This user manual is intended both for users and for developers. First, the Section B.1 gives just the necessary instructions to import the data and get a first result. A TEST ME example is also provided. Section B.2 presents all the adjustments parameters to test and compare different configurations whose results could be assessed with the tools presented in Section B.3. Finally, Section B.4 explains the methods used in the Bundle-Adjustment algorithm. The equations presented in this section are referred in the code.

## B.1 Quick Start: import file and process it

### B.1.1 Input file

#### Images Observations from Photoscan

This Bundle-adjustment uses as inputs the image observations outputted by the software *Agisoft Metashape*. These measurements are obtained by creating an *Agisoft Metashape* project, in which the images are inputted. To match the images, the *adjustment* procedure must be clicked. Then, the user must to click on `file/Export/Export Cameras . . .` as in Figure B.1.

## Appendix B. Bundle-Adjustment Algorithm: Code description

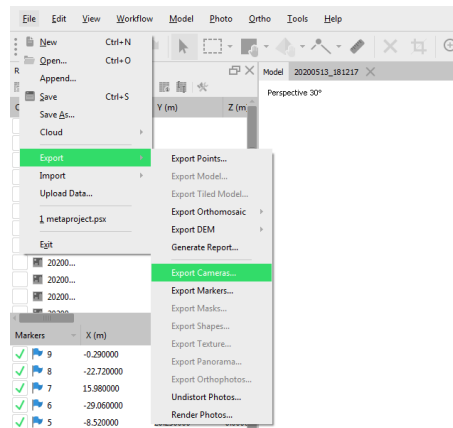


Figure B.1. Click File/export/export cameras...

The file that will be used as input in our custom Bundle-Adjustment is an *.xml Blocks exchange file*: Figure B.2.

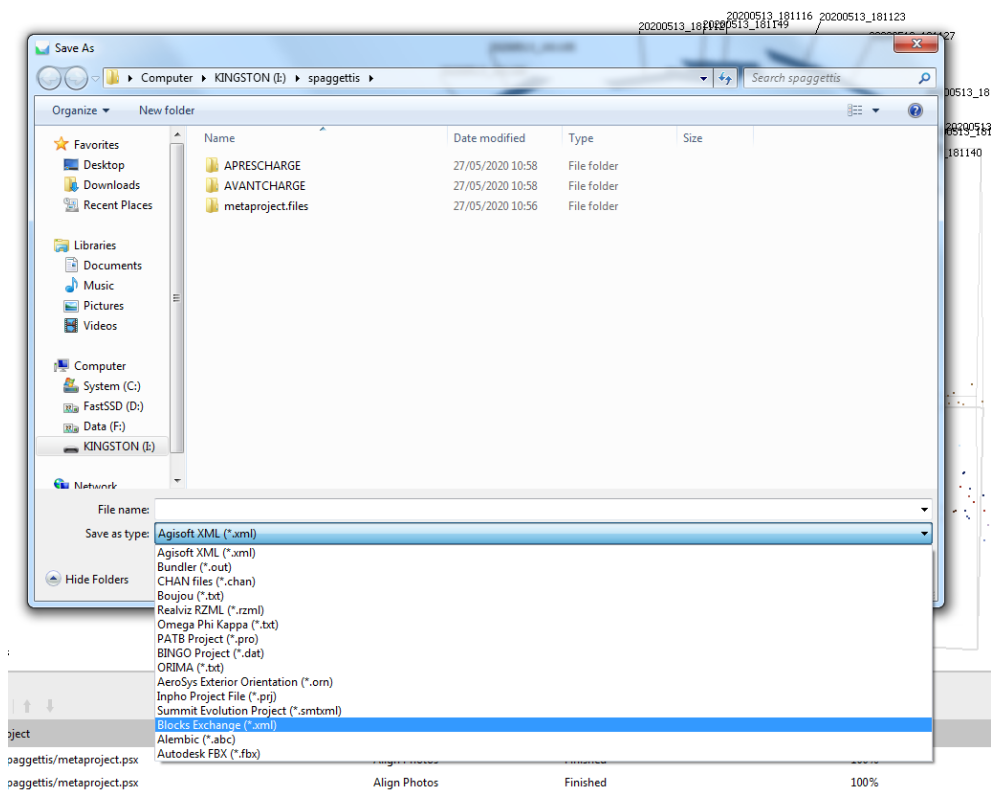


Figure B.2. Choose Blocks Exchange .xml

---

## B.1. Quick Start: import file and process it

This first step is crucial for the functioning of the Bundle-Adjustment. If the Bundle-Adjustment fails, it is probably because this step has not been proceed properly. If some photos are not aligned in photoscan software, it is recommended to create a new project including only the photos that can be aligned.

### Trajectory

The trajectory is computed by integrating the IMU data with the GNSS positioning. The file that must be inputted in our custom Bundle-Adjustment is a `.fmb` file.

### Creating a project in our custom Bundle-Adjustment

Our custom Bundle-Adjustment software is a *Matlab* function called `topo-bun-f` which takes as input a structure in which all the parameters of the adjustment are given. This structure is presented in detail in the section B.2. Three elements are important to create a project. The `PROJECT-NAME`, the `PROJECT-STORAGE-FOLDER` and the `NAME-PHOTO-INPUT`. A time-consuming step is the parsing of the `.xml` file. This parsing is achieved at the first run of `topo-bun-f` on a new project. A file called `imported.mat` is created in the `PROJECT-STORAGE-FOLDER`. The given `PROJECT-NAME` is stored in the `imported.mat` file together with the images observations. If the `topo-bun-f` function is run again on the same project with different parameters or a different GCP configuration, the `.xml` file is not parsed again. To check that the correct `imported.mat` is imported, the `PROJECT-NAME` must remain the same at each run of the same project (if not, a warning pop-up).

### The GCP/CP Graphical User Interface

The GCP/CP Graphical User Interface could be used to select which point must be used as a Ground Control Point (GPS) or as a Checkpoint (CP). This functionality is opened if the `GCP-GUI` element of the input structure is set to `true`.

## Appendix B. Bundle-Adjustment Algorithm: Code description

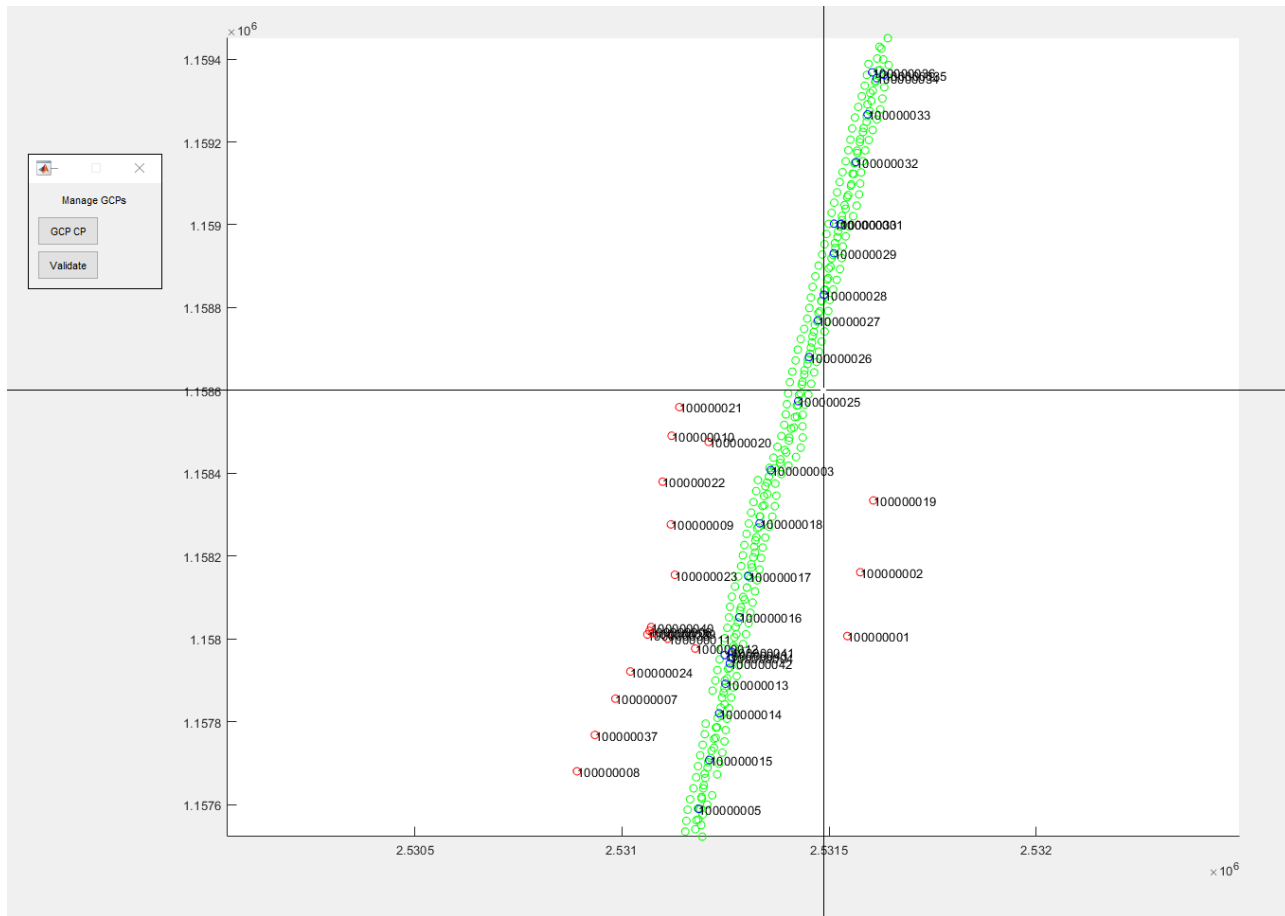


Figure B.3. GCP/CP Graphical User Interface. Green: Images position, Red: GCPs, Blue: CPs

The GCPs are represented on the map (right part of Figure B.3) by red dots, the CP are represented by blue dots. GCPs could be transformed in CP (and conversely) by clicking on the GCP CP button (left box in Figure B.3) and then by clicking on (or as close as possible) the points to modify. This mode must be stopped by clicking anywhere on the map with the right button of the mouse. It is thus possible to zoom or un-zoom to click others points. When the GCP-CP configuration is satisfying, click on the button `Validate`.

### The TEST ME example

The TEST ME example is provided to understand quickly the functioning of our Bundle-Adjustment. The files are already stored in the `example-project-folder`.

## B.2 Tuning parameters

One of the advantage of our Bundle-adjustment is the wide range of parameters that can be tuned. These parameters are summarized in the table next page. However, the choice of these parameters is interdependent.

If the number of parameters to model the Interior Orientation is high (such as when the Brown model contains high degree polynomials or when INTERIOR-ORIENTATION-MODEL is ORTHO-POLY18), it is recommended to slow (i.e. to damp) the Interior Orientation parameters (by setting SLOW-INTERIOR-PARAM-NUM to 18 and SIGMA-SLOW-PARAM to 30 in the example of ORTHO-POLY18). Moreover, when the slowing method is used, the iterative process is slower, and thus, the number of iteration must be increased and the tolerance on the increment could be reduced (NB-MAX-IT could be increased to 10 000 and tol-conv could be reduced to  $1e-8$  in the example of ORTHO-POLY18).

If the CAMERA-PARAMETERS-ORIGIN is CONFIG-FILE, the camera parameters must be given in the param-struct. This param-struct.CAMERA-PARAMETERS contains the IO parameters as described in Table B.1.

	Name parameter	Variable type	description
Brown	CAMERA_PARAMETERS.f	float	Principal distance
	CAMERA_PARAMETERS.pp	2*1 vector	Principal point
	CAMERA_PARAMETERS.R123	n*1 vector	Where n = CALIBRATE_RADIALS
	CAMERA_PARAMETERS.T12	n*1 vector	Where n = CALIBRATE_TANGENTIALS
	CAMERA_PARAMETERS.B12	2*1 vector	Skew parameters
Ortho poly 18	CAMERA_PARAMETERS.ortho_a	9*1 vector	Elements of the orthogonal polynomial matrix for x
	CAMERA_PARAMETERS.ortho_b	9*1 vector	Elements of the orthogonal polynomial matrix for y
Ortho poly 15	CAMERA_PARAMETERS.a11	float	Elements of the orthogonal polynomial with constraints for the x and y coordinates
	CAMERA_PARAMETERS.a13	float	
	CAMERA_PARAMETERS.a21	float	
	CAMERA_PARAMETERS.a22	float	
	CAMERA_PARAMETERS.a23	float	
	CAMERA_PARAMETERS.a32	float	
	CAMERA_PARAMETERS.a33	float	
	CAMERA_PARAMETERS.b11	float	
	CAMERA_PARAMETERS.b12	float	
	CAMERA_PARAMETERS.b21	float	
	CAMERA_PARAMETERS.b22	float	
	CAMERA_PARAMETERS.b23	float	
	CAMERA_PARAMETERS.b31	float	
	CAMERA_PARAMETERS.b32	float	
CAMERA_PARAMETERS.b33	float		

Table B.1. Structure of camera Interior Orientation model

Name parameter	Variable type or Possible value	description
PROJECT_NAME	string	Name of the project
PROJECT_STORAGE_FOLDER	string	Folder where temp files and output are stored
NAME_output	string	Name of the output file
NAME_PHOTO_INPUT	Empty string	Put an empty string if the file has been already loaded
	string	Path of block exchange.xml
NAME_IMU_INPUT	string	Path to the .fmb trajectory file
TYPE_GNSS_AERIAL_CONTROL	'NONE'	No GNSS on the drone
	'CENTERED'	The lever-arm between GNSS phase center and camera perspective center is null.
	'KNOWN_LEVER_ARM'	The lever-arm between GNSS phase center and camera perspective center is known and equal to <code>lever_arm</code>
	'CALIB_LEVER_ARM'	The lever-arm between GNSS phase center and camera perspective center is unknown.
TYPE_IMU_AERIAL_CONTROL	'NONE'	No IMU on the drone
	'ABSOLUTE'	Absolute orientation with known <code>bore_sight</code> matrix between the IMU and the camera.
	'ABSOLUTE_CALIB'	Absolute orientation with calibration of the <code>bore_sight</code> matrix between the IMU and the camera.
	'RELATIVE'	Relative orientation between successive poses. The <code>bore_sight</code> matrix is not needed.
INTERIOR_ORIENTATION_MODEL	'BROWN'	Uses Brown camera model
	'ORTHO_POLY18'	Uses the full Orthogonal polynomials with 18 coefficients to model the camera
	'ORTHO_POLY15'	Uses the constrained Orthogonal polynomials with 15 coefficients to model the camera
INTERIOR_CALIBRATION	'FIX'	The camera IO parameters are known in advance and is fixed in the adjustment
	'LEADING'	The principal point and principal distances are adjusted (free) while the other parameters are fixed in the adjustment
	'FREE'	All the IO parameters are adjusted
CALIBRATE_F	0 or 1	If INTERIOR_CALIBRATION is FREE, 0 if the principal distance is fixed, 1 if it is free
CALIBRATE_PP	0 or 2	If INTERIOR_CALIBRATION is FREE, 0 if the principal point is fixed, 2 if it is free
CALIBRATE_RADIALS	positive integer	If INTERIOR_CALIBRATION is FREE, number of radial coefficients
CALIBRATE_TANGENTIALS	0 or a positive integer > 1	If INTERIOR_CALIBRATION is FREE, 0 if the tangential distortions are fixed, 2 or more if it is free (corresponds to the number of tangential distortions parameters)
CALIBRATE_SKEW	0 or 2	If INTERIOR_CALIBRATION is FREE, 0 if the skew parameters is fixed, 2 if it is free

DIRECT_OBSERVATION	Column Vector of observations	Direct observation of the IO parameters. To add these measurements, INTERIOR_CALIBRATION must be FREE, DIRECT_OBSERVATION_NUM must be the length of the DIRECT_OBSERVATION vector and of the DIRECT_OBSERVATION_SIG
DIRECT_OBSERVATION_NUM	positive integer	Length of the DIRECT_OBSERVATION
DIRECT_OBSERVATION_COV	false	The covariance matrix of the DIRECT_OBSERVATION vector is diagonal, and the values of this diagonal are given in the vector DIRECT_OBSERVATION_SIG
	true	The covariance matrix of the DIRECT_OBSERVATION vector is full, and the values of this squared matrix are given in the vector DIRECT_OBSERVATION_SIG
DIRECT_OBSERVATION_SIG	Column Vector of sigma of direct observations	If DIRECT_OBSERVATION_COV is false
	Square covariance matrix of direct observations	If DIRECT_OBSERVATION_COV is true
SLOW_INTERIOR_PARAM_NUM	positive integer	Number of IO parameters to slow
SIGMA_SLOW_PARAM	Real number	Sigma of the pseudo-observation to slow
CAMERA_PARAMETERS_ORIGIN	'INPUT_FILE'	IO are taken in the .xml file
	'CONFIG_FILE'	IO are given in the CAMERA_PARAMETERS
	'OTHER_FILE'	IO are taken in file whose path is given by CAMERA_PARAMETERS_ORIGIN
CAMERA_PARAMETERS	Structure containing the camera parameters (Table B.1)	
CAMERA_PARAMETERS_ORIGIN	string	Path to the camera parameters
NB_MAX_IT	positive integer	Maximum number of iterations
tol_conv	float	Tolerance on the maximum position displacement of the tie-points
lever_arm	3*1 float vector	Lever-arm between the GNSS phase center and the camera perspective center
bore_sight	3*3 float rotation matrix	Boresight matrix between the IMU and the camera
sig_pix	float	Sigma of a tie-point image observation
sig_pix_GCP	float	Sigma of a CP or a GCP image observation
sig_inertia	float	If TYPE_IMU_AERIAL_CONTROL is ABSOLUTE_CALIB or ABSOLUTE_CALIB Sigma of direct observation in rad
	float	If TYPE_IMU_AERIAL_CONTROL is RELATIVE Sigma of direct observation in rad/ $\sqrt{s}$
sig_GCP	3*1 float vector	Sigma of the GCPs (X,Y,Z)
load_GCPs	false or true	If true, load the GCPs in list_GCPs
list_GCPs	string	Path to the list of GCPs to import
GCP_GUI	false or true	If true, shows the GUI for GCPs selection

### B.3 Output and visualisation

#### B.3.1 Convergence rate

The first visible output is the convergence of the maximum value of  $\delta x$  as a function of the iteration. If the adjustment converges properly, the  $\max(\delta x)$  globally decreases until it reaches the tolerance.

#### B.3.2 Covariances matrices

The total number of observation is given by `nb-obs` and the total number of parameters is given by `u`. Thus, the redundancy is given by `nb-obs - u`.

The precision of an observation of unitary weight  $\sigma_0$  could be computed as `sigma-0 = sqrt(v'*P*v/(nb-obs - u))`.

The inversion of  $A^T P A$  is notoriously time and memory consuming. Thus, it is important to inverse only part of this matrix as described in section 1.2.10.

`compute-block-inverse( N , interval-row , interval-column )` compute subset of the inverse of the matrix `N`. `interval-row` and `interval-column` are respectively the set of integers of the rows and the set of integers of the columns.

To extract the proper elements of the inverse of  $A^T P A$ , it is necessary to refer the columns of  $A$  properly as described in Figure B.5. In particular, `string-param-calib` contains the IO parameters name.

#### B.3.3 GCP residuals

The `residus-GCPs` variable contains the GCP residuals in the same order as the `list-GCP`.

### B.4 Code and Algorithm description

The goal of this section is to explain the functioning of Topo-Bundle and what is the structure of the code in order to help permit potential developer modifying the code.

The adjustment algorithm uses Gauss-Newton algorithm (it could be damped for some variables if needed). This Bundle-Adjustment algorithm needs to build the Design matrix as the Jacobian of the observation equations.

The philosophy of the topo-Bundle code structure is to show the computations that are



meaningful for surveyors/photogrammetrists, and to hide the technical informatics subtleties.

### B.4.1 Important variables

The structure variable `poses` describes the poses parameters of the camera while a photo is taken. For example, `poses(5)` describes the poses parameters while the 5<sup>th</sup> photo was taken. `poses(5).P` is the position of the camera perspective center, `poses(5).R` is the rotation matrix from local frame to camera frame for the 5<sup>th</sup> photo. These two values are adjusted parameters in the Bundle-Adjustment (i.e. they are updated at each step). `poses(5).Tin` and `poses(5).Rin` are the corresponding values given by the inputted trajectory from GNSS/IMU fusion. They are considered in the Bundle-Adjustment as observations (thus, they are constant).

`obs` are the 2D image observations. The first column describes in which image the point is seen. The second column describes which point is seen, and the two last one are the images observation (The unit is usually the `pix` and must be coherent with the one describing the `ID`). This `obs` is linked to the variable `terrain` in which all the points of the terrain are described (Figure B.4). The first column is the unique identifier of the point, and the three other columns are the points coordinates. This `terrain` variable is an adjusted parameter in the Bundle-Adjustment (i.e. it is updated at each step).

### B.4.2 Update step

The orientation of the camera (`poses(i).R`) is updated in the Bundle-Adjustment with the right multiplication as in the right column of Table 1.4. The boresight-matrix is updated in the Bundle-Adjustment with the left multiplication as in the left column of Table 1.4. This choice is justified by the input of absolute and relative orientation. The derivatives of Table 1.4 and Table 1.5 will be used for the derivation of the observation equations.

### B.4.3 Observation models and derivations

#### Mathematic Vs Informatic notations

For the scope of conciseness, the notations (Table B.2) in this appendix are slightly different from the one used in the code.

## Appendix B. Bundle-Adjustment Algorithm: Code description

	Notation	In the matlab code
Camera perspective center	$T$	<code>poses(i).T</code>
Camera orientation	$R$	<code>poses(i).R</code>
$\partial$ Camera orientation	$\omega$	<code>dx(jposes+(i*6-2:i*6))</code>
Tie-point position	$P$	<code>terrain(j,2:4)</code>
Camera position from trajectory	$\tilde{T}$	<code>poses(i).Pin</code>
Camera orientation from trajectory	$\tilde{R}$	<code>poses(i).Rin</code>
GNSS antenna Lever arm	$T_{la}$	<code>lever-arm</code>
camera-IMU Boresight matrix	$R_{bs}$	<code>bore-sight</code>
Projection function	$\pi$	<code>pi-</code>
Projection function Jacobian	$\Pi$	<code>PI</code>
IO function jacobian	$\Xi$	<code>XI</code>

Table B.2. Notations

**Image observations**

The 2D image observation  $\ell_{im}$  is simulated as follows, where  $\pi$  is the  $\mathbb{R}^3 \rightarrow \mathbb{R}^2$  perspective projection function (1.2) and  $\xi$  models the camera IO (As described in 1.1.6 and discussed in Chapter 2)

$$\ell_{im} = \xi(\pi(R^T(P - T))) \tag{B.1}$$

$\Xi$  is the jacobian matrix of the function  $\xi$ , and  $\Pi$  is the jacobian matrix of the function  $\pi$ .

$$\Xi = \frac{\partial \xi(p)}{\partial p} \tag{B.2}$$

$$\Pi = \frac{\partial \pi(P')}{\partial P'} \tag{B.3}$$

The computation of the derivatives with respect to elements in an euclidian space is straightforward (Equation B.4 and B.5), while the one with respect to a  $\mathbb{S}\mathbb{O}_3$  rotation uses Table 1.4 (Equation B.6).

$$\frac{\partial \ell}{\partial P} = \Xi \Pi R^T \tag{B.4}$$

$$\frac{\partial \ell}{\partial T} = -\Xi \Pi R^T \tag{B.5}$$

$$\frac{\partial \ell}{\partial \omega} = \Xi \Pi [R^T(P - T)]_x \tag{B.6}$$

## Appendix B. Bundle-Adjustment Algorithm: Code description

---

### GCP observations

The GCP observation model is trivial since the GCP coordinates are directly measured.

$$\ell_{GCP} = P_{GCP} \quad (\text{B.7})$$

$$\frac{\partial \ell_{GCP}}{\partial P_{GCP}} = I_3 \quad (\text{B.8})$$

### Embedded GNSS observations

The GNSS antenna embedded on the platform measures the position of the antenna phase center which differs from the position of the camera perspective center by the lever-arm  $T_{la}$  (expressed in the camera frame).

$$\ell_{GNSS} = T + R T_{la} \quad (\text{B.9})$$

$$\frac{\partial \ell_{GNSS}}{\partial T} = I_3 \quad (\text{B.10})$$

$$\frac{\partial \ell_{GNSS}}{\partial \omega} = -R [T_{la}]_{\times} \quad (\text{B.11})$$

$$\frac{\partial \ell_{GNSS}}{\partial T_{la}} = R \quad (\text{B.12})$$

### Absolute orientation observations

Absolute or Relative orientation input observations belongs to  $\mathbb{S}\mathbb{O}_3$ . Section 1.5.6 gives the theoretical background for its input in the Bundle-Adjustment.

## B.4. Code and Algorithm description

---

A  $\mathbb{S}\mathbb{O}_3$  element cannot be concatenated inside an observation vector. This is why, neither the observations vector  $\ell$  nor the  $\overset{\circ}{\ell}$  is explicitly built. However, the misclosure of each observation equation is explicitly built as equation B.13 for absolute orientation and B.16 for relative orientation. The derivatives of these misclosures will be computed with respect to the increment that will be used for the update-step of the parameters.

$$\overset{\circ}{v}_{\tilde{R}} = \log(R^T \tilde{R} R_{bs}^T) \quad (\text{B.13})$$

The derivatives are computed with the help of Table 1.5.

$$\frac{\partial \overset{\circ}{v}_{\tilde{R}}}{\partial \omega} = I_3 \quad (\text{B.14})$$

$$\frac{\partial \overset{\circ}{v}_{\tilde{R}}}{\partial \omega_{bs}} = I_3 \quad (\text{B.15})$$

### Relative orientation observations

The misclosure for the equation of relative orientation observations is given below, with their derivatives used in the Bundle Adjustment.

$$\overset{\circ}{v}_{\Delta\tilde{R}} = \log(R_1^T \tilde{R}_1 \tilde{R}_2^T R_2) \quad (\text{B.16})$$

$$\frac{\partial \overset{\circ}{v}_{\Delta\tilde{R}}}{\partial \omega_1} = I_3 \quad (\text{B.17})$$

$$\frac{\partial \overset{\circ}{v}_{\Delta\tilde{R}}}{\partial \omega_2} = -I_3 \quad (\text{B.18})$$

## Appendix B. Bundle-Adjustment Algorithm: Code description

---

### Direct observations

The direct observations consist in giving the values of the IO given by a previous calibration (and optionally of the lever-arm and of the boresight-matrix) as observations  $\ell_{\Theta}$  weighted by their uncertainty:  $\Sigma_{\Theta}^{-1}$ . This process is explained and discussed in 2.4.

The observation equation between the IO parameters  $\Theta$  and there direct observation is thus an equality, whose jacobian matrix is the identity.

$$\ell_{\Theta} = \Theta \tag{B.19}$$

If the weight matrix  $\Sigma_{\Theta}^{-1}$  is diagonal, the values are concatenated in the vector  $p$  containing all the weights of the Bundle-Adjustment. This  $p$  vector is then transformed in a diagonal matrix  $P$ . If the weight matrix  $\Sigma_{\Theta}^{-1}$  is full, the  $P$  matrix is built with the others observations, and then, the  $\Sigma_{\Theta}^{-1}$  are added to create a block-diagonal matrix.

### Damping pseudo-observations

The Damping pseudo-observations permit to Damp some of the parameters (it could be considered as a modified version of the Levenberg-Maquard algorithm where the damping parameter is given by the weight of the Damping pseudo-observations for the observations that are damped, and is 0 for the others).

We choose to Damp these observations by putting a Identity matrix as the derivative, and to give null values to the  $\dot{v}$  values associated to these damped observations.

#### B.4.4 Management of the position of the sub-matrix-block

The derivatives computed in the previous section needs to be appropriately placed in the  $A$  matrix.

The index for the row (on the left side of Figure B.5) and for the columns (on the top side of Figure B.5) corresponds to the division between respectively the observation type, and the parameter type. These integer values correspond to the last element before the considered block.

For example, we know that a GCP observation contains 3 element. The column of the derivative of the first GCP will be:

## B.4. Code and Algorithm description

$i\_GCP + [1\ 2\ 3]$

The column of the derivative of the second GCP will be:

$i\_GCP + [4\ 5\ 6]$

and so on.

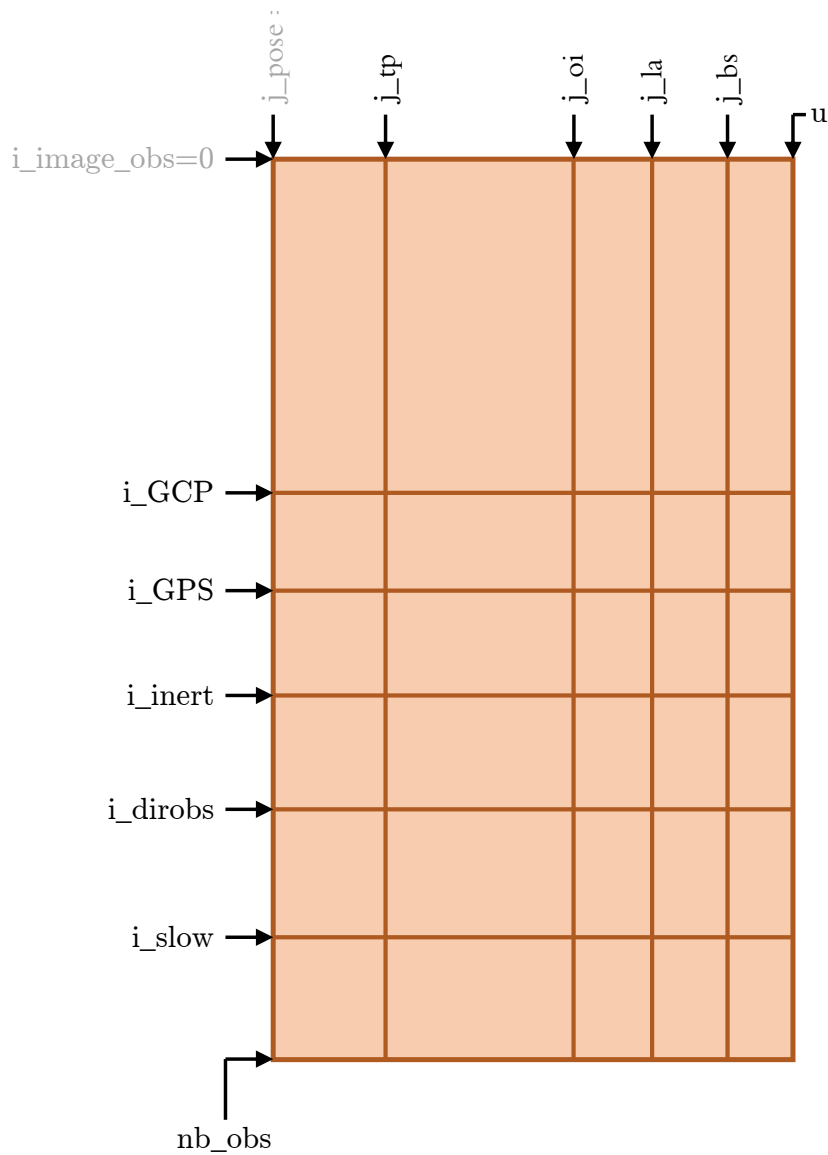


Figure B.5. Matrix A structure

### B.4.5 Management of the construction of the sparse matrix

The design matrix A is sparse. Thus, it should be build in a sparse way to improve performances. However, the construction and the modification of a sparse matrix is non-trivial. Therefore, the construction of the Design matrix A is hidden in an object structure.

This object is constructed as follows.

```
A = Sparse_A_matrix( number_derivative );
```

where `number-derivative` is the number of single values derivatives i.e. the number of scalar numbers that will be inserted in the sparse matrix A.

Then, the matrix is fill with blocks as follow.

Figure and Code B.6 illustrates the insertion a sub matrix at the rows 137, 138 and at the columns 42, 43, 44, 45.



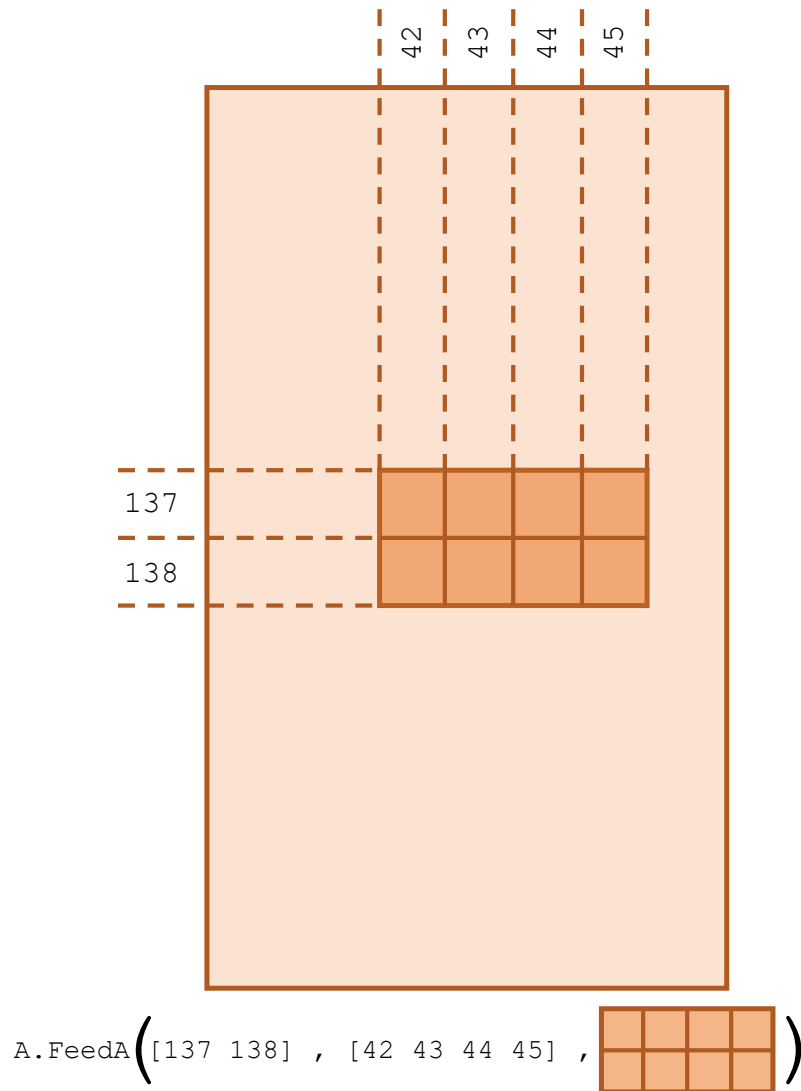


Figure B.6. Adding sub-matrix in matrix A

If the block-sub-matrix to insert is diagonal, the insertion could be done as on the Figure and code B.7, where the  $3 \times 3$  diagonal matrix is inserted at the rows 92, 93, 94 and at the columns 51, 52, 53.

## Appendix B. Bundle-Adjustment Algorithm: Code description

---

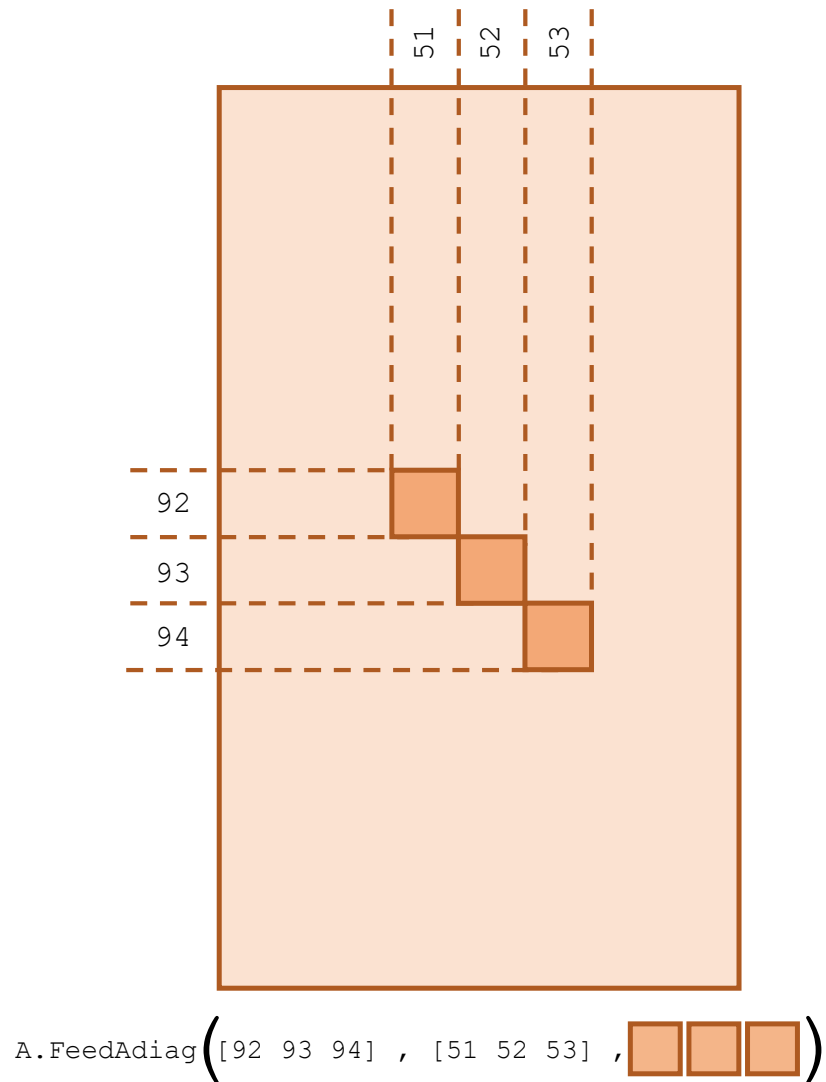


Figure B.7. Adding sub-diagonal-matrix in matrix A

Before being used in algebraic computation, the A sparse matrix object must be converted in actual matrix as follows.

```
A = A.return_matrix;
```

The produced A matrix could be visualised with the command `spy(A)`.

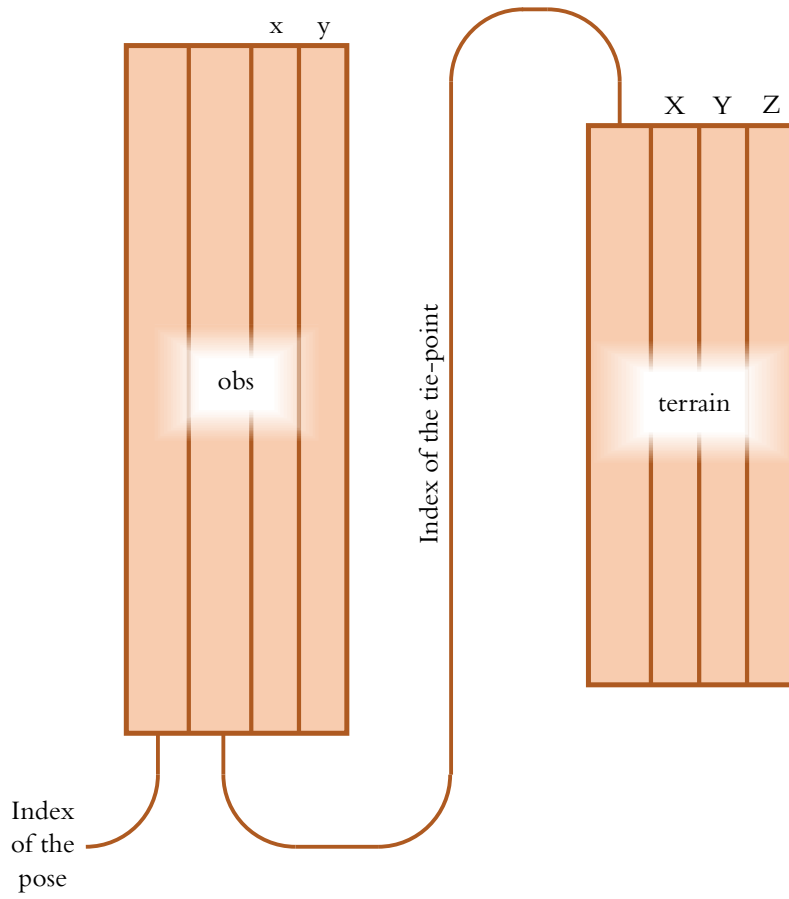


Figure B.4. Link between obs and terrain variable



## Photographic credits

The ortho-photo, the DTM and the snow pack thickness of Figure 1 is originated from [140].

The point-cloud and the Point-cloud section of Figure 1 have been created from the data provided by Julien Vallet.

Table 1 is a compilation of images from various sources.

- The hand laser-scanner is a ZG 3d laser scanner.
- The terrestrial laser scanner is a Leica geosystem laser-scanner <https://leica-geosystems.com/products/laser-scanners/scanners>
- The drone airborne Laser-scanner is developed by Northern Robotics Laboratory, Ulaval: <https://norlab.ulaval.ca/publications/survey-lidar-UAV/>
- The Helicopter airborne Laser-scanner is the Helimap system: <http://www.helimap.ch/>
- The plane airborne Laser-scanner is an image from the following article <https://www.auav.com.au/articles/drone-data-vs-lidar/>
- The laser-scanner embedded on a Satellite orbiting around Mars Planet is the Mars Orbiter Laser Altimeter: Mars Global Surveyor Missions by NASA <https://mars.nasa.gov/mars-exploration/missions/mars-global-surveyor>
- The macro photogrammetry is the scan of the Maori head of the museum of Rouen, described in [162]
- The terrestrial photogrammetry is an image of the *Arco di Constantino in Rome, Italy* created with the software and the data provided by Niclas Börlin, described in [17].
- The drone for airborne photogrammetry is an eBee+ [142].

## **Appendix B. Bundle-Adjustment Algorithm: Code description**

---

- The explanation of the plane airborne photogrammetry was found in [112] but the author of the original drawing is unknown to us.

Figure 3 is a compilation of several images including the terrestrial and the aerial vehicle of the mapKITE project <http://www.mapkite.com>, and the lower drone of the DoDo project [75].

# Bibliography

- [1] F. Abad, E. Camahort, and R. Vivó. Camera calibration using two concentric circles. In *International Conference Image Analysis and Recognition*, pages 688–696. Springer, 2004.
- [2] R. Achanta, P. Marques Neila, P. Fua, and S. Süsstrunk. Scale-adaptive superpixels. *26th Color and Imaging Conference Final Program and Proceedings*, pages 1–6(6), 2018.
- [3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. *EPFL Technical Report 149300*, 2010.
- [4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):8. 2274–2282, 2012. A previous version of this article was published as a EPFL Technical Report in 2010: <http://infoscience.epfl.ch/record/149300>. Supplementary material can be found at: <http://ivrg.epfl.ch/research/superpixels>.
- [5] N. Aimaiti. A Comparison of Rotation Parameterisations for Bundle Adjustment. Master’s thesis, Umeå University, Department of Computing Science, 2015.
- [6] J. Albrecht and M. Wiggenghagen. *Taschenbuch zur Photogrammetrie und Fernerkundung: Guide for Photogrammetry and Remote Sensing*. Wichmann, Heidelberg, 5. Auflage, 2008.
- [7] Applanix. POSPac MMS documentation. <https://www.applanix.com/products/pospac-mms.htm>, 2016. [Online].
- [8] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, Aug. 2006.
- [9] G. S. C. Avellar, G. A. S. Pereira, L. C. A. Pimenta, and P. Iscold. Multi-UAV Routing for Area Coverage and Remote Sensing with Minimum Time. *Sensors*, 15(11):27783–27803, Nov. 2015.

## Bibliography

---

- [10] W. Baarda. Criteria for precision of geodetic networks. In *Archives 13th International Congress of Surveyors*, number paper 503.1, 1971.
- [11] E. Barney. Tacoma Narrows Bridge Collapse, video by british pathe gazelle. <https://www.youtube.com/watch?v=XggxeuFDaDU>, 1940.
- [12] F. A. Bayoud. *Development of a robotic mobile mapping system by vision-aided inertial navigation a geomatics approach*. PhD thesis, EPFL, 2006.
- [13] A. Bjerhammar. *Theory of errors and generalized matrix inverses*. Elsevier, Amsterdam, 1973.
- [14] M. Blázquez. A new approach to spatio-temporal calibration of multi-sensor systems. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B1. Beijing*, 2008.
- [15] M. Blázquez and I. Colomina. On the role of self-calibration functions in integrated sensor orientation. In *Proceedings of the International Calibration and Orientation Workshop (EuroCOW). Castelldefels, Spain*, 2010.
- [16] M. Blázquez and I. Colomina. Relative INS/GNSS aerial control in integrated sensor orientation: models and performance. *ISPRS Journal of Photogrammetric Engineering and Remote Sensing*, 67:120–133, 2012.
- [17] N. Börlin and P. Grussenmeyer. Bundle adjustment with and without damping. *Photogrammetric Record*, 28(144):396–415, 2013.
- [18] N. Börlin, A. Murtyoso, P. Grussenmeyer, F. Menna, and E. Nocerino. Modular bundle adjustment for photogrammetric computations. *ISPRS Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2:133–140, 2018.
- [19] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen. *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*. Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2007.
- [20] S. Botton, F. Duquenne, Y. Egels, M. Even, and P. Willis. *GPS. Localisation et navigation*. Hermes Science Publications, Paris, Mar. 1998.
- [21] J.-Y. Bouguet. Matlab camera calibration toolbox. *Caltech Technical Report*, 2000.
- [22] D. C. Brown. An advanced plate reduction for photogrammetric cameras. *Air Force Cambridge Research Laboratories*, Report No. 64-40, 1964.
- [23] D. C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, Vol. XXXII, No. 3., 1966.



- [24] D. C. Brown. Close-range camera calibration. *Symposium on Close-Range Photogrammetry, Urbana, Illinois, January*, 1971.
- [25] F. Buffa, A. Pinna, and G. Sanna. *A Simulation Tool Assisting The Design Of A Close Range Photogrammetry System For The Sardinia Radio Telescope*, volume ISPRS annal III-5. June 2016.
- [26] L. M. B. d. C. Campos. *Generalized Calculus with Applications to Matter and Forces*. CRC Press, Apr. 2014. Google-Books-ID: bMiSAwAAQBAJ.
- [27] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.
- [28] P. Cattin. Trinet+ un logiciel d’ajustement tridimensionnel d’observations terrestres et satellitaires. In *Bulletin d’informations des Ingénieurs Géomètres de Suisse Occidentale*, 15, 2007.
- [29] A. Chatterjee and V. Madhav Govindu. Efficient and robust large-scale rotation averaging. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.
- [30] T. Chen and R. S. Shibasaki. Determination of camera’s orientation parameters based on line features. *International Archives of Photogrammetry and Remote Sensing*, XXXII/5:23–28, 1998.
- [31] P. Clausen. *Calibration Aspects of INS Navigation*. PhD thesis, EPFL, Lausanne, 2019.
- [32] P. Clausen and J. Skaloud. On the calibration aspects of MEMS-IMUs used in micro uavs for sensor orientation. In *IEEE-ION Position Location and Navigation Symposium (PLANS)*, Portland, OR, USA, 2020.
- [33] E. Cledat. Conception d’une méthode de consolidation de grands réseaux lasergrammétriques. Master thesis, Institut National des Sciences Appliquées de Strasbourg - Electricité de France, 2015.
- [34] E. Cledat. Couplage fort de données photogrammétriques et lasergrammétriques dans un ajustement en bloc. In *WorkShop LIDAR*, Lyon, France, Mar 2019.
- [35] E. Cledat, D. Cucci, and J. Skaloud. Planning of UAV mission for precise terrain reconstruction in steep terrain without ground control points. Swiss Geoscience Meeting, Davos, Nov. 2017.
- [36] E. Cledat and D. A. Cucci. Mapping GNSS restricted environments with a drone tandem and indirect position control. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W3:1–7, 2017.

## Bibliography

---

- [37] E. Cledat and M. Gebhard. Prédiction de la précision d'un relevé photogrammétrique aéroporté par drone - die genauigkeitsabschätzung einer drohnen-gestützten photogrammetrischen aufnahme. Geosummit, Bern, June 2018.
- [38] E. Cledat, L. V. Jospin, A. S. Dubois, A. Weltermann, D. A. Cucci, and J. Skaloud. Navigation de précision pour l'intervention de drones dans des zones difficiles. Geosummit, Bern, June 2018.
- [39] I. Colomina and P. Molina. Unmanned aerial systems for photogrammetry and remote sensing: a review. *ISPRS Journal of Photogrammetric Engineering and Remote Sensing*, 92(6):79–97, 2014.
- [40] M. Cramer, N. Haala, D. Laupheimer, G. Mandlbürger, and P. Havel. Ultra-high precision UAV-based lidar and dense image matching. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1:115–120, 2018.
- [41] M. Cramer, H. Przybilla, and A. Zurhorst. UAV cameras: overview and geometric calibration benchmark. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:85, 2017.
- [42] D. A. Cucci. Accurate optical target pose determination for applications in aerial photogrammetry. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:257–262, 2016.
- [43] D. A. Cucci, M. Rehak, and J. Skaloud. Bundle adjustment with raw inertial observations in UAV applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:1–12, Aug. 2017.
- [44] M. Daakir. *Centimetric absolute localization using Unmanned Aerial Vehicles with airborne photogrammetry and on-board GPS*. phd thesis, Université Paris-Est, Dec. 2017.
- [45] E. Eade. Lie Groups for 2d and 3d Transformations. 2017. [http://ethaneade.com/lie\\_groups.pdf](http://ethaneade.com/lie_groups.pdf).
- [46] H. Ebner. Self calibrating block adjustment. *Bildmessung und Luftbildwesen*, 44:128–139, 1976.
- [47] Y. Egels and E. Laroze. La topo au Disto, bon marché, tient dans la poche. *XYZ 149 pp 49-52, ISSN: 0290-9057*, 2016.
- [48] C. Eling, L. Klingbeil, M. Wieland, and H. Kuhlmann. Direct georeferencing of micro aerial vehicles system design, system calibration and first evaluation tests. *Photogrammetrie-Fernerkundung-Geoinformation*, 2014(4):227–237, 2014.

- [49] A. W. Fitzgibbon, R. B. Fisher, et al. A buyer's guide to conic fitting. *DAI Research paper*, 1996.
- [50] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [51] W. Förstner and B. P. Wrobel. *Photogrammetric Computer Vision – Statistics, Geometry, and Reconstruction*. Springer, 2015.
- [52] C. S. Fraser. Digital camera self-calibration. *ISPRS Journal of Photogrammetry and Remote sensing*, 52(4):149–159, 1997.
- [53] P. Fua. Object delineation as an optimization problem, a connection machine implementation. *Artificial Intelligence Center, SRI International*, 1989.
- [54] P. Fua and A. J. Hanson. Using generic geometric models for intelligent shape extraction. *American Association for Artificial Intelligence*, 1987.
- [55] J. Gallier. Basics of Manifolds and Classical Lie Groups: The Exponential Map, Lie Groups, and Lie Algebras. In *Geometric Methods and Applications: For Computer Science and Engineering*, Texts in Applied Mathematics. Springer-Verlag, New York, 2 edition, 2011.
- [56] F. Gandor, M. Rehak, and J. Skaloud. Photogrammetric Mission Planner for RPAS. *ISPRS - The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(1):61, 2015.
- [57] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.
- [58] P. Glira, N. Pfeifer, C. Briese, and C. Ressel. A correspondence framework for ALS strip adjustments based on variants of the ICP algorithm. *Photogrammetrie - Fernerkundung - Geoinformation*, 2015, 08 2015.
- [59] P. Glira, N. Pfeifer, C. Briese, and C. Ressel. Rigorous strip adjustment of airborne laser-scanning data based on the ICP algorithm. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-3/W5*, 09 2015.
- [60] P. Glira, N. Pfeifer, and G. Mandlbürger. Rigorous strip adjustment of UAV-based laser-scanning data including time-dependent correction of trajectory errors. *Photogrammetric Engineering and Remote Sensing*, 82:945–954, 12 2016.

## Bibliography

---

- [61] P. Glira, N. Pfeifer, and G. Mandlburger. Hybrid orientation of airborne lidar point clouds and aerial images. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:567–574, May 2019.
- [62] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The Perfect Match: 3D Point Cloud Matching with Smoothed Densities. *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019) Pages 5540 - 5549, 32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019), Long Beach, CA, USA, June 16-20, 2019*.
- [63] B. Grocholsky and N. Michael. Exploiting mobility heterogeneity in micro-aerial vehicle deployments for environment exploration. In *IEEE Globecom Workshops*, pages 1421–1425, Dec. 2013.
- [64] P. Grussenmeyer and O. A. Khalil. Solutions for Exterior Orientation in Photogrammetry: A Review. *The Photogrammetric Record*, 17(100):615–634, 2002.
- [65] A. Grün. Photogrammetrische punktbestimmung mit der bündelmethode. *Mitteilungen, Vol. 40, Institut für Geodäsie und Photogrammetrie, ETHZ, Zürich, Switzerland*, 1986.
- [66] GSA. Almanac. European Global Navigation Satellite System Agency, European GNSS service center <https://www.gsc-europa.eu/product-almanacs>, 2020. [Online; accessed 20-March-2020].
- [67] G. Guidi, J.-A. Beraldin, and C. Atzeni. High-accuracy 3D modeling of cultural heritage: the digitizing of Donatello's "Maddalena". *IEEE Transactions on Image Processing*, 13(3):370–380, Mar. 2004.
- [68] R. Hartley, J. Trunpf, Y. Dai, and H. Li. Rotation Averaging. *International Journal of Computer Vision*, 103(3):267–305, July 2013.
- [69] M. Hebel and U. Stilla. Simultaneous calibration of ALS systems and alignment of multiview lidar scans of urban areas. *IEEE Transactions on Geoscience and Remote Sensing*, 11 2011.
- [70] R. d. Hollander and F. Favory. *Sciences géographiques dans l'Antiquité: Connaissance du monde : Conception de l'univers*. Association Française de Topographie, 2002. ISBN: 978-2-901264-53-8.
- [71] J.-F. Hullo. Fine Registration of kilo-Station Networks - A Modern Procedure for Terrestrial Laser Scanning Data sets. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLI-B5, pages 485–492. Copernicus GmbH, June 2016.

- 
- [72] Ixblue. Airins Datasheet. [https://www.ixblue.com/sites/default/files/2019-11/Airins\\_DS.pdf](https://www.ixblue.com/sites/default/files/2019-11/Airins_DS.pdf), June 2018. (Accessed on 02/2020).
- [73] M. Jayendra-Lakshman and V. Devarajan. A new feature descriptor for LIDAR image matching. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-2/W1, 09 2013.
- [74] G. Jiang and L. Quan. Detection of concentric circles for camera calibration. In *Computer Vision, ICCV, Tenth IEEE International Conference on*, volume 1, pages 333–340. IEEE, 2005.
- [75] L. Jospin, A. Stoven-Dubois, and D. Cucci. Photometric long-range positioning of LED targets for cooperative navigation in UAVs. *Drones*, 3:69, 08 2019.
- [76] M. Kalantari and M. Kasser. Photogramm trie et vision par ordinateur. *Revue XYZ Nu 117*, pages 49–54, 2008.
- [77] M. Kasser. *Le GPS : utilisation en positionnement et surveillance*. Ed. Techniques Ing nieur, 2012. Google-Books-ID: PdzKNjrsnUcC.
- [78] M. Kasser and Y. Egels. *Photogram trie num rique*. Collection ENSG IGN, Paris, Herm s sciences publications, 2001.
- [79] A. P. Kersting, A. F. Habib, K.-I. Bang, and J. Skaloud. Automated approach for rigorous light detection and ranging system calibration without preprocessing and strict terrain coverage requirements. *Optical Engineering*, 51(7):19. 076201–1–19, 2012.
- [80] J.-S. Kim, P. Gurdjos, and I.-S. Kweon. Geometric and algebraic constraints of projected concentric circles and their applications to camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):637–642, April 2005.
- [81] T. Kluter. Gecko4nav technical reference manual. *Revision 1.0. Internal Document*, 2012.
- [82] T. Krarup. «*Mechanics of adjustment*», *Book chapter in Mathematical Foundation of Geodesy: selected papers of Torben Krarup*, pp. 333 – 339. Springer Science & Business Media, Sept. 1982. Google-Books-ID: ZPIM5F3hbZQC.
- [83] R. Kumar Mishra. A Review of Optical Imagery and Airborne LiDAR Data Registration Methods. *The Open Remote Sensing Journal*, 5(1):54–63, July 2012.
- [84] R. K mmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613, May 2011.

## Bibliography

---

- [85] O. Küng, C. Strecha, P. Fua, D. Gurdan, M. Achtelik, K.-M. Doth, and J. Stumpf. Simplified building models extraction from ultra-light uav imagery. *UAV-g 2011 - Unmanned Aerial Vehicle in Geomatics, Zürich, CH, September 14-16, 2011*.
- [86] T. Läbe and W. Förstner. Geometric stability of low-cost digital consumer cameras. In *Proceedings of the 20th ISPRS Congress, Istanbul, Turkey*, pages 528–535. Citeseer, 2004.
- [87] Lakeside Labs. Flying High - Autonomous Multi-UAV System For Wide Area Coverage. <https://www.youtube.com/watch?v=SvL8rTUNh1c>, Aug. 2013. [Online].
- [88] V. Lepetit and P. Fua. Keypoint recognition using random forests and random ferns. *Decision Forests for Computer Vision and Medical Image Analysis*, (Chapter 9):111–124, 2013.
- [89] V. Lepetit, F. Moreno-Noguer, and P. Fua. EpnP: An accurate  $o(n)$  solution to the PnP problem. *International Journal Of Computer Vision*, 81:155–166, 2009.
- [90] J. Levallois. *MESURER LA TERRE. 300 ans de géodésie française, De la toise du Châtelet au satellite*. Presses de l'école nationale des Ponts et Chaussées, Association Française de Topographie, 1988.
- [91] D. D. Lichti, G. B. Sharma, G. Kuntze, B. Mund, J. E. Beveridge, and J. L. Ronsky. Rigorous Geometric Self-Calibrating Bundle Adjustment for a Dual Fluoroscopic Imaging System. *IEEE Transactions on Medical Imaging*, 34(2):589–598, Feb. 2015.
- [92] D. D. Lichti, J. Skaloud, and P. Schaer. On the calibration strategy of medium format cameras for direct georeferencing. In *International Calibration and Orientation Workshop EuroCOW*, number POST\_TALK, 2008.
- [93] M. Lösler. Javagraticule3d: Least squares adjustment: Principal component analysis [Dokumentation]. <http://wiki.derletztekick.com/javagraticule3d/least-squares-adjustment/principal-component-analysis>, 2014.
- [94] T. Luhmann. 3d Imaging - How to Achieve Highest Accuracy. *Proc. SPIE 8085, Videometrics, Range Imaging, and Applications XI, 808502*, 8085, June 2011. DOI: 10.1117/12.892070.
- [95] T. Lupton and S. Sukkarieh. Efficient integration of inertial observations into visual SLAM without initialization. In *Intelligent Robots and Systems (IROS), 2009 IEEE/RSJ International Conference on*, pages 1547–1552. IEEE, 2009.
- [96] K. Madsen, H. Nielsen, and O. Tingleff. *Methods for non-linear least squares problems*, 2nd Edition. Informatics and Mathematical Modelling - Technical University of Denmark, 2004.

- [97] G. Mandlbürger, K. Wenzel, A. Spitzer, N. Haala, P. Glira, and N. Pfeifer. Improved topographic models via concurrent airborne LIDAR and dense image matching. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W4:259–266, 09 2017.
- [98] A. Marjovi, J. Nunes, P. Sousa, R. Faria, and L. Marques. An olfactory-based robot swarm navigation method. In *IEEE International Conference on Robotics and Automation*, pages 4958–4963, May 2010.
- [99] O. Martin, C. Meynard, M. Pierrot Deseilligny, J.-P. Souchon, and C. Thom. Réalisation d’une caméra photogrammétrique ultralégère et de haute résolution. In *Proceedings of the colloque drones et moyens légers aéroportés d’observation, Montpellier, France*, pages 24–26, 2014.
- [100] F. J. Massey. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association.*, Vol. 46(No. 253):68–78, 1951.
- [101] T. Matsuda, T. Maki, Y. Sato, and T. Sakamaki. Performance verification of the alternating landmark navigation by multiple AUVs through sea experiments. In *OCEANS 2015 - Genova*, pages 1–9, May 2015.
- [102] Mavinci. Sirius Pro documentation. <http://www.mavinci.com>, 2016.
- [103] P. Meissl. *A priori prediction of roundoff error accumulation in the solution of a super-large geodetic normal equation system*. U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, 1980.
- [104] B. Merminod. *Méthodes d’estimation*. Course book, École polytechnique fédérale de Lausanne EPFL, Faculté de l’environnement naturel, architectural et construit, Laboratoire de topométrie, 2019.
- [105] O. Mian, J. Lutes, G. Lipa, J. Hutton, E. Gavelle, and S. Borghini. Direct georeferencing on small unmanned aerial platforms for improved reliability and accuracy of mapping without the need for ground control points. *ISPRS - The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(1):397, 2015.
- [106] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2681–2684, Nov. 2012.
- [107] M. Moakher. Means and Averaging in the Group of Rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, Jan. 2002.

## Bibliography

---

- [108] P. Molina, M. Blázquez, D. A. Cucci, and I. Colomina. First results of a tandem terrestrial-unmanned aerial mapkite system with kinematic ground control points for corridor mapping. *Remote Sensing*, 9(1):60, 2017.
- [109] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative  $O(n)$  solution to the pnp problem. *IEEE 11Th International Conference On Computer Vision, Vols 1-6*, pages 2252–2259, 2007.
- [110] P. Moulon. *Positionnement robuste et précis de réseaux d'images*. Phd thesis, Université Paris Est, 2014.
- [111] NASA and METI. ASTER Global Digital Elevation Map. Published by NASA (National Aeronautics and Space Administration of the USA) and METI (Ministry of Economy, Trade, and Industry of Japan) <https://asterweb.jpl.nasa.gov/gdem.asp>, 2011.
- [112] H. NASRUL. Introduction to photogrammetry and air survey, presentation. <https://slideplayer.fr/slide/9362136/>.
- [113] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.
- [114] Novatel. SPAN-IGM-A1, product specifications. <http://www.novatel.com/products/span-gnss-inertial-systems/span-combined-systems/span-igm-a1/>, 2016. [Online].
- [115] A. Nowak. Dynamic GNSS Mission Planning Using DTM for Precise Navigation of Autonomous Vehicles. *The Journal of Navigation*, 70(3):483–504, May 2017.
- [116] S. O. Mason. *Expert system-based design of photogrammetric networks*. PhD thesis, ETH-Zürich, Zürich, May 1994.
- [117] M. Osborne. Mission planner software, 2019. <http://ardupilot.org/planner/>.
- [118] G. Olgague. *Planification du placement de caméras pour des mesures 3D de précision*. PhD thesis, Institut National Polytechnique de Grenoble, 1998.
- [119] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.
- [120] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [121] S. A. Papert. The Summer Vision Project, project mac, vision memo. no. 100. *Massachusetts Institute of Technology, Artificial Intelligence group* <http://dspace.mit.edu/handle/1721.1/6125>, July 1966.



- [122] N. Pfeifer, P. Glira, and C. Briese. Direct georeferencing with on board navigation components of light weight UAV platforms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B7:487–492, 08 2012.
- [123] PhaseOne. Phase One Aerial Adds iXA-R Camera Platform documentation. [https://industrial.phaseone.com/documents/Phase\\_One\\_PR\\_New\\_Products.pdf](https://industrial.phaseone.com/documents/Phase_One_PR_New_Products.pdf), September 2014. (Accessed on 02/2020).
- [124] E. J. Piatti and J. L. Lerma. Virtual Worlds for Photogrammetric Image-Based Simulation and Learning. *The Photogrammetric Record*, 28(141):27–42, Mar. 2013.
- [125] A. G. Pires, D. G. Macharet, and L. Chaimowicz. Towards Cooperative Localization in Robotic Swarms. In *Distributed Autonomous Robotic Systems*, pages 105–118. Springer, Tokyo, 2016.
- [126] A. Pujol-Miro, J. Ruiz-Hidalgo, and J. R. Casas. Registration of images to unorganized 3D point clouds using contour cues. In *25th European Signal Processing Conference (EUSIPCO)*, pages 81–85, Aug. 2017. ISSN: 2076-1465.
- [127] M. Rehak. *Integrated sensor orientation on micro aerial vehicles*. PhD thesis, EPFL, 2017.
- [128] M. Rehak, R. Mabillard, and J. Skaloud. A micro-UAV with the capability of direct georeferencing. *ISPRS - The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 317–323, 2013.
- [129] M. Rehak, R. Mabillard, and J. Skaloud. A micro aerial vehicle with precise position and attitude sensors. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 4:239–251, 2014.
- [130] M. Rehak and J. Skaloud. Fixed-wing micro aerial vehicle for accurate corridor mapping. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-1/W1:23–31, 2015.
- [131] M. Rehak and J. Skaloud. Applicability of new approaches of sensor orientation to micro aerial vehicles. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:441–447, 2016.
- [132] M. Rehak and J. Skaloud. Performance assessment of integrated sensor orientation with a low-cost gnss receiver. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-2-W3, pages 75–80. Copernicus GmbH, Aug. 2017.

## Bibliography

---

- [133] F. Remondino and C. Fraser. Digital camera calibration methods: considerations and comparisons. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):266–272, 2006.
- [134] Riegl. VQ-48-U documentation. [http://www.riegl.com/uploads/tx\\_pxpriegldownloads/DataSheet\\_VQ-480-U\\_2015-03-24.pdf](http://www.riegl.com/uploads/tx_pxpriegldownloads/DataSheet_VQ-480-U_2015-03-24.pdf), March 2015. (Accessed on 02/2020).
- [135] RIEGL. RIEGL VUX-1 UAV documentation. <http://www.riegl.com/products/unmanned-scanning/riegl-vux-1uav/>, 2020.
- [136] F.-H. Rouet. Partial computation of the inverse of a large sparse matrix - application to astrophysics. Master's thesis, Institut National Polytechnique de Toulouse, Toulouse, 2009.
- [137] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov. 2011.
- [138] E. Rupnik, M. Daakir, and M. P. Deseilligny. Micmac, apero, pastis and other beverages in a nutshell! ign. <https://github.com/micmacIGN/Documentation>, Jan. 2018.
- [139] P. Schaer, J. Skaloud, Y. Stebler, P. Tome, and R. Stengele. Airborne lidar in-flight accuracy estimation. *GPS World*, 20(8):37–41, 2009.
- [140] S. Schiess. UAV based photogrammetry for snow height determination with GNSS/INS assisted aerial control. Master thesis, EPFL, June 2019.
- [141] senseFly. senseFly drones and cameras. <http://www.sensefly.com>, 2015. [Online; accessed 13-August-2016].
- [142] senseFly SA. eBee plus: Aerial efficiency, photogrammetric accuracy. <https://www.sensefly.com/drones/ebec-plus.html>, 2016. [Online].
- [143] J. Skaloud. LIEO – lidar orientation and point cloud generation. *Software description No. 6.0704, Swiss Federal Institute of Technology Lausanne (EPFL)*, 2017.
- [144] J. Skaloud and D. D. Lichti. Rigorous approach to bore-sight self calibration in airborne laser scanning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61:47–59, 10 2006.
- [145] J. Skaloud, M. Rehak, and D. D. Lichti. Mapping with MAV: Experimental study on the contribution of absolute and relative position control. In *ISPRS - The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 40-3/W1, pages 123–129, Castelldefels, Spain, 2014.

- [146] J. Skaloud and P. Schaer. Automated assessment of digital terrain models derived from airborne laser scanning. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 2:105–114, 2012.
- [147] SketchUp. *3D Design Software, 3D Modeling on the Web*. Trimble, <https://www.sketchup.com/>, 2019.
- [148] A. Stoven-Dubois, L. Jospin, and D. Cucci. Cooperative navigation for an UAV tandem in GNSS denied environments. *Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018)*, pp. 2332–2339, Miami, Florida, September, 2018.
- [149] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College London - Department of Computing, 2012.
- [150] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldhash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:66–78, 2012.
- [151] C. Strecha, O. Küng, and P. Fua. Automatic mapping from ultra-light UAV imagery. *EuroCOW 2012, Barcelona, Spain, February 8-10*, 2012.
- [152] C. Strecha, L. Van Gool, and P. Fua. Ia generative model for true orthorectification. *ISPRS, XXXVII(Part B3a):303–308*, 2008.
- [153] J. Stubbs and D. Akos. GNSS/GPS robustness for UAS. In *Proceedings of the International Technical Meeting of The Institute of Navigation, Monterey, California*, pages 485–493, January 2016.
- [154] X. Sun. *Localizing Polygonal Objects in Man-Made Environments*. PhD thesis, ISIM, EPFL, Lausanne, 2015.
- [155] F. Taillandier and R. Deriche. Reconstruction of 3d linear primitives from multiple views for urban areas modelisation. *International Archives of Photogrammetry and Remote Sensing, XXXIV-3:267–272*, 2012.
- [156] P. Teunissen, P. Joosten, and D. Odijk. The reliability of GPS ambiguity resolution. *GPS Solutions*, 2(3):63–69, 1999.
- [157] E. Thierry. Technologie et DP3 - La technologie au collège. <http://tkcollege.fr/informatique---sketchup---appartement.html>.
- [158] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

## Bibliography

---

- [159] S. Tully, G. Kantor, and H. Choset. Leap-Frog Path Design for Multi-Robot Cooperative Localization. In A. Howard, K. Iagnemma, and A. Kelly, editors, *Field and Service Robotics*, number 62 in Springer Tracts in Advanced Robotics, pages 307–317. Springer Berlin Heidelberg, 2010.
- [160] S. Tushev and B. Sukhovilov. Photogrammetric system accuracy estimation by simulation modelling. In *International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–6, May 2017.
- [161] J. Vallet, A. Gressin, P. Clausen, and J. Skaloud. Airborne and mobile lidar, which sensors for which application? In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLIII-B1, Nice, France, 2020.
- [162] S. Varea, C. Larson, and S. Minchin. La fabuleuse histoire de la tête de maori de rouen et sa numérisation. *XYZ*, No. 127:29–32, 2011.
- [163] Velodyne. Velodyne Lidar website. <https://velodynelidar.com/>, 2020.
- [164] H. A. Vu. Geometry of Industrial Panoramic Images, Problems and Solutions. Master's thesis, l'X École Polytechnique, EDF: Electricité de France, 2014.
- [165] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.
- [166] T. R. Wanasinghe, G. K. I. Mann, and R. G. Gosine. Distributed Leader-Assistive Localization Method for a Heterogeneous Multirobotic System. *IEEE Transactions on Automation Science and Engineering*, 12(3):795–809, July 2015.
- [167] Y. Wu and Z. Hu. Pnp problem revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, 2006.