Università degli Studi di Udine
Dipartimento Politecnico di Ingegneria e Architettura
Ph.D in Industrial and Information Engineering

Eleonora Maset

Advanced Methods for LiDAR and Photogrammetric Data Processing: from Procrustes Analysis to Deep Learning

Supervision
Prof. Andrea Fusiello

Co-Supervision
Prof. Fabio Crosilla

January 31, 2019

Abstract

The soul of the thesis is dual, reflecting the different research activities that characterized this PhD project, which was carried out part at the University and part at Helica s.r.l., an Italian company specialized in airborne remote sensing.
In the first part, orientation problems in photogrammetry and laser scanning are studied from a methodological point of view and solved via *Procustes Analysis*, a set of least squares mathematical tools used to perform transformations among corresponding points belonging to a generic k-dimensional space, in order to satisfy their maximum agreement. In particular, novel Procrustes models are developed and exploited in several applications, ranging from the exterior orientation of an image, to the matching between two sets of keypoints and to the registration of multiple point clouds.
The second part focuses on the implementation of novel algorithms for the processing of photogrammetric and LiDAR data acquired by helicopters or Unmanned Aerial Vehicles. In particular, we present an innovative method based on deep learning and Convolutional Neural Networks to perform the classification of full-waveform LiDAR data. Moreover, we propose an original pipeline developed to create seamless planar mosaics from aerial images, based on a global approach known as *synchronization*, applied for image registration and colour correction.

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Andrea Fusiello, and my co-supervisor, Prof. Fabio Crosilla, for guiding me during these three years. This work would not have been possible without their support and help.

I would also like to thank all my colleagues at Helica s.r.l. for sharing with me their knowledge, for their cooperation and for trusting me since the first day.

My deepest gratitude goes to my parents for being always beside me, for their never-ending patience, support and encouragement.

To Andrea C.

Contents

Chapter 1

Introduction

Information about the physical environment can be nowadays easily obtained thanks to *photogrammetry* and *laser scanning*. The aim of these two technologies is to provide automated or semi-automated procedures for solving tasks in mapping, surveying and high-precision metrology, with a particular attention on accuracy, reliability and completeness of the extracted information [53].
During this PhD, we focused on different phases of the photogrammetric and laser scanning data processing, developing innovative methods and procedures for the solution of problems such as image orientation, multi-view matching and LiDAR (Light Detection and Ranging) data classification. Carried out in collaboration with Helica s.r.l., an Italian company specialized in airborne remote sensing, the PhD project was divided into two main branches, from which the dual soul of this thesis originates.
On the one hand, together with the members of the research group *Geomatics and Computer Vision* of the Polytechnic Department of Engineering and Architecture of Udine University, orientation problems in photogrammetry and laser scanning were studied from a methodological point of view and *Procustes Analysis* was proposed for their solution. On the other hand, the research activity carried out within the company was focused on remote sensing applications. Advanced techniques were applied and evaluated for the processing of photogrammetric and LiDAR data acquired from helicopters or Unmanned Aerial Vehicles (UAVs), with the aim of creating cartographic products useful in infrastructure management, environmental monitoring and land cover change detection.

The thesis is therefore divided into two parts, reflecting the parallel research activities completed in these three years. More in detail, it is organized as follows.
As already mentioned, in the first part *Procrustes Analysis* is applied to solve orientation problems. The term *Procrustes Analysis* is referred to a set of least squares mathematical models used to perform transformations among corresponding matrix elements belonging to a generic k-dimensional space, in order to satisfy their maximum agreement. Several variations of Procrustes models have been proposed in the literature, according to the searched transformation parameters. In Chapter 2 we provide a comprehensive survey on Procrustes Analysis and at the same time we develop novel total least squares solutions, applying the *Errors-In-Variables* model to classical Procrustes Analysis formulations. One of the proposed method is employed to compute the image exterior orientation parameters. Chapter 3 addresses instead the problem of finding correspondences among element sets, that can be seen as a Permutation Procrustes problem in the case of two sets. In particular, we propose an innovative efficient solution to the multi-view matching problem, based on a spectral decomposition. An extended formulation of the *Generalized Procrustes Analysis* is presented in Chapter 4, which allows to apply Procrustes Analysis even if the correspondences between matrix elements are unknown. This method finds application in the registration of multiple point clouds. Finally, Chapter 5 proposes an affine space formulation of the *Extended Orthogonal Procrustes Analysis* that allows to compute the transformation between two matrices composed by both points and vectors. The method is successfully applied to perform the Virtual Trial Assembly of a complex steel structure.

The second part of the thesis is instead devoted to remote sensing applications. Chapter 6 describes an innovative algorithm to perform LiDAR point cloud classification, that is based on Convolutional Neural Networks and takes advantage of full-waveform data registered by modern laser scanners. Thanks to the employed architecture, even challenging classes such as power line and transmission tower can be automatically identified, representing a valuable support tool in the data analysis for the management and maintenance of electric power lines. Chapter 7 presents instead a novel method to create high-quality seamless planar mosaics. The proposed approach allows to stitch together multiple images, ensuring at the same time good robustness against many common mosaicking problems (e.g., misalignments, colour distortion, moving objects, parallax). The creation of a unique wide image facilitates the usage and the interpretation of the single photos acquired from a helicopter or a UAV.

Part of the material contained in this thesis is taken from the following works.

1. Fabio Crosilla, Alberto Beinat, Andrea Fusiello, Eleonora Maset and Domenico Visintini. Advanced Procrustes Analysis Models in Photogrammetric Computer Vision. *CISM International Centre for Mechanical Sciences*, Springer, in press.

2. Stefano Zorzi, Eleonora Maset, Andrea Fusiello and Fabio Crosilla. Full-Waveform Airborne LiDAR Data Classification using Convolutional Neural Networks. *Under Minor Revision.*

3. Daniele Zonta, Eleonora Maset, Ivan Mario Alba, Fabio Crosilla and Andrea Fusiello. Virtual Trial Assembly of Complex Steel Structures by Affine Procrustes Analysis: the Case of *Vessel* in New York. *Submitted.*

4. Emanuele Santellani, Eleonora Maset and Andrea Fusiello. Seamless image mosaicking via synchronization. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume IV-2, pages 247–254, 2018.

5. Eleonora Maset, Federica Arrigoni and Andrea Fusiello. Practical and efficient multiview matching. *International Conference on Computer Vision (ICCV)*, pages 4568 – 4576, 2017.

6. Federica Arrigoni, Eleonora Maset and Andrea Fusiello. Synchronization in the symmetric inverse semigroup. *International Conference on Image Analysis and Processing (ICIAP)*, volume 10485 of *Lecture Notes in Computer Science*, pages 70 – 81. Springer International Publishing, 2017.

7. Fabio Crosilla, Eleonora Maset and Andrea Fusiello. Procrustean Photogrammetry: from exterior orientation to bundle adjustment. *New Advanced GNSS and 3D Spatial Techniques. Lecture Notes in Geoinformation and Cartography*, pages 157–165. Springer, Cham, 2017.

8. Eleonora Maset, Fabio Crosilla and Andrea Fusiello. Errors-in-Variables Anisotropic Extended Orthogonal Procrustes Analysis. *IEEE Geoscience and Remote Sensing Letters*, 14(1), pages 57–61, 2017.

Part I

Solving Orientation Problems via Procrustes Analysis

Photogrammetry and laser scanning are two scientific fields whose analytical models are fundamentally based on geometrical transformations of point coordinates expressed in different reference frames. From the analytical point of view, the main issue of these transformations is that they are expressed, almost always, by nonlinear models. To solve the transformation problems, it is common to resort to the linearization of the original models and to the solution of linearized systems of equations in order to reach, through the introduction of a proper error distribution model, the best estimation of the unknown parameters and of their precision. To carry out the computational procedure, it is therefore necessary to determine, by a different method, the approximate value of the unknown parameters.
The advantage of using computational tools that do not require a linearization process of the equation system and the consequent knowledge of the approximate values of the unknown parameters is therefore clearly evident. For this reason, in this part we propose an alternative procedure, well known in the multifactorial analysis field as *Procrustes Analysis*, that allows to directly solve via least squares nonlinear systems that generally characterize orientation problem in photogrammetry and laser scanning. Procrustes algorithms do not require approximate numerical solutions of the unknown parameters and offer the capability to take into consideration weights of the systems involved, requiring only matrix products and the singular value decomposition of a matrix.
As it will be shown in the following chapters, applications of Procrustes Analysis range from the exterior orientation of an image, to the matching between two sets of keypoints and to the registration of multiple point clouds. Moreover, it will be demonstrated how it represents a valuable tool also to perform the Virtual Trial Assembly of the elements of a steel structure.

Chapter 2

Procrustes Analysis Models

Procrustes Analysis is a well known least squares technique that allows to directly find the transformation parameters between origin and destination sets of observations belonging to a k-dimensional space. This chapter provides a unifying view on the many variants of Procrustes models, that can be employed to solve nonlinear transformation problems that generally characterize the analytical theory behind photogrammetry and laser scanning. In particular, we apply an *Errors-In-Variables* model to Procrustes Analysis and derive novel total least squares solutions that can deal with the uncertainty affecting both sets of observations.

2.1 Introduction

The terms *Procrustes Analysis* or *Procrustes Techniques* are referred to a set of least squares mathematical models used to perform transformations among corresponding matrix elements belonging to a generic k-dimensional space, in order to satisfy their maximum agreement. They are particularly appealing from the computational point of view, for they employ only matrix products and the singular value decomposition of a $k \times k$ matrix, requiring neither the linearization of equations nor the knowledge of approximate parameters values.
The name *Procrustes* comes from the Greek mythology. Procrustes was in fact a giant that tortured his victims forcing them to lie on an iron bed, and fitting their height to the bed length. If the victim was shorter than the bed length, his body was lengthened and hammered, if the victim height was longer, the outgoing part was cut off.
The term was first proposed by Hurley and Cattel in 1962 [81], but the origins of these techniques are older. Goodall [61] quotes the psychometrist Mosier [118] as one of the first developers of the method. The morphometrist Cole [33], instead, indicates Boas [17] e his fellow Phelps [126] as inventors of the Orthogonal Procrustes Analysis and of an initial form of the Generalized Procrustes Analysis, respectively.
A continuous branch of research on Procrustes Analysis, that starts from the fifties of the last century and that is still active nowadays, is the one followed by the psychometrists. To the fundamental works by Green [70], Cliff [32], Schönemann [133], Schönemann and Carroll [134] and Gower [66] in multifactorial analysis, the studies by Lissitz et al. [104], ten Berge [147], ten Berge and Knol [150] and ten Berge et al. [149] can be added, until the most recent contributions by ten Berge [148] and Bennani Dosse and ten Berge [15].
Parallel to this activity, Procrustes techniques have been successfully applied in shape analysis. The reference works are those from Mardia et al. [115], Kendall [85], Bookstein [18, 19], Dryden and Mardia [44], Goodall [61], Kent [87] and Stoyan et al. [29]. Some books on this topic are from Stoyan et al. [29], Small [140] and Dryden and Mardia [44]. Recent contributions in different fields are given, for instance, by Gower and Dijksterhuis [67], Larsen [96], Wang and Mahadevan [157], Kenobi and Dryden [86].
The first application of Procrustes algorithms in geodesy is due to Crosilla [36, 37], who proposed this technique for the construction of an ideal variance-covariance matrix for control networks.

Currently, Procrustes techniques are widely used in statistics, in multifactorial analysis, in biometrics, in management engineering, as well as in computer applications like image matching and robotic vision. Procrustes Analysis has been successfully applied in geodesy, photogrammetry and laser scanning by Awange [9], Crosilla [38], Beinat, Crosilla and Visintini [14], Beinat [11], Crosilla and Beinat [39], Grafarend and Awange [68, 69]. More recent contributions in these fields are due to Crosilla and Beinat [12, 39, 13, 40], Awange and Grafarend [10], Toldo et al. [151], Garro et al. [57], Fusiello et al. [56] and Fusiello and Crosilla [54, 55].

2.1.1 Contribution

In this chapter we provide a comprehensive survey on Procrustes Analysis, gathering several models that have been proposed in the last decades in different communities, including also photogrammetry, computer vision and laser scanning. At the same time we propose novel total least squares solution, applying the Errors-In-Variables model to classical Procrustes Analysis formulations. More precisely, our contributions are the following.
First, Section 2.2 gives a brief overview and a classification of the variants of Procrustes Analysis. Then, in Section 2.3 we review the least squares (LS) solutions of the most common orthogonal Procrustes models, namely the so called *Orthogonal* (OPA), *Extended Orthogonal* (EOPA), *Weighted Extended Orthogonal* (WEOPA) and *Anisotropic Extended Orthogonal* (AEOPA) models.
Secondly, we introduce in Section 2.4 the so called *Errors-in-Variables* (EIV) model and its solutions that can be found in the literature. Section 2.5 represents the main contribution of this chapter and is devoted to the generalization of Procrustes Analysis to the EIV model. In contrast to ordinary least squares Procrustes approach, that finds the transformation parameters between origin and destination sets of observations minimizing errors affecting only the destination one, we present the total least squares (TLS) solution of several isotropic and anisotropic Procrustes models, that can deal with the uncertainty affecting both sets of observations. In particular, we derive novel TLS solutions for the WEOPA (Section 2.5.2) and AEOPA (Section 2.5.3) models.
Finally, in Section 2.6 the TLS solution of the WEOPA model is tested to perform the datum transformation of points belonging to a small topographic network, while the TLS version of the AEOPA is applied to solve the exterior orientation of an image and compared with the ordinary LS AEOPA solution.

2.2 Classification of Orthogonal Procrustes Models

A first subdivision in the wide field of Procrustes algorithms can be made considering *Oblique* and *Orthogonal* models. While the first ones aim to separately rotate each axis of the origin datum in such a way to find the best approximation with respect to the destination set [107, 20], the latter impose an orthogonal constraint to the transformation matrix. Due to the nature of the problems encountered in photogrammetry and laser scanning, we will focus only on orthogonal models and, in particular, we are interested in rotation matrices, i.e., orthogonal matrices with positive determinant.
According to the searched parameters and to the stochastic model associated to the point configurations, the various orthogonal Procrustes models can be identified as:

- OPA – *Orthogonal Procrustes Analysis* that allows to determine the least squares rotations between two configurations;
- EOPA – *Extended OPA* that permits to define translations, rotations and an isotropic scale factor of a least squares similarity transformation between two geometrical configurations;
- WEOPA – *Weighted EOPA* that allows to obtain the expressions for the rotations, translations and the isotropic scale factor between two sets of points separately weighted;

- AEOPA – *Anisotropic EOPA* that permits to define translations, rotations and anisotropic scale factors for each component or for each measurement of two geometrical configurations.

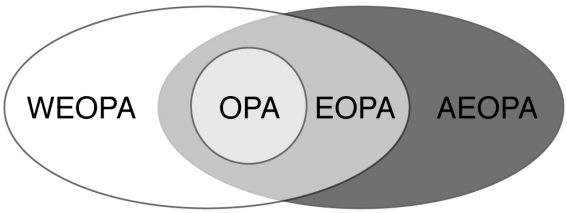


FIGURE 2.1: Functional relationship among the Orthogonal Procrustes Analysis models.

Figure 2.1 shows the relationships existing among the various models. OPA model can be considered a particular case of the EOPA, that in turn can be considered as a simple case of the most general WEOPA model. The more recent AEOPA formulation can be seen as a further generalization, allowing a different scaling of the components or of the points belonging to the two configurations. Alongside these models, the *Generalized Procrustes Analysis* (GPA) has also been proposed to simultaneously consider a virtually unlimited number of matrix configurations. Employing the tools of the Orthogonal Procrustes models, GPA determines the mean configuration among the given ones and the parameters of the various similarity transformations that relate each configuration to the mean one. GPA will be described in detail in Chapter 4, showing how it can be successfully applied to perform the global registration of multiple point clouds.

2.3 Least Squares Solutions of Procrustes Models

In the following sections we will review in detail the aforementioned Orthogonal Procrustes models and report their least squares solutions. This is a preparatory step to subsequently introduce the EIV approach and derive the novel total least squares solutions.

2.3.1 Orthogonal Procrustes Analysis (OPA)

This is the fundamental model, formulated by Green in [70], that allows to match two matrices only by rotation, satisfying the least squares principle. The following description is inspired by the works by Schönemann [133], Crosilla [36] and Beinat [11] with appropriate integration. Let us consider a matrix $\mathbf{A}$ (origin or source) and a matrix $\mathbf{B}$ (destination or objective), containing a set of numerical data, e.g., the coordinates of p-points in $\mathbb{R}^k$, so defined:

$$\mathbf{A} = \begin{bmatrix} x_{11}^A & x_{12}^A & \cdots & x_{1k}^A \\ x_{21}^A & x_{22}^A & \cdots & x_{2k}^A \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1}^A & x_{p2}^A & \cdots & x_{pk}^A \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} x_{11}^B & x_{12}^B & \cdots & x_{1k}^B \\ x_{21}^B & x_{22}^B & \cdots & x_{2k}^B \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1}^B & x_{p2}^B & \cdots & x_{pk}^B \end{bmatrix}. \tag{2.1}$$

The method allows to directly estimate the unknown rotation matrix $\mathbf{R}$, for which the square of the Frobenius (or Euclidean) norm of the transformation residual matrix is minimum, i.e.

$$\|\mathbf{E}\|_F^2 = \|\mathbf{AR} - \mathbf{B}\|_F^2 = \mathrm{tr}\,(\mathbf{E}'\mathbf{E}) = \mathrm{tr}\,(\mathbf{E}\mathbf{E}') = \min \tag{2.2}$$

with the condition that the $k \times k$ matrix $\mathbf{R}$ is orthogonal, that is $\mathbf{R}'\mathbf{R} = \mathbf{I}$. Defining a function $F = F(\mathbf{E}, \mathbf{R}, \mathbf{L})$, where $\mathbf{L}$ is a diagonal $k \times k$ matrix of unknown Lagrangian

multipliers, and exploiting the trace properties, the cost function F can be expressed as:

$$\begin{aligned} F =& \mathrm{tr}\left(\mathbf{E}'\mathbf{E}\right) + \mathrm{tr}\left[\mathbf{L}\left(\mathbf{R}'\mathbf{R} - \mathbf{I}\right)\right] \\ =& \mathrm{tr}\left[\left(\mathbf{AR} - \mathbf{B}\right)'\left(\mathbf{AR} - \mathbf{B}\right) + \mathbf{L}\left(\mathbf{R}'\mathbf{R} - \mathbf{I}\right)\right] \\ =& \mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{AR} - \mathbf{R}'\mathbf{A}'\mathbf{B} - \mathbf{B}'\mathbf{AR} + \mathbf{B}'\mathbf{B} + \mathbf{LR}'\mathbf{R} - \mathbf{L}\right]. \end{aligned} \tag{2.3}$$

The minimum condition is reached imposing the partial derivative of F with respect to $\mathbf{R}$ equal to zero:

$$\begin{aligned} \frac{\partial F}{\partial \mathbf{R}} =& \left(\mathbf{A}'\mathbf{A} + \mathbf{A}'\mathbf{A}\right)\mathbf{R} - 2\mathbf{A}'\mathbf{B} + \mathbf{R}\left(\mathbf{L} + \mathbf{L}'\right) \\ =& 2\mathbf{A}'\mathbf{AR} - 2\mathbf{A}'\mathbf{B} + \mathbf{R}\left(\mathbf{L} + \mathbf{L}'\right) = 0. \end{aligned} \tag{2.4}$$

Multiplying on the left all the elements by $\mathbf{R}'$

$$2\mathbf{R}'\mathbf{A}'\mathbf{AR} - 2\mathbf{R}'\mathbf{A}'\mathbf{B} + \mathbf{R}'\mathbf{R}\left(\mathbf{L} + \mathbf{L}'\right) = 0 \tag{2.5}$$

one can see that, being $\mathbf{R}'\mathbf{A}'\mathbf{AR}$ and $\mathbf{L}+\mathbf{L}'$ symmetric matrices, and exploiting the orthogonal condition of matrix $\mathbf{R}$, also $\mathbf{R}'\mathbf{A}'\mathbf{B}$ must be symmetric. Calling

$$\mathbf{S} = \mathbf{A}'\mathbf{B} \tag{2.6}$$

and taking into account the symmetry of $\mathbf{R}'\mathbf{S}$, it follows:

$$\mathbf{R}'\mathbf{S} = \left(\mathbf{R}'\mathbf{S}\right)' = \mathbf{S}'\mathbf{R}. \tag{2.7}$$

Carrying out the Singular Value Decomposition (SVD) of $\mathbf{S} = \mathbf{V}\mathbf{D}_S\mathbf{W}'$ and substituting it in the previous relationship yields:

$$\mathbf{R}'\mathbf{V}\mathbf{D}_S\mathbf{W}' = \left(\mathbf{V}\mathbf{D}_S\mathbf{W}'\right)'\mathbf{R} = \mathbf{W}\mathbf{D}_S\mathbf{V}'\mathbf{R}. \tag{2.8}$$

From a comparison of the corresponding terms of the matrix products, it results:

$$\mathbf{W}' = \mathbf{V}'\mathbf{R}.$$

Reminding that $\mathbf{V}$ and $\mathbf{W}$ are orthonormal, it is possible to derive the expression of matrix $\mathbf{R}$, that represents the final result obtained as a product of two eigenvector matrices:

$$\mathbf{R} = \mathbf{V}\mathbf{W}'. \tag{2.9}$$

This formula only guarantees that $\mathbf{R}$ is orthogonal. The least squares estimate of a rotation matrix is obtained by [156]

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}\left(1, 1, \det(\mathbf{V}\mathbf{W}')\right)\mathbf{W}'. \tag{2.10}$$

Summarizing, from the computational point of view, given $\mathbf{A}$ and $\mathbf{B}$, after computing the SVD of the matrix product $\mathbf{A}'\mathbf{B}$, that is $\mathbf{A}'\mathbf{B} = \mathbf{V}\mathbf{D}_S\mathbf{W}'$, it is possible to directly determine the rotation matrix $\mathbf{R}$ as $\mathbf{R} = \mathbf{V}\,\mathrm{diag}\left(1, 1, \det(\mathbf{V}\mathbf{W}')\right)\mathbf{W}'$.

2.3.2 Extended Orthogonal Procrustes Analysis (EOPA)

EOPA model computes the least squares similarity transformation between two matrices. Its first formulation was given by Schönemann and Carroll [134] and is reported in the following.
Given matrices $\mathbf{A}$ and $\mathbf{B}$, with the meaning assigned in the preceding section, EOPA allows to define, in addition to the rotation matrix $\mathbf{R}$ already retrieved with the OPA model, also a translation vector $\mathbf{t}$ and a global scale factor c. It corresponds to searching a matrix $\tilde{\mathbf{B}}$

defined as:

$$\tilde{\mathbf{B}} = c\mathbf{AR} + \mathbf{jt}' \tag{2.11}$$

with $\mathbf{j} = (1, 1, \cdots, 1)'$ of dimensions $p \times 1$, for which the following condition is satisfied:

$$\|\mathbf{E}\|_F^2 = \left\|\mathbf{B} - \tilde{\mathbf{B}}\right\|_F^2 = \min \tag{2.12}$$

under the orthogonality constraint $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$. The minimum condition reported in (2.12) is equivalent to:

$$\operatorname{tr}(\mathbf{E}'\mathbf{E}) = \min. \tag{2.13}$$

Defining a Lagrangian function $F = F(\mathbf{E}, \mathbf{t}, \mathbf{R}, c, \mathbf{L})$, where $\mathbf{L}$ is an unknown matrix of Lagrangian multipliers, it follows

$$\begin{aligned} F =& \operatorname{tr}(\mathbf{E}'\mathbf{E}) + \operatorname{tr}[\mathbf{L}(\mathbf{R}'\mathbf{R} - \mathbf{I})] \\ =& \operatorname{tr}[(\mathbf{B} - c\mathbf{AR} - \mathbf{jt}')'(\mathbf{B} - c\mathbf{AR} - \mathbf{jt}') + \mathbf{L}(\mathbf{R}'\mathbf{R} - \mathbf{I})] \\ =& \operatorname{tr}\left[\mathbf{B}'\mathbf{B} + c^2\mathbf{R}'\mathbf{A}'\mathbf{AR} + \mathbf{tj}'\mathbf{jt}' - 2c\mathbf{R}'\mathbf{A}'\mathbf{jt}' - 2\mathbf{B}'\mathbf{jt}' - 2c\mathbf{B}'\mathbf{AR} + \mathbf{LR}'\mathbf{R} - \mathbf{L}\right] \end{aligned} \tag{2.14}$$

and imposing the partial derivatives of F with respect to $\mathbf{R}$, $\mathbf{t}$ and c equal to zero, it results:

$$\frac{\partial F}{\partial \mathbf{R}} = 2c^2\mathbf{A}'\mathbf{AR} - 2c\mathbf{A}'\mathbf{B} + 2c\mathbf{A}'\mathbf{jt}' + \mathbf{R}(\mathbf{L} + \mathbf{L}') = 0 \tag{2.15}$$

$$\frac{\partial F}{\partial \mathbf{t}} = 2\mathbf{j}'\mathbf{jt} + 2c\mathbf{R}'\mathbf{A}'\mathbf{j} - 2\mathbf{B}'\mathbf{j} = 0 \tag{2.16}$$

$$\frac{\partial F}{\partial c} = 2c\operatorname{tr}(\mathbf{R}'\mathbf{A}'\mathbf{AR}) - 2\operatorname{tr}(\mathbf{R}'\mathbf{A}'\mathbf{B} - \mathbf{R}'\mathbf{A}'\mathbf{jt}') = 0 \tag{2.17}$$

Please note that the product $\mathbf{j}'\mathbf{j}$ generates a scalar p, corresponding to the number of rows of $\mathbf{A}$ and $\mathbf{B}$. After multiplying Formula (2.15) on the left by $\mathbf{R}'$, observing that $\mathbf{R}'\mathbf{A}'\mathbf{AR}$ and $\mathbf{R}'\mathbf{R}(\mathbf{L} + \mathbf{L}')$ are symmetric, one can notice that also

$$\mathbf{R}'\mathbf{A}'\mathbf{B} - \mathbf{R}'\mathbf{A}'\mathbf{jt}' \tag{2.18}$$

must be symmetric. Substituting in Equation (2.18) the expression of $\mathbf{t}$ that results from (2.16):

$$\mathbf{t} = (\mathbf{B} - c\mathbf{AR})'\frac{\mathbf{j}}{\mathbf{j}'\mathbf{j}} \tag{2.19}$$

the symmetric condition for the global term becomes:

$$\mathbf{R}'\mathbf{A}'\mathbf{B} - \mathbf{R}'\mathbf{A}'\mathbf{j}\frac{\mathbf{j}'}{\mathbf{j}'\mathbf{j}}(\mathbf{B} - c\mathbf{AR}) = \mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{jj}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B} + c\mathbf{R}'\mathbf{A}'\left(\frac{\mathbf{jj}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{AR}. \tag{2.20}$$

Since $\mathbf{R}'\mathbf{A}'\left(\frac{\mathbf{jj}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{AR}$ is symmetric, it consequently happens that:

$$\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{jj}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B} \tag{2.21}$$

is also symmetric. Matrix $\mathbf{I} - \mathbf{jj}'/\mathbf{j}'\mathbf{j}$ is symmetric and idempotent and its role is to translate the matrix values to which it is applied (in this case $\mathbf{A}$ and $\mathbf{B}$) to the corresponding barycenter. Calling

$$\mathbf{S} = \mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{jj}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B} \tag{2.22}$$

as done for the OPA model, it results:

$$\mathbf{R}'\mathbf{S} = (\mathbf{R}'\mathbf{S})' = \mathbf{S}'\mathbf{R} \tag{2.23}$$

from which one obtains

$$\mathbf{R} = \mathbf{V}\mathbf{W}' \tag{2.24}$$

with $\mathbf{S} = \mathbf{V}\mathbf{D}_s\mathbf{W}'$ SVD of $\mathbf{S}$. In order to retrieve a rotation matrix, one must resort to the following formulation:

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}\,(1, 1, \det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}'. \tag{2.25}$$

Once $\mathbf{R}$ is known, one can compute $\mathbf{t}$ from (2.19) and the scale factor c from Equation (2.17). Substituting the expression for vector $\mathbf{t}$, the solution becomes:

$$c = \frac{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B}\right]}{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{A}\mathbf{R}\right]} = \frac{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B}\right]}{\mathrm{tr}\left[\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{A}\right]} \tag{2.26}$$

after having considered the orthogonality condition of $\mathbf{R}$. It is easy to see that the residual matrix $\mathbf{E}$, that measures the discrepancies of matrices $\mathbf{A}$ and $\mathbf{B}$ after having applied the similarity transformation, is independent of the translation vector $\mathbf{t}$. In other words, the matching result does not depend on the relative distance between the barycenters of $\mathbf{A}$ and $\mathbf{B}$. In fact, one can retrieve $\mathbf{E}$ as:

$$\mathbf{E} = \mathbf{B} - \tilde{\mathbf{B}} = \left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)(\mathbf{B} - c\mathbf{A}\mathbf{R}) \tag{2.27}$$

where $\mathbf{t}$ is not considered. Summarizing, given $\mathbf{A}$ and $\mathbf{B}$, it is convenient to first calculate the barycenter matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$:

$$\bar{\mathbf{A}} = \left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{A} = \left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{p}\right)\mathbf{A} \tag{2.28}$$

$$\bar{\mathbf{B}} = \left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{p}\right)\mathbf{B}. \tag{2.29}$$

Once the SVD decomposition of the matrix product $\bar{\mathbf{A}}'\bar{\mathbf{B}}$ has been carried out, that is $\bar{\mathbf{A}}'\bar{\mathbf{B}} = \mathbf{V}\mathbf{D}_s\mathbf{W}'$, it is possible to directly compute $\mathbf{R} = \mathbf{V}\mathbf{W}'$. The scale factor is then given by:

$$c = \frac{\mathrm{tr}\left[\mathbf{R}'\bar{\mathbf{A}}'\bar{\mathbf{B}}\right]}{\mathrm{tr}\left[\bar{\mathbf{A}}'\bar{\mathbf{A}}\right]} \tag{2.30}$$

which corresponds to the least squares solution of $\bar{\mathbf{B}} = c\bar{\mathbf{A}}\mathbf{R}$, while the translation is:

$$\mathbf{t} = (\mathbf{B} - c\mathbf{A}\mathbf{R})'\mathbf{j}/p. \tag{2.31}$$

Please note that multiplying by the term $\mathbf{j}/p$ in the previous expression corresponds to perform the mean of the difference $\mathbf{B} - c\mathbf{A}\mathbf{R}$.

2.3.3 Weighted Extended Orthogonal Procrustes Analysis (WEOPA)

This model allows to determine the similarity transformation between two matrices, permitting a different weighing for each point (by rows), for each component (by columns) or for both. The first formulation was proposed by [61] and the solution is presented below, following the derivation reported in [11]. Given a $p \times p$ weight matrix $\mathbf{P}_R^2$ and a $k \times k$ weight matrix $\mathbf{P}_C^2$, the unknown parameters $\mathbf{R}$ (rotation), $\mathbf{t}$ (translation) and c (isotropic scale factor) are searched so to satisfy the following minimum condition:

$$\mathrm{tr}\left[(\mathbf{B} - c\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}')'\mathbf{P}_R^2(\mathbf{B} - c\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}')\mathbf{P}_C^2\right] = \min \tag{2.32}$$

under the orthogonality constraint $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$. Please note that $\mathbf{P}_R^2$ weights the points, while $\mathbf{P}_C^2$ weights the components. We shall allow $\mathbf{P}_R^2$ and $\mathbf{P}_C^2$ to be any square matrix, but in most applications they will be diagonal [67].
Let us separately consider the different situations, according to the value assumed by the weight matrices.

First case: Point weighing ($\mathbf{P}_R^2 \neq \mathbf{I}$ and $\mathbf{P}_C^2 = \mathbf{I}$) .

If $\mathbf{P}_R^2 \neq \mathbf{I}$ is real, symmetric and positive definite, the minimum condition (2.32) corresponds to the minimum of the Lagrangian function:

$$\begin{aligned} F =& \operatorname{tr}\left[\mathbf{B}'\mathbf{P}_R^2\mathbf{B} + c^2\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{A}\mathbf{R} + \mathbf{t}\mathbf{j}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}' - 2c\mathbf{B}'\mathbf{P}_R^2\mathbf{A}\mathbf{R} - 2\mathbf{B}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}' + \right. \\ & \left. + 2c\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}'\right] + \operatorname{tr}\left[\mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\right]. \end{aligned} \tag{2.33}$$

Imposing the partial derivatives of F with respect to $\mathbf{R}$, c and $\mathbf{t}$ equal to zero, it follows:

$$\frac{\partial F}{\partial \mathbf{R}} = 2c^2\mathbf{A}'\mathbf{P}_R^2\mathbf{A}\mathbf{R} - 2c\mathbf{A}'\mathbf{P}_R^2\mathbf{B} + 2c\mathbf{A}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}' + \mathbf{R}\left(\mathbf{L} + \mathbf{L}'\right) = 0 \tag{2.34}$$

$$\frac{\partial F}{\partial \mathbf{t}} = 2\mathbf{t}\mathbf{j}'\mathbf{P}_R^2\mathbf{j} + 2c\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{j} - 2\mathbf{B}'\mathbf{P}_R^2\mathbf{j} = 0 \tag{2.35}$$

$$\frac{\partial F}{\partial c} = 2c\operatorname{tr}\left(\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{A}\mathbf{R}\right) - 2\operatorname{tr}\left(\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{B} - \mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}'\right) = 0 \tag{2.36}$$

and from (2.35) it results:

$$\mathbf{t} = \left(\mathbf{B} - c\mathbf{A}\mathbf{R}\right)'\frac{\mathbf{P}_R^2\mathbf{j}}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}. \tag{2.37}$$

From (2.34), after having multiplied on the left all the terms by $\mathbf{R}'$ and observing that $\mathbf{R}'\mathbf{R}\left(\mathbf{L} + \mathbf{L}'\right)$ and $\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{A}\mathbf{R}$ are symmetric, it follows that also $\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{B} - \mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{j}\mathbf{t}'$ must be symmetric.
Substituting the expression of $\mathbf{t}$ given by (2.37), one gets[1]:

$$\operatorname{sym}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{B} - \mathbf{R}'\mathbf{A}'\frac{\mathbf{P}_R^2\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\mathbf{B} + c\mathbf{R}'\mathbf{A}'\frac{\mathbf{P}_R^2\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\mathbf{A}\mathbf{R}\right]. \tag{2.38}$$

Furthermore, observing that the term

$$\mathbf{R}'\mathbf{A}'\frac{\mathbf{P}_R^2\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\mathbf{A}\mathbf{R} \tag{2.39}$$

in Equation (2.38) is also symmetric, it is possible to conclude that both terms of the equation:

$$\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\mathbf{B} - \mathbf{R}'\mathbf{A}'\frac{\mathbf{P}_R^2\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\mathbf{B} = \mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{B} \tag{2.40}$$

must be symmetric.
Defining

$$\mathbf{S} = \mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{B} \tag{2.41}$$

one reaches the same condition of (2.7) and (2.23). Being

$$\mathbf{S} = \mathbf{V}\mathbf{D}_s\mathbf{W}' \tag{2.42}$$

[1]The predicate sym[ ] is true when the argument is a symmetric matrix.

the SVD of matrix $\mathbf{S}$, as for Equations (2.10) and (2.25), the rotation is given by:

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}\,(1, 1, \det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}'. \tag{2.43}$$

For what concerns the scale factor c, substituting in (2.36) the expression of $\mathbf{t}$ given by (2.37), it is possible to obtain:

$$c = \frac{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{B}\right]}{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{A}\mathbf{R}\right]}. \tag{2.44}$$

Summarizing, given matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{P}_R^2$, matrix $\mathbf{S}$ is computed by means of (2.41) and the SVD of $\mathbf{S}$ is carried out to directly obtain $\mathbf{R}$. The scale factor c is then computed by (2.44) and finally $\mathbf{t}$ is derived by means of (2.37).
Being $\mathbf{P}_R^2$ symmetric and positive definite, it admits the Cholesky decomposition $\mathbf{P}_R^2 = \mathbf{P}_R{}'\mathbf{P}_R$. It is then useful to observe that the WEOPA model is nothing but the EOPA model applied to the matrices $\mathbf{A}$ and $\mathbf{B}$ "extended" by the component $\mathbf{P}_R$ of the weight matrix $\mathbf{P}_R^2$, namely $\mathbf{P}_R\mathbf{A}$ and $\mathbf{P}_R\mathbf{B}$. This can be proven substituting matrix $\mathbf{A}$ with $\mathbf{P}_R\mathbf{A}$, matrix $\mathbf{B}$ with $\mathbf{P}_R\mathbf{B}$ and $\mathbf{j}$ with $\mathbf{P}_R\mathbf{j}$ in the formulas of the EOPA model, obtaining the same expressions (2.37), (2.41) and (2.44) derived up to now. In fact, the expression of the translation $\mathbf{t}$ of the EOPA model, given by (2.19), becomes:

$$\mathbf{t} = (\mathbf{P}_R\mathbf{B} - c\mathbf{P}_R\mathbf{A}\mathbf{R})'\frac{\mathbf{P}_R\mathbf{j}}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}} = (\mathbf{B} - c\mathbf{A}\mathbf{R})'\frac{\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}} = (\mathbf{B} - c\mathbf{A}\mathbf{R})'\frac{\mathbf{P}_R^2\mathbf{j}}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}$$

while matrix $\mathbf{S}$ of the EOPA model (given by (2.22)), from which the components of $\mathbf{R}$ are obtained, results from:

$$\mathbf{S} = \mathbf{A}'\mathbf{P}_R{}'\left(\mathbf{I} - \frac{\mathbf{P}_R\mathbf{j}\mathbf{j}'\mathbf{P}_R{}'}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}}\right)\mathbf{P}_R\mathbf{B} = \mathbf{A}'\mathbf{P}_R{}'\mathbf{P}_R\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}}\right)\mathbf{B} = \mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{B}.$$

Finally, the expression of the scale factor c, computed from Equation (2.26) of the EOPA model, is:

$$c = \frac{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R{}'\left(\mathbf{I} - \frac{\mathbf{P}_R\mathbf{j}\mathbf{j}'\mathbf{P}_R{}'}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}}\right)\mathbf{P}_R\mathbf{B}\right]}{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R{}'\left(\mathbf{I} - \frac{\mathbf{P}_R\mathbf{j}\mathbf{j}'\mathbf{P}_R{}'}{\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}}\right)\mathbf{P}_R\mathbf{A}\mathbf{R}\right]} = \frac{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{B}\right]}{\mathrm{tr}\left[\mathbf{R}'\mathbf{A}'\mathbf{P}_R^2\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'\mathbf{P}_R^2}{\mathbf{j}'\mathbf{P}_R^2\mathbf{j}}\right)\mathbf{A}\mathbf{R}\right]}.$$

Second case: Component weighing ($\mathbf{P}_R^2 = \mathbf{I}$ and $\mathbf{P}_C^2 \neq \mathbf{I}$) .

In this case, no algebraic direct solutions are known in the literature [91, 67]. In fact, starting from Equation (2.32), one can demonstrate that matrix $\mathbf{R}$ can be directly computed via a SVD decomposition only if

$$\mathbf{R}\mathbf{P}_C^2\mathbf{R}' = \bar{\mathbf{R}}'\bar{\mathbf{R}} = \mathbf{I} \tag{2.45}$$

that is satisfied when

$$\mathbf{P}_C^2 = \mathbf{I} \tag{2.46}$$

in contrast to the initial assumption $\mathbf{P}_C^2 \neq \mathbf{I}$. Therefore, such a problem can be faced only by adopting an iterative algorithm like the one proposed in [91].

2.3.4 Anisotropic Extended Orthogonal Procrustes Analysis (AEOPA)

The EOPA model, described in Section 2.3.2, can be further extended by substituting the isotropic scale factor c with an anisotropic scaling characterized by a diagonal matrix $\mathbf{\Gamma}$ of

different scale factors [66]. *Anisotropic Extended Orthogonal Procrustes Analysis* (AEOPA) comes in – at least – three flavors, which implies the minimization of different cost functions. Following the pre-scaling on the variables (or space dimension) approach [66, 15], the model can be formulated as:

$$\min\|\mathbf{B} - \mathbf{A}\boldsymbol{\Gamma}\mathbf{R} - \mathbf{j}\mathbf{t}'\|_F^2 \tag{2.47}$$

whereas the post-scaling on the variables leads to:

$$\min\|\mathbf{B} - \mathbf{A}\mathbf{R}'\boldsymbol{\Gamma} - \mathbf{j}\mathbf{t}'\|_F^2 \tag{2.48}$$

both subject to $\mathbf{R}'\mathbf{R} = \mathbf{I}$. These models find application in data analysis, where the columns of $\mathbf{A}$ are variables that can be independently scaled, and in shape analysis, constituting a solution to problems where the transformation of a configuration is nonlinear [62].
Finally, data (or row) scaling can be considered, where each data point or measurement is scaled independently of the others [57]. The model, defined by [67] as Anisotropic EOPA *with row scaling*, can be formulated as:

$$\min\|\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}'\|_F^2 \text{ subject to } \mathbf{R}'\mathbf{R} = \mathbf{I}. \tag{2.49}$$

In the next paragraphs, we will focus the attention on this last case, that allows to solve in an innovative and efficient way a classical photogrammetric problem, i.e. the image exterior orientation. We will report the derivation of its general solution following the procedure proposed in [57].
To obtain the least squares solution for model (2.49), let us make explicit the residual matrix $\mathbf{E}$:

$$\mathbf{B} = \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} + \mathbf{j}\mathbf{t}' + \mathbf{E} \tag{2.50}$$

where $\boldsymbol{\Gamma}$ is an unknown $p \times p$ diagonal matrix. So Equation (2.49) can be written as

$$\min\|\mathbf{E}\|_F^2 = \min \operatorname{tr}(\mathbf{E}'\mathbf{E}) \text{ subject to } \mathbf{R}'\mathbf{R} = \mathbf{I}. \tag{2.51}$$

The problem is equivalent to the minimization of the Lagrangian function:

$$F = \operatorname{tr}(\mathbf{E}'\mathbf{E}) + \operatorname{tr}(\mathbf{L}(\mathbf{R}'\mathbf{R} - \mathbf{I})) \tag{2.52}$$

where $\mathbf{L}$ is the matrix of Lagrangian multipliers. This can be solved by setting to zero the partial derivatives of F with respect to the unknowns $\mathbf{R}$, $\mathbf{t}$ and the diagonal matrix $\boldsymbol{\Gamma}$.
Let us start from (2.52), and substitute (2.50) for $\mathbf{E}$:

$$\begin{aligned} F = {} & \operatorname{tr}(\mathbf{B}'\mathbf{B}) + \operatorname{tr}(\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\boldsymbol{\Gamma}\mathbf{A}\mathbf{R}) + p\operatorname{tr}(\mathbf{t}\mathbf{t}') - \\ & - 2\operatorname{tr}(\mathbf{B}'\mathbf{j}\mathbf{t}') - 2\operatorname{tr}(\mathbf{B}'\boldsymbol{\Gamma}\mathbf{A}\mathbf{R}) + \\ & + 2\operatorname{tr}(\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{j}\mathbf{t}') + \operatorname{tr}(\mathbf{L}(\mathbf{R}'\mathbf{R} - \mathbf{I})). \end{aligned} \tag{2.53}$$

The translation vector $\mathbf{t}$ can be obtained by equating to zero the partial derivative:

$$\frac{\partial F}{\partial \mathbf{t}} = 2p\mathbf{t} - 2\mathbf{B}'\mathbf{j} + 2\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{j} = 0. \tag{2.54}$$

Hence:

$$\mathbf{t} = (\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R})'\mathbf{j}/p. \tag{2.55}$$

Once the derivative of F with respect to $\mathbf{R}$ is set to zero, it results:

$$\frac{\partial F}{\partial \mathbf{R}} = \mathbf{A}'\boldsymbol{\Gamma}'\boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{A}'\boldsymbol{\Gamma}'\mathbf{B} + \mathbf{A}'\boldsymbol{\Gamma}'\mathbf{j}\mathbf{t}' + \mathbf{R}(\mathbf{L} + \mathbf{L}')/2 = 0. \tag{2.56}$$

Let us multiply (2.56) on the left by $\mathbf{R}'$:

$$\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{B} + \mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{j}\mathbf{t}' + \mathbf{R}'\mathbf{R}(\mathbf{L} + \mathbf{L}')/2 = 0. \tag{2.57}$$

Since matrices $\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{A}\mathbf{R}$ and $(\mathbf{L}+\mathbf{L}')/2$ are symmetric, then

$$\text{sym}\,[\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{B}-\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{j}\mathbf{t}']. \tag{2.58}$$

Substituting (2.55) in (2.58), it results

$$\text{sym}\,[\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{B}-\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{j}\mathbf{j}'/p)\,(\mathbf{B}-\mathbf{\Gamma}\mathbf{A}\mathbf{R})\,] \tag{2.59}$$

which is equivalent to

$$\text{sym}[\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{B}-\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{j}\mathbf{j}'/p)\,\mathbf{B}+\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{j}\mathbf{j}'/p)\,\mathbf{\Gamma}\mathbf{A}\mathbf{R}\,] \tag{2.60}$$

and finally:

$$\text{sym}[\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{I}-\mathbf{j}\mathbf{j}'/p)\,\mathbf{B}+\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{j}\mathbf{j}'/p)\,\mathbf{\Gamma}\mathbf{A}\mathbf{R}\,]. \tag{2.61}$$

Since $\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{j}\mathbf{j}'/p)\,\mathbf{\Gamma}\mathbf{A}\mathbf{R}$ is symmetric, also the first term must be symmetric, i.e.,

$$\text{sym}\,[\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{I}-\mathbf{j}\mathbf{j}'/p)\,\mathbf{B}]. \tag{2.62}$$

Let us define the matrix $\mathbf{S}$ equal to

$$\mathbf{S}=\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{I}-\mathbf{j}\mathbf{j}'/p)\,\mathbf{B} \tag{2.63}$$

Matrix $\mathbf{R}'\mathbf{S}$ is symmetric, therefore the following condition must be satisfied

$$\mathbf{R}'\mathbf{S}=\mathbf{S}'\mathbf{R} \tag{2.64}$$

that leads to the usual solution $\mathbf{R}=\mathbf{V}\mathbf{W}'$, with $\mathbf{S}=\mathbf{V}\mathbf{D}_S\mathbf{W}'$ the SVD of $\mathbf{S}$. In order to guarantee that $\mathbf{R}$ is not only orthogonal but has positive determinant [156], we shall use:

$$\mathbf{R}=\mathbf{V}\,\text{diag}\,(1,1,\det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}'. \tag{2.65}$$

The least squares solution for the diagonal matrix $\mathbf{\Gamma}$ can be retrieved by first computing the partial derivatives of (2.53) with respect to $\mathbf{\Gamma}$:

$$\begin{aligned}\frac{\partial F}{\partial \mathbf{\Gamma}}&=\frac{\partial}{\partial \mathbf{\Gamma}}\,\text{tr}\,(\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{A}\mathbf{R})-2\frac{\partial}{\partial \mathbf{\Gamma}}\,\text{tr}\,(\mathbf{B}'\mathbf{\Gamma}\mathbf{A}\mathbf{R})+2\frac{\partial}{\partial \mathbf{\Gamma}}\,\text{tr}\,(\mathbf{R}'\mathbf{A}'\mathbf{\Gamma}'\mathbf{j}\mathbf{t}')\\ \frac{\partial F}{\partial \mathbf{\Gamma}}&=2\mathbf{\Gamma}\mathbf{A}\mathbf{A}'-2\mathbf{A}\mathbf{R}\mathbf{B}'+2\mathbf{A}\mathbf{R}\mathbf{t}\mathbf{j}'\end{aligned} \tag{2.66}$$

and then setting the derivatives to zero, obtaining:

$$\mathbf{\Gamma}\mathbf{A}\mathbf{A}'=\mathbf{A}\mathbf{R}\,(\mathbf{B}'-\mathbf{t}\mathbf{j}') \tag{2.67}$$

and then

$$\mathbf{\Gamma}=(\mathbf{A}\mathbf{A}'\odot\mathbf{I})^{-1}\,(\mathbf{A}\mathbf{R}(\mathbf{B}'-\mathbf{t}\mathbf{j}')\odot\mathbf{I}) \tag{2.68}$$

where $\odot$ is the Hadamard (or element-wise) product.
We now demonstrate that Expression (2.68) for matrix $\mathbf{\Gamma}$ can be derived also in an alternative and more intuitive way, without resorting to the partial derivative of F with respect to $\mathbf{\Gamma}$. Assuming that $\mathbf{R}$ and $\mathbf{t}$ are known and defining $\mathbf{U}=\mathbf{R}\,(\mathbf{B}'-\mathbf{t}\mathbf{j}')$, Procrustes model $\mathbf{B}=\mathbf{\Gamma}\mathbf{A}\mathbf{R}+\mathbf{j}\mathbf{t}'$ becomes

$$\mathbf{A}'\mathbf{\Gamma}=\mathbf{U}. \tag{2.69}$$

Exploiting the Khatri-Rao product, denoted with $\circ$, and its property involving diagonal matrices and the vec operator[2], Equation (2.69) can be written as

$$(\mathbf{I}\circ\mathbf{A})\,\text{diag}^{-1}(\mathbf{\Gamma})=\text{vec}\,\mathbf{U} \tag{2.70}$$

[2]The vec operator transforms a matrix into a vector by stacking its columns.

where diag^{-1} returns a vector containing the diagonal elements of its argument. In the over-determined case, the least squares solution of (2.70) is given by

$$\mathrm{diag}^{-1}(\mathbf{\Gamma}) = \left[(\mathbf{I} \odot \mathbf{A})' (\mathbf{I} \circ \mathbf{A})\right]^{-1} (\mathbf{I} \circ \mathbf{A})' \,\mathrm{vec}\,\mathbf{U} \tag{2.71}$$

which is equivalent to the following more compact formulation applying the Hadamard product [99]:

$$\mathrm{diag}^{-1}(\mathbf{\Gamma}) = (\mathbf{I} \odot \mathbf{A}\mathbf{A}')^{-1} \,\mathrm{diag}^{-1}(\mathbf{A}\mathbf{U}). \tag{2.72}$$

It is easy to see that (2.72) corresponds to the original solution (2.68).
Please note that, whereas in the classical solution of the EOPA problem (Section 2.3.2) one can recover first $\mathbf{R}$, that does not depend on the other unknowns, then the isotropic scale (that depends only on $\mathbf{R}$) and finally $\mathbf{t}$, in the anisotropic case the unknowns are entangled in such a way that one must resort to the so called *block relaxation* scheme [42] (also known as *alternating least squares*), where each variable is alternatively estimated while keeping the others fixed. The algorithm can be summarized as:

1. Assuming known $\mathbf{\Gamma}$, find $\mathbf{R} = \mathbf{V}\,\mathrm{diag}\,(1, 1, \det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}'$, with $\mathbf{A}'\mathbf{\Gamma}'\,(\mathbf{I} - \mathbf{j}\mathbf{j}'/p)\,\mathbf{B} = \mathbf{V}\mathbf{D}_S\mathbf{W}'$, and $\mathbf{t} = (\mathbf{B} - \mathbf{\Gamma}\mathbf{A}\mathbf{R})'\mathbf{j}/p$;
2. Given $\mathbf{R}$ and $\mathbf{t}$, solve for $\mathbf{\Gamma} = (\mathbf{A}\mathbf{A}' \odot \mathbf{I})^{-1}\,(\mathbf{A}\mathbf{R}(\mathbf{B}' - \mathbf{t}\mathbf{j}') \odot \mathbf{I})$;
3. Iterate over steps 1. and 2. until convergence.

2.4 Errors-In-Variables Model And Total Least Squares Solutions

As shown in the previous sections, given a matrix $\mathbf{A}$ (origin) and a matrix $\mathbf{B}$ (destination), containing the coordinates of p points in $\mathbb{R}^k$, classical least squares (LS) Procrustes solutions find the transformation parameters between the two point sets assuming that all random errors are confined to the destination matrix $\mathbf{B}$, whereas $\mathbf{A}$ is noise-free. However, this assumption is often unrealistic, since both $\mathbf{A}$ and $\mathbf{B}$ are corrupted by errors if they derive from measurements. Therefore, it seems appropriate to introduce the *Errors-In-Variables* (EIV) model, which is a more general model wherein both matrices $\mathbf{A}$ and $\mathbf{B}$ are assumed to be contaminated by errors. The problem of parameters estimation in the EIV model is often called the total least squares (TLS) problem [59].
The first TLS solutions were developed in literature to solve linear problems expressed by the following EIV model:

$$\mathbf{y} + \mathbf{e}_\mathbf{y} = (\mathbf{\Pi} + \mathbf{E}_\mathbf{\Pi})\,\mathbf{x} \tag{2.73}$$

where $\mathbf{e}_\mathbf{y}$ and $\mathbf{E}_\mathbf{\Pi}$ are the error vector of observations $\mathbf{y}$ and the error matrix of the design matrix $\mathbf{\Pi}$, respectively, and $\mathbf{x}$ is the vector of unknown parameters. Golub and Van Loan [59] introduced in mathematical literature the total least squares method applied to the EIV model (2.73) to treat regression problems where all the data are affected by random errors. The solution they developed was based on a SVD approach.
In geoinformatics, one of the first applications of the TLS method was described in [50]. The authors developed a Structured Total Least Squares (STLS) algorithm to solve a planar linear conformal transformation with a particular structure of the coefficient matrix $\mathbf{\Pi}$. Moreover, Shaffrin and Felus [131] proposed a method based on the nonlinear Euler-Lagrange condition equations for estimating a planar affine transformation by multivariate TLS problem (MTLS). Results showed that the differences between the total least squares and the classical least squares estimated parameters are small; nevertheless they can affect significantly the final accuracy of the transformed coordinates.
Another issue that has received considerable attention in recent years is the weighing of observations according to their accuracy. Mahboub [111] developed an algorithm for the

weighted TLS problem that allows for a general weight matrix, preserving at the same time the structure of the design matrix $\mathbf{\Pi}$.
Moreover, the EIV-model can also be considered as a nonlinear Gauss-Helmert model (GHM), since both minimize the same objective function. Amiri-Simkooei and Jazaeri [3], who formulated a simple solution to the weighted TLS problem as an extension of the standard least squares solution, showed that the results obtained with their algorithm are identical to those provided by a rigorous nonlinear GHM. However, the simplicity of the TLS solution avoids the critical issues that must be handled when solving the nonlinear GHM.
Furthermore, in photogrammetric and geomatics applications, constraints on the unknown parameters are often introduced, e.g., to take into account prior knowledge such as orthogonality condition. Schaffrin [130] solved a TLS problem with linear constraints, while in [132] a procedure to treat linear and quadratic constraints has been proposed. The TLS solution of [48] can be used instead with arbitrary applicable constraints.
In any case, all these methods are based on the linear EIV model (2.73), that requires to linearize condition equations in order to compute the parameters of the transformation. The Procrustes solutions that will be presented in the next sections are closed-form ones, and no linearization nor approximate parameter values are needed.

2.5 Total Least Squares Solutions of Procrustes Models

Arun [8] was the first to deal with the TLS solution of the orthogonal Procrustes problem, proving that for rigid transformations the classical LS Procrustes solution coincides with the TLS one. Felus and Burtch [49] started instead from the EIV-Extended Orthogonal Procrustes model (EIV-EOPA) to compute the unknown parameters of a similarity transformation (rotation matrix, translation vector and scale factor), showing that, in this case, LS and TLS are not coincident. In the following, we will report the derivation of these solutions, highlighting the differences with respect to the LS ones. In Section 2.5.2 we will extend the TLS solution of the weighted model (EIV-WEOPA) presented in [49], allowing a different weighing of the two coordinate matrices. Moreover, we will develop in Section 2.5.3 a new TLS solution of the Anisotropic Extended Orthogonal Procrustes model (EIV-AEOPA).
First of all, let us recall the EOPA model introduced in Section 2.3.2, calling for clarity reasons the error matrix affecting the destination set as $\mathbf{E_B}$:

$$c\mathbf{AR} + \mathbf{jt}' = \mathbf{B} + \mathbf{E_B}. \tag{2.74}$$

The LS Procrustes solution allows to directly estimate the unknown rotation matrix $\mathbf{R}$, the translation vector $\mathbf{t}$ and the global scale factor c minimizing the square of the Frobenius norm of the residual matrix $\mathbf{E_B}$, i.e.

$$\min \|\mathbf{E_B}\|_F^2 = \min \operatorname{tr}(\mathbf{E_B}\mathbf{E_B}') \tag{2.75}$$

under the orthogonality condition $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$. In order to take into account that also matrix $\mathbf{A}$ can be noisy, (2.74) should be replaced by the EIV model expressed as follows:

$$c(\mathbf{A} + \mathbf{E_A})\mathbf{R} + \mathbf{jt}' = \mathbf{B} + \mathbf{E_B} \tag{2.76}$$

where matrix $\mathbf{E_A}$ represents the errors affecting the coordinates contained in matrix $\mathbf{A}$. The unknowns are then computed by minimizing the square of the Frobenius norm of the matrix $(\mathbf{E_A}|\mathbf{E_B})$, i.e.

$$\min\left(\|\mathbf{E_A}\|_F^2 + \|\mathbf{E_B}\|_F^2\right) = \min \operatorname{tr}\left(\left[\mathbf{E_A}|\mathbf{E_B}\right]\begin{bmatrix}\mathbf{E_A}'\\ \mathbf{E_B}'\end{bmatrix}\right) \tag{2.77}$$

under the constraint that matrix $\mathbf{R}$ is orthogonal. Let us now consider a rigid transformation, where the only unknown parameters are the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$.

As shown in [8], the solution to the EIV model (2.76) with c=1 is the same as the solution to the orthogonal Procrustes problem (2.74). In fact, solving problem (2.77) with c=1 leads to the cost function:

$$F = \mathrm{tr}\left[\frac{1}{2}\Big(\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R} + \mathbf{B}'\mathbf{B} + \mathbf{t}\mathbf{j}'\mathbf{j}\mathbf{t}' + 2\mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{A}\mathbf{R}\Big) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\right] \quad (2.78)$$

where $\mathbf{L}$ is a diagonal matrix of unknown Lagrangian multipliers. Considering instead the classical orthogonal Procrustes problem (2.75) with c=1, the cost function F is

$$F = \mathrm{tr}\left(\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R} + \mathbf{B}'\mathbf{B} + \mathbf{t}\mathbf{j}'\mathbf{j}\mathbf{t}' + 2\mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{A}\mathbf{R} + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\right). \quad (2.79)$$

The two cost functions (2.78) and (2.79) are identical, except for the constant value 1/2, and their minimization leads then to the same solution. Therefore, the Procrustes solution for a rigid transformation (c=1) also solves the EIV model. However, this property is not valid for similarity transformations ($c\neq1$), as we will show in the next section.

2.5.1 EIV-Extended Orthogonal Procrustes Analysis (EIV-EOPA)

The original solution of the EIV-EOPA model is due to Felus and Burtch [49]. In the following, we retrieve the same TLS solution in a different way, first rewriting Equation (2.76) as

$$\left[c\mathbf{R}'|-\mathbf{I}\right]\begin{bmatrix}\mathbf{E_A}'\\ \mathbf{E_B}'\end{bmatrix} = -\left[c\mathbf{R}'|-\mathbf{I}\right]\begin{bmatrix}\mathbf{A}'\\ \mathbf{B}'-\mathbf{t}\mathbf{j}'\end{bmatrix} \quad (2.80)$$

and then exploiting the minimum norm solution to the above system, namely

$$\begin{aligned}\left[\mathbf{E_A}|\mathbf{E_B}\right] &= -\frac{1}{c^2+1}\left[\mathbf{A}|\left(\mathbf{B}-\mathbf{j}\mathbf{t}'\right)\right]\begin{bmatrix}c^2 & -c\mathbf{R}\\ -c\mathbf{R}' & \mathbf{I}\end{bmatrix}\\ &= -\frac{1}{c^2+1}\left[c^2\mathbf{A}-c\left(\mathbf{B}-\mathbf{j}\mathbf{t}'\right)\mathbf{R}'|-c\mathbf{A}\mathbf{R}+\left(\mathbf{B}-\mathbf{j}\mathbf{t}'\right)\right]\end{aligned} \quad (2.81)$$

which has been computed taking into account the orthogonality condition $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$. Starting from Expression (2.81), the cost F as a function of $\mathbf{R}$, $\mathbf{t}$ and c to be minimized to solve (2.77) is therefore

$$\begin{aligned}F =& \,\mathrm{tr}\left(\left[\mathbf{E_A}|\mathbf{E_B}\right]\begin{bmatrix}\mathbf{E_A}'\\ \mathbf{E_B}'\end{bmatrix}\right) + \mathrm{tr}\left(\mathbf{L}\left(\mathbf{R}'\mathbf{R}-\mathbf{I}\right)\right)\\ =& \,\mathrm{tr}\left(\frac{1}{\left(c^2+1\right)^2}\left[c^2\mathbf{A}-c\left(\mathbf{B}-\mathbf{j}\mathbf{t}'\right)\mathbf{R}'|-c\mathbf{A}\mathbf{R}+\left(\mathbf{B}-\mathbf{j}\mathbf{t}'\right)\right]\right.\\ &\left.\cdot\begin{bmatrix}c^2\mathbf{A}'-c\mathbf{R}\left(\mathbf{B}'-\mathbf{t}\mathbf{j}'\right)\\ -c\mathbf{R}'\mathbf{A}'+\left(\mathbf{B}'-\mathbf{t}\mathbf{j}'\right)\end{bmatrix}\right) + \mathrm{tr}\left(\mathbf{L}\left(\mathbf{R}'\mathbf{R}-\mathbf{I}\right)\right).\end{aligned} \quad (2.82)$$

Using the properties of the trace, it follows

$$\begin{aligned}F = \mathrm{tr}\left[\frac{1}{c^2+1}\Big(c^2\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R} + \mathbf{B}'\mathbf{B} + \mathbf{t}\mathbf{j}'\mathbf{j}\mathbf{t}'\right.\\ \left.+ 2c\mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{j}\mathbf{t}' - 2c\mathbf{B}'\mathbf{A}\mathbf{R}\Big) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\right].\end{aligned} \quad (2.83)$$

The translation vector $\mathbf{t}$ can be obtained by equating to zero the partial derivative:

$$\frac{\partial F}{\partial \mathbf{t}} = \frac{1}{c^2+1}\left[2\mathbf{j}'\mathbf{j}\mathbf{t} + 2c\mathbf{R}'\mathbf{A}'\mathbf{j} - 2\mathbf{B}'\mathbf{j}\right] = 0 \quad (2.84)$$

from which it results

$$\mathbf{t} = (\mathbf{B}' - c\mathbf{R}'\mathbf{A}')\,\frac{\mathbf{j}}{\mathbf{j}'\mathbf{j}}. \tag{2.85}$$

Setting to zero the derivative of F with respect to $\mathbf{R}$, it follows

$$\frac{\partial F}{\partial \mathbf{R}} = \frac{1}{c^2+1}\left[2c^2\mathbf{A}'\mathbf{A}\mathbf{R} + 2c\mathbf{A}'\mathbf{t}\mathbf{j}' - 2c\mathbf{A}'\mathbf{B}\right] + \mathbf{R}(\mathbf{L}+\mathbf{L}') = 0. \tag{2.86}$$

Let us multiply (2.86) on the left by $\mathbf{R}'$:

$$c\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R} + \mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' - \mathbf{R}'\mathbf{A}'\mathbf{B} + \mathbf{R}'\mathbf{R}\boldsymbol{\Theta} = 0 \tag{2.87}$$

where $\boldsymbol{\Theta} = (c^2+1)(\mathbf{L}+\mathbf{L}')/2$. Observing that matrices $\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R}$ and $\mathbf{R}'\mathbf{R}\boldsymbol{\Theta}$ are symmetric, then

$$\operatorname{sym}\left[-\mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' + \mathbf{R}'\mathbf{A}'\mathbf{B}\right]. \tag{2.88}$$

Substituting (2.85) in (2.88), it results

$$\operatorname{sym}\left[\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B} + c\mathbf{R}'\mathbf{A}'\frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\mathbf{A}\mathbf{R}\right]. \tag{2.89}$$

Since $c\mathbf{R}'\mathbf{A}'\dfrac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\mathbf{A}\mathbf{R}$ is symmetric, it must be

$$\operatorname{sym}\left[\mathbf{R}'\mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B}\right]. \tag{2.90}$$

As already discussed in Section 2.3.2, matrix $\mathbf{I} - \mathbf{j}\mathbf{j}'/(\mathbf{j}'\mathbf{j})$ is symmetric and idempotent and it translates the matrix values to which it is applied to the corresponding barycenter. Let us compute the intermediate matrix $\bar{\mathbf{A}}$

$$\bar{\mathbf{A}} = \mathbf{A}'\left(\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right) \tag{2.91}$$

that is the transposed barycentric matrix $\mathbf{A}$. Calling

$$\mathbf{S} = \bar{\mathbf{A}}'\bar{\mathbf{B}} \tag{2.92}$$

with $\bar{\mathbf{B}} = \left(\mathbf{I} - \dfrac{\mathbf{j}\mathbf{j}'}{\mathbf{j}'\mathbf{j}}\right)\mathbf{B}$ the barycentric matrix $\mathbf{B}$, from (2.90) it follows that

$$\mathbf{R}'\mathbf{S} = (\mathbf{R}'\mathbf{S})' = \mathbf{S}'\mathbf{R} \tag{2.93}$$

and finally

$$\mathbf{R} = \mathbf{V}\operatorname{diag}(1, 1, \det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}' \tag{2.94}$$

where $\mathbf{S} = \mathbf{V}\mathbf{D}_{\mathbf{S}}\mathbf{W}'$ is the SVD of $\mathbf{S}$ and the term $\operatorname{diag}(1, 1, \det(\mathbf{V}\mathbf{W}'))$ guarantees that $\mathbf{R}$ has positive determinant. In order to compute the scale factor c, let us substitute (2.85) in (2.83). After some rewriting, the cost function F becomes

$$F = \operatorname{tr}\left[\frac{1}{c^2+1}\left(c^2\bar{\mathbf{A}}'\bar{\mathbf{A}} + \bar{\mathbf{B}}'\bar{\mathbf{B}} - 2c\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\right]. \tag{2.95}$$

Setting to zero the partial derivatives of (2.95) with respect to c, one obtains

$$\operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right)c^2 + \operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}} - \bar{\mathbf{B}}'\bar{\mathbf{B}}\right)c - \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) = 0. \tag{2.96}$$

Analyzing Equation (2.96), it can be noticed that there are two real solutions, one positive and one negative. The searched value of the scale factor c is the positive one computed from the expression:

$$c_{1,2} = \frac{-\operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}} - \bar{\mathbf{B}}'\bar{\mathbf{B}}\right) \pm \sqrt{\Delta}}{2\operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right)} \tag{2.97}$$

$$\Delta = \operatorname{tr}^2\left(\bar{\mathbf{A}}'\bar{\mathbf{A}} - \bar{\mathbf{B}}'\bar{\mathbf{B}}\right) + 4\operatorname{tr}^2\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right). \tag{2.98}$$

Interestingly, these results correspond to the ones obtained in [26] with an alternative derivation. In fact, [26] found a closed-form least squares solution for the 3D similarity transformation problem under a Gauss-Helmert model with equal weights, that is equivalent to the EIV-EOPA solution.
Comparing this solution with the results deriving from the classical EOPA (Section 2.3.2), it is easy to see that the the scale factor c – and consequently the translation vector $\mathbf{t}$ – resulting from the EOPA is different from the one derived with the EIV-EOPA. In fact, (2.30) is not a root of (2.96), unless c=1. On the contrary, the rotation matrix $\mathbf{R}$ is the same.

2.5.2 EIV-Weighted Extended Orthogonal Procrustes Analysis (EIV-WEOPA)

The EIV-EOPA can be augmented by weighing the coordinate matrices in two different ways: by rows or by columns. Let us first consider the weighing by rows, which is applied to the case when distinct points have different accuracy. From the analytical point of view, this corresponds to minimizing the following function:

$$F = \operatorname{tr}\left(\mathbf{P}_R\left[\mathbf{E_A}|\mathbf{E_B}\right]\begin{bmatrix}\mathbf{E_A}'\\ \mathbf{E_B}'\end{bmatrix}\mathbf{P}_R{}'\right) + \operatorname{tr}\left[\mathbf{L}\left(\mathbf{R}'\mathbf{R}-\mathbf{I}\right)\right] \tag{2.99}$$

where $\mathbf{P}_R$ is a $p \times p$ diagonal weight matrix and, after some rewriting

$$\begin{aligned} F = \operatorname{tr}\Bigg[\frac{1}{c^2+1}\big(&c^2\mathbf{R}'\mathbf{A}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{A}\mathbf{R} + \mathbf{P}_R\mathbf{B}\mathbf{B}'\mathbf{P}_R{}' + \mathbf{t}\mathbf{j}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}\mathbf{t}' \\ &+ 2c\mathbf{R}'\mathbf{A}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{j}\mathbf{t}' - 2c\mathbf{B}'\mathbf{P}_R{}'\mathbf{P}_R\mathbf{A}\mathbf{R}\big) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L}\Bigg]. \end{aligned} \tag{2.100}$$

One can notice that (2.100) can be derived also by substituting matrix $\mathbf{A}$ with $\mathbf{P}_R\mathbf{A}$, matrix $\mathbf{B}$ with $\mathbf{P}_R\mathbf{B}$ and $\mathbf{j}$ with $\mathbf{P}_R\mathbf{j}$ in (2.83). Therefore the solution to the EIV-Weighted Orthogonal Procrustes Analysis model, corresponding to the minimization of (2.99), is nothing but the EIV-EOPA solution to which the aforementioned substitutions have been applied.
Coordinate matrix can be weighted also by columns, that corresponds to assigning a different accuracy to the point coordinate components. The function to minimize is

$$F = \operatorname{tr}\left(\left[\mathbf{E_A}|\mathbf{E_B}\right]\mathbf{P}_C\mathbf{P}_C{}'\begin{bmatrix}\mathbf{E_A}'\\ \mathbf{E_B}'\end{bmatrix}\right) + \operatorname{tr}\left[\mathbf{L}\left(\mathbf{R}'\mathbf{R}-\mathbf{I}\right)\right] \tag{2.101}$$

where $\mathbf{P}_C$ is a $2k \times 2k$ diagonal matrix able to differently weighs the various components. As already shown in Section 2.3.3, if $\mathbf{P}_C \neq \mathbf{I}$, no algebraic solutions are known in the literature [38, 67, 91]; an iterative solution has been proposed in [91]. However, if we are not interested in weighing differently each component of $\mathbf{A}$ and $\mathbf{B}$, but we want to assign a different weight to the entries of $\mathbf{A}$ versus those of $\mathbf{B}$ as a whole, assuming that the two datasets $\mathbf{A}$ and $\mathbf{B}$ have been measured with different accuracy, a closed-form solution can be derived.

Let $1/\alpha$ and $1/\beta$ be the weights assigned to $\mathbf{A}$ and $\mathbf{B}$, respectively. Matrix $\mathbf{P}_C$ can then be written as

$$\mathbf{P}_C = \begin{bmatrix} \alpha^{-1}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \beta^{-1}\mathbf{I} \end{bmatrix}. \tag{2.102}$$

Substituting (2.102) in (2.101), it results

$$F = \operatorname{tr}\left(\left[\alpha^{-1}\mathbf{E_A} | \beta^{-1}\mathbf{E_B}\right] \mathbf{I}\mathbf{I}' \begin{bmatrix} \alpha^{-1}\mathbf{E_A}' \\ \beta^{-1}\mathbf{E_B}' \end{bmatrix} \right) + \operatorname{tr}\left[\mathbf{L}\left(\mathbf{R}'\mathbf{R} - \mathbf{I}\right)\right] \tag{2.103}$$

and, after some manipulations

$$\begin{aligned} F = \operatorname{tr}\Bigg[\left(\frac{1}{c^2+1}\right)^2 \left(\frac{c^2}{\alpha^2} + \frac{1}{\beta^2}\right) \Big(& c^2\mathbf{R}'\mathbf{A}'\mathbf{A}\mathbf{R} + \mathbf{B}'\mathbf{B} + \mathbf{t}\mathbf{j}'\mathbf{j}\mathbf{t}' \\ & + 2c\mathbf{R}'\mathbf{A}'\mathbf{j}\mathbf{t}' - 2\mathbf{B}'\mathbf{j}\mathbf{t}' - 2c\mathbf{B}'\mathbf{A}\mathbf{R}\Big) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L} \Bigg]. \end{aligned} \tag{2.104}$$

The translation vector $\mathbf{t}$ derives from equation

$$\frac{\partial F}{\partial \mathbf{t}} = \frac{\left(c^2/\alpha^2 + 1/\beta^2\right)}{\left(c^2+1\right)^2} \left[2\mathbf{j}'\mathbf{j}\mathbf{t} + 2c\mathbf{R}'\mathbf{A}'\mathbf{j} - 2\mathbf{B}'\mathbf{j}\right] = 0 \tag{2.105}$$

hence

$$\mathbf{t} = \left(\mathbf{B}' - c\mathbf{R}'\mathbf{A}'\right) \frac{\mathbf{j}}{\mathbf{j}'\mathbf{j}}. \tag{2.106}$$

Setting to zero the derivative of F with respect to $\mathbf{R}$, it follows

$$\frac{\partial F}{\partial \mathbf{R}} = \frac{\left(c^2/\alpha^2 + 1/\beta^2\right)}{\left(c^2+1\right)^2} \left[2c^2\mathbf{A}'\mathbf{A}\mathbf{R} + 2c\mathbf{A}'\mathbf{t}\mathbf{j}' - 2c\mathbf{A}'\mathbf{B}\right] + \mathbf{R}(\mathbf{L} + \mathbf{L}'). \tag{2.107}$$

Applying the same procedure used to derive $\mathbf{R}$ for the EIV-EOPA, one obtains the rotation matrix

$$\mathbf{R} = \mathbf{V} \operatorname{diag}\left(1, 1, \det(\mathbf{V}\mathbf{W}')\right) \mathbf{W}' \tag{2.108}$$

where $\mathbf{S} = \mathbf{V}\mathbf{D}_\mathbf{S}\mathbf{W}'$ is the SVD of $\mathbf{S}$, with$\mathbf{S} = \bar{\mathbf{A}}'\bar{\mathbf{B}}$. Substituting (2.106) in (2.104), it results

$$F = \operatorname{tr}\left[\frac{\left(c^2/\alpha^2 + 1/\beta^2\right)}{\left(c^2+1\right)^2} \left(c^2\bar{\mathbf{A}}'\bar{\mathbf{A}} + \bar{\mathbf{B}}'\bar{\mathbf{B}} - 2c\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R} \right) + \mathbf{L}\mathbf{R}'\mathbf{R} - \mathbf{L} \right]. \tag{2.109}$$

The solution for the scale factor c can be obtained by setting to zero the partial derivative of (2.109) with respect to c, which leads to the following equation

$$\begin{aligned} & \frac{2}{\alpha^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) c^4 \\ & + \left[\frac{4}{\alpha^2} \operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}}\right) - \frac{2}{\beta^2} \operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}}\right) - \frac{2}{\alpha^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{B}}\right) \right] c^3 \\ & + \left[-\frac{6}{\alpha^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) + \frac{6}{\beta^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) \right] c^2 \\ & + \left[\frac{2}{\alpha^2} \operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}}\right) + \frac{2}{\beta^2} \operatorname{tr}\left(\bar{\mathbf{A}}'\bar{\mathbf{A}}\right) - \frac{4}{\beta^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{B}}\right) \right] c \\ & - \frac{2}{\beta^2} \operatorname{tr}\left(\bar{\mathbf{B}}'\bar{\mathbf{A}}\mathbf{R}\right) = 0. \end{aligned} \tag{2.110}$$

The searched value of c is the root of the 4^{th} order polynomial (2.110) that minimizes the cost function (2.109).
It is worth noting that the rotation matrix $\mathbf{R}$ computed through (2.108) is the same as that

one resulting from the EIV-EOPA and from the classical EOPA. The expression of the scale factor c and the translation vector $\mathbf{t}$ are different instead.

2.5.3 EIV-Anisotropic Extended Orthogonal Procrustes Analysis (EIV-AEOPA)

Let us now introduce the EIV-AEOPA model, that allows an anisotropic scaling and considers the errors affecting both origin matrix $\mathbf{A}$ and destination matrix $\mathbf{B}$, namely:

$$\mathbf{B} + \mathbf{E_B} = \boldsymbol{\Gamma}(\mathbf{A} + \mathbf{E_A})\mathbf{R} + \mathbf{j}\mathbf{t}'. \tag{2.111}$$

A TLS estimation of the parameters of this model is novel and we report the derivation of its solution below. Following the traditional Lagrangian approach, the target function to be minimized can be written as

$$\begin{aligned} F\left(\mathbf{E_B}, \mathbf{E_A}, \mathbf{G}, \mathbf{L}, \mathbf{t}, \mathbf{R}, \boldsymbol{\Gamma}\right) = \frac{1}{\beta}\operatorname{tr}\left(\mathbf{E_B}\mathbf{E_B}'\right) + \frac{1}{\alpha}\operatorname{tr}\left(\mathbf{E_A}\mathbf{E_A}'\right) + \\ 2\operatorname{tr}\left[\mathbf{G}'\left(\mathbf{B} + \mathbf{E_B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \boldsymbol{\Gamma}\mathbf{E_A}\mathbf{R} - \mathbf{j}\mathbf{t}'\right)\right] + 2\operatorname{tr}\left[\mathbf{L}\left(\mathbf{R}'\mathbf{R} - \mathbf{I}\right)\right] \end{aligned} \tag{2.112}$$

where $\mathbf{G}$ and $\mathbf{L}$ are the matrix of Lagrangian multipliers, whereas $1/\alpha$ and $1/\beta$ are different weights assigned to $\mathbf{A}$ and $\mathbf{B}$, respectively. This can be solved by setting to zero the partial derivatives with respect to the unknowns. Eventually, the following necessary equations can be obtained:

$$\frac{\partial F}{\partial \mathbf{E_B}} = \frac{2}{\beta}\mathbf{E_B} + 2\mathbf{G} = 0 \tag{2.113}$$

$$\frac{\partial F}{\partial \mathbf{E_A}} = \frac{2}{\alpha}\mathbf{E_A} - 2\boldsymbol{\Gamma}'\mathbf{G}\mathbf{R}' = 0 \tag{2.114}$$

$$\frac{\partial F}{\partial \mathbf{G}} = 2\left(\mathbf{B} + \mathbf{E_B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \boldsymbol{\Gamma}\mathbf{E_A}\mathbf{R} - \mathbf{j}\mathbf{t}'\right) = 0 \tag{2.115}$$

$$\frac{\partial F}{\partial \mathbf{L}} = 2\left(\mathbf{R}'\mathbf{R} - \mathbf{I}\right) = 0 \tag{2.116}$$

$$\frac{\partial F}{\partial \mathbf{t}} = -2\mathbf{G}'\mathbf{j} = 0 \tag{2.117}$$

$$\frac{\partial F}{\partial \mathbf{R}} = 2\left(-\mathbf{A}'\boldsymbol{\Gamma}' - \mathbf{E_A}'\boldsymbol{\Gamma}'\right)\mathbf{G} + 2\mathbf{R}\left(\mathbf{L}' + \mathbf{L}\right) = 0 \tag{2.118}$$

$$\frac{\partial F}{\partial \boldsymbol{\Gamma}} = \mathbf{G}\left(-\mathbf{R}'\mathbf{A}' - \mathbf{R}'\mathbf{E_A}'\right) \odot \mathbf{I} = 0 \tag{2.119}$$

The translation vector $\mathbf{t}$ can be computed substituting (2.113) and (2.114) in (2.115), thus obtaining

$$\mathbf{j}\mathbf{t}' = \mathbf{B} - \beta\mathbf{G} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \alpha\boldsymbol{\Gamma}\boldsymbol{\Gamma}'\mathbf{G}. \tag{2.120}$$

Transposing the whole Expression (2.120) and multiplying on the right by $(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\mathbf{j}$, with $\boldsymbol{\Gamma}^2 = \boldsymbol{\Gamma}\boldsymbol{\Gamma}'$, one gets

$$\mathbf{t}\mathbf{j}'\left(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2\right)^{-1}\mathbf{j} = \left(\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R}\right)'\left(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2\right)^{-1}\mathbf{j} - \mathbf{G}'\mathbf{j}. \tag{2.121}$$

Taking into account also condition (2.117) leads to the solving expression:

$$\mathbf{t} = \left(\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R}\right)'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\frac{\mathbf{j}}{\varepsilon} \tag{2.122}$$

where $\varepsilon = \mathbf{j}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\mathbf{j} = \operatorname{tr}\left[(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]$.

To retrieve matrix $\mathbf{R}$, let us first substitute (2.114) in (2.118) and, after multiplying on the left by $\mathbf{R}'$, one obtains

$$\alpha\mathbf{G}'\boldsymbol{\Gamma}\boldsymbol{\Gamma}'\mathbf{G} + \mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{G} - \mathbf{R}'\mathbf{R}\left(\mathbf{L}' + \mathbf{L}\right) = 0. \tag{2.123}$$

Since $\alpha\mathbf{G}'\boldsymbol{\Gamma}\boldsymbol{\Gamma}'\mathbf{G}$ and $\mathbf{R}'\mathbf{R}\left(\mathbf{L}' + \mathbf{L}\right)$ are symmetric, then

$$\operatorname{sym}\left[\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'\mathbf{G}\right]. \tag{2.124}$$

From (2.120), $\mathbf{G}$ can be written also as

$$\mathbf{G} = \left(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2\right)^{-1}\left(\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}'\right). \tag{2.125}$$

Hence, substituting (2.125) and (2.122) in (2.124), it results

$$\begin{aligned}\operatorname{sym}\Bigg[&\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\left[\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\varepsilon}(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]\mathbf{B} - \\ &\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\left[\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\varepsilon}(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]\boldsymbol{\Gamma}\mathbf{A}\mathbf{R}\Bigg].\end{aligned} \tag{2.126}$$

It is easy to see that $\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\left[\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\varepsilon}(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]\boldsymbol{\Gamma}\mathbf{A}\mathbf{R}$ is symmetric, so

$$\operatorname{sym}\left[\mathbf{R}'\mathbf{A}'\boldsymbol{\Gamma}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\left[\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\varepsilon}(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]\mathbf{B}\right]. \tag{2.127}$$

Calling

$$\mathbf{S} = \mathbf{A}'\boldsymbol{\Gamma}'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\left[\mathbf{I} - \frac{\mathbf{j}\mathbf{j}'}{\varepsilon}(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\right]\mathbf{B} \tag{2.128}$$

from (2.127) it follows that

$$\mathbf{R} = \mathbf{V}\mathbf{W}' \tag{2.129}$$

where $\mathbf{S} = \mathbf{V}\mathbf{D}_{\mathbf{S}}\mathbf{W}'$ is the SVD of $\mathbf{S}$. As previously observed, to retrieve a rotation matrix one resorts to

$$\mathbf{R} = \mathbf{V}\operatorname{diag}\left(1, 1, \det(\mathbf{V}\mathbf{W}')\right)\mathbf{W}'. \tag{2.130}$$

Finally, substituting (2.114) and (2.125) in (2.119)

$$\left(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2\right)^{-1}\left(\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}'\right)\left[\alpha\left(\mathbf{B} - \boldsymbol{\Gamma}\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}'\right)'(\beta\mathbf{I} + \alpha\boldsymbol{\Gamma}^2)^{-1}\boldsymbol{\Gamma} + \mathbf{R}'\mathbf{A}'\right] \odot \mathbf{I} = 0 \tag{2.131}$$

and defining $\mathbf{X} = \mathbf{A}\mathbf{R}$ and $\mathbf{Y} = \mathbf{B} - \mathbf{j}\mathbf{t}'$, after some rewriting one obtains:

$$(\alpha\boldsymbol{\Gamma}\mathbf{X}\mathbf{Y}'\boldsymbol{\Gamma} + \beta\boldsymbol{\Gamma}\mathbf{X}\mathbf{X}' - \alpha\mathbf{Y}\mathbf{Y}'\boldsymbol{\Gamma} - \beta\mathbf{Y}\mathbf{X}') \odot \mathbf{I} = 0. \tag{2.132}$$

Equation (2.132) can be solved independently for each element c_i of matrix $\boldsymbol{\Gamma}$. The value of each scale factor is retrieved from the following equation:

$$\left(\alpha\sum_{j=1}^{3}\mathbf{X}_{ij}\mathbf{Y}_{ij}\right)c_i^2 + \left(\beta\sum_{j=1}^{3}\mathbf{X}_{ij}^2 - \alpha\sum_{j=1}^{3}\mathbf{Y}_{ij}^2\right)c_i - \beta\left(\sum_{j=1}^{3}\mathbf{X}_{ij}\mathbf{Y}_{ij}\right) = 0 \tag{2.133}$$

Analyzing Equation (2.133), it can be noticed that there are two real solutions, one negative and one positive; the latter is the searched value of the scale factor c_i.
Let us recall the solving equation for $\boldsymbol{\Gamma}$ derived from the classical AEOPA model (2.67), that can be written also as follows:

$$(\boldsymbol{\Gamma}\mathbf{X}\mathbf{X}' - \mathbf{Y}\mathbf{X}') \odot \mathbf{I} = 0. \tag{2.134}$$

Comparing Equation (2.134) with the one derived from the EIV model (2.111), it is easy to see that the two terms of the LS solution (2.134) are contained in the TLS solution (2.133). Let us rewrite (2.134) and (2.133) as follows:

$$[(\mathbf{\Gamma X} - \mathbf{Y})\mathbf{X}'] \odot \mathbf{I} = 0 \tag{2.135}$$

$$[\beta(\mathbf{\Gamma X} - \mathbf{Y})\mathbf{Y}'\mathbf{\Gamma} + \alpha(\mathbf{\Gamma X} - \mathbf{Y})\mathbf{X}'] \odot \mathbf{I} = 0. \tag{2.136}$$

It is important to underline that, if the system $\mathbf{\Gamma X} = \mathbf{Y}$ has an exact solution (i.e., matrices $\mathbf{A}$ and $\mathbf{B}$ are error-less), this satisfies both (2.135) and (2.136). Therefore in the ideal case ordinary LS and TLS approaches coincide, as expected, whereas they lead to different solutions when the data are affected by noise.
Similarly to the LS solution of the AEOPA problem, the TLS solution of the EIV-AEOPA model requires a block relaxation scheme to estimate the unknowns $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{\Gamma}$.

2.6 Experiments

The EIV-EOPA and EIV-WEOPA algorithms described in Sections 2.5.1 and 2.5.2 were tested to perform the datum transformation of four points belonging to a small topographic network. The points coordinates were determined in the datum WGS84 through GPS measurements (Table 2.1) and they were also measured in a local reference system using a total station (Table 2.2). The GPS measurements were assumed to have a standard deviation of $\sigma_{GPS} = \pm 5$ cm, while the coordinate in the local datum were assumed to have a standard deviation of $\sigma_{EDM} = \pm 1$ cm.

Table 2.1: Point coordinates in the source system WGS84.

Point	X_{GPS} [m]	Y_{GPS} [m]	Z_{GPS} [m]
A	4314478.698	1013256.717	4571659.536
B	4314521.907	1013215.197	4571628.184
C	4314570.538	1013205.789	4571584.703
D	4314530.926	1013136.124	4571637.601

Table 2.2: Point coordinates in the target local system.

Point	X_{EDM} [m]	Y_{EDM} [m]	Z_{EDM} [m]
A	0.000	0.000	100.000
B	0.000	67.655	100.066
C	-33.056	124.680	100.091
D	62.684	117.572	100.691

The parameters of the similarity transformation between the two reference systems were computed by applying the classical least squares EOPA solution reported in Section 2.3.2, the EIV-EOPA model and the novel EIV-WEOPA solution. The weights assigned to the two datasets for the latter case were $\alpha = 1/\sigma^2_{GPS} = 0.04$ and $\beta = 1/\sigma^2_{EDM} = 1$.
The rotation matrix obtained with all the three methods is the following:

$$\mathbf{R} = \begin{bmatrix} -0.3706961890 & 0.6380215670 & 0.6749168953 \\ -0.7739159876 & -0.6139475490 & 0.1553140405 \\ 0.5134572812 & -0.4647546526 & 0.7213631078 \end{bmatrix}.$$

It is easy to see that $\mathbf{R}$ is an orthogonal matrix, with $\det(\mathbf{R}) = 1$. In Table 2.3 the values of the scale factor c and the translation vector $\mathbf{t}$ computed by the three different algorithms are

reported. The differences between the estimated parameters can be considered significant when a high accuracy of the coordinate values resulting from the transformation is required.

TABLE 2.3: Parameters c and $\mathbf{t}$ estimated using different methods.

Parameter	EOPA	EIV-EOPA	EIV-WEOPA
c [-]	1.0000852732	1.0000853646	1.0000855921
$\mathbf{t_x}$ [m]	36187.583	36187.586	36187.594
$\mathbf{t_y}$ [m]	-5944.436	-5944.436	-5944.437
$\mathbf{t_z}$ [m]	-6367557.047	-6367557.629	-6367559.077

The AEOPA (Section 2.3.4) and the EIV-AEOPA (Section 2.5.3) can instead be applied and compared for solving the problem of estimating the position and orientation of a perspective camera given its intrinsic parameters and a set of world-to-image correspondences [57], known as *exterior orientation* problem in photogrammetry or *Perspective-n-Point* camera pose in computer vision.
First of all, we briefly review here how this problem can be formulated in terms of model (2.50) or (2.111). Given at least three control points and their projections, the exterior orientation problem requires to find a rotation matrix $\mathbf{R}$ and a vector $\mathbf{t}$ (specifying attitude and position of the camera) such that the vector form of collinearity equations:

$$\mathbf{a}_i = c_i^{-1}\mathbf{R}(\mathbf{b}_i - \mathbf{t}) \tag{2.137}$$

is satisfied for some positive scalar c_i, where

- $\mathbf{b}_i$ is the coordinate vector of the i-th control point in the external system;
- $\mathbf{t}$ is the coordinate vector of the projection center in the external system;
- c_i is a positive scalar proportional to the "depth" of the point, i.e., the distance from the i-th control point to the plane containing the projection center and parallel to the image plane;
- $\mathbf{R}$ is the rotation matrix transforming from the external system to the camera system;
- $\mathbf{a}_i$ is the coordinate vector of the i-th control point in the camera system, where the third component is equal to the principal distance or focal length.

Expressing (2.137) with respect to $\mathbf{b}_i$ yields:

$$\mathbf{b}_i = c_i\mathbf{R}'\mathbf{a}_i + \mathbf{t}. \tag{2.138}$$

After transposing and extending to p control points $\mathbf{b}_1 \dots \mathbf{b}_p$, it results:

$$\mathbf{B} = \mathbf{\Gamma}\mathbf{A}\mathbf{R} + \mathbf{j}\mathbf{t}' \tag{2.139}$$

where $\mathbf{A}$ is the matrix by rows of image point coordinates defined in the camera frame, $\mathbf{B}$ is the matrix by rows of point coordinates defined in the external system, $\mathbf{\Gamma}$ is the diagonal (positive) depth matrix.
Formula (2.139) can then be rewritten in the form of the AEOPA model (2.50) or the EIV-AEOPA model (2.111), according to whether the error is assumed to affect only $\mathbf{B}$ or both $\mathbf{A}$ and $\mathbf{B}$. It is particularly significant in this photogrammetric application the capability of our solution (Section 2.5.3) to take into account the different variances of $\mathbf{E_A}$ and $\mathbf{E_B}$, since image coordinate $\mathbf{a}_i$ and 3D points coordinate $\mathbf{b}_i$ are measured with different accuracy.
Garro et al. [57] compared the LS solution of (2.50) with state-of-the-art algorithms that perform the exterior orientation, showing that AEOPA reaches the best trade-off between speed and accuracy. In this chapter, we tested the proposed TLS solution of the AEOPA for solving the exterior orientation of an image against the LS formulation [57].

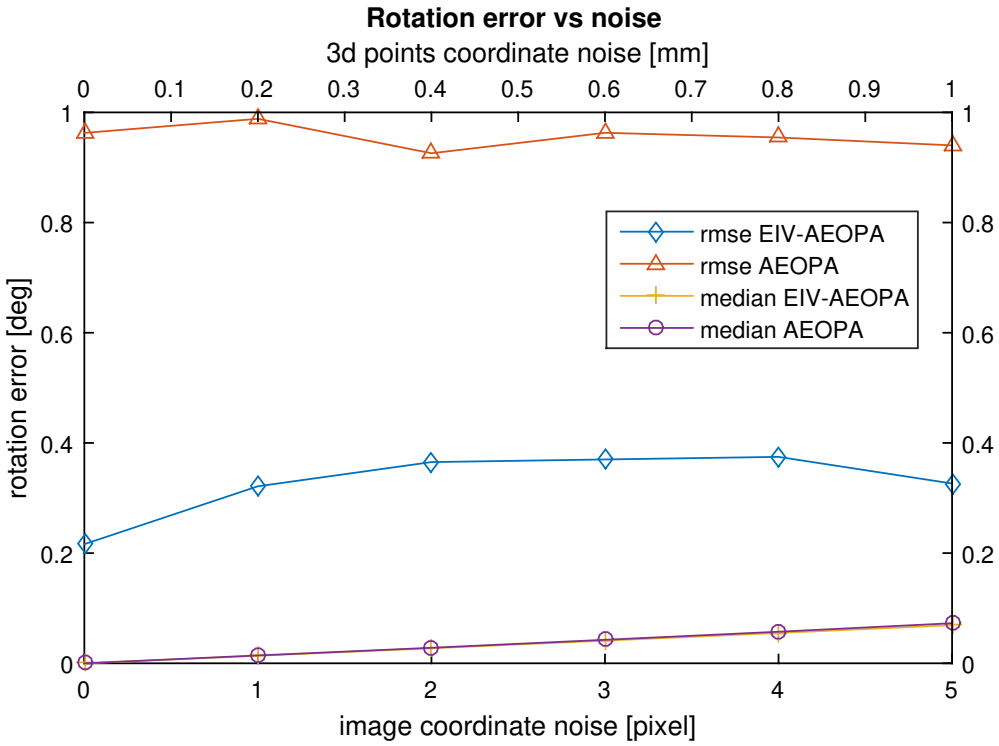


FIGURE 2.2: Rotation error vs noise using 10 correspondences and a distance of the camera from the origin equals to 10 m. The RMSE and the median rotation errors are plotted against the standard deviation of the noise added to image coordinates and 3D points coordinate.

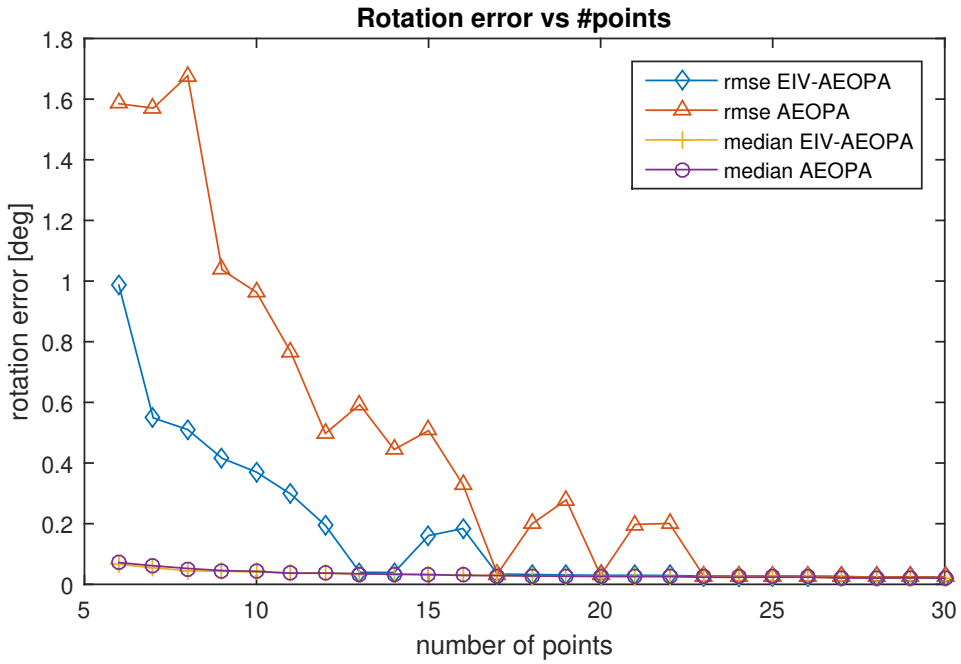


FIGURE 2.3: Rotation error vs number of points. The RMSE and the median rotation errors are plotted against the number of points that have been used. The distance of the camera from the origin is equal to 10 m and the standard deviation of the noise added is 3 pixel for the image coordinate and 0.71 mm for the 3D points coordinate.

To carry out the simulation, $p = \{6, ..., 30\}$ 3D points were randomly distributed in a sphere of unit radius centered on the origin and perturbed with random noise with standard deviation $\sigma_B = \{0.10, 0.27, 0.71, 1.88, 5.00\}$ [mm] (corresponding to five logarithmically spaced values from 0.10 to 5.00). The camera was positioned at distances of 5 and 10 meters from the origin and the focal length was chosen so as to yield a view angle of 60° with an image size of 1000 x 1000 pixels. Different values of noise $\sigma_A = \{1, ..., 5\}$ [pixel] were added to the image coordinates obtained from the projection of the noise-free 3D points. For each setting the test was run 500 times and the mean and median error norms were computed. In all the experiments the initial depths were set to one. Results are reported in Figures 2.2 and 2.3. As a figure of merit only the rotation errors are shown, since the behavior of the translations errors is similar. The rotation error is the angle of the residual rotation, computed as $\|\log(\mathbf{R}'\hat{\mathbf{R}})\|_F$, where $\mathbf{R}$ is the ground truth, $\hat{\mathbf{R}}$ is the actual rotation and $\|\cdot\|_F$ is the Frobenius norm.
Comparing the median error, one can notice that EIV-AEOPA and AEOPA lead almost to the same accuracy. On the other hand, the root mean square error (RMSE) is different when the number of correspondences in the image is small ($n \leq 15$) (this is clear in Figure 2.2 where 10 points are considered). This is due to the fact that AEOPA in a few cases converges to wrong results, that are not sufficient to skew the median, but affects the mean error. The rundown of this simulation is that the new TLS solution works better than the ordinary LS algorithm when the number of reference points (whose coordinate are known in both the camera and the external reference frame) is small, which is a common situation in photogrammetry.

2.7 Conclusion

The aim of this chapter was twofold. On the one hand, we gave a comprehensive view of the classical least squares orthogonal Procrustes variants, a set of mathematical tools used to perform transformations among corresponding matrix elements. Procrustes Analysis can play a significant role in photogrammetry, since it represents the basic tool for an alternative and efficient solution of classical photogrammetric problems such as image exterior orientation and bundle block adjustment. On the other we applied the Errors-In-Variables model, developing novel total least squares solutions of the Weighted Extended Orthogonal Procrustes Analysis and the Anisotropic Extended Orthogonal Procrustes Analysis that can cope with the uncertainty affecting both origin and destination sets of observations. In this way, we obtained algorithms that maintain all the advantages of the classical Procrustes Analysis, i.e., problems related to linearization and approximate parameters values determination are avoided, and at the same time removing the unrealistic assumption that the source coordinates are error-less. This Errors-In-Variables model is closer to reality and leads to a more realistic estimation of the unknown transformation parameters.

Chapter 3

Permutation Procrustes Problem and Variations

In this chapter we address the problem of finding correspondences among element sets, where the elements can be, e.g., keypoints extracted from images or point clouds. When two sets are involved, this can be seen as a Permutation Procrustes problem. In particular, we propose a novel solution to the multi-view matching problem that, given a set of noisy pairwise correspondences, jointly updates them so as to maximize their consistency. Our method is based on a spectral decomposition, resulting in a closed-form efficient algorithm, in contrast to other iterative techniques that can be found in the literature. Experiments on both synthetic and real datasets show that our method achieves comparable or superior accuracy to state-of-the-art algorithms in significantly less time. We also demonstrate that our solution can efficiently handle datasets of hundreds of views, which is unprecedented in the literature.

3.1 Introduction

Establishing correspondences between keypoint sets is a fundamental problem in photogrammetry and computer vision, that lies at the basis of any geometric computation (e.g., structure from motion and point cloud registration). In this chapter we consider the case in which keypoints are extracted from a collection of images, while in Chapter 4 we will show an application to the point cloud registration problem.

3.1.1 Permutation Procrustes Analysis

The majority of works on this topic focus on finding correspondences between two sets [106, 98, 103, 102, 109, 128], that can be formulated as a Permutation Procrustes Analysis (PPA) [67]. In fact, given two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times m'}$, the matching problem corresponds to identifying the permutation $\mathbf{Q}$ that best align $\mathbf{B}$ onto $\mathbf{A}$, i.e., solving

$$\|\mathbf{A} - \mathbf{BQ}\|_F = \min \tag{3.1}$$

under the condition that $\mathbf{Q}$ is a permutation matrix, i.e., exactly one entry in each row and column is equal to 1 and all other entries are 0. Permutation matrices are orthogonal, so this is a special case of the OPA model (see Section 2.3.1). The minimization of (3.1) is equivalent to

$$\mathrm{tr}(\mathbf{PC}) = \max \tag{3.2}$$

with $\mathbf{P} = \mathbf{Q}'$ and $\mathbf{C} = \mathbf{B}'\mathbf{A}$.

Let us consider a relaxed problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}c_{ij} \\ \text{subject to} \quad & p_{ij} \geq 0 \\ & \sum_{i=1}^{n} p_{ij} = 1 \\ & \sum_{j=1}^{n} p_{ij} = 1. \end{aligned} \tag{3.3}$$

This is a linear programming problem, hence its solution must occur at a vertex of the feasible region, which corresponds to a permutation matrix. This is in fact an assignment problem, which can be solved by the Hungarian method [94, 119] ($-c_{ij}$ is the cost of assigning agent i to task j, and p_{ij} represents the assignment).
Setting $\mathbf{B} = \mathbf{I}$ in (3.1), the solution $\mathbf{Q}$ is the closets permutation matrix to $\mathbf{A}$ and this can be used also to project a matrix onto the set of permutations.

3.1.2 Multi-View Matching

In many tasks it is often required to find matches across multiple views and recent studies have suggested that jointly optimizing the correspondences across the whole set can lead to significant improvements when compared to computing matches between pairs of views in isolation [124, 166], since pairwise matching algorithms can generate noisy and unreliable results. Hence, one can resort to higher level constraints that arise from the closed-loop consistency of matching across multiple views. This is called *joint matching* or *multi-view matching* by some authors and it is a problem that cannot be solved via the Permutation Procrustes model, so one must resort to different formulations.
A key concept in this context is that of *cycle consistency*, namely the composition of pairwise matches along any loop should give the identity. This property is exploited in several algorithms to remove outliers among pairwise correspondences [164, 122, 88]. However, in practice, a few number of consistent cycles may be found due to noise, and considering all the cycles is computationally intractable. It is shown in some recent works [124, 78, 166] that, if all the pairwise correspondences are collected in a block-matrix, then cycle consistency can be reduced to the requirement that such a matrix is positive semidefinite and low-rank. In [124] and [6] multi-view matching is expressed as a synchronization problem, which is approximately solved via spectral decomposition. In [78] a solution based on semidefinite programming is proposed, which, however, assumes total correspondences between all views. Such technique is extended in [28] in order to handle partial correspondences, and theoretical guarantees for exact matching in the presence of corrupted input are provided, assuming a certain noise model. In [166] the joint matching problem is formulated as a low-rank matrix recovery task and the nuclear-norm relaxation for rank minimization is employed. The resulting cost function is optimized via the Alternating Direction Method of Multipliers (ADMM). Finally, in [162] a solution based on the proximal Gauss-Seidel method is provided, which, as [78], assumes total correspondences between all the views, thus limiting its applicability to real scenarios.

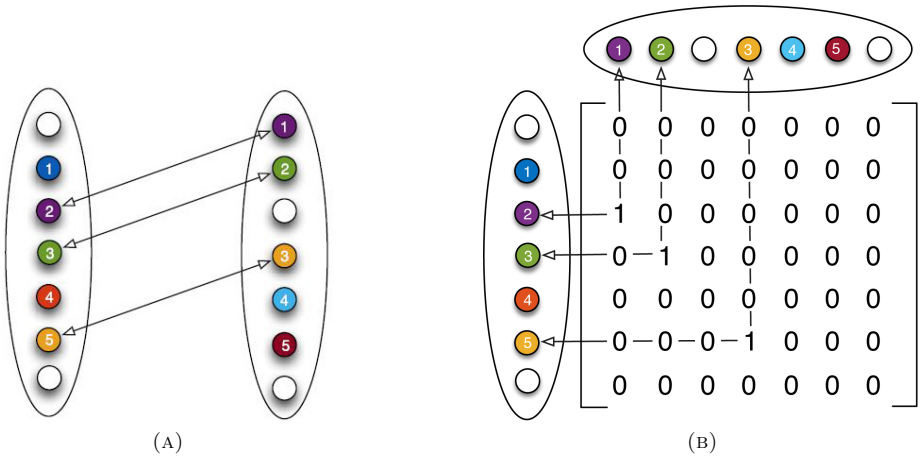
3.1.3 Contribution

After formally defining the problem in Section 3.2, in this chapter we propose a novel closed-form method for multi-view matching, called MatchEIG, that is extremely simple and can be coded in a few lines of Matlab. The developed solution share the same framework as [124], since it is based on a spectral decomposition. Differences with respect to [124] are detailed in Section 3.3 and they imply a significant improvement in performance, as demonstrated by

the experimental validation. Synthetic and real experiments, presented in Section 3.4, show in fact that the accuracy of MatchEIG is comparable or superior to the state of the art and it is significantly faster than all the competing methods. Thanks to its computational efficiency, it successfully handles sets of hundreds of views, where others fail to produce a solution.

3.2 Problem Formulation

The goal of multi-view matching is to establish correspondences between all pairs of views. Let n denote the number of views and let m_i denote the number of keypoints in view i. The correspondences between the keypoints in view j and those in view i can be represented as a *partial permutation matrix* $\mathbf{P}_{ij} \in \{0,1\}^{m_i \times m_j}$. It can be constructed as follows: $[\mathbf{P}_{ij}]_{h,k} = 1$ if keypoint k in view j is matched with keypoint h in view i; $[\mathbf{P}_{ij}]_{h,k} = 0$ otherwise. If row $[\mathbf{P}_{ij}]_{h,\cdot}$ is a row of zeros, then keypoint h in view i does not have a matching keypoint in view j. If column $[\mathbf{P}_{ij}]_{\cdot,k}$ is a column of zeros, then keypoint k in view j does not have a matching keypoint in view i. Figure 3.1 graphically shows how a permutation matrix is related to keypoints correspondences.
A partial permutation matrix has at most one nonzero entry in each row and column, and these nonzero entries are all 1. If exactly one entry in each row and column is equal to 1 (and all other entries are 0), then the permutation is *total*. Partial permutations are suitable to model matches in practical scenarios, since they can represent missing correspondences, whereas the usage of total permutations requires that the same set of keypoints is present in all the views, which is an unrealistic assumption.



(A) (B)

Figure 3.1: (A) Keypoint matches between two sets of views and (B) corresponding partial permutation matrix.

Let us assume that all the keypoints belong to a *universe* set. Let $\mathbf{P}_i \in \{0,1\}^{m_i \times d}$ denote the partial permutation matrix representing the correspondences between the keypoints in view i and those in the universe, where d denotes the size of the universe. In the absence of noise, the correspondences between view j and view i can be equivalently represented by first computing the matches between view j and the universe, and then from the universe to view i, namely

$$\mathbf{P}_{ij} = \mathbf{P}_i \mathbf{P}_j{}'. \tag{3.4}$$

Equation (3.4) is called the *consistency constraint*. The matrix $\mathbf{P}_{ij}$ is referred to as the *relative* permutation of the pair (i,j), and the matrix $\mathbf{P}_i$ (resp. $\mathbf{P}_j$) is referred to as the *absolute*

permutation of view i (resp. j). According to Equation (3.4), the solution to the multi-view matching problem can also be achieved by first computing n absolute permutations $\mathbf{P}_1, \dots, \mathbf{P}_n$ and then setting $\mathbf{P}_{ij} = \mathbf{P}_i \mathbf{P}_j{}'$.

3.2.1 Spectral Properties

As observed in [124, 78, 166, 6], the consistency constraint can be expressed in a compact matrix form if all the absolute and relative permutations are collected in two block-matrices $\mathbf{X} \in \{0,1\}^{m \times d}$ and $\mathbf{Z} \in \{0,1\}^{m \times m}$ respectively, where $m = \sum_{i=1}^{n} m_i$, namely

$$\mathbf{X} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \dots \\ \mathbf{P}_n \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \dots & \mathbf{P}_{1n} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \dots & \mathbf{P}_{2n} \\ \dots & & & \dots \\ \mathbf{P}_{n1} & \mathbf{P}_{n2} & \dots & \mathbf{P}_{nn} \end{bmatrix}. \tag{3.5}$$

Note that $\mathbf{Z}$ may contain zero blocks: if all the keypoints in view i do not match with any keypoint in view j, then $\mathbf{P}_{ij} = 0$. Using this notation, Equation (3.4) becomes

$$\mathbf{Z} = \mathbf{X}\mathbf{X}' \tag{3.6}$$

which implies that $\mathbf{Z}$ is symmetric positive semidefinite and has rank d.

Proposition 3.1. *[124, 139] The columns of $\mathbf{X}$ are d (orthogonal) eigenvectors of $\mathbf{Z}$ corresponding to the eigenvalue n, assuming that all the permutations are total.*

Proof. In the case of total permutations we have $\mathbf{P}_i{}'\mathbf{P}_i = \mathbf{I}_d$ and hence $\mathbf{X}'\mathbf{X} = n\mathbf{I}_d$, where $\mathbf{I}_d$ denotes the $d \times d$ identity matrix. Thus $\mathbf{Z}\mathbf{X} = \mathbf{X}\mathbf{X}'\mathbf{X} = n\mathbf{X}$, which means that the columns of $\mathbf{X}$ are d eigenvectors of $\mathbf{Z}$ corresponding to the eigenvalue n. □

The following new result generalizes Proposition 3.1 to the case of partial permutations.

Proposition 3.2. *The columns of $\mathbf{X}$ are d (orthogonal) eigenvectors of $\mathbf{Z}$ and the corresponding eigenvalues are given by the diagonal of $\mathbf{V} := \sum_{i=1}^{n} \mathbf{P}_i{}'\mathbf{P}_i$.*

Proof. In the case of partial permutations, $\mathbf{P}_i$ does not have an inverse, thus the $d \times d$ (diagonal) matrix $\mathbf{P}_i{}'\mathbf{P}_i$ is not equal to the identity: $[\mathbf{P}_i{}'\mathbf{P}_i]_{k,k} = 1$ if the k-th keypoint in the universe is present in view i; $[\mathbf{P}_i{}'\mathbf{P}_i]_{k,k} = 0$ otherwise. Define the $d \times d$ diagonal matrix $\mathbf{V} := \mathbf{X}'\mathbf{X} = \sum_{i=1}^{n} \mathbf{P}_i{}'\mathbf{P}_i$. Then

$$\mathbf{Z}\mathbf{X} = \mathbf{X}\mathbf{V} \tag{3.7}$$

which is a spectral decomposition, i.e., the columns of $\mathbf{X}$ are d eigenvectors of $\mathbf{Z}$ and the corresponding eigenvalues are given by the diagonal of $\mathbf{V}$. Specifically, the k-th eigenvalue is an integer which counts how many views match the k-th keypoint in the universe. □

Note that in the case of total permutations all the keypoints are present in all the views, therefore $\mathbf{V} = n\mathbf{I}_d$ and all the eigenvalues are equal, hence we get Proposition 3.1. Since $\mathbf{Z}$ has rank d, the matrix $\mathbf{V}$ contains the *largest* eigenvalues of $\mathbf{Z}$ and all the other eigenvalues are zero. Thus, in the presence of noise, we can take the eigenvectors of $\mathbf{Z}$ corresponding to the d largest eigenvalues as an estimate of $\mathbf{X}$. In Section 3.2.2 we describe the meaning of this procedure in terms of an optimization problem, and in Section 3.3 we show how it can be exploited to derive an efficient method for multi-view matching.

3.2.2 Optimization Problem

In practice, pairwise correspondences contain errors, hence what we measure is an estimate $\widehat{\mathbf{P}}_{ij}$ of the relative permutation between view i and view j (in this chapter we use the hat accent to denote approximate quantities). The goal is to compute a set of partial permutation matrices $\{\mathbf{P}_{ij}\}_{i,j=1}^{n}$ such that the consistency constraint is satisfied and $\mathbf{P}_{ij}$ is as close as

possible to its measure $\widehat{\mathbf{P}}_{ij}$, namely $\mathbf{P}_{ij} \approx \widehat{\mathbf{P}}_{ij}$ for all $i, j \in \{1, \dots, n\}$. A possible approach consists in considering the following optimization problem

$$\max_{\{\mathbf{P}_{ij}\}_{i,j=1}^{n}} \sum_{i,j=1}^{n} \langle \widehat{\mathbf{P}}_{ij}, \mathbf{P}_{ij} \rangle \quad \text{s.t. } \mathbf{P}_{ij} = \mathbf{P}_i \mathbf{P}_j{}' \tag{3.8}$$

where each optimization variable is constrained to be a partial permutation matrix. Here $\langle \cdot, \cdot \rangle$ denotes the matrix inner product. The cost function in (3.8) counts, for each view pair (i, j), the number of keypoints equally matched by permutations $\mathbf{P}_{ij}$ and $\widehat{\mathbf{P}}_{ij}$.
If $\widehat{\mathbf{Z}}$ denotes the block-matrix containing the measured relative permutations $\widehat{\mathbf{P}}_{ij}$, then Equation (3.8) rewrites

$$\max_{\mathbf{Z}} \ \langle \widehat{\mathbf{Z}}, \mathbf{Z} \rangle = \max_{\mathbf{Z}} \ \mathrm{tr}(\widehat{\mathbf{Z}}\mathbf{Z}') \quad \text{s.t. } \mathbf{Z} = \mathbf{X}\mathbf{X}' \tag{3.9}$$

$$\iff \max_{\mathbf{X}} \ \langle \widehat{\mathbf{Z}}, \mathbf{X}\mathbf{X}' \rangle = \max_{\mathbf{X}} \ \mathrm{tr}(\mathbf{X}'\widehat{\mathbf{Z}}\mathbf{X}) \tag{3.10}$$

where $\mathbf{X}$ is constrained to be composed of partial permutation matrices. Maximizing the objective function in Equation (3.10) is a challenging task since the feasible set consists of binary variables which makes the problem combinatorially NP-hard. Moreover, optimizing with respect to multiple permutation matrices simultaneously increases the difficulty of the problem. For these reasons, it is common practice to relax some constraints on the optimization variables, thus providing tractable approaches that solve the multi-view matching problem approximately but efficiently. Some examples include the *semidefinite* relaxation [28], the *low-rank* relaxation [166] and the *spectral* relaxation [124, 6]. The SPECTRAL method of [124] treats $\mathbf{X}$ as a real matrix instead of a binary matrix and enforces the columns of $\mathbf{X}$ to be orthogonal, resulting in the following optimization problem

$$\max_{\mathbf{U}'\mathbf{U}=\mathbf{I}_d} \ \mathrm{tr}(\mathbf{U}'\widehat{\mathbf{Z}}\mathbf{U}) \tag{3.11}$$

where the notation $\mathbf{U}$ instead of $\mathbf{X}$ is used to underline that, due to the relaxation, the optimal $\mathbf{U}$ will not be composed of partial permutation matrices. Equation (3.11) is a generalized Rayleigh problem, whose solution is given by the d leading eigenvectors of $\widehat{\mathbf{Z}}$.

3.2.3 Partial Permutation Procrustes Analysis

In order to obtain proper correspondences from $\mathbf{U}$, each $m_i \times d$ block is projected onto the nearest partial permutation matrix, i.e., one must solve a variation of the Permutation Procrustes Analysis (3.2) formulated in Section 3.1.1, which we will refer to as *Partial Permutation Procrustes Analysis* (PPPA).
In order to cater for partial permutation the relaxed constraint is changed to $\sum_{i=1}^{n} p_{ij} \leq 1$ (and likewise for the columns). Moreover, a soft thresholding is applied to c_{ij} with a small threshold, otherwise even a negligible c_{ij} would justify a $p_{ij} = 1$, which would compel the solution to be always a total permutation. The resulting linear programming problem is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} S_\lambda(c_{ij}) \\ \text{subject to} \quad & p_{ij} \geq 0 \\ & \sum_{i=1}^{n} p_{ij} \leq 1 \\ & \sum_{j=1}^{n} p_{ij} \leq 1. \end{aligned} \tag{3.12}$$

which can be seen as an assignment problem and solved by the Hungarian method, as in the PPA case.

3.3 Our Method

The SPECTRAL method is extremely fast, as multi-view matching is solved in one shot via spectral decomposition. However, since *absolute* permutations are computed, the knowledge of the size of the universe d is required, which is not available in practice. The importance of a correct estimate of d is also demonstrated experimentally in Section 3.4.1.
We introduce here a novel technique for multi-view matching, dubbed MATCHEIG, which inherits the positive aspects of the SPECTRAL method, namely efficiency and simplicity, and at the same time it overcomes its drawback, i.e., the need of the correct value of d as input. The key observation is that relative permutations are independent from d, thus a method that aims at producing *relative* permutations instead of absolute ones can get by without knowing precisely d. Specifically, our method proceeds as follows. First, the top d eigenvectors of $\widehat{\mathbf{Z}}$ are computed and collected in a $m \times d$ matrix $\mathbf{U}$, as done by SPECTRAL. Let $\mathbf{D}$ be the diagonal matrix containing the corresponding d eigenvalues $\lambda_1, \dots \lambda_d$. The matrix

$$\widehat{\mathbf{Z}}_d = \mathbf{U}\mathbf{D}\mathbf{U}' \tag{3.13}$$

is the solution of (3.9) under the spectral relaxation. In this way we get an estimate of $\mathbf{Z}$ – which contains *relative* permutations, and this is a key difference with respect to the SPECTRAL method that provides an estimate of $\mathbf{X}$ – which contains *absolute* permutations.
Suppose that we are given an estimate $\widehat{d}$ of the size of the universe such that $\widehat{d} \geq d$, and we compute $\widehat{\mathbf{Z}}_{\widehat{d}}$ accordingly. Since $\widehat{\mathbf{Z}}$ has approximately rank d, we expect that the least $\widehat{d} - d$ eigenvalues $\lambda_{d+1}, \dots \lambda_{\widehat{d}}$ are smaller than the top d eigenvalues, thus the corresponding eigenvectors in $\mathbf{U}$ have a limited impact on $\widehat{\mathbf{Z}}_{\widehat{d}}$, in particular: $\|\widehat{\mathbf{Z}}_d - \widehat{\mathbf{Z}}_{\widehat{d}}\|_2 = |\lambda_{d+1}|$.
Note that, due to the relaxation, the $m_i \times m_j$ blocks of $\widehat{\mathbf{Z}}_d$ are not guaranteed to be partial permutation matrices. In order to enforce this constraint we analyze two different strategies. A first stage common to both consists in setting to zero all the entries smaller than a given threshold t. We set $t = 0.25$ in simulations and $t = 0.5$ in real experiments. A higher threshold allows for more missing matches, and this is useful in real datasets to model the presence of isolated keypoints.
Then, a principled approach consists in projecting each block onto the closest partial permutation matrix via PPPA. We call this method MATCHEIG-CP, where CP stands for "closest permutation". This projection, however, slows down the computing time, so in our MATCHEIG algorithm we use a greedy strategy that, if applied to each block, returns a valid permutation, although not necessarily the closest one. This strategy is approximate but it produces no noticeable loss in accuracy, while greatly boosting the speed, as experiments will demonstrate.
The proposed projection method takes a matrix $\mathbf{C}$ as input and returns a (partial) permutation matrix $\mathbf{P}$ constructed as follows: search among the non-zero entries of $\mathbf{C}$ for the ones where the maximum over the corresponding row or column is achieved. These entries are then sorted by decreasing magnitude and examined sequentially starting from the largest element: let (i, j) be the index of the current entry, and let $\mathbf{P}$ be the output matrix, initialized to 0; then $[\mathbf{P}]_{i,j}$ is set to 1 provided that $\mathbf{P}$ remains a partial permutation.
The idea behind this procedure is the following. For a given row i, which corresponds to a keypoint in one view, each entry $[\mathbf{C}]_{i,j}$ represents the extent of pairing between keypoint i and keypoint j, and the greatest element in this row can be regarded as the most likely correspondence. The same holds for each column. To these putative matches we need to apply the principle of exclusion, and we do it in a greedy way, as in [136]: the strongest match wins and inhibits other 1s to be placed in its row or column.
To summarize, MATCHEIG proceeds as follows:

1. Compute the eigenvalue decomposition of matrix $\widehat{\mathbf{Z}}$, keeping only the $\widehat{d}$ largest eigenvectors and eigenvalues:
$$\widehat{\mathbf{Z}}_{\widehat{d}} = \mathbf{U}\mathbf{D}\mathbf{U}'.$$
2. Set to zero all the entries smaller than a given threshold t.
3. Project each block of $\widehat{\mathbf{Z}}_{\widehat{d}}$ onto the space of permutation matrices using the following greedy strategy:
   a) Compute maximums over rows/columns, which represent putative matches;
   b) Sort such maximums by decreasing magnitude;
   c) Initialize the output matrix $\mathbf{P}$ to zero;
   d) Starting from the largest element, analyze sequentially each maximum, indexed by (i, j), and place a 1 in correspondence of $[\mathbf{P}]_{i,j}$ provided that $\mathbf{P}$ remains a partial permutation.

When implementing the algorithm, some properties of MatchEIG can be exploited in order to reduce the computing time and the memory requirement. In fact, note that, because of noise, $\widehat{\mathbf{Z}}_{\widehat{d}}$ is full, in general, and its size can become large in practical scenarios. However, this matrix needs not to be explicitly computed, for only one block is needed at a time. Specifically, when a view pair (i, j) is considered, the product $\mathbf{U}_i\mathbf{D}\mathbf{U}_j{}'$ need to be computed, where $\mathbf{U}_i$ denotes the $m_i \times \widehat{d}$ block in $\mathbf{U}$ corresponding to view i and $\mathbf{U}_j$ denotes the $m_j \times \widehat{d}$ block in $\mathbf{U}$ corresponding to view j. Therefore we only need to store the matrix $\mathbf{U}\mathbf{D}^{\frac{1}{2}}$ instead of $\widehat{\mathbf{Z}}_{\widehat{d}}$, and this observation considerably reduces the storage space necessary to run the algorithm.
Note also that the projection step (either via the PPPA or via the approximate strategy) can be performed in parallel, since each view pair is independent from the others, thus speeding up the process.

3.3.1 Computational Complexity

The core of the algorithm is the eigenvalue decomposition of a sparse matrix, which is computed with a Lanczos method. The method is iterative: every iteration is $O(m)$ [60], but the number of iterations cannot be bounded by a constant.
The second and final step of our algorithm is the projection onto permutation matrices. For a matrix of dimension r, computing the nearest permutation via PPPA takes $O(r^3)$ time [21]. As for the approximate strategy, we have to sum the cost for computing the maximums over rows/columns, which is $O(r^2)$, and the cost for sorting such values, which is $O(r \log(r))$, resulting in $O(r^2)$. Since the average dimension of each block in $\widehat{\mathbf{Z}}$ is m/n and the number of blocks is n^2, the total cost for the projection step is $O(m^3/n)$ for MatchEIG-CP and $O(m^2)$ for MatchEIG.

3.3.2 Numerical Example

For a better comprehension of the proposed method, we report a simple numerical example, showing also the intermediate results of the algorithm. The analyzed situation is illustrated in Figure 3.2, where a total number of $d = 5$ points are seen in $n = 3$ views. Please note that views 1 and 2 see only four points.

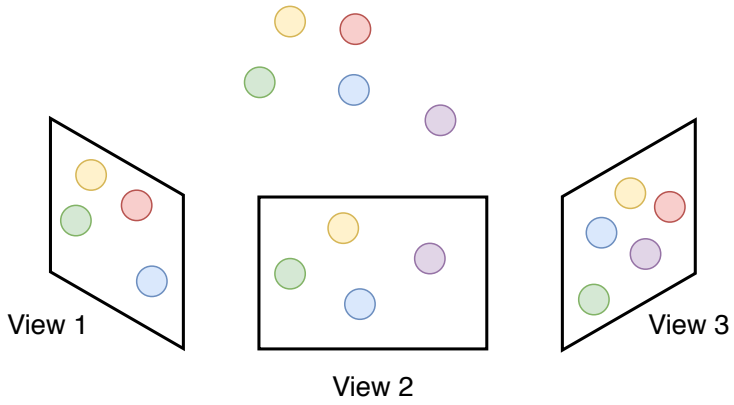


FIGURE 3.2: Example of multi-view matching. A total number of $d = 5$ points are seen in $n = 3$ views. Please note that not all the points are seen in all views.

The ground-truth matrix $\mathbf{X}_{\text{gt}}$ of absolute permutations, that describes the correspondences between the keypoints in each view and those in the universe, is:

$$\mathbf{X}_{\text{gt}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.14}$$

which corresponds to the ground-truth matrix of relative permutations:

$$\mathbf{Z}_{\text{gt}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.15}$$

Simulating errors in the pairwise correspondences, we assume that the noisy input matrix is

$$\hat{\mathbf{Z}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.16}$$

where wrong entries are highlighted in red. Following the procedure described in Section 3.3, we first determine the eigenvalue decomposition of $\hat{\mathbf{Z}}$, obtaining matrix $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} 0 & 0.60 & 0 & 0 & 0 \\ 0.58 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.56 & 0.44 & 0 \\ 0 & 0 & 0 & 0.394 & 0 \\ 0.58 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.39 & 0.31 & 0 \\ 0 & 0 & 0 & 0.56 & 0 \\ 0 & 0.60 & 0 & 0 & 0.37 \\ 0 & 0.37 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 & 0.60 \\ 0 & 0 & 0 & 0.39 & 0 \\ 0.58 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.39 & 0.31 & 0 \end{bmatrix}. \tag{3.17}$$

Then, the product $\mathbf{UDU}'$ is computed:

$$\mathbf{UDU}' = \begin{bmatrix} 1.17 & 0 & 0 & 0 & 0 & 0 & 0 & 0.72 & 0.95 & 0.22 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1.21 & 0 & 0 & 0.85 & 0 & 0 & 0 & 0 & 0 & 0 & 0.85 \\ 0 & 0 & 0 & 0.60 & 0 & 0 & 0.85 & 0 & 0 & 0 & 0.60 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.85 & 0 & 0 & 0.60 & 0 & 0 & 0 & 0 & 0 & 0 & 0.60 \\ 0 & 0 & 0 & 0.85 & 0 & 0 & 1.21 & 0 & 0 & 0 & 0.85 & 0 & 0 \\ 0.72 & 0 & 0 & 0 & 0 & 0 & 0 & 1.17 & 0.22 & 0.95 & 0 & 0 & 0 \\ 0.95 & 0 & 0 & 0 & 0 & 0 & 0 & 0.22 & 0.95 & 0 & 0 & 0 & 0 \\ 0.22 & 0 & 0 & 0 & 0 & 0 & 0 & 0.95 & 0 & 0.95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.604 & 0 & 0 & 0.85 & 0 & 0 & 0 & 0.60 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.85 & 0 & 0 & 0.60 & 0 & 0 & 0 & 0 & 0 & 0 & 0.60 \end{bmatrix} \tag{3.18}$$

and, after setting to zero all the entries smaller that 0.25, the desired output matrix $\mathbf{Z}$ is obtained by projecting each block of $\mathbf{UDU}'$ onto the space of permutation matrices:

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.19}$$

Please note that in the noisy input matrix $\hat{\mathbf{Z}}$ three matches are incorrect, whereas the output of our algorithm has only one incorrect match. The entries of the recovered matrix $\mathbf{Z}$ that differs from the ones of the input matrix $\hat{\mathbf{Z}}$ are highlighted in red.

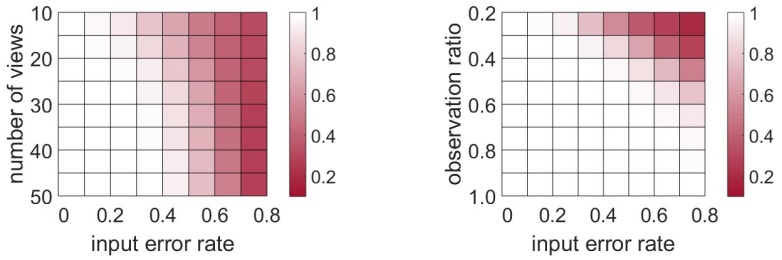
3.4 Experiments

In order to evaluate the performance of the proposed method, we ran experiments on both synthetic and real datasets. In the synthetic experiments, performances have been measured in terms of *precision* (number of correct matches returned divided by the number of matches returned) and *recall* (number of correct matches returned divided by the number of correct matches that should have been returned). In order to provide a single figure of merit we computed the *F-score* (twice the product of precision and recall divided by their sum), which is a measure of accuracy and reaches its best value at 1 and worst at 0. In the real experiments the number of matches that should have been returned is not known, hence only the precision can be computed.
Results in terms of accuracy (or precision) and computing time are compared with methods which, as ours, first compute pairwise matches and then jointly update them without involving the keypoints, namely MatchALS [166] and Spectral [124]. The MatchALS code is available online[1] and the Spectral code is courtesy of the authors of [166]. The method described in [28] has already been shown to have accuracy comparable with [166] with a much higher computing time. For this reason we did not consider it in the experiments. All the algorithms are implemented in Matlab and tested on a PC with an Intel Core i5-4200M CPU @ 2.50GHz and 8GB RAM. Our implementation of MatchEIG is available on the web[2].

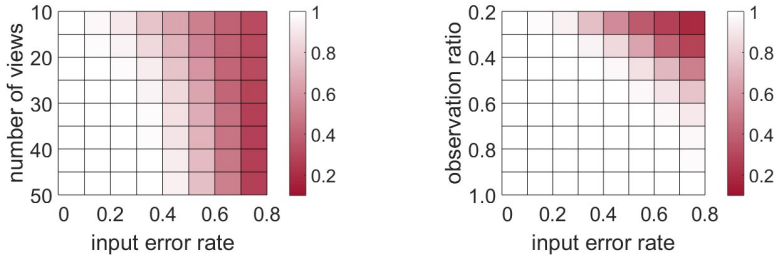
3.4.1 Synthetic Experiments

For the synthetic case, the size of the universe was set to $d = 100$, while the number of views varied from $n = 10$ to $n = 50$. The observation ratio r_o, i.e., the probability that a keypoint is seen in a view, decreased from 1 (that corresponds to total permutations) to 0.2. After generating ground-truth absolute permutations, pairwise matches were computed from Equation (3.6), and random errors were added to relative permutations by switching two matches, removing true matches or adding false ones. In the experiments the input error rate r_e, i.e., the ratio of mismatches, varied from 0 to 0.8. For each configuration the test was run 10 times and the average F-score was evaluated.

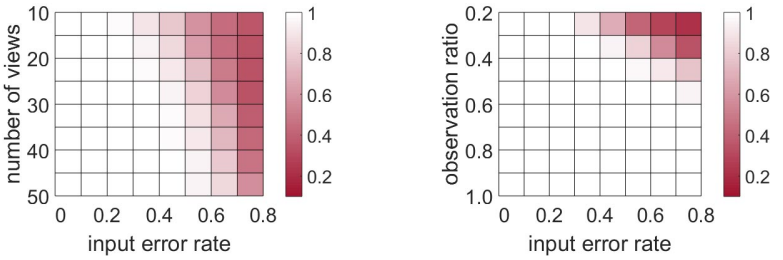
[1] https://fling.seas.upenn.edu/ xiaowz/dynamic/wordpress/matching/
[2] http://www.diegm.uniud.it/fusiello/demo/mvm/



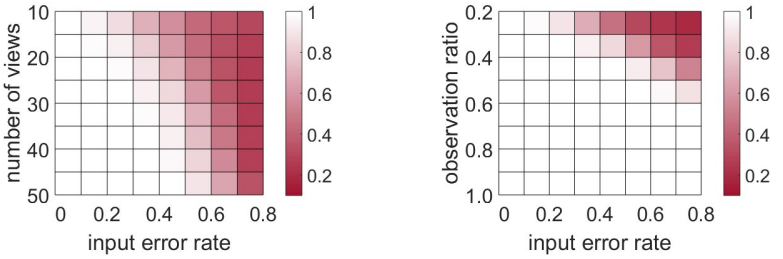
(A) MatchEIG



(B) MatchEIG-CP



(C) MatchALS [166]



(D) Spectral [124]

Figure 3.3: F-score for the competing methods (the higher the better). In the left column, the number of views n and the input error rate r_e are varying, while the observation ratio r_o is constant and equal to 0.4. In the right column, r_o and r_e are varying, while $n = 30$. In all the experiments, the size of the universe d is set to 100.

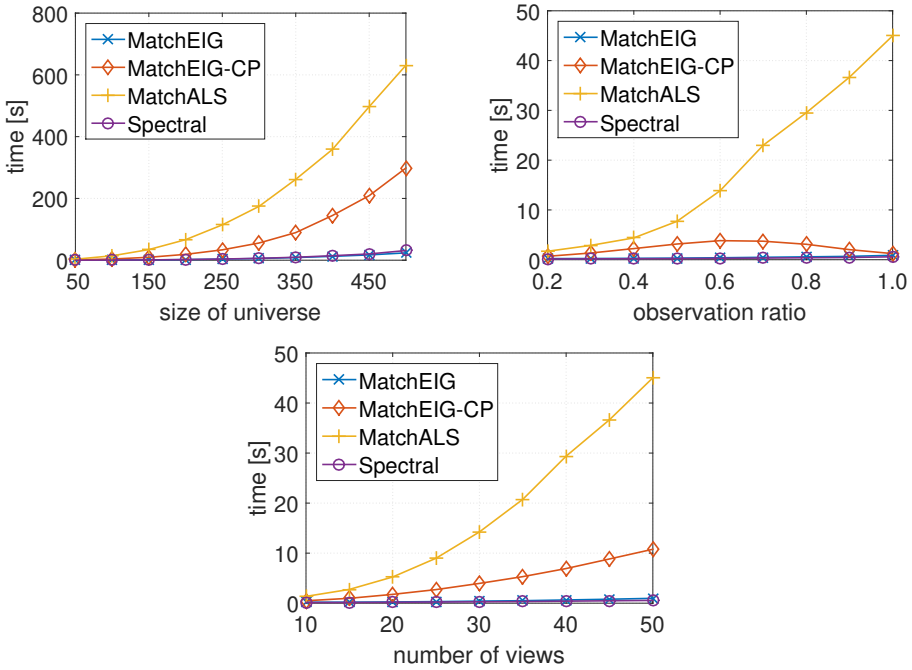


FIGURE 3.4: Average computing time against size of the universe, observation ratio and number of views. MATCHEIG and SPECTRAL [124] show similar performance, while MATCHALS [166] is the slowest algorithm.

Synthetic experiments were run providing the true rank (equal to the size of the universe d) to all the methods (MATCHALS uses it to compute the parameter $k = 2d$, as suggested in [166]). Moreover, for MATCHALS we fixed the number of possible iterations to 100 and we chose the values proposed by the authors for all the other parameters.
Results in terms of accuracy are illustrated in Figure 3.3. The analyzed methods show similar behaviors, achieving high accuracy rates even in the presence of high noise contamination, especially when the number of views is large. Please note that there is no loss of accuracy when using the approximated projection onto permutations (MATCHEIG) with respect to the exact closest permutation, as done by MATCHEIG-CP.
We also evaluated the computing time, varying the size of the universe, the observation ratio and the number of views. Figure 3.4 shows that MATCHEIG and SPECTRAL are the fastest algorithms, while MATCHALS is on average an order of magnitude slower. Comparing MATCHEIG and MATCHEIG-CP, it is clear that the PPPA is computationally much more expensive than the approximate procedure. Since the accuracy provided by the two projection algorithms is the same, as demonstrated by Figure 3.3, only the fastest version (MATCHEIG) will be used in real experiments.

Sensitivity to Rank Estimate

All the evaluated methods require as input an estimate of the size of the universe, which corresponds to the rank of the ground-truth matrix $\mathbf{Z}$. However, when the input matrix $\widehat{\mathbf{Z}}$ is noisy, estimating this rank can be difficult [166], hence the sensitivity of a method to the estimated rank $\hat{d}$ becomes crucial. As demonstrated in Figure 3.5, MATCHEIG and MATCHALS give good results whenever $\widehat{d} \geq d$. The SPECTRAL method, instead, is extremely sensitive to this parameter and performs well only if $\widehat{d} = d$.

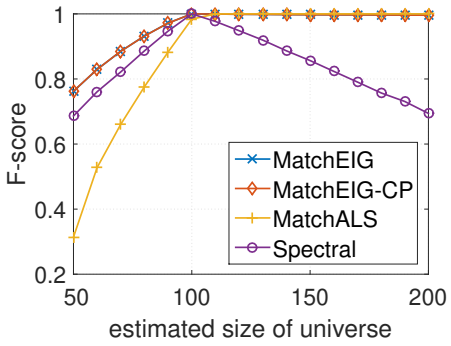


FIGURE 3.5: F-score versus estimated size of the universe $\widehat{d}$. The true size of universe is $d = 100$, $n = 30$, $r_o = 0.6$, and $r_e = 0.1$.

3.4.2 Real Experiments

For evaluating the applicability of the proposed method in real scenarios, we tested MATCHEIG, MATCHALS and SPECTRAL on popular benchmark datasets (Graffiti, EPFL [142] and Middlebury [137] datasets) with up to 363 images. To generate the input to the algorithms, a set of keypoints was first extracted in each image with SIFT [106] using the VLFeat library[3]. Subsequently, correspondences between pairs of images were established using nearest neighbor and ratio test as in [106] and refined using RANSAC [52]. Finally, keypoints with matches in less than two images were removed, since they are not significant in joint matching.
In these experiments the universe set is not known, so we estimated its dimension as twice the average number of keypoints present in each image, and provided all the methods with this estimate. For the same reason only the precision value is reported, together with the absolute number of correct matches returned, that can give some relative indication on the recall. For MATCHALS, we set the parameter m'/m to 0.7 (see [166]) to take into consideration the presence of isolated keypoints in real images, as suggested by the authors. The maximum number of iterations was fixed to 100 and the default values were used for all the other parameters.
Matches are considered correct if the corresponding point is located within a given distance threshold from what is predicted. In the case of the Graffiti datasets the ground-truth homographies allow to predict the position of the point, whereas for EPFL and Middlebury datasets the ground-truth cameras allow to predict the epipolar line where the corresponding point should lie. In both cases the threshold (in pixels) has been set equal to 0.01 times the image diagonal.

Graffiti Dataset

The Graffiti datasets[4] consist of eight sequences with six images each, showing different structured and textured planar scenes. Each dataset is characterized by different image transformations, e.g., change of viewpoint, zoom, blur, illumination and rotation.
Table 3.1 shows the performances of joint matching on these datasets. The input error is already small, and all the methods achieve a precision higher than 95%, with a comparable number of correct matches returned, confirming the results of the synthetic experiments. This dataset is not particularly challenging, as it consists of few images with little differences in visual content among them; however it is widely used for testing multi-view matching algorithms, therefore it has been included here. A consideration can be made regarding the computing time, with MATCHEIG being on average 5 times and 1.5 times faster than MATCHALS and SPECTRAL, respectively.

[3] http://www.vlfeat.org/
[4] http://www.robots.ox.ac.uk/ vgg/data/data-aff.html

			Input	MatchEIG			MatchALS [166]			Spectral [124]		
Dataset	n	$\widehat{d}$	PR [%]	PR [%]	CM	T [s]	PR [%]	CM	T [s]	PR [%]	CM	T [s]
Graffiti	6	382	93.85	95.63	678	12	96.02	747	91	**96.24**	614	24
Boat	6	578	98.75	**99.03**	1731	35	98.63	1154	209	98.66	1544	45
Bark	6	684	99.71	99.77	1323	58	**100.00**	914	307	99.67	1225	77
Ubc	6	891	99.36	99.66	3828	139	**99.67**	3030	681	99.63	3533	207
Trees	6	1015	98.48	98.46	2885	184	**98.73**	2484	971	98.51	2719	255
Light	6	1113	98.67	99.39	4105	336	**99.43**	3287	1258	98.95	2253	416
Wall	6	1236	99.40	99.45	3253	341	99.38	2871	1644	**99.57**	2760	456
Bikes	6	1759	99.12	99.39	4866	954	**99.53**	4228	3828	99.26	4149	1298

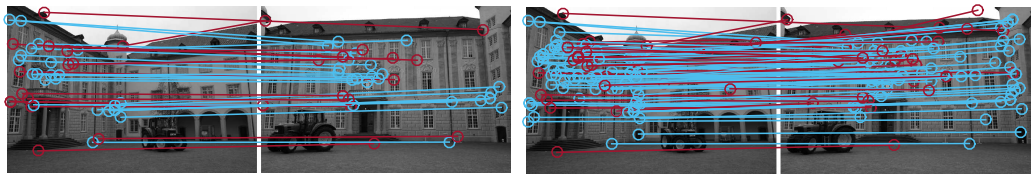
Table 3.1: Results on the Graffiti dataset. n is the number of images, PR is the precision, CM is the number of correct matches returned, T is the time expressed in seconds.

			Input	MatchEIG			MatchALS [166]			Spectral [124]		
Dataset	n	$\widehat{d}$	PR [%]	PR [%]	CM	T [m]	PR [%]	CM	T [m]	PR [%]	CM	T [m]
Herz-Jesu-P8	8	386	94.40	**95.08**	4545	< 1	94.87	4047	2	94.41	3987	< 1
Entry-P10	10	432	75.11	**79.24**	5978	5	74.17	5726	4	76.10	6236	4
Fountain-P11	11	374	94.35	**94.70**	6988	3	94.15	6717	3	91.92	7849	3
Castle-P19	19	314	70.29	**75.21**	5109	3	66.22	7014	9	34.41	7605	3
Herz-Jesu-P25	25	517	90.20	**93.45**	25120	7	89.23	32528	41	47.86	32876	8
Castle-P30	30	445	72.32	**81.01**	16754	8	68.92	24844	57	34.67	25884	10
Temple Ring	47	396	73.72	**88.25**	18426	6	55.91	40096	260	28.99	46432	7
Dino Ring	48	340	75.37	**92.11**	23406	2	66.66	44215	94	34.49	48979	3
Temple	312	689	55.50	**89.06**	$3.8 \cdot 10^5$	153	–	–	–	14.56	$1.6 \cdot 10^6$	228
Dino	363	493	63.48	**95.66**	$8.6 \cdot 10^5$	88	–	–	–	18.97	$2.2 \cdot 10^6$	111

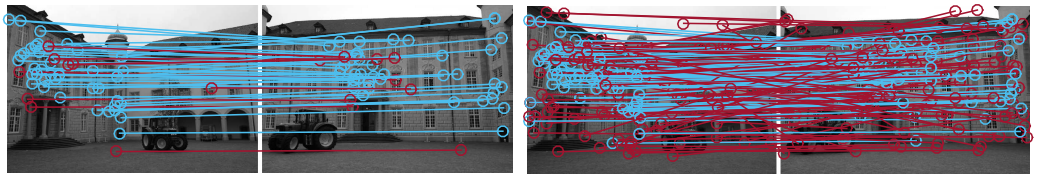
Table 3.2: Results on the EPFL [142] and Middlebury datasets [137]. n is the number of images, PR is the precision, CM is the number of correct matches returned, T is the time expressed in minutes.

EPFL Dense Multi-View Stereo Test Images

A more challenging benchmark are the EPFL dense multi-view stereo test images [142]. These are six image sets representing outdoor scenes, composed of a number of images that varies from 8 to 30. Performing multi-view matching on the *Entry-P10* and *Castle-P** sequences is particularly difficult due to the presence of repetitive structures. For practical reasons, we rescaled images to 20% of the original size.
As can be seen in the upper part of Table 3.2, the quality of the input matches is lower than that on the Graffiti datasets and it can be significantly improved by joint matching. On these datasets, MatchEIG outperforms the other algorithms, both in terms of precision and computing time.
Note that on some sequences Spectral achieves a very low precision, probably due to the fact that the chosen value of $\widehat{d}$ is not an accurate estimate of d. Figure 3.6 shows a representative example of the results obtained by the competing methods. With respect to pairwise matching, joint matching reduces the number of false matches and complete the matches with new ones retrieved indirectly via loop closure.



(a) Input (b) MatchALS

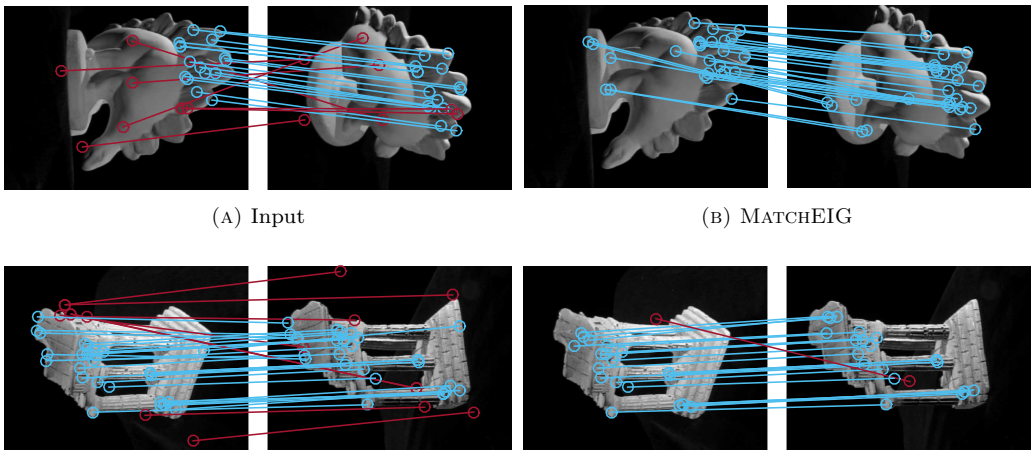


(c) MatchEIG (d) Spectral

Figure 3.6: Representation of the matches between two images of the *Castle-P19* set [142]. Wrong matches are drawn in red.

Middlebury Multi-View Stereo Dataset

Multi-view matching often needs to be applied on sets of hundreds of images. Therefore we selected the two largest sets among the Middlebury multi-view stereo datasets [137] to evaluate the practical applicability of MatchEIG, MatchALS and Spectral. The *Temple* and the *Dino* sets consist of 312 and 363 views respectively, sampled on a hemisphere. We also considered the smaller *Dino Ring* and *Temple Ring* sets, that contain approximately 50 views each, sampled on a ring around the object.
Results are reported at the bottom of Table 3.2. In these cases, the initial pairwise matching provides a noisy input, upon which only MatchEIG is able to improve. In fact, on the smaller datasets, MatchALS and Spectral produced results worse than the input (MatchALS taking a very long time), while on larger sets MatchALS did not achieve a solution: due to a memory request that exceeded the available space it aborted prematurely. Figure 3.7 gives a visual demonstration of the improvements that can be achieved through MatchEIG with respect to pairwise matching.
The precision/recall trade-off in MatchEIG is affected by the threshold t, which controls the degree of "partiality" of the permutations. A small t yields more matches, but with low precision. On the other hand, a high t produces few but extremely reliable matches.



(A) Input (B) MatchEIG

(C) Input (D) MatchEIG

Figure 3.7: Representation of the matches between two images of the *Dino* and *Temple* sets [137]. Wrong matches are drawn in red.

Figure 3.8 shows the precision vs CM curve (recall is not available) on a typical dataset: MatchALS has an almost constant precision for a wide range of the controlling parameter (m'/m), whereas in the left part of the curve MatchEIG has a superior precision, and this is where we suggest our algorithm should be used. We tuned the value of t empirically so as to be in this part of the curve in all the considered datasets. This choice is motivated by applications: with respect to the null hypothesis that a match is an inlier, a Type I error (rejecting an inlier) is less serious than a Type II error (not rejecting an outlier) when using matches to compute geometrical models.

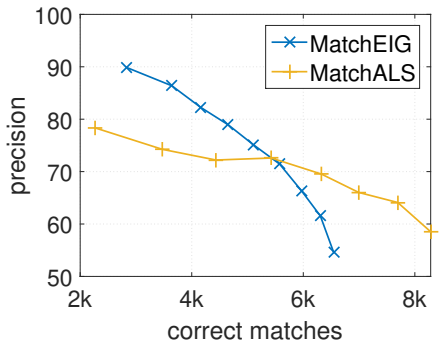


Figure 3.8: Precision vs CM curve on the *Castle-P19* set [142].

3.5 Conclusion

In this chapter we presented a closed-form solution to (joint) multi-view matching, based on a spectral decomposition. MatchEIG handles realistic situations, such as partial permutations and image sets of unprecedented size in the literature. While experiments on simulated data – and one easy real dataset – only highlight the superior computational efficiency of the method, with the accuracy being on a par with the others, on challenging real datasets MatchEIG outperforms the competing methods both in speed and precision.

Applications for a practical multi-view matching are countless, which motivates this study as well as future developments. Chapter 4 will show how the proposed algorithm can be profitably used in the point cloud registration process.

Chapter 4

Closest Point Variation of the Generalized Procrustes Analysis

The aim of this chapter is to extend the *Generalized Procrustes Analysis* (GPA) to the case in which correspondences among matrix elements are unknown. In particular, we embed the matching step of Chapter 3 inside the GPA iterative scheme, obtaining a more generalized formulation that can be profitably applied to the point cloud registration problem. Moreover, we show that our method generalizes the well known Iterative Closest Point (ICP) to the case of multiple point sets.

4.1 Introduction

Procrustes models presented in the previous chapters allow to compute transformations only between two sets. In order to simultaneously consider $m > 2$ matrix configurations, one must resort to the so called *Generalized Procrustes Analysis* (GPA), proposed in [66, 61]. In this case, all the matrices are independently and simultaneously rotated, scaled and translated so to satisfy a prefixed objective function. Figure 4.1 gives a geometric interpretation of the problem. Given m matrices $\mathbf{A}_1, \ldots, \mathbf{A}_m$, each one containing the coordinates of the same p points in $\mathbb{R}^k$ defined in m different reference systems, GPA retrieves for each matrix $\mathbf{A}_i$ the similarity transformation that allows to obtain the best alignment with respect to all the other ones or, equivalently, the transformation that minimizes the residual with respect to the unknown matrix $\mathbf{Z}$, containing the true coordinates of the p points defined into a common mean coordinate frame.

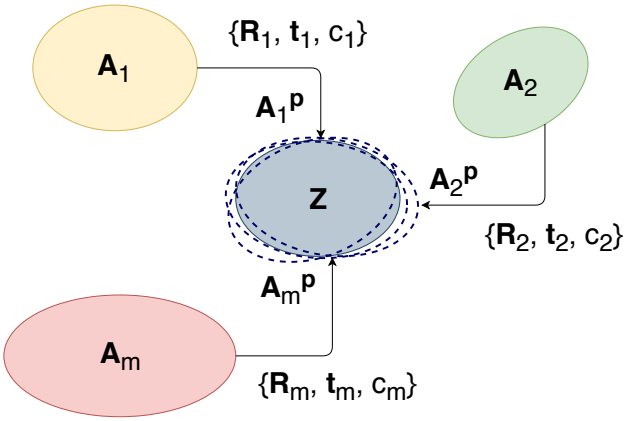


FIGURE 4.1: The Generalized Procrustes Analysis concept.

In the last years, GPA has been applied to several fields, including photogrammetry and laser scanning. In [39] , e.g., it is used for the alignment of different 3D models deriving from pairs

of images, whereas in [12] it is adopted to register several point clouds acquired by a laser scanner, relying on manually selected corresponding tie points. A main limitation emerges from these works: to apply the GPA solution, correspondences among points must be known in advance. This restriction has been partially overcome in [151], that takes advantage of the GPA to define an iterative closest point algorithm that can register multiple views in a simultaneous way. To broaden the GPA application fields, avoiding the manual search of the point matches, it is therefore necessary to introduce inside the GPA framework the automatically estimation of the correspondences.

4.1.1 Contribution

In this chapter we first review the GPA model and the solutions that can be found in the literature (Section 4.2). Then, in Section 4.3, we describe an extension of the GPA, that we call *Closest Point Generalized Procrustes Analysis* (CP-GPA), that embeds the correspondence estimation among point sets inside the classical framework of the GPA. The well known Iterative Closest Point (ICP) algorithm [16] is also described as a special case of the CP-GPA. Finally, in Section 4.4 we show some preliminary results of our method to the registration of multiple point clouds.

4.2 Original GPA Solutions

Generalized Procrustes Analysis allows to simultaneously align m matrices $\mathbf{A}_1, \ldots, \mathbf{A}_m$, which contain the same set of p points, whose coordinates are expressed in m different k-dimensional reference systems, so as to minimize the cost function:

$$\begin{aligned} F &= \sum_{i<j}^{m} \left\| \mathbf{A}_i^p - \mathbf{A}_j^p \right\| \\ &= \operatorname{tr} \sum_{i<j}^{m} \left[(c_i \mathbf{A}_i \mathbf{R}_i + \mathbf{j}\mathbf{t}_i) - \left(c_j \mathbf{A}_j \mathbf{R}_j + \mathbf{j}\mathbf{t}_j \right) \right]' \left[(c_i \mathbf{A}_i \mathbf{R}_i + \mathbf{j}\mathbf{t}_i) - \left(c_j \mathbf{A}_j \mathbf{R}_j + \mathbf{j}\mathbf{t}_j \right) \right] \end{aligned} \tag{4.1}$$

where $\mathbf{A}_i^p = c_i \mathbf{A}_i \mathbf{R}_i + \mathbf{j}\mathbf{t}_i$ represents the transformed matrix $\mathbf{A}_i$, while c_i, $\mathbf{t}_i$ and $\mathbf{R}_i$ ($i = 1 \ldots m$) are the unknown parameters of the similarity transformation. The solution can be retrieved in two different ways, described in the following.
The original formulation and the first solution of (4.1) are due to Gower [66], based on a iterative scheme previously proposed in [92]. According to this approach, the minimum condition is reached through the following algorithm:

1. Solve for all the translations by computing the barycenter of the matrices $\mathbf{A}_i$ ($i = 1 \ldots m$), multiplying them by $\mathbf{I} - \dfrac{\mathbf{j}\mathbf{j}'}{p}$;
2. Solve for the rotations applying the following iterative method, based on consecutive OPA solutions (Section 2.3.1). At each iteration t, rotate every $\mathbf{A}_i^p \mathbf{R}_i^{(t-1)}$ matrix with respect to the sum of the remaining ones:
   a) Matrix $\mathbf{A}_1^p \mathbf{R}_1^{(t-1)}$ is rotated towards $\sum_{j=2}^{m} \mathbf{A}_j^p \mathbf{R}_j^{(t-1)}$, so to obtain $\mathbf{A}_1^p \mathbf{R}_1^{(t)}$
   b) Matrix $\mathbf{A}_2^p \mathbf{R}_2^{(t-1)}$ is rotated towards $\mathbf{A}_1^p \mathbf{R}_1^{(t)} + \sum_{j=3}^{m} \mathbf{A}_j^p \mathbf{R}_j^{(t-1)}$, so to obtain $\mathbf{A}_2^p \mathbf{R}_2^{(t)}$
   c) Matrix $\mathbf{A}_m^p \mathbf{R}_m^{(t-1)}$ is rotated towards $\sum_{j=2}^{m-1} \mathbf{A}_j^p \mathbf{R}_j^{(t)}$, so to obtain $\mathbf{A}_m^p \mathbf{R}_m^{(t)}$
   d) Iterate from step a) until internal convergence.

3. Solve for the scale factors, computing the correlation matrix of the vector $\mathrm{vec}\,(\mathbf{A}_i^p)$, with $\mathbf{f}$ the eigenvector corresponding to the largest eigenvalue. The scale factor c_i is given by:

$$c_i = \left\{ \left(\sum_{j=1}^{m} \left\| \mathbf{A}_j^p \right\|^2 \right) / \left\| \mathbf{A}_i^p \right\|^2 \right\}^{\frac{1}{2}} f_i \tag{4.2}$$

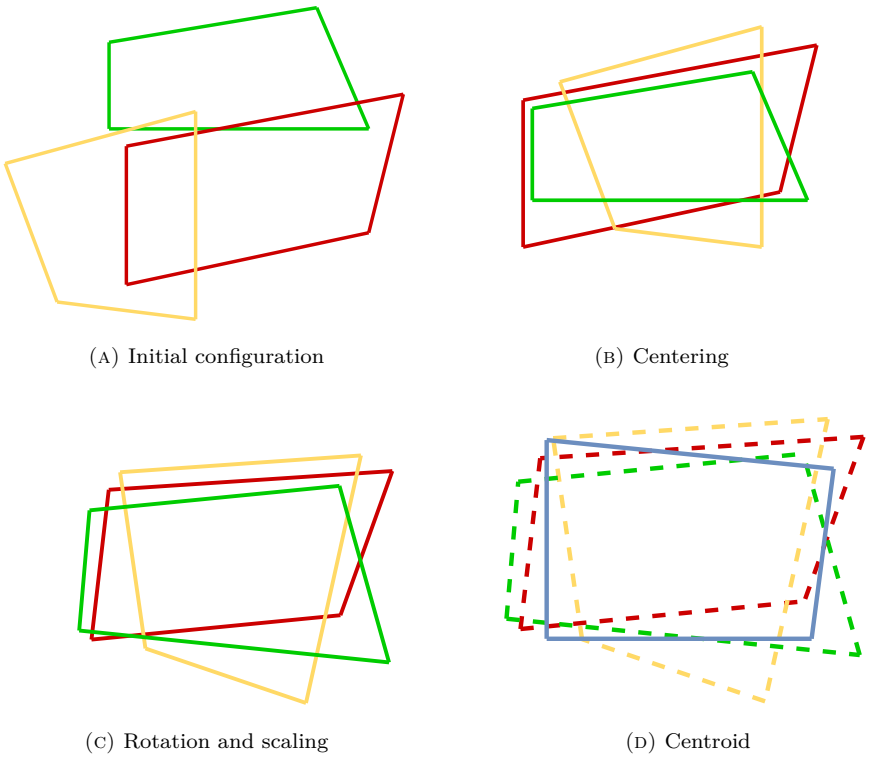
   where f_i is the i-th element of eigenvector $\mathbf{f}$.

4. Iterate from step 2. until global convergence.

Please note that, at each step of the procedure $\mathbf{A}_i^p$ is updated, until the final value reached at convergence. At the end it is possible to determine the so called *centroid* of the matrices, given by the arithmetic mean of all the sets $\mathbf{A}_i^p$:

$$\mathbf{C} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{A}_i^p. \tag{4.3}$$

Figure 4.2 offers a graphical description of the GPA solution and of the meaning of the centroid. In this example there are three sets $\mathbf{A}_i$, each one represented by a quadrilateral. Figure 4.2(A) shows the initial conditions, while Figure 4.2(B) illustrates the centering effect: the sets are overlapped in correspondence of their barycenter. Figure 4.2(C) emphasizes a step of the rotation and scaling sequence maintaining unchanged the common barycenter until the satisfaction of the prefixed minimum condition. Figure 4.2(D) shows in blue the *centroid*, that is *the mean shape* of the final transformed sets.



(A) Initial configuration (B) Centering

(C) Rotation and scaling (D) Centroid

FIGURE 4.2: Visualization of the GPA procedure.

An alternative and more efficient solution of the GPA problem is described in [34]. Instead of searching the transformation parameters c_i, $\mathbf{t}_i$ and $\mathbf{R}_i$ that minimize Equation (4.1), the

cost function is rewritten in terms of the centroid, expressed by (4.2). More in detail, the author starts from the proposition:

Proposition 4.1. *[34] Given m points in a k-dimensional space, the sum of the squared distances among the m points is equal to m times the sum of the squared distances among the m points and their centroid.*

That is:

$$\sum_{i<j}^{m} (\mathbf{x}_i - \mathbf{x}_j)' (\mathbf{x}_i - \mathbf{x}_j) = m \sum_{i}^{m} (\mathbf{c} - \mathbf{x}_i)' (\mathbf{c} - \mathbf{x}_i) \tag{4.4}$$

where $\mathbf{x}_i$, $\mathbf{x}_j$ represent the generic column vectors containing the coordinates of the m points and $\mathbf{c} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$ is the centroid.

Proof. The first term of (4.4) can be expressed also as:

$$\sum_{i<j}^{m} (\mathbf{x}_i - \mathbf{x}_j)' (\mathbf{x}_i - \mathbf{x}_j) = \sum_{i<j}^{m} (\mathbf{x}_i{}'\mathbf{x}_i + \mathbf{x}_j{}'\mathbf{x}_j) - 2\sum_{i<j}^{m} \mathbf{x}_i{}'\mathbf{x}_j = (m-1)\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i - 2\sum_{i<j}^{m} \mathbf{x}_i{}'\mathbf{x}_j.$$

The right-hand term of (4.4) can instead be rewritten as:

$$\begin{aligned} m\sum_{i}^{m} (\mathbf{c} - \mathbf{x}_i)' (\mathbf{c} - \mathbf{x}_i) =& m\sum_{i}^{m} \mathbf{c}'\mathbf{c} + m\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i - 2m\mathbf{c}'\sum_{i}^{m} \mathbf{x}_i \\ =& \sum_{j}^{m} \mathbf{x}_j{}' \sum_{i}^{m} \mathbf{x}_i + m\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i - 2\sum_{i}^{m} \mathbf{x}_i{}' \sum_{i}^{m} \mathbf{x}_i \end{aligned}$$

from which one obtains

$$\begin{aligned} m\sum_{i}^{m} (\mathbf{c} - \mathbf{x}_i)' (\mathbf{c} - \mathbf{x}_i) =& 2\sum_{i<j}^{m} \mathbf{x}_i{}'\mathbf{x}_j + (1+m)\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i - 2\left(\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i + 2\sum_{i<j}^{m} \mathbf{x}_i{}'\mathbf{x}_j\right) \\ =& (m-1)\sum_{i}^{m} \mathbf{x}_i{}'\mathbf{x}_i - 2\sum_{i<j}^{m} \mathbf{x}_i{}'\mathbf{x}_j \end{aligned}$$

that completes the proof. □

The proposition can be extended to the case of m matrices, each one containing p points in $\mathbb{R}^k$. The formulation is equivalent when applied to matrices instead of vectors. This allows us to write:

$$\sum_{i<j}^{m} \left\|\mathbf{A}_i^p - \mathbf{A}_j^p\right\|^2 = m\sum_{i}^{m} \|\mathbf{A}_i^p - \mathbf{C}\|^2. \tag{4.5}$$

So, the objective function (4.1) becomes:

$$F = m\sum_{i}^{m} \|\mathbf{A}_i^p - \mathbf{C}\|^2 = m\sum_{i}^{m} \operatorname{tr}(\mathbf{A}_i^p - \mathbf{C})' (\mathbf{A}_i^p - \mathbf{C}). \tag{4.6}$$

The GPA problem can therefore be solved through the following iterative scheme, that leads to significant advantages in terms of simplicity and efficiency with respect to the previously described algorithm.

1. For each matrix $\mathbf{A}_i$, compute via EOPA (Section 2.3.2) the similarity transformation that best aligns matrix $\mathbf{A}_i$ to the centroid $\mathbf{C}$, obtaining $\mathbf{A}_i^p = c_i\mathbf{A}_i\mathbf{R}_i + \mathbf{jt}_i$;
2. Update the centroid $\mathbf{C}$ with Equation (4.3);

3. Iterate from step 1. until convergence, i.e., until the stabilization of the centroid $\mathbf{C}$.

4.2.1 Weighted GPA with Missing Points

GPA can be applied also to the case in which the m matrices do not contain all p points, i.e., one or more rows are missing from each matrix.
The problem was completely solved in [34], where it is demonstrated that Proposition 4.1 can be generalized premultiplying each $\mathbf{A}_i^p$ by a diagonal matrix $\mathbf{D}_i$, that is a matrix having 0 in correspondence of the missing values and 1 otherwise, obtaining:

$$\sum_{i=1}^{m} \operatorname{tr}\left(\mathbf{A}_i^p - \mathbf{C}\right)' \mathbf{D}_i \left(\mathbf{A}_i^p - \mathbf{C}\right) = \sum_{i<j}^{m} \operatorname{tr}\left(\mathbf{A}_i^p - \mathbf{A}_j^p\right)' \mathbf{D}_i \mathbf{D}_j \left(\sum_{k=1}^{m} \mathbf{D}_k\right)^{-1} \left(\mathbf{A}_i^p - \mathbf{A}_j^p\right) \tag{4.7}$$

where:

$$\mathbf{C} = \left(\sum_{k=1}^{m} \mathbf{D}_k\right)^{-1} \sum_{i=1}^{m} \mathbf{D}_i \mathbf{A}_i^p. \tag{4.8}$$

Thanks to this formulation, sets with missing points can be easily managed. In fact, Equation (4.7) leads to simply substitute EOPA in step 1. of the previous algorithm with its weighted variant, i.e., WEOPA (Section 2.3.3).
Moreover, the binary matrix $\mathbf{D}_i$ can be replaced by a weight matrix, having 0 in correspondence of the missing points and real values otherwise. This allows to differently weight each point of each set.
Figure 4.3 geometrically explains the GPA solution with missing values. The complete point sets would represent three hexagons. In two cases the required number of points is missing and the figures become a rectangle and a trapezium, respectively. The GPA is carried out according to the corresponding vertices of the figures.

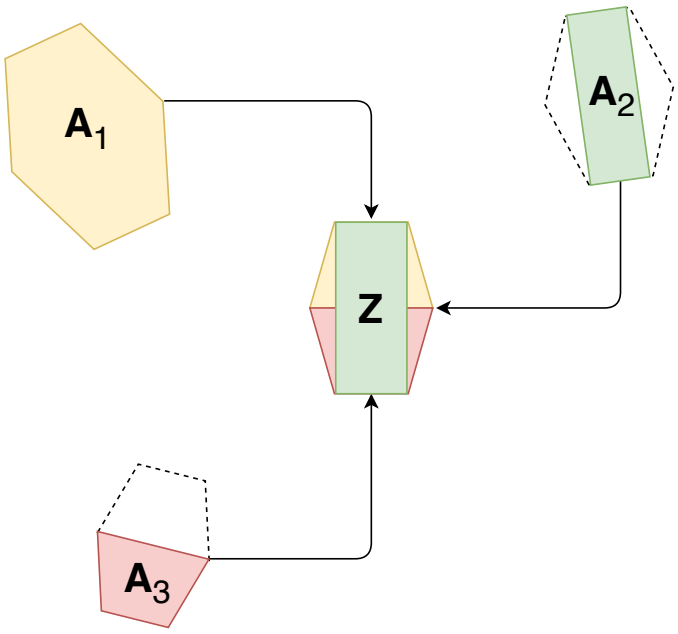


FIGURE 4.3: Geometric representation of GPA with missing points.

4.3 Closest Point Generalized Procrustes Analysis

As already mentioned, previous solutions assume that the correspondences between matrix elements are known in advanced. However, this requirement is usually hardly satisfied and a manual identification of the correspondences is often unfeasible. It appears therefore useful to extend the GPA solution to the case in which correspondences are missing, embedding the matching step within the GPA iterative process.
The general scheme of this, called *Closest Point Generalized Procrustes Analysis* (CP-GPA), is the following:

1. Heuristically establish *tentative* correspondences among points in different views;
2. Compute the centroid using Equation (4.8);
3. Align each view to the centroid using WEOPA (Section 2.3.3);
4. Iterate from step 1. until convergence.

Step 1. can be instantiated with any heuristic, but usually it is one informed by the "closest point" relation, defined as follows: given two sets A and B, two points $a \in A$ and $b \in B$ are in the relation $cp(a, b)$ iff b has the shortest distance (a notion of distance must be defined, usually the Euclidean one) to a among all the points in B. Please note that cp is not symmetric ($cp(a, b) \neq cp(b, a)$) and not transitive ($cp(a, b) \wedge cp(b, c) \nRightarrow cp(a, c)$).
Better results are usually obtained by considering the "mutual closest-point" symmetric relation, that can be formally defined as $cp^* = cp \cap cp^{-1}$ (this is also the greatest symmetric subset of cp).
When working with multiple point clouds, the problem is how to extend pairwise matches to multiple-view correspondences. A straightforward approach is taking the transitive closure of cp, cp^*. The points in the relation constitute a *track*, in the same way as (a, b) with $cp^*(a, b)$ is a match.
Another way to define the tracks is the following: consider the undirected graph where points are the nodes and edges represent matches; a track is a connected component of that graph.
The heuristic we propose for step 1. include our multi-view matching algorithm (Chapter 3) and the subsequent detection of closed-loop consistency violation: if vertices of the aforementioned graph are labeled with the view the points belong to, an inconsistency arises when in a track a label occurs more than once.
In summary, our heuristic for step 1. of CP-GPA is the following:

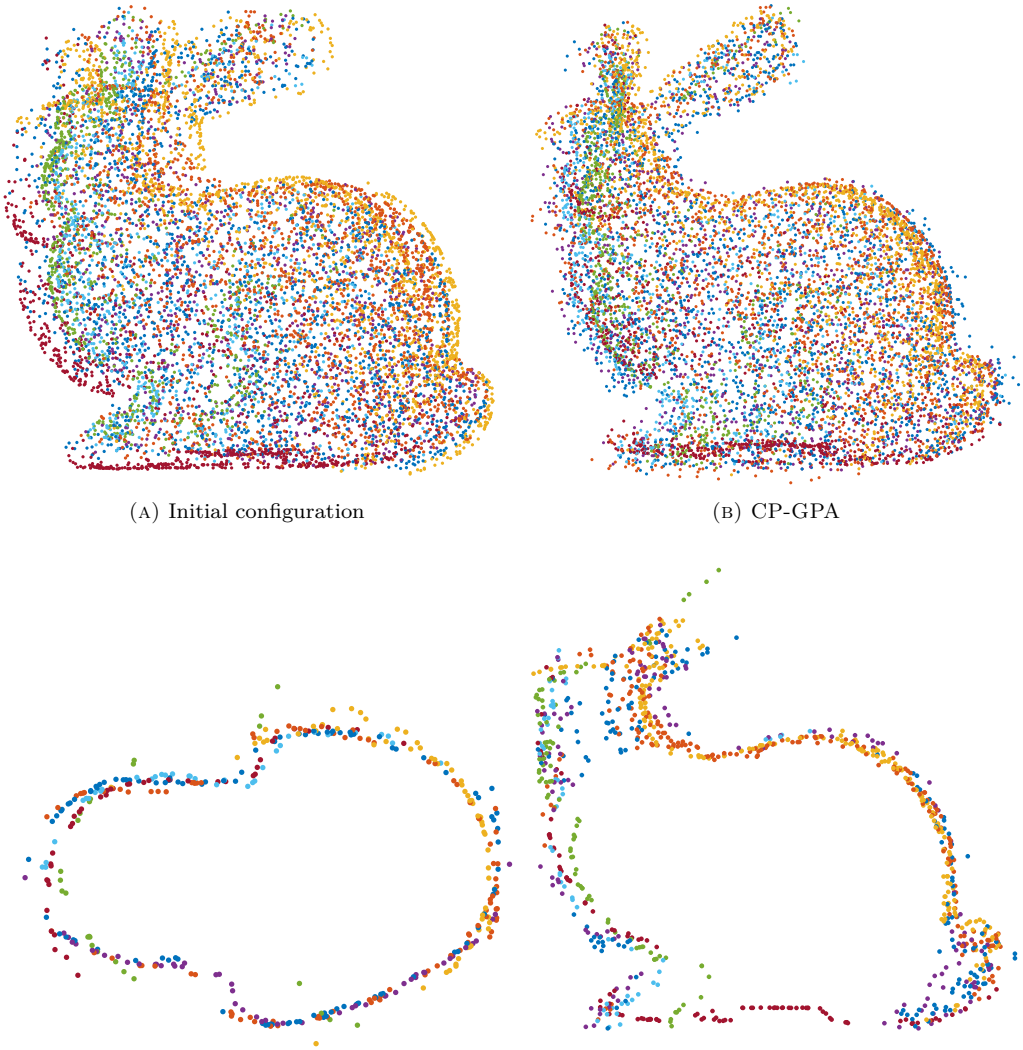
a) Compute mutual closest-points between pairs of views;
b) Apply the multi-view matching algorithm MatchEIG, presented in Chapter 3;
c) Connect point matching into tracks and discard the inconsistent ones.

In [151] a different heuristic have been described, which is very fast but does not not guarantee closed-loop consistency.

As a final remark, it is interesting to note that CP-GPA does not only extend the GPA, but it is also a generalization of the Iterative Closest Point (ICP) algorithm [16], the most common solution in literature to align two point clouds. ICP can be summarized as follows:

1. Compute mutual closest-points between the two views and pretend these are corresponding points;
2. Align the two views using EOPA (Section 2.3.2);
3. Iterate from step 1. until convergence.

This is CP-GPA restricted to two views and with a simple mutual closest-point heuristic in step 1.
Please note that in this case it is not needed to resort to the WEOPA, unless one want to assign different weights to each point. In fact, since only two views are involved, correspondences are bijective and one do not have to manage missing rows in the two matrices of points.

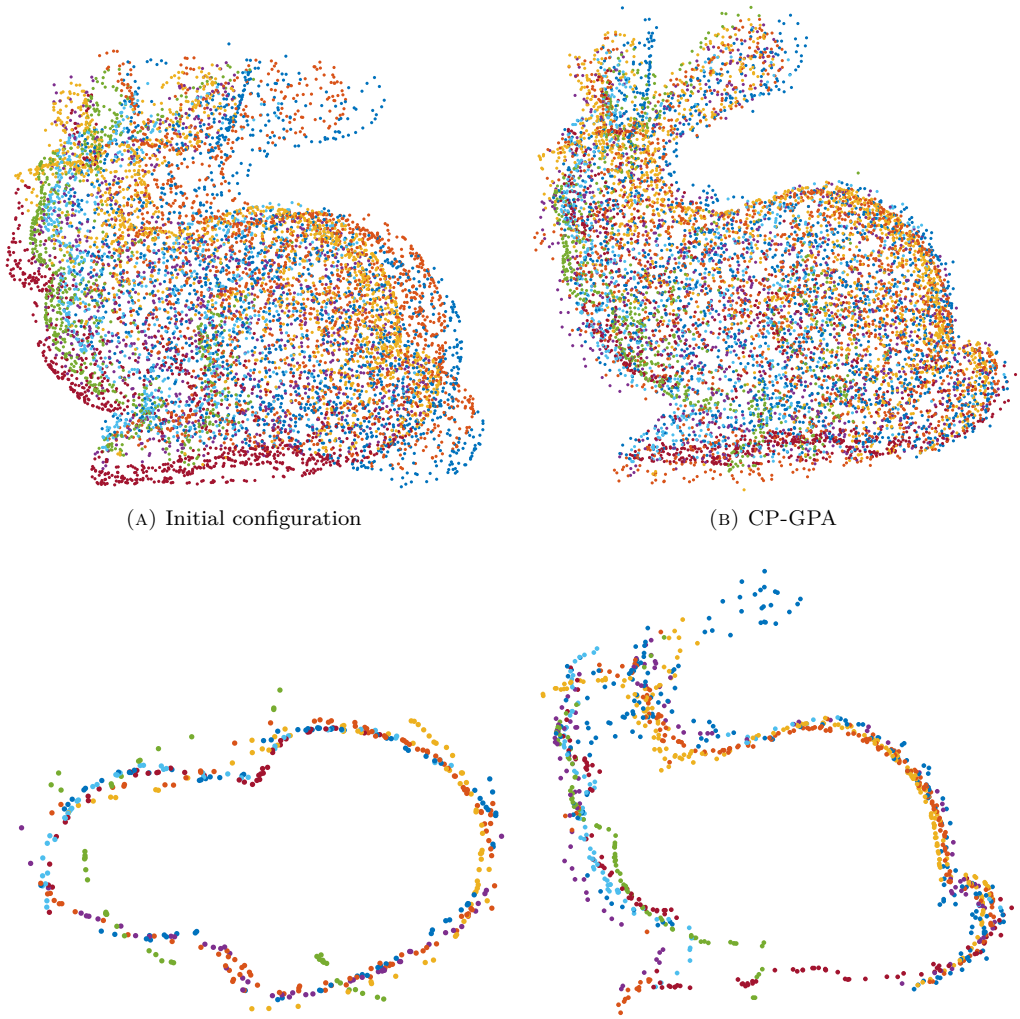


(A) Initial configuration

(B) CP-GPA

(C) Cross sections

FIGURE 4.4: Results of the point cloud registration via CP-GPA. Initial configuration is obtained by perturbing the ground-truth motions by a rotation with random axis and angle uniformly distributed over $[0, 5°]$.



(A) Initial configuration

(B) CP-GPA

(C) Cross sections

FIGURE 4.5: Results of the point cloud registration via CP-GPA. Initial configuration is obtained by perturbing the ground-truth motions by a rotation with random axis and angle uniformly distributed over $[0, 20°]$.

4.4 Point Cloud Registration via CP-GPA: Preliminary Results

In this section we present some preliminary results of the application of CP-GPA to the registration of multiple point clouds. In particular, we use the *Bunny* dataset from Stanford repository[1] which contain 10 point sets. Point clouds were initially randomly downsampled, in order to have approximately 1000 points per views. We perturbed the available ground-truth motions by a rotation with random axis and angle uniformly distributed over $[0, \alpha]$, with $\alpha = \{5°, ..., 20°\}$.
In all cases, CP-GPA managed to register the views. For results visualization purposes, we report in Figure 4.4 and 4.5 the alignments obtained with a noise of $\alpha = 5°$ and $\alpha = 20°$, respectively.

4.5 Conclusion

In this chapter, we extended the GPA formulation to the case in which correspondences among matrix elements are unknown. In particular, we embedded the multi-view matching algorithm inside the GPA iterative scheme, developing a procedure that we called *Closest Point Generalized Procrustes Analysis* (CP-GPA). Moreover, we demonstrated that our method generalizes ICP to the case of multiple point clouds registration. Some preliminary experiments on a simple dataset were proposed, showing promising results. However, a more accurate evaluation of the developed method will be performed in the future, in order to assess the robustness of CP-GPA in many different scenarios.

[1] http://graphics.stanford.edu/data/3Dscanrep/

Chapter 5

Affine Procrustes Analysis

In this chapter we propose a variation of the classical *Extended Orthogonal Procrustes Analysis*, that allows to compute the transformation between two matrices composed by both points and vectors. The method is called *Affine Extended Orthogonal Procrustes Analysis* (Affine-EOPA), because it actually extends the space where EOPA can be applied from Euclidean (points) to affine (points and vectors).
Motivated by a Virtual Trial Assembly application, we derive from Affine-EOPA a new model which is capable of catering for undetermined motion components, i.e., can retrieve the best transformation among corresponding matrix elements under the condition that the position of each plane is undetermined along its normal. The case study that will be described in this chapter is the VTA of the elements of a complex steel structure, the so called *dogbones* of *Vessel* in New York. Taking into account the geometrical characteristics of the structural elements under study, the developed procedure allows to easily verify the parallelism condition of the flange planes and the satisfaction of the tolerances on the bolt hole positions. In this way, possible manufacturing defects are effectively and automatically identified, together with the relative corrections that have to be made.

5.1 Introduction

As extensively discussed in the previous chapters, Procrustes Analysis can be applied in several fields, ranging from geoinformatics to photogrammetry and computer vision, to solve tasks such as the coordinates transformation between different reference systems, the exterior orientation of an image or the registration of multiple point clouds. In this chapter we will introduce a variation of EOPA that finds application in civil engineering, and in particular in the Virtual Trial Assembly (VTA) of a steel structure [24].
In the field of steel constructions it is often necessary to verify in the factory the geometric congruence of the manufactured elements with respect to the nominal values and whether all the connecting points of the various elements guarantee the final assembly of the building, in compliance with the required tolerances.
As a matter of fact, considering the performances in the carpentry field and in mechanical works, there always remains a residual between the size and shape of a workpiece and its technical specifications. Therefore, once the composing parts are manufactured, it is necessary to proceed with their geometric control. This is carried out by using different instruments according to the desired accuracy: metrology-grade laser trackers in the case of very high precision measurements, topographic total stations otherwise.
The compliance of the single elements, however, does not guarantee the assemblability of the whole structure: small errors below the tolerance can accumulate in the assembling and result in an inadmissible error at some stage. It is therefore necessary to proceed with a trial assembly, which can be physical or virtual.
It often happens that it is not possible to perform a complete Physical Trial Assembly (PTA) of the various elements in the manufacturing site, due to the large dimensions of some building structures, and also because of time and cost reasons. It becomes therefore crucial to resort to a Virtual Trial Assembly, that analyzes the discrepancies between the

workpieces and their nominal parameters and also simulate the assembly of the structural parts on a computer, thereby reducing time and costs. If this process detects some problems, within the VTA it is possible to define shape and dimensions of the corrective elements, by means of which the assembly of the structure can be achieved.
Once the VTA has been successfully carried out, the actual position of the elements during the construction phases can be predicted by adding the nominal deformation values, computed through a FEM (Finite Element Method) model, to the coordinates computed by the VTA. This is a necessary step to compare the predicted positions with the ones that can be subsequently measured on site, since the VTA alone does not consider the deformations to which the structure is subjected.

5.1.1 Contribution

Procrustes models, and in particular the use of the *Generalized Procrustes Analysis*, have already been proposed in the literature [24] for the global matching of all the manufactured structural elements of the New Safe Confinement of the Chernobyl nuclear reactor. In this chapter, instead, we present an affine variation of the *Extended Orthogonal Procrustes Analysis* that allows to compute the trasformation between two matrices composed by both points and vectors. The method, that we call Affine-EOPA, is presented in Section 5.2, and applied in Section 5.3 to perform the VTA of *Vessel*, a complex steel structure in New York. After describing the characteristics of *Vessel* (Section 5.4), we discuss the differences between EOPA and Affine-EOPA (Section 5.4.1), showing with the experimental validation presented in Section 5.4.3 that, for the case under study, the novel Affine-EOPA is more suitable to perform the VTA than the classical EOPA.

5.2 Affine Extended Orthogonal Procrustes Analysis (Affine-EOPA)

Procrustes Analysis has been described in Chapter 2, illustrating the different models that have been proposed in the literature to perform transformations among corresponding points belonging to a generic k-dimensional Euclidean space, in order to satisfy their maximum agreement. Let us consider a matrix $\mathbf{A}$ (origin) and a matrix $\mathbf{B}$ (destination), containing the coordinates of p points in $\mathbb{R}^k$. Classical EOPA model (Section 2.3.2) allows to directly estimate the unknown rotation matrix $\mathbf{R}$, the translation vector $\mathbf{t}$ and a global scale factor c for which the residual:

$$\|\mathbf{B} - c\mathbf{A}\mathbf{R} - \mathbf{j}\mathbf{t}'\|_F^2 \tag{5.1}$$

is minimum, under the orthogonality condition: $\mathbf{R}'\mathbf{R} = \mathbf{R}\mathbf{R}' = \mathbf{I}$.
The space where the EOPA operates is a Euclidean one, whose elements are *points*. We can straightforward extend this to an *affine* space, where points and *vectors* are represented, each one by $\mathbb{R}^k$.
If matrix $\mathbf{A}$ is partitioned into points $\mathbf{A}_{pt}$ and vectors $\mathbf{A}_n$, and $\mathbf{B}$ accordingly into $\mathbf{B}_{pt}$ and $\mathbf{B}_n$, it is easy to see that the rotation is computed as usual (see Section 2.3.2) by

$$\mathbf{R} = \mathbf{V}\,\mathrm{diag}\,(1, 1, \det(\mathbf{V}\mathbf{W}'))\,\mathbf{W}' \tag{5.2}$$

where $\mathbf{S} = \mathbf{V}\mathbf{D}_s\mathbf{W}'$ is the Singular Value Decomposition of

$$\mathbf{S} = [\mathbf{A}_{pt}\mathbf{J}, \mathbf{A}_n]'[\mathbf{B}_{pt}\mathbf{J}, \mathbf{B}_n] \tag{5.3}$$

and $\mathbf{J} = \left(\mathbf{I} - \dfrac{\mathbf{j}\mathbf{j}'}{p}\right)$ is the usual centering matrix, whose role is to translate the matrix values to which it is applied to the corresponding barycenter. Please note that points are "centered", whereas normals are not. The translation vector instead depends only on points

(not on vectors):

$$\mathbf{t} = (\mathbf{B}_{pt} - c\mathbf{A}_{pt}\mathbf{R})'\mathbf{j}/p. \tag{5.4}$$

This model will be henceforth dubbed *Affine Extended Orthogonal Procrustes Analysis* (Affine-EOPA).

5.2.1 Affine-EOPA with Undetermined Motion Components (Affine-EOPA*)

Let us now assume that we can partition the point set into planes (at least two) and that each plane can slide along its normal without this influencing the EOPA result. In other words, the position of each plane is undetermined along its normal. Point coordinates shall be used in the Affine-EOPA in such a way they do not pose any constraint on $\mathbf{t}$ along the normal of the plane they belong to. The Affine-EOPA solution can be then constructed as follows:

1. Rotation is computed only from plane normals;
2. For each plane, the formula for $\mathbf{t}$ (5.4) is projected onto the plane itself, therefore cancelling any component along the normal.

Hence, rotation is computed from the SVD of

$$\mathbf{S} = \mathbf{A}'_n\mathbf{B}_n. \tag{5.5}$$

As for the translation, let us consider plane i and let $\mathbf{n}_i$ be its normal. Moreover, let $\mathbf{N}_i = \mathbf{I} - \mathbf{n}_i\mathbf{n}_i'$ be the projector onto the plane orthogonal to $\mathbf{n}_i$. By applying the projection to Equation (5.4) we get:

$$\mathbf{N}_i\mathbf{t} = \mathbf{N}_i\left(\mathbf{B}^i_{pt} - \mathbf{A}^i_{pt}\mathbf{R}\right)'\mathbf{j}/p. \tag{5.6}$$

This is a system of three equations in the unknown $\mathbf{t}$. Since $\text{rank}(\mathbf{N}_i) = 2$ by construction, only two are independent. With at least two planes we can stack enough independent equations and solve for $\mathbf{t}$ the resulting least squares system.
In the remaining part of this chapter, we will refer to Affine-EOPA with undetermined motion components as Affine-EOPA*.
For a better comprehension of the proposed model, we now give a simple geometric interpretation of the searched solution. In the analyzed problem points belong to planes and we want to minimize the distance between destination points and the projection of the origin points on the destination plane, previously rotated through matrix $\mathbf{R}$. The situation is illustrated in Figure 5.1.
Let $\mathbf{p^A} = (x^A, y^A, z^A)'$ and $\mathbf{p^B} = (x^B, y^B, z^B)'$ be the origin and destination point, respectively. Let also $\mathbf{p^{rt}} = (x^{rt}, y^{rt}, z^{rt})'$ be the transformed coordinates of the origin point, i.e.,

$$\mathbf{p^{rt}} = \mathbf{R}'\mathbf{p^A} + \mathbf{t} \tag{5.7}$$

where $\mathbf{t} = (t_x, t_y, t_z)'$ is the unknown translation vector and $\mathbf{R}$ is the rotation matrix computed with (5.2). Finally, let $\mathbf{p^n} = (x^n, y^n, z^n)'$ be the projection of $\mathbf{p^{rt}}$ onto the destination plane, defined by the following equation

$$n_1x + n_2y + n_3z + s = 0 \tag{5.8}$$

with $\mathbf{n} = (n_1, n_2, n_3)'$ normal vector, $s = -(n_1x^g + n_2y^g + n_3z^g)$ constant term and (x^g, y^g, z^g) coordinates of a generic point belonging to the plane. The distance to be minimized is therefore

$$d_{(\mathbf{p^n}, \mathbf{p^B})} = \|\mathbf{p^n} - \mathbf{p^B}\|_2^2. \tag{5.9}$$

One can easily verify that the coordinates of the projected point $\mathbf{p^n}$ can be expressed in the form

$$\mathbf{p^n} = \mathbf{N}\mathbf{p^{rt}} - s\mathbf{n} \tag{5.10}$$

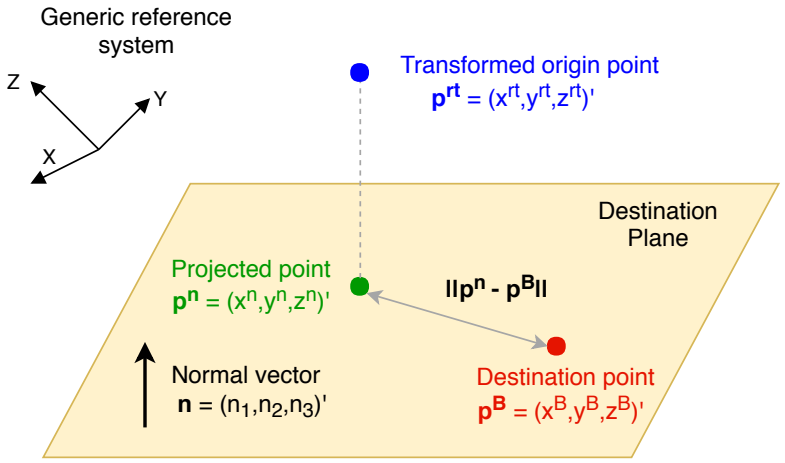


FIGURE 5.1: Minimization of the distance on the destination plane between destination and origin points.

with the projector $\mathbf{N}$

$$\mathbf{N} = \begin{bmatrix} 1-n_1^2 & -n_1n_2 & -n_1n_3 \\ -n_1n_2 & 1-n_2^2 & -n_2n_3 \\ -n_1n_3 & -n_2n_3 & 1-n_3^2 \end{bmatrix} \tag{5.11}$$

depending only on the known nominal normal vector $\mathbf{n}$. Substituting (5.7) in (5.10), one obtains

$$\mathbf{p^n} = \mathbf{Nt} + \mathbf{NR}'\mathbf{p^A} - s\mathbf{n}. \tag{5.12}$$

The objective function to be minimized takes into account all the k_{pt} points belonging to each i-th plane. Assuming a total number of k_n planes, the cost can be written as

$$F(\mathbf{t}) = \sum_{i=1}^{k_n}\sum_{j=1}^{k_{pt}} \|(\mathbf{p^n})_{i,j} - (\mathbf{p^B})_{i,j}\|_2^2 = \sum_{i=1}^{k_n}\sum_{j=1}^{k_{pt}} \|\mathbf{N}_i\mathbf{t} + \mathbf{N}_i\mathbf{R}'(\mathbf{p^A})_{i,j} - s\mathbf{n}_i - (\mathbf{p^B})_{i,j}\|_2^2 \tag{5.13}$$

where subscript j refers to the j-th point of plane i. The components of matrix $\mathbf{N}$ depend only from the destination planes, therefore a matrix $\mathbf{N}_i$ is defined for each plane. Setting to zero the derivatives of $F(\mathbf{t})$ with respect to the unknowns (t_x, t_y, t_z), it is easy to verify that one obtains the normal equations of system (5.6).

5.3 Application of Affine-EOPA* for the Virtual Trial Assembly of a Steel Structure

An emerging trend in manufacturing and design is represented by product proliferation, heterogeneous market, customization increase and shorter product life cycle [25]. Thus, the challenge that industries like manufacturing, automotive and construction are facing nowadays is to reduce production time and cost, in order to remain competitive in the marketplace. Virtual assembly (VA) and virtual prototyping are powerful tools to reach this goal, since visualizing and testing CAD models, before they are physically fabricated or during the early production stage, are effective ways to decrease product development cycle time. In addition, virtual assembly systems could be used to identify and analyze problems that might arise during service and maintainability operations, and they could also provide a platform for the training of assembly workers [138].
In the last years, several computer-based tools to perform virtual assembly, that allow to map the real assembly operation process in a virtual environment, have been proposed in the

literature. Exploiting virtual reality technology, real-time collision detection and assembly path planning can be achieved interactively.
Commonly, virtual assembly systems are based on CAD models of ideal size, however, VA can also be implemented by using object scanning models. These can reflect the real surface characteristics and the actual machining dimensions of the part, leading to a more realistic assembly simulation and a more accurate collision detection [161].
In complex multistage manufacturing systems, in addition to an accurate design of each step of the assembly, it is important to simulate and predict dimensional variation propagation as well. In [25] and [80] the stream-of-variation analysis (SOVA) is applied in the design phase to generate math-based prediction of potential individual assembly errors that contribute to an accumulating set of dimensional variations, which can lead to parts and products that do not respect tolerances.
VA methods have been developed mainly for the manufacturing, automotive and aerospace industry. On the contrary, in the construction industry virtual assembly and virtual prototyping techniques are not widely used. As noticed in [100], this can be caused by the fact that construction industry does not have a production line and each building can be considered a one of a kind. Nevertheless, even the civil engineering world can benefit from the use of virtual assembly techniques. A Construction Virtual Prototyping (CVP) system was developed by Li et al. [100] to support the construction planning in virtual environment. The proposed framework allows, among other things, an effective assemblability analysis, the reduction of construction risks, the optimization of construction schedules and the effective management of design changes. Applied in various scenarios, CVP proved to increase accuracy of process planning and to shorten planning times [79].
Further innovations have been recently introduced not only in the design phase, but also in the construction step for the as-built test of a civil work. Algorithms for construction defect detection [65] and automated visual assessment of changes on a building site [108] were proposed in the last decade. These methods rely on both 3D design models and 3D as-built models, the latter deriving usually from photogrammetric and laser scanning survey techniques.
Another issue in the construction field, pointed out in [163], concerns the steel structures, for which a trial assembly is required in order to verify their assemblability, satisfying the tolerances and the requirements of the project. Since a Physical Trial Assembly is often unfeasible due to time and cost, a Virtual Trial Assembly can be alternatively performed. To the best of our knowledge, in the literature only a few works can be found that apply the virtual assembly to as-built 3D models. Among them, Case et al. [24] proposed an algorithm based on Generalized Procrustes Analysis to simulate the Virtual Trial Assembly of the New Safe Confinement of the Chernobyl nuclear reactor. This work demonstrates the usefulness of VTA and highlights how Procrustes Analysis can be a valuable tool also for this application.

5.4 Precision Controls of the Structural Elements of *Vessel*

Vessel is a steel structure under construction at the center of the Hudson Yards district of New York. To design Vessel, the architect, Thomas Heatherwick, found inspiration by the Indian water tanks, the so called *pushkarani*, with a central small water lake at the bottom, reachable by a segmented staircase. The shape of Vessel is similar to a chalice or a vase (Figure 5.2), with a diameter at the base of almost 17 m, increasing to 40 m at the top of the construction. The total height is almost 46 m: it is possible to reach the top by walking along the stair ramps or by using an elevator that follows the structure profile. Vessel should be ready and opened to the public in 2019. At the moment, the steel structure has been already assembled.

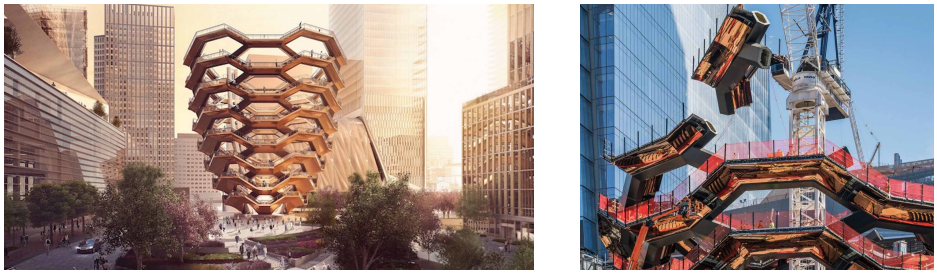


FIGURE 5.2: Rendering of *Vessel* and a construction phase (courtesy of Cimolai S.p.A.).

Among the different structure elements characterizing *Vessel*, the most important for the aims of this work are the so called *dogbones*. These represent the steel units of the building, that are connected to the neighbor ones by a series of four connection flanges per dogbone. The total number of dogbones is 65 (5 for each level, for a total of 13 levels) and the weight of each element is almost 25 t. Rising in elevation, the dimensions of the dogbones become wider, since the diameter of *Vessel* increases with height. Only the dogbones belonging to adjacent levels are connected; elements of the same level instead do not touch each other. In this way, every dogbone constitutes a landing to which four stair ramps are branched off, two upstairs and two downstairs.
The body of the dogbone (Figure 5.3) is constituted by a central gabion, to which four horns are successively welded, while staircases are welded in a second moment. A shim plate is located between the flanges of two adjacent dogbones, whose thickness can be modified in order to correct errors in the elements geometry, if necessary.

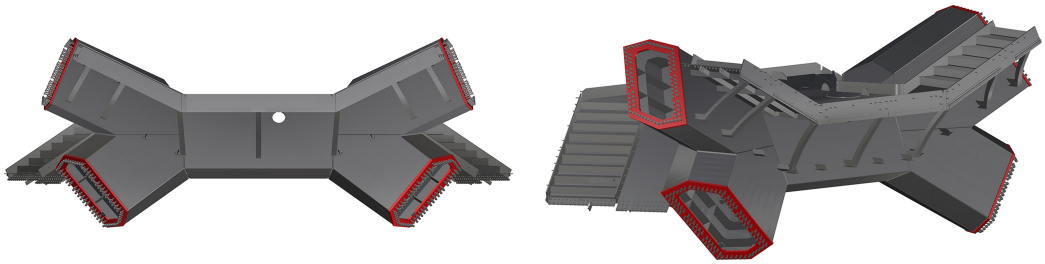


FIGURE 5.3: The characteristic *Vessel* element: the *dogbone* (courtesy of Cimolai S.p.A.).

The building of the dogbone starts with the parallel construction of the central body and of the horns. At this stage, the dogbones have not yet undergone mechanical processing such as milling and drilling. The flanges appear therefore without any hole and their surface is not leveled by machine tools. In this phase the geometric control is directly carried out by the carpenters with proper instruments, so as to verify the project tolerances of the elements. After this test, two horns of the same dogbone are joined one to each other. The topographic control begins when the core steel skeleton of the dogbone is enveloped by a cladding. It is therefore necessary to guarantee the expected tolerances for what concerns the geometry of the produced element not only along the dogbone-dogbone connections, but also along the dogbone-cladding connections. The topographer verifies with a total station that the discrepancies are lower than 10 mm. If this value is satisfied, it is possible to carry on with the welding process of the dogbone central core and the two pairs of horns and with a further topographic control that identifies possible deformations caused by the welding cooling.
Two different kinds of mechanical machining follow the welding step: first, the milling process of each flange and the lateral planes of the structure, so as to satisfy a planarity error of less than 0.1 mm, then the boring of the flange plane. All the operations are managed by a machine tool, using a CAD/CAM technology. The geometric survey of a completed element

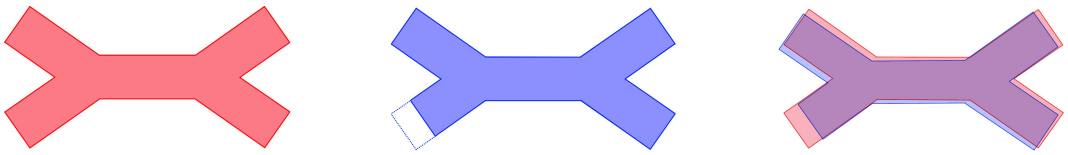
is carried out by a laser tracker, an instrument commonly used in the industrial metrology that can reach a precision of the order of 10^{-6} m.
In particular, the laser tracker measures:

- the plane of the flanges. About 50 points on the flange are measured and the interpolating plane is fitted. It is important to evaluate the planarity of the milled surface and its inclination w.r.t. the theoretical one;
- the edges of the flange. Each edge is geometrically projected on the flange plane and deviations from the project geometry have to be compatible with the misalignment tolerance;
- the holes on the flange. The hole axes are represented by points that have to respect manufacturing tolerances. As for the edges, also these points are projected on the flange plane.

Eventually, the result of the topographic survey is constituted by a series of points and vectors belonging to the interpolated flange plane.
Thanks to the laser tracker survey, the as-built geometry of each dogbone is known with high precision and can be compared to the project values. Moreover, the Virtual Trial Assembly of the whole structure can be performed to understand how possible defects of the dogbones belonging to the lower levels and discrepancies with respect to the theoretical shape can influence the assembly of the upper level elements. In the next section, we demonstrate that Affine-EOPA* is the best Procrustes model to perform these tasks.

5.4.1 EOPA vs Affine-EOPA* in the Virtual Trial Assembly Process

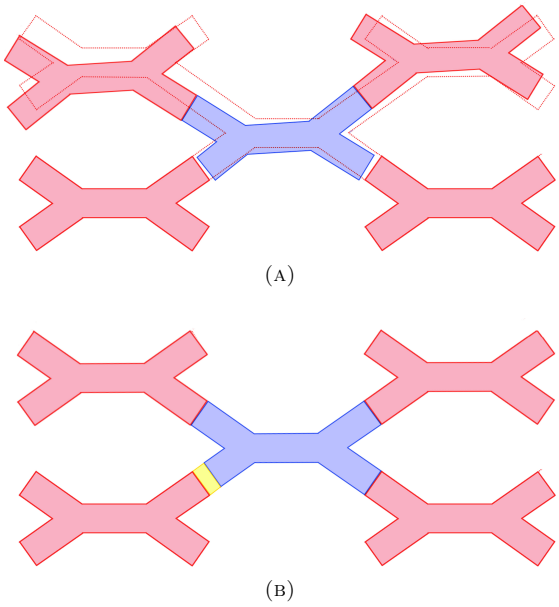
Let us first demonstrate how the classical EOPA can produce misleading results, if applied for the VTA of *Vessel* elements. When comparing the as-built geometry of an element to the project one, the origin and destination matrices contain the coordinates of the measured and nominal holes, respectively. In particular, exploiting the EOPA solution with fixed unitary scale factor ($c = 1$), the survey of the dogbone can be aligned to the theoretical configuration via a roto-translation and the residuals represent the differences between real and project values. Similarly, for the VTA process the origin matrix contains the surveyed points of the dogbone to be assembled, whereas the destination matrix is composed by the coordinates of the dogbones belonging to the lower level, virtually assembled in a previous step. Please note that, since dogbones of the same level are not directly connected and the assembly of an element depends only on the position assumed by the lower level ones, it is not necessary to consider together the configuration of several elements, i.e., it is not required to resort to the Generalized Procrustes Analysis, that simultaneously aligns multiple elements.
Moreover, it is important to consider that, during the machining of the dogbone flanges, the most common flaw that can be generated is that some of them can be milled with an *offset* of some millimeters in the orthogonal direction to the plane. This difference does not represent a problem for the assembly of the whole structure, because it can be easily filled by a shim plate of adequate thickness. Nevertheless, ordinary EOPA does not take into account the possible offset and does not allow to reach the correct alignment between the survey and the project model, since it minimizes the 3D Euclidean distance between corresponding points. Figure 5.4 illustrates the problem and the result obtained by the direct application of the EOPA solution, assuming a unitary scale factor.
It is therefore easy to see that the solution is represented by Affine-EOPA with undetermined motion components (Affine-EOPA*). In fact, the method proposed in Section 5.2.1 allows to first find the rotation that best aligns the plane normals. The alignment between the bolt holes is determined in a subsequent step, through an estimate of the translation between the rotated configuration and the theoretical one, that takes also into account that the position of the bolt holes does not pose any constraint along the normal of the plane they belong to. In this way, when performing the VTA of the structure, machining flaws can be easily identified and corrected by the shim plates, avoiding to compute incorrect rigid transformations



(A) (B) (C)

FIGURE 5.4: Problem connected to the EOPA application. The red dogbone represents the theoretical configuration, the blue element is instead the measured one. Please note that the lower-left flange is machined with an offset that can be corrected with a customized shim plate. The alignment obtained by ordinary EOPA solution is shown in (C).

that, propagating through the various levels, lead to a configuration far from the project values (see Figure 5.5).
Further evidence of how the Affine-EOPA* model is more suitable than the EOPA one will be given in the Section 5.4.3.



(A)

(B)

FIGURE 5.5: VTA performed by the ordinary EOPA solution (with $c = 1$) (A) and using Affine-EOPA* (B). The yellow rectangle in (B) represents the shim plate placed to correct the machining flaw.

5.4.2 Proposed Procedure

Starting from the considerations made in the previous section, we developed a robust procedure based on the Affine-EOPA* model, that allows to

- verify the actual geometry of each dogbone compared to the nominal one;
- perform the Virtual Trial Assembly of the whole structure, highlighting potential flaws that can prevent the structure to be assembled.

As shown in Figure 5.6, the proposed procedure is composed of two main steps. The goal of the first stage is to retrieve the rotation matrix $\mathbf{R}$ that allows to align the planes of the flanges surveyed with the laser tracker to the theoretical configuration. The input data are the measured and nominal points corresponding to the center of the bolt holes. So,

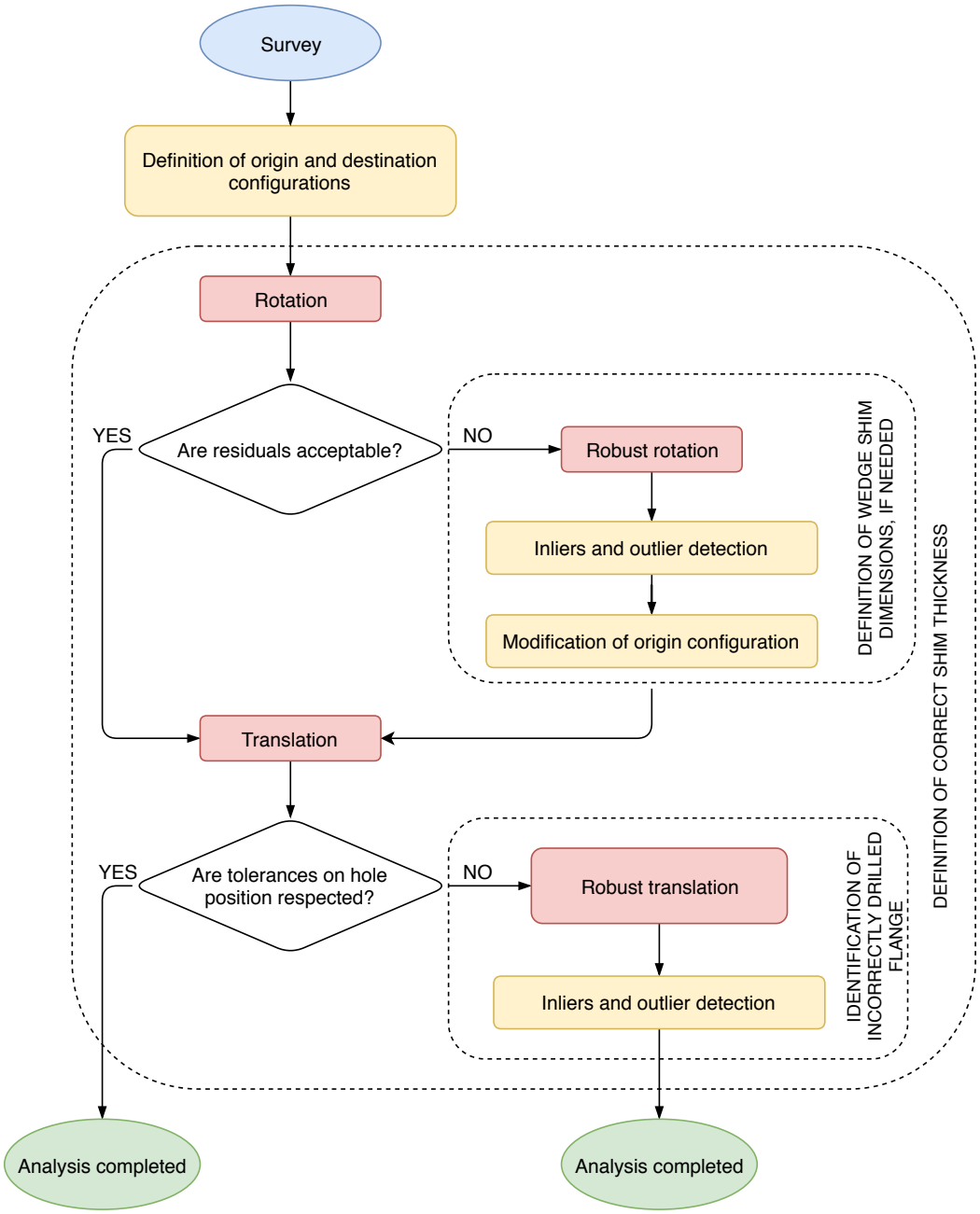


FIGURE 5.6: Flowchart of the proposed method.

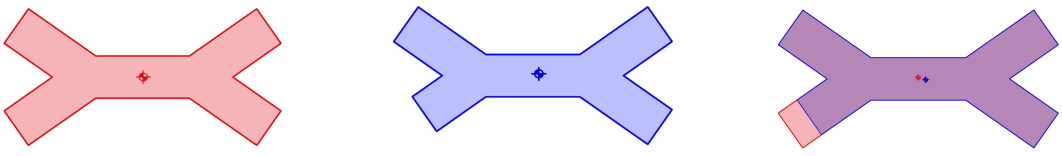
for each flange f_i $(i = 1, ..., 4)$, the measured and nominal planes are estimated from the subset of k_{pt} points belonging to the i-th flange. More in detail, the best fitting plane is estimated exploiting the Principal Component Analysis (PCA) [84] and the normal vector is computed as the eigenvector corresponding to the smallest eigenvalue of the covariance matrix built from the point coordinates. It is important to underline that, for each plane, two normals with the same direction but different orientation can be defined. In this case, the outward-pointing normal vectors are chosen.
Hence, origin matrix $\mathbf{A}_n$ and destination matrix $\mathbf{B}_n$ containing in each row measured and nominal normals, respectively, are used to estimate through Equations (5.2) and (5.5) the rotation $\mathbf{R}$ that maximizes in the least squares sense the parallelism of measured and nominal flange planes.
Residuals deriving from this transformation are correlated to the discrepancy in inclination between the as-built flanges and the nominal ones. The angular difference γ_i between the rotated and theoretical configuration of each flange i (described by the i-th row of matrix $\mathbf{A}_n\mathbf{R}$ and $\mathbf{B}_n$, respectively) can be computed as

$$\gamma_i = \arccos \frac{(\mathbf{A}_n\mathbf{R})_{i,\cdot}(\mathbf{B}_n)'_{i,\cdot}}{\|(\mathbf{A}_n\mathbf{R})_{i,\cdot}\|\|(\mathbf{B}_n)_{i,\cdot}\|} \tag{5.14}$$

and compared with the maximum allowed deviation γ_{max}, defined by the regulations. If the following condition

$$\gamma_i \leq \gamma_{max} \tag{5.15}$$

is not satisfied for all flanges, further analysis is needed. More specifically, a robust procedure is carried out and the rotation is calculated again excluding one flange at a time from the input data, i.e., eliminating a row from matrices $\mathbf{A}_n$ and $\mathbf{B}_n$ and verifying condition (5.15) at each iteration. This method, with a breakdown point of 0.25, allows to detect a possible outlier plane, whose inclination is far from the nominal value, and at the same time to estimate a rotation matrix that is not biased by the outlier flange. In this way, the rotated configuration of inlier flanges is perfectly aligned to the nominal geometry, while the inclination error of the outlier plane can be corrected during the assembly of the structure using, e.g., a wedge shim.
Once the rotation has been estimated, the final translation that aligns measured and nominal holes is computed as described in Section 5.2.1, solving the linear system (5.6). Thanks to this approach, if a flange present an offset, it is successfully aligned to the nominal configuration, as shown in Figure 5.7.



(A) (B) (C)

FIGURE 5.7: Project configuration (A), as-built geometry (B) and alignment obtained via Affine-EOPA* (C).

After the application of the roto-translation parameters estimated by the Affine-EOPA* solution to the measured points, the residuals with respect to the nominal configuration can be studied. In particular, the components of the residual vector perpendicular to the nominal flange plane are used to adjust the thickness of the shim plate placed between adjacent dogbones. The projection on the plane of the residual vector, instead, is compared to the tolerance imposed for the realization of the bolted connection. If the condition is not verified, a robust procedure is carried out, similarly to what is done to estimate a robust rotation matrix. More in detail, the computation is repeated considering three flanges at

a time, in order to robustly compute $\mathbf{t}$ and to localize the flange that does not respect the tolerance. Again, the breakdown point value is equal to 0.25.
To summarize, when applied to a single dogbone, the proposed procedure allows to check the as-built geometry w.r.t. the nominal one, identifying possible outlier flanges, and permits to preliminary evaluate the correct thickness of the shims and the possibility to realize the bolted connections between adjacent dogbones.
As already mentioned, the developed algorithm can be applied with minor changes for further analysis on the assemblability of the whole structure. In fact, it can be used to perform the VTA of all the workpieces, verifying in this way not only the as-built geometry of each single dogbone, but also that they can be connected to each other, respecting the tolerances imposed by the regulations.
Since dogbones of the same level do not touch each other, each element can be analyzed independently. Going into detail, the VTA of a dogbone belonging to level n is achieved assuming in this case that the destination configuration is represented by the upper flanges of the dogbones belonging to level $n-1$. The origin configuration, instead, is constituted by the two lower flanges of the dogbone to be assembled. Thus the roto-translation that best aligns a dogbone with the lower level ones is computed by the same algorithm previously presented, with the difference that in this situation only two flanges can be considered. For this reason, the robust analysis and the subsequent detection of the outlier flange that must be corrected is performed only during the comparison between as-built and theoretical geometry, as it requires four flanges.
Figure 5.8 illustrates the VTA process. Red dogbones represent the elements already assembled, whereas the gray workpiece is the one under study.

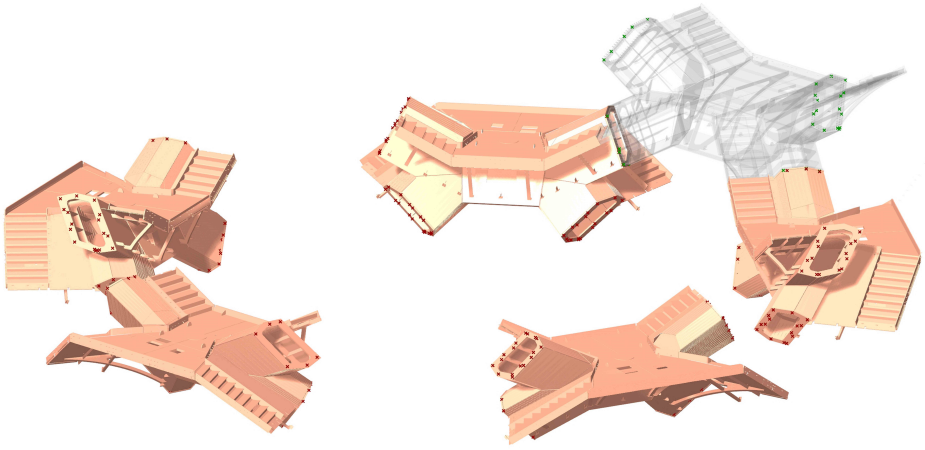


FIGURE 5.8: A step of the Virtual Trial Assembly process. Red dogbones represent the elements already assembled, whereas the gray workpiece is the one under study.

As stated above, a shim plate is placed between adjacent dogbones. Its thickness has a default value of 25 mm, but it can be modified in order to compensate for the flaws generated by the machining. A first value of the thickness correction is computed in the previous step, i.e., when the geometry of each single dogbone is verified. However, a more accurate correction can be estimated during the VTA process, that allows to take into account also the geometry of lower level dogbones that influence the real assembly. The final thickness value is directly computed by the proposed procedure, analyzing the residual distance in the normal direction between a flange of level n and the corresponding one of level $n-1$ after the translation process. The VTA procedure allows therefore to estimate the most appropriate

shim thickness between dogbones and to check the feasibility of the bolted connections in an efficient and automatic way.

Flange	EOPA			Affine-EOPA*		
	Max [mm]	Min [mm]	Average [mm]	Max [mm]	Min [mm]	Average [mm]
1	0.91	0.21	0.46	0.48	0.15	0.28
2	1.94	1.16	1.51	0.38	0.07	0.20
3	1.54	0.84	1.28	0.41	0.08	0.24
4	0.79	0.03	0.52	0.48	0.08	0.25

Table 5.1: Summary values of the residuals projected on the nominal plane obtained through EOPA and Affine-EOPA*.

Flange	EOPA			Affine-EOPA*		
	Max [mm]	Min [mm]	Range [mm]	Max [mm]	Min [mm]	Range [mm]
1	+2.44	+1.12	1.32	+3.82	+2.87	0.95
2	+2.26	+1.60	0.66	+1.70	+1.12	0.58
3	-0.19	-0.42	0.23	+0.22	-0.08	0.30
4	-1.58	-1.89	0.31	-3.23	-3.66	0.43

Table 5.2: Comparison between residuals in the normal direction of the nominal plane obtained through ordinary EOPA and Affine-EOPA*.

5.4.3 Experimental Validation

The proposed method has been successfully applied to verify the as-built geometry of the elements of *Vessel* and to perform the VTA of some levels. In the following, results of the comparison between surveyed and nominal geometry of a dogbone are reported in detail, together with the differences that arise when applying the ordinary EOPA method (with unitary scale factor) instead of the proposed Affine-EOPA* model. The element chosen as case study (dogbone 421 belonging to level 8) presents some flanges that were milled with an offset with respect to the nominal geometry. As previously illustrated, these flaws do not constitute a problem for the erection of the structure, as they can be corrected through shims of adequate thickness. However, they can alter the results obtained by the ordinary EOPA.

Table 5.1 reports the distances projected on the nominal plane between roto-translated surveyed holes and nominal ones. One can easily see that, according to Affine-EOPA*, differences between surveyed and nominal holes are always less than 1 mm, in compliance with the tolerances imposed by the regulations. Ordinary EOPA, instead, furnishes results that are biased by the flaws generated during the milling procedure. These results could lead to the erroneous conclusion that two flanges (number 2 and 3) cannot be bolted.

It is interesting to evaluate also the residuals in the normal direction of the nominal plane (Table 5.2). For each flange, the maximum and minimum residual values and their absolute difference are reported. The latter is a measure of the discrepancy in inclination between the as-built flange and the theoretical one. If it is low (e.g., less than 1 mm), the lack of parallelism is negligible and no customized wedge shim is needed. For the dogbone under study, one can notice that Flange 1 presents a higher deviation; nevertheless, the discrepancy is acceptable and no further robust analysis is required. The mean value of the residual along

		EOPA		Affine-EOPA*	
		Shim thickness [mm]	Residual [mm]	Shim thickness [mm]	Residual [mm]
Dogbone 405	Flange 1	24.8	0.5	25.9	0.6
	Flange 2	24.9	0.4	21.2	0.4
Dogbone 406	Flange 1	24.5	0.6	26.4	0.5
	Flange 2	24.6	0.7	21.7	0.6
Dogbone 407	Flange 1	25.4	1.9	25.9	0.4
	Flange 2	23.2	1.8	22.4	0.8
Dogbone 408	Flange 1	25.1	1.8	23.2	1.0
	Flange 2	26.0	1.2	28.1	1.4
Dogbone 409	Flange 1	23.7	0.9	21.9	1.5
	Flange 2	24.4	1.1	24.3	0.9

TABLE 5.3: Results of the VTA process. Shim thickness and the average residual projected on the flange plane are reported.

the normal direction can be taken as an approximate correction of the shim thickness, that can be refined during the subsequent VTA process. It is important to underline that higher residuals along the normal direction do not mean worse results, since in this case they only represent the shim correction needed and do not affect the structure assemblability. Lower residual values generated by the EOPA method for Flanges 1 and 4 (Table 5.2) derive from the different roto-translation applied, that changes the residual distribution in the 3D space. As already mentioned, the proposed method has been applied with minor changes to simulate the assembly of the structure, allowing to identify any critical issue that may arise during the assembly phases on site. Specifically, we assume as starting point the position of the dogbones belonging to level 4. The surveyed dogbones of this level are individually roto-translated with respect to their theoretical configuration and represent the destination configuration of the elements belonging to level 5. Dogbones of the upper levels are then virtually assembled following the procedure described in Section 5.4.2, i.e., the lower flanges of each dogbone of level n are aligned with the upper flanges of the workpieces of level $n-1$. Results obtained for the VTA of dogbones belonging to level 5 are reported in Table 5.3. The VTA process highlights another advantage of the Affine-EOPA* compared to the ordinary EOPA. In fact, the developed algorithm implicitly takes into account the presence of the shims and the VTA of an element can be performed directly, without any modification of the hole coordinates. Ordinary EOPA, instead, requires the preliminary adjustment of the destination configuration: points on the dogbone flanges of the lower level must be translated along the normal direction of 25 mm, that is the theoretical shim thickness. Without this preprocessing step, ordinary EOPA, that aims at minimizing the 3D distance between corresponding points, tends to "squash" the dogbones of the upper level on those of the lower level. Results of the ordinary EOPA reported in Table 5.3 are obtained after the application of the aforementioned correction to the destination configuration.
The validity of the proposed method was confirmed during the construction of the structure. In fact, the results of the VTA were compared with the surveys carried out on site, showing a tight correspondence between what was predicted by the VTA and what was realized on site. Moreover, the absence of problems for the dogbones installation demonstrated the correctness of the values chosen for the shim thickness, calculated through the procedure described in Section 5.4.2.

5.5 Conclusion

In this chapter we proposed the Affine-EOPA model, a variation of the Extended Orthogonal Procrustes Analysis that allows to compute the trasformation between two matrices composed by both points and vectors and can be further customized to take into account undetermined motion components.
Its application in the structural engineering context is innovative. The algorithm allows to automatically verify the geometry of the manufactured steel elements and to perform the Virtual Trial Assembly of the whole structure, taking into account the geometrical characteristics of the workpieces. The method, in fact, is thought to maximize the parallelism of the planes belonging to adjacent elements and to optimize the possibility to realize the bolted connections between them. Although it was designed for a particular structure, it can be employed to successfully perform the VTA of other steel constructions sharing analogous characteristics.

Part II

Advanced Methods for Remote Sensing Data Processing

Digital terrain models, aerial image mosaics, topographic and geologic maps as well as digital cartography are essential tools for the management and control of a territory. Applications of remotely sensed data are countless: environmental monitoring, planning, management and maintenance of electric power lines, quantification of mining activities, monitoring of anthropic activities like deforestation and urban development, evaluation of land use or surveillance, just to name a few. However, in order to be used for these purposes, remotely sensed data such as LiDAR point clouds and aerial images require some processing steps that are often expensive and time consuming. For this reason, the research activity carried out at Helica s.r.l. was application oriented and focused on developing novel algorithms for the processing of photogrammetric and LiDAR data acquired by helicopters or drones. In the following chapters we will first present a novel method based on deep learning to perform the classification of full-waveform LiDAR data. Then, we will propose an innovative algorithm, based on a mathematical tool known as *synchronization*, to create seamless planar mosaics from aerial images.

Chapter 6

Full-Waveform Airborne LiDAR Data Classification using Convolutional Neural Networks

Point cloud classification is one of the most important and time consuming stages of airborne LiDAR data processing, playing a key role in the generation of cartographic products. This chapter describes an innovative algorithm to perform LiDAR point cloud classification, that is based on Convolutional Neural Networks and takes advantage of full-waveform data registered by modern laser scanners. The employed architecture allows to accurately identify even challenging classes such as power line and transmission tower.

6.1 Introduction

Airborne laser scanning (ALS) relies on the LiDAR (Light Detection and Ranging) principle, namely to measure the time of flight of a short laser pulse travelling to the target and back, that allows to compute the distance between the sensor and the target. Ranges are then converted to discrete 3D points exploiting GNSS (Global Navigation Satellite System) and IMU (Inertial Measurement Unit) data. During its path, the laser ray can be reflected by more than one surface placed at different heights, e.g. part of the laser beam can be reflected from the top of a tree and some part within the tree or the ground surface. The first commercial laser scanners detected only the first and last echo per emitted pulse. Nowadays, most instruments have the ability to record up to six reflections for each emitted pulse and, since 2004, these multi-echo laser scanners have been joined by a new category, the so called *full-waveform* laser scanners, that are finally able to record the entire waveform of the reflected signal. Several studies have shown that these instruments provide a higher spatial point density as well as additional information on the characteristics of the target [155, 154, 112]. In fact, the shape and size of the backscattered waveform is related to the geometry and the reflectance properties of the hit surface.
ALS is currently being employed in a variety of applications, including urban planning, natural hazard management, forestry and facilities monitoring. In almost all the applications, the classification of LiDAR point cloud is required, being a necessary processing step, e.g., to create Digital Terrain Models (DTMs), to perform analyses on data belonging to particular classes (e.g., to evaluate the vegetation density) and to automatically determine the relationships between different classes (e.g., to calculate the distance between power line conductors and vegetation or buildings).

6.1.1 Contribution

The aim of this chapter is to develop a new classification method for full-waveform airborne LiDAR data using Convolutional Neural Networks (CNNs) and exploiting both full-waveform and spatial information. First of all, a brief review of the literature on full-waveform LiDAR data classification is presented in Section 6.2. In Sections 6.3 and 6.4 we introduce deep networks and some basic concepts of deep learning, that can be useful for a better comprehension of the proposed method, described in detail in Section 6.5. As shown in Section 6.6, thanks to the combination of a first CNN that provides a compact representation of the waveforms, and a subsequent Fully Convolutional Network (FCN) that takes into account also the spatial relations between the points, the proposed architecture is able to distinguish (e.g.) among six classes, namely: *ground*, *vegetation*, *building*, *power line*, *transmission tower* and *street path*, with an overall accuracy of 92.6%.

6.2 Related Work

As shown in several studies [47, 114, 121, 127], the LiDAR point cloud classification process can significantly benefit from the data collected by full-waveform laser scanners. In fact, the waveform registered by these instruments offers the possibility to extract additional features related to the reflectivity characteristics of the target. Over the last years, several classification methods have been proposed in the literature using full-waveform data and the features derived from them [51]. Among these, we mention simple thresholds, both set up manually [153] and automatically [2]. The first method distinguishes between vegetation and non-vegetation point with an accuracy of 89.9% for a dense natural forest and 93.7% for a baroque garden area, while the latter exploits the backscatter coefficient derived from the waveform to classify urban areas into vegetation, roads and building roofs. Amplitude, pulse width and number of pulses are the features used in [47] to perform a binary classification and extract vegetation points via a decision tree. Other methods are based on statistical learning classifiers like Support Vector Machines (SVMs, [113]), which belong to non-parametric methods and perform nonlinear classification. This algorithm is well suited for high dimensional problems with limited training set and proved to reach high accuracy (around 95%) when distinguishing between three classes, namely ground, vegetation and building. For urban vegetation detection Höfle et al. [76] use instead geometric and radiometric features that are fed to an artificial neural network classifier consisting of a single hidden layer of neurons and trained by back propagation. Finally, Wang and Glennie [158] apply a "voxelization" method that divides the waveform data into voxels, merging the ones falling in the same voxel into a synthesized waveform. Features are then extracted and fused with the information derived from hyperspectral images, constituting the input of a SVM that is able to discriminate among 9 classes with an overall accuracy of 92.6%.
All these algorithms rely on hand-crafted features, that are subsequently fed to statistical classifiers or simple machine learning algorithms. An alternative approach is the one proposed by Maset et al. [117], that exploits a Kohonen's Self Organizing Maps (SOMs) to perform the unsupervised classification of raw full-waveform data without the need of extracting features from them. The method proved to reach an accuracy of 93.1% over three different classes: grass, trees and road.
In the last years disciplines such as computer vision and robotics have pushed forward and exploited the potential of deep learning [63]. Approaches based on hand-engineered features can nowadays be effectively replaced by methods that learn both features and classifier from the data end-to-end. In particular, Convolutional Neural Networks (CNNs) represent the most powerful and reliable tool for classification and segmentation [144, 129].
While many researchers are focused on the development of new architectures for image and video processing, the application of deep learning to LiDAR data – and, notably, to full-waveform data – is still almost unexplored.

In the case of conventional LiDAR data, the recent works of Hu et al. [77] and Yang et al. [160] can be recalled, in which the potential of CNNs for the classification of LiDAR data is demonstrated. More specifically, in [77] a CNN is used to detect ground points, exploiting a point-to-image framework. For each point in the dataset, context information are computed from the neighbouring points in a window and subsequently transformed into an image that is fed to a CNN. In this way, point classification is treated as the binary classification of an image. Similarly, Yang et al. [160] perform a multi-class segmentation of the point cloud by first transforming the 3D neighbourhood features of a point into a 2D image that is then classified by a CNN. The method reaches an overall accuracy of 82.3% when distinguishing between nine classes, showing however poor performances in the identification of points belonging to small and thin objects such as power line and fences.
Our system is novel both in the type of data it consumes – full-waveform – and in the approach to the problem. Unlike the aforementioned methods, we treat the LiDAR data classification task as a problem of image segmentation solved with a FCN that takes advantages also on the full-waveform data processed by a CNN classifier.

6.3 An Introduction to Deep Learning

In recent years, deep learning has revolutionized many fields, including computer vision, speech and audio processing, robotics, bioinformatics and finance. Deep learning can be defined as a machine learning technique that learns features and tasks directly from data, where data can be, e.g., images, text or sound. Deep learning is therefore often referred to as *end-to-end* learning. Let us assume that we have a set of images and we want to recognize which category of objects each image belongs to: cars, trucks or boat. The starting point is represented by the labelled set of images, or training data. The labels correspond to the desired output of the task. The deep learning algorithm needs these labels as they tell the algorithm about the specific features and objects in the image, so it can learn how to classify input images into the desired categories.
Many of the techniques used in deep learning today have been known for decades and applied to solve important commercial application. For example, they have been used to recognize handwritten postal codes in the US mail service since the 1990s. However, the use of deep learning has surged over the last ten years primarily due to three factors. First, deep learning methods, and in particular *Convolutional Neural Networks* (CNNs), proved to be extremly powerful in several machine learning and computer vision contests, starting when Krizhevsky et al. [93] won the ImageNet object recognition challenge. Deep learning algorithms are now more accurate than people, e.g., at classifying images. Second, GPUs enable us to now train deep networks in less time. Finally, large amount of labelled data required for deep learning has become accessible over the last few years.

6.3.1 Deep vs Shallow Machine Learning

Deep learning and shallow machine learning algorithms both offer ways to train models and classify data. Shallow approaches first require to extract hand-crafted features from the data, that are then used to train a model that describes or predict the object.
On the other hand, deep learning skips the manual step of extracting relevant features. Instead, data are fed directly into the deep learning algorithm, which then predicts the object. So, deep learning is a subtype of machine learning, it deals directly with data and is often more complex. When choosing between shallow machine learning and deep learning, it is first important to consider whether lots of labelled data and a high performance GPU are available. To work successfully with deep learning, in fact, at least a few thousand samples are required to get reliable results and powerful computational resources are needed to train the model. However, deep learning offers many advantages in terms of accuracy and one does not have to understand which features are the best representation of the object, because thay are learned directly.

Hence, the choice between shallow machine learning and deep learning depends on the data and the problem one is trying to solve. The comparison between deep and shallow machine learning approaches is summarized in Table 6.1.

	Deep Learning	**Shallow Machine Learning**
Training dataset	Large	Small
Choose your own features	No	Yes
# trainable parameters	Many	Few
Training time	Long	Short

TABLE 6.1: Comparison between deep learning and shallow machine learning approaches.

6.3.2 Deep Feed-forward Networks

As already mentioned, among all deep learning techniques, image processing has benefited from the development of the so-called *Convolutional Neural Networks* (CNNs), a specialized kind of deep feed-forward networks employed also in this chapter to solve the point cloud classification task. It seems therefore appropriate to introduce deep feed-forward networks, the characteristics of some common kind of layers and some aspects related to their training. The aim of a feed-forward network is to approximate a function f^*. Considering the case of a classifier, e.g., $y = f^*(\mathbf{x})$ maps an input $\mathbf{x}$ to a class y. A feed-forward network defines a mapping $y = f(\mathbf{x}; \boldsymbol{\theta})$, learning the parameter values $\boldsymbol{\theta}$ that give the best function approximation. The term *feed-forward* derives from the fact that information flows unidirectionally through the function being evaluated from $\mathbf{x}$, passing through the intermediate layers that define f, and finally reaching the output y.
Feed-forward models are typically represented by composing together many different functions, for this reason they are called *networks*. The model is associated with a graph describing how the functions are composed together. Let us consider the case of three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain to form $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Function $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, and so on for all the hidden layers, until the final one that is called output layer. The overall length of the chain gives the depth of the model. During the training, $f(\mathbf{x})$ is driven to match $f^*(\mathbf{x})$ by the learning algorithm and by the training data that directly specifies what the output layer must return at each point $\mathbf{x}$.
When talking about these models, we call them *neural* because they are partially inspired by neuroscience. In fact, each unit of a hidden layer of the network can be seen as playing a role analogous to a neuron, since it receives input from many other units and computes its own activation value. However, it is important to underline that modern neural network research is guided by many mathematical and engineering disciplines, and the aim of neural networks is not to reproduce the behaviour of the brain [95, 64].
Usually, in deep learning neurons apply a nonlinear transformation to the input in order to avoid linear models limitations. In particular, linear networks have the drawback that they cannot learn nonlinear functions, so they are not able to understand interactions between input variables. A way to solve this issue is to use a model that learns a different feature space in which a linear model is able to represent the solution. This can be done by a neural network by computing an affine transformation followed by a fixed nonlinear function called activation function. A generic input-output function of a layer can be written as:

$$\mathbf{h} = f(\mathbf{x}) = g(\mathbf{W}^\top \mathbf{x} + \mathbf{c}) \tag{6.1}$$

where $\mathbf{W}$ is a matrix that defines the parameters used to learn the desired function (also called weights of the linear transformation), $\mathbf{c}$ is a vector of biases and g is the activation

function. Every neuron of the layer applies the activation function, so, the generic equation that describes a neuron i is:

$$h_i = g(\mathbf{x}^\top \mathbf{W}_{:,\mathbf{i}} + c_i). \tag{6.2}$$

The most common activation function is the rectified linear unit, or ReLU [83, 120], and it is defined by the function $g(z) = \max\{0, z\}$, represented in Figure 6.1.

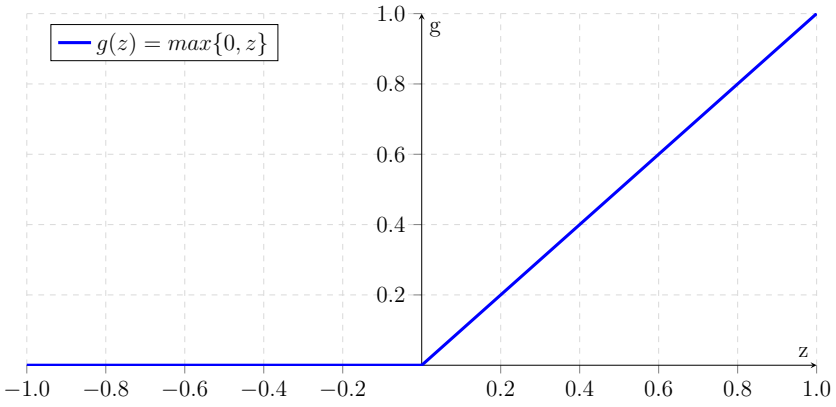


FIGURE 6.1: Rectified linear unit (ReLU) activation function.

As already mentioned, the goal of a feed-forward network is to approximate a function and the tunable parameters of the layers are the weights and biases. When a neural network is deep (with several layers) and has a lot of hidden units, the tunable parameters may even be tens of thousand. Usually, deep learning models are trained using gradient based learning techniques that allow to automatically find the parameters that fit the model. In order to do that, it is necessary to make some design decisions: e.g., choosing the optimizer, the right cost function and the shape of output units. In the following sections we will give some insights on these aspects.

6.3.3 Cost Functions

The training of a neural network implies the iterative computation of a cost value (cost function). The cost is a relation between the output predicted by the network and the expected value, deriving from the training data. The cost is then lowered by making slight adjustments to the weights and biases throughout the training process, until a minimum is reached. The training algorithm is almost always based on using the gradient to decrease the cost function, because the nonlinearity introduced by the activation functions causes loss function to become non convex [75].
The chosen cost function must be appropriate for the task solved by the neural network. For example, a deep neural network designed to perform a classification task could use a cost function called *cross entropy*, defined as follows:

$$J(\theta) = -\sum_{n=1}^{n} p_i \cdot \log(q_i) \tag{6.3}$$

where $\mathbf{p}$ and $\mathbf{q}$ are two probability distribution vectors over n values, that describe the ideal output of the classifier and the predicted one, respectively.
Another example of cost function is the *mean square error*, defined as:

$$J(\theta) = \frac{1}{2}\sum_{n=1}^{n}(y_i - x_i)^2. \tag{6.4}$$

This cost function is the sum of squared differences between the model inputs x_i and the model outputs y_i and it is often associated to *autoencoders*. Without going into detail, an autoencoder is a neural network that is trained to attempt to copy its input to its output [63]. If input values of an autoencoder are real and not binary, then the mean square error is the right choice because it provides a metric to measure the distance between the input and the output.

6.3.4 Output Units

Many tasks require to predict the value of a binary variable y, including, e.g., binary classifiers that have to assign a label to the input data distinguishing between two classes. An example can be a network trained to recognize if an input image represents a vehicle. In this case, the output unit predicts only the probability $P(y = 1|\mathbf{x})$, or, in this example, the probability of recognizing the vehicle in the given input image.
To be a valid probability distribution, this number must lie in the interval $[0, 1]$. Some careful design efforts are required in order to satisfy this constraint. A first solution could be to threshold the output of linear units in order to obtain a valid probability, but with this approach the model would not be able to train effectively with gradient descend. In fact, the thresholding would result in a gradient of the output equal to zero, negatively influencing the learning algorithm that no longer has a guide to update weights. Another approach that is used to always ensure a strong gradient, regardless of the model output, is based on the *sigmoid* output unit, that can be thought as composed by two stages. The first stage uses a linear layer to compute $z = \mathbf{w}^\top \cdot \mathbf{h} + b$, while the second stage uses the sigmoid activation function, represented in Figure 6.2 and defined by

$$g(z) = \frac{1}{1 + \exp\{-z\}}, \tag{6.5}$$

to convert z into a probability, as follows

$$\hat{y} = \frac{1}{1 + \exp\{-(\mathbf{w}^\top \mathbf{h} + b)\}}. \tag{6.6}$$

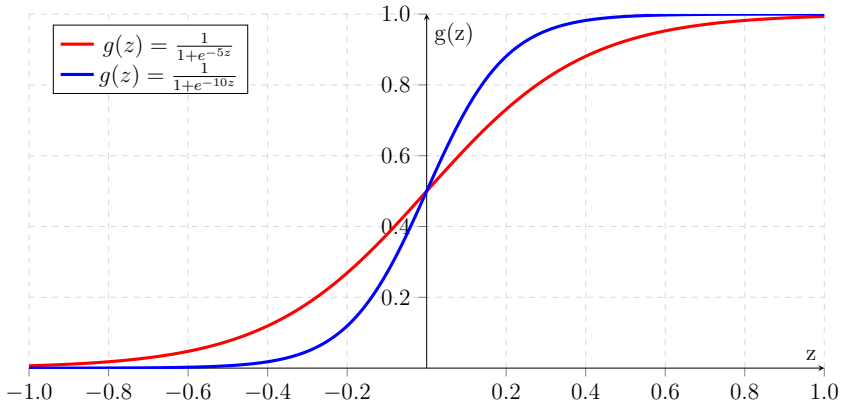


FIGURE 6.2: The sigmoid function produces a valid probability distribution, with values in the range [0,1].

Let us now consider the case in which a network is trained to classify objects given an input image. In this situation, the task of the network consists in representing a probability distribution over a discrete variable with n possible values, i.e., the desired output must be a vector $\hat{\mathbf{y}}$, with $\hat{y}_i = P(y = i|\mathbf{x})$, with each element $\hat{y}_i$ bounded between 0 and 1. It is also required that the sum of the elements of the output vector is equal to 1, in order to describe

a valid probability distribution. Even in this case, the output units are characterized by two stages. First, a linear layer provides unnormalized probabilities:

$$\mathbf{z} = \mathbf{W}^\top \cdot \mathbf{h} + \mathbf{b} \tag{6.7}$$

then one can apply the softmax function, defined by

$$\text{softmax}(\mathbf{z})_i = \frac{\exp\{z_i\}}{\sum_j \exp\{z_j\}}. \tag{6.8}$$

As highlighted in [63], it is interesting to think of the softmax output layer as a way to establish a form of competition between the units that participate in it. The sum of the output values, in fact, is always equal to 1, so an increase in the value of one unit necessary corresponds to a decrease in another unit.

6.3.5 Hidden Units

The choice and the design of hidden units is a very active topic of research, since predicting in advance which kind of hidden layers performs better is nowadays almost impossible. The typical approach consists in trial and error, guessing which type of activation function may work well, training the network and comparing the results on a validation set with the ones obtained with different hidden units. As already described, rectified linear units, or ReLUs, have $g(z) = \max\{0, z\}$ as activation function. These units have the advantage that they are easy to optimize due to their similarity to linear units. As a result, when the unit is active the derivative is significantly larger that zero [83, 120]. On the contrary, when $z_i < 0$ their derivative is zero, thus in this situation rectified linear units cannot learn exploiting gradient based methods. Various alternative activation functions have been proposed to overcome the problem. Equation (6.9) describes a general rectified linear function that uses a non zero slope α_i for $z_i < 0$:

$$h_i = g(\mathbf{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \cdot \min(0, z_i). \tag{6.9}$$

Absolute value rectification [83] sets $\alpha_i = -1$ in order to have $g(z) = |z|$. Other examples are the *leaky ReLU* [110] that fixes α_i to a small value and *parametric ReLU* that treats α_i as a parameter to learn [74].
Anyway, rectified linear units and their generalizations are the most commonly used because they are easy to optimize when they are close to linearity. Many other kinds of hidden units and activation functions are possible but are less frequent. One can mention, e.g., the *Sigmoid* activation function (mostly used in the output layer and shown in the previous section), the *Hyperbolic Tangent* and the *Softplus* activation functions.

6.3.6 Training Algorithms

When a neural network is used to make a prediction, it produces an output $\hat{\mathbf{y}}$ starting from an input $\mathbf{x}$. This process, called *forward propagation*, allows information to flow from the input layer to the output one through the hidden layers. During the training process, forward propagation produces a scalar cost $J(\boldsymbol{\theta})$. The *back propagation*, instead, allows the information from the cost backward through the model until the first layer, in order to simply compute the gradient that is necessary to update the weights of the network.
This section describes how to estimate the gradient $\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y})$ for a generic function f, where $\mathbf{y}$ is a vector of variables that are inputs of the function, and $\mathbf{x}$ is another vector of variables whose derivatives are desired. In deep learning indeed it is required to compute the gradient of the cost function with respect to the weights and biases $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. The method described here is restricted to the case in which the output of the function is a scalar, but it can be easily extended to a function with multiple outputs.

To describe back propagation more in detail let us first briefly introduce computational graph language. Figure 6.3 shows some graphs of known neuron structures. Each node in the graph indicates a variable that can be a tensor, a matrix, a vector or a scalar, while an operation is a function between two or more variables and it returns a value as output.
Back propagation is an algorithm that retrieves gradients using the chain rule of calculus. The chain rule is exploited to calculate derivatives of functions formed by the composition of other functions whose derivative are known [63]. If x is a scalar, f and g are both functions that map a real number to a real number, $y = g(x)$ and $z = f(g(x)) = f(y)$, then for the chain rule it is possible to write:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial f(y)}{\partial y}\frac{\partial g(x)}{\partial x}. \tag{6.10}$$

Generalizing to the case in which x and y are vectors ($\mathbf{y} = g(\mathbf{x})$ and $z = f(\mathbf{y})$), yields

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^{\top} \nabla_{\mathbf{y}} z \tag{6.11}$$

where $(\frac{\partial \mathbf{y}}{\partial \mathbf{x}})^{\top}$ is the Jacobian matrix of g. From (6.11) it is possible to notice that the gradient with respect to a variable $\mathbf{x}$ can be computed multiplying a Jacobian matrix for a gradient. The back propagation algorithm consists in performing this product for every operation in the network graph.

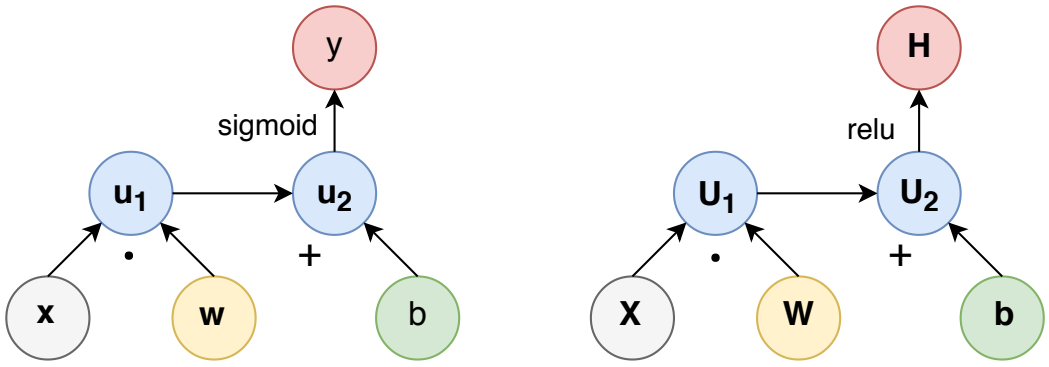


FIGURE 6.3: Examples of computational graphs. (A) Graph of the sigmoid unit $y = \sigma(\mathbf{x}^{\top}\mathbf{w} + b)$. (B) Graph for the expression $\mathbf{H} = \max\{0, \mathbf{XW} + \mathbf{b}\}$, which describes a rectified linear unit [63].

In general, many subexpressions may be repeated a lot of times within the expression of the gradient. Every algorithm that retrieves the gradient must handle this phenomenon choosing between storing the subexpressions or computing them several times. For a very wide graph, there can be millions of this wasted computations that can make the back propagation algorithm computationally unfeasible. In other cases, computing the same factor multiple times can be a good way to reduce memory consumption at the cost of runtime.

Computing the gradient $\nabla_{\theta} J(\theta)$ exactly is computationally expensive because it requires to evaluate the neural network model for every training sample in the entire dataset. It is possible to calculate the expected gradient by randomly sampling a small number of examples from the training dataset, then the expectation is taken averaging those examples. Ad advantage of this approach is that all the optimization algorithms converge faster updating the gradient frequently using its estimate rather than slowly calculating the exact gradient.

Optimization algorithms can be categorized in three groups, depending on the kind of use of the dataset [63]:

- Optimization algorithms that exploit all the training samples to compute the exact gradient are called *batch* or *deterministic* gradient methods. They process the entire dataset simultaneously in a wide *batch*.
- Optimization algorithms that compute the gradient for a single sample at a time are called *stochastic* or *online methods*.
- Optimization algorithms that exploits more than one but fewer than all training samples are called *minibatch* or *minibatch stochastic* methods. These are the most commonly used optimization methods in deep learning models.

The choice of the minibatch size is usually influenced by several factors:

- In order to retrieve a more accurate estimate of the gradient, one should resort to wide batches.
- Multicore and GPUs architectures are not well exploited using very small batches.
- Typically, all the samples in the batch are processed in parallel. In this case the amount of memory necessary for the task scales with the size of the batch.
- Some types of hardware train the network faster using specific sizes of tensors. It is common, for example, to use power of 2 as batch size when using GPUs.

In order to compute an unbiased estimate of the expected gradient from a minibatch, it is absolutely necessary that those samples are independent. The minibatch, in fact, must be selected randomly. This operation can be easily performed shuffling and then splitting into minibatches the entire dataset.
Let us now describe in detail the most common optimization method used to perform the training using the gradient: the *Stochastic gradient descent* algorithm [63].
Stochastic Gradient Descent (SGD) updates parameters at each training iteration, exploiting the following procedure:

1. Prepare a minibatch of m samples from the training dataset $\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m}\}$ with their corresponding labels $\{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_m}\}$;
2. Compute the gradient estimate: $\hat{\mathbf{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i^m L(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i))$;
3. Update the weights and biases: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\mathbf{g}}$;
4. Repeat from Step 1. until convergence or for a predefined number of iterations.

The most important parameter of the algorithm is the learning rate ϵ. Usually it is necessary to decrease gradually the learning rate over the training epochs. It is a common procedure to decrease the learning rate linearly until the iteration τ with the following rule:

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \tag{6.12}$$

where ϵ_k denotes the learning rate at iteration k, $\alpha = k/\tau$ and after epoch τ it is common to leave ϵ constant. The learning rate value is chosen by trial and error, monitoring the learning curves as function of time. The main question about Expression (6.12) is how to set ϵ_0. If it is too large, the cost function could often increase significantly and the learning curve may oscillate violently. If the learning rate is too low, the training procedure progresses slowly.
There are many other optimization algorithms that automatically adapt the learning rate during the learning procedure. The most famous are *Momentum* [143], *AdaGrad* [46] and *Adam* [90].

6.3.7 Regularization

A typical issue in deep learning, and in general in machine learning, is called overfitting. Overfitting occurs when the test (or generalization) error, i.e., the error made by the network in the prediction of a test set, is higher than the training error, i.e., the error made on the data employed in the training stage. It is due to the fact that the model is too complex, i.e., it has an excessive number of parameters, relative to the number of samples in the training dataset. Every technique that acts to reduce the generalization error of the machine learning model is called regularization method. In the following we describe some of the most commonly used methods for regularization in deep learning, that we also adopt in the architecture proposed in Section 6.5.

First of all, one must underline that the quality of the training dataset is a very important factor in order to obtain a machine learning model that performs well in the test dataset. A good way to obtain a low generalization error is to train the network with more data creating fake samples to expand the training dataset. This task is quite simple for classification. A classifier receives complex and multidimensional input and has to recognize it, specifying to which class the input belongs, using a label. So, a classifier must be robust to transformations of the represented object. Variations of the image can be easily simulated translating, scaling, and rotating training images. In some situations it is even possible to inject noise in the input or in the hidden layers of the neural network in order to make the model more robust.

Bagging, instead, is a method to reduce generalization error exploiting the predictions of multiple network models [22]. The idea is to train multiple networks sharing the same architecture with different training datasets. During the test evaluation every model votes the output for the test sample. The particularity of bagging is that, although all the models have the same network design, they are trained with different datasets, so also the parameters are different. Bagging is an example of a general strategy in deep learning called *model averaging*, employed by the so-called *ensemble methods*.

Ensemble methods work well because the trained models usually make different errors in the test dataset. It is possible to demonstrate, in fact, that if the errors are perfectly uncorrelated, the expected squared error of the ensemble is v/k, where k is the number of models and v is the variance of the error. This means that the expected squared error decreases linearly with the number of models used for the prediction. In order to apply a bagging method, it is necessary to construct k different datasets. Each dataset has the same number of samples as the original one, but each of them is made by sampling with replacement from the original. Using this procedure, with high probability, each dataset is missing some of the samples from the original dataset and contains several duplicates. During the test evaluation, the result can be chosen making models to vote.

Dropout is a very used regularization method in deep learning because it is computationally inexpensive but powerful [141]. Dropout can be seen as a form of averaging multiple models or as a bagging method that can be used for very large neural networks. As previously described, bagging consists on using multiple models trained separately. This procedure seems impractical when each model is a wide and deep neural network because the training and the evaluation steps have high runtime and memory cost. Dropout provides an inexpensive approximation to train an ensemble of an exponential number of neural networks. It consists in removing, at each training iteration, some nodes in the network in a random way, reactivating instead all the neurons during the prediction phase.

Finally, *batch normalization* [82] technique is one of the most important and interesting innovations in recent years regarding neural networks optimization. It is a method of adaptive reparametrization, that was developed to overcome the difficulties in training very deep network models. As already mentioned in previous sections, the gradient tells to the model how to update every weight, with the assumption that the other layers do not change. However, in practice all the layers are updated simultaneously. The updating may cause unexpected results because many functions composed together are changed simultaneously, exploiting updates computed under the assumption that the other functions are kept constant. Of course, this phenomenon makes the model harder to train. Batch normalization reduces the

problem of coordinating updates across many layers, normalizing the activations of the layer to which it is applied through the mean and the standard deviation of each unit. Thanks to batch normalization, much higher learning rates can be used. Moreover, since it acts as a regularizer, it prevents overfitting and in some cases it allows to avoid the use of dropout.

6.4 Convolutional Neural Networks

Convolutional Neural Networks, or CNNs, are deep network models that have completely dominated the machine vision space in recent years. These networks are so influential that they have made deep learning one of the hottest topics in artificial intelligence today [63]. First proposed by Lecun in 1998 [97], they currently represent the best solution for many machine vision tasks.
CNNs are particularly suited for processing data that has a known grid-like topology. These data usually consist of several channels, corresponding to the observation of a different quantity at some point in space or time. Examples of data types usually processed with CNNs are the following:

- 1D data: time-series data such as audio waveforms, with each element corresponding to the amplitude of the waveform at a specific time interval.
- 2D data: image data, that are represented by a 2D grid of pixels. RGB images are composed by three channels, one for each color (reg, green, blue).
- 3D data: volumetric data such as medical images or color video data, where one axis corresponds to time, one to the height and one the width of the video frame.

6.4.1 Convolution and Receptive Fields

The name *Convolutional Neural Network* originates from the mathematical operation that lies at the basis of these networks, i.e., convolution. In its most general form, the convolution is an operation between two functions of real argument, and can be written as:

$$s(b) = \int x(a)w(b-a)da. \tag{6.13}$$

This operation is also typically denoted as $s(b) = (x \star w)(b)$, where the first function is often referred as the input while the second function is called kernel. The output is sometimes called feature map. Machine vision algorithms often involve convolutions over more than one axis at a time. For example, if the model has to process an image, the convolutional kernel must slide along rows and columns. Given a two-dimensional image I as input and a two-dimensional kernel K, convolution can be defined as:

$$S(i,j) = (K \star I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{6.14}$$

An example of convolution applied to a 2D input is represented in Figure 6.4.
The success of CNNs is due to three important concepts connected to the use of the convolution operation, that can improve a machine learning model: *sparse interactions*, *parameter sharing*, and *equivariant representations* [63].
Fully-connected neural networks are composed by layers that compute a matrix multiplication between the parameters of the units and the inputs. In contrast to fully-connected neural networks, where each neuron in the input layer is connected to a neuron in the hidden layer, in a CNN only a small region of input layer neurons can connect to neurons in the hidden layer. These regions are referred to as local receptive fields. The local receptive field is translated across the input data to create a feature map from the input layer to the hidden layer neurons. This characteristic has a positive impact on the memory consumption: in

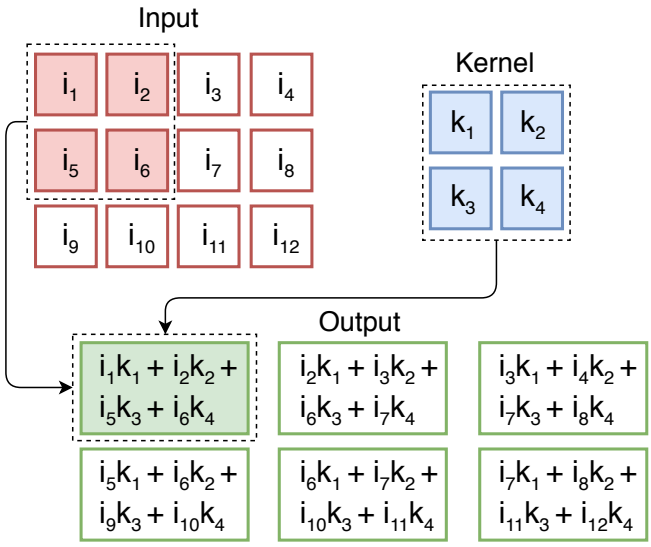


FIGURE 6.4: Example of 2D convolution. The output is computed only for positions where the kernel lies entirely within the image. Note how the upper-left element of the output is computed by applying the kernel to the corresponding upper-left region of the input.

fact, the number of parameters to store is lower, as well as the number of operations required to compute the output. The concept of sparse interactions and local receptive fields is illustrated in Figures 6.5 and 6.6.
The idea of parameter sharing refers instead to the fact that each element of the kernel is applied at every position of the input, in contrast to a fully-connected neural network, where each element of the weight matrix is applied only to an element of the input. This results in learning a small set of parameters for each convolutional layer (the parameters of the kernel), rather than learning a separate set of parameters for every input location.
Sharing parameters allows the filter to look for the same pattern in different regions of the input, which leads to a property called equivariance to translations. For example, when an image is processed through a convolutional layer, the resulting output is a 2D map that describes where certain features appear in the input sample. If the object shown in the input is moved, the output representation will move of the same amount of pixels. This can be useful when we want to detect the same edges, corners or patterns that may appear

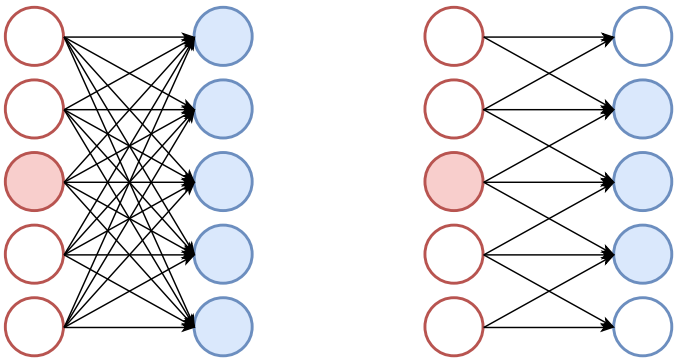


FIGURE 6.5: Left: in fully-connected neural networks, each neuron in the input layer is connected to all the neurons in the hidden layer. Right: Convolutional Neural Networks are characterized by sparse interactions, i.e., a neuron in the input layer affects only a subet of neurons in the following layer.

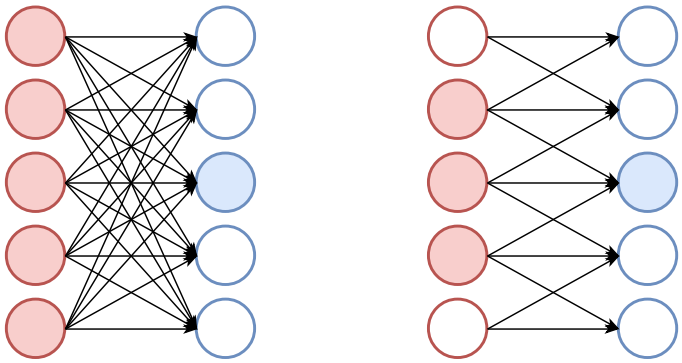


FIGURE 6.6: Left: in fully-connected neural networks, each neuron is affected by all the neurons in the previous layer. Right: in Convolutional Neural Networks only a small region of input layer neurons is connected to a neuron in the hidden layer. This region is known as local receptive field.

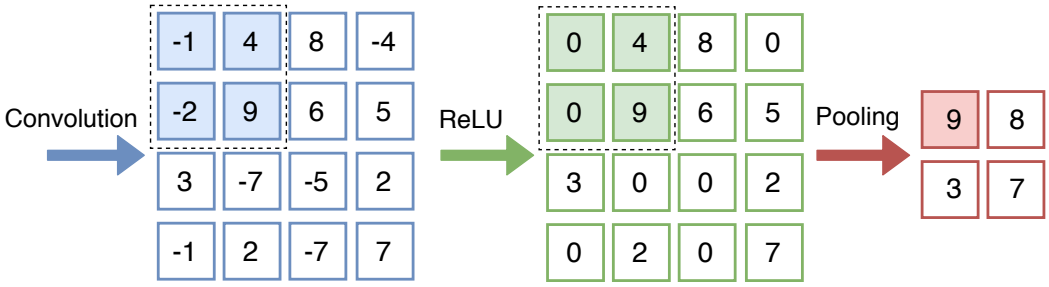


FIGURE 6.7: Example of ReLU and pooling functions applied to the output of the convolution operation. The pooling step keeps the maximum value in a 2×2 neighbourhood.

everywhere in the image.

6.4.2 Pooling

Typically, a convolutional layer is composed of three stages. In the first one the layer performs several convolutions in parallel and produce a linear representation of the input. This linear stage is followed by the usual nonlinear activation function (Section 6.3.5), which represents the detector stage. Finally, a pooling stage is performed, that replaces the output of the convolutional layer at a certain position with a summary statistic of the neighboring outputs (see Figure 6.7 for a numerical example of these steps).
In this way, pooling stage helps to make the output representation of the convolutional layer invariant to small translations of the input. A typical example of pooling operation is the *max pooling* [165] that keeps the maximum output in a rectangular neighbourhood. Pooling reduces the dimensionality of the feature map by condensing the output of small regions of neurons into a single output. This helps simplifying the following layers and reduces the number of parameters that the model needs to learn.
Pooling is an essential operation also to handle inputs of varying size. For example, to perform the classification of images of variable dimensions, the final classification layer must have a fixed size. This can be obtained by changing the size of an offset between pooling regions so that the classification layer always receives the same number of summay features regardless of the image dimensions.
To summarize, combining convolution and pooling in tens or hundreds of hidden layer, a CNN can learn to detect different features in an image. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer learns how to

detect edges and the last learns how to detect more complex shapes. Just like in a fully-connected network, the final layer connects every neuron from the last hidden layer to the output neurons, to produce the final output.
There are three ways to use CNN for image analysis. The first method is to train a CNN from scratch. This method is highly accurate, although it is also the most challenging, as one might need hundreds of thousands of labelled images and significant computational resources. The second method relies on transfer learning, which is based on the idea that one can use knowledge of one type of problem to solve a similar problem. For example, one could use a CNN model that has been trained to recognize animals to initialize and train a new model that differentiates between cars and trucks. This method requires less data and fewer computational resources than the first. With the third method, one can use a pre-trained CNN to extract features for training a machine learning model. For example, a hidden layer that has learned how to detect edges in an image is probably relevant in images from many different domains.
The characteristics of CNNs illustrated so far led us to think of their use also to solve the full-waveform LiDAR data classification problem, as described in the following.

6.5 Proposed Framework

As previously mentioned, the novel method proposed in this chapter for point cloud classification tries to take advantage of the useful information provided by waveforms recorded by modern laser scanners and of the potentialities offered by deep learning for solving classification and segmentation tasks. The entire architecture is summarized in Figure 6.8 and described in detail in the following sections.

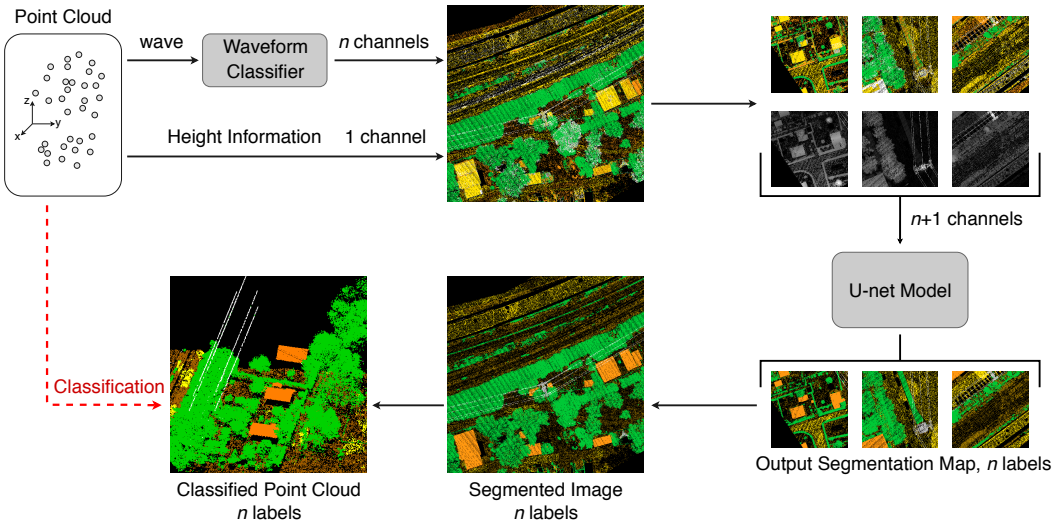


FIGURE 6.8: Workflow of the proposed classification method. First, the waveform classifier (a standard CNN) predicts the point class only exploiting full-waveform data. Predictions are then mapped into an image, together with the height information derived from the 3D coordinates of the points. The resulting multi-channel image is then processed by a FCN (U-net) that refines predictions using spatial information.

6.5.1 Waveform Classifier

In the first step of the algorithm, raw waveform data are given as input to a classifier that outputs a vector of length n (with n total number of classes) containing the probability that the analyzed input belongs to a certain class. The idea is to train a CNN classifier that

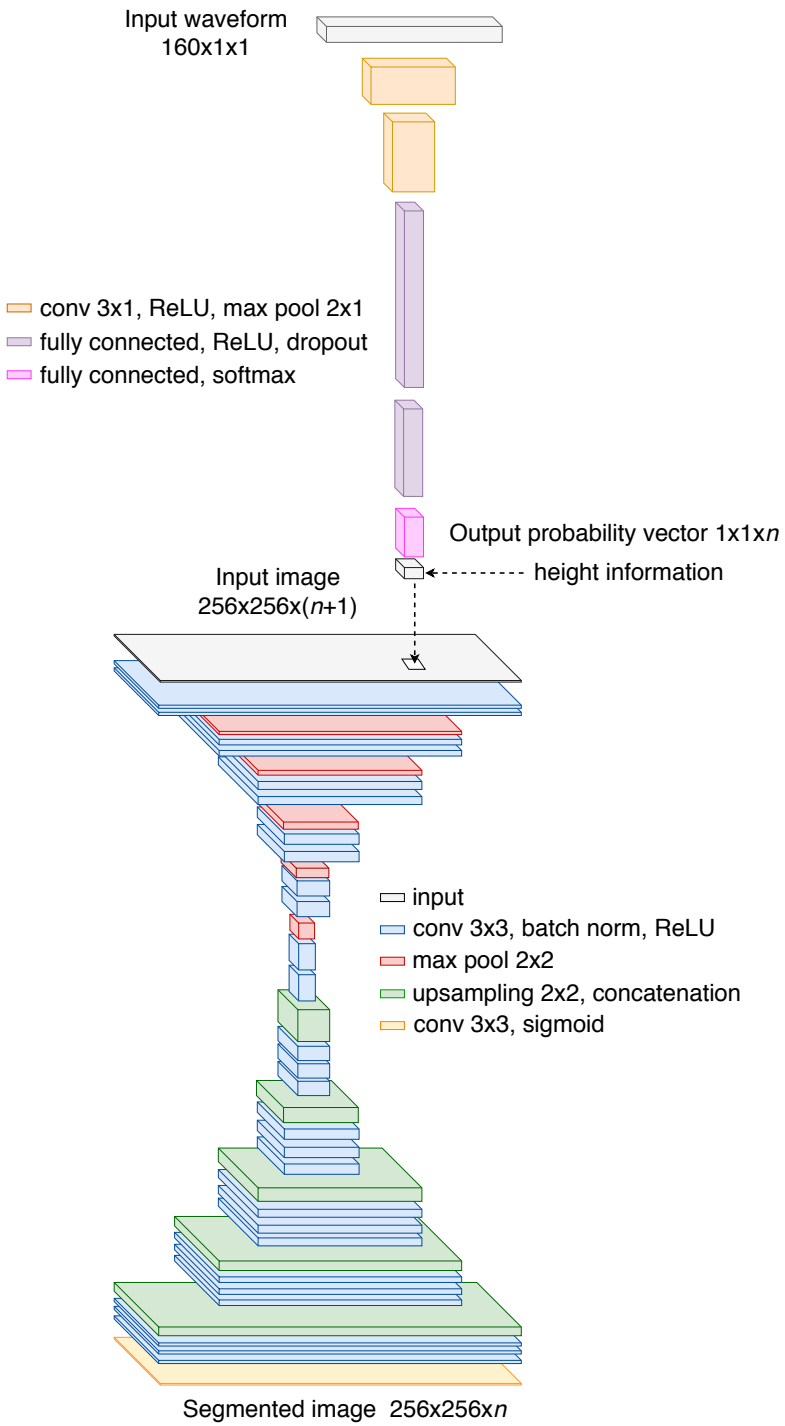


FIGURE 6.9: Architecture of the proposed networks. At the top, the waveform classifier. At the bottom, the U-net model used for the image segmentation.

provides a compact way to describe each waveform. In fact, as discussed in Section 6.4, CNNs are particularly suited for processing data that have a known grid-like topology, so they can also be applied to time series data such as audio tracks or, as in this case, the recorded waveforms.
The architecture of the CNN used in the proposed method is shown in the upper part of Figure 6.9. More in detail, the waveform, consisting of a vector of 160 elements, is fed into two consecutive 1D convolutional layers with kernel size 3, that have 32 and 64 filters, respectively. Both layers are followed by a rectified linear unit (ReLU) activation function and a max-pooling layer with kernel size 2. As shown in Section 6.3.5, the activation function is necessary to introduce nonlinearity, whereas the max-pooling layer helps to make the representation approximately invariant to small translations of the input and decreases the computational effort reducing the representation size (Section 6.4).
After the convolutional layers, the network exploits two fully connected layers to perform the classification. The number of neurons is 2048 and 1024, respectively. Both fully connected layers are followed by a ReLU activation function and a dropout layer (Section 6.3.7) with a dropout rate of 0.5. The output layer is a n neurons layer followed by *softmax* activation function (Section 6.3.4) which produces a probability distribution over n classes.
As an alternative to this model, we tested also various autoencoder configurations to generate a description of the waveform. However, as it will be shown in Section 6.6.3, the classifier proved to perform better.

6.5.2 Point Cloud to Image

The accuracy that can be achieved by the first CNN, that exploits only raw waveform data, is not sufficient, thus additional spatial information must be considered for a precise classification.
The idea is then to map the point cloud into a two-dimensional image, exploiting (x, y) coordinates of the points that correspond to the first return (echo) registered in each waveform. In this way, spatial positions and geometrical relationships between neighbouring data are taken into account. The resulting image has multiple channels: every pixel stores the n-dimensional probability distribution vector, provided by the classifier employed in the first stage of the procedure, and the height of the data falling in the pixel. The point cloud classification problem can therefore be cast to the segmentation of an image, that assigns a class label *per-pixel*. This task can be solved by a Fully Convolutional Network (FCN), as described in detail in Section 6.5.3.
During the mapping procedure a loss of information inevitably occurs because of *collisions*, i.e., more than one point is mapped to the same pixel. This phenomenon has a negative impact on the accuracy of the algorithm. It is possible to limit collisions by reducing the pixel size, which entails enlarging the image, and at the same time increasing the computing time. Please note however that collisions are problematic only when involving points of different classes, otherwise a single class label is adequate for all the points.
With the parameters chosen in our experiments, approximately 5% of the points of the dataset collides, but fortunately less than 0.5% involve points with different labels. In that cases, the point with the highest altitude value is chosen, in order to improve classification of small and thin objects such as towers and power lines, which are the most critical classes.

6.5.3 Image Segmentation via U-net

CNNs were firstly designed to solve image classification tasks, where the desired output is a single class label assigned to the input image. However, in recent years several architectures have been proposed to perform image segmentation [31, 105], allowing to assign a class label to each pixel. In particular, we started from the so called U-net model [129] and implemented a FCN to segment the multi-channel image created as described in the previous section. A FCN is composed only of convolutional layers without any fully-connected ones. This allows

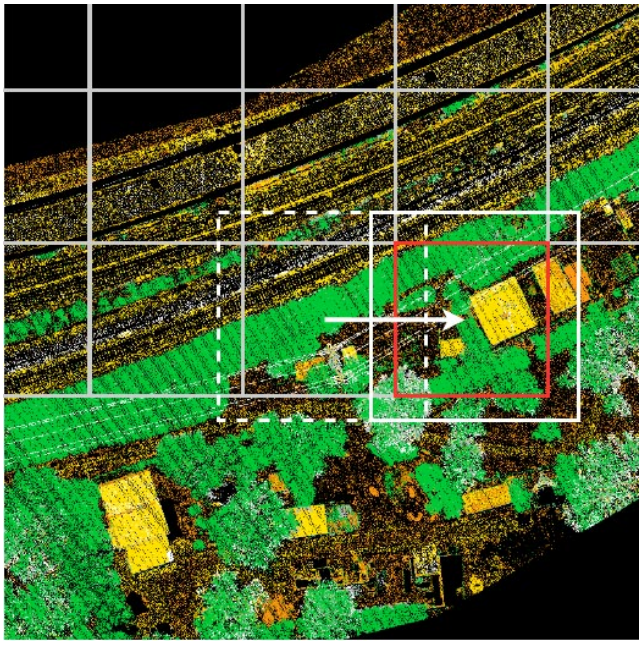


FIGURE 6.10: In order to be processed by the U-net model, an image of arbitrary size is splitted in overlapping tiles of size 256×256 with an overlap of 28 pixels.

to operate on an input of any size, producing an output of corresponding spatial dimensions [105].
The network we employed, illustrated in Figure 6.9, consists of a contracting path (upper part) and an almost symmetrical expansive path (bottom part). In the contracting path, the network looks like a typical CNN able to recognize both low and high level features. Each layer is composed by two 3×3 convolutions, each followed by batch normalization and ReLU activation function. A 2×2 max-pooling operation is then applied to reduce the representation size by a factor of two, starting from an input of dimensions 256×256 and reaching a size of 8×8 at the final layer of the contracting path. The number of feature channels is doubled at each layer with respect to the previous one. The first layer outputs 64 feature maps, whereas the last one 2048.
Every layer in the expansive path consists instead of an upsampling of the feature maps that increases the resolution of the output of the previous layer, a concatenation with the corresponding feature maps from the contracting path and three 3×3 convolutions, each followed by batch normalization and a ReLU activation function. At the final layer a 1×1 convolution is used to map each 64 components feature vector to the desired number of classes. While the contracting path captures context information, the expansive path enables precise localization [129], thus allowing a *per-pixel* labelling.
The U-net consumes the multi-channel image created as described in Section 6.5.2. The first layer of the U-net model is designed so as to take in input images of fixed size (256×256 in our case) but a point cloud can be mapped in a much larger image. An image of arbitrary size can be processed by an overlap-tile strategy. Since convolutions in our U-net are padded, the *valid* portion of the 256×256 output layer is reduced by 14 pixels at each side. Therefore input tiles must overlap (by 28 pixels) in order to provide a valid output for each pixel, as shown in Figure 6.10.

6.6 Experiments and Results

The networks have been implemented in Keras [30] and run on a Tesla K40c GPU. Validation has been performed on a dataset that we manually labelled and will made available on the

Table 6.2: Points distribution over the six classes, divided into training and test sets.

		Training		Test	
Label	**Class**	**# Points**	**%**	**# Points**	**%**
1	*ground*	1787352	20.4	193070	18.1
2	*vegetation*	4719634	53.9	765327	71.7
3	*building*	1514486	17.3	49138	4.6
4	*power line*	71978	0.8	8151	0.8
5	*tower*	32008	0.4	1829	0.2
6	*street path*	633606	7.2	49580	4.6

web to allow for future comparisons.

6.6.1 Dataset

Our networks have been trained and validated using a dataset acquired by Helica s.r.l. with a Riegl LMS-Q780 full-waveform airborne laser scanner. The surveyed area contains both natural surfaces such as ground and vegetation, as well as artificial objects such as buildings, power lines and transmission towers.

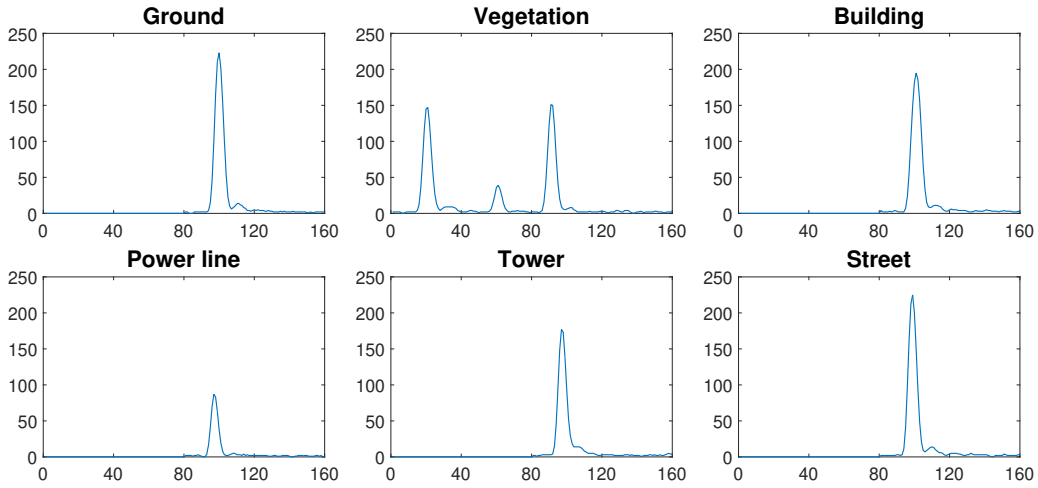


Figure 6.11: Waveform samples.

Three different information are associated to every measured point contained in the dataset, namely the waveform registered by the LiDAR full-waveform sensor, described by a vector of 160 values, the 3D coordinates of the point and the label that shows the class to which the point belongs. These labels have been assigned manually among six classes that were identified: *ground*, *vegetation*, *building*, *power line*, *transmission tower* and *street path*.
The point cloud is composed by more than 9.8 million points, unevenly distributed over the classes. The dataset is indeed very imbalanced due to the different shape of the scanned objects and the occupied area: e.g., the number of points belonging to vegetation and ground is much higher than the number of points belonging to power line and transmission tower classes. Table 6.2 shows in detail the points distribution over the classes.
To handle the entire point cloud, the dataset is divided into tiles, each containing a different number of points. In the experiments, one tile is used as test dataset (corresponding to

approximately 10% of the total number of points), while the remaining tiles are exploited to train the models.

6.6.2 Training

To overcome the imbalanced distribution of the points over the six classes, when training the waveform classifier (Section 6.5.1) we sample with replacement a fixed number of waveforms for each class. More specifically, we employ 200 thousand waveforms per class, for a total of 1.2 million samples. We tested also techniques to balance the class distribution for the training stage [27, 73] but no significant improvement on the final results can be noticed.
The training is performed using categorical cross-entropy as loss function and Adam optimizer [89] with 0.001 learning rate (Section 6.3.6), while dropout is applied with rate 0.5 on the two fully-connected layers. The weights are initialized as described in [58]. The CNN has 12 million trainable parameters and, fixing the batch size to 256, a training epoch takes approximately 30 seconds and it converges after a few minutes.
Regarding the U-net (Section 6.5.2), the training is done using 15 thousand 256×256 windows with 7 channels for each pixel (see Figure 6.13). Six channels correspond to the probability vector over the six classes provided as output by the classifier, while one channel contains the height information. Please note that the training images are randomly cut out and extracted from the much larger image in which the training point cloud is mapped. To take into account the unbalancing of the point distribution over the classes, it is ensured that 1700, 3400 and 3400 training images contain pixels belonging to *building*, *power line* and *transmission tower*, respectively, which are the under-represented classes.
For the training of this FCN, categorical cross-entropy is used as loss function and Adam optimizer [89] is applied with learning rate 0.0002, while the weights are initialized as described in [58]. Choosing a batch size of 8 images, the training of the U-net model (with 138 million trainable parameters) takes approximately 80 minutes per training epoch, reaching convergence after 30 epochs.

	ground	*vegetation*	*building*	*power line*	*tower*	*street*
ground	0.07	0.13	0.34	0.07	0.02	0.37
vegetation	0.01	0.79	0.02	0.06	0.09	0.03
building	0.05	0.04	0.13	0.03	0.04	0.72
power line	0.00	0.03	0.00	0.91	0.03	0.03
tower	0.03	0.09	0.02	0.29	0.42	0.15
street	0.04	0.03	0.29	0.07	0.01	0.56

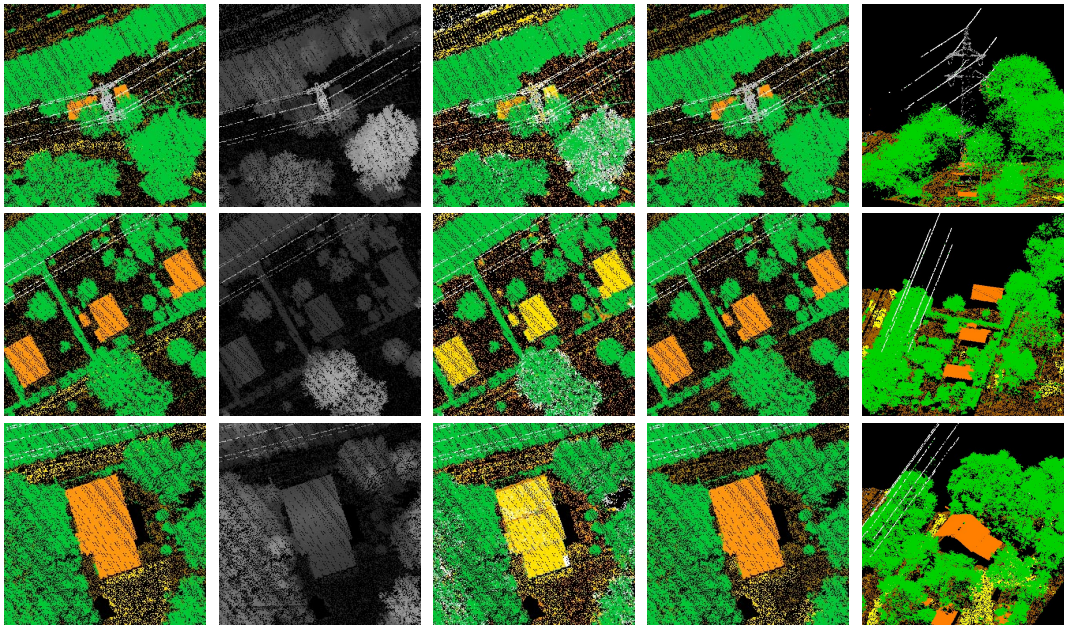
	ground	*vegetation*	*building*	*power line*	*tower*	*street*
ground	0.84	0.07	0.00	0.00	0.00	0.09
vegetation	0.03	0.97	0.00	0.00	0.00	0.00
building	0.01	0.06	0.93	0.00	0.00	0.00
power line	0.00	0.05	0.00	0.91	0.04	0.00
tower	0.01	0.07	0.00	0.04	0.88	0.00
street	0.30	0.01	0.00	0.00	0.00	0.69

FIGURE 6.12: Confusion matrices: each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. Values are normalized so that the sum of every row is equal to 1. Left: output of the waveform classifier (first stage). Right: Output of the U-net (second stage).

6.6.3 Testing

In order to report results that are independent from the training stage, to some extent, five trainings were performed independently, each time initializing the weights from scratch and randomly extracting the training dataset from the entire point cloud, as described in Section 6.6.2. The resulting overall accuracy, computed on the test set, is equal to $92.6(\pm 0.7)\%$, while the average per class accuracy is $87.0(\pm 0.3)\%$.
As can be noticed from the confusion matrix represented in Figure 6.12 (right), that reports the results for one out of the five trainings, the network performs very well for the classes *vegetation*, *building*, *power line* and *transmission tower*. Instead, points belonging to the class *street path* are often confused with the class *ground*. This is probably due to the fact

that the shape of the waveforms belonging to these two classes are often indistinguishable (see Figure 6.11) and also the geometric characteristics of *ground* and *street path* points can be very similar. In practical applications (e.g., for the creation of DTMs) these two classes are usually merged together. If we consider *ground* and *street path* as a unique class, the overall accuracy increases to 96.1(±0.2)% and the average per class accuracy to 92.5(±0.5)%.



(A) (B) (C) (D) (E)

FIGURE 6.13: Sample images (256 × 256) and results. (A) Ground truth images used for training and validation; (B) Height channel; (C) Labels predicted by the waveform classifier (maximum probability) that are fed to the U-net; (D) Labels produced by the U-net (maximum probability); (E) 3D views of the classified point cloud, coloured with the predicted labels. Classes: *ground* (brown), *vegetation* (green), *building* (orange), *power line* (white), *transmission tower* (grey), *street path* (yellow).

Although a direct comparison with other methods using full-waveform LiDAR is not possible, for the labelled full-waveform data used in our experiments is the first public dataset of this kind, Table 6.3 suggest that our method compares favourably with the state of the art (the table refers to the methods described in Section 6.2).

TABLE 6.3: Synopsis of state-of-the-art methods.

Ref	**# classes**	**Method**	**Accuracy**
Ours	6	CNN	92.6
Ours	5	CNN	96.1
[153]	2	threshold	89.9 - 93.7
[113]	3	SVM	95.3
[158]	9	SVM	92.6 (+ hyperspectral)
[117]	3	SOM	93.1

Examples of the input provided to the U-net model and of the obtained results for the test set are shown in Figures 6.13, 6.14 and 6.15.

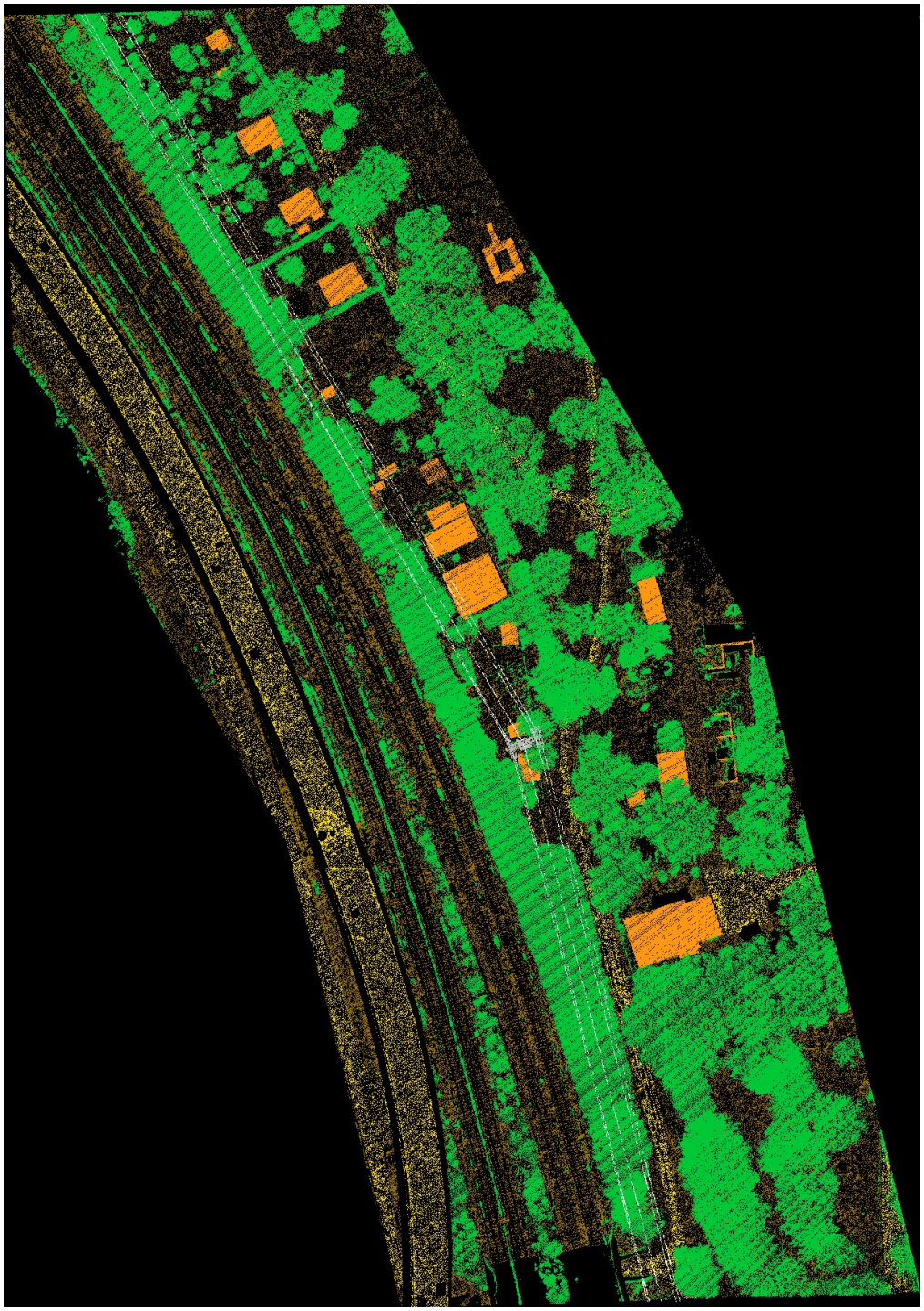


FIGURE 6.14: Ground truth.

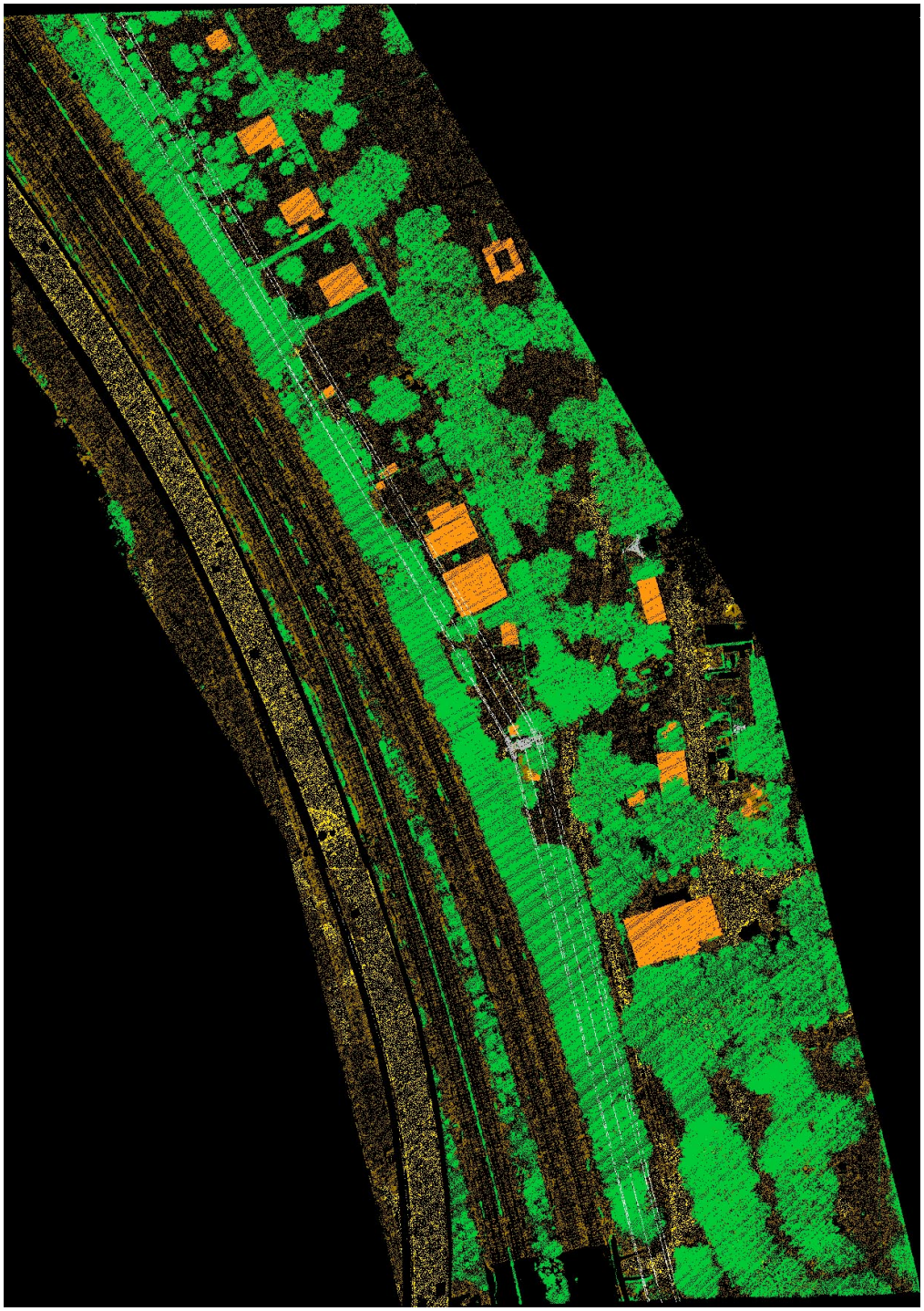


FIGURE 6.15: Point cloud classified by our method.

We also tested the performances of the waveform classifier alone (Section 6.5.1), which turns out to be unsatisfactory, for it reaches 61.1% overall accuracy in the test set. The confusion matrix shown in Figure 6.12 (left) indicates that some classes are merged together, namely *ground*, *building* and *street path*, and also the class *transmission tower* is often misclassified. This confirms that our approach reaches high accuracy in the point cloud classification thanks to the combination of full-waveform data and spatial support.
As previously mentioned, we tried to replace the waveform classifier with autoencoders with different code dimensions. The best performance was achieved with code dimensions equal to the number of classes, but the overall accuracy was only 84.9%. When merging the classes *ground* and *street path*, the overall accuracy increases to 89.5%.

6.7 Conclusion

In this chapter we presented an innovative algorithm based on CNNs to perform full-waveform LiDAR point cloud classification. The proposed network employs directly the raw full-waveform data, learning both features and classifier end-to-end, unlike other methods that require preliminary extraction of features. It can be applied to the classification of points belonging to any kind of area and no prior knowledge on the data characteristics is required.
Experiments report an overall accuracy of 92.6%, on six classes including challenging instances such as *power line* and *transmission tower*. Although a direct comparison with other methods using full-waveform LiDAR is not possible, experiments suggest that our method compares favourably with the state of the art. The labelled dataset that we will made available to the public domain will allow reproducibility and comparison by other authors.

Chapter 7

Seamless Image Mosaicking via Synchronization

This chapter proposes an innovative method to create high-quality seamless planar mosaics. The developed pipeline ensures good robustness against many common mosaicking problems (e.g., misalignments, colour distortion, moving objects, parallax) and differs from other works in the literature because a global approach, known as *synchronization*, is used for image registration and colour correction. To better conceal the mosaic seamlines, images are cut along specific paths, computed using a Voronoi decomposition of the mosaic area and a shortest path algorithm. Results obtained on challenging real datasets show that the colour correction mitigates significantly the colour variations between the original images and the seams on the final mosaic are not evident.

7.1 Introduction

Aligning and stitching together multiple images is a classical problem in photogrammetry [125] and computer vision [152, 145, 23]. Image mosaicking finds application in various scenarios, ranging from satellite or aerial imagery [45], street-view panoramas [101] or video stabilization [72], to name a few.
Since there are many technical difficulties in taking a photo with a very large *field of view* (FOV), often the only practical solution is to acquire multiple images with smaller FOV and merge them together. *Image mosaicking* can be therefore defined as the process of stitching different photos of the same scene in a single wide image. It can be performed independently from (and prior to) the structure-from-motion and dense matching phases, that are instead required to generate orthophotos. The goal of mosaic creation is, in fact, to visualize a wide area on a single image under perspective projection, whereas orthophotos are orthographic projections. Image mosaicking exploits homography as a transformation, thus a mosaic can be correctly created only if images are either captured from the same position or the images depict a planar surface in object space. When these conditions are not satisfied, the use of homography for image alignment can lead to a result in which parallax effects are evident and it is therefore necessary to resort to techniques that are able to conceal this issue and to create pleasant looking mosaics. The final mosaic should be as natural as possible, ideally indistinguishable from a real photo that covers the entire scene.
Aligning and stitching images into seamless mosaics is a procedure usually composed by three main steps: image registration, colour correction and blending. In the last decades several methods for automatic image mosaicking appeared in the literature, proposing a complete pipeline for the final mosaic generation [41, 116, 23] or focusing the attention on the optimization of one of the previously cited steps [135, 123, 101].
Algorithms for image alignment can be divided into two broad categories [145]: direct (pixel-based) and feature-based. Direct methods exploit the entire image data, thus providing very accurate registration but requiring at the same time a close initialization. Feature-based algorithms, instead, do not require initialization and can be computationally less expensive.

Moreover, since the introduction of invariant features (e.g. SIFT, [106]) and robust feature matching, feature-based methods have gained increasing attention and are nowadays widely used. [23] proved that, formulating stitching as a multi-image matching problem and using invariant local features to find matching between the images, lead to a method insensitive to the ordering, orientation, scale and illumination of the input images.
To obtain a clean, pleasant looking mosaic, a robust alignment process must be followed by colour correction. Neighbouring images can indeed show colour and appearance differences due to exposure level variation, changes in lighting condition and different camera settings. Colour correction methods proposed in the literature can be divided into model-based parametric approaches and non parametric ones [159]. The former assume that the relation between two images can be described by a colour transfer function, whereas the latter consider no particular parametric format of the colour mapping function and typically use a look-up table to directly record the mapping of the colour levels. [159] evaluated the performance of various colour correction approaches, showing how the gain compensation method by [23] and the local colour transfer approach by [146] are fast, effective and general (applicable in various scenarios).
Even after colour correction, seams and artifacts can be visible in the mosaic. Image blending techniques are able to conceal the colour differences along the seamlines but cannot handle residual geometric misalignment deriving from parallax and moving objects. For these reasons, it is mandatory to compute optimal seamlines that avoid crossing overlap regions with high image discrepancies. [41] used the Dijksta's algorithm [43] to compute the best cutting path dividing overlapping regions, segmenting the mosaic into disjoint regions and sampling pixels in each region from a single source image. [152] proposed a weighted vertex cover algorithm in order to remove effects caused by moving objects and [101] formulated the seamline optimization as a unified graph cuts energy minimization problem, concealing the image parallax in the resulting mosaic.

7.1.1 Contribution

The goal of this chapter is to develop an innovative procedure to create seamless mosaics exploiting a global approach, known as *synchronization* [139], described in Section 7.2. Starting from the geometric and radiometric information between pairs of overlapping images, the synchronization method is able to simultaneously estimate global homographies and colour corrections for all the images, avoiding the errors that accumulate when adding an image at a time to the mosaic. Finally, in order to minimize and conceal the seams that can still be visible after the global colour correction (e.g., due to parallax and moving objects), the cutting paths are determined using a Voronoi tessellation and optimized with the Dijkstra's algorithm [43]. The entire pipeline is presented in Section 7.3.
The experimental validation, illustrated in Section 7.4, was conducted on datasets composed by tens of images, acquired by a helicopter or an Unmanned Aerial Vehicle (UAV). Although the scenes are not perfectly planar, there are moving objects and strong illumination and intensity differences, the obtained mosaics appear homogeneous, without artifacts, and the seams are well concealed.

7.2 Synchronization

Given a network of nodes, where each node is characterized by an unknown state and pairs of nodes can measure the ratio (or difference) between their states, the goal of *synchronization* is to estimate the unknown states from the pairwise measures. The problem can be modeled as a graph where nodes correspond to the unknown states and edges encode the pairwise measures, and it is well-posed only if such a graph is connected.
As an example, consider the graph in Figure 7.1, where nodes and edges are labelled with integer numbers: the task is to recover the unknown numbers in the nodes by measuring their differences (on the edges). As one can notice, this problem corresponds to a topographic

levelling, where differences in elevation between pairs of points are measured, and one want to retrieve the height of each point. Two things can be immediately observed: a solution exists only if the sum of the differences along any cycle is zero, and, when it exists, the solution is not unique, for adding a constant to the nodes does not change the differences.

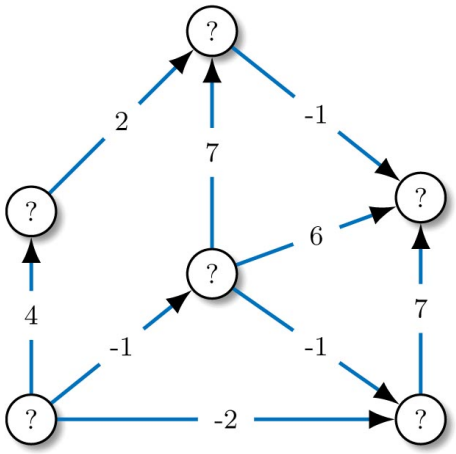


FIGURE 7.1: The synchronization problem. Each node is characterized by an unknown state and measures on the edges are differences (or ratio) of states. The goal is to compute the states that best agree with the measures.

Mathematically, states are represented by elements of a group Σ. Several instances of synchronization have been studied in the literature, which correspond to different instantiations of Σ. Among them, it is worth citing $SE(d)$ for *rigid-motion synchronization* [7], $SL(d)$ for *homograpy synchronization* [135] and $GA(d)$ for *affine matrix synchronization*. Please note that $SL(d)$ is a subgroup of $GL(d)$, whereas $SE(d)$ and $GA(d)$ are subgroups of $GL(d+1)$. Thanks to the formalism of synchronization, several photogrammetric and computer vision problems can be addressed without relying on features or points, since the problem is formulated in frame space, or, more abstractly, in a group.
In this chapter the attention is focused on synchronization over $SL(3)$, that will be applied for image registration, and over $GA(1)$ for colour correction.
In order to formally define the problem and its solution, let Σ be a group and let $*$ denote its operation. Suppose that the pairwise relations between the index pairs $(i,j) \subseteq \{1,..,n\}$ x $\{1,..,n\}$ are known, and refer to them as z_{ij}. *Synchronization* can be formulated as the problem of recovering $x_i \in \Sigma$ for $i = 1,..,n$ such that the following *consistency constraint* is satisfied

$$z_{ij} = x_i * x_j^{-1}. \tag{7.1}$$

The solution is defined up to a global (right) product with any group element, i.e., if $x_i \in \Sigma$ satisfies (7.1) then also $x_i * y$ satisfies (7.1) for any (fixed) $y \in \Sigma$.
If the known pairwise measures are noisy, the consistency constraint cannot be satisfied exactly. Thus, the searched solution is the one that minimizes the *consistency error*:

$$\epsilon(x_1, x_2, .., x_n) = \sum_{(i,j)} \delta(z_{ij}, x_i * x_j^{-1}) \tag{7.2}$$

where $\delta : \Sigma \times \Sigma \to \mathbb{R}^+$ is a metric function for Σ [5].

7.2.1 Synchronization over $(GL(d), \cdot)$

In this section we consider the synchronization problem over the General Linear Group $GL(d)$, which is the set of all $d \times d$ invertible matrices, where the group operation $*$ is

matrix multiplication and $\mathbf{1}_\Sigma = \mathbf{I}_d$. Let $\mathbf{X}_i \in \mathbb{R}^{d\times d}$ and $\mathbf{Z}_{ij} \in \mathbb{R}^{d\times d}$ denote the matrix representations of $x_i \in \Sigma$ and $z_{ij} \in \Sigma$, respectively. Using this notation, Equation (7.1) rewrites $\mathbf{Z}_{ij} = \mathbf{X}_i\mathbf{X}_j^{-1}$.
Let us collect the unknown group elements and all the measures in two matrices $\mathbf{X} \in \mathbb{R}^{dn\times d}$ and $\mathbf{Z} \in \mathbb{R}^{dn\times dn}$ respectively, which are composed of $d \times d$ blocks, namely

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \dots \\ \mathbf{X}_n \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} \mathbf{I}_d & \mathbf{Z}_{12} & \dots & \mathbf{Z}_{1n} \\ \mathbf{Z}_{21} & \mathbf{I}_d & \dots & \mathbf{Z}_{2n} \\ \dots & & & \dots \\ \mathbf{Z}_{n1} & \mathbf{Z}_{n2} & \dots & \mathbf{I}_d \end{bmatrix}. \tag{7.3}$$

If not all the pairwise measures $\mathbf{Z}_{ij}$ are available, the input matrix becomes $\mathbf{Z}_\mathbf{A} := \mathbf{Z} \odot (\mathbf{A} \otimes \mathbf{1}_{d\times d})$, where $\odot$ denotes the Hadamard product, $\mathbf{A}$ is the adjacency matrix and the Kronecker product with $\mathbf{1}_{d\times d}$ is required to match the block structure of the measures. The $n \times n$ adjacency matrix is constructed as follows: $\mathbf{A}_{ij} = 1$ if the pairwise measure $\mathbf{Z}_{ij}$ exists, $\mathbf{A}_{ij} = 0$ otherwise. Accordingly, the consistency constraint writes

$$\mathbf{Z}_A = (\mathbf{X}\mathbf{X}^{-\flat}) \odot (\mathbf{A} \otimes \mathbf{1}_{d\times d}) \tag{7.4}$$

where $\mathbf{X}^{-\flat} \in \mathbb{R}^{d\times dn}$ denotes the block-matrix containing the inverse of each $d \times d$ block of $\mathbf{X}$.
It can be shown [4] that

$$\mathbf{Z}_\mathbf{A}\mathbf{X} = (\mathbf{D} \otimes \mathbf{I}_d)\mathbf{X} \tag{7.5}$$

thus an estimate of $\mathbf{X}$ is represented by the eigenvectors of $(\mathbf{D} \otimes \mathbf{I}_d)^{-1}\mathbf{Z}_\mathbf{A}$ corresponding to the d largest eigenvalues, where $\mathbf{D}$ is the degree matrix defined as $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_{n\times 1})$. This is also called the spectral solution.

7.2.2 Synchronization over $SL(d)$

Consider now the Special Linear Group $SL(d)$, that is the set of $d \times d$ matrices with unit determinant

$$SL(d) = \{\mathbf{R} \in \mathbb{R}^{d\times d} \; s.t. \; \det(\mathbf{R}) = 1\}. \tag{7.6}$$

Synchronization over $SL(3)$ corresponds to the *homography synchronization* problem. Since $SL(d)$ is a subgroup of $GL(d)$, the problem can be addressed via the spectral solution, which computes the top d eigenvectors of $(\mathbf{D} \otimes \mathbf{I}_d)^{-1}\mathbf{Z}_\mathbf{A}$, that are collected in a $dn \times d$ matrix $\mathbf{U}$. In order to obtain elements of $SL(d)$ from $\mathbf{U}$, each $d \times d$ block in $\mathbf{U}$, denoted by $\mathbf{U}_i$, must be scaled to unit determinant [135], which can be done by dividing $\mathbf{U}_i$ by $\sqrt[d]{\det(\mathbf{U}_i)}$. However, if $\det(\mathbf{U}_i)$ is negative and d is even, real roots do not exist; in this case the determinant can be always made positive by exchanging two columns of $\mathbf{U}$.

7.2.3 Synchronization over $GA(d)$

Let us finally consider the Affine Group $GA(d)$, that is the set of invertible affine transformations in d-space, which admits a matrix representation through $(d+1) \times (d+1)$ matrices

$$GA(d) = \left\{ \begin{bmatrix} \mathbf{M} & \mathbf{v} \\ \mathbf{0}' & 1 \end{bmatrix}, \; s.t. \; \mathbf{M} \in \mathbb{R}^{d\times d}, \mathbf{v} \in \mathbb{R}^d \right\}. \tag{7.7}$$

$GA(d)$ is a subgroup of $GL(d+1)$, therefore the synchronization problem can be solved by computing the top $d+1$ eigenvectors of $(\mathbf{D} \otimes \mathbf{I}_{d+1})^{-1}\mathbf{Z}_\mathbf{A}$. Since this approach leads to an algebraic solution, it does not enforce constraints that matrices in $GA(d)$ should satisfy.
Specifically, the output matrix $\mathbf{U}$ will not have vector $[\mathbf{0}_{1\times d} \; 1]$ in rows multiple of $d+1$. In order to recover $\mathbf{X}$ from $\mathbf{U}$ it is sufficient to choose a different basis for the resulting eigenvectors that satisfies such constraint, which can be found by taking a suitable linear

combination of the columns of $\mathbf{U}$, as explained in [7]. More precisely, let $\mathbf{F} \in \mathbb{R}^{n\times(d+1)n}$ be the 0/1-matrix such that $\mathbf{FU} \in \mathbb{R}^{n\times(d+1)}$ consists of the rows of $\mathbf{U}$ with indices multiple of $d+1$. The coefficients $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{d+1}$ of the linear combination are solution of

$$\mathbf{FUa} = \mathbf{0}_{n\times 1}, \quad \mathbf{FUb} = \mathbf{1}_{n\times 1} \tag{7.8}$$

where the first equation has a d-dimensional solution space. Let $\mathbf{a}_1, \ldots, \mathbf{a}_d$ be a basis for the null-space of $\mathbf{FU}$. Thus $\mathbf{X}$ is recovered as $\mathbf{X} = \mathbf{U}[\mathbf{a}_1, \ldots, \mathbf{a}_d, \mathbf{b}]$.
In the presence of noise, Equation (7.8) is solved in the least squares sense. Then, such a solution is projected onto $GA(d)$ by forcing the rows multiple of $d+1$ to $[\mathbf{0}_{1\times d}\ 1]$.

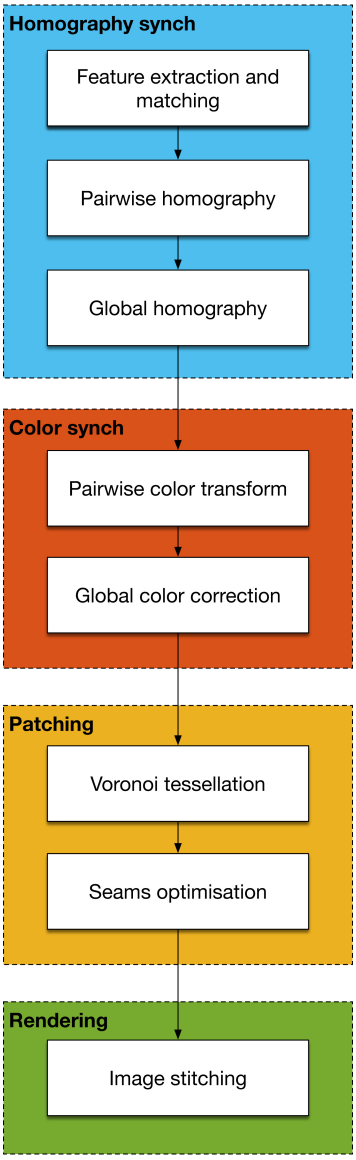


FIGURE 7.2: Flowchart of the proposed method.

7.3 Proposed Method

The novel algorithm proposed in this paper tries to overcome some common issues in mosaic generation (e.g. misalignments, colour correction, moving objects) thanks to the use

of synchronization and the search for an optimal cutting path between overlapping images. The entire process is summarized in Figure 7.2 and described in detail in the following paragraphs.

7.3.1 Image Alignment

The first step of the proposed procedure is to extract features from all the images (e.g. SIFT features, [106]) and match them. A robust feature matching algorithm should be used in order to avoid wrong matches that can cause strong misalignments between the images. For this reason, the method proposed in Chapter 3 has been chosen.
Pairwise homographies are then robustly estimated using RANSAC [52], computing image transformation parameters through the Direct Linear Transformation (DLT) method [1]. A possible solution to project all the images in the same reference system for mosaic generation is to compose relative transformations multiplying the obtained pairwise homographies. However, this approach accumulates error at each successive multiplication. To solve this problem, synchronization over $SL(3)$ (see Section 7.2.2) is applied, converting in this way pairwise homographies into absolute ones. This guarantees that all relative information are considered simultaneously, minimizing misalignment errors among the whole dataset. The process of homography synchronization is illustrated in Figure 7.3.

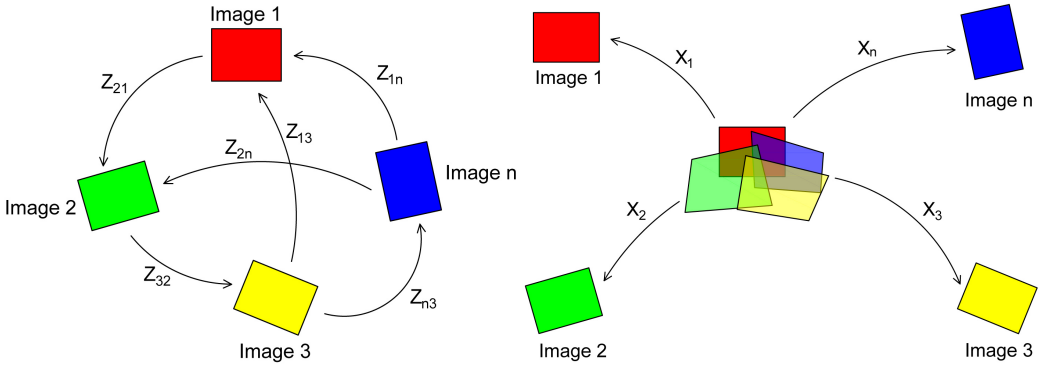


FIGURE 7.3: Pairwise homographies (A) are converted into absolute ones (B) via synchronization.

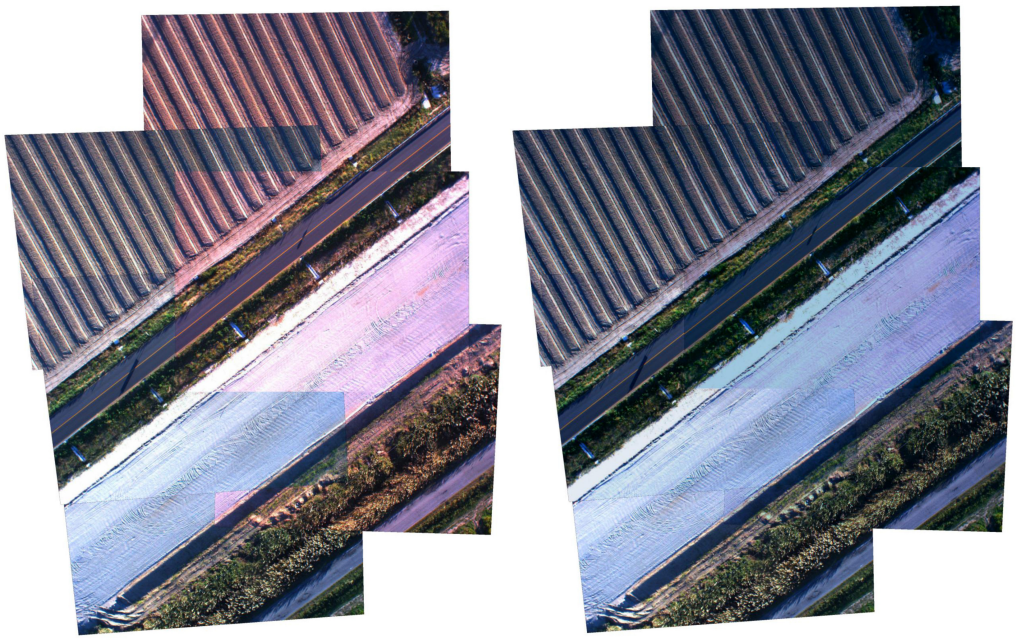
To improve the accuracy of the synchronization process, a weighing factor can be assigned to each pairwise homography, that describes its reliability. In practice, the unitary elements of the adjacency matrix $\mathbf{A}$ contained in Equation (7.4) are replaced by the estimated weights. In the proposed procedure, these weights are assumed to be proportional to the area of the convex hull that contains the features matched in each image pairs.

7.3.2 Colour Correction

Changes of the illumination conditions, different camera settings and vignetting are some of the causes that make the seams of the mosaic visible, even when the scene is planar, the images are sharp and the alignment is perfect. Colour variations between overlapping images should be modelled by a nonlinear function and often involve the three colour channels simultaneously. However, the simplified approach that considers the RGB channels independently and that models the transformation with an affinity proved to work well. Thus, in the proposed method the relation between the three colour channels of adjacent images (i, j) is assumed to be an affine transformation, that can be written in matrix form as

$$\begin{bmatrix} C \\ 1 \end{bmatrix}_i = \begin{bmatrix} a_c & b_c \\ 0 & 1 \end{bmatrix}_{i,j} \cdot \begin{bmatrix} C \\ 1 \end{bmatrix}_j \tag{7.9}$$

where C is in turn R,G, or B. Formulating the problem in this way corresponds to estimating the parameters of three affine transformations between each pair of overlapping images, that have to be then composed in order to compute a global colour correction for each single image. It is easy to see that this problem can be solved via the synchronization over the *Affine Group* $GA(1)$, described in Section 7.2.3.
In the presence of small residual misalignments, a pixel-based method used to estimate the pairwise affine transformations can lead to inaccurate results. An alternative robust approach, adopted in this chapter, consists in exploiting the histograms of the overlapping area computed for both images. The parameters of the affine transformation are computed as the angular coefficient and intercept of the straight line that fits the plot of one cumulative histogram versus the other cumulative histogram [35].
Once all the relative affine transformations have been computed for each colour channel, the absolute ones can be retrieved via synchronization, as done for the homographies. A weighing matrix can be introduced, where the weights are proportional to the overlapping area size, in order to give more confidence to the most reliable pairwise colour transformation. Please note that synchronization retrieves absolute affine transformation, up to a global one. This degree of freedom can be fixed by choosing one image that does not undergo colour correction. The unaltered image can be identified automatically as the one that has the best colour balance, or it can be defined by the user.
An example of the results achieved with this approach is represented in Figure 7.4.



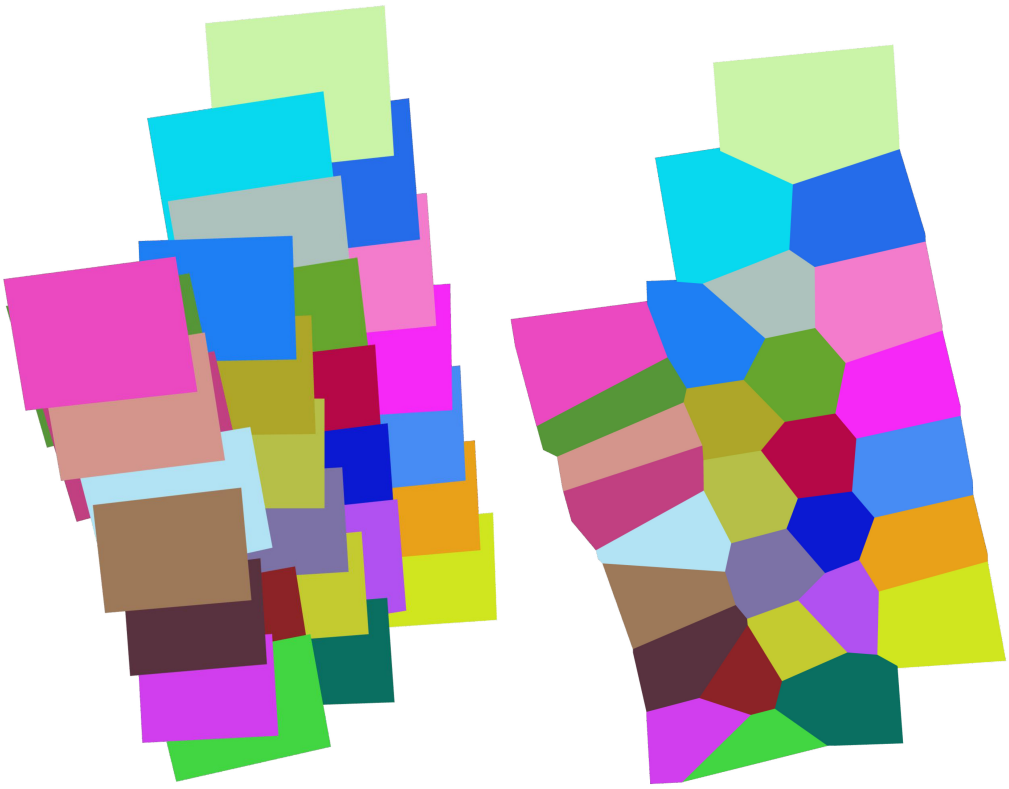
(A) (B)

FIGURE 7.4: Mosaic before (A) and after (B) colour correction.

7.3.3 Voronoi Tessellation and Seams Optimization

Even if image alignment and colour correction produce optimal results, seams can still be visible on the final mosaic due to, e.g., parallax or moving objects. We therefore propose an approach that first reduces the seamlines total length and then conceals the remaining ones. First of all, it is advisable to remove redundant images, i.e., images completely covered by the adjacent ones. Using all the images can indeed generate a mosaic composed by many little patches, increasing at the same time the total length of the seamlines. This issue

can be faced searching for a subset of images with the property that no one is completely covered by the union of the others. An iterative greedy approach can be followed to discard redundant elements, considering an image at a time and evaluating if its projection in the mosaic reference frame is completely covered by the projection of the other images. If so, the current image is discarded and the dataset is updated. The process is repeated until all the remaining elements are verified. The drawback of this procedure is that it is order-dependent and does not guarantee that the minimum subset is kept. The process can be driven by arranging the dataset in a convenient way, since the first images analyzed are more likely to be discarded. A possible choice is to sort the images according to their colour balancing or to the alignment error.
The previously described step reduces the number of tiles that compose the mosaic. To find a method that further reduces the overall length of the seams, one can start from the following considerations. The *honeycomb conjecture* [71] states that the hexagonal tiling is the best way to divide a surface into regions of equal area with the least total perimeter. Moreover, the Voronoi tessellation of 2D lattices of points gives an irregular honeycomb tessellation. For these reasons, the Voronoi decomposition can be applied to approximately minimize the total length of the seams. The seed points for the Voronoi tessellation are assumed to be the centroid of the images projected in the mosaic plane; for each seed the method determines a corresponding region consisting of all points closer to that seed than to any other.



(A) (B)

FIGURE 7.5: (A) Image frames projected onto the mosaic reference. (B) Voronoi tessellation.

Figure 7.5(A) shows the area covered by each image after projecting it on the mosaic surface, whereas Figure 7.5(B) represents the area assigned to the images using Voronoi decomposition.
Please note that we assume that the Voronoi region is contained in the corresponding image; this happens in most cases, but it cannot be formally guaranteed unless a constrained

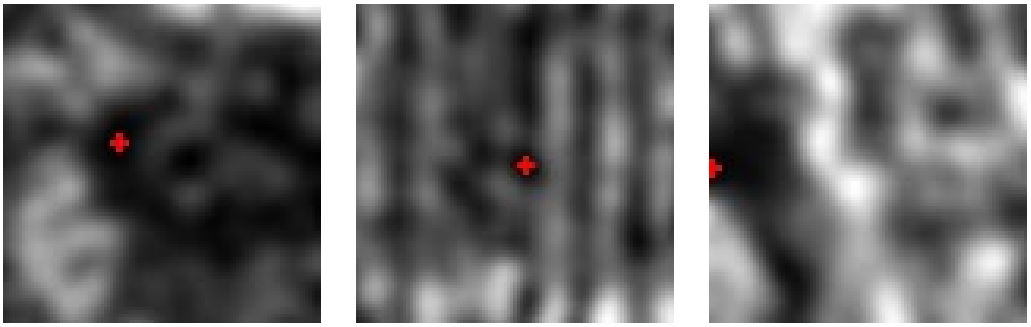
tessellation is used.
Finally, the best cutting paths between overlapping images are computed, in order to avoid the creation of seamlines that pass through regions where there are significant differences between adjacent images. For each seamline, the following procedure is used:

1. *Costmap computation:* a costmap that provides the information for the search of the best cutting path is constructed as the squared difference (pixel by pixel) between the images in the overlap area.

2. *Costmap to graph conversion:* the 2D costmap is converted into a graph assuming that each pixel becomes a node and that each adjacent nodes pair is connected by two oriented edges. The edge takes the weight from the value of the end pixel.

3. *Best path identification:* Dijkstra's algorithm [43] is used over the generated graph to find the path of minimum cost.

When multiple images overlap on the same area, a costmap is computed for each image pairs and then averaged.
The procedure described requires as input the starting and end points of the seamlines. Since the Voronoi vertices computed before are independent from the image contents (they depend only on the position of the image centroids), they can fall in an area where the differences between adjacent images are high, thus negatively influencing the search of the best cutting path. To maintain the Voronoi polygons and at the same time determine optimal cutting paths, starting and end points are chosen as the ones that have the lowest cost in a neighbourhood of each Voronoi vertex, as shown in Figure 7.6.



(A) (B) (C)

FIGURE 7.6: Costmap for the vertex optimization. The chosen vertices are represented by a red cross.

Figure 7.7 reports an example of the seamlines detected through the proposed procedure. First the Voronoi tessellation is performed (Figure 7.7(A)), then straight cuts between adjacent images are substituted by the optimal seamlines obtained through the Dijkstra's algorithm (Figure 7.7(B)). One can notice that the cutting paths follow the space between the trees and rarely cross them. This is due to the fact that the ground between the trees is less textured and the cost corresponding to this zone is low. Thus, during the best cutting path computation, this area is preferred over the ones covered by trees.
The whole procedure described in this section determines a segmentation of the mosaic plane into disjoint regions. The final mosaic is obtained by filling each region with pixels sampled from a single source image.

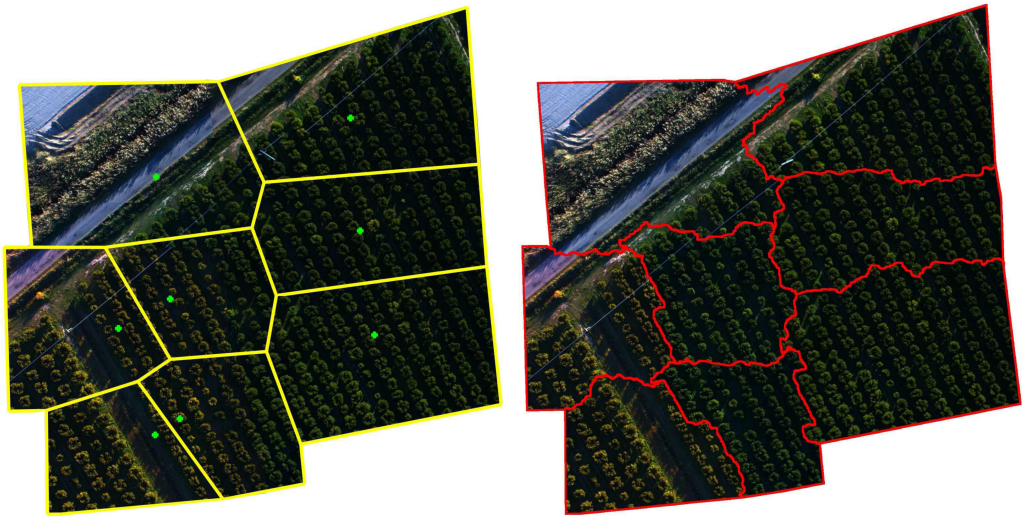


FIGURE 7.7: Seams produces by Voronoi tessellation (left) and after optimization.

7.4 Experimental Validation

The proposed method was validated on two challenging dataset of aerial images. The results were compared to the ones obtained with *AutoStitch*[1] [23], a popular software for image stitching.

Dataset 1 is composed by 29 images of size 1600×1200 pixels, acquired by Helica s.r.l. with a helicopter during a four strips flight. The images are characterized by small overlap areas and varying camera settings were used for each strip acquisition, thus the images present significant colour differences that have to be corrected during the mosaic generation. In particular, some images have a preponderant unrealistic red colour. The scene is almost planar (parallax effects are limited) but there are moving vehicles.

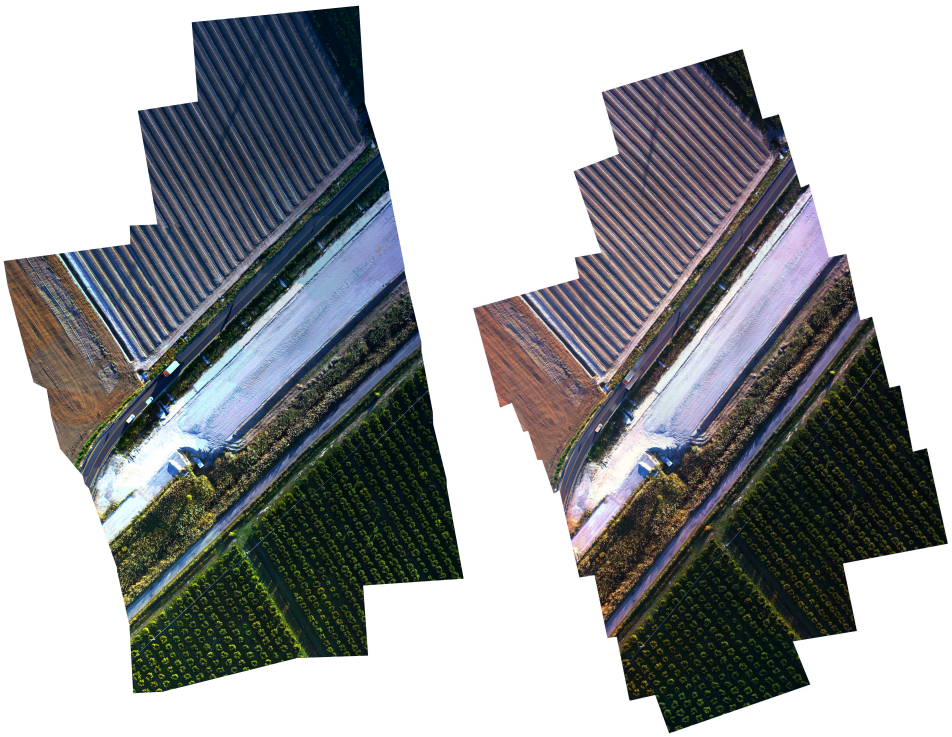
Figure 7.8 shows the mosaic created by aligning the images via homography synchronization (see Section 7.3.1) and stitching them together according to a simple painter's algorithm (newer images overwrites older ones). Small misalignments can be noticed and strong colour distortions are evident between adjacent strips.

The result obtained with the complete method proposed in Section 7.3 is represented in Figure 7.9(A). Please note how the colours have been corrected with the affinity synchronization step (Section 7.3.2) and differences between adjacent strips have been removed (reddish images visible in Figure 7.8 are now indistinguishable). Moreover, misalignments have been concealed and the seamline optimization carried out with the Dijkstra's algorithm (Section 7.3.3) preserved the integrity of the moving objects and avoided ghosting effects. A limited number of seams are still visible on the street and they could be easily eliminated by applying a blending algorithm at the end of the proposed procedure. The mosaic generated with *AutoStitch* is shown in Figure 7.9(B). The software uses gain compensation and blending to conceal seams between overlapping images. The obtained mosaic is good, no misalignment is visible and the colour correction algorithm worked quite well. However, images with a strong red component are still visible and the blending in correspondence of moving objects caused ghosting effects.

Dataset 2 is composed by 27 images of size 4000×3000 pixels, acquired with a UAV with a forward and a side overlap of 80%. The scene is not planar (there is a large building in the middle and high trees in the bottom left part) and this determines strong misalignments after the homography computation. There are no evident colour differences except for weak illumination changes. No moving objects can be noticed in the images.

[1] available at http://matthewalunbrown.com/autostitch/autostitch.html

FIGURE 7.8: Dataset 1: Mosaic obtained after image alignment, with no colour correction and seamline optimization.



(A) (B)

FIGURE 7.9: Dataset 1. (A) Mosaic obtained with the proposed method. (B) Mosaic obtained with *AutoStitch*.

FIGURE 7.10: Dataset 2: Mosaic obtained after image alignment, with no colour correction and seamline optimization.

(A) (B)

FIGURE 7.11: Dataset 2. (A) Mosaic obtained with the proposed method. (B) Mosaic obtained with *AutoStitch*.

Figure 7.10 shows the mosaic generated after homography synchronization, stitching images together according to the painter's algorithm, whereas Figure 7.11(A) represents the final mosaic, in which colour variations and visible cuts are perfectly concealed. Due to the high overlap, 13 images were removed because completely covered by the others. Minimizing the number of images used to cover the entire scene results in seams of shorter length and reduces the misalignment and parallax visible effects. For this dataset, the mosaic generated with *AutoStitch* (Figure 7.11(B)) is not satisfactory. The blending algorithm in presence of strong parallax caused blurred areas and other artifacts.

7.5 Conclusion

In this chapter we proposed a novel procedure for image mosaicking, based on the approach known as *synchronization*. The image alignment and colour correction steps that usually characterize mosaic generation were addressed as a homography synchronization and an affine synchronization problem, respectively. Synchronization can be seen as upgrading from relative information, which involves two overlapping images at a time, onto absolute information, which involves all the images simultaneously. Thanks to this approach, misalignments and colour differences are globally minimized, thus leading to the generation of homogeneous, visually appealing mosaics. In order to conceal the seams that can still be visible after image registration and colour correction, a final step is applied, that is composed by Voronoi tessellation and seams optimization via the Dijkstra's algorithm [43]. Results shown in Section 7.4 proved that this novel procedure generates pleasant looking mosaic, without resorting to blending algorithms. As a matter of fact, each pixel of the resulting mosaic comes from a *single* image.
More experiments will be performed in the future, in order to assess the validity of the proposed method also for different kind of datasets, such as satellite images.

Chapter 8

Conclusion

In the first part of this thesis, **orientation problems** in photogrammetry and laser scanning were studied from a methodological point of view and solved via *Procustes Analysis*, a set of least squares mathematical tools used to perform transformations among corresponding points belonging to a generic k-dimensional space, in order to satisfy their maximum agreement. We provided a comprehensive survey on Procrustes Analysis, gathering several models that were proposed in the last decades by different communities, and proposed novel Procrustes models that can be useful in several applications.
First of all, we applied an *Errors-In-Variables* formulation to Procrustes Analysis and derived total least squares solutions that can deal with the uncertainty affecting both sets of observations.
We then addressed the problem of establishing correspondences between keypoint sets, that can be formulated as a Permutation Procrustes Analysis when two sets are involved. In particular, we proposed a novel solution to the multi-view matching problem that, given a set of noisy pairwise correspondences, jointly updates them so as to maximize their consistency. Our method is based on a spectral decomposition, resulting in a closed-form efficient algorithm, in contrast to other iterative techniques that can be found in the literature. We also embedded the developed multi-view matching inside the framework of the *Generalized Procrustes Analysis* (GPA), a well known technique used to compute transformations among multiple element sets. The new GPA formulation that we proposed permits to overcome the main limitation of the classical model, i.e., correspondences among matrix elements must be known in advance.
Finally, we implemented a variation of the classical *Extended Orthogonal Procrustes Analysis*, called *Affine Extended Orthogonal Procrustes Analysis* (Affine-EOPA), that allows to compute the transformation between two matrices composed by both points and vectors. Motivated by a Virtual Trial Assembly (VTA) application, we derived from Affine-EOPA an innovative model which is capable of catering for undetermined motion components, i.e., can retrieve the best transformation among corresponding matrix elements under the condition that the position of each plane is undetermined along its normal. The proposed method was successfully applied for the VTA of *Vessel* in New York.
The second part of this work was instead devoted to **remote sensing** applications.
We described a novel method to perform LiDAR point cloud classification, that is based on Convolutional Neural Networks and takes advantage of full-waveform data registered by modern laser scanners. Thanks to the employed architecture, even challenging classes such as power line and transmission tower can be automatically identified.
Moreover, we developed an innovative procedure to create seamless mosaics, exploiting a global approach known as *synchronization*. Starting from the geometric and radiometric information between pairs of overlapping images, synchronization allows to simultaneously estimate global homographies and colour corrections for all the images, avoiding the errors that accumulate when adding an image at a time to the mosaic. A final seamline optimization step leads to the creation of high-quality planar mosaics.

The algorithms developed in the second part of the thesis could represent valuable tools to reduce time and costs of the remote sensing data processing, facilitating the use and interpretation of the acquired data.

Bibliography

[1] Y. Abdel-Aziz and H. Karara. Direct linear transformation into object space coordinates in close-range photogrammetry. In *Proc. Symp. close-range Photogrammetry, Univ. Illinois at Urbana-Champaign*, pages 1–18, 1971.

[2] C. Alexander, K. Tansey, J. Kaduk, D. Holland, and N. J. Tate. Backscatter coefficient as an attribute for the classification of full-waveform airborne laser scanning data in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(5):423–432, 2010.

[3] A. R. Amiri-Simkooei and S. Jazaeri. Weighted total least squares formulated by standard least squares theory. *Journal of Geodetic Science*, 2(2):113–124, 2012.

[4] M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri. Global motion estimation from point matches. *Proceedings of the Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2012.

[5] F. Arrigoni, A. Fusiello, and B. Rossi. Camera motion from group synchronization. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 546–555, 2016.

[6] F. Arrigoni, E. Maset, and A. Fusiello. Synchronization in the symmetric inverse semigroup. In *International Conference on Image Analysis and Processing*, pages 70–81. Springer, 2017.

[7] F. Arrigoni, B. Rossi, and A. Fusiello. Spectral synchronization of multiple views in SE(3). *SIAM Journal on Imaging Sciences*, 9(4):1963 – 1990, 2016.

[8] K. S. Arun. A unitarily constrained total least squares problem in signal processing. *SIAM Journal on Matrix Analysis and Applications*, 13(3):729–745, 1992.

[9] J. L. Awange. Partial procrustes solution of the three dimensional orientation problem from gps/lps observations. *In: Krumm F, Schwarze vs (eds) Quo vadis geodesia?*, pages 41–51, 1999.

[10] J. L. Awange and E. W. Grafarend. *Solving algebraic computational problems in geodesy and geoinformatics: The Answer to Modern Challenges*. Springer, 2005.

[11] A. Beinat. *Tecniche di analisi procustiana e trasformazioni di datum in topografia e fotogrammetria*. PhD thesis, Dipartimento di Ingegneria Idraulica Ambientale e del Rilevamento, Politecnico of Milan, 2000.

[12] A. Beinat and F. Crosilla. Generalized procrustes analysis for size and shape 3D object reconstruction. In *Optical 3-D Measurement Techniques*, pages 345–353. Wichmann Verlag, 2001.

[13] A. Beinat and F. Crosilla. Procrustes statistics to test for significant ols similarity transformation parameters. In *V Hotine-Marussi Symposium on Mathematical Geodesy*, pages 113–119. Springer, 2004.

[14] A. Beinat, F. Crosilla, and D. Visintini. Examples of georeferenced data transformations in GIS and digital photogrammetry by procrustes analysis techniques. *The International Archives of Photogrammetry and Remote Sensing*, XXXII, 6W8/2:2–5, 2000.

[15] M. Bennani Dosse and J. M. F. ten Berge. Anisotropic orthogonal procrustes analysis. *Journal of Classification*, 27(1):111–128, 2010.

[16] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[17] F. Boas. The horizontal plane of the skull and the general problem of the comparison of variable forms. *Science*, 21(544):862–863, 1905.

[18] F. L. Bookstein. Size and shape spaces for landmark data in two dimensions. *Statistical Science*, 1(2):181–222, 1986.

[19] F. L. Bookstein. *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press, 1997.

[20] I. Borg and P. Groenen. Modern multidimensional scaling: theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.

[21] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802 – 804, 1971.

[22] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[23] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[24] F. Case, A. Beinat, F. Crosilla, and I. Alba. Virtual trial assembly of a complex steel structure by generalized procrustes analysis techniques. *Automation in Construction*, 37:155–165, 2014.

[25] D. Ceglarek, W. Huang, S. Zhou, Y. Ding, R. Kumar, and Y. Zhou. Time-based competition in multistage manufacturing: Stream-of-variation analysis (sova) methodology. *International Journal of Flexible Manufacturing Systems*, 16(1):11–44, 2004.

[26] G. Chang. On least-squares solution to 3d similarity transformation problem under gauss–helmert model. *Journal of Geodesy*, 89:573–576, 2015.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[28] Y. Chen, L. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. In *Proceedings of the International Conference on Machine Learning*, pages 100–108, 2014.

[29] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic geometry and its applications*. John Wiley & Sons, 2013.

[30] F. Chollet. Keras. https://github.com/ fchollet/keras, 2015.

[31] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.

[32] N. Cliff. Orthogonal rotation to congruence. *Psychometrika*, 31(1):33–42, 1966.

[33] T. M. Cole. Historical note: early anthropological contributions to "geometric morphometrics". *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, 101(2):291–296, 1996.

[34] J. J. F. Commandeur. *Matching configurations*. DSWO Press, 1991.

[35] I. J. Cox, S. Hingorani, B. M. Maggs, and S. B. Rao. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[36] F. Crosilla. A criterion matrix for the second order design of control networks. *Bulletin géodésique*, 57(1-4):226–239, 1983.

[37] F. Crosilla. Procrustean transformation as a tool for the construction of a criterion matrix for control networks. *Manuscripta geodetica*, 8, 1983.

[38] F. Crosilla. Procrustes analysis and geodetic sciences. In *Quo vadis geodesia...?*, volume 1, pages 69–78. Department of Geodesy and GeoInformatics, University of Stuttgart, 1999.

[39] F. Crosilla and A. Beinat. Use of generalised procrustes analysis for the photogrammetric block adjustment by independent models. *ISPRS Journal of Photogrammetry & Remote Sensing*, 56(3):195–209, 2002.

[40] F. Crosilla and A. Beinat. A forward search method for robust generalised procrustes analysis. In *Data analysis, classification and the forward search*, pages 199–208. Springer, 2006.

[41] J. Davis. Mosaics of scenes with moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 354–360, 1998.

[42] J. de Leeuw. Block-relaxation algorithms in statistics. In *Information Systems and Data Analysis*, pages 308–325. Springer-Verlag, 1994.

[43] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[44] I. Dryden and K. Mardia. *Statistical shape analysis*. Wiley New York, 1998.

[45] Q. Du, N. Raksuntorn, A. Orduyilmaz, and L. M. Bruce. Automatic registration and mosaicking for airborne multispectral image sequences. *Photogrammetric Engineering & Remote Sensing*, 74(2):169–181, 2008.

[46] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[47] V. Ducic, M. Hollaus, A. Ullrich, W. Wagner, and T. Melzer. 3d vegetation mapping and classification using full-waveform laser scanning. In *Proc. Workshop on 3D Remote Sensing in Forestry. EARSeL/ISPRS, Vienna, Austria, 14–15 February 2006*, pages 211–217, 2006.

[48] X. Fang. Weighted total least-squares with constraints: a universal formula for geodetic symmetrical transformations. *Journal of Geodesy*, 89(5):459–469, 2015.

[49] Y. Felus and R. C. Burtch. On symmetrical three-dimensional datum conversion. *GPS Solutions*, 13(1):65–74, 2009.

[50] Y. Felus and B. Schaffrin. Performing similarity transformations using the errors-in-variables-model. In *Proceedings of the ASPRS Meeting, Baltimore/MD*, 2005.

[51] K. D. Fieber, I. J. Davenport, J. M. Ferryman, R. J. Gurney, J. P. Walker, and J. M. Hacker. Analysis of full-waveform lidar data for classification of an orange orchard scene. *ISPRS journal of photogrammetry and remote sensing*, 82:63–82, 2013.

[52] M. A. Fischler and R. C. Bolles. Random Sample Consensus: a paradigm model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[53] W. Förstner and B. P. Wrobel. *Photogrammetric computer vision*. Springer, 2016.

[54] A. Fusiello and F. Crosilla. Solving bundle block adjustment by generalized anisotropic procrustes analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 102:209–221, 2015.

[55] A. Fusiello and F. Crosilla. Fast and resistant procrustean bundle adjustment. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:35–41, 2016.

[56] A. Fusiello, E. Maset, and F. Crosilla. Reliable exterior orientation by a robust anisotropic orthogonal procrustes algorithm. In *Proceedings of the ISPRS Workshop 3D-ARCH 2013: 3D Virtual Reconstruction and Visualization of Complex Architectures*, volume XL-5/W1 of *ISPRS Archives*, pages 81–86, 2013.

[57] V. Garro, F. Crosilla, and A. Fusiello. Solving the pnp problem with anisotropic orthogonal procrustes analysis. In *Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 262–269, 2012.

[58] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[59] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 17(2):883–893, 1980.

[60] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[61] C. R. Goodall. Procrustes Methods in the Statistical Analysis of Shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):285–339, 1991.

[62] C. R. Goodall and P. B. Green. Quantitative analysis of surface growth. *Botanical Gazette*, 147(1):1–15, 1986.

[63] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[64] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[65] C. Gordon, F. Boukamp, D. Huber, E. Latimer, K. Park, and B. Akinci. Combining reality capture technologies for construction defect detection: A case study. In *EIA9: E-Activities and Intelligent Support in Design and the Built Environment, 9th EuropIA International Conference*, pages 99–108, 2003.

[66] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[67] J. C. Gower and G. B. Dijksterhuis. *Procrustes problems*. Oxford Statistical Science Series. Oxford University Press, 2004.

[68] E. W. Grafarend and J. L. Awange. Determination of the vertical deflection by gps/lps measurements. *Zeitschrift für Vermessungswesen*, 125(8):279–288, 2000.

[69] E. W. Grafarend and J. L. Awange. Nonlinear analysis of the three-dimensional datum transformation [conformal group $\mathbb{C}_7(3)$]. *Journal of Geodesy*, 77(1-2):66–76, 2003.

[70] B. F. Green. The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, 17(4):429–440, 1952.

[71] T. C. Hales. The honeycomb conjecture. *Discrete & Computational Geometry*, 25(1):1–22, 2001.

[72] M. Hansen, P. Anandan, K. Data, G. Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *Proceedings of IEEE Workshop on Applications of Computer Vision*, pages 54–62, 1994.

[73] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[75] G. E. Hinton, D. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[76] B. Höfle, M. Hollaus, and J. Hagenauer. Urban vegetation detection using radiometrically calibrated small-footprint full-waveform airborne lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:134–147, 2012.

[77] X. Hu and Y. Yuan. Deep-learning-based classification for dtm extraction from als point cloud. *Remote sensing*, 8(9):730, 2016.

[78] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Eurographics Symposium on Geometry Processing*, volume 32, pages 177–186, 2013.

[79] T. Huang, H. Li, H. Guo, N. Chan, S. Kong, G. Chan, and M. Skitmore. Construction virtual prototyping: a survey of use. *Construction Innovation*, 9(4):420–433, 2009.

[80] W. Huang and Z. Kong. Simulation and integration of geometric and rigid body kinematics errors for assembly variation analysis. *Journal of manufacturing systems*, 27(1):36–44, 2008.

[81] J. R. Hurley and R. B. Cattell. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral science*, 7(2):258–262, 1962.

[82] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[83] K. Jarrett, K. Kavukcuoglu, Y. LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[84] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[85] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

[86] K. Kenobi and I. L. Dryden. Bayesian matching of unlabeled point sets using procrustes and configuration models. *Bayesian Analysis*, 7(3):547–566, 2012.

[87] J. T. Kent. The complex bingham distribution and shape analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(2):285–299, 1994.

[88] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3D models using fuzzy correspondences. *ACM Transactions on Graphics*, 31(4), 2012.

[89] D. Kinga and J. B. Adam. A method for stochastic optimization. In *International Conference on Learning Representations*, volume 5, 2015.

[90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[91] M. A. Koschat and D. F. Swayne. A weighted procrustes criterion. *Psychometrika*, 56(2):229–239, 1991.

[92] W. Kristof and B. Wingersky. A generalization of the orthogonal procrustes rotation procedure to more than two matrices. In *Proceedings of the Annual Convention of the American Psychological Association*. American Psychological Association, 1971.

[93] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[94] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 2:83 – 97, 1955.

[95] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[96] R. Larsen. Functional 2d procrustes shape analysis. In *Scandinavian Conference on Image Analysis*, pages 205–213. Springer, 2005.

[97] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[98] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the International Conference on Computer Vision*, pages 1482 – 1489, 2005.

[99] H. Lev-Ari. Efficient solution of linear matrix equations with application to multistatic antenna array processing. *Communications in Information & Systems*, 05(1):123–130, 2005.

[100] H. Li, T. Huang, C. W. Kong, H. L. Guo, A. Baldwin, N. Chan, and J. Wong. Integrating design and construction through virtual prototyping. *Automation in construction*, 17(8):915–922, 2008.

[101] L. Li, J. Yao, X. Lu, J. Tu, and J. Shan. Optimal seamline detection for multiple image mosaicking via graph cuts. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:1–16, 2016.

[102] W.-Y. D. Lin, M.-M. Cheng, J. Lu, H. Yang, M. N. Do, and P. Torr. Bilateral functions for global motion modeling. In *Proceedings of the European Conference on Computer Vision*, pages 341–356. Springer International Publishing, 2014.

[103] Y. Lipman, S. Yagev, R. Poranne, D. W. Jacobs, and R. Basri. Feature matching with bounded distortion. *ACM Transactions on Graphics*, 33(3):26:1–26:14, 2014.

[104] R. W. Lissitz, P. H. Schönemann, and J. C. Lingoes. A solution to the weighted procrustes problem in which the transformation is in agreement with the loss function. *Psychometrika*, 41(4):547–550, 1976.

[105] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[106] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[107] F. T. Luk. Oblique procrustes rotations in factor analysis. *SIAM journal on scientific and statistical computing*, 5(4):764–770, 1984.

[108] T. C. Lukins and E. Trucco. Towards automated visual assessment of progress in construction projects. In *BMVC*, pages 1–10, 2007.

[109] J. Ma, W. Qiu, J. Zhao, Y. Ma, A. L. Yuille, and Z. Tu. Robust L_2E estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing*, 63(5):1115 – 1129, 2015.

[110] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[111] V. Mahboub. On weighted total least-squares for geodetic transformations. *Journal of Geodesy*, 86(5):359–367, 2012.

[112] C. Mallet and F. Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of photogrammetry and remote sensing*, 64(1):1–16, 2009.

[113] C. Mallet, F. Bretar, M. Roux, U. Soergel, and C. Heipke. Relevance assessment of full-waveform lidar data for urban area classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S71–S84, 2011.

[114] C. Mallet, F. Bretar, and U. Soergel. Analysis of full-waveform lidar data for classification of urban areas. *Photogrammetrie Fernerkundung Geoinformation*, 5:337–349, 2008.

[115] K. V. Mardia, R. Edwards, and M. L. Puri. Analysis of central place theory. *Bulletin of the International Statistical Institute*, 47(2):93–110, 1977.

[116] R. Marzotto, A. Fusiello, and V. Murino. High resolution video mosaicing with global alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 692–698. IEEE Computer Society, 27 June - 2 July 2004.

[117] E. Maset, R. Carniel, and F. Crosilla. Unsupervised classification of raw full-waveform airborne lidar data by self organizing maps. In *International Conference on Image Analysis and Processing*, pages 62–72. Springer, 2015.

[118] C. I. Mosier. Determining a simple structure when loadings for certain tests are known. *Psychometrika*, 4(2):149–162, 1939.

[119] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[120] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[121] A. L. Neuenschwander, L. A. Magruder, and M. Tyler. Landcover classification of small-footprint, full-waveform lidar data. *Journal of applied remote sensing*, 3(1):033544, 2009.

[122] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas. An optimization approach to improving collections of shape maps. In *Eurographics Symposium on Geometry Processing*, volume 30, pages 1481–1491, 2011.

[123] M. Oliveira, A. D. Sappa, and V. Santos. Unsupervised local color correction for coarsely registered images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 201–208. IEEE, 2011.

[124] D. Pachauri, R. Kondor, and V. Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems 26*, pages 1860–1868. 2013.

[125] J. Pan, M. Wang, D. Li, and J. Li. Automatic generation of seamline network using area voronoi diagrams with overlap. *IEEE Transactions on Geoscience and Remote Sensing*, 47(6):1737–1744, 2009.

[126] E. M. Phelps. A critique of the principle of the horizontal plane of the skull. *American Journal of Physical Anthropology*, 17(1):71–98, 1932.

[127] J. Reitberger, P. Krzystek, and U. Stilla. Benefit of airborne full waveform lidar for 3d segmentation and classification of single trees. In *ASPRS 2009 Annual Conference*, pages 1–9, 2009.

[128] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint probabilistic matching using m-best solutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 136–145, 2016.

[129] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[130] B. Schaffrin. A note on constrained total least-squares estimation. *Linear Alg. Appl.*, 417:245–258, 2006.

[131] B. Schaffrin and Y. Felus. On the multivariate total leastsquares approach to empirical coordinate transformations. three algorithms. *Journal of Geodesy*, 82(6):415–421, 2008.

[132] B. Schaffrin and Y. Felus. An algorithmic approach to the total least-squares problem with linear and quadratic constraints. *Studia Geophysica et Geodaetica*, 53(1):1–16, 2009.

[133] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[134] P. Schönemann and R. Carroll. Fitting one matrix to another under choice of a central dilation and a rigid motion. *Psychometrika*, 35(2):245–255, 1970.

[135] P. Schroeder, A. Bartoli, P. Georgel, and N. Navab. Closed-form solutions to multiple-view homography estimation. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 650–657, 2011.

[136] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of London B*, volume 244, pages 21–26, 1991.

[137] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006.

[138] A. Seth, J. M. Vance, and J. H. Oliver. Virtual reality for assembly methods prototyping: a review. *Virtual reality*, 15(1):5–20, 2011.

[139] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30(1):20 – 36, 2011.

[140] C. G. Small. *The statistical theory of shape*. Springer Science & Business Media, 2012.

[141] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[142] C. Strecha, W. von Hansen, L. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[143] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[144] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[145] R. Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, 2006.

[146] Y.-W. Tai, J. Jia, and C.-K. Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 747–754. IEEE, 2005.

[147] J. M. F. ten Berge. Orthogonal procrustes rotation for two or more matrices. *Psychometrika*, 42(2):267–276, 1977.

[148] J. M. F. ten Berge. The rigid orthogonal procrustes rotation problem. *Psychometrika*, 71(1):201–205, 2006.

[149] J. M. F. ten Berge, H. A. L. Kiers, and J. J. F. Commandeur. Orthogonal procrustes rotation for matrices with missing values. *British Journal of Mathematical and Statistical Psychology*, 46(1):119–134, 1993.

[150] J. M. F. ten Berge and D. L. Knol. Orthogonal rotations to maximal agreement for two or more matrices of different column orders. *Psychometrika*, 49(1):49–55, 1984.

[151] R. Toldo, A. Beinat, and F. Crosilla. Global registration of multiple point clouds embedding the generalized procrustes analysis into an ICP framework. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 109–122, 2010.

[152] M. Uyttendaele, A. Eden, and R. Skeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 509–516, 2001.

[153] W. Wagner, M. Hollaus, C. Briese, and V. Ducic. 3d vegetation mapping using small-footprint full-waveform airborne laser scanners. *International Journal of Remote Sensing*, 29(5):1433–1452, 2008.

[154] W. Wagner, A. Ullrich, V. Ducic, T. Melzer, and N. Studnicka. Gaussian decomposition and calibration of a novel small-footprint full-waveform digitising airborne laser scanner. *ISPRS journal of Photogrammetry and Remote Sensing*, 60(2):100–112, 2006.

[155] W. Wagner, A. Ullrich, T. Melzer, C. Briese, and K. Kraus. From single-pulse to full-waveform airborne laser scanners: potential and practical challenges. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35:201–206, 2004.

[156] G. Wahba. A Least Squares Estimate of Satellite Attitude. *SIAM Review*, 7(3), 1965.

[157] C. Wang and S. Mahadevan. Manifold alignment using procrustes analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1120–1127. ACM, 2008.

[158] H. Wang and C. Glennie. Fusion of waveform lidar data and hyperspectral imagery for land cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:1–11, 2015.

[159] W. Xu and J. Mulligan. Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 263–270. IEEE, 2010.

[160] Z. Yang, W. Jiang, B. Xu, Q. Zhu, S. Jiang, and W. Huang. A convolutional neural network-based 3d semantic labeling method for als point clouds. *Remote Sensing*, 9(9):936, 2017.

[161] A. Yu, Z. Liu, G. Duan, J. Tan, L. Che, and X. Chen. Geometric design model and object scanning mode based virtual assembly and repair analysis. *Procedia CIRP*, 44:144–150, 2016.

[162] J.-G. Yu, G.-S. Xia, A. Samal, and J. Tian. Globally consistent correspondence of multiple feature sets using proximal Gauss–Seidel relaxation. *Pattern Recognition*, 51:255 – 267, 2016.

[163] J. C. Yuan. The control of fabrication and pre-assembly's dimension of tubular chimney tower. *Progress in Steel Building Structures*, 3:56–63, 2011.

[164] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1426 – 1433, 2010.

[165] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

[166] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the International Conference on Computer Vision*, pages 4032 – 4040, 2015.