Non-Rigid Registration of Visual Objects

Ph.D. Thesis

by

Zsolt Sánta

Supervisor:
Dr. Zoltan Kato

Doctoral School of Computer Science

Institute of Informatics

University of Szeged

Szeged

2018

Contents

Contents i

List of Figures iii

List of Tables vii

List of Algorithms ix

Notation and Abbreviation xi

Competetive Methods xi

Acknowledgment xiii

1 Introduction 1

2 Fundamentals 3

2.1 Image Registration Methods 3

2.1.1 State of the Art 4

2.1.2 Deformable Registration of Triangular Surface Meshes 9

2.2 Camera Network Calibration 10

2.2.1 Relative Pose Estimation 12

2.2.2 Register the Cameras into a Common Frame 13

2.2.3 Bundle Adjustment 14

3 Deformable Registration of 3D Objects 17

3.1 Problem Statement 17

3.2 Modeling the Deformation 21

3.2.1 Polynomial Model 21

3.2.2 Thin Plate Splines 21

3.3 Efficient Computation of Integrals over 3D Objects 23

3.3.1 Voxel Representation 24

3.3.2 Triangular Surface Mesh Representation 26

3.4 Numerical Implementation 33

3.4.1 Voxel Representation 34

3.4.2 Triangular Surface Mesh Based Algorithm 34

3.5 Experimental Results 36

3.5.1 Synthetic Tests on Volumetric Data 36

3.5.2 Synthetic Tests on Surface Data 42

3.5.3 Robustness Against Noise 43

3.5.4 Medical Applications . . . 45
3.5.5 3D Face Alignment . . . 48
3.6 Summary . . . 51

4 Robust Registration of 2D Images **55**
4.1 Registration Framework . . . 55
4.2 Affine Alignment of Occluded Shapes . . . 56
4.2.1 Calculating the Integrals Efficiently . . . 58
4.2.2 Numerical Implementation . . . 60
4.2.3 Experimental Results . . . 62
4.3 Non-rigid Registration of Covariant Functions . . . 64
4.3.1 Numerical Implementation . . . 67
4.3.2 Experimental Results . . . 69
4.4 Conclusion . . . 75

5 Ad-hoc Mobile Camera Network Calibration **77**
5.1 The Calibration Framework . . . 78
5.1.1 Relative Pose of the Cameras Within the Network . . . 79
5.1.2 Localizing the Camera Network in the 3D Scene . . . 80
5.1.3 Registering the Camera Network with the Extracted Plane . . . 82
5.2 Experimental Results . . . 84
5.2.1 Results on Real Data . . . 87
5.3 Conclusion and Future Work . . . 88

6 Conclusions **91**

Appendix A Summary in English **93**
A.1 Key Points of the Thesis . . . 93

Appendix B Summary in Hungarian **97**
B.1. Az eredmények tézisszerű összefoglalása . . . 97

Publications **101**

Bibliography **103**

List of Figures

3.1 The difference between surface (a) and volumetric (b) integration over the triangular surface mesh of a torus. In (a), the integrals are computed over each triangle, while in (b) the integration domains are tetrahedrons composed of the triangles and an arbitrary chosen common point. In the latter case, the orientation of the triangles is important, because it defines the sign of the volume of the corresponding tetrahedrons. As an example, the blue tetrahedron has positive, and the red has negative volume. 32

3.2 Sample results from the polynomial synthetic dataset. The first two columns show the *templates* and the *observations*, respectively, and in the last column, we present the overlapping voxels of the *registered* objects in yellow and the non-overlapping ones in red and green. 37

3.3 Sample results from the synthetic TPS database. The first two columns show the *templates* and the *observations*, respectively, and in the other columns, we present the overlapping voxels of the *registered* objects in yellow and the non-overlapping ones in red and green. For GMMREG [5] and CPD [6], we present the best results for each particular test case. 38

3.4 Comparison of registration error and computing time with various r values for the Delaunay sphere in surface mesh extraction. 39

3.5 Results on the volumetric synthetic dataset. On these plots we show the median of the registration results with respect to the δ-measure from Equation (3.72) and the running times for each subset categorized by the number of control points used for the generation. The rows contain results obtained by `trisurf_vol` and by `trisurf_surf`, respectively. For each row, the first two columns present the results using the grid control point placement strategy, while the results of the second two columns are obtained by using the surface sampling based strategy (see Section 3.2.2 for more details). 39

3.6 Comparison between the two surface based approaches on the volumetric dataset. The `trisurf_surf` algorithm achieves slightly better results on the alignment accuracy, but the `trisurf_surf-approx` approach has much lower runtime. 41

3.7 Comparison of the best runs with the surface based and volumetric approaches with GMMREG [5] and CPD [6]. The number at end of each plot's name denotes the corresponding r value used for triangular surface generation. . . 41

3.8 Some examples from the artificially cropped dataset and the solution given by the surface-based approach. The first three columns show the *template*, the *observation*, and the transformed *template*, respectively. In the fourth column, each point of the *observation* colored based on the distance from the closest point of the transformed *template*. Note that, the values are expressed in voxels. 42

3.9 Synthetic results on the open surface dataset using the approximate surface integration approach. The control points have been placed using the *Farthest Point Sampling* strategy on the *template* surfaces. In the diagram, the D_{area} metric (y-axis) plotted against the D_{RMS} metric (x-axis). The areas denoted by the yellow and gray rectangles meaning the visually good and the visually acceptable results, respectively. From the 750 test cases, 66% are visually good and 83.33% are visually acceptable solutions. 43

3.10 Robustness test results for various degree of synthetically generated surface errors. For each test, samples of surface errors on the same voxel slice are shown. 44

3.11 The alignment to an artificially cropped observation. The solution is on the boundary of visual acceptability if considering the cropped observation, although it is very far from the original one. 45

3.12 The quantitative results in the presence of surface errors and overfitting. On the bottom diagram, the overfitting problem has been partially handled by minimizing the bending energy of the TPS model. 45

3.13 Alignment of lung CT volumes. For each block, we present the *template*, the *observation*, the overlaid result image and a sample slice combining the grayscale values of the original and the transformed image as a checkerboard pattern, respectively. Segmented 3D lung images were generated by the *InterView Fusion* software of Mediso Ltd. 46

3.14 Two results of the brain surface alignment tests from different resolution groups having various qualities. 47

3.15 An example for the membership functions. The green area denotes the inner parts having membership value of 1, while the yellow and red regions denoting the areas between λ_1 and λ_2, and the areas above λ_2, respectively. 49

3.16 The obtained alignment accuracy in the first experiment estimated by D_{GT}. Each plot shows results with different weight functions. 50

3.17 Example alignment on the Bosphorus dataset. The $D_{RMS} = 1.17mm$ and the $D_{GT} = 5.4mm$. 50

3.18 Alignment errors and running times for each resolution of $r \in \{2, 3, 5, 10\}$. The alignment errors are determined as the Euclidean distance of the transformed and the ground truth landmark locations. The first line corresponds to the `step`, the second line to the `linear` interpolation method. 52

3.19 Sample results on the Bosphorus dataset. In each row, we show the *template*, the *observation*, the warped *template* and the landmark locations, respectively. In the last column, green denotes the ground truth position and red is the estimated location. While the proposed approach achieved good results near nose and mouth areas, the highest errors are near the eyebrows. 53

3.20 Comparison between the proposed, the GMMREG [5] and the CPD [6] algorithms. On the left diagram, the D_{GT} measure is presented, while on the right we show the running times of each algorithm. The GMMREG slightly outperformed the proposed approach and the CPD algorithm in terms of alignment accuracy, but the proposed approach has the best running time. 53

4.1 Relation between the *template* and the occluded *observation*. 57
4.2 The ambiguity of the proposed system of equations. Each image shows the transformed observation and the occlusion denoted by black and red colors, respectively. 58
4.3 Examples from the synthetic dataset. In the last three columns, we denote the template shape with black and the alignment errors with gray colors. 63
4.4 The quantitative results of 3000 synthetic test cases, comparing the proposed approach to CPD [6] and AICPBD [44] algorithms. The diagrams in the first row show the performance measured by δ and in the second row by the ϵ metrics. Each column is showing results on the same amount of synthetically generated occlusion ($0\%, 10\%, 20\%, 30\%$, respectively). 64
4.5 Results on real images. The first two columns contain the *template* and the *observation* images, respectively with the contours of the segmented shapes. In the last column, the results are visualized, where the contour of the *template* shape is denoted by red, the contour of the transformed *observation* is denoted by green and the intersection is denoted by yellow colors. 65
4.6 The ambiguity of shape-based registration. In this example, two different deformable transformations applied to the same template give very different observations, but they have the same foreground region after segmentation. . 66
4.7 Two covariant intensity functions defined by green and blue quadrilaterals. The mapping between the regions is denoted by red. 66
4.8 A comparison between the proposed polynomial affine method and the linear model presented in [130]. The rows correspond to the `affine_nearest` and the `affine_bicubic` datasets, respectively. The first two column show the NCC and the ϵ metrics, respectively, and in the third, we compare the running times. 70
4.9 Robustness test results using affine transformation. The first two rows contain plots of NCC and RMS metrics, while the last row show some examples from the dataset. 71
4.10 Sample images from the synthetic datasets and comparison between DROP [11], SPECDEM [12] and the proposed approach. For each row, the first two columns show the *template* and *observation*, respectively. In subsequent columns, the results of DROP, SPECDEM, and the proposed algorithms are presented as a combined RGB image: the red channel contains the *observation* and the green channel contains the transformed *template*. . . . 72
4.11 The results of the control point placement test using 25, 49 and 100 points placed on a uniform grid. The first row contains the results on `Set_15` and the second for `Set_25`. The columns show plots of the NCC, RMS, and runtime (in seconds) statistics. 73
4.12 Comparative test results. Columns show plots of NCC, RMS, and runtime (in seconds) statistics. 74

4.13 Robustness test results using Thin Plate Splines (TPS). The columns show plots for the NCC and the RMS metrics, respectively. 74
4.14 Samples from the real experiments. The first two rows contain images of shirts with textured patches. Each block contains three images: *template*, *observation* and an overlay of the transformed *template* over the *observation*. The images have been acquired with an ordinary camera. The last row shows samples from the bending paper dataset obtained from [9]. The first column contains the *template*, then we show two different stages of the bending with the *observations* and the overlay images. 74

5.1 The proposed pipeline for camera network calibration. 78
5.2 Illustration of the scale estimation. 83
5.3 Sample low-rank patterns from the database. 85
5.4 Images from the synthetic tests. For each row, the first five pictures show the images of the low-rank pattern as seen by the random cameras, while the last one contains the overlayed reprojected frames using the estimated camera matrices. 85
5.5 Quantitative results of rotation angle estimation. 86
5.6 The camera position error and reprojection error of the centroid of the *low-rank* pattern. 86
5.7 Stability test with varying *main cameras*. Each plot shows the average of absolute distances from the medians of each configurations. 87
5.8 Results on real images. For every test case, the last image shows the overlayed back projected camera images using the estimated camera matrices. The low-rank patterns used for calibration are marked on the main camera image as well as on the result. 88

List of Tables

3.1 Comparison between `voxel_poly` and `voxel` on the polynomial dataset. While maintaining the same accuracy, `voxel_poly` provided results are 10 times faster than `voxel`. 38

3.2 Results on 750 synthetic images using 64 control points for each configuration (m – median, μ – mean and σ – standard deviation). 40

3.3 Comparison of the surface based algorithm with the GMMREG and the CPD methods on the open surface dataset (m – median, μ – mean and σ – standard deviation). We used the approximate computation scheme with the surface sampling control point placement strategy of 64 points. The proposed approach outperformed the other methods in the terms of D_{RMS} and D_{area} metrics and achieved better running times. Note that, for these experiments, we used the same triangular surface meshes generated using the $r = 5$ configuration. 44

3.4 The results on brain surface alignment (m – median, μ – mean, σ – standard deviation). The input data has been classified into three groups based on the physical resolution of each image. 47

3.5 A comparison of different mesh resolutions and the input point cloud [118]. We show the number of points for the point cloud and the number of vertices and triangles, respectively, for the triangular surface meshes. 51

4.1 Results of the first experiment using the same numbers and locations for the control points as for the generation. Quantitative evaluation is given in terms of NCC and RMS metrics (μ – mean, m – median, σ – standard deviation) . 71

5.1 Runtime statistics on the synthetic images (m – median, μ – mean and σ – standard deviation). 86

5.2 The average gain of solving the homography scale estimation for all cameras (m – median, μ – mean and σ – standard deviation). 87

A.1 The connection between the thesis points and publications. 95

B.1. A tézispontokhoz kapcsolódó publikációk. 99

List of Algorithms

3.1 Pseudo code of the voxel-based algorithm 35
3.2 Pseudo code of the triangular surface based algorithm 35
4.1 Pseudo code of the affine registration algorithm for occluded shapes 62
4.2 Pseudo code of the 2D registration algorithm using covariant functions 69
5.1 Pseudo code of the calibration framework. 84

Notation and Abbreviation

N	number of parameters
$\mathbb{P}^n$	n-dimensional Projective Space
$\mathbb{R}^n$	n-dimensional Euclidean Space
$\mathbf{x}$, $\mathbf{X}$	Point of the n-dimensional Euclidean Space: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$
$\bar{\mathbf{x}}$, $\bar{\mathbf{X}}$	Point of the n-dimensional Projective Space: $\bar{\mathbf{x}} = [x_1, x_2, \ldots, x_n, w]^T \in \mathbb{P}^n$
2D	two-dimensional
3D	three-dimensional
DoF	Degrees of Freedom
ICP	Iterative Closest Points
ROI	Region of Interest
SSD	Sum of Squared Differences
TILT	Transform Invariant Low-rank Texture
TPS	Thin Plate Splines

Competetive Methods

AICPBD	Bidirectional Affine ICP [44]
CPD	Coherent Point Drift [6]
DROP	Deformable Image Registration using Discrete Optimization [11]
GMMREG	Robust Point Set Registration Using Gaussian Mixture Models [5]
SPECDEM	Spectral Demon's Algorithm [12]

Acknowledgments

I would like to address the first of my acknowledgments to my advisor, Dr. Zoltán Kató. I always could rely on his guidance and useful advices. I hope that, during the work we did together we both gained valuable experiences for the future.

I would like to thank my wife Zsuzsi for her support and help during my work. I would especially thank her for proof reading the first version of my thesis. I am grateful to my family for their support and for the opportunities they have created for me to achieve my goals. I am also thankful to my colleagues at the Institute of Informatics for their advices and help in times of need.

The research was carried out at the University of Szeged. The work has been partially supported by the Doctoral School of the University of Szeged; the grant CNK80370 of the National Innovation Office (NIH) & the Hungarian Scientific Research Fund (OTKA); the European Union and co-financed by the European Regional Development Fund within the project *TÁMOP-4.2.1/B-09/1/KONV-2010-0005* and by the European Social Fund through project FuturICT.hu (grant no.: *TÁMOP-4.2.2.C-11/1/KONV-2012-0013*). Finally, supported by the NKFI-6 fund through project K-120366; and by the AUF and IFA under the AUF-RO project NETASSIST.

Zsolt Sánta, March 2018.

Chapter 1

Introduction

Computer vision is a field of analyzing and interpreting visual objects by making use of various computational tools. The fundamental aim is to create *automatized* systems for such tasks, to help or, in some cases, replace human interpretation. Historically, first applications were working with two-dimensional (2D) data. Nowadays with the development of new three-dimensional (3D) acquisition techniques, like Lidar, Microsoft Kinect™ or Intel® RealSense™, processing 3D data becomes more and more important.

An important preparatory step in almost every computer vision process is *image registration*. The main goal of this task is to estimate mappings between different observations of the same scene. The most frequent problem is to match the origins and orientations of the main axes of the underlying coordinate systems by finding a rigid alignment between the input. While in many cases the underlying deformation can be efficiently handled by this simple alignment, there are various applications where we have to deal with larger deformations. An example for such problem is when the difference in the observations are caused by several independent motions, yielding a non-global deformation between them. The state of the art provides efficient methods to cope with these problems, however, there are lots of open questions as well.

The dissertation addresses the author's research results in multiple areas of image registration. These results are providing solutions for estimating parameters of a wide range of transformations consisting linear, perspective and deformable models.

The thesis is structured as follows. In Chapter 2, we give a brief introduction to the fundamental concepts behind the addressed topics. The state of the art is also discussed here. Then, the next three chapters present the contributions of the thesis. In Chapter 3, a general framework is presented to solve deformable registration problem between 3D objects. The parameters of the pursued transformation are directly provided as a solution of a system of non-linear equations. The framework is generalized to different input representations, leading to efficient implementations for each case. The practical applications include 3D face alignment, lung CT, and brain surface registration.

In Chapter 4, robust registration of 2D images are investigated by addressing two interesting problems. The first one deals with how to handle larger segmentation errors (occlusions and disocclusions) in the input data using geometric information only. This problem is a great challenge for area-based methods, working on the whole image domain to estimate a solution. The proposed approach is built on affine transformations, thus it is able to work with camera images taken in less controlled circumstances (*e.g.* processing images of surveillance systems).

The second problem considers an ambiguity issue caused by large physical deformations when using geometric information only. Global shape symmetry is a well-known example for a possible cause of such ambiguity, albeit working with non-linear deformations inherently leads to multiple equivalent solutions at the level of shapes. The presented algorithm is able to work with more information by making use of a pair of *covariant* functions. Herein, we show an example for using simple grayscale intensity functions.

Finally, in Chapter 5, a new approach is presented for calibrating ad-hoc networks composed of a set of smartphones. These phones have become very popular in the last couple of years, with powerful embedded processors, networking capabilities and multiple types of sensors. The algorithm utilizes the distributed architecture of the network by making use of parallel processing. The proposed approach is able to align the camera network to an arbitrary chosen 3D plane, containing a *low-rank* pattern.

Chapter 2

Fundamentals

In this chapter we will introduce the two central topics of the current thesis: *image registration* and *camera network calibration*. The topics are closely related since in several steps of the calibration process we have to work with the outcome of a registration algorithm. Moreover, the whole calibration process could be considered as a registration approach since the fundamental aim behind both topics is to reconstruct a mapping function between the input domains.

2.1 Image Registration Methods

Registration is the process of establishing geometric relationships between two or more images [1–4]. The aim is to express the input data in the same spatial space. It is a fundamental task in many vision-based systems when the input is coming from different data sources, *e.g.* images of the same scene taken with different cameras, times or viewpoints, such as shape alignment [5–7], motion estimation and tracking [8–10] or medical imaging [4, 11–15].

When registering a pair of observations of the same scene, the general map between the coordinate systems will be a vector field $\phi : \mathbb{R}^n \to \mathbb{R}^m$ and the registration process address to recover this mapping from the input data. While in the current section we will assume that $n = m \in \{2, 3\}$, in Section 2.2 we will investigate projections between different dimensions too. In practice only discrete approximations of the input data are available, therefore the deformation field could be reconstructed only approximately as well.

While in several cases it is convenient to approximate the general deformation in a point-wise manner, it is a common practice to reduce the complexity of the registration problem by replacing the general map with a parametric deformation model [2, 4, 16, 17]. In such cases, the number of parameters correlates with the degrees of freedom of the transformation. The applicability of parametric models depends on the particular application. As an example, when we assume linear motion between the observations, it is quite straightforward to use an n-dimensional linear map with $n(n + 1)$ parameters.

Following [16] and [4], the deformation models could be organized into three main groups: *physical model based*, models derived from *interpolation and approximation theories* and *knowledge-based* geometric transformations. In some applications, the transformation is needed to ensure special properties, like inverse consistency, topology preservation or to be a smooth diffeomorphism. These properties can be implemented to most of the models

expressed as part of the regularization [4, 16].

The next major factor in creating the right algorithm for a particular problem is observing the input representation. In 2D the standard representation is a digital image sampled on a regular grid by a simple digital camera or other sensing devices. A cell of an image is called a pixel and it is the smallest measurable unit by the camera with predefined dimensions (based on the parameters of the particular device). Each pixel holds the measured light intensity of the corresponding point of the observed scene. Since this is the smallest unit in this representation, there is no ”gap” between the neighboring elements. The representation could be extended to 3D as well, where a cell is called voxel. In 3D, these images are usually obtained by a kind of *tomographic reconstruction* [18].

After image acquisition, the input is usually preprocessed by multiple low-level techniques like point-wise filtering, image segmentation or building a multi-scale representation by making use of a pyramid technique [19]. Since in the current work we will focus on high-level operations, for most of the proposed algorithms we will assume that the Regions of Interest (ROIs) are already obtained by an image segmentation algorithm. In addition, for shape-based approaches, we will assume the availability of the shape contours or object surfaces as well. Reviewing of the image segmentation approaches is far beyond the scope of the current work and in most of our cases, segmentation is obtained via simple methods like intensity thresholding.

2.1.1 State of the Art

In the last few decades, several good surveys and textbooks have been published on image registration [1, 2, 20], including a comprehensive survey on deformable medical image registration in [4]. Although most of the well-known approaches are focusing on registering two-dimensional (2D) images, recently numerous efficient algorithms for three-dimensional (3D) surface registration and matching have also appeared in the literature [21–25].

Surveys on image registration methods are usually investigating the whole registration pipeline of the observed approaches. The key elements in these pipelines are detecting and matching salient objects of the scenes and estimating the deformation by a particular numerical procedure [2, 4]. While a complete review of the state of the art is well beyond the scope of the current work, we will give a brief introduction structured around the main methodological aspects of the available methods. Similarly, despite the fact that many conceptual and methodological similarities can be found between the image registration and the optical flow methods, we will limit our review to the registration approaches only. A recent survey of optical flow methods could be found in [26].

From a methodological point of view, the registration approaches can be differentiated into two main groups: geometric (or landmark-based) methods and area-based (or iconic) approaches [4]. The fundamental difference between these groups is while the geometric methods are relying on extracted landmarks placed in salient image locations, the area-based methods are using the whole image domain in an aggregated manner to determine the transformation. Therefore geometric methods could be easily made robust against outliers and missing data, but they are challenged by landmark localization and correspondence establishing problems. The accuracy of the landmark extraction has a great influence on the registration accuracy and outliers caused by wrong correspondences could introduce further problems into the estimation.

Iconic approaches are typically relying on the availability of rich radiometric information

which is used to construct a similarity measure based on a kind of intensity correlation. The aligning transformation is then found by maximizing the similarity between the objects, which usually yields a complex non-linear optimization procedure. These methods have higher accuracy compared to the geometric approaches, but since they use the whole image for the registration, thus the computational complexity could be higher as well. Moreover, most of the practical similarity measures will eventually lead to the optimization of a non-convex energy function, therefore the initialization has a great influence on the outcome. Handling occlusions and disocclusions is also an important issue due to the global nature of these methods. Hybrid approaches are also available, combining the best properties of both worlds [4].

Geometric Methods

Now let us elaborate the review by first observing geometric methods. These methods are usually build up from two major steps [4, 20]:

1. detect salient image locations and extract local image properties (features),
2. infer the pursued geometric relationship using the extracted landmarks.

The detection phase of Step 1. could be realized as an outcome of an image segmentation method (like edge or corner detection) or by making use of a robust keypoint detector algorithm [27–32]. While we refer as ”points” to the extracted elements, have to mention that the obtained landmarks are not necessarily built up from one point, but could be edges or regions as well. In practice, however, most of the available methods are working on individual point sets, therefore either a specific point of the landmark (*e.g.* the centroid of the extracted region) is used in the subsequent steps (see the experiments from [32] for example) or the extracted landmarks considered as an unstructured set of points.

Over the years the problem of point detection and description have been studied intensely by the computer vision community developing successful methods like SIFT [27], SURF [28], ASIFT [29] or KAZE [30, 31]. These methods are built upon the results of scale space analysis. The main idea behind these approaches is to create a multiscale representation from the input images in order to detect and describe the landmarks on different scale levels, hence achieving scale invariance. Traditionally Gaussian scale space have been used for such tasks obtained by applying multiple Laplacian of Gaussian (LoG) operators to the images. However, this procedure is impractical because its high computational complexity. In order to overcome this limitation, Lowe’s SIFT [27] is using Difference of Gaussian operators to approximate the LoG and the landmarks are obtained as local extrema in this representation. This approximation reduced the running time to an acceptable level. Further enhancements have been achieved by the Speeded-Up Robust Features (SURF) [28] through approximating the derivatives of the Gaussian by simple box filters. Morel and Yu [29] proposed a fully affine invariant framework for SIFT-like features by simulating the longitudinal and latitudinal angles of the camera as well. Using these modifications enabled the extracted features to become fully affine invariant, at the expense of higher computational cost.

While the Gaussian scale space has some nice properties like its straightforward computation and robustness against Gaussian noise, the Gaussian blurring does not respect the natural boundaries of the objects. This, unfortunately, reduces localization accuracy and distinctiveness of the features. In order to overcome this drawback, Alcantarilla *et al.* propose to use a nonlinear scale space by means of nonlinear diffusion filtering [30, 31].

Once we have the detected landmarks, we can advance to the next step to infer the pursued geometric relationship. In landmark-based methods, this is usually done in two phases [4]. First, a sparse correspondence is established by matching the descriptors of the extracted landmarks. Then, if it is needed by the particular application, a dense displacement is inferred using the sparse correspondences. This is achieved by fitting a parametric model to extracted points.

The landmark matching is preceded by the process of extracting image or shape properties from the detected keypoints. From the current point of view, the properties are either *invariant* or *covariant* with respect to the physical deformation. An invariant property does not change due to the physical deformation, therefore they used to obtain correspondences between the different observations of the landmarks. A covariant property, however, will change accordingly to the deformation, thus they are able to measure the effect of the deformation and eventually estimate the spatial transformation. While each of the surveyed detectors has its own invariant feature descriptor (*i.e.* a multidimensional vector containing invariant properties), the covariant properties are usually limited to the spatial coordinates of the landmarks. Therefore, current methods are either based on matching extracted invariant descriptors or minimizing spatial distance of the landmarks.

A detailed review on descriptor matching techniques is out of the scope of the current work, for more details refer to [20]. Briefly, the main idea behind the match is to compare the extracted feature vectors by making use of a distance function (in most cases the Euclidean distance is a good choice). Then, we can assign one-by-one correspondences by taking the closest pairs of points. In order to reduce the possible ambiguities, it is a good practice to assign a rank to each matching by observing the distance ratio between the closest and second closest point in the feature space and reject the matching if the rank is below of an arbitrary threshold.

Once the correspondences are available, the parameters of the transformation can be estimated using the landmark locations [20, 33, 34]. The model specific estimators are constructed to maximize the robustness against various types of noise and inaccuracy. The most frequently used family of estimators is based on the least squares approach, thus it aims to find a solution which minimizes the Sum of Squared Differences (SSD) between the transformed coordinates of the source point set and the target point set. These estimators are intended to be robust against Gaussian noise, but they are ineffective in the presence of outliers, *i.e.* bad correspondences or points deforming differently from the actual physical deformation [20]. M-estimators are another example where the SSD function is replaced by an arbitrary function, which increases less rapidly with increasing residuals [20].

Another common way to deal with outliers is to use the RANSAC algorithm jointly with the model fitting method [35]. Herein, we randomly sample the minimal number of point correspondences to fit the model. Then using the acquired transformation, a model specific distance function is evaluated in all point correspondences. The current model's inlier set is established by thresholding the distances for each pair of correspondences. Eventually, the algorithm will choose the biggest inlier set as its result.

Now, let us observe methods using invariant feature descriptors. SIFT descriptors are known to be invariant against rotation and scaling, which is achieved by first estimating the scale and the main orientation of each keypoint. Then using a rotated subregion around the landmark's position, a 3D histogram is created from the image gradients [27]. SURF uses similar oriented regions around each landmark locations, but the statistics obtained as a summation of Haar wavelet responses [28]. These descriptors are usually composed of

128 elements, but it also depends on the parametrization of the algorithms (*e.g.* the size of the subregions). The first version of KAZE uses SURF descriptors computed in its nonlinear scale space [30], while the Accelerated KAZE estimates modified Local Difference Binary (LDB) [36] features based on simple binary tests on intensity and gradient data [30].

A recent method for utilizing invariant descriptors have been presented in [37]. This robust estimation based approach iteratively determines the parameters of a transformation from a reproducing kernel Hilbert space (RKHS) and gives the inlier set of correspondences as well. The estimation made by a novel algorithm based on the L_2E formulation [38]. For each iteration, the initial point correspondences are established by making use of extracted SIFT landmarks in 2D and on MeshDOG/MeshHOG keypoints [39] in 3D.

Let us continue our survey with the spatial methods, *i.e.* methods based on covariant properties only. The observed methods are either using an alternating scheme to simultaneously determine the correspondences and the transformation [40, 41] or extract the parameters directly [5, 42].

The most common algorithm to find the best geometric transformation between two point sets is the Iterative Closest Points (ICP) algorithm [40, 43]. This approach is based on the idea of assigning the correspondences of the source point set by choosing the closest point (w.r.t. the Euclidean distance) from the target set in each iteration. The process aims to minimize the SSD function between the corresponding points. The classical ICP algorithm is used to estimate rigid-body transformations only, but through the years several ICP variants have been proposed for transformations with higher Degrees of Freedom (DoF) [44]. Moreover, there are methods introducing context specific enhancements as well [45, 46].

The Bidirectional Affine ICP (AICPBD) [44] extends the ICP idea by introducing a bidirectional distance based least-square problem for solving the affine registration between point sets. The original ICP method is inherently ill-posed for affine registration since in extreme cases an affine transformation is able to transform the whole source point set into one point of the target set causing minimal SSD value. The bidirectional distance, however, is able to handle this problem by estimating the closest points from the target point set to the source set as well using the inverse transformation.

An early result of robust point matching introduced in [41], where the proposed algorithm alternates between estimating the correspondences with a *soft assign* method and computing the underlying deformation by making use of an arbitrary deformation model. Since the authors used Thin Plate Splines (TPS) for their experiments, this algorithm is often referred as TPS-RPM by the scientific community.

Representing point sets using Gaussian mixture models is another convenient way to solve the registration problem between point sets. In [6, 42], a probabilistic model is proposed where a Gaussian mixture with centroids corresponding to the input point set is fit to the target set by maximizing the likelihood. Thus, an energy function, composed of the negative log-likelihood and an additional regularization term, is minimized using the Expectation-Maximization (EM) algorithm [47]. The transformation can be modeled by an arbitrary parametric model, like affine or rigid-body [6] in linear and a Gaussian kernel based radial basis function (RBF) model in non-rigid cases [42]. In [5], both point sets are represented by Gaussian Mixture Models and then the L_2 distance of the two mixtures is minimized. The authors use a closed-form expression to calculate the distance between the Gaussian mixtures efficiently. The underlying deformation is modeled using TPS. Both approaches are reported to be robust against high occlusions, however, they are inefficient for large point sets, due to their computational cost.

Area-based methods

In this section, we will continue the review with the area-based (or iconic) methods. Herein, the main aim is to establish a dense spatial connection between the input images by making use of the whole domains, while maximizing a similarity or minimizing a dissimilarity criterion [2, 4]. One of the main problems in this area is how to choose the similarity criterion with respect to the application. A good criterion assigns higher similarity values when we are comparing the same points of the scenes, which is particularly difficult in the presence of higher radiometric distortions (*e.g.* multi-modal image registration). Most of the classical methods are built upon *cross-correlation (CC)* or its modifications, while the *mutual information (MI)* based techniques are dominating for multi-modal registration [2, 4].

One of the most commonly used methods for deformable registration of medical images is the Demon's Algorithm [12, 48–50]. The original method is inspired by Maxwell's Demons and it is used to solve the registration problem as a diffusion process [48]. In this approach, the deformation field is approximated point-wise and optimized with respect to a similarity criterion. The solution is obtained via iteratively updating the deformation by searching for better correspondences near the current solution.

In [49], the authors proposed a Gradient Descent based interpretation of the Demon's Algorithm for registering deformable 3D medical images. They express the original method as an approximation of a second order gradient descent on the SSD of the intensity values. Although the experiments confirmed that the method gives fast and accurate results, the authors noted the importance of the proper regularization and the drawbacks of the SSD criteria in the case of higher radiometric distortion or multi-modal registration [49].

The main problem with the original Demon's Algorithm is that the search for a better solution is limited to a small area around the current result, therefore the optimization could stuck in a local optimum. In [12], Lombaert *et al.* extend the Demon's Algorithm to handle large scale non-rigid deformations. This approach aims to establish global correspondences between the images by making use of simple nearest neighbor searches. This so-called Direct Feature Matching is working on densely extracted spectral features based on the eigenvectors of the graph Laplacian of each image, enforcing an intrinsic geometric consistency in the technique [12].

Another volumetric approach has been proposed by Glocker *et al.*, where the registration problem has been formulated as a discrete multi-labelling problem by employing Markov Random Fields (MRFs) [51, 52]. MRF is a popular graph-based model for representing such problems giving tools to describe the behavior of the input entities and the labels. In most cases, the model has unary (also called singleton), pairwise (or doubleton) and higher order potentials giving the relationship between one, two or more entities, respectively. In [53], the deformation field is approximated by a free-form deformation (FFD) model based on cubic B-Splines. The FFD is defined by a grid of control points and the algorithm aims to find a discrete displacement vector of the grid with respect to an energy function. This function is composed of an application specific similarity measure and a smoothness term. These type of approaches are often referred as discrete techniques by the literature, while the methods related to the Demon's Algorithm are called continuous techniques [11].

As we mentioned earlier, one of the main challenges in area based methods is how to deal with occlusion or disocclusions. While there are particular applications where occlusions are not present like industrial inspection or many problems of medical image processing, it is very common when working with images taken in less controlled environments (*e.g.* images

obtained from unknown sources). Within the area based approaches, the issue is often addressed as a segmentation problem, thus it is recommended to be handled accordingly in the segmentation phase [7]. Based on the same assumption, another common way is formalizing the problem as a simultaneous segmentation–registration process and handle both the alignment and the occlusion handling within the same framework [54–56]. For example, in [54] the authors proposed an active contour based approach, where the aim is to segment the curves of an object on the source and the target images. However, unlike to the traditional active contour approaches, the contours are assumed to be related by an unknown parametric transformation, thus the curve evolution is coupled with the parameter estimation. Another interesting MRF-based approach has been proposed in [56].

A related, well-studied problem is template matching, where the goal is to localize a small patch in a much larger observation image. This could be considered as solving the registration problem in the presence of extreme occlusions [57, 58]. In [58] an efficient template matching algorithm has been proposed to handle arbitrary 2D affine transformations. It finds the best solution with respect to the sum of absolute differences (SAD) metric between the image intensities. The space of possible transformations are sampled according to the smoothness of the images and for each possible transformation, the SAD metric is approximated in a small number of random points (this type of algorithm is referred as sublinear by the literature) [58].

2.1.2 Deformable Registration of Triangular Surface Meshes

Unlike to the 2D cases, in 3D cases we have several types of input representation, therefore the range of applicable methods is much wider. Today, the three main object representations used mainly in the literature are volumetric (voxel) images, unstructured point sets and triangular surface meshes [21]. Most of the already addressed methods are capable of handling volumetric images or unstructured point sets. Now, let us focus on registering triangular surfaces.

Triangular surface meshes are the most used general 3D surface representations. The main advantage over point sets is that each triangle defines a piecewise planar region between the neighboring points. With this representation, we can consider the input as volumetric objects enclosed by surface meshes (*i.e.* objects with non-zero volumes) and also surface like objects defined as a set of triangles (*e.g.* a 3D face scan). In this work, we will refer to the triangular surfaces of volumetric objects as closed surfaces and the non-closed ones as open surfaces. The closed surfaces given by a more strict topological structure without surface holes or missing triangles. Moreover, the objects must have consistently oriented triangles, thus they separate the 3D space into interior and exterior spaces giving infinitely many additional points. Triangular surface meshes are typically produced by stereo reconstruction methods [59, 60], range imaging [61] or by laser scanner devices [62], but they can be easily extracted from voxel images [63] as well. In most cases, triangular surfaces contain spatial information only, but it is possible to involve textural information as well.

Triangular surfaces can be registered by either using the vertices of the meshes only, by applying any point set registration approach [5, 6, 41] to them. Another way is to utilize the structure information of triangles as well [37, 64, 65]. Modeling the deformation is done similarly to the previous cases by using interpolating splines or element-wise (vertices or faces) correspondences.

Graph matching [64, 66, 67] is a popular tool for encoding structural constraints to

the registration approaches, moreover the fundamental structure of triangular surfaces also implying the applicability of these methods. The idea behind these algorithms are related to the Markov Random Field approaches by modeling the problem as weighted sums of interaction functions, but there is a major difference in the interpretation of the energy function. While MRF methods are built upon tools from the probability theory, the graph-based approaches work with general interaction functions and the optimal solution is usually found by simple algorithms from combinatorial optimization.

A graph matching method has been proposed in [64] to achieve dense, vertex-wise correspondences between the input meshes. The constructed graph model is built upon the space of all pair of vertex correspondences, assuming that the *unary* potential based on one pair, the higher order potentials based on three pairs, yielding triangle-wise correspondences. The *unary* potential is constructed using the differences between the Gaussian curvatures and texture information of each pair of vertices. The higher order potentials are given as a sum of a Möbius transform and a Gaussian map based function. The algorithm determines correspondences in two stages. During the first stage, the method estimates a sparse correspondence between the input meshes, then using this result as an input, a dense correspondence is extracted.

A similar approach based on the MRF model has been presented in [65]. The singleton potentials are modeled as differences between extracted local features and the higher order potentials defined as constraints on the deformation using a novel model called Canonical Distortion Coefficient (CDC). The CDC describes the deformations for each pair of triangles as distortions along the two principal directions in a canonical parametrization domain. The potential function gives low values if the CDC of each examined triangle pair is coming from the range of possible deformations (given as prior information to the algorithm).

Finally, there are several recent surface matching algorithms coming from spectral shape analysis field. The main goal of these approaches is to define an intrinsic surface representation that is invariant to the non-rigid motion [22, 24, 68–71]. Using these models, the surfaces are embedded into a new feature space, where due to the invariance the solution of the correspondence problem becomes much easier to solve. These methods can deal with a large scope of deformations, *e.g.* articulated motion of people or animals.

2.2 Camera Network Calibration

Camera calibration is one of the fundamental problems in computer vision [72]. Herein, the main goal is to reconstruct the image acquisition function and it is an important preparatory step in several applications like visual odometry [73] and 3D reconstruction [72]. While in a real application many optical settings are available, in the following we will focus on the simple pinhole camera model. This model describes a central projection of 3D points of the scene onto the image plane of the camera. Let us assume that the points of the scene and the image plane are embedded into the corresponding projective spaces, $\mathbb{P}^3$ and $\mathbb{P}^2$, respectively. This can be easily achieved by representing each point as a homogeneous vector. Then, a point $\bar{\mathbf{X}} = [X_1, X_2, X_3, 1]^T \in \mathbb{P}^3$ from the scene and its projection on the image plane $\bar{\mathbf{x}} = [x_1, x_2, w]^T \in \mathbb{P}^2$ related by a linear mapping:

$$\bar{\mathbf{x}} = \mathbf{P}\bar{\mathbf{X}}, \tag{2.1}$$

where $\mathbf{P}$ is a 3×4 homogeneous camera projection matrix [72]. While the problem is linear between the projective coordinates of the points, the non-linearity is introduced by the homogeneous division, which is applied to convert the coordinates back to the Euclidean plane:

$$\bar{\mathbf{x}} \mapsto [x_1/w, x_2/w]^T. \tag{2.2}$$

In general, $\mathbf{P}$ is composed of the product of two matrices:

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R} \mid \mathbf{R}\mathbf{t}\right], \tag{2.3}$$

where

$$\mathbf{K} = \begin{pmatrix} \alpha_1 & s & c_1 \\ & \alpha_2 & c_2 \\ & & 1 \end{pmatrix} \tag{2.4}$$

corresponds to the *intrinsic parameters* and $\mathbf{R} \in \mathbb{R}^{3\times3}$ orthonormal matrix and $\mathbf{t} \in \mathbb{R}^3$ translation vector are the *extrinsic parameters* or the *pose* of the camera. The intrinsic parameters contain camera specific properties, like the focal length and the physical size of the pixels. The pose corresponds to the camera location and orientation in the world coordinate system.

The aim of calibration process is to estimate camera parameters. This could mean reconstruction of the whole $\mathbf{P}$ matrix [72, 74] or estimate either intrinsic or extrinsic parameters alone. The former case is often referred as auto- or self-calibration by the literature [75, 76], while the latter case as camera pose estimation [77, 78]. The parameters are usually estimated from multiple images taken by the same camera. For this, first we extract several point correspondences, then fit a transformation model. However, if none of 3D coordinates of the extracted points are available, the world coordinate system can only be reconstructed up to a similarity transformation [72].

In camera network calibration our aim is to recover the parameters of N (probably different) cameras. In this setting, the image of the point $\bar{\mathbf{X}}$ in the i^{th} camera could be described as

$$\bar{\mathbf{x}}^i = \mathbf{P}_i\bar{\mathbf{X}}, \tag{2.5}$$

where $\bar{\mathbf{x}}^i$ denotes the projection of $\bar{\mathbf{X}}$, $\mathbf{P}_i = \mathbf{K}_i[\mathbf{R}_i \mid \mathbf{R}_i\mathbf{t}_i]$ and $i = 1, \ldots, N$. Since in the current work we are dealing with special mobile cameras having fixed focal lengths, we can assume that the intrinsic parameters are precalculated and the possible radial distortions are eliminated from the images.

When the intrinsic parameters are available, we can obtain the *normalized coordinates* of $\bar{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\bar{\mathbf{x}}. \tag{2.6}$$

This point can be considered as a special projection created using the so-called *normalized camera matrix* [72]:

$$\hat{\mathbf{P}} = \mathbf{K}^{-1}\mathbf{P} = \left[\mathbf{R} \mid \mathbf{R}\mathbf{t}\right], \tag{2.7}$$

as

$$\hat{\mathbf{x}} = \hat{\mathbf{P}}\bar{\mathbf{X}} = \left[\mathbf{R} \mid \mathbf{R}\mathbf{t}\right]\bar{\mathbf{X}}. \tag{2.8}$$

The first step in calibrating a camera network is to determine which images share any common regions or parts. Using this information we can eliminate the independent cameras from the process and decompose the network to smaller subnetworks to work with. The

most frequent tool to handle such structures is the visibility or vision graph [79–81]. Let

$$\mathcal{V} = (V, E) \tag{2.9}$$

be an undirected graph where V and $E \subset V \times V$ denote the vertices and the edges, respectively, and $|V| = N$, thus each element of V corresponds to a camera of the network. The edge set is generated as follows, if any pair of P_i and P_j cameras have a reasonable amount of common areas in their images, then let $(v_i, v_j) \in E$, where $v_i, v_j \in V$ are the corresponding nodes in V for P_i and P_j, respectively. The vision graph can be obtained using sensor information [81] and by matching extracted features as described in Section 2.1.1 [79–81]. A similar structure called skeletal graph have been proposed by [82], which encodes geometric information as well.

Assuming a $\mathcal{V} = (V, E)$ vision graph is available, let us continue with the estimation of the relative pose of the network. Most of the current approaches use 3D information for estimating the relative poses, therefore the 3D points might be available initially or determined simultaneously [73]. In the former approaches, the relative pose is extracted using 3D-2D point correspondences by solving the *Perspective-n-point* problem [73, 83, 84], where n denotes the number of point correspondences available. The minimal amount of correspondences needed for this method is three, therefore the fast and exact solution of the P3P problem is particularly interesting. Usually, it is used within the RANSAC framework to be robust against outliers [35].

The latter approaches are often referred as *structure from motion* methods by the literature [72, 77, 85]. Herein, the parameter estimation usually built up from the following three steps [77]:

1. Estimate the relative poses between pairs or triplets of cameras.
2. Register the extracted camera poses into a common coordinate system.
3. Apply a global non-linear optimization procedure to enhance the precision of the estimated results (*e.g.* the bundle adjustment algorithm to minimize the reprojection error [86, 87]).

In the following, we will review some methods published for solving these three problems.

2.2.1 Relative Pose Estimation

There are several applicable methods for solving the first problem surveyed by numerous papers [88, 89]. The aim is to determine the relative pose from each camera to its neighbors in the vision graph. In practice, this is usually handled by extracting the so-called fundamental matrix or when the intrinsic parameters are available the essential matrix. Both these matrices are rank 2 homogeneous matrices, which describes the epipolar geometry between the corresponding pairs of cameras [72]. The fundamental matrix, acting between the images of cameras P_i and P_j, maps each point $\bar{\mathbf{x}}^i$ on the first image to a line (called epipolar line) on the second image ensuring that the corresponding $\bar{\mathbf{x}}^j$ projection will be on the line. Formally, assuming that $\bar{\mathbf{x}}^i \leftrightarrow \bar{\mathbf{x}}^j$ is a pair of corresponding points and $\mathbf{F}_{ij} \in \mathbb{R}^{3\times3}$ is a fundamental matrix between the cameras P_i and P_j, the following holds [72]:

$$\bar{\mathbf{x}}^{jT}\mathbf{F}_{ij}\bar{\mathbf{x}}^i = 0. \tag{2.10}$$

The essential matrix $\mathbf{E}_{ij} \in \mathbb{R}^{3\times3}$ acts similarly between the normalized coordinates of the projections, therefore it can be expressed as [72]

$$\mathbf{E}_{ij} = \mathbf{K}_j^T \mathbf{F}_{ij} \mathbf{K}_i. \tag{2.11}$$

The fundamental matrix is estimated using at least 7 point correspondences, but this approach could result in one or three solutions [72]. Therefore, it is often replaced by the normalized 8 point algorithm, which eliminates the ambiguities at the sake of an additional point correspondence. Both of these approaches can be applied to estimate the essential matrix too, by normalizing the point correspondences with the intrinsic parameters. In practice, however, a version of the 5 point algorithm is applied [78] to reduce the necessary amount of correspondences. Similarly to the 7 point algorithm, there will be more than one solutions, *e.g.* the algorithm published in [78] could return 4.55 solutions on average.[1] The disambiguation is based on cheirality tests (*i.e.* how many points lie in front of the cameras) [90, 91] and the comparison of the first order geometric error (or Sampson-distance) of the solutions [89]. In practice, as we described in the previous section, the point correspondences could be challenged with false matches and other types of noise, which have a strong influence on the outcome. Therefore, the correspondences are usually pre-filtered using the RANSAC algorithm [35] as discussed in Section 2.1.1, and the essential matrix is obtained by solving an overdetermined system constructed using the filtered correspondences.

Another advantage of the essential matrix against the fundamental matrix is since it is free from the intrinsic parameters the relative pose can be easily extracted from it [72]. First, we assume that $\hat{\mathbf{P}}_i = [\mathbf{I} \mid \mathbf{0}]$, then we are looking for $\hat{\mathbf{P}}_j = [\mathbf{R}_j \mid \mathbf{R}_j\mathbf{t}_j]$, *i.e.* the relative pose of camera j with respect to camera i. While the extraction is straightforward [72], it results in four possible solutions and a scale ambiguity which affects the length of $\mathbf{t}_j$. Fortunately, the correct solution can be easily chosen from the four candidates by observing which setting has the most points in front of both cameras [72]. The scale ambiguity is handled by normalizing the length of $\mathbf{t}_j$ to be 1. This works perfectly for a stereo setting containing two cameras only, but it has to be made consistent within a camera network.

2.2.2 Register the Cameras into a Common Frame

In the next problem, we have to register the extracted relative poses into a common world coordinate system. In order to do this, we have to assign a global origin with the principal axes to the system and make the relative positions consistent inside the network. In most approaches the origin is attached to an arbitrary *main camera* [72]. Without the loss of generality, let us assume that $\mathbf{P}_1$ is the main camera of the system, therefore $\hat{\mathbf{P}}_1 = [\mathbf{I} \mid \mathbf{0}]$.

Achieving scale consistency within the network is a more challenging task [77]. The main goal in this step is to estimate a scale factor for each camera to express the length of its relative translation vector in the world coordinate frame. Formally, we are looking for a λ_i value, which transforms the camera parameters in the following way:

$$\hat{\mathbf{P}}_i = [\mathbf{R}_i \mid \lambda_i\mathbf{R}_i\mathbf{t}_i]. \tag{2.12}$$

Conventional approaches register the cameras incrementally, by first adding the neighbors of the *main camera* to the frame, then iteratively growing the network by considering the

[1] According to [78] the problem has at most 10 solutions including the complex ones too.

neighbors of the cameras already in the network [73, 77, 92, 93]. Note that, for cameras which are not connected to the *main camera* in the vision graph the relative pose have to be adjusted as well. The λ_i values estimated using a set of 3D points reconstructed via triangulation [72]. Theoretically, the availability of one [77] or two [73] points are enough to estimate the scales, in practice the estimation is made from much more points, thus the final solution will be an aggregated value (*e.g.* by taking the median of the estimated values).

The main idea behind these scale estimation processes is that if the scales are inconsistent between two pairs of cameras, the same set of corresponding points will have very different reconstructed 3D coordinates as well. Then, let us assume that P_i and P_j are already in a consistent system and we would like to add a new P_k camera by estimating λ_k. Moreover, we assume that there are at least one or two corresponding points (and hence we use the one or two point algorithms, respectively), which are visible from all of these three cameras. The one point algorithm first reconstructs a 3D point using P_i and P_j, then using P_i and P_k. Since P_k is inconsistent with the camera network the firstly reconstructed $\bar{\mathbf{X}}^{ij}$ will be different to the secondly obtained $\bar{\mathbf{X}}^{ik}$. The λ_k ratio is then obtained as

$$\lambda_k = \frac{\|\bar{\mathbf{X}}^{ij} - \mathbf{R}_i\mathbf{t}_i\|}{\|\bar{\mathbf{X}}^{ik} - \mathbf{R}_i\mathbf{t}_i\|}, \tag{2.13}$$

i.e. the ratio of distances between the reconstructed points and the common camera P_i [77]. The two point algorithm, works similarly, but it uses the ratios of distances between a pair of reconstructed 3D points [73]:

$$\lambda_k = \frac{\|\bar{\mathbf{X}}_1^{ij} - \bar{\mathbf{X}}_2^{ij}\|}{\|\bar{\mathbf{X}}_1^{ik} - \bar{\mathbf{X}}_2^{ik}\|}. \tag{2.14}$$

Another type of common algorithms is the factorization based methods [85, 94]. Herein, camera poses, 3D coordinates and scale factors are obtained simultaneously via matrix factorization techniques. While these techniques introduce compact frameworks for the estimation, they are inherently affected by missing data and outlier handling issues. These problems have been addressed in a recent work [95], by formalizing the factorization problem using an Augmented Lagrangian numerical scheme.

Finally, there are methods for estimating all of the camera poses together in two steps, the so-called global methods [77]. These methods first extract the absolute rotations, then estimate the translations. In [77], the camera centers obtained via a solution of a linear system, which does not involve any 3D reconstruction at all. However, the method does not work when the camera centers are collinear, for such cases the algorithm falls back to a previously mentioned one point scale estimation method [77].

2.2.3 Bundle Adjustment

In the last step of a typical structure from motion pipeline, the estimated relative poses and the reconstructed 3D information are optimized [80, 93, 96]. This process is called bundle adjustment (BA) [86, 87] and it aims to minimize the reprojection error of the 3D points by adjusting all of the estimated camera parameters (extrinsic and/or intrinsic) and the 3D coordinates.

Observing bundle adjustment as a numerical method, it corresponds to a large-scale nonlinear least-squares problem [86, 87]. However, due to a large number of parameters involved in the process, the general purpose algorithms are not efficient in solving the

problem [87]. Fortunately, the interactions between parameters are limited, therefore the corresponding the Jacobian matrix will be sparse [87], which could be used to construct an optimal solver.

Chapter 3

Deformable Registration of 3D Objects

In this chapter, we will continue with one of the main topics of the current work and propose a general framework to solve the non-linear registration problem between three-dimensional (3D) objects. As we already mentioned in the previous chapters, with the spread of modern 3D content acquisition devices, also the need for registering objects coming from various sources is increasing. In the following, we will focus on surface alignment, where the effect of the physical deformation is only measurable on the surface of the objects. Since the proposed approach relies on geometrical information only, we are dealing with a true binary registration problem here. The proposed framework is rooted in a successful 2D approach presented in [7], however, the extension to 3D is not straightforward: there are various 3D specific details, which should be handled differently.

The current work aims to investigate the practical usefulness of various data representation available in 3D. As an example, state of the art methods of the topic do not distinguish between open and closed surfaces. In this work, we will show that each representation could lead to very different approaches in the proposed framework, having its own advantages and limitations. The general framework is able to work with voxel representation [Sánta and Kato, 2012a][Sánta and Kato, 2012b] and with open and closed triangular surfaces [Sánta and Kato, 2018][Sánta and Kato, 2016a][Sánta and Kato, 2013a]. Possible practical applications will be also investigated, like lung surface and face alignment.

3.1 Problem Statement

Let us formulate now the alignment problem: Given a pair of *template* and *observation* objects denoted by $\mathcal{F}_t \subset \mathbb{R}^3$ and $\mathcal{F}_o \subset \mathbb{R}^3$, respectively, we are looking for the aligning transformation φ such that for all $\mathbf{x} \in \mathcal{F}_t$ there exists a $\mathbf{y} \in \mathcal{F}_o$ satisfying the so-called *identity relation* [Sánta and Kato, 2018][Sánta and Kato, 2013a]

$$\varphi(\mathbf{x}) \quad = \quad \mathbf{y} \tag{3.1}$$

In classical landmark-based approaches, a large number of corresponding landmarks extracted from $\mathcal{F}_t$ and $\mathcal{F}_o$, giving sufficiently many constraints through Equation (3.1) to find the parameters of the transformation. What can we do if such point correspondences are not

available? Let us integrate out individual point pairs in Equation (3.1) over the foreground domains of the objects yielding the following equation [Sánta and Kato, 2018][Sánta and Kato, 2013a]:

$$\int_{\mathcal{F}_o} \mathbf{y}\, d\mathbf{y} = \int_{\varphi(\mathcal{F}_t)} \mathbf{z}\, d\mathbf{z}. \tag{3.2}$$

One can consider Equation (3.2) as an object level *identity relation*, because here we only require, that the object domains $\mathcal{F}_t$ and $\mathcal{F}_o$ are in correspondence as a whole. While in landmark-based approaches each point correspondence will generate a new equation of the form Equation (3.1); Equation (3.2) provides only a very limited number of equations (exactly three for $\mathbb{R}^3$)! As a consequence, Equation (3.2) alone will not provide sufficiently many equations to solve for the transformation parameters. In order to generate more equations, observe that Equation (3.1) (hence Equation (3.2)) remains valid when a non-linear $\omega : \mathbb{R}^3 \to \mathbb{R}$ function is acting on both sides [7]. Thus adopting a set of independent non-linear functions $\{\omega_i\}_{i=1}^{\ell}$ yields a system of ℓ equations [Sánta and Kato, 2018][Sánta and Kato, 2013a]:

$$\int_{\mathcal{F}_o} \omega_i(\mathbf{y})\, d\mathbf{y} = \int_{\varphi(\mathcal{F}_t)} \omega_i(\mathbf{z})\, d\mathbf{z} \qquad i = 1, \ldots, \ell. \tag{3.3}$$

The parameters of φ estimated directly as a solution of the system of equations. This implies that we will need ℓ to be at least N, where N is the number of parameters of φ. In practice, however, the input objects are usually subject to various types of noise, therefore an overdetermined system is recommended, thus $\ell \geq N$. This system is solved in the least squares sense with an arbitrary iterative optimizer. Now let us continue with observing the behavior of the proposed formalism on different types of input objects.

When we are dealing with objects in the 3D Euclidean space, practically, the possible domains are either surfaces or volumetric objects. Therefore, we will formalize our framework to handle these two types of domains.

Now, let us assume that the object domains $\mathcal{F}_t$ and $\mathcal{F}_o$ are surfaces, thus Equation (3.3) involves surface integrals over these domains [Sánta and Kato, 2018][Sánta and Kato, 2016a]. To get an explicit formula for these integrals, let us consider that the surface patches $\mathcal{F}_t$, $\varphi(\mathcal{F}_t)$ and $\mathcal{F}_o$ are parameterized over the 2D domains $\mathcal{S}_t \subset \mathbb{R}^2$, $\mathcal{S}_\varphi \subset \mathbb{R}^2$ and $\mathcal{S}_o \subset \mathbb{R}^2$ via

$$\begin{aligned} r_t &: \mathcal{S}_t \to \mathcal{F}_t, \\ r_\varphi &: \mathcal{S}_\varphi \to \varphi(\mathcal{F}_t), \\ r_o &: \mathcal{S}_o \to \mathcal{F}_o, \end{aligned}$$

functions, respectively. Using these vector functions, the surfaces can be parametrized in a 2D space

$$\begin{aligned} \forall \mathbf{x} \in \mathcal{F}_t &: \mathbf{x} = r_t(\mathbf{w}), \mathbf{w} \in \mathcal{S}_t \\ \forall \mathbf{z} \in \varphi(\mathcal{F}_t) &: \mathbf{z} = r_\varphi(\mathbf{u}), \mathbf{u} \in \mathcal{S}_\varphi \\ \forall \mathbf{y} \in \mathcal{F}_o &: \mathbf{y} = r_o(\mathbf{v}), \mathbf{v} \in \mathcal{S}_o, \end{aligned}$$

yielding the following form of the integral equation in Equation (3.3)

$$\int_{\mathcal{F}_o} \omega_i(\mathbf{y})\, d\mathbf{y} = \iint_{\mathcal{S}_o} \omega_i(r_o(\mathbf{v})) \left\| \frac{\partial r_o}{\partial v_1} \times \frac{\partial r_o}{\partial v_2} \right\| d\mathbf{v}, \tag{3.4}$$

$$\int_{\varphi(\mathcal{F}_t)} \omega_i(\mathbf{z})\, d\mathbf{z} = \iint_{\mathcal{S}_\varphi} \omega_i(r_\varphi(\mathbf{u})) \left\| \frac{\partial r_\varphi}{\partial u_1} \times \frac{\partial r_\varphi}{\partial u_2} \right\| d\mathbf{u}, \tag{3.5}$$

where on the right-hand sides each vector function is multiplied by the surface element, *i.e.* the magnitude of the cross product of the embedding functions' partial derivatives.

The computational complexity of the proposed method is largely determined by the calculation of the integrals in Equation (3.5): The integral over $\mathcal{S}_o$ from Equation (3.4) is constant which needs to be computed only once. However, since the unknown transformation φ is involved in the integrals of Equation (3.5), this formalism needs to generate the $\varphi(\mathcal{F}_t)$ domain at each iteration of the least squares solver, which might be ineffective in a practical application. In such cases, Equation (3.5) could be rewritten as

$$\int_{\varphi(\mathcal{F}_t)} \omega_i(\mathbf{z})\, d\mathbf{z} = \iint_{\mathcal{S}_t} \omega_i(\varphi(r_t(\mathbf{u}))) \left\| \frac{\partial(\varphi \circ r_t)}{\partial u_1} \times \frac{\partial(\varphi \circ r_t)}{\partial u_2} \right\| d\mathbf{u}, \tag{3.6}$$

by making use of the integral transformation. Herein, $r_t(\mathbf{u})$ is constant, hence it can be precomputed, although the computation of the surface elements becomes more complex. First, let us rewrite the derivatives of the composite function $\varphi \circ r_t$ in terms of the Jacobian $\mathbf{J}_\varphi$ of φ and the gradients of r_t:

$$\left\| \frac{\partial(\varphi \circ r_t)}{\partial u_1} \times \frac{\partial(\varphi \circ r_t)}{\partial u_2} \right\| = \left\| \mathbf{J}_\varphi(r_t(\mathbf{u})) \frac{\partial r_t}{\partial u_1} \times \mathbf{J}_\varphi(r_t(\mathbf{u})) \frac{\partial r_t}{\partial u_2} \right\|. \tag{3.7}$$

Since the gradients of r_t are independent of φ, they can also be precomputed and used in the above formula. Consequently, only $\varphi(r_t(\mathbf{u}))$ and $\mathbf{J}_\varphi(r_t(\mathbf{u}))$ have to be calculated during the iterations.

When the input objects $\mathcal{F}_o$ and $\mathcal{F}_t$ have non-vanishing volumes, the integrals in Equation (3.3) can also be interpreted as volume integrals [Sánta and Kato, 2013a][Sánta and Kato, 2012a]. From a practical point of view, it is easy to see that such an interpretation yields numerically more stable algorithm: any error in the surfaces $\mathcal{F}_o$ and $\mathcal{F}_t$ will inherently cause an error in the integrals and hence the equality in Equation (3.3) will not be true anymore. The smaller these errors are, the better we approximation we get for the true equality and hence we can expect a better performance of the algorithm.

Handling the equations from Equation (3.3) as volumetric integrals leads to a much simpler derivation than the one we obtained for the surface integration case, because the estimation does not involve any further parametrization of the objects here. Although, the formalism is simpler, the volumetric objects could contain much larger amount of points than the surface objects. Hence the generation of the integration domains could be much less effective when we use the formulas from Equation (3.3). Similarly to the surface integration case, we can handle the problem by using the corresponding integral transformation [Sánta and Kato, 2012a]:

$$\int_{\mathcal{F}_o} \omega_i(\mathbf{y})\, d\mathbf{y} = \int_{\mathcal{F}_t} \omega_i(\varphi(\mathbf{x}))|J_\varphi(\mathbf{x})|\, d\mathbf{x}, \quad i = 1, \ldots, \ell. \tag{3.8}$$

While theoretically, we can use the integral transformation for both integration approaches, in practice we choose the most effective formalism. Note that, mathematically the

formalisms are equal, thus the difference is in the computational complexity. Choosing the best formalism depends on many factors. On one hand, for the formula described in Equation (3.3), we have to generate the whole domains in each iteration, which could increase the complexity depending on the input representation. On the other hand, the formula from Equation (3.8) involves the Jacobian of the transformation. For linear or piece-wise linear transformations the Jacobian will be constant, therefore its computation is independent from the input size. For a general non-rigid transformation, the Jacobian could be different at each point of the input domain and it needs to be recalculated for each iteration. Although, in practice, the exact formulas are available for possible parametric transformations, thus it can be estimated efficiently.

Besides the mentioned components, choosing the right $\{\omega_i\}$ set is also an important issue. In the current framework, the ω_i should be chosen to ensure that the obtained equations are independent and covariant with respect to the pursued transformation and the whole set is rich enough to describe the effect of the deformation. This issue has been addressed by several preceding works [7, 97]. Generally speaking, in methods built on linear systems, it is particularly important to choose the right function set for the right outcome [97]. However, for non-linear systems, it has a minor influence on the solution [7]. Although using different $\{\omega_i\}$ sets could lead to very different, even inferior solutions; Domokos *et al.* have shown that this issue is rather a numeric than a conceptual problem [7]. Therefore, it could be handled easily by normalizing the system with properly generated, function set dependent constants [7].

For the matter of computational efficiency, it is a good practice to choose the functions to be as simple as possible. Therefore, in the current work, we use simple *power functions* in the form of

$$\omega_i(\mathbf{x}) = x_1^{m_i} x_2^{n_i} x_3^{o_i} \qquad m_i, n_i, o_i \geq 0, \tag{3.9}$$

where $i = 1, \ldots, \ell$.

Now, let us elaborate the review from Section 2.1 by observing the relationship between the proposed approaches and the literature. The proposed framework is related to the surface registration approaches [21, 98], because the effect of the transformation can be measured on the surface only. However, there are some differences in handling of the input data, which needs further investigations.

For any surface registration method, one of the main challenges is how to handle noise, which usually caused by perturbations of vertices or by errors in the segmentation [21]. Since the input domain of these algorithms is restricted to the surface, this could lead to an inefficiently small signal-to-noise ratio. This observation will be valid for our surface integration based approach as well, since any amount of noise on each vertex will challenge drastically the surfaces from Equations (3.4)–(3.5) too. However, the surface noise has much lower impact on the volumetric approach, by the reason of the error is distributed on a much larger volumetric domain in Equation (3.3). This also yields that the volumetric approach will be more stable numerically than the surface-based approach.

3.2 Modeling the Deformation

3.2.1 Polynomial Model

A broadly used class of deformations is the polynomial family. In the three-dimensional case, the deformation field $\pi : \mathbb{R}^3 \to \mathbb{R}^3$, $\pi(\mathbf{x}) = [\pi_1(\mathbf{x}), \pi_2(\mathbf{x}), \pi_3(\mathbf{x})]$ is given by three polynomial functions $\pi_i : \mathbb{R}^3 \to \mathbb{R}$ [Sánta and Kato, 2012b]. Without loss of generality, we can assume that $d = \deg(\pi_1) = \deg(\pi_2) = \deg(\pi_3)$:

$$\begin{aligned}
\pi_1(\mathbf{x}) &= \sum_{i=0}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} a_{ijk} x_1^i x_2^j x_3^k, \\
\pi_2(\mathbf{x}) &= \sum_{i=0}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} b_{ijk} x_1^i x_2^j x_3^k, \\
\pi_3(\mathbf{x}) &= \sum_{i=0}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} c_{ijk} x_1^i x_2^j x_3^k.
\end{aligned} \tag{3.10}$$

The transformation has a total of $N = (d+3)(d+2)(d+1)/2$ parameters. The Jacobian determinant of the transformation composed of the following partial derivatives:

$$\frac{\partial \pi_1}{\partial x_1} = \sum_{i=1}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} i a_{ijk} x_1^{i-1} x_2^j x_3^k, \tag{3.11}$$

$$\frac{\partial \pi_1}{\partial x_2} = \sum_{j=1}^{d} \sum_{i=0}^{d-j} \sum_{k=0}^{d-i-j} j a_{ijk} x_1^i x_2^{j-1} x_3^k, \tag{3.12}$$

$$\frac{\partial \pi_1}{\partial x_3} = \sum_{k=1}^{d} \sum_{i=0}^{d-k} \sum_{j=0}^{d-i-k} k a_{ijk} x_1^i x_2^j x_3^{k-1}, \tag{3.13}$$

and for π_2 and π_3, we get a similar formula.

The main advantage of polynomial deformations over the spline based models is the fewer number parameters and the fact that polynomials are acting *globally* on the shapes, hence regularization is not needed. Moreover, many non-polynomial transformations can be approximated by a polynomial one *e.g.* via a Taylor expansion [7]. The proposed framework also benefits from using polynomial transformations with volumetric images which will be elaborated in Section 3.3.1.

3.2.2 Thin Plate Splines

Thin Plate Splines (TPS) [33, 34, 99] transformation is commonly used as a parametric model for elastic deformations. In 3D, a TPS transformation $\varsigma : \mathbb{R}^3 \to \mathbb{R}^3$ can be decomposed as three coordinate functions $\varsigma(\mathbf{x}) = [\varsigma_1(\mathbf{x}), \varsigma_2(\mathbf{x}), \varsigma_3(\mathbf{x})]^T$, $\forall \varsigma_i(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$. Given a set of control points $c_k \in \mathbb{R}^3$ and associated mapping coefficients $a_{ij}, w_{ki} \in \mathbb{R}$ with $i = 1, \ldots, 3, j = 1, \ldots, 4$ and $k = 1, \ldots, K$, the TPS functions are

$$\varsigma_i(\mathbf{x}) = a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + a_{i4} + \sum_{k=1}^{K} w_{ki} U(\|c_k - \mathbf{x}\|), \tag{3.14}$$

where $U(r)$ is called radial basis function. The number of parameters $N = 3(K+4)$, consisting of 12 affine parameters a_{ij} and 3 local coefficients w_{ki} for each of the K control points

c_k. Note that, in 3D the radial basis function has the following form: $U(r) = -r$ [99].[1] The elements of the Jacobian matrix of ς are obtained as:

$$\frac{\partial \varsigma_i}{\partial x_j} = a_{ij} - \sum_{k=1}^{K} w_{ki} \frac{c_{kj} - x_j}{\|\mathbf{c}_k - \mathbf{x}\|} \qquad i, j = 1, 2, 3. \tag{3.15}$$

The local parameters are also required to satisfy the following additional constraints [34, 99], ensuring that the TPS at infinity behaves according to its affine term:

$$\sum_{k=1}^{K} w_{ki} = 0 \quad \text{and} \quad \sum_{k=1}^{K} c_{k_j} w_{ki} = 0 \qquad i, j = 1, 2, 3. \tag{3.16}$$

The algebraical meaning of these constraints is to ensure that the w_{ki} parameters are coming from the left null space of $[\mathbf{1} \mid \mathbf{CP}]$, a matrix containing the homogeneous coordinates of the control points [5].

When correspondences are available, exact mappings of the control points are also known which, using Equation (3.14), provides constraints on the unknown parameters. Thus in classical correspondence based approaches, control points placed at extracted point matches, and the deformation at other positions is interpolated by the TPS. Therefore in such cases, a TPS can be regarded as an optimal *interpolating* function, whose parameters are usually recovered via a complex optimization procedure [33, 34].

However, in the current approach, TPS is used in a quite unusual way, introduced by [7]. Herein, it is a parametric model to *approximate* the true deformation and the parameters determined as the solution of Equation (3.3) [Sánta and Kato, 2018][Sánta and Kato, 2013a][Sánta and Kato, 2012a].

Considering registration as a model fitting problem, we can use the terms *over-* and *underfitting* [21]. Herein underfitting refers to the case where the underlying deformation has higher Degrees of Freedom (DoF) than the approximating transformation, while overfitting occurs when the positions are reversed. In non-rigid registration, the overfitting problem is particularly interesting when the input is challenged by noise and outliers since the transformations with higher DoF could align the template to the outliers as well. Both issues are affected by the positions of the control points and overfitting usually handled by applying additional regularization constraints on the transformation.

Control Point Placement

Since the presented framework aims to find the globally best alignment, the control point placement has major effect on the outcome. In the presence of prior knowledge about the physical deformation or the image acquisition method, it is easier to find the right control point positions or learn an application specific TPS transformation [100, 101].

In the current approach, we have examined two control point placement strategies without the assumption of any prior knowledge. In the first approach, the control points have been placed on a uniform grid in order to capture local deformations everywhere. Obviously, a finer grid would allow a more refined approximation of the deformation field at the price of increased number of free parameters [Sánta and Kato, 2018][Sánta and Kato, 2013a][Sánta and Kato, 2012a]. In the second strategy, the control points were uniformly

[1] Following the work of [99], there is an additional constant term in the radial basis function, but in practice, it is encoded into the local parameters.

sampled on the *template* surface maximizing the geodesic distance between the closest points. This method is also known as the *Farthest Point Sampling* [102]. Note that, this approach places the control points on the surface only [Sánta and Kato, 2018][Sánta and Kato, 2016a].

Regularization

In some applications, using spatial information only does not provide enough constraints on the transformation model, thus many equivalent solutions are available on the level of objects. Moreover, in several scenarios, it is necessary to determine a smooth, diffeomorphic solution. It is a well-known fact that thin plate splines may not be diffeomorphic without further regularization [103–105], although several techniques can be found in the literature on determining a diffeomorphic solution. For example, in the landmark based approaches an energy function, composed of the derivatives of the displacement field, is minimized [103] or a flow of diffeomorphisms is defined, then using its velocity field and a linear differential operator the deformation energy is minimized to achieve a diffeomorphic transformation [104, 105].

In our approach we are looking for a solution which not only solves our system of equations but also minimizes the *bending energy* of the transformation:

$$E_{bending} = \lambda \int_{\mathcal{F}_t} \left\{ \left(\frac{\partial^2 \varsigma}{\partial x_1^2} \right)^2 + \left(\frac{\partial^2 \varsigma}{\partial x_2^2} \right)^2 + \left(\frac{\partial^2 \varsigma}{\partial x_3^2} \right)^2 + 2 \left(\frac{\partial^2 \varsigma}{\partial x_1 \partial x_2} \right)^2 + 2 \left(\frac{\partial^2 \varsigma}{\partial x_2 \partial x_3} \right)^2 + 2 \left(\frac{\partial^2 \varsigma}{\partial x_3 \partial x_1} \right)^2 \right\} d\mathbf{x}, \quad (3.17)$$

where $\lambda > 0$ is an application dependent parameter.

In our experiments, using the *bending energy* minimization provided sufficient constraint to obtain diffeomorphic solutions. This finding has been experimentally confirmed on synthetic volumetric datasets by computing the Jacobian of the obtained transformations for all of the input voxels: the values were positive in every case.

3.3 Efficient Computation of Integrals over 3D Objects

Object representation also has a strong influence on the nature of the underlying optimization problem. As discussed in Section 2.1.2, the most common object representations are unstructured point sets, voxel representation, and triangular surface meshes. In the current work, we are dealing with the latter two representations. Since these representations are only discrete approximations of the true objects, the integrals of the generated systems will be only approximately valid too.

In the first subsection, we will focus on voxel images. Recall that, this representation is mainly obtained via tomographic reconstruction and practical applications usually come from medical fields. Unfortunately, the huge amount of data leads to higher computational complexity, but using polynomial equations the voxel coordinates could be separated from the parameters. In practice, this means that we only have to iterate the coordinates once for each pair of input objects.

In the subsequent sections, we will work with triangular surface meshes. A triangular surface can be either open (non-closed) or closed. Open surfaces have zero volume, thus they can be used with surface integration only. This approach is more general, hence it is

not limited to *watertight* meshes (*i.e.* assuming strict topological properties like consistent orientation and the mesh have to be free from surface holes), yet surface noise has bigger influence on the accuracy. Closed surfaces, however, could be used with surface and volumetric integrals too. Opposed to the surface base method, the volumetric approach has very good tolerance against the surface noise, but it relies on the topological properties described above. Both of these approaches could be used with arbitrary transformation models and $\{\omega_i\}$ set, although in the current work we will focus on thin plate splines and power functions from Equation (3.9).

3.3.1 Voxel Representation

In voxel representation, we assume that the input is defined as a set of identical volumetric elements (or voxels) with 1 as volume. Let us denote the voxel approximation of the *template* and the *observation* by F_t and F_o, respectively. Using the notations from Section 3.1:

$$F_t \approx \mathcal{F}_t \quad \text{and} \quad F_o \approx \mathcal{F}_o. \tag{3.18}$$

Since voxels are volumetric objects, integration over these domains will lead to a volumetric integral, which can be estimated as a finite sum over the voxels [Sánta and Kato, 2012a][Sánta and Kato, 2012b]. Therefore, the sides of the equations from Equation (3.3) will become

$$\sum_{\mathbf{Y}\in F_o} \omega_i(\mathbf{Y}) \approx \int_{\mathcal{F}_o} \omega_i(\mathbf{y})\, d\mathbf{y}, \tag{3.19}$$

$$\sum_{\mathbf{Z}\in \varphi(F_t)} \omega_i(\mathbf{Z}) \approx \int_{\varphi(\mathcal{F}_t)} \omega_i(\mathbf{z})\, d\mathbf{z}, \tag{3.20}$$

where $i = 1, \ldots, \ell$. However, the generation of $\varphi(F_t)$ could be practically ineffective for high resolution images, hence we will use the integral transformation equations from Equation (3.8) instead. The final form of the system using voxel representation will be

$$\sum_{\mathbf{Y}\in F_o} \omega_i(\mathbf{Y}) = \sum_{\mathbf{X}\in F_t} \omega_i(\varphi(\mathbf{X}))|J_\varphi(\mathbf{X})|, \tag{3.21}$$

where $i = 1, \ldots, \ell$. Substituting Equation (3.9) into Equation (3.21) we get

$$\sum_{\mathbf{Y}\in F_o} Y_1^{m_i} Y_2^{n_i} Y_3^{o_i} = \sum_{\mathbf{X}\in F_t} \varphi_1(\mathbf{X})^{m_i} \varphi_2(\mathbf{X})^{n_i} \varphi_3(\mathbf{X})^{o_i} |J_\varphi(\mathbf{X})|, \tag{3.22}$$

where $i = 1, \ldots, \ell$. The overall computational complexity is $\mathcal{O}(\ell I|F_t| + \ell|F_o|)$, where I is the number of evaluation made by the solver and $|F_t|$ and $|F_o|$ are the number of voxels in the *template* and the *observation*, respectively. While this system could be used with any parametric transformation model [Sánta and Kato, 2012a][Sánta and Kato, 2012b], an efficient numerical scheme can be derived for models, where the product on the right-hand side is polynomial [Sánta and Kato, 2012a]. An obvious candidate for such setting is the general polynomial transformation from Equation (3.10).

Computation of Polynomial Equations over Voxel Objects

Now let us investigate how to utilize polynomial equations within the proposed framework. Consider the right-hand side of the i^{th} equation from Equation (3.22) with polynomial transformation from Equation (3.10):

$$\sum_{\mathbf{X}\in F_t} \pi_1(\mathbf{X})^{n_i}\pi_2(\mathbf{X})^{m_i}\pi_3(\mathbf{X})^{o_1}|J_\pi(\mathbf{X})|, \qquad i=1,\ldots,\ell. \tag{3.23}$$

Using such model, products of the powered transformation functions are polynomials too, and so as the Jacobian determinant of the transformation Equation (3.13). Moreover, their product will be also polynomial, with the degree of $d_i = d(n_i + m_i + o_i) + 3(d-1)$,

$$\sum_{\mathbf{X}\in F_t} \pi_1(\mathbf{X})^{n_i}\pi_2(\mathbf{X})^{m_i}\pi_3(\mathbf{X})^{o_1}|J_\pi(\mathbf{X})| = \sum_{\mathbf{X}\in F_t}\sum_{q=0}^{d_i}\sum_{r=0}^{d_i-q}\sum_{s=0}^{d_i-q-r} g_{iqrs}X_1^qX_2^rX_3^s, \tag{3.24}$$

where $i = 1,\ldots,\ell$ and d is the degree of the transformation polynomial. According to the *Multinomial theorem*, g_i is a polynomial of the parameters of the transformation [Sánta and Kato, 2012b]. Notice that, the g_i functions are independent from the voxels. Therefore, using the basic properties of integration, the sums can be rearranged as

$$\sum_{\mathbf{X}\in F_t} \pi_1(\mathbf{X})^{n_i}\pi_2(\mathbf{X})^{m_i}\pi_3(\mathbf{X})^{o_1}|J_\pi(\mathbf{X})| = \sum_{q=0}^{d_i}\sum_{r=0}^{d_i-q}\sum_{s=0}^{d_i-q-r} g_{iqrs}\sum_{\mathbf{X}\in F_t} X_1^qX_2^rX_3^s, \tag{3.25}$$

Observe that the sum over the voxel coordinates is independent from the parameters of the transformation functions. This means the integrals over the template coordinates are independent from the unknown parameters, hence these integrals have to be computed only once for the whole registration procedure.

Using these results, the integrals can be separated from the transformation parameters which can reduce the computational time, but note that these modifications do not affect the left-hand side of the equations [Sánta and Kato, 2012b].

Let us follow by observing the computational complexity of the polynomial scheme. First, we will denote the number of terms in a three-dimensional full polynomial by $\gamma(d) = (d+3)(d+2)(d+1)/6$, where d is the degree of the polynomial. Since the integrals from Equation (3.25) are independent from the parameters of the transformation model, we could precalculate the sum over the coordinates for all ω_i function in $\mathcal{O}(|F_t|\gamma(d_M))$ time, where $d_M = \max\{d_i\}$, using a simple recursive calculation scheme.

Now, let us examine the g_{iqrs} functions. In the worst case, they are full polynomials of the transformation parameters with a degree of $n_i + m_i + o_i + 1$, which comes from the degree of the power functions and from the Jacobian determinant. The number of these functions for the i^{th} equation is $\gamma(d_i)$. Therefore the total number of operations are $\mathcal{O}(\gamma(d_i)\gamma(n_i+m_i+o_i+1))$, if we assume that one term of q_{iqrs} can be computed in constant time.

The overall computation time with this modifications is

$$\mathcal{O}(\ell|F_o| + \gamma(d_M)|F_t| + I\sum_{i=1}^{\ell}\gamma(d_i)\gamma(n_i+m_i+o_i+1)) \tag{3.26}$$

where I is the number of the necessary function calls made by the least squares solver.

How does this complexity relate to the complexity of the general system from Equation (3.22)? For an example, consider the following case: we want to use a second order polynomial deformation (*i.e.* $d = 2$) to approximate the underlying deformation field. To solve that we need at least 30 ω_i functions. Let us use the first 35 power functions with a maximal degree of 3 from the following set:

$$\{(n_i, m_i, o_i)\}_{i=1}^{35} = \{(a, b, c) \mid 0 \leq a = 0, \ldots, 2, b = 0, \ldots, 2, c = 0, \ldots, 3\}. \tag{3.27}$$

Using the formulas above we have

$$\begin{aligned} d_M &= 15, \\ \gamma(d_M) &= 816, \\ \sum_{i=1}^{35} \gamma(d_i)\gamma(n_i + m_i + o_i + 1) &= 816832. \end{aligned}$$

In our experiments, the average of the number of function calls was 377 for this model and ω_i set. Note, that the number of iterations is the same for both methods since the modifications affect only the computational time of a single iteration. Hence, the computational complexity of the optimized scheme will be

$$\mathcal{O}(35|F_o| + 816|F_t| + 377 \cdot 816, 832). \tag{3.28}$$

The complexity of the naive computation scheme for Equation (3.22) is

$$\mathcal{O}(35|F_o| + 377(35 + 30 + 36)|F_t|), \tag{3.29}$$

where the terms of the sum $(35 + 30 + 36)$ correspond to the numbers of necessary steps for calculating the ω_i functions, transforming the voxel coordinates and estimating Jacobian of the transformation, respectively. Comparing these two complexities leads to that it is worth to use the modifications as long as

$$|F_t| \geq 8, 265. \tag{3.30}$$

3.3.2 Triangular Surface Mesh Representation

In the following, we will assume that the *template* and the *observation* surfaces are represented by their triangular surface meshes. Let us denote the meshes of the *template* and the *observation* surfaces by $T_\triangle$ and $O_\triangle \subset \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$, respectively. Next, observe that the transformation φ acting on the mesh $T_\triangle$ generates a new surface mesh by transforming the vertices of the triangles:

$$\varphi(T_\triangle) = \{(\varphi(\mathbf{A}), \varphi(\mathbf{B}), \varphi(\mathbf{C})) \mid (\mathbf{A}, \mathbf{B}, \mathbf{C}) \in T_\triangle\}. \tag{3.31}$$

Since the transformed object can be estimated easily, herein we do not have to use the integral transformation, thus we can work directly with Equation (3.3).

Computing Surface Integrals over Triangular Meshes

Let us focus on the surface integration first, by building upon Equation (3.4) and Equation (3.5). Note that, triangular surface meshes give piecewise linear approximations of the true surfaces only, hence

$$\begin{aligned} T_\triangle &\approx \mathcal{F}_t, \\ O_\triangle &\approx \mathcal{F}_o, \\ \varphi(T_\triangle) &\approx \varphi(\mathcal{F}_t), \end{aligned} \tag{3.32}$$

and thus the integrals over the triangular surfaces will be approximations of the integrals over the true surfaces as well. Using these notations, surface integrals over triangular surfaces can be expressed as sums of integrals over the triangles of each mesh [Sánta and Kato, 2018][Sánta and Kato, 2016a]:

$$\int_{O_\triangle} \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}} = \sum_{o \in O_\triangle} \int_o \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}}, \tag{3.33}$$

$$\int_{\varphi(T_\triangle)} \omega_i(\hat{\mathbf{z}})\, d\hat{\mathbf{z}} = \sum_{\pi \in \varphi(T_\triangle)} \int_\pi \omega_i(\hat{\mathbf{z}})\, d\hat{\mathbf{z}}, \tag{3.34}$$

where $i = 1, \ldots, \ell$. The triangular mesh approximation also simplifies the parametrization of the surfaces, since unlike the formalism described in Equations (3.4)–(3.5), not a complex 3D surface have to be parameterized, only the triangles of the approximate meshes.

Now, we will derive two different numerical schemes for the calculation of surface integrals. While the first approach aims to estimate the exact integrals over the triangular surfaces, the second one uses a linear approximation of the ω_i functions, giving only an approximately valid solution [Sánta and Kato, 2018][Sánta and Kato, 2016a].

First, let us focus on the exact approach. Following [106], in order to obtain the integrals analytically, we will use the barycentric parametrization of the triangles. Considering an arbitrary triangle $o = (\mathbf{A}, \mathbf{B}, \mathbf{C})$, every $\mathbf{p}$ point of the triangle can be expressed as weighted sums of the vertices:

$$\mathbf{p} = u\mathbf{A} + v\mathbf{B} + w\mathbf{C}, \tag{3.35}$$

with $u, v, w \geq 0$ and $u + v + w = 1$. The w parameter could be given by using u and v as $w = 1 - u - v$, yielding only two free parameters. For giving the integrals in the barycentric domain, we have to estimate the area changes induced by the parametrization too. Similarly to Equation (3.5), the area term is given as the magnitude of the cross product of the partial derivatives of the coordinate transformation w.r.t. u and v. Thus, using these notations we get:

$$\|(\mathbf{A} - \mathbf{C}) \times (\mathbf{B} - \mathbf{C})\| = 2\,\mathrm{area}(o),$$

where $\mathrm{area}(o)$ corresponds to the area of the triangle o. Now, we can give the integral of Equation (3.33) over a triangle in the barycentric domain as

$$\sum_{o \in O_\triangle} \int_o \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}} = \sum_{o \in O_\triangle} 2\,\mathrm{area}(o) \int_0^1 \int_0^{1-u} \omega_i(u\mathbf{A} + v\mathbf{B} + w\mathbf{C})\, dv\, du, \tag{3.36}$$

where $i = 1, \ldots, \ell$ and $w = 1 - u - v$. A similar formula can be derived for Equation (3.34) also [106].

Let us use the $\{\omega_i\}$ set from Equation (3.9) and define the exponents as

$$\{(n_i, m_i, o_i)\}_{i=1}^{\ell} = \{(a, b, c) \mid a + b + c = O\}, \tag{3.37}$$

where $O \in \{0, \ldots, O_{max}\}$. Note that, using these ω_i functions the integrands will be various geometric moments of each triangle, which can be efficiently computed by making use of the methods proposed in [106, 107].

Applying these $\{\omega_i\}$ set to Equation (3.36), we get the following integrals over an arbitrary triangle $o = (\mathbf{A}, \mathbf{B}, \mathbf{C})$:

$$\begin{aligned} \int_o \hat{y}_1^{m_i} \hat{y}_2^{n_i} \hat{y}_3^{o_i} \, d\hat{\mathbf{y}} = 2 \operatorname{area}(o) \int_0^1 \int_0^{1-u} & (uA_1 + vB_1 + wC_1)^{m_i} \quad \times \\ & (uA_2 + vB_2 + wC_2)^{n_i} \quad \times \\ & (uA_3 + vB_3 + wC_3)^{o_i} \quad dv\, du, \end{aligned} \tag{3.38}$$

where A_i, B_i and C_i $(i = 1, 2, 3)$ are the corresponding coordinates of the $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ vertices, respectively.

Now let us focus on the integrals on the right-hand side. After expanding the power function and taking the vertex coordinates out of the integral, we get

$$\begin{aligned} & S_{m_i n_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \\ & \sum_{a_1+a_2+a_3=m_i} \sum_{b_1+b_2+b_3=n_i} \sum_{c_1+c_2+c_3=o_i} (a_1, b_1, c_1) A_1^{a_1} A_2^{b_1} A_3^{c_1} \quad \times \\ & \qquad (a_2, b_2, c_2) B_1^{a_2} B_2^{b_2} B_3^{c_2} \quad \times \\ & \qquad (a_3, b_3, c_3) C_1^{a_3} C_2^{b_3} C_3^{c_3} \quad \times \\ & \qquad \int_0^1 \int_0^{1-u} u^{a_1+b_1+c_1} v^{a_2+b_2+c_2} w^{a_3+b_3+c_3} \, dv\, du, \end{aligned} \tag{3.39}$$

and since the last integral depends only on the exponents:

$$\begin{aligned} \int_0^1 \int_0^{1-u} u^{a_1+b_1+c_1} v^{a_2+b_2+c_2} w^{a_3+b_3+c_3} \, dv\, du = \\ \frac{(a_1+b_1+c_1)!(a_2+b_2+c_2)!(a_3+b_3+c_3)!}{(m_i+n_i+o_i+2)!}. \end{aligned} \tag{3.40}$$

Substituting these formulas into Equation (3.39) and rearranging the summations leads to

$$\begin{aligned} & S_{m_i n_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \\ & \frac{m_i! n_i! o_i!}{(m_i+n_i+o_i)!} \sum_{a_1+a_2+a_3=m_i} \sum_{b_1+b_2+b_3=n_i} \sum_{c_1+c_2+c_3=o_i} (a_1, b_1, c_1) A_1^{a_1} A_2^{b_1} A_3^{c_1} \quad \times \\ & \qquad (a_2, b_2, c_2) B_1^{a_2} B_2^{b_2} B_3^{c_2} \quad \times \\ & \qquad (a_3, b_3, c_3) C_1^{a_3} C_2^{b_3} C_3^{c_3}, \end{aligned} \tag{3.41}$$

where $(a, b, c) : \mathbb{N}^3 \to N$ is the trinomial coefficient defined as

$$(a, b, c) = \frac{(a + b + c)!}{a!b!c!}.$$

Summarizing the results from Equation (3.36) and Equation (3.41), the integral over a triangle o can then be written as [107]

$$\int_o \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}} = 2\,\text{area}(o) S_{m_i n_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) \tag{3.42}$$

and thus the system of equations from Equation (3.3) will become

$$\sum_{o=(\mathbf{A},\mathbf{B},\mathbf{C})\in O_\triangle} 2\,\text{area}\,(o) S_{m_i n_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) \approx \sum_{\pi=(\mathbf{D},\mathbf{E},\mathbf{F})\in\varphi(T_\triangle)} 2\,\text{area}\,(\pi) S_{m_i n_i o_i}(\mathbf{D}, \mathbf{E}, \mathbf{F}), \tag{3.43}$$

where $i = 1, \ldots, \ell$.

The computational complexity of the i^{th} integral would be $\mathcal{O}(M_i^9)$, where M_i is the order of the i^{th} power function given by $M_i = n_i + m_i + o_i$. However, the summations in Equation (3.41) can be rearranged to drastically reduce the complexity.

According to [106], let us define the following recursive formulas:

$$C_{ijk}(\mathbf{A}) = (i, j, k) A_1^i A_2^j A_3^k, \tag{3.44}$$

$$D_{abc}(\mathbf{A}, \mathbf{B}) = \sum_{i=0}^{a} \sum_{j=0}^{b} \sum_{k=0}^{c} C_{ijk}(\mathbf{A}) C_{a-i,b-j,c-k}(\mathbf{B}). \tag{3.45}$$

Then Equation (3.41) can be rewritten as

$$S_{m_i n_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \sum_{a=0}^{m_i} \sum_{b=0}^{n_i} \sum_{c=0}^{o_i} C_{abc}(\mathbf{A}) D_{m_i-a,n_i-b,o_i-c}(\mathbf{B}, \mathbf{C}). \tag{3.46}$$

As a result, the complexity reduces to $\mathcal{O}(M_i^6)$ [106]. A further reduction to $\mathcal{O}(M_i^3)$ can be achieved by applying the results of [107] for rearranging Equation (3.44), Equation (3.45), and Equation (3.46) as

$$C_{ijk}(\mathbf{A}) = A_1 C_{i-1,j,k}(\mathbf{A}) + A_2 C_{i,j-1,k}(\mathbf{A}) + A_3 C_{i,j,k-1}(\mathbf{A}), \tag{3.47}$$

$$D_{ijk}(\mathbf{A}, \mathbf{B}) = \begin{cases} 0, & \text{if} \quad l < 0 \quad \text{for any} \quad l \in \{i, j, k\} \\ 1, & \text{if} \quad l = 0 \quad \text{for all} \quad l \in \{i, j, k\} \\ A_1 D_{i-1,j,k}(\mathbf{A}, \mathbf{B}) + A_2 D_{i,j-1,k}(\mathbf{A}, \mathbf{B}) + \\ A_3 D_{i,j,k-1}(\mathbf{A}, \mathbf{B}) + C_{ijk}(\mathbf{B}), \text{ otherwise} \end{cases}, \tag{3.48}$$

$$S_{ijk}(\mathbf{A},\mathbf{B},\mathbf{C}) = \begin{cases} 0, & \text{if} \quad l<0 \quad \text{for some} \quad l \in \{i,j,k\} \\ 1, & \text{if} \quad l=0 \quad \text{for all} \quad\quad l \in \{i,j,k\} \\ A_1 S_{i-1,j,k}(\mathbf{A},\mathbf{B},\mathbf{C}) + A_2 S_{i,j-1,k}(\mathbf{A},\mathbf{B},\mathbf{C}) + \\ A_3 S_{i,j,k-1}(\mathbf{A},\mathbf{B},\mathbf{C}) + D_{ijk}(\mathbf{B},\mathbf{C}), \ \text{otherwise} \end{cases} . \tag{3.49}$$

Since all of the above formulas are recursive, we can store the results of lower order polynomials to compute the higher order ones. Hence the complexity of computing all of the integrals for $i = 1,\dots,\ell$ over one triangular element will be $\mathcal{O}(M^3)$, where M is the maximal degree of polynomials from the $\{\omega_i\}_{i=1}^{\ell}$ set.

Now, let us focus on the approximate calculation of the integrals [Sánta and Kato, 2018][Sánta and Kato, 2016a]. Using the barycentric parametrization from Equation (3.35) we linearly interpolate the $\{\omega_i\}$ functions using the vertices of each triangle $o = (\mathbf{A},\mathbf{B},\mathbf{C})$ as

$$\omega_i(\mathbf{p}) \approx u\omega_i(\mathbf{A}) + v\omega_i(\mathbf{B}) + w\omega_i(\mathbf{C}) \qquad i = 1,\dots,\ell. \tag{3.50}$$

From Equation (3.36) and Equation (3.50) we can derive the following

$$\sum_{o\in O_\triangle} \int_o \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}} \approx \sum_{o\in O_\triangle} 2\,\text{area}(o) \int_0^1 \int_0^{1-u} u\omega_i(\mathbf{A}) + v\omega_i(\mathbf{B}) + w\omega_i(\mathbf{C})\, dv\, du, \tag{3.51}$$

where $w = 1-u-v$ and $i = 1,\dots,\ell$.

Observe that ω_i does not depend on the integration variables in Equation (3.51), hence the integrals can be evaluated

$$\int_o \omega_i(\hat{\mathbf{y}})\, d\hat{\mathbf{y}} \approx \text{area}(o)\frac{\omega_i(\mathbf{A}) + \omega_i(\mathbf{B}) + \omega_i(\mathbf{C})}{3}, \tag{3.52}$$

which then becomes the mean of the function values in the vertices weighted by the area of the triangle. Note that, this formula will be exact for every linear function. For non-linear functions, the precision of approximation depends on the local linearity of the particular function and the size of the triangles [Sánta and Kato, 2018][Sánta and Kato, 2016a].

This method is similar to zero-th order approximation introduced by [106], but that method is approximating the $S_{m_i n_i o_i}$ function, ending up in a different scheme.

The computational complexity of calculating the whole $\{\omega_i\}_{i=1}^{\ell}$ set from Equation (3.9) over one triangle is $\mathcal{O}(M^3)$, where M is the maximal order the polynomials from the set. Observe that, the complexity of the exact and the approximate schemes are equivalent asymptotically. However, there is a difference between the necessary amount of constant operations, which leads to faster algorithm in the approximate case [107].

Computing Volumetric Integrals over Closed Triangular Meshes

When the surfaces $\mathcal{F}_o$ and $\mathcal{F}_t$ are closed, the integrals in Equation (3.3) can also be interpreted as volume integrals, similarly to the voxel-based representation [Sánta and Kato, 2018][Sánta and Kato, 2013a]. From a practical point of view, it is easy to see that such interpretation yields a numerically more stable algorithm: any error on the surfaces $\mathcal{F}_o$ and $\mathcal{F}_t$ will inherently cause an error in the integrals and hence the equality in Equation (3.3) will not be true anymore. The smaller these errors are, the better we approximate true equality and hence we can expect a better performance of the algorithm. Furthermore, we

have seen in the previous section that for our particular set of ω_i functions, the integrals in Equation (3.3) are the moments of $\mathcal{F}_o$ and $\mathcal{F}_t$. Obviously, surface errors will not influence the inner part of an object, hence the moments of complete volumes will be less affected.

Similarly to the previous section, let us assume that the *template* and the *observation* surfaces are approximated by $T_\triangle$ and $O_\triangle \subset \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$ triangular surface meshes, respectively. Moreover, we will use the transformation described in Equation (3.31).

Following [Sánta and Kato, 2018][Sánta and Kato, 2013a], let us denote the volumes enclosed by $T_\triangle$ and $O_\triangle$ by $F_t^\triangle$ and $F_o^\triangle$, respectively. The volume enclosed by the transformed surface $\varphi(\mathcal{F}_t)$ and transformed surface mesh $\varphi(T_\triangle)$ is denoted by $\mathcal{D}_\varphi$ and $F_\varphi^\triangle$, respectively. Using these notations it is obvious that $\mathcal{F}_t \approx F_t^\triangle$ and $\mathcal{F}_o \approx F_o^\triangle$. Using the corresponding volumetric integrals, the integrals of Equation (3.2) can be approximated as

$$\int_{F_o^\triangle} \hat{\mathbf{y}}\, d\hat{\mathbf{y}} \approx \int_{\mathcal{F}_o} \mathbf{y}\, d\mathbf{y}, \tag{3.53}$$

$$\int_{F_\varphi^\triangle} \hat{\mathbf{z}}\, d\hat{\mathbf{z}} \approx \int_{\mathcal{D}_\varphi} \mathbf{z}\, d\mathbf{z}. \tag{3.54}$$

Assuming that every triangle is oriented consistently counter-clockwise when seen from the closest exterior point, the integrals on the right-hand sides of Equation (3.53) and Equation (3.54) can be computed exactly as the sums of signed integrals over properly generated tetrahedrons:

$$\int_{F_o^\triangle} \hat{\mathbf{y}}\, d\hat{\mathbf{y}} = \sum_{o \in O_\triangle} \operatorname{sgn}(\operatorname{vol}(\mathcal{T}_o)) \int_{\mathcal{T}_o} \hat{\mathbf{y}}\, d\hat{\mathbf{y}}, \tag{3.55}$$

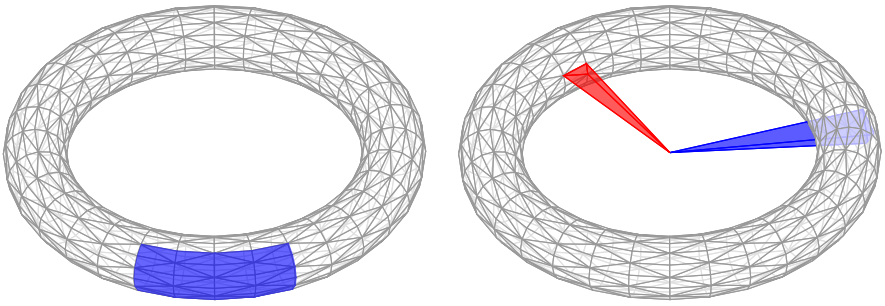
$$\int_{F_\varphi^\triangle} \hat{\mathbf{z}}\, d\hat{\mathbf{z}} = \sum_{\pi \in \varphi(T_\triangle)} \operatorname{sgn}(\operatorname{vol}(\mathcal{T}_\pi)) \int_{\mathcal{T}_\pi} \hat{\mathbf{z}}\, d\hat{\mathbf{z}}. \tag{3.56}$$

The key point here is the creation of the tetrahedrons: For each triangle $o = (\mathbf{A}, \mathbf{B}, \mathbf{C})$, let us create a tetrahedron $\mathcal{T}_o = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{0})$ defined by the vertices of the corresponding triangle o and the origin $\mathbf{0}$. Thus for all triangle, we generate a tetrahedron with the fourth vertex shared by all of these tetrahedrons (see Figure 3.1). Although the choice of the fourth vertex is arbitrary, setting it to the origin will greatly simplify our computations and eventually, the formalism will be similar to the exact integration approach described in Section 3.3.2. The computation of the signed volume of such tetrahedron $\mathcal{T}_o$ is straightforward:

$$\operatorname{vol}(\mathcal{T}_o) = \frac{1}{6} \begin{vmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{vmatrix}, \tag{3.57}$$

where $\mathbf{A} = (A_1, A_2, A_3)$, $\mathbf{B} = (B_1, B_2, B_3)$, $\mathbf{C} = (C_1, C_2, C_3) \in \mathbb{R}^3$ are the vertices of the corresponding triangle o. The sign is important because it describes the orientation of triangles seen from the origin for non-convex shapes. For example in Figure 3.1, there is a volumetric object given by its triangular surface mesh. The volumes of the tetrahedrons generated by the two marked triangles have different signs, caused by the different vertex order [Sánta and Kato, 2018][Sánta and Kato, 2013a].

Then from Equation (3.55) and Equation (3.56), we get the following approximation for



(a) (b)

Figure 3.1: The difference between surface (a) and volumetric (b) integration over the triangular surface mesh of a torus. In (a), the integrals are computed over each triangle, while in (b) the integration domains are tetrahedrons composed of the triangles and an arbitrary chosen common point. In the latter case, the orientation of the triangles is important, because it defines the sign of the volume of the corresponding tetrahedrons. As an example, the blue tetrahedron has positive, and the red has negative volume.

our basic equation Equation (3.2):

$$\sum_{o\in O_\triangle} \operatorname{sgn}(\operatorname{vol}(T_o)) \int_{\mathcal{T}_o} \hat{\mathbf{y}}\, d\hat{\mathbf{y}} \approx \sum_{\pi\in\varphi(T_\triangle)} \operatorname{sgn}(\operatorname{vol}(\mathcal{T}_\pi)) \int_{\mathcal{T}_\pi} \hat{\mathbf{z}}\, d\hat{\mathbf{z}}. \tag{3.58}$$

Applying the $\{\omega_i\}_{i=1}^{\ell}$ set from Equation (3.9), we get the following system of equations:

$$\sum_{o\in O_\triangle} \operatorname{sgn}(\operatorname{vol}(T_o)) \int_{\mathcal{T}_o} \hat{y}_1^{m_i}\hat{y}_2^{n_i}\hat{y}_3^{o_i}\, d\mathbf{y} \approx \sum_{\pi\in\varphi(T_\triangle)} \operatorname{sgn}(\operatorname{vol}(\mathcal{T}_\pi)) \int_{\mathcal{T}_\pi} \hat{z}_1^{m_i}\hat{z}_2^{n_i}\hat{z}_3^{o_i}\, d\mathbf{z}. \tag{3.59}$$

Now, we will derive the exact computation formulas for calculating the integrals for one tetrahedron. Similarly to the previous section, we will describe each point of a tetrahedron in the barycentric coordinate system. Considering a tetrahedron $\mathcal{T}_o$, defined by the triangle $o = (\mathbf{A}, \mathbf{B}, \mathbf{C})$, an arbitrary point $\mathbf{p}$ can be given as [106]

$$p = \rho(u\mathbf{A} + v\mathbf{B} + w\mathbf{C}),$$

where $u, v, w \geq 0$, $u + v + w = 1$ and $0 \leq \rho \leq 1$. Note that, w can be described using u and v as $w = 1 - u - v$. Due to the change of the variables, we have to compensate the change of the volumes in each integral by multiplying the variables with the Jacobian determinant of the coordinate transformation. Herein, the Jacobian will be $J = 6|\operatorname{vol}(\mathcal{T}_o)|\rho^2$ [106].

The integral over a tetrahedron $\mathcal{T}_o$ in the barycentric coordinate system then can be written as [106]

$$\begin{aligned}\int_{\mathcal{T}_o} \hat{y}_1^{m_i}\hat{y}_2^{n_i}\hat{y}_3^{o_i}\, d\hat{\mathbf{y}} = \quad 6|\operatorname{vol}(\mathcal{T}_o)| \int_0^1 \rho^{M_i+2} \int_0^1 \int_0^{1-u} \quad & (uA_1 + vB_1 + wC_1)^{m_i} \times \\ & (uA_2 + vB_2 + wC_2)^{n_i} \times \\ & (uA_3 + vB_3 + wC_3)^{o_i}\, dv\, du\, d\rho,\end{aligned} \tag{3.60}$$

where $M_i = n_i + m_i + o_i$ and $i = 1, \ldots, \ell$. Since $\int_0^1 \rho^{n+2}\, d\rho$ is independent of the rest of the expression, thus it can be evaluated separately, resulting in $\int_0^1 \rho^{n+2}\, d\rho = 1/(M_i + 3)$ [106].

Hence, the integral will be

$$\int_{\mathcal{T}_o} \hat{y}_1^{m_i} \hat{y}_2^{n_i} \hat{y}_3^{o_i} \, d\hat{\mathbf{y}} = \frac{6|\mathrm{vol}(\mathcal{T}_o)|}{M_i + 3} \int_0^1 \int_0^{1-u} \begin{aligned} &(uA_1 + vB_1 + wC_1)^{m_i} \times \\ &(uA_2 + vB_2 + wC_2)^{n_i} \times \\ &(uA_3 + vB_3 + wC_3)^{o_i} \, dv \, du. \end{aligned} \tag{3.61}$$

Observe that on the right-hand side we got the same integrals as in Equation (3.38), thus

$$\int_{\mathcal{T}_o} \hat{y}_1^{m_i} \hat{y}_2^{n_i} \hat{y}_3^{o_i} \, d\hat{\mathbf{y}} = \frac{6|\mathrm{vol}(\mathcal{T}_o)|}{M_i + 3} S_{n_i m_i o_i}(\mathbf{A}, \mathbf{B}, \mathbf{C}), \tag{3.62}$$

where $S_{n_i m_i o_i}$ can be calculated using the recursive formulas from Equations (3.47)–(3.49). Hence, the computational complexity for the whole set will be $\mathcal{O}(M^3)$, where M is the maximal degree of polynomials from the $\{\omega_i\}$ set.

Similarly to the exact surface integration approach, the volumetric scheme also provides the exact integrals over each tetrahedron. Moreover, it is possible to derive an approximate scheme for the volumetric case too. However, the applied naive tetrahedron generation gives much larger integration domains than the triangles used in the surface integration case, which leads to higher interpolation errors. Note that, a more accurate approximation could be achieved using a more sophisticated tetrahedron generation method, but it also increases the number of tetrahedrons along with the computational time. Therefore, in practice, it does not worth to use the approximate volumetric scheme.

3.4 Numerical Implementation

Let us continue by analyzing the numerical aspects of the proposed framework. As we observed above, representing the continuous $\mathcal{F}_t$ and $\mathcal{F}_o$ by either a set of voxels or by triangular surface meshes, will lead to discrete approximations only. Hence, the integrals from Equation (3.3) will be approximately valid as well. In a real application, the objects are subject to various segmentation errors and image acquisition artifacts, which again could introduce errors into the systems. Therefore, we construct overdetermined systems, *i.e.* $\ell > N$ in Equation (3.3), which are solved in the least squares sense via a *Levenberg-Marquardt* algorithm.

Since the problem is solved by minimizing the algebraic error, proper normalization is critical for numerical stability. For that purpose, the *template* and *observation* coordinates are normalized into the $[-0.5, 0.5]$ interval by applying normalizing transformations N_o and N_t,

respectively:

$$N_o = \begin{pmatrix} o_1 & 0 & 0 & -o_1 O_1 \\ 0 & o_2 & 0 & -o_2 O_2 \\ 0 & 0 & o_3 & -o_3 O_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.63}$$

$$N_t = \begin{pmatrix} t_1 & 0 & 0 & -t_1 T_1 \\ 0 & t_2 & 0 & -t_2 T_2 \\ 0 & 0 & t_3 & -t_3 T_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.64}$$

where T_i, O_i $(i = 1, \ldots, 3)$ are the centroids of the template and the observation, respectively and t_i, o_i $(i = 1, \ldots, 3)$ are the corresponding scale factors. This normalization will basically transform the input objects into a unit cube centered at the origin.

The resulting φ^* transformation acts between the normalized objects, hence we have to denormalize it. In our experiments, we register the *template* image to the *observation*:

$$\begin{array}{ccc} \mathcal{F}_t & \xrightarrow{\varphi} & \mathcal{F}_o \\ \downarrow & & \downarrow \\ N_t(\mathcal{F}_t) & \xrightarrow{\varphi^*} & N_o(\mathcal{F}_o) \end{array} \tag{3.65}$$

therefore to obtain φ from φ^*, we have to use N_t and the inverse of N_o:

$$N_o^{-1} = \begin{pmatrix} 1/o_1 & 0 & 0 & O_1 \\ 0 & 1/o_2 & 0 & O_2 \\ 0 & 0 & 1/o_3 & O_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.66}$$

For linear transformations, denormalization is straightforward, hence we only have to multiply each side of the estimated matrix with the particular normalizer or denormalizer transformation. In non-linear cases, however, the final transformation will be a newly composed function in the form of

$$\varphi = N_o^{-1} \circ \varphi^* \circ N_t. \tag{3.67}$$

The range of the ω_i functions also should be normalized into $[-1, 1]$, in order to ensure a balanced contribution of the equations to the algebraic error. In [7], this is achieved by dividing the equations with the maximal magnitude of each integral over the zero-centered unit circle. Analogously, our experiments showed that the input objects are never transformed out the zero-centered unit sphere [Sánta and Kato, 2018][Sánta and Kato, 2013a], thus we can use this domain to determine the maximal magnitude integrals for volumetric images. For the surface integration based method, the object having the largest possible surface area with respect to the objects used in our experiments is also happens to be the unit sphere. Therefore our integrals have been divided by the following constants:

$$N_i = \int_S |\omega_i(\mathbf{x})| \, d\mathbf{x}, \tag{3.68}$$

where S domain will be the zero-centered unit sphere for the volumetric and the surface of the same sphere for the surface approaches. These integrals are computed by making use of

the proposed numerical schemes from Section 3.3.

3.4.1 Voxel Representation

Combining the general voxel-based system from Equation (3.21) with the normalizations above leads to the following system

$$\frac{|N_o|}{N_i} \sum_{\mathbf{Y} \in \hat{F}_o} \omega_i(\mathbf{Y}) = \frac{|N_t|}{N_i} \sum_{\mathbf{X} \in \hat{F}_t} \omega(\pi(\mathbf{X}))|J_\pi(\mathbf{X})| \tag{3.69}$$

where $\hat{F}_o$ and $\hat{F}_t$ are sets of the normalized voxel coordinates of the *observation* and *template* respectively, and $|N_o|$ and $|N_t|$ are the determinants of the corresponding normalizing transformations. Applying the proposed normalizations to the polynomial system described in Section 3.3.1 results in a similar formula. The outline of the algorithm can be found in Algorithm 3.1. For the sake of simplicity, this algorithm will be referred to as `voxel` in the following. Moreover, when we use the polynomial framework described in Section 3.3.1, we will use `voxel_poly` notation.

Algorithm 3.1 Pseudo code of the voxel-based algorithm

Input: *template* and *observation* voxels
Output: The transformation parameters of φ

1: Choose a set of $\ell > k$ non-linear functions $\{\omega_i\}_{i=1}^{\ell}$, and for each ω_i compute the normalizing constant N_i using Equation (3.68).
2: Compute the normalizing transformations N_o and N_t as in Equation (3.63) and Equation (3.64), respectively, which maps voxel coordinates into $[-0.5, 0.5]$.
3: Find a least squares solution of the system from Equation (3.69) using the *Levenberg-Marquardt* algorithm. Use the identity transformation for initialization.
4: Denormalize the solution φ^* using Equation (3.67), which provides the parameters of the aligning transformation.

3.4.2 Triangular Surface Mesh Based Algorithm

Now let us observe how the normalizations are acting on the systems of equations when we are dealing with triangular surfaces. Using the volumetric integration from Section 3.3.2, we get the following approximation of our system:

$$\frac{1}{N_i} \sum_{o \in N_o(O_\triangle)} \operatorname{sgn}(\operatorname{vol}(T_o)) \int_{\mathcal{T}_o} y_1^{m_i} y_2^{n_i} y_3^{o_i} \, d\mathbf{y} \approx \frac{1}{N_i} \sum_{\pi \in \varphi(N_t(T_\triangle))} \operatorname{sgn}(\operatorname{vol}(\mathcal{T}_\pi)) \int_{\mathcal{T}_\pi} z_1^{m_i} z_2^{n_i} z_3^{o_i} \, d\mathbf{z}, \tag{3.70}$$

where $i = 1 \dots \ell$ and the integrals over tetrahedrons can be computed via Equation (3.62).

With the surface integration approach from Section 3.3.2, we get a similar system

$$\frac{1}{N_i} \sum_{o \in N_o(O_\triangle)} \int_o y_1^{m_i} y_2^{n_i} y_3^{o_i} \, d\mathbf{y} \approx \frac{1}{N_i} \sum_{\pi \in \varphi(N_t(T_\triangle))} \int_\pi z_1^{m_i} z_2^{n_i} z_3^{o_i} \, d\mathbf{z}, \tag{3.71}$$

where $i = 1, \dots, \ell$ and the surface integrals over the triangles given by Equation (3.42).

The pseudo code of the triangular surface based algorithm can be found in Algorithm 3.2. In order to distinguish between the possible integration schemes, volumetric (based on Equa-

tion (3.70)), surface (built upon of Equation (3.71)) and approximate surface integration (using Equation (3.52)) approaches will be referred to as `trisurf_vol`, `trisurf_surf` and `trisurf_surf-approx`, respectively.

Algorithm 3.2 Pseudo code of the triangular surface based algorithm

Input: *template* and *observation* triangular surface meshes
Output: The transformation parameters of φ

1: Choose a $\{\omega_i\}_{i=1}^{\ell}$ set of power functions described in Equation (3.9), such that $\ell > N$ and for each ω_i compute the normalizing constant N_i using Equation (3.68).
2: Compute the normalizing transformations N_t and N_o which maps vertex coordinates into $[-0.5, 0.5]$.
3: Construct the system of equations Equation (3.71) or Equation (3.70) for surface or volumetric schemes, respectively. The integrals could be determined exactly using the recursive formulas from Equations (3.47)–(3.49) or approximated in the surface case by Equation (3.52).
4: Find a least squares solution of the system using the *Levenberg-Marquardt* algorithm initialized with the identity transformation.
5: Denormalizing the solution gives the parameters of the aligning transformation.

3.5 Experimental Results

In the following section, we will summarize our experimental results with the proposed approaches on multiple synthetic and real datasets. On volumetric images the registration error has been quantitatively evaluated based on a Dice coefficient inspired metric:

$$\delta = \frac{|F_r \triangle F_o|}{|F_r| + |F_o|} \cdot 100\%, \tag{3.72}$$

where F_o and F_r denote the set of foreground voxels of the *observation* and *registered* objects respectively. We observed that $\delta < 10\%$ corresponds to visually acceptable alignment.

This metric describes the accuracy of the alignment well when volumetric objects are available. These objects, however, could not be obtained when we work with open surfaces. In these cases, we used two different metrics in a complementary manner to describe the differences between two open surfaces. The first metric is the normalized difference of the surface areas of the objects, the second is the maximal root mean square (RMS) distance between the closest points of the triangular meshes, denoted by D_{area} and D_{RMS}, respectively:

$$D_{area} = \frac{|\operatorname{area}(O_\triangle) - \operatorname{area}(\varphi(T_\triangle))|}{|\operatorname{area} O_\triangle|} \cdot 100\% \tag{3.73}$$

$$D_{RMS} = \max\{RMS(O_\triangle, \varphi(T_\triangle)), RMS(\varphi(T_\triangle), O_\triangle)\}, \tag{3.74}$$

where

$$RMS(S_1, S_2) = \sqrt{\frac{1}{|V(S_1)|} \sum_{\mathbf{p} \in V(S_1)} \inf_{\mathbf{q} \in S_2} \|\mathbf{p} - \mathbf{q}\|^2},$$

and S_1, S_2 are the corresponding surfaces, while $V(S_1)$ denotes the set of all vertices of S_1. The RMS function basically estimates the distance between each vertex of the first triangular surface and the closest (not necessarily vertex) point of the second triangular surface. This measure is more accurate than the simple vertex-wise distances and taking the maximum of

the values computed with swapped arguments leads to a symmetric measure between the surfaces.

Note that, neither of the last two metrics could be used *per se* because each one has its own drawback. For example, using only the D_{RMS} measure leads to accepting extreme solutions like transforming the whole *template* object into one triangle of the *observation*. Similarly, the D_{area} measure will be minimal for all test cases where the physical deformation preserves the surface area of the input object.

The algorithms have been implemented in `C++` using the *Levenberg-Marquardt* implementation of Lourakis [108]. All tests were run on a PC with Core i5 3.1 GHz architecture.

3.5.1 Synthetic Tests on Volumetric Data

In our first block of experiments, we will focus on the alignment of volumetric data. In order to quantitatively evaluate the performance of the proposed methods, we used large, synthetically generated volumetric datasets.

The registered objects, as well as synthetic observations, were generated by the following procedure (in Matlab): First, a smooth triangular mesh has been created from the object's surface using Matlab's internal `isosurface` function. Then the transformation was applied to the vertices yielding a transformed triangular mesh, from which the final voxelized object was obtained by the `binvox` program available from `http://www.patrickmin.com/binvox/` [109]. Following this process, two polynomial and three thin plate splines based datasets were generated containing observations of size $0.1 - 2.5$ million voxels.

Polynomial Dataset

In the first block of experiments, we will present our results on the polynomial dataset. The dataset was randomly generated using second and third order polynomial deformations, which were applied to different *template* objects. The transformation parameters were randomly picked from the following intervals: $a_{11}, b_{21}, c_{31} \in [0.5; 1.5]$, $a_{21}, a_{31}, b_{11}, b_{31}, c_{11}, c_{21} \in [-0.25; 0.25]$ and all other parameters are from $[-0.5; 0.5]$. Note that, $a_{00} = b_{00} = c_{00} = 0$ (*i.e.* no translations), because initial normalization would remove any larger translations. The overall size of the generated dataset is 500. Topology preservation was another important requirement of the generated transformation. In order to achieve this, diffeomorphic transformations were generated by constraining the Jacobian of the generated transformation to be positive everywhere over the objects [Sánta and Kato, 2018][Sánta and Kato, 2013a].

For these experiments, we used second and third order polynomial models from Equation (3.10) (*i.e.* $d = 2, 3$), yielding 30 and 60 parameters, respectively [Sánta and Kato, 2012b]. The experiments were made by making use of Algorithm 3.2. On the whole polynomial dataset, using third order transformation model `voxel` approach has achieved good results, having 7.02% average alignment accuracy measured by Equation (3.72). The average runtime was 13.96 minutes. Sample results can be found in Figure 3.2.

Experiments have been made with `voxel_poly` as well, where we used a second order polynomial model as described in Section 3.3.1. The necessary equations were generated by a simple Maple program, which also exported them to `C++` code. Then, the generated code was integrated into the implemented framework. As expected, the separation reduced the runtime of the algorithm by one order of magnitude, but, since it estimates the same

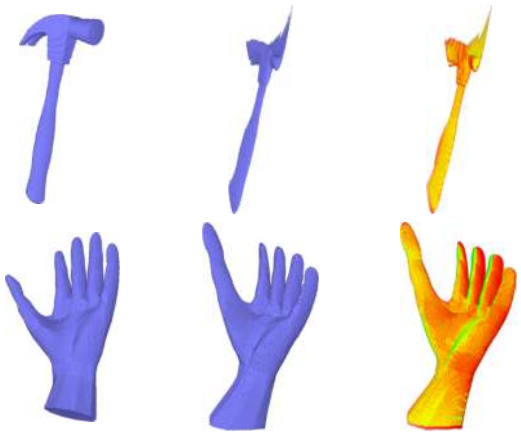


Figure 3.2: Sample results from the polynomial synthetic dataset. The first two columns show the *templates* and the *observations*, respectively, and in the last column, we present the overlapping voxels of the *registered* objects in yellow and the non-overlapping ones in red and green.

equations as the naive volumetric approach, the alignment accuracy remained the same. The outline of this experiment can be found in Table 3.1.

Method	Runtime (sec)			$\delta(\%)$		
	m	μ	σ	m	μ	σ
`voxel_poly`	9	10.09	4.21	7.41	7.92	4.3
`voxel`	105	124.07	103.19			

Table 3.1: Comparison between `voxel_poly` and `voxel` on the polynomial dataset. While maintaining the same accuracy, `voxel_poly` provided results are 10 times faster than `voxel`.

Thin Plate Spline Dataset

In this section, we will focus on solving the registration problem with Algorithm 3.2 on the synthetic TPS dataset. For this dataset 750 observations were generated based on thin plate splines with 16, 32, and 64 control points placed on the corresponding grids, yielding three different subsets with various degrees of freedom. Since the number of control points is proportional to the degrees of freedom, we will refer to each subset by denoting the number of control points used for the generation. The parameters of the synthetic deformations were obtained by applying random translations with elements from $[-0.2, 0.2]$ to the control point positions. The resulting point correspondences were used to construct interpolating thin plate splines, which were applied to different *template* objects.

In the first experiment, we observed the sensitivity of the proposed approaches with respect to the mesh resolution [Sánta and Kato, 2018][Sánta and Kato, 2013a]. In this experiment, we used the surface extractor algorithms from the CGAL library [63, 110]. This library implements a *Delaunay refinement* algorithm, giving the opportunity to construct meshes with different qualities and resolutions [110, 111]. Herein, the resolution of the triangular meshes is controlled by the maximal radius r of the corresponding Delaunay sphere for each triangle [Sánta and Kato, 2018][Sánta and Kato, 2013a]. Using our synthetic dataset, triangular meshes have been created for $r \in \{1, 3, 5, 10\}$ and the registration problem has been solved by the `trisurf_vol` algorithm. The performance has been quantitatively

evaluated by the δ measure from Equation (3.72). The results are presented in Figure 3.4.

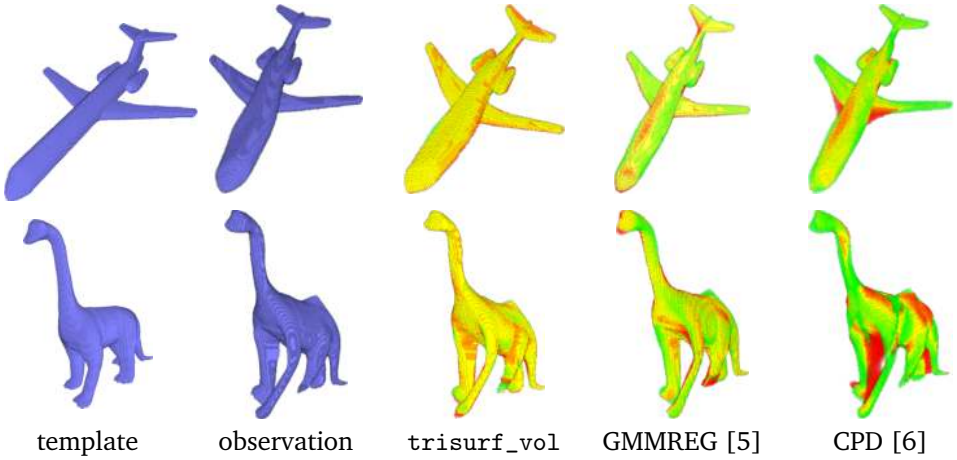


Figure 3.3: Sample results from the synthetic TPS database. The first two columns show the *templates* and the *observations*, respectively, and in the other columns, we present the overlapping voxels of the *registered* objects in yellow and the non-overlapping ones in red and green. For GMMREG [5] and CPD [6], we present the best results for each particular test case.

Obviously, the resolution of the triangular mesh affects the computational time. Hence, the computational time could be significantly reduced by decreasing r at the price of a slightly lower registration accuracy. We found that $r = 5$ has the best quality over time ratio on our synthetic dataset [Sánta and Kato, 2018][Sánta and Kato, 2013a].

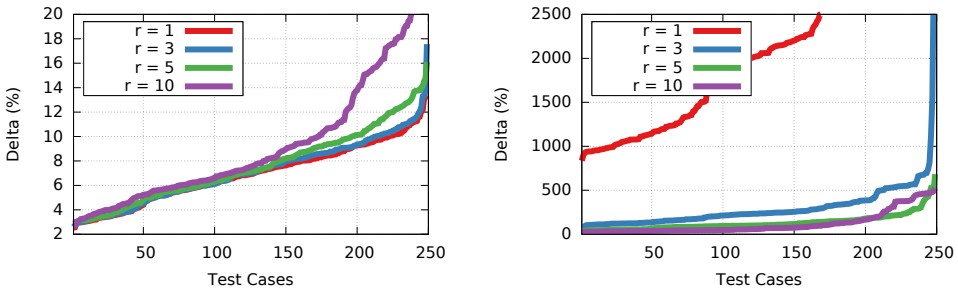


Figure 3.4: Comparison of registration error and computing time with various r values for the Delaunay sphere in surface mesh extraction.

In the next experiment, the registration problem has been solved for all test cases by making use of the `trisurf_vol` and `trisurf_surf` approaches. We used multiple control point configurations with 16, 32 and 64 points generated by the proposed control point placement strategies from Section 3.2.2. As we stated in Chapter 3.4, the runtime is proportional to the number of ω functions used for our system, which based on the number of parameters of the transformation model. For the current tests, we used the first 89, 120 and 220 power functions from the set $\omega_i(\mathbf{x}) = x_1^{m_i} x_2^{n_i} x_3^{o_i}$, where $(m_i, n_i, o_i)_{i=1}^{220} = \{(a, b, c) \mid a + b + c = O, O = 1, \ldots, 9\}$.

Our results on each synthetic subset are presented in Figure 3.5 and the best results on the whole dataset have been summarized in Table 3.2. In order to show the categorized results in Figure 3.5, the median of the estimated values have been taken for each synthetic subset and beside the registration accuracy, the running times have been presented as well.

Let us start our investigation by focusing on the results of `trisurf_vol` first by observing

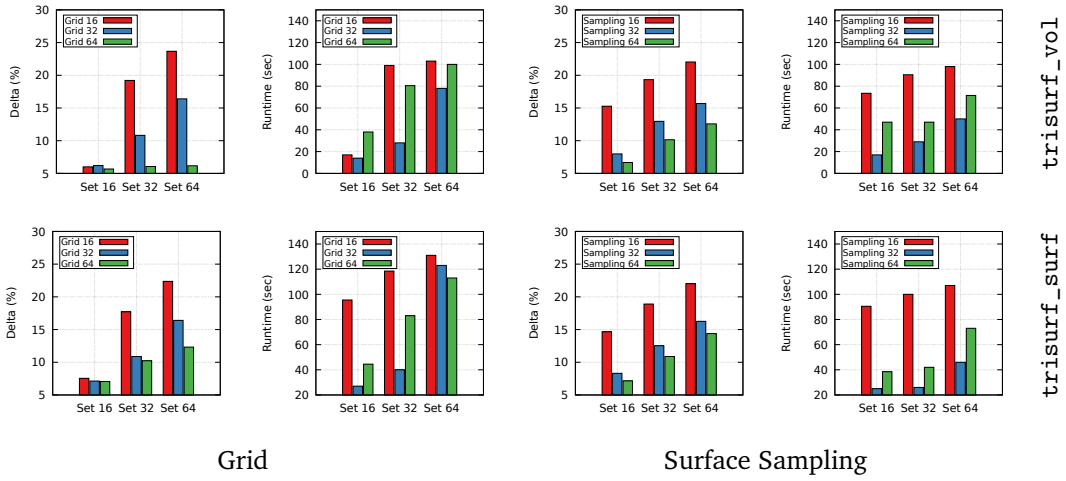


Figure 3.5: Results on the volumetric synthetic dataset. On these plots we show the median of the registration results with respect to the δ-measure from Equation (3.72) and the running times for each subset categorized by the number of control points used for the generation. The rows contain results obtained by `trisurf_vol` and by `trisurf_surf`, respectively. For each row, the first two columns present the results using the grid control point placement strategy, while the results of the second two columns are obtained by using the surface sampling based strategy (see Section 3.2.2 for more details).

the statistics presented in the first row of Figure 3.5. The tests have confirmed our preliminary assumptions on control point placement strategies as the grid based strategy outperformed the sampling approach when we used at least the same number of control points for the registration as we used for the test case generation. This effect is caused by the fact that we used a similar strategy for the synthetic generation. However, the surface sampling strategy seems to handle the underfitting problem more accurately, although it still does not give visually acceptable results overall. It is true for both strategies that the accuracy benefits from the overfitting of the model, but note that in the current experiment the dataset has not been affected by any type of noise. Moreover, for both strategies, the models containing 64 control points have achieved the best results statistically. Observe that the grid model with 64 control points seems to be independent from the degrees of freedom of the underlying deformation in terms of alignment accuracy.

Observing the running times yields some interesting findings, too. As we mentioned above, the runtime of the algorithm is mostly affected by the number of determinable parameters, which also defines the necessary number of ω functions. At the same time, we experienced that increasing the number of control points also makes the estimation process more stable, hence decreases the number of necessary iteration steps of the *Levenberg-Marquardt* solver. As an example, examine that any configuration with 32 control points has lower running time than the results with 16 points. Moreover, while the 64 grid configuration achieved the same alignment accuracy on each set, the running time was increasing with the complexity of the corresponding set. Also, note that the surface sampling strategy achieved smaller running times than the corresponding grid based case.

Now, let us continue with the results of `trisurf_surf` presented in the second row of Figure 3.5. While most of our findings from the volumetric case can be applied to the surface approach, we have to mention that the volumetric approach outperformed the surface based algorithm in the accuracy of the alignments. This is caused by the stability issues we have mentioned in Section 3.1, which also explains the higher values in the running

times. Furthermore, there is no significant difference between the control point placement strategies in terms of alignment accuracy, although the surface sampling approach achieved better running times.

Method	Runtime (min)			$\delta(\%)$		
	m	μ	σ	m	μ	σ
`trisurf_vol` Grid	1.27	1.52	1.24	**5.98**	**6.88**	5.09
`trisurf_vol` Sampling	**0.87**	**1.23**	**1.04**	8.96	9.45	**3.49**
`trisurf_surf` Grid	1.37	1.83	1.76	**9.18**	**9.62**	**3.64**
`trisurf_surf` Sampling	**0.72**	**1.09**	**0.98**	9.80	10.38	4.12
`voxel` Grid	31.42	59.77	67.74	7.19	7.05	2.3

Table 3.2: Results on 750 synthetic images using 64 control points for each configuration (m – median, μ – mean and σ – standard deviation).

In the last line of Table 3.2, we compared the registration quality and computing times of the `voxel` and `trisurf_vol` algorithms for $r = 5$. The mesh-based algorithm has outperformed the voxel-based approach both in terms of alignment accuracy and computational time. This latter result is not surprising, as the mesh-based algorithm works only with triangle vertices whose number is less than 9000, whereas the voxel-based method has to deal with several hundred thousand voxels (see also Figure 3.7).

The difference between the alignment accuracy, however, needs further elaboration. While both methods based on the volumetric scheme, there is a difference between the way they approximate the continuous integrals in Equation (3.21) and Equation (3.34). In the voxel-based case, approximation error is due to 1) the discretization error on the object's surface (inner voxels do not produce such errors) and 2) the Jacobian is implicitly assumed to be constant within each voxel (including the inner ones!). However, the mesh-based algorithm computes the exact (continuous) integrals over each tetrahedron, therefore the only source of the approximation error is the difference between the true object surface and its approximating triangular mesh.

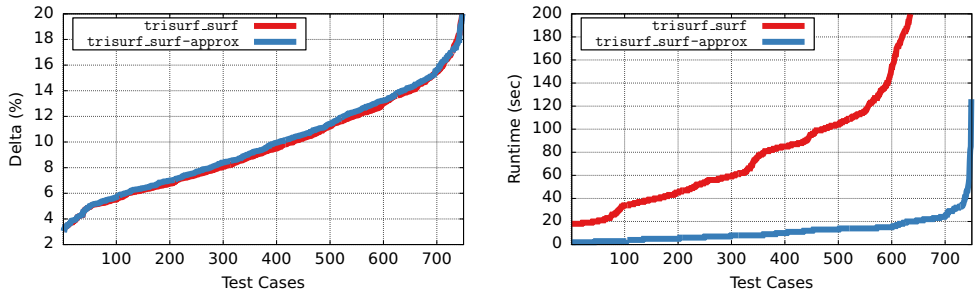


Figure 3.6: Comparison between the two surface based approaches on the volumetric dataset. The `trisurf_surf` algorithm achieves slightly better results on the alignment accuracy, but the `trisurf_surf-approx` approach has much lower runtime.

We have also compared the two proposed surface integration based approaches, the exact (`trisurf_surf`) and the approximate approach (`trisurf_surf-approx`) on the volumetric dataset and we presented the results in Figure 3.6. Fulfilling the expectations, the approximate approach gives a result almost 10 times faster than the exact algorithm. Moreover, the differences between the accuracy remain low for the whole dataset.

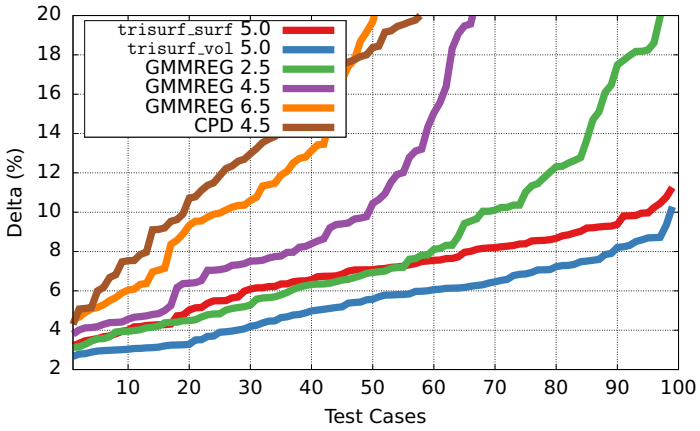


Figure 3.7: Comparison of the best runs with the surface based and volumetric approaches with GMMREG [5] and CPD [6]. The number at end of each plot's name denotes the corresponding r value used for triangular surface generation.

In the last experiment of the current block, we have compared our results to the point-based registration frameworks presented in [5] (GMMREG) and [6] (CPD) on a smaller synthetic subset. We used the `C++` implementation of these methods available from `http://code.google.com/p/gmmreg` and set the parameters to their default values (within the given Matlab framework). The input of these algorithms were the vertices of the extracted triangular surface meshes, using the same extraction technique as in the previous tests. For the best GMMREG set both the average and median runtimes were 30 minutes, for the lower resolutions the average and median runtimes were 3 minutes. In Figure 3.7, quantitative results for $r \in \{2.5, 4.5, 6.5\}$ indicate that these methods provide inferior alignments to the volumetric approach and similar, but less stable results compared to the surface based algorithm. Sample results presented in Figure 3.3.

3.5.2 Synthetic Tests on Surface Data

In the next block of experiments, we will focus on registering open triangular surfaces. For these tests, an artificially cropped dataset has been created from the synthetically generated volumetric data. Each *template* object has been cut into two pieces using randomly generated 3D planes, keeping only the vertices on the positive sides of each plane. Using the volumetric dataset, we transformed the obtained *templates* by the original transformation, in order to crop the corresponding parts from the *observations* too. This procedure ensures that the produced *observation* will be the transformed *template*, without any larger additional or missing parts. The triangular meshes were generated using the $r = 5$ configuration. For examples, see Figure 3.8.

In order to register these open surfaces, the `trisurf_surf` and `trisurf_surf-approx` approaches have been used and experiments have been made with the grid and the surface sampling strategies as well. Unfortunately, we have lost the exact degrees of freedom of the obtained dataset due to the slicing, thus we will use TPS models with 64 control points for all tests. Some of our results could be found in Figure 3.8.

In order to quantitatively evaluate the results, we used the D_{area} and the D_{RMS} metrics from Equation (3.73) and Equation (3.74), respectively. Our experiments have confirmed that if the D_{area} is below 1.5% and the D_{RMS} is below 2 voxels the determined transformation gives a visually good alignment. Furthermore, if the D_{area} is below 2% and the D_{RMS}

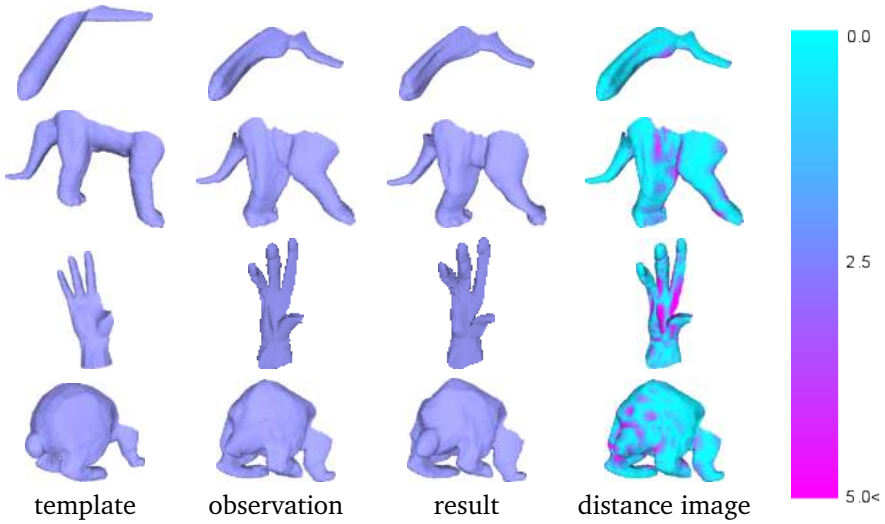


Figure 3.8: Some examples from the artificially cropped dataset and the solution given by the surface-based approach. The first three columns show the *template*, the *observation*, and the transformed *template*, respectively. In the fourth column, each point of the *observation* colored based on the distance from the closest point of the transformed *template*. Note that, the values are expressed in voxels.

is below 2.5 voxels the solution is visually acceptable.

On this dataset, the `trisurf_surf` method has achieved better results using the surface sampling control point placement strategy than using the grid based approach. In the former case, 66% of visually good and 83.33% of visually acceptable solutions have been provided from the 750 test cases. In the latter case, the proposed approach achieved visually good transformations on 62.67% and visually acceptable alignments on 77.87% of the whole dataset. We have evaluated the approximate algorithm as well, which achieved slightly better results on the alignment accuracy (87.73% of visually acceptable and 67.74% of visually good alignments) and reduced the computational time significantly from an average 127 seconds to 17 seconds. The quantitative results using the surface sampling strategy presented in Figure 3.9.

Our best results have been compared to the point-based frameworks on a smaller subset, too. The proposed approach outperformed the GMMREG [5] and CPD [6] algorithms on the alignment accuracy and running times. Note that, for these experiments, we used the same triangular surface meshes generated by using $r = 5$ as the maximal radius of a Delaunay sphere for each triangle. The results of this experiment summarized in Table 3.3.

3.5.3 Robustness Against Noise

In practice, segmentation never produces perfect shapes, therefore robustness against segmentation errors also was evaluated on simulated volumetric data: we randomly added or removed squares uniformly around the boundary of each slice of the observations yielding surface error of 10%, 20% and 30% of the original object volume (see sample slices in Figure 3.10). In the first experiment, the *templates* have been registered to the noisy *observations* using `trisurf_vol` and `trisurf_surf-approx` using TPS model with the same control point positions as we used for the dataset generation. The plots in Figure 3.10 show the quantitative evaluation of the alignment error δ on 125 objects estimated between the registration results and the original *observations*. The results confirm the statements from

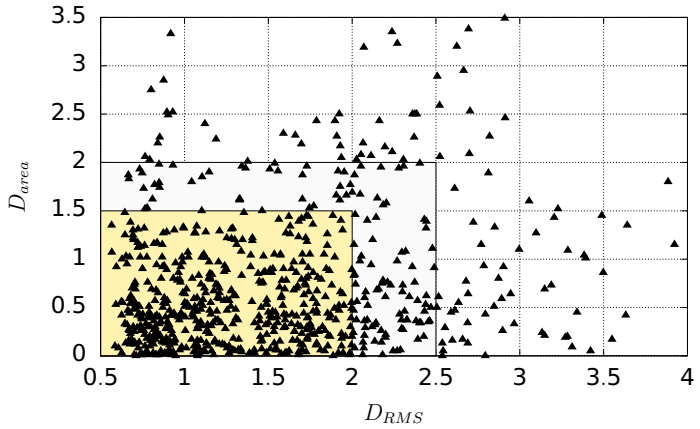


Figure 3.9: Synthetic results on the open surface dataset using the approximate surface integration approach. The control points have been placed using the *Farthest Point Sampling* strategy on the *template* surfaces. In the diagram, the D_{area} metric (y-axis) plotted against the D_{RMS} metric (x-axis). The areas denoted by the yellow and gray rectangles meaning the visually good and the visually acceptable results, respectively. From the 750 test cases, 66% are visually good and 83.33% are visually acceptable solutions.

		GMMREG	CPD	`trisurf_surf-approx`
D_{RMS}	m	2.11	1.76	**1.26**
	μ	7.12	2.49	**1.20**
	σ	13.65	1.59	**0.5**
$D_{area}(\%)$	m	2.52	7.23	**0.6**
	μ	16.87	8.62	**0.44**
	σ	45.95	6.06	**0.52**
Runtime (sec)	m	13.93	19.53	**10**
	μ	38.1	57.48	**12.92**
	σ	37.67	57.66	**13.41**

Table 3.3: Comparison of the surface based algorithm with the GMMREG and the CPD methods on the open surface dataset (m – median, μ – mean and σ – standard deviation). We used the approximate computation scheme with the surface sampling control point placement strategy of 64 points. The proposed approach outperformed the other methods in the terms of D_{RMS} and D_{area} metrics and achieved better running times. Note that, for these experiments, we used the same triangular surface meshes generated using the $r = 5$ configuration.

Section 3.1 as while the volumetric approach is quite robust against the segmentation errors up to as high as 20% of the object volume, the surface integration approach gives poor results for even 5% of surface errors.

In our volumetric experiments, we claimed that the proposed approach benefits from the cases when the approximating transformation has higher degrees of freedom than the underlying deformation, *i.e.* the case of model overfitting. Although this statement is true for noiseless cases, there are several situations when the overfitting has a negative effect on the outcome of the algorithm.

One of these cases is the problem of occlusions. Just as every area based approach, none of the proposed approaches could handle occlusions well. Essentially, the algorithms will find a TPS, which aligns the *template* with the occluded *observation*, since usually the degrees of freedom is high enough to handle the missing parts. As an example for this issue, see Figure 3.11, where the *template* is aligned to an artificially cropped observation.

In the next experiment, we are investigating the effect of overfitting when surface errors

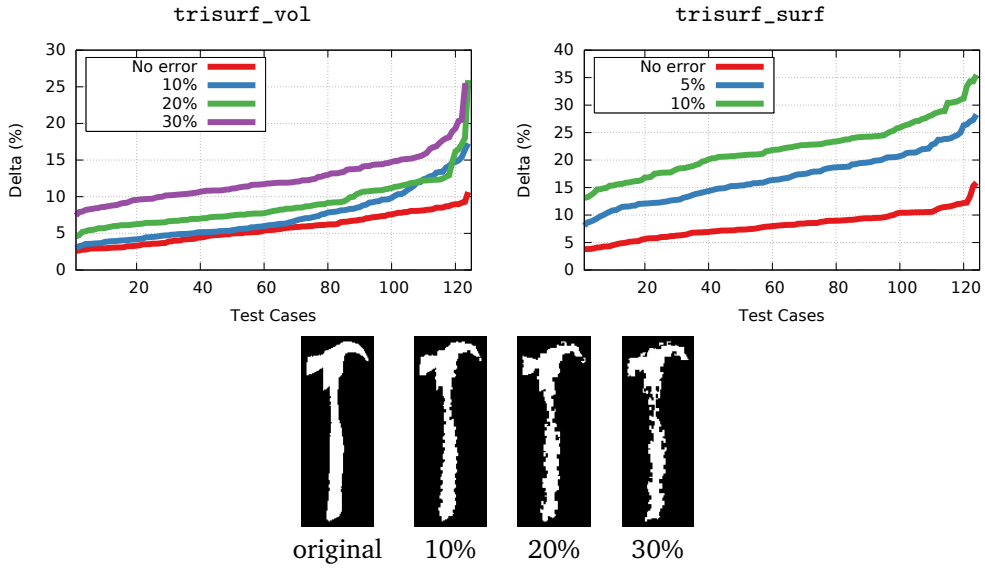


Figure 3.10: Robustness test results for various degree of synthetically generated surface errors. For each test, samples of surface errors on the same voxel slice are shown.

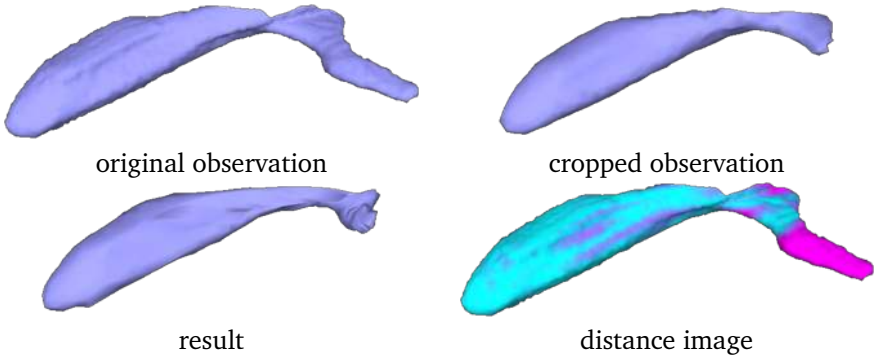


Figure 3.11: The alignment to an artificially cropped observation. The solution is on the boundary of visual acceptability if considering the cropped observation, although it is very far from the original one.

are available. The problem has been solved using a TPS model built from 64 control points placed on a uniform grid. The results of this experiment are presented in the first diagram of Figure 3.12. While in the non-overfitting cases the algorithm was robust against the surface noise as long as it does not exceeds 20%, in the presence of overfitting this border moves down to 10%. In order to overcome this problem, a regularizer term, more precisely the *bending energy* of the TPS model from Equation (3.17), has been added to the algebraic error of the system. The λ parameter of then *bending energy* term has been set to 10^{-6}. The gain caused by the regularization in the noise tolerance is around 5%.

3.5.4 Medical Applications

Surface registration has several possible application areas in the medical field [4, 98]. The input images could be subject to stronger non-rigid deformations due to the motion of the patients, in addition, the registration across incompatible modalities could also raise difficulties. In the following, we will introduce two different applications from the medical field. First, we will show the capabilities in registering deformable lung CT scans, then

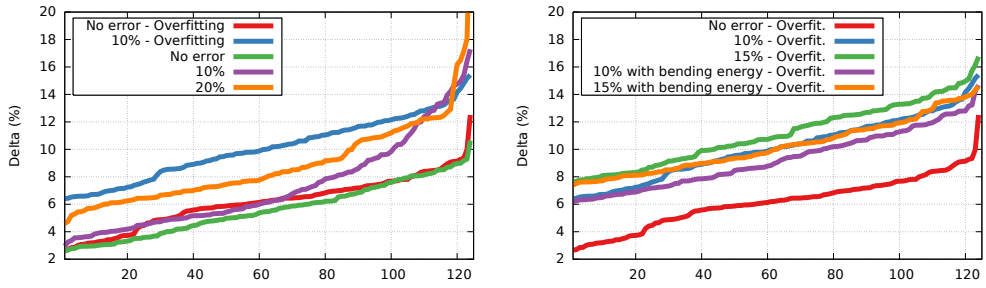


Figure 3.12: The quantitative results in the presence of surface errors and overfitting. On the bottom diagram, the overfitting problem has been partially handled by minimizing the bending energy of the TPS model.

we present an experiment on inter-patient registration of the pial surfaces of human brains extracted from MRI images.

Lung Alignment

Lung alignment is a crucial task in lung cancer diagnosis [112]. During the treatment, changes in the tumor size are determined by comparing *follow-up* PET/CT scans which are taken at regular intervals depending on the treatment and the size of the tumor. Due to respiratory motion, the lung is subject to a nonlinear deformation between such *follow-up*s, hence it is hard to automatically find correspondences. A common practice is to determine corresponding regions by hand, but this makes the procedure time consuming and the obtained alignments may not be accurate enough for measuring changes [Sánta and Kato, 2018][Sánta and Kato, 2013a].

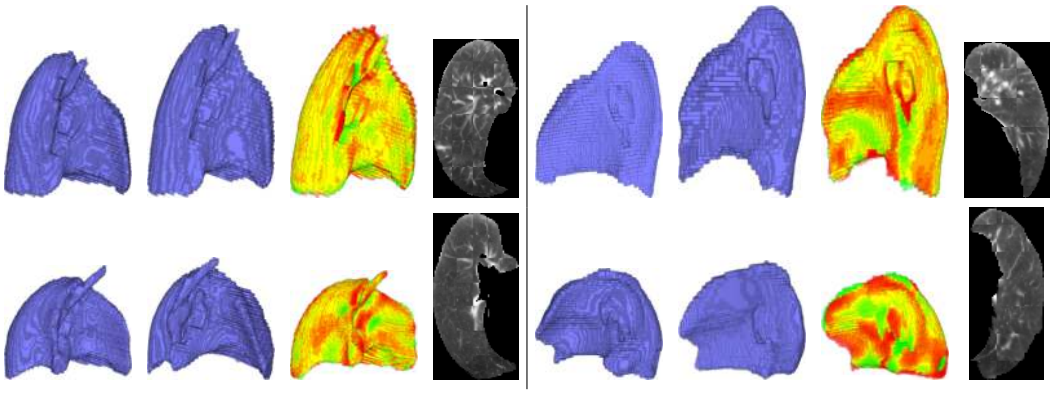


Figure 3.13: Alignment of lung CT volumes. For each block, we present the *template*, the *observation*, the overlaid result image and a sample slice combining the grayscale values of the original and the transformed image as a checkerboard pattern, respectively. Segmented 3D lung images were generated by the *InterView Fusion* software of Mediso Ltd.

We successfully applied the proposed approach to align 3D lung CT scans [Sánta and Kato, 2018][Sánta and Kato, 2013a]. The input voxel volumes were first transformed into a triangular mesh using $r = 5$, and then the TPS parameters were recovered by `trisurf_vol`. For this experiment, we used 64 control points placed on a uniform grid. Promising results were obtained on the available 8 image pairs with a median δ error of 5.41% (the mean and standard deviation were 5.83% and 2.09%, respectively). Similarly to robustness tests, we added the *bending energy* from Equation (3.17) to the algebraic error of the system of

equations, which is then minimized simultaneously. The minimizing of the *bending energy* gave a smooth diffeomorphic alignment, which is consistent on the grayscale level also. The λ parameter was set experimentally to 10^{-8}.

Some of our results presented in Figure 3.13, where we also show the achieved inner alignment on grayscale slices of the original lung CT images. For these slices, the original and transformed images were combined as an 8×8 checkerboard pattern [Sánta and Kato, 2018][Sánta and Kato, 2013a].

Brain Surface Alignment

Pairwise brain registration is a very important and actively studied field in medical image processing [11, 53, 113]. Herein, motivated by the algorithm described in [113], the proposed approach is applied to register the pial surfaces of human brains between different patients. Following [113], our aim was to develop a surface based registration approach acting as an initial step in a multi-phase registration algorithm. Accordingly, the algorithm is intended to be fast and accurate on the surface, but we did not require proper alignment of the inner parts.

The MR brain data sets and their manual segmentations were provided by the Center for Morphometric Analysis at Massachusetts General Hospital and are available at `http://www.cma.mgh.harvard.edu/ibsr/`. Before the registration, the images have been resliced to achieve isotropic voxels in millimeters, then a triangular mesh with $r = 5$ have been extracted. Also note that, the volumetric images have been "positionally normalized" into the Talairach orientation (rotation only) by default. The dataset could be rearranged into three groups based on the physical resolution of each image, hence we solved the registration between images from the same resolution group.

In our first experiment, we tried to align the extracted surfaces using TPS model built from 64 control points and while we got promising results on the accuracy of the alignment, the runtime of each test case became too high for an initial alignment (*e.g.* 3.7 minutes in average). Then inspired by [113], we aligned the left and right pial surfaces separately using 32 control points for each, which made our results more accurate and the runtime became 7 times faster. In both cases, we used the `trisurf_vol` approach for estimating the parameters of the TPS model with the bending energy minimization having $\lambda = 10^{-5}$.

The outline of this experiment can be found in Table 3.4 and we show two examples in Figure 3.14. The results have been quantitatively evaluated by the δ and the D_{RMS} measures, too, because the latter metric raises higher penalties on larger surface distances. The results are encouraging, emphasizing the running times of the algorithm giving an acceptable result in 35 seconds for one complete surface.

3.5.5 3D Face Alignment

The analysis of 3D faces is an important task in many applications, like face comparison or face motion capture [114–117]. A core component of many 3D face analysis tasks is the geometric alignment of faces. Most of the alignment algorithms are relying on the extraction of well-defined feature points or landmarks but, unfortunately, landmark localization in 3D data is a hard task even by user interaction [Sánta and Kato, 2018][Sánta and Kato, 2016a].

Our goal is to find a non-rigid alignment between triangulated 3D faces and we used the Bosphorus Dataset [118] for our experiments [Sánta and Kato, 2018][Sánta and Kato,

		Group 1	Group 2	Group 3
D_{RMS}	m	2.45	2.81	1.82
	μ	2.86	2.85	1.92
	σ	1.13	0.4	0.31
δ (%)	m	4.26	7.2	3.9
	μ	4.27	7.18	3.83
	σ	0.29	0.58	0.19
Runtime (sec)	m	34	35	35.5
	μ	34.36	34.93	37
	σ	7.77	4.5	7.24

Table 3.4: The results on brain surface alignment (m – median, μ – mean, σ – standard deviation). The input data has been classified into three groups based on the physical resolution of each image.

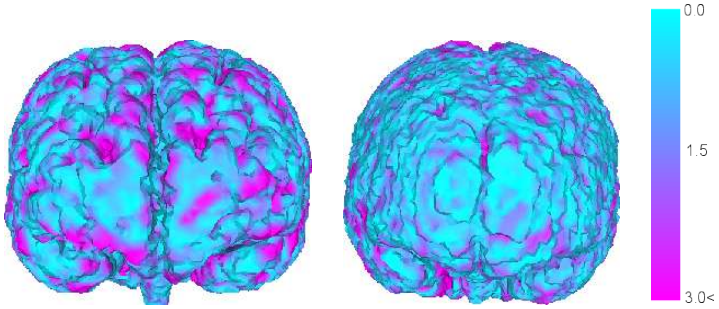


Figure 3.14: Two results of the brain surface alignment tests from different resolution groups having various qualities.

2016a]. This dataset consists 4666 face scans with 2D color images and 3D surface points. The ground truth landmark locations are also denoted in 24 feature points, giving the opportunity to evaluate our approach in the landmarks.

The triangular surface meshes have been generated from the Bosphorus 3D point clouds by making use of the *Poisson algorithm* [119] implemented using CGAL [120]. Similarly to the synthetic tests, the mesh resolution is controlled by the maximal radius of the corresponding Delaunay sphere for each triangle. Since only one side of the human head can be reconstructed from the data, the triangular surfaces will be not closed. Therefore, we used the `trisurf_surf-approx` algorithm with 64 control points for the TPS model, placed on the surface by the *Farthest Point Sampling* based strategy [Sánta and Kato, 2018][Sánta and Kato, 2016a].

During our tests, we observed that not every part of the triangular surfaces are equally important for aligning the faces, but our algorithm generally assume this [Sánta and Kato, 2018][Sánta and Kato, 2016a]. Moreover, facial scans typically focus on the frontal face, but depending on the actual setting, other parts of the head are also visible. Therefore the scanned surfaces will not match as a whole! The exact segmentation of corresponding parts is a rather complex problem as there are no clearly defined borders of a face in a 3D scan.

Instead of solving a hard 3D face segmentation problem, let us define the integration domains in Equation (3.34) as *fuzzy sets* [121, 122] with a $W_T : T_\triangle \to [0,1]$ and $W_O : O_\triangle \to [0,1]$ for the *template* and the *observation*, respectively [Sánta and Kato, 2018][Sánta and Kato, 2016a]. These membership functions are assigning a weight to

each triangle in the integrals of Equation (3.34):

$$\sum_{o \in O_{\triangle}} W_O(o) \int_o \omega_i(\mathbf{y})\, d\mathbf{y} \approx \sum_{\pi \in \varphi(T_{\triangle})} W_T(\pi) \int_{\pi} \omega_i(\mathbf{z})\, d\mathbf{z}, \tag{3.75}$$

where $i = 1, \ldots, \ell$. We refer to triangles with membership value 1 as *inner parts* of the face, while triangles with 0 membership value will be *outer parts*. According to Equation (3.75), the outer parts have no contribution to the equations, however, the inner parts are always considered. Every triangle which is not in these two sets will be referred to as the *fuzzy parts* of the faces. A membership function is given by three parameters:

- λ_1 the upper threshold of the inner parts (the value is constant 1)
- λ_2 the lower threshold of the outer parts (the value is constant 0)
- the interpolation method for the area between λ_1 and λ_2

The thresholds are expressed as the percentage of the maximal geodesic distance within the face. We tried two different maximal values: the true maximal geodesic distance from the closest point to the camera and the mean of the top 5% geodesic distances from the same point. When triangle membership functions are applied, the control point placement is also restricted to the inner parts of the *template*.

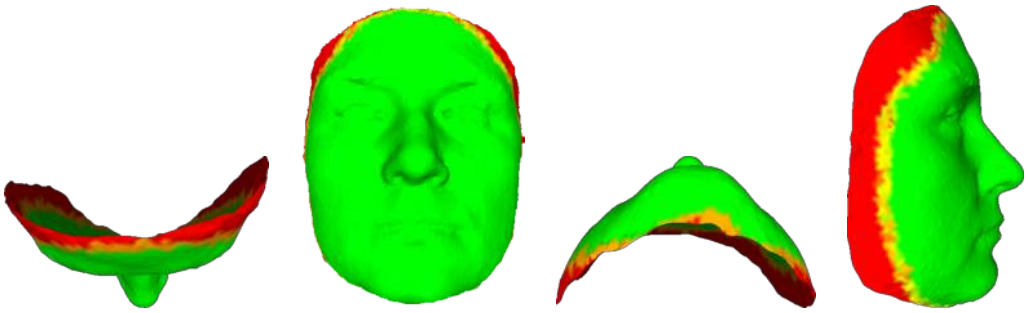


Figure 3.15: An example for the membership functions. The green area denotes the inner parts having membership value of 1, while the yellow and red regions denoting the areas between λ_1 and λ_2, and the areas above λ_2, respectively.

How to determine λ_1 and λ_2 thresholds? In the following, we will assume that the corresponding areas of the faces contain all of the main facial landmarks, thus the estimation will be based on the normalized geodesic distances of the landmarks [Sánta and Kato, 2018][Sánta and Kato, 2016a]. According to our experiments, taking the mean of the maximal normalized distances of the landmarks for each face in a training set will provide a general upper bound, where the resulting interval will contain the distances for most of the facial landmarks in the set. Based on this observation the thresholds can be estimated by making use of the ground truth landmark locations. Let us denote the set of landmarks of the i^{th} training face by $L^i = \{\mathbf{l}_j \in \mathbb{R}^3\}$. First, for each face from the training set, we calculate the normalized geodesic distances of its landmarks w.r.t. the nose tip as described above. Then, according to our observation, we determine the farthest landmark for the i^{th} face as

$$\lambda_1^i = \max\left(\{D_G(\mathbf{l}_j) \mid \mathbf{l}_j \in L^i\}\right). \tag{3.76}$$

Finally, taking the mean of the estimated λ_1^i set will provide λ_1 for the whole dataset

$$\lambda_1 = \mu\left(\{\lambda_1^i\}\right). \tag{3.77}$$

Similarly, for λ_2 we simply add the standard deviation of the maximal normalized distances to λ_1:

$$\lambda_2 = \lambda_1 + \sigma\left(\{\lambda_1^i\}\right). \tag{3.78}$$

Now, let us summarize the experimental results on the Bosphorus Dataset [118]. The tests have been made with a randomly generated subset containing 153 pairs of faces with neutral facial expression and we determined the deformations between faces of different people (*i.e. inter-patient* registration). The results have been quantitatively evaluated based on the average landmark distances between the transformed and the ground truth locations:

$$D_{GT} = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|, \tag{3.79}$$

where N is the number of available landmarks for a pair of scans, $\mathbf{x}_i \in O_\triangle$ is the ground truth and $\hat{\mathbf{x}}_i = \varphi(\mathbf{z}_i), \mathbf{z}_i \in T_\triangle$ is the transformed landmark position of the i^{th} landmark using the estimated aligning transformation φ. The overall surface alignment accuracy also has been characterized by Equation (3.74) metric.

In our first experiment, we show the benefits of the membership function on meshes generated by $r = 10$ as the maximal radius of the enclosing Delaunay sphere. As described above, each membership function depends on three parameters: the value of the thresholds (λ_1 and λ_2) and the interpolation method for the interval between the thresholds. For the interpolation, we define two possible functions:

$$I_{\lambda_1\lambda_2}^{step}(d) = 0.5, \tag{3.80}$$

$$I_{\lambda_1\lambda_2}^{linear}(d) = 1 - \frac{d-\lambda_1}{\lambda_2-\lambda_1}, \tag{3.81}$$

where Equation (3.80) is a simple constant function referred to as `step`, while Equation (3.81) is a linear interpolation function called `linear` in the following. In order to determine the threshold values, we have to detect the nose tip for estimating the geodesic distances. Fortunately, the coordinate system of the scans in the Bosphorus dataset is established in a way, that the point having the maximal value on the Z axis is a good estimate for the nose tip, thus we used this point in the geodesic distance calculations. Then, using the algorithm described above, we determined the λ_1 and λ_2 parameters and got 71.07% and 75.82% for λ_1 and λ_2, respectively. The results of this test are shown in Figure 3.16. According to the D_{GT} error measure from Equation (3.79), we have noticed that the interpolation method has no strong influence on the outcome of the algorithm, therefore we recommend to use the `step` function from Equation (3.80).

For our next experiments, we tried several triangular surface resolutions with $r \in \{2, 3, 5\}$ as the maximal radius for the enclosing Delaunay sphere of each triangle (similarly to the synthetic experiments). Note that, these surface approximations will eventually reduce the resolution of the input data by removing points and smoothing the surface. The loss in the resolution can be measured by the number of vertices and triangles, thus the corresponding statistics can be found in Table 3.5. Just as in the synthetic case, the aim of the current experiment is to find the best runtime over quality ratio. The outline of this test is shown

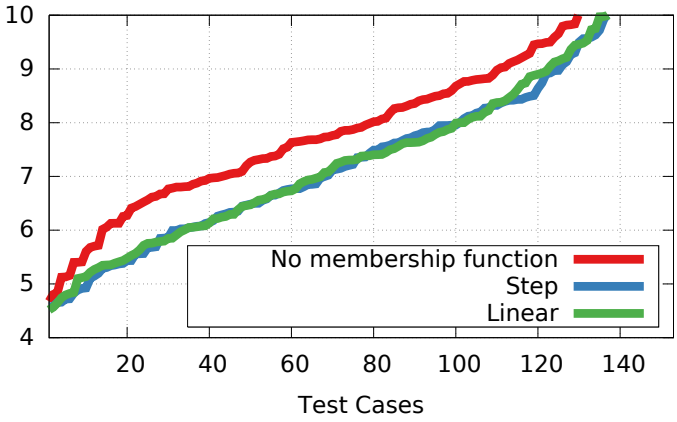
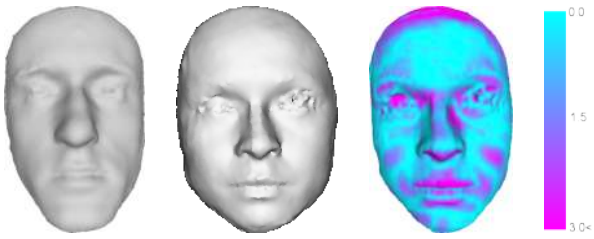


Figure 3.16: The obtained alignment accuracy in the first experiment estimated by D_{GT}. Each plot shows results with different weight functions.



template observation distances

Figure 3.17: Example alignment on the Bosphorus dataset. The $D_{RMS} = 1.17mm$ and the $D_{GT} = 5.4mm$.

in Figure 3.18. We achieved the best results on the $r = 3$ case, with respect to the D_{GT} measure and the runtime of the algorithm. The average D_{GT} error for the test dataset is $6.71mm$ with an average running time of 16.4 seconds. Some results are presented in Figure 3.19. For the surface alignment accuracy we got $D_{RMS} = 1.76mm$ (see Figure 3.17 for an example). From the results we conclude that the algorithm achieves good results near the areas with significant curvature changes (*e.g.* nose and mouth), however, performs poorly near the noisy eyebrows.

Point Cloud [118]	$r = 10$	$r = 5$	$r = 3$	$r = 2$
36684	1821 3826	2358 4487	4128 7994	8298 16274

Table 3.5: A comparison of different mesh resolutions and the input point cloud [118]. We show the number of points for the point cloud and the number of vertices and triangles, respectively, for the triangular surface meshes.

In the last experiment, we have compared our results to GMMREG [5] and CPD [6] approaches. Since the runtime of these algorithms are enormously high for the full original point sets (the average is around 7-8 hours for one pair of faces), the methods have been applied to the vertices of the same surface meshes as we used for our algorithm. The results can be found in Figure 3.20. We used the `step` interpolation method for this test. The GMMREG algorithm has achieved very good results, slightly outperforming our method and the CPD algorithm on the D_{GT} error, but CPD gives inferior alignment compared to our approach. However, the proposed approach achieved the lowest running time.

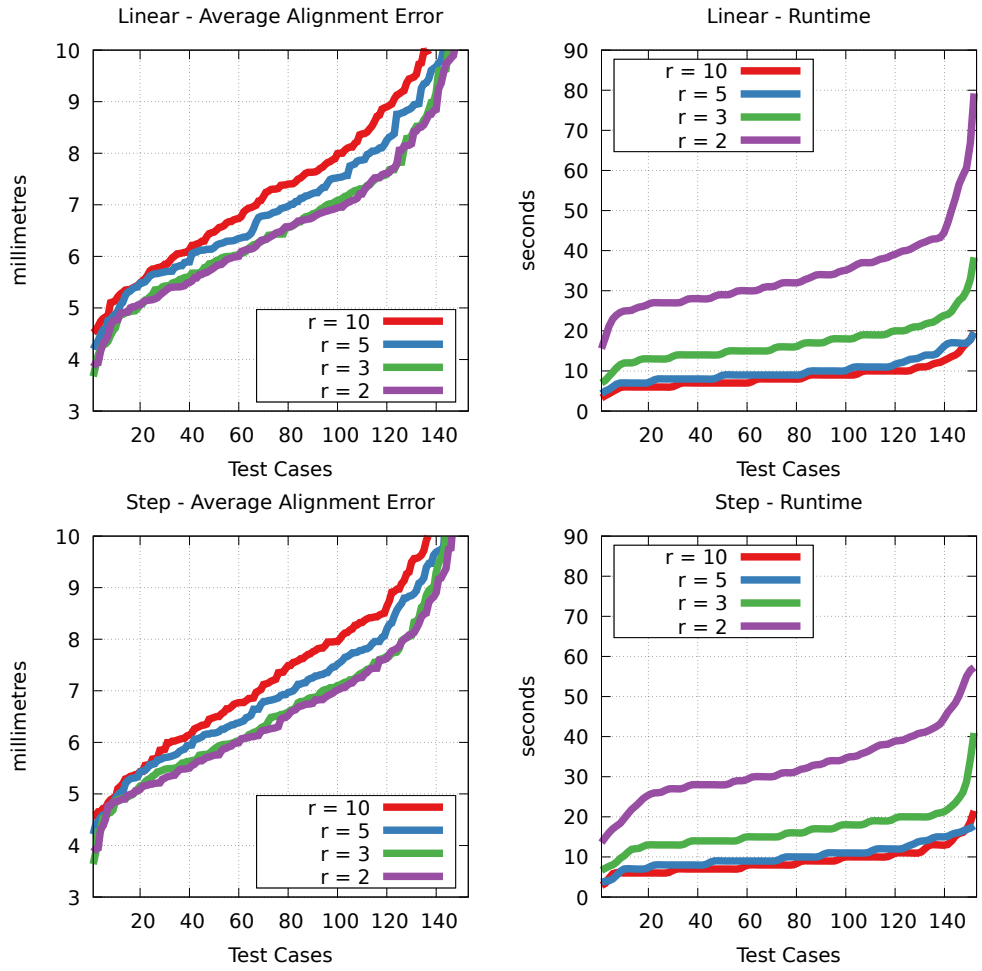


Figure 3.18: Alignment errors and running times for each resolution of $r \in \{2, 3, 5, 10\}$. The alignment errors are determined as the Euclidean distance of the transformed and the ground truth landmark locations. The first line corresponds to the `step`, the second line to the `linear` interpolation method.

3.6 Summary

We have proposed a novel registration framework for aligning 3D objects without established point-wise correspondences. The framework is able to work with multiple data representations. The basic idea is to set-up a system of non-linear equations whose solution directly provides the parameters of the aligning transformation modeled by a parametric transformation model. Efficient numerical schemes were proposed for voxel and for closed and opened triangular mesh representations. The efficiency and robustness of the proposed approach have been demonstrated on large synthetic datasets. Our method compares favorably to two recent 3D matching algorithms [5, 6]. Finally, the algorithm achieved promising results in aligning lung CT images, brain surfaces and 3D facial scans, which demonstrates the usefulness of the method in real life applications.

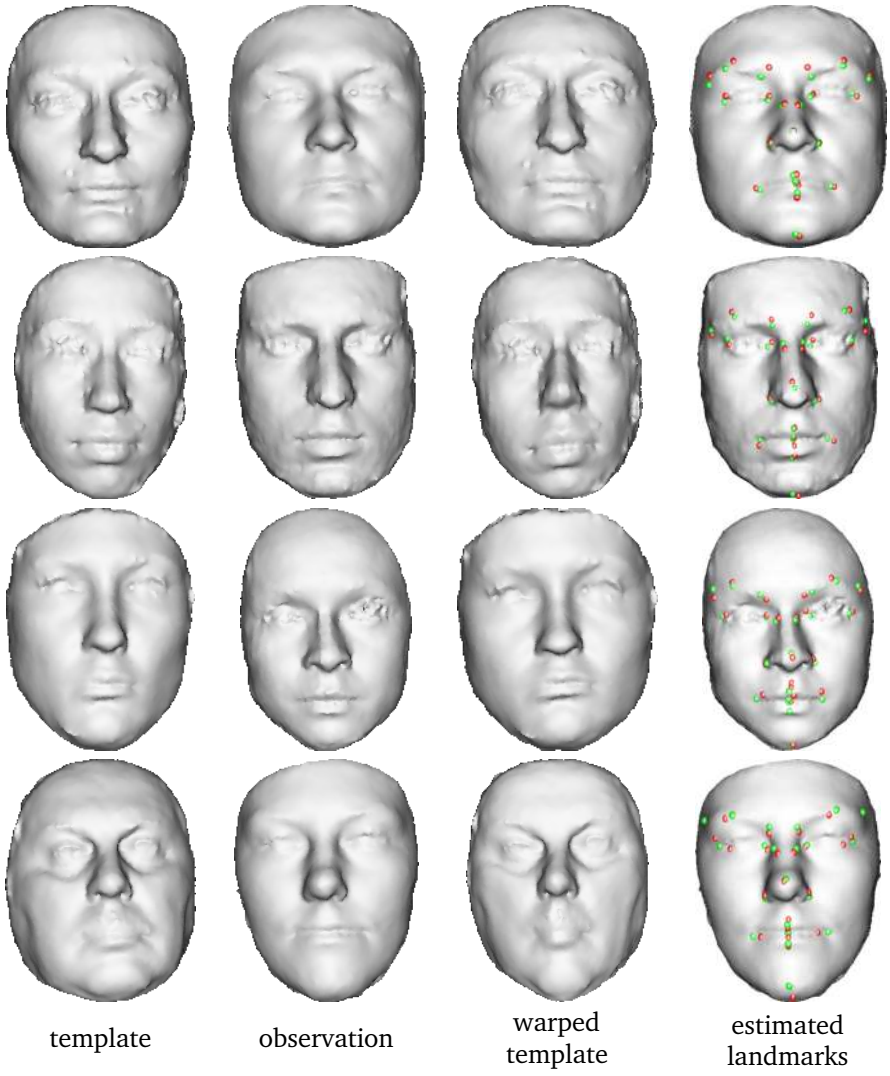


Figure 3.19: Sample results on the Bosphorus dataset. In each row, we show the *template*, the *observation*, the warped *template* and the landmark locations, respectively. In the last column, green denotes the ground truth position and red is the estimated location. While the proposed approach achieved good results near nose and mouth areas, the highest errors are near the eyebrows.

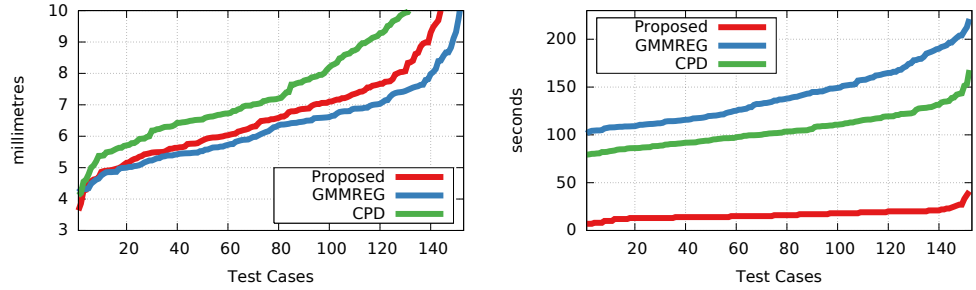


Figure 3.20: Comparison between the proposed, the GMMREG [5] and the CPD [6] algorithms. On the left diagram, the D_{GT} measure is presented, while on the right we show the running times of each algorithm. The GMMREG slightly outperformed the proposed approach and the CPD algorithm in terms of alignment accuracy, but the proposed approach has the best running time.

Chapter 4

Robust Registration of 2D Images

In this chapter, we will present our results on registration of 2D images. The work has addressed two different problems related to the algebraic framework introduced in the previous chapter. In the first problem, the method's tolerance against larger segmentation errors has been investigated [Sánta and Kato, 2014].

In the second problem, we dealt with the ambiguity of the inner parts of shapes, when they are registered using non-rigid transformation models with higher degrees of freedom. This issue has been handled by regularization in the previous chapter, while herein we investigate the possibility of using radiometric information within our algebraic framework [Sánta and Kato, 2016b].

Each proposed method has been experimentally validated on synthetic and real datasets. In addition, we have compared them to multiple state of the art approaches developed in the recent years.

4.1 Registration Framework

In the following, we will build our methods upon the 2D registration frameworks introduced in [7, 122]. Let us focus on the pair-wise registration of 2D image regions by estimating the transformation between a *template* and an *observation* patch. Let us assume that the points of these regions are denoted by $\mathcal{F}_t \subset \mathbb{R}^2$ and $\mathcal{F}_o \subset \mathbb{R}^2$, respectively. We are looking for the parameters of a $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ application specific transformation model which aligns the *template* and the *observation*.

Using these notations, the following relation holds for an arbitrary pair of corresponding points $\mathbf{x} = [x_1, x_2] \in \mathcal{F}_t$ and $\mathbf{y} = [y_1, y_2] \in \mathcal{F}_o$:

$$\varphi(\mathbf{x}) = \mathbf{y} \tag{4.1}$$

While each of these point correspondences give exactly two constraints on the parameters of φ, reliable landmark extraction could be a challenging task in the presence of non-rigid geometric distortions.

Following [7, 122], instead of extracting individual point-correspondences, let us integrate both sides of Equation (4.1):

$$\int_{\varphi(\mathcal{F}_t)} \mathbf{z}\, d\mathbf{z} = \int_{\mathcal{F}_o} \mathbf{y}\, d\mathbf{y}. \tag{4.2}$$

In order to avoid generation of the transformed domain, we can apply the integral transformation to Equation (4.2), similarly to the 3D framework described in the previous chapter:

$$\int_{\mathcal{F}_t} \varphi(\mathbf{x}) |J_\varphi(\mathbf{x})| \, d\mathbf{x} = \int_{\mathcal{F}_o} \mathbf{y} \, d\mathbf{y} \tag{4.3}$$

Recall that the integral transformation involves the Jacobian determinant of φ on the left-hand side of the equation, which describes the relative change induced by the transformation in the area of an infinitesimal region around each point. $|J_\varphi(\mathbf{x})|$ contains the partial derivatives of the transformation.

Although, the equations of Equation (4.2) define two constraints on the parameters of φ, generally, the number of parameters is much higher than two. One way to overcome this problem is to follow [7, 122] and apply a set of independent non-linear functions $\{\omega_i \mid \omega_i : \mathbb{R}^2 \to \mathbb{R}\}_{i=1}^{\ell}$ to the coordinates of Equation (4.2) and Equation (4.3), respectively, as

$$\int_{\varphi(\mathcal{F}_t)} \omega_i(\mathbf{z}) \, d\mathbf{z} = \int_{\mathcal{F}_o} \omega_i(\mathbf{y}) \, d\mathbf{y}, \tag{4.4}$$

$$\int_{\mathcal{F}_t} \omega_i(\varphi(\mathbf{x})) |J_\varphi(\mathbf{x})| \, d\mathbf{x} = \int_{\mathcal{F}_o} \omega_i(\mathbf{y}) \, d\mathbf{y}, \tag{4.5}$$

where $i = 1, \ldots, \ell$. Using polynomial ω functions will eventually lead to a polynomial system of equations and according to [7, 122], the framework is quite robust against several types of geometric noise and segmentation error [7, 122].

Yet, in previous works, the larger segmentation errors (*i.e.* occlusions or disocclusions) are categorized as a trivial case of failure, similarly to most area based methods. In many applications originated from industrial and medical areas, we only have to deal with the physical deformation due to the well-controlled circumstances of image acquisition, therefore this is not a serious issue in such cases. In a less controlled environment (*e.g.* processing images of surveillance systems), however, we always have to deal with different types of noise and occlusions [Sánta and Kato, 2014].

In the next section, we will propose a solution for registering occluded shapes in the presence of affine transformation. This group of transformations is useful in many practical applications when we are dealing with linear or perspective motions. In the latter case, the affine model is used as a first-order approximation of the deformation. This is a convenient way to model the transformation when we work on images taken with classical pin-hole cameras, but the distance of the camera from the objects is large compared to the size of the object [2].

4.2 Affine Alignment of Occluded Shapes

In order to adapt the proposed framework above, let us assume that pursued transformation is defined by a homogeneous affine matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.6}$$

A full affine transformation can be expressed using a single matrix multiplication between the homogeneous coordinates of the *template* and *observation* denoted by $\bar{\mathbf{x}} = [x_1, x_2, 1]^T \in \mathcal{F}_t \subset \mathbb{P}^2$ and $\bar{\mathbf{y}} = [y_1, y_2, 1]^T \in \mathcal{F}_o \subset \mathbb{P}^2$, respectively. Note that, herein the appliance of homogeneous coordinates only has practical importance. Moreover, since every affine transformation preserves the last coordinate of the points, it could be removed anytime to get the corresponding Cartesian point (see Section 2.2 for more details). Using such notations, the identity relation from Equation (4.1) will become [Sánta and Kato, 2014]:

$$\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{y}} \quad \Leftrightarrow \quad \bar{\mathbf{x}} = \mathbf{A}^{-1}\bar{\mathbf{y}}. \tag{4.7}$$

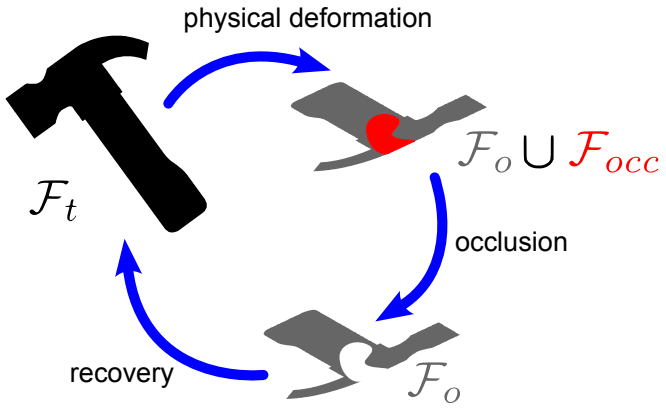


Figure 4.1: Relation between the *template* and the occluded *observation*.

While this relation is true when the *observation* is a perfect image of the *template*, in the presence of occlusions this will not be valid anymore. Let us denote the *occluded* regions of the *observation* by $\mathcal{F}_{occ}$ (see Figure 4.1 for an example) [Sánta and Kato, 2014]. Then, we can define an object level identity relation as:

$$\mathbf{A}(\mathcal{F}_t) = \mathcal{F}_o \cup^* \mathcal{F}_{occ} \iff \mathcal{F}_t = \mathbf{A}^{-1}(\mathcal{F}_t \cup^* \mathcal{F}_{occ}), \tag{4.8}$$

where $\cup^*$ denotes the disjoint union of two sets (*i.e.* there are no overlaps between the sets). Using these sets as domains in Equation (4.5) leads to the following system:

$$\int_{\mathbf{A}(\mathcal{F}_t)} \omega_i(\bar{\mathbf{z}})\, d\bar{\mathbf{z}} = \int_{\mathcal{F}_o \cup^* \mathcal{F}_{occ}} \omega_i(\bar{\mathbf{y}})\, d\bar{\mathbf{y}} \tag{4.9}$$

$$\int_{\mathcal{F}_t} \omega_i(\bar{\mathbf{x}})\, d\bar{\mathbf{x}} = \int_{\mathbf{A}^{-1}(\mathcal{F}_o \cup^* \mathcal{F}_{occ})} \omega_i(\bar{\mathbf{p}})\, d\bar{\mathbf{p}}, \tag{4.10}$$

where $i = 1, \ldots, \ell$. For the sake of simplicity, we will assume that the ω_i functions are acting on the Cartesian coordinates of the points [Sánta and Kato, 2014]. Observe that for all non-zero $\mathbf{A}$ affine transformations, the following equation is valid:

$$\mathbf{A}^{-1}(\mathcal{F}_o \cup^* \mathcal{F}_{occ}) = \mathbf{A}^{-1}(\mathcal{F}_o) \cup^* \mathbf{A}^{-1}(\mathcal{F}_{occ}) \tag{4.11}$$

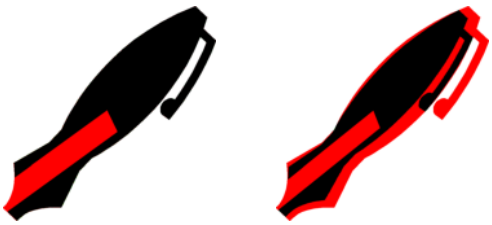
Substituting this observation into the integration domain of the systems from Equation (4.9)

and Equation (4.10) and applying the basic properties of Lebesgue-integrals, we get:

$$\int_{\mathbf{A}(\mathcal{F}_t)} \omega_i(\bar{\mathbf{z}})\, d\bar{\mathbf{z}} = \int_{\mathcal{F}_o} \omega_i(\bar{\mathbf{y}})\, d\bar{\mathbf{y}} + \int_{\mathcal{F}_{occ}} \omega_i(\bar{\mathbf{y}}')\, d\bar{\mathbf{y}}' \tag{4.12}$$

$$\int_{\mathcal{F}_t} \omega_i(\bar{\mathbf{x}})\, d\bar{\mathbf{x}} = \int_{\mathbf{A}^{-1}(\mathcal{F}_o)} \omega_i(\bar{\mathbf{q}})\, d\bar{\mathbf{q}} + \int_{\mathbf{A}^{-1}(\mathcal{F}_{occ})} \omega_i(\bar{\mathbf{q}}')\, d\bar{\mathbf{q}}' \tag{4.13}$$

where $i = 1, \ldots, \ell$. Similarly to the affine framework proposed in [122], we use simple power functions for the $\{\omega_i\}_{i=1}^{\ell}$ set and choosing $\ell \geq 6$ yields an overdetermined system, which is then solved in the least-squares sense [Sánta and Kato, 2014].



without regularization with regularization

Figure 4.2: The ambiguity of the proposed system of equations. Each image shows the transformed observation and the occlusion denoted by black and red colors, respectively.

Within the proposed framework, several algebraically equivalent solutions are possible. For example, a trivial solution is when all parameters of the transformation become zero and the target shape will be the occlusion. Another example can be found in Figure 4.2. In order to overcome this issue, the problem has to be regularized [Sánta and Kato, 2014]. Herein, we use a simple regularization by restricting the size of the occluded areas to be minimal and using Equation (4.13) yields the following problem:

$$\begin{aligned} &\int_{\mathcal{F}_t} \omega_i(\bar{\mathbf{x}})\, d\bar{\mathbf{x}} = \int_{\mathbf{A}^{-1}(\mathcal{F}_o)} \omega_i(\bar{\mathbf{q}})\, d\bar{\mathbf{q}} + \int_{\mathbf{A}^{-1}(\mathcal{F}_{occ})} \omega_i(\bar{\mathbf{q}}')\, d\bar{\mathbf{q}}' \\ &\text{such that } |\mathbf{A}^{-1}(\mathcal{F}_{occ})| \to \min \qquad\qquad i = 1, \ldots, \ell, \end{aligned} \tag{4.14}$$

where $|\mathbf{A}^{-1}(\mathcal{F}_{occ})|$ denotes the area of $\mathbf{A}^{-1}(\mathcal{F}_{occ})$. Note that, for Equation (4.13) a similar problem can be defined, which will be equivalent to Equation (4.14).

4.2.1 Calculating the Integrals Efficiently

Although the construction of Equation (4.14) is straightforward, similarly to the 3D case discussed in Chapter 3, the computational efficiency strongly depends on the shape representation and the chosen $\{\omega_i\}$ sets. Since most of the suitable least-square solvers for the current problem use iterative minimization techniques, we have to adjust the integration domains and recompute the integrals for each iteration.

First, let us focus on the generation of the integration domains. Obviously, this process is very time consuming with the classical pixel-based shape representation. In order to overcome this difficulty, we will extract polygonal approximations of the contour of the shapes with consistent vertex orders [Sánta and Kato, 2014]. Conventionally, vertices of the outer contour are ordered counter-clockwise, while vertices of the holes are ordered clockwise. A polygonal approximation can be easily extracted from the contour of each shape by making use of the algorithm from [123]. Note that, this representation is a 2D analogy for the closed triangular surfaces in 3D as presented in Chapter 3.

Herein, we assume that a polygon $P_\Diamond$ corresponds to an ordered set of vertices, where each pair of subsequent points is connected by a straight line segment. The polygon is closed, thus the first and the last points are also connected. Each $P_\Diamond$ closed polygon defines a continuous domain enclosed by its border which will be denoted by $\mathcal{D}_{P_\Diamond}$. Let us denote the polygonal approximations of the borders of $\mathcal{F}_t$, $\mathcal{F}_o$ and $\mathcal{F}_{occ}$ by $T_\Diamond$, $O_\Diamond$ and $M_\Diamond$, respectively. Moreover, let us extend an affine transformation to a polygon as

$$\mathbf{A}(P_\Diamond) = \{\mathbf{A}\bar{\mathbf{v}} \mid \bar{\mathbf{v}} \in P_\Diamond\}. \tag{4.15}$$

Once we have the polygonal representation of the input shapes, let us observe how to calculate the polygonal approximation for borders of the shapes from Equation (4.11). For each iteration of the solver, we get a candidate solution for $\mathbf{A}$ and the inverse $\mathbf{A}^{-1}$ can be easily obtained by making use of

$$\mathbf{A}^{-1} = \begin{pmatrix} \frac{a_{22}}{|\mathbf{A}|} & \frac{-a_{12}}{|\mathbf{A}|} & \frac{a_{12}a_{23}-a_{13}a_{22}}{|\mathbf{A}|} \\ \frac{-a_{21}}{|\mathbf{A}|} & \frac{a_{11}}{|\mathbf{A}|} & \frac{a_{13}a_{21}-a_{11}a_{23}}{|\mathbf{A}|} \\ 0 & 0 & 1 \end{pmatrix} \tag{4.16}$$

where a_{ij}, $i = 1, \ldots, 2$, $j = 1, \ldots, 3$ are the elements and $|\mathbf{A}| = a_{11}a_{22} - a_{21}a_{12}$ is the determinant of $\mathbf{A}$. Therefore, the polygonal approximation for $\mathbf{A}^{-1}(\mathcal{F}_o)$ can be easily obtained by applying the $\mathbf{A}^{-1}$ to $O_\Diamond$ [Sánta and Kato, 2014]. Then from Equation (4.11) and the right-hand side of Equation (4.8) we can obtain the polygon of the transformed occlusion by making use of simple boolean operators

$$\mathbf{A}^{-1}(M_\Diamond) = T_\Diamond \setminus \mathbf{A}^{-1}(O_\Diamond), \tag{4.17}$$

$$M_\Diamond = \mathbf{A}^{-1}(O_\Diamond) \setminus \mathbf{A}(T_\Diamond), \tag{4.18}$$

where $\setminus$ denotes the relative complement operator between two polygons. This task can be easily solved by the *General Polygon Clipper (GPC)* library [124]. Using the polynomial approximations we have:

$$\begin{aligned} \mathcal{D}_{T_\Diamond} &\approx \mathcal{F}_t \\ \mathcal{D}_{O_\Diamond} &\approx \mathcal{F}_o \\ \mathcal{D}_{M_\Diamond} &\approx \mathcal{F}_{occ}. \end{aligned}$$

Integrating power functions over polygons in 2D is similarly convenient as integrating over triangular surfaces in 3D. Moreover, the computational complexity is also minimal due to recursive formulas. Therefore the $\{\omega_i\}$ can be defined as follows:

$$\omega_i(\bar{\mathbf{x}}) = x_1^{n_i} x_2^{m_i}, \tag{4.19}$$

where $\{(n_i, m_i)\}_{i=1}^{l} = \{(a, b) \mid a+b = O\}$ and $O \in \{0, \ldots, O_{max}\}$. Substituting the polygonal representation and the $\{\omega_i\}$ set to the system from Equation (4.14) we get

$$\begin{aligned} &\int_{\mathcal{D}_{T_\Diamond}} x_1^{n_i} x_2^{m_i} \, d\bar{\mathbf{x}} = \int_{\mathcal{D}_{\mathbf{A}^{-1}(O_\Diamond)}} y_1'^{n_i} y_2'^{m_i} \, d\bar{\mathbf{y}}' + \int_{\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}} z_1'^{n_i} z_2'^{m_i} \, d\bar{\mathbf{z}}' \\ &\text{such that } |\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}| \to \min \qquad\qquad i = 1, \ldots, \ell. \end{aligned} \tag{4.20}$$

Note that, the integrals in Equation (4.20) are simple geometric moments of the domains, which can be efficiently computed using the algorithm proposed in [125, 126]. Let us consider the approximating polygon as an ordered list of n vertices, denoted by $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2, \ldots, \bar{\mathbf{p}}_n \in P_\diamond$. With each pair of adjacent vertices $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{p}}_{i+1}$ and using the origin of the coordinates $(0,0)$, a set of triangles can be created. Let us denote the triangle created from $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{p}}_{i+1}$ as $P_i^\triangle$ and the continuous domain enclosed by this triangle as $\mathcal{D}_i^\triangle$. Note that, $P_i^\triangle$ will be the triangle defined by $\bar{\mathbf{p}}_n$, $\bar{\mathbf{p}}_1$ and the origin. Using these notations, the first integral of Equation (4.20) can be written as

$$\int_{\mathcal{D}_{T_\diamond}} x_1^{n_i} x_2^{m_i} \, d\bar{\mathbf{y}} = \sum_{j=1}^{n} \int_{\mathcal{D}_j^\triangle} X_1^{n_i} X_2^{m_i} \, d\bar{\mathbf{X}}, \tag{4.21}$$

and the other integrals similarly [Sánta and Kato, 2014]. According to [125, 126], the moments of a triangle can be calculated exactly using the following formula

$$\int_{\mathcal{D}_j^\triangle} X_1^{n_i} X_2^{m_i} \, d\bar{\mathbf{X}} = |\mathcal{D}_j^\triangle| \sum_{k=0}^{n_i} \sum_{l=0}^{m_i} C_{kl}(n_i, m_i) u_j^k u_{j+1}^{n_i-k} v_j^l v_{j+1}^{m_i-l}, \tag{4.22}$$

where

$$C_{kl}(m_i, n_i) = \frac{2 n_i! m_i! (k+l)! (n_i + m_i - k - l)!}{(n_i - k)! (m_i - l)! k! l! (n_i + m_i + 2)!} \tag{4.23}$$

is a constant coefficient for each degree, (u_j, v_j) and (u_{j+1}, v_{j+1}) are the coordinates of $\bar{\mathbf{p}}_j$ and $\bar{\mathbf{p}}_{j+1}$, respectively, while $|\mathcal{D}_j^\triangle|$ denotes the area of the j^{th} triangle [Sánta and Kato, 2014].

4.2.2 Numerical Implementation

In order to solve our system of equations, we have to deal with several numerical issues. Note that the polygonal representation is only an approximation of a real object, thus the integrals from Equation (4.20) are only approximately valid [Sánta and Kato, 2014]. Moreover, the objects extracted from the images are subject to various segmentation errors which are not explicitly modeled, thus they introduce additional errors in Equation (4.20). Fortunately, we are able to handle these by using an overdetermined system.

Similarly to [122], we observed that the given framework is not robust against higher rotations. To ensure the robustness against these local optima, the algorithm use a branch-and-bound like optimization process, starting the estimation from several initial configurations [122]. The optimizer stops after a few steps, then the main algorithm is initialized with the best solution of the branch-and-bound phase. Since this particular problem is restricted only to the rotational part of the transformation, the whole $360°$ angle interval is divided into six $60°$ ranges [Sánta and Kato, 2014].

To achieve higher numerical stability, we also include equations from Equation (4.13). Although it is mathematically redundant, numerically it has stabilizing effect on the solver. With both systems involved, the problem from Equation (4.14) becomes

$$\begin{aligned}
\int_{\mathcal{D}_{T_\diamond}} x_1^{n_i} x_2^{m_i} \, d\bar{\mathbf{x}} &= \int_{\mathcal{D}_{\mathbf{A}^{-1}(O_\diamond)}} y_1'^{n_i} y_2'^{m_i} \, d\bar{\mathbf{y}}' + \int_{\mathcal{D}_{\mathbf{A}^{-1}(M_\diamond)}} z_1'^{n_i} z_2'^{m_i} \, d\bar{\mathbf{z}}' \\
\int_{\mathcal{D}_{O_\diamond}} y_1^{n_i} y_2^{m_i} \, d\bar{\mathbf{y}} &= \int_{\mathcal{D}_{\mathbf{A}(T_\diamond)}} x_1'^{n_i} x_2'^{m_i} \, d\bar{\mathbf{x}}' - \int_{\mathcal{D}_{M_\diamond}} z_1^{n_i} z_2^{m_i} \, d\bar{\mathbf{z}}
\end{aligned} \tag{4.24}$$

such that $|\mathcal{D}_{\mathbf{A}(M_\diamond)}| \to \min \qquad i = 1, \ldots, \ell.$

Another interesting problem is the question of introducing the regularization term to the system of equations? Since the system is solved in the least-square sense, the sum of squared differences of the equations (*i.e.* the algebraic error) is minimized. With this consideration, we construct a non-linear cost function from the algebraic error and the regularization term as:

$$\begin{aligned}&\sum_{i=1}^{\ell}\left(\int_{\mathcal{D}_{T_\Diamond}} x_1^{n_i}x_2^{m_i}\,d\bar{\mathbf{x}} - \int_{\mathcal{D}_{\mathbf{A}^{-1}(O_\Diamond)}} y_1'^{n_i}y_2'^{m_i}\,d\bar{\mathbf{y}}' - \int_{\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}} z_1'^{n_i}z_2'^{m_i}\,d\bar{\mathbf{z}}'\right)^2 + \\ &\sum_{i=1}^{\ell}\left(\int_{\mathcal{D}_{O_\Diamond}} y_1^{n_i}y_2^{m_i}\,d\bar{\mathbf{y}} - \int_{\mathcal{D}_{\mathbf{A}(T_\Diamond)}} x_1'^{n_i}x_2'^{m_i}\,d\bar{\mathbf{x}}' + \int_{\mathcal{D}_{M_\Diamond}} z_1'^{n_i}z_2'^{m_i}\,d\bar{\mathbf{z}}'\right)^2 + \\ &\lambda|\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}|,\end{aligned} \tag{4.25}$$

where λ is a non-zero weight, denoting the contribution of the regularization term to the cost function [Sánta and Kato, 2014]. Although it is a very straightforward solution, the output of the algorithm highly depends on the choice of the λ parameter. In order to reduce the dependence on this value, we have to solve the problem differently.

Observe that, the problem can be reformalized as a constrained minimization problem with non-linear constraints in the form of

$$\begin{aligned}&\min_{\mathbf{A}} |\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}| \qquad \text{s. t.} \\ &\int_{\mathcal{D}_{T_\Diamond}} x_1^{n_i}x_2^{m_i}\,d\bar{\mathbf{x}} = \int_{\mathcal{D}_{\mathbf{A}^{-1}(O_\Diamond)}} y_1'^{n_i}y_2'^{m_i}\,d\bar{\mathbf{y}}' + \int_{\mathcal{D}_{\mathbf{A}^{-1}(M_\Diamond)}} z_1'^{n_i}z_2'^{m_i}\,d\bar{\mathbf{z}}'\end{aligned} \tag{4.26}$$

where $i = 1, \ldots, \ell$. This scheme is not affected by the λ parameter, but the cost function has many local optima, which makes the problem harder to solve. Therefore, in our numerical solution, we combine the two formalisms by creating an alternating optimizer [Sánta and Kato, 2014]. In one iteration step of the optimizer, we minimize the cost function of Equation (4.25), then using the result as initialization we solve the minimization problem of Equation (4.26). Unfortunately, we observed that these two problems could not be handled efficiently with the same algorithm, since Equation (4.25) is a system of non-linear equations, while Equation (4.26) is a minimization problem with non-linear constraints. That is why we use the *Levenberg-Marquardt* algorithm for minimizing Equation (4.25) and the *Sequential Quadratic Programming (SQP)* algorithm to solve Equation (4.26). In each iteration step of the alternating optimizer we run the *Levenberg-Marquardt* for 10 iterations, then the *Sequential Quadratic Programming (SQP)* algorithm for 5 iterations (these numbers were determined experimentally) [Sánta and Kato, 2014].

The integrals are computed efficiently using the formulas given in Section 4.2.1. One can consider this alternating scheme as follows: in the first step we are looking for an algebraic solution of a system of equations and in the second step a geometrical constraint is enforced. This dual approach yields stable convergence and we found out experimentally that small fluctuations in the value of λ do not affect the results (in our tests we were using $\lambda = 3$ for all cases) [Sánta and Kato, 2013a].

Since the solution of the problem is based on iterative minimization, proper normalization is critical for numerical stability. In this approach, following [7], we normalize both the coordinates of the objects and the range of the ω_i functions. The shapes are transformed into the unit circle centered at the origin, yielding all coordinates to be in $[-0.5, 0.5]$. The range of the ω_i functions is normalized into $[-1, 1]$, by making use the normalization constants

from [7].

Algorithm 4.1 Pseudo code of the affine registration algorithm for occluded shapes

Input: *template* and *observation* objects
Output: The transformation parameters of $\mathbf{A}$

1: Extract the polygonal approximations of the objects, then apply the normalization from [7].
2: Construct the system of equations Equation (4.24).
3: Find a branch-and-bound initialization transformation.
4: **while** not converged **do**
5: Run the *Levenberg-Marquardt* algorithm for 10 iterations on minimizing Equation (4.25).
6: Run the *Sequential Quadratic Programming (SQP)* for 5 iterations on minimizing Equation (4.26).
7: The optimizer is converged to a solution if the L_2-distance of the current and the last solution is smaller than a problem dependent threshold.
8: **end while**
9: Denormalizing the solution gives the parameters of the aligning transformation.

4.2.3 Experimental Results

In our experiments, a system of 9 equations has been generated using the formula $\omega_i(\bar{\mathbf{x}}) = x_1^{n_i} x_2^{m_i}$, where $\{(n_i, m_i)\}_{i=1}^{9} = \{(a, b) \mid a + b = O\}$ and $O \in \{0, \ldots, 3\}$. The algorithm has been implemented in MATLAB, except for the *GPC* library, which is called through a MEX interface. The *Levenberg Marquardt* and *Sequential Quadratic Programming (SQP)* solvers are used from MATLAB's Optimization Toolbox. All tests have been run under a Linux system on a virtualized Core i5 3.1 GHz architecture.

In order to quantitatively evaluate the performance of the proposed method, we have tested the algorithm on 3000 synthetically generated shapes obtained from the homepage of [127] (`http://www.inf.u-szeged.hu/~kato/software/affbinregdemo.html`). According to [127], these images have been generated with the following parameters: rotation angle from $[0°, 350°]$, shearing from $[0, 1.2]$, scaling from $[0.5, 1.9]$ and translation from $[-20, 20]$ pixels. The average resolution of the images is 1000×1000.

For testing the behavior of the algorithm with occlusions, we used the following algorithm to generate occluded observations: Choose a random point of the contour and add this point to the border of the occlusion (initially this is empty). Iteratively, grow the occlusion by adding the neighboring pixels of the current border, then update the border. The region growing is stopped if the occlusion has achieved the desired size. The size of occlusion is expressed in percentages of the size of the whole *observation* shape. We have tested our algorithm with $0\%, 10\%, 20\%$ and 30% of occlusions [Sánta and Kato, 2014].

We have also compared our results to two recent registration approaches the Bidirectional Affine ICP (AICPBD) [44] and the Coherent Point Drift (CPD) [6] methods. The implementation for AICPBD was obtained from the authors and CPD is publicly available from `https://sites.google.com/site/myronenko/research/cpd`. Both of these implementations are in MATLAB using MEX subroutines implemented in `C++`, thus a runtime comparison is inherently unfair with our approach. Still, CPD has achieved 70 seconds of average CPU time, which is close to our average 90 seconds running times. The AICPBD, however, heavily outperformed both approaches by obtaining a result in 2.4 seconds, on average. For both competitive approaches, we gave the points of the contour as an input

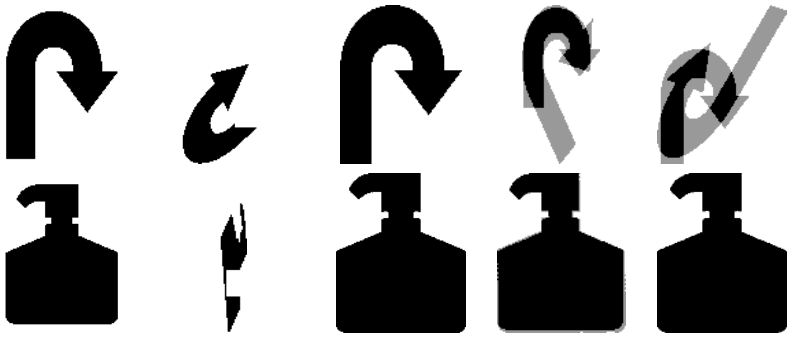


Figure 4.3: Examples from the synthetic dataset. In the last three columns, we denote the template shape with black and the alignment errors with gray colors.

and initialized them with the default parameters of the algorithms. We present our results in Figure 4.4. The registration error has been quantitatively evaluated based on the Dice coefficient of the aligned shapes (denoted by δ):

$$\delta = \frac{|F_r \triangle F_t|}{|F_r| + |F_t|} \cdot 100\%, \tag{4.27}$$

where F_t and F_r are the sets of foreground pixels of the *template* and transformed perfect *observation*, respectively. Since for this dataset, the ground truth transformations are available, we calculated the distance between the true $\mathbf{A}^{-1}$ and the estimated $\hat{\mathbf{A}}^{-1}$ transformations based on all pixels of the perfect *observation* F_o:

$$\epsilon = \frac{1}{|F_o|} \sum_{\bar{\mathbf{p}} \in F_t} \|(\mathbf{A}^{-1} - \hat{\mathbf{A}}^{-1})\bar{\mathbf{p}}\| \tag{4.28}$$

In Figure 4.4, the first row corresponds to the δ, while the second row to the ϵ measure.

The columns of Figure 4.4 show these error measures for various amount of occlusions ($0\%, 10\%, 20\%, 30\%$, respectively). We found that a result with at most 5% δ error gives a visually acceptable alignment. The plots of Figure 4.4 show that the performance of the proposed approach stays under 5% for most of the test cases as long as the occlusion is below 20%. It is also clear that our method outperforms the two other algorithms in accuracy since CPD and AICPBD are quickly reaching the 5% error limit already for 10% of occlusions. All of the algorithms achieved poor results for 30% of occlusions (the last column of Figure 4.4). The same behavior can be observed with the ϵ error (the second row of Figure 4.4). Summarizing the results, the proposed approach gives good alignment as long as the occlusion is below 20% and compares favorably to the two other algorithms [Sánta and Kato, 2014].

The algorithm has been tested on real images, too, and examples of the results can be found in Figure 4.5. In a real application, occlusions could be static (caused by trees or poles) and dynamic (caused by moving cars or walking people). In addition, if some of the segmented shapes from the first image are missing on the second image, we can consider the problem as an occlusion (*e.g.* the fourth row in Figure 4.5). All of these images have been taken in urban environments and the shapes are subject to different types of occlusions. The segmentations are created by classical thresholding and simple region growing algorithms [Sánta and Kato, 2014].

Summarizing these tests, we can say that the algorithm works quite efficiently when the

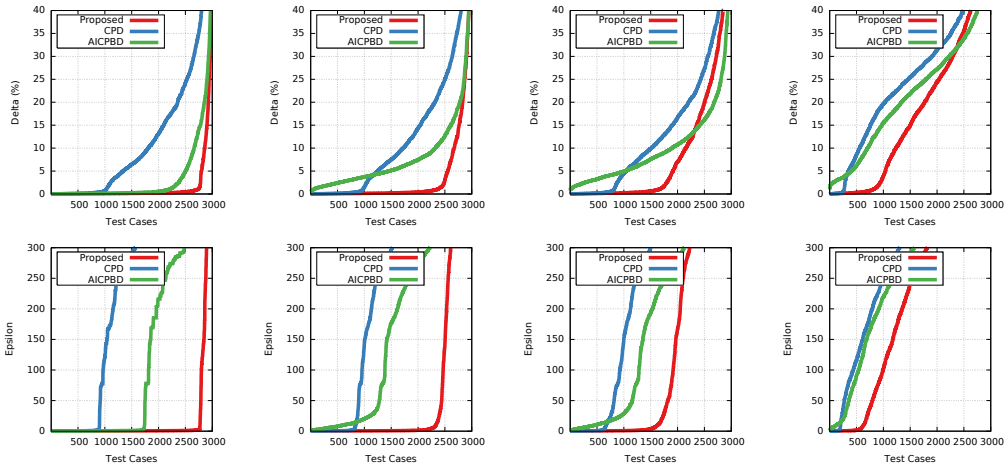


Figure 4.4: The quantitative results of 3000 synthetic test cases, comparing the proposed approach to CPD [6] and AICPBD [44] algorithms. The diagrams in the first row show the performance measured by δ and in the second row by the ϵ metrics. Each column is showing results on the same amount of synthetically generated occlusion ($0\%, 10\%, 20\%, 30\%$, respectively).

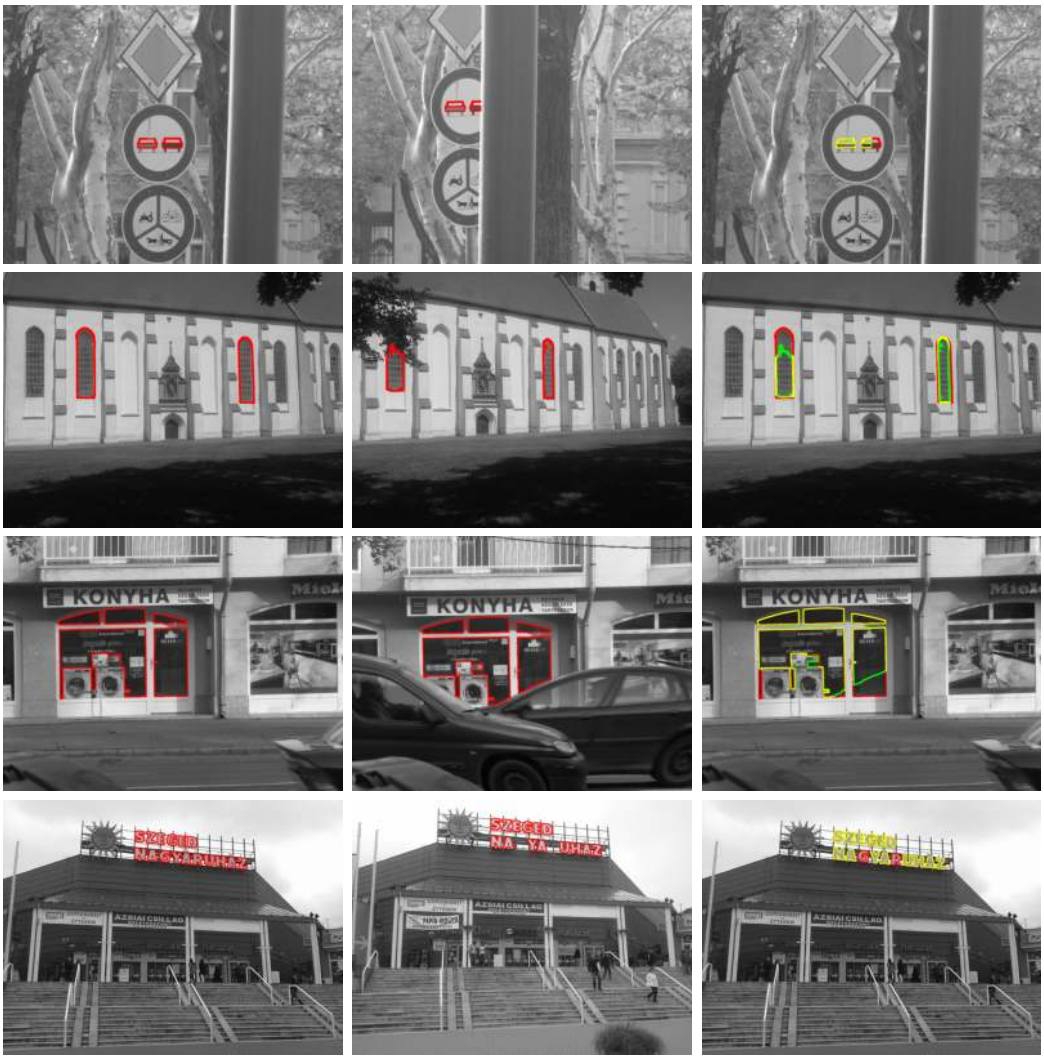
underlying deformation is affine or can be approximated by an affine transformation well, as long as the size of occlusion is under 20%. In addition, the proposed approach can handle shapes with higher perspective distortion (see the second column in Figure 4.5) and it is not affected by moderate segmentation errors.

4.3 Non-rigid Registration of Covariant Functions

In the previous topic, we were focusing on using only geometric information to solve the registration problem. While in many cases this is sufficient, there are problems where the geometric information is not meaningful enough to give constraints on the solution. For an example, see Figure 4.6, which implies that a particular *template* shape can have many different *observations* which are identical as *shapes* but very different as *images* [Sánta and Kato, 2016b].

A convenient way to deal with such difficulties is to introduce more prior information to the problem. When we are dealing with 2D images the simplest idea is to add intensity information, if it is available. In our current topic, this raises two questions: does the proposed algebraic framework could benefit from the availability of radiometric information and how to integrate it into the system? In the following, we will investigate the answers to these questions [Sánta and Kato, 2016b].

Let us assume that we have two intensity functions for the *template* and the *observation* denoted by $T : \mathcal{F}_t \subset \mathbb{R}^2 \to \mathbb{R}$ and $O : \mathcal{F}_o \subset \mathbb{R}^2 \to \mathbb{R}$, respectively. The simplest examples for these functions are the grayscale intensity functions produced by an arbitrary image acquisition device, but it is possible to use more sophisticated, artificially generated functions too [128]. Assuming that the identity relation from Equation (4.1) holds and these functions are *covariant* with respect to the φ transformation, we get the following [Sánta and Kato,



template observation result

Figure 4.5: Results on real images. The first two columns contain the *template* and the *observation* images, respectively with the contours of the segmented shapes. In the last column, the results are visualized, where the contour of the *template* shape is denoted by red, the contour of the transformed *observation* is denoted by green and the intersection is denoted by yellow colors.

2016b]

$$T \circ \varphi = O \tag{4.29}$$

$$T(\mathbf{x}) = O(\varphi(\mathbf{x})) = O(\mathbf{y}). \tag{4.30}$$

The first relation describes the covariance of the intensity functions, while the second one implies that the functions are point-wise invariant in any pair of correspondences. At first, this might look to be a contradiction, but let us clarify using Figure 4.7. In this figure, there is a pair of regions, marked by the green and blue quadrilaterals. The corresponding intensity functions defined over the regions are covariant since they are changed by the application of the transformation. This relation is described by Equation (4.29).

However, if we take an arbitrary pair of corresponding points denoted by the centers of green and blue circles, the intensity values in these points will be the same. This means that

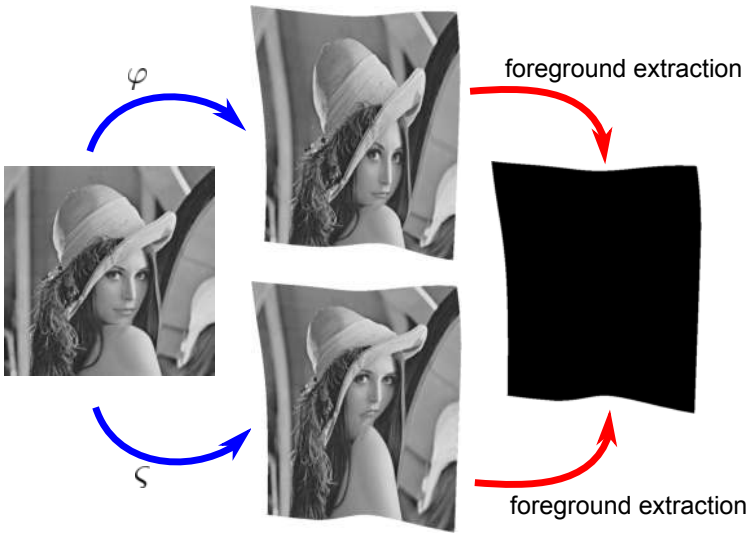


Figure 4.6: The ambiguity of shape-based registration. In this example, two different deformable transformations applied to the same template give very different observations, but they have the same foreground region after segmentation.

Figure 4.7: Two covariant intensity functions defined by green and blue quadrilaterals. The mapping between the regions is denoted by red.

the corresponding property is not affected by the transformation, therefore it is invariant with respect to the deformation. This property is formalized in Equation (4.30) and it can be utilized by the artificial generation of these functions. As an example, in [128] the covariant functions are generated by estimating the Mahalanobis-distance between each point and the centroids of the objects, which maps the points to an affine invariant value.

Once we have the covariant functions, we have to combine this information with the algebraic system [Sánta and Kato, 2016b]. One way is to integrate both sides of Equation (4.29) and apply a set of non-linear functions $\{\upsilon_i \mid \upsilon_i : \mathbb{R} \to \mathbb{R}\}_{i=1}^{\ell}$ to the intensity functions as

$$\int_{\mathcal{F}_t} \upsilon_i(T(\mathbf{x}))|J_\varphi(\mathbf{x})|\, d\mathbf{x} = \int_{\mathcal{F}_o} \upsilon_i(O(\mathbf{y}))\, d\mathbf{y}. \tag{4.31}$$

This type of equation has been used to recover a specific non-linear transformation model having a linear Jacobian in [129]. The main advantage is that the above equations are *linear* in the transformation parameters, hence it is computationally favorable. However, in general, the Jacobian may not contain all transformation parameters (*e.g.* translation parameters vanish in the Jacobian) or it may not be linear (*e.g.* in the case of the Thin

Plate Splines (TPS) deformation model). If the transformation φ is linear, then the Jacobian becomes constant. In such cases, [128, 130, 131] proposes to multiply Equation (4.30) and Equation (4.1) yielding the following form of equations:

$$\int_{\mathcal{F}_t} \varphi(\mathbf{x})\ \upsilon_i(T(\mathbf{x}))\ |J_\varphi(\mathbf{x})|\ d\mathbf{x} = \int_{\mathcal{F}_o} \mathbf{y}\ \upsilon_i(O(\mathbf{y}))\ d\mathbf{y}. \tag{4.32}$$

The main disadvantage of this approach is that because of the υ_i functions have been directly applied to the intensity functions, even a small amount of noise or model mismatch present in the input grayscale values will greatly challenge parameter estimation [129, 131]. This can be handled by adopting a *template-*, transformation-, and generator function-specific noise model into the system, which can be learned from an appropriate training set [130, 131]. However, learning is not straightforward and the training set might not be available for arbitrary applications.

An interesting variant of [131] has been proposed in [130]. Herein, the authors assume that the range of the intensity function has been quantized into the $\{0, \dots, L\}$ discrete set. With this intensity function, the $\{\upsilon_i\}$ set is defined as follows

$$\upsilon_i(z) = \begin{cases} 1 & z = i \\ 0 & \text{otherwise} \end{cases}, \tag{4.33}$$

where $i = 1, \dots, L$. This function set can be considered as a set of *characteristic functions* for each intensity level, yielding L independent integration domains. The main advantage of such representation is to be able to express the noise model as a probability transition matrix [130]. However, this is one of its main drawbacks, too, because the functions move the intensity degradations into the domain of the integrals which could lead to a very inaccurate system.

In the current approach, our aim is to combine the robustness of the geometric approach with the regularizing effect of the covariant functions [Sánta and Kato, 2016b]. This is obtained by multiplying Equation (4.1) and Equation (4.30):

$$\int_{\mathcal{F}_t} \varphi(\mathbf{x})\ T(\mathbf{x})\ |J_\varphi(\mathbf{x})|\ d\mathbf{x} = \int_{\mathcal{F}_o} \mathbf{y}\ O(\mathbf{y})\ d\mathbf{y}. \tag{4.34}$$

Then, similarly to Equation (4.5), we will apply the $\{\omega_i\}$ set to the image coordinates

$$\int_{\mathcal{F}_t} \omega_i(\varphi(\mathbf{x}))\ T(\mathbf{x})\ |J_\varphi(\mathbf{x})|\ d\mathbf{x} = \int_{\mathcal{F}_o} \omega_i(\mathbf{y})\ O(\mathbf{y})\ d\mathbf{y}, \tag{4.35}$$

where $i = 1, \dots, \ell$. This coupled system of equations is solved in the least-squares sense, providing the parameters of the pursued transformation [Sánta and Kato, 2016b].

4.3.1 Numerical Implementation

Unlike to the method described in Section 4.2, herein we will work on pixel representation of the images. In this representation, we have a finite number of pixels with their coordinates and intensity values from the covariant functions. That means, the coordinates and the range of the covariant functions are discretized, thus the equations from Equation (4.35) are only approximately valid. In the current work, we use simple grayscale images where the intensity values are coming from the $[0, 255]$ interval. With this representation the integrals

are approximated by finite sums over the pixels [Sánta and Kato, 2016b]:

$$\sum_{\mathbf{X}\in F_T} \tilde{T}(\mathbf{X})\,\varphi_1(\mathbf{X})^{p_i}\varphi_2(\mathbf{X})^{q_i}\,|J_\varphi(\mathbf{X})| \approx \sum_{\mathbf{Y}\in F_O} \tilde{O}(\mathbf{Y})\,Y_1^{p_i}Y_2^{q_i}, \tag{4.36}$$

where $\mathbf{X},\mathbf{Y}\in\mathbb{R}^2$ are the pixel coordinates; $\tilde{T}$ and $\tilde{O}$ are the discretized intensity functions. The system is then solved in the least-squares sense by iteratively minimizing the algebraic error via the *Levenberg-Marquardt* algorithm. Similarly to the occluded case, the coordinates are normalized into the unit square, and the range of ω_i functions has been scaled into $[-1,1]$. The former normalization is achieved by applying simple normalizing transformations S_T and S_O for the foreground regions of the *template* and the *observation*, respectively. For the latter one we used a set of constants $\{N_i\}_{i=1}^{\ell}$ corresponding to the maximal magnitude integrals over a zero centered disk with a radius $\frac{\sqrt{2}}{2}$ as described in [7]. Furthermore, the range of the grayscale intensities has also been normalized into the $(0,1]$ interval by applying the same scale factor for both images [Sánta and Kato, 2016b].

Applying the normalizations to Equation (4.36), we get the final form of our system:

$$\frac{|S_T|}{N_i}\sum_{\hat{\mathbf{X}}} \tilde{T}(\hat{\mathbf{X}})\,\varphi_1(\hat{X})^{p_i}\varphi_2(\hat{X})^{q_i}\,|J_\varphi(\hat{\mathbf{X}})| \approx \frac{|S_O|}{N_i}\sum_{\hat{\mathbf{Y}}} \tilde{O}(\hat{\mathbf{Y}})\,\hat{Y}_1^{p_i}\hat{Y}_2^{q_i}, \tag{4.37}$$

where $i=1,\ldots,\ell$ and $\hat{X}$ and $\hat{Y}$ are the normalized image coordinates of the *template* and *observation*, respectively. Note that, in order to apply the normalizations to the image coordinates, we have to multiply each side by the determinant of the corresponding normalizing transformation. This value corresponds to the Jacobian of the normalizing transformation [Sánta and Kato, 2016b].

Our experiments have shown that the runtime of the algorithm can be reduced by solving the problem with a coarse-to-fine hierarchical strategy [2]. In our implementation, we use Gaussian image pyramids computed by the algorithm described in [132]. For each level, we construct the system from Equation (4.37), then solve it using the outcome of the previous level as initialization. On the first level, the algorithm is initialized with the identity transformation [Sánta and Kato, 2016b].

In our experiments, we used affine and TPS transformations. Similarly to the 3D case described in Section 3.2.2, the TPS transformation is defined using a set of control points $\{\mathbf{c}_k\}_{k=1}^{K}\subset R^2$ and two sets of parameters: $a_{ij},w_{ik}\in\mathbb{R}$, where $i=1,2$, $j=1,\ldots,3$ and $k=1,\ldots,K$. ς is composed of two coordinate-wise transformation functions $\varsigma(\mathbf{x}) = [\varsigma_1(\mathbf{x}),\varsigma_2(\mathbf{x})]^T$ in the form of

$$\varsigma_i(\mathbf{x}) = a_{i1}x_1 + a_{i2}x_2 + a_{i3} + \sum_{k=1}^{K} w_{ik}Q(\|\mathbf{x}-\mathbf{c}_k\|), \tag{4.38}$$

where $i=1,2$ and $Q:\mathbb{R}\to\mathbb{R}$ is the *radial basis function* defined in 2D as $Q(r)=r^2\log r^2$ [7, 33]. The model has $N=2K+6$ parameters. Note that, the local parameters have to satisfy the same additional constraints as described in Equation (3.16) for 3D.

Of course, the solution of Equation (4.37) will act between the normalized coordinates, hence the result has to be denormalized to obtain the parameters of the transformation acting between the original images. For the affine transformation, the denormalization can be achieved by multiplying the obtained solution by S_T and S_O. However, since S_T and S_O contain non-uniform scalings, the TPS model cannot be denormalized by scaling its

parameters only, it will be a compound transformation:

$$\varphi(\mathbf{x}) = S_O^{-1} \circ \varsigma \circ S_T. \tag{4.39}$$

The outline of the algorithm can be found in Algorithm 4.2. The computational complexity of the proposed approach is linear in the size of the images, since the overall complexity is $\mathcal{O}(L(C|F_T| + |F_O|))$, where L denotes the number of levels in the Gaussian pyramid, C denotes the number of function calls made by the *Levenberg-Marquardt* solver, $|F_T|$ and $|F_O|$ correspond to the number of pixels in the input image regions [Sánta and Kato, 2016b].

Algorithm 4.2 Pseudo code of the 2D registration algorithm using covariant functions

Input: *template* and *observation* images
Output: The aligning transformation φ

1: Construct the Gaussian pyramids from the input images.
2: **for** each level of the pyramids **do**
3: Extract the pixel coordinates and intensity values.
4: Construct the system of equations Equation (4.37).
5: Solve the system in the least-squares sense by the *Levenberg-Marquardt* algorithm. On the first level initialize the solver by the identity transformation and on each other level use the solution from the previous one.
6: **end for**
7: Construct the final transformation as described in Section 4.3.1.

4.3.2 Experimental Results

In the following, we will summarize our experimental results with the covariant registration algorithm. The main goal of these experiments was to present the capabilities of the algorithm by registering pairs of synthetically generated and real images. The robustness of the algorithm against image noise is also tested for both models. We compare our affine results to the algebraic method proposed in [130] and for the deformable case, we compared our results to the outcomes of two recent registration methods [11, 12]. Finally, for the TPS model, we also study the significance of the control point positions.

The estimated transformations have been evaluated by the normalized cross correlation [2]

$$NCC = \frac{\sum_{\mathbf{x}}(F_O(\mathbf{x}) - \overline{F_O})((F_R(\mathbf{x}) - \overline{F_R}))}{\sqrt{\sum_{\mathbf{x}}(F_O(\mathbf{x}) - \overline{F_O})^2}\sqrt{\sum_{\mathbf{x}}(F_R(\mathbf{x}) - \overline{F_R})^2}} \tag{4.40}$$

and the normalized root-mean-square (RMS) distance

$$RMS = \sqrt{\frac{1}{n}\sum_{x}\left(\frac{F_O(\mathbf{x}) - F_R(\mathbf{x})}{255}\right)^2} \tag{4.41}$$

between the intensities of the *observation* F_O and the transformed *template* F_R. $\overline{F_O}$ and $\overline{F_R}$ denote the mean intensity value. We observed that the RMS metric is more sensitive to the intensity differences, but the NCC metric is more meaningful to describe the results. Observing the results qualitatively, we found that a registration with $NCC > 0.85$ corresponds to a visually acceptable, with $NCC > 0.9$ to a visually good and with $NCC > 0.95$ to a visually excellent registration. For the affine results, we estimated the ϵ metric from Equation (4.28) as well.

The algorithm has been implemented in Matlab using a MEX interface for the ω function calculation. We used the *Levenberg-Marquardt* implementation of Lourakis [108]. All tests have been run on a regular laptop with Core i5 2.5 GHz architecture (the implementation used only one processing core).

For the affine method, the integrals have been rearranged according to the scheme described in [122]. Using this approach, the integrals over the pixel coordinates will become independent from the parameters, therefore it could be precalculated. Hence, in each iteration of the solver, we do not have to iterate through the pixels, which greatly reduces the computational time.

Experiments with Affine Transformation

First, let us present our results on affine registration. Herein, the main aim was to show the capabilities of the proposed approach with respect to the previous linear algebraic framework published in [130].

In order to test the proposed approach, we generated a synthetic dataset containing 100 images. To obtain a deformed image we used randomly generated transformations applied to a set of *template* images. The parameters of the affine transformation have been randomly selected from uniform distribution according to the following rules: rotation from $[0, 2\pi)$, scale from $[0.5, 1.5]$ and shear from $[-1, 1]$. For the image generation, we used nearest neighbor and bicubic interpolations resulting in two different sets. In the following, we will refer to these sets as `affine_nearest` and `affine_bicubic`, respectively. Emphasizing the difference between the interpolation methods might seem to be unnecessary from the point of view of a real application, but it will illustrate well the difference between the proposed polynomial and linear [130] methods.

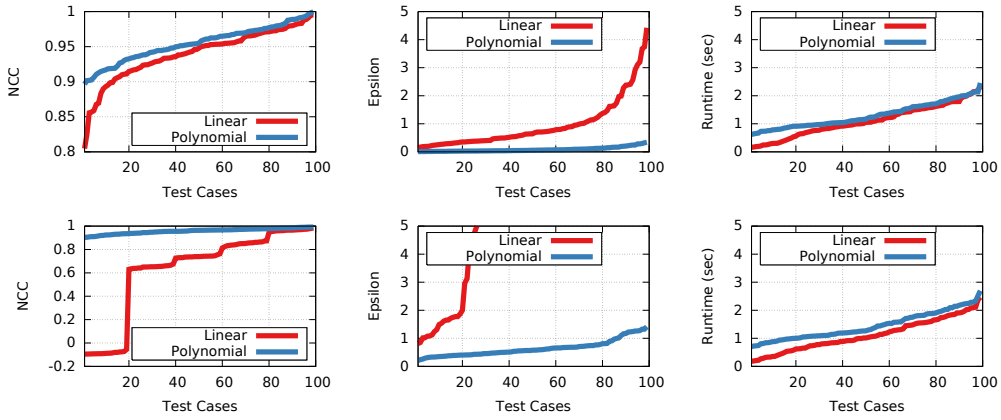


Figure 4.8: A comparison between the proposed polynomial affine method and the linear model presented in [130]. The rows correspond to the `affine_nearest` and the `affine_bicubic` datasets, respectively. The first two column show the NCC and the ϵ metrics, respectively, and in the third, we compare the running times.

Our results on the synthetic test are presented in Figure 4.8. Unfortunately, for the linear method there is no public implementation available, therefore we used own implementation without any additional training applied. According to the experiments, the linear method is very unstable, even a small deviation in the intensity values, caused by the different interpolation methods, decreases the accuracy drastically. Observing the NCC values for the bicubic set, each ”step” in the plot corresponds to a different *template*. This also confirms

that the trained noise model should depend on particular *template* image [130]. Surprisingly, the running times are nearly the same, although the linear approach is slightly faster as expected.

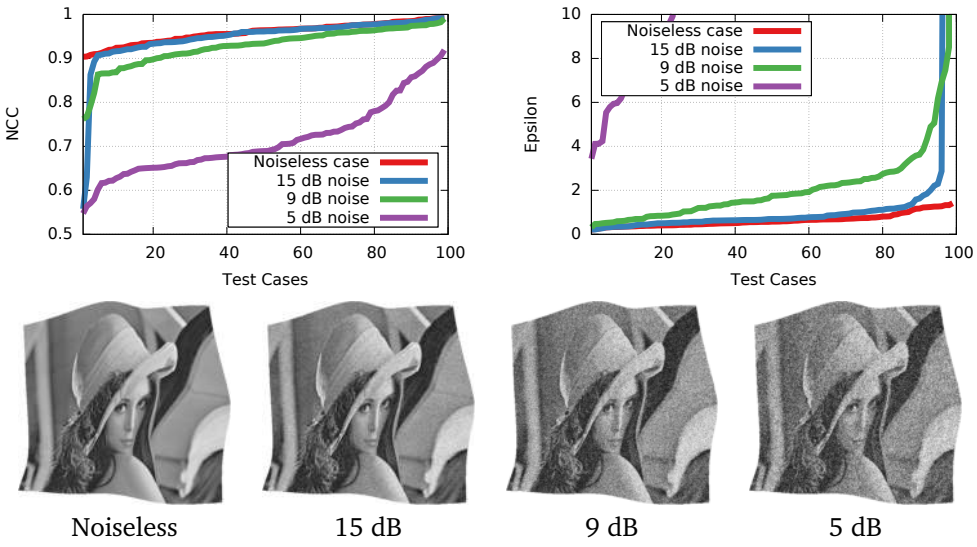


Figure 4.9: Robustness test results using affine transformation. The first two rows contain plots of NCC and RMS metrics, while the last row show some examples from the dataset.

In practice, the input images are affected by various types of noise. In order to study the behavior of the algorithm in the presence of Gaussian noise, we have created a dataset with additive zero-mean noise. For noise generation, we used the variances $\sigma^2 \in \{0.01, 0.05, 0.1\}$. These values are corresponding to 15, 9, 5 dB signal-to-noise (SNR) ratios, respectively. Note that, for generating the noisy images, we used the bicubic set only. Sample images and results can be found in Figure 4.9. The plots clearly show that the algorithm provides acceptable results as long as the SNR is higher than 9 dB.

Experiments with Thin Plate Spline Transformation

Similarly to the affine case, we used two randomly generated synthetic datasets, each of them containing 250 images [Sánta and Kato, 2016b]. The TPS transformations have been generated using the following procedure: First, we randomly sampled a set of initial control point positions, then using random displacements we constructed interpolating thin plate splines. As usual with TPS models, the generated transformations are required to preserve the topology, thus the transformation has been rejected when its Jacobian was negative or nearly zero. We used 15 and 25 control points for the first and second set, respectively. The displacements have been generated as random samples from a zero mean normal distribution with 15 pixels standard deviation. Some examples can be found in Figure 4.10. Hereafter, we will refer to the first set as `Set_15` and to the second set as `Set_25`.

In the first experiment of this block, the alignments have been estimated using the same control point locations as for the generation. The results of this experiment are summarized in Table 4.1. According to our bounds on the error metrics, the algorithm successfully reconstructed the deformations for `Set_15` (with 0.95 mean and median NCC values) and achieved good results for `Set_25` (with 0.91 mean and 0.90 median NCC values). This difference is possibly caused by the radiometric distortions and sampling errors introduced by the higher geometric distortion of the transformations [Sánta and Kato, 2016b]. We

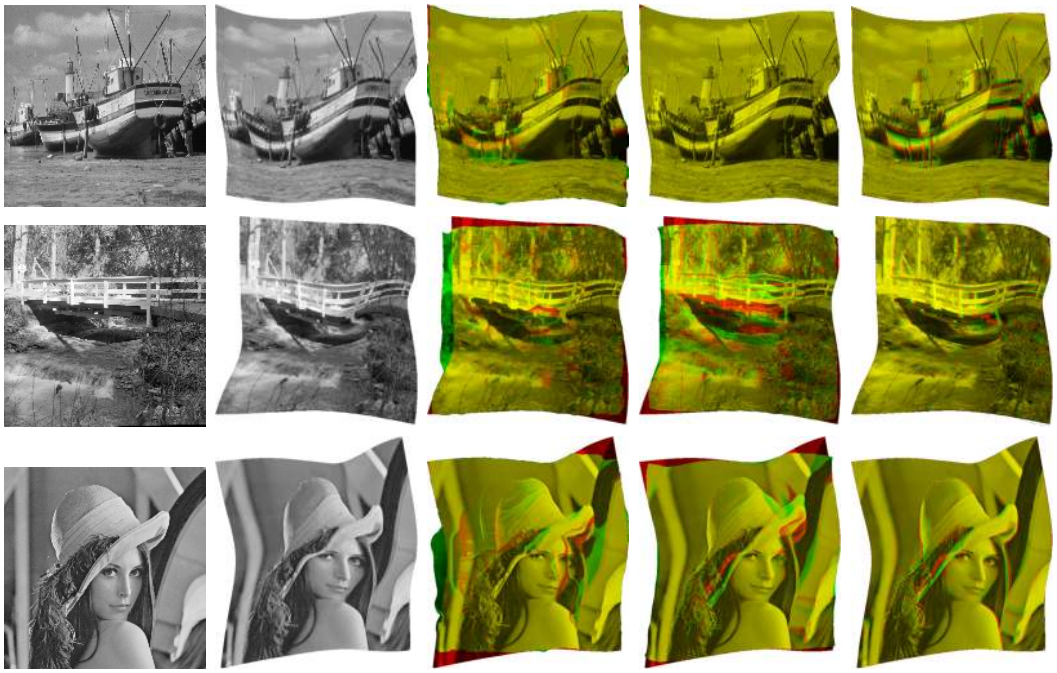


Figure 4.10: Sample images from the synthetic datasets and comparison between DROP [11], SPECDEM [12] and the proposed approach. For each row, the first two columns show the *template* and *observation*, respectively. In subsequent columns, the results of DROP, SPECDEM, and the proposed algorithms are presented as a combined RGB image: the red channel contains the *observation* and the green channel contains the transformed *template*.

		Set_15	Set_25	Overall
NCC	μ	0.95	0.91	0.93
	m	0.95	0.90	0.93
	σ	0.02	0.04	0.04
RMS	μ	0.08	0.11	0.09
	m	0.08	0.11	0.09
	σ	0.02	0.02	0.02
Runtime (sec)	μ	20.88	51.28	40.45
	m	18.43	60.03	34.70
	σ	5.63	26.26	27.28

Table 4.1: Results of the first experiment using the same numbers and locations for the control points as for the generation. Quantitative evaluation is given in terms of NCC and RMS metrics (μ – mean, m – median, σ – standard deviation)

used 21 and 31 ω functions for `Set_15` and `Set_25`, respectively. The runtime was around 1 minute. Note that, for this experiment, we used only one level for the Gaussian pyramids, because we experienced that for cases with lower number of control points, the algorithm does not benefit from the additional levels.

In the next experiment, we tried three different grid configurations for the positions of the control points. These grids contained 25, 49 and 100 points placed uniformly over the bounding box of the *template*. For each configuration, we used 66, 114 and 216 ω functions. Three levels have been used for the Gaussian pyramids which made our algorithm three times faster for the 49 and 100 point grids, while remained the same for the 25 grid. The summary of the experiments can be found in Figure 4.11.

From these results, we can conclude that whenever the optimal control point locations are unknown, the algorithm can solve the problem using a much finer grid. As an example, for the generation of `Set_15`, we used 15 control points, however, it was necessary to use a grid with 49 or 100 control points to obtain a visually good result. The algorithm achieved better results on `Set_15` than on `Set_25`, as excepted. However, increasing the number of control points could also increase the precision of the algorithm, at the price of higher computational complexity. None of the configurations could reach the level of accuracy we obtained using the exact control point positions [Sánta and Kato, 2016b].

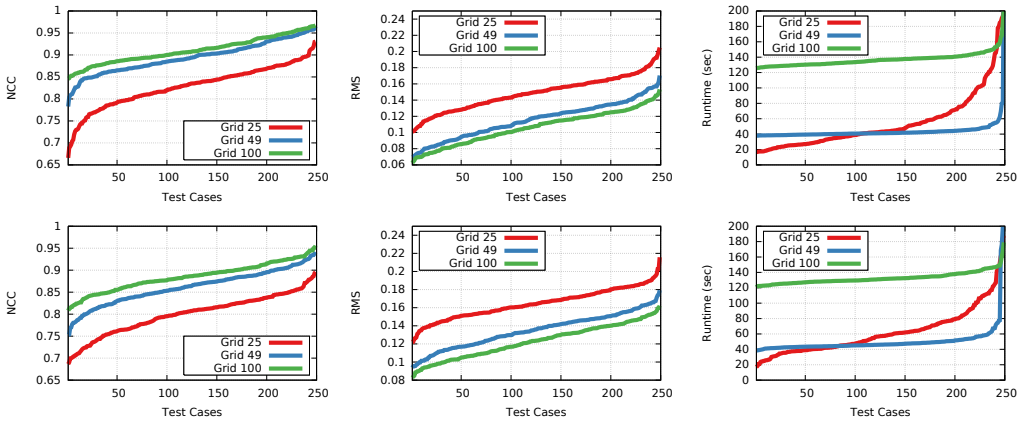


Figure 4.11: The results of the control point placement test using 25, 49 and 100 points placed on a uniform grid. The first row contains the results on `Set_15` and the second for `Set_25`. The columns show plots of the NCC, RMS, and runtime (in seconds) statistics.

We have also compared our method to two recent deformable registration algorithms the Deformable Image Registration using Discrete Optimization (DROP) [11] and the Spectral Demon's Algorithm (SPECDEM) [12]. The implementations of these algorithms have been provided by the authors. DROP uses a Markov Random Field formulation to describe the registration problem as a minimal cost graph problem. The deformation is modeled using a free-form deformation (FFD) model with discrete displacements. The SPECDEM approach is based on the popular Log-Demons algorithm [50]. This approach aims to find diffeomorphic point-wise correspondence between the images [12]. Each algorithm has been applied to our synthetic dataset and initialized with their default parameters [Sánta and Kato, 2016b].

In Figure 4.12, we compared the accuracy and runtimes of the DROP and SPECDEM algorithms with our best results using 100 control points, as well as with the results obtained by using the exact control point locations. Note that, the transformation models used by DROP and SPECDEM have much higher degrees of freedom than our TPS model. In spite of this, both approaches achieved inferior results in terms of alignment accuracy. Remark that, these approaches use a more general, non-parametric transformation model, while our method works only with parametric deformation models. Therefore, [11, 12] would certainly outperform our method when a parametric TPS model cannot be used. Unfortunately, DROP had a pure `C++` implementation only, hence runtime comparison was inherently unfair with the other two algorithms. However, SPECDEM is implemented in Matlab/MEX similarly to our approach. According to Figure 4.12, our algorithm was two times faster than SPECDEM on this dataset.

The robustness of the TPS based algorithm has also been tested similarly to the affine case. The results can be found in Figure 4.13. Just as in the affine case, the algorithm

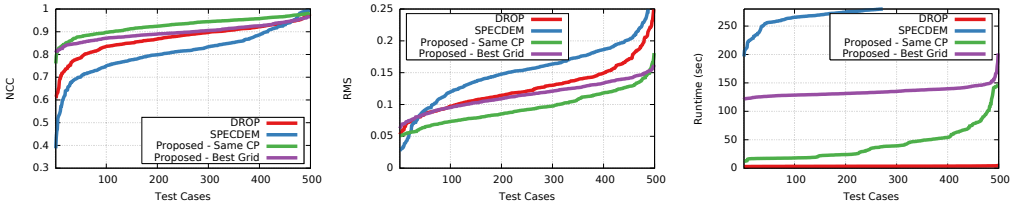


Figure 4.12: Comparative test results. Columns show plots of NCC, RMS, and runtime (in seconds) statistics.

provides acceptable results as long as the SNR is higher than 9 dB.

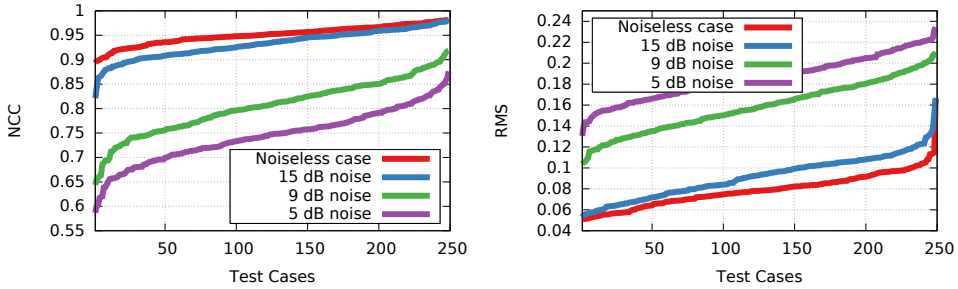


Figure 4.13: Robustness test results using TPS. The columns show plots for the NCC and the RMS metrics, respectively.

Real Datasets In the following, we will present our results on cloth and paper deformations. Many real applications, such as motion estimation, tracking or 3D reconstruction require dense correspondence between the 2D observations of such types of objects [8, 9]. The task is challenged not just by the non-linear geometric deformations, but by radiometric deformations too, caused by *e.g.* inner shadows of the objects or changes in the illumination.

We tested our algorithm on two datasets. In the first dataset, we used photos taken from shirts containing textured patterns [Sánta and Kato, 2016b]. Each shirt had an undistorted and several distorted photos. The undistorted image served as the *template*, while the distorted images were the *observation*. Before registration, the outlines of the patterns have been segmented by simple color-based thresholding. For the registration, we used 100 control points placed on a uniform grid. The algorithm achieved promising results, despite of the presence of higher radiometric deformations caused by the inner shadows of the material. See the first row of Figure 4.14, for example. For the second dataset, we used the paper-bend set from [9]. The proposed method achieved acceptable results on the textured paper containing repetitive elements (see Figure 4.14). The average NCC and RMS values for the shirt images were 0.938 and 0.045, respectively, and 0.904 and 0.092 for the paper bend images [Sánta and Kato, 2016b].

4.4 Conclusion

In this chapter, we have addressed two challenges of registering 2D images with an algebraic framework. First, we have proposed a novel algorithm in order to deal with higher occlusions for affine deformations [Sánta and Kato, 2014]. The basic idea is to represent the shapes as

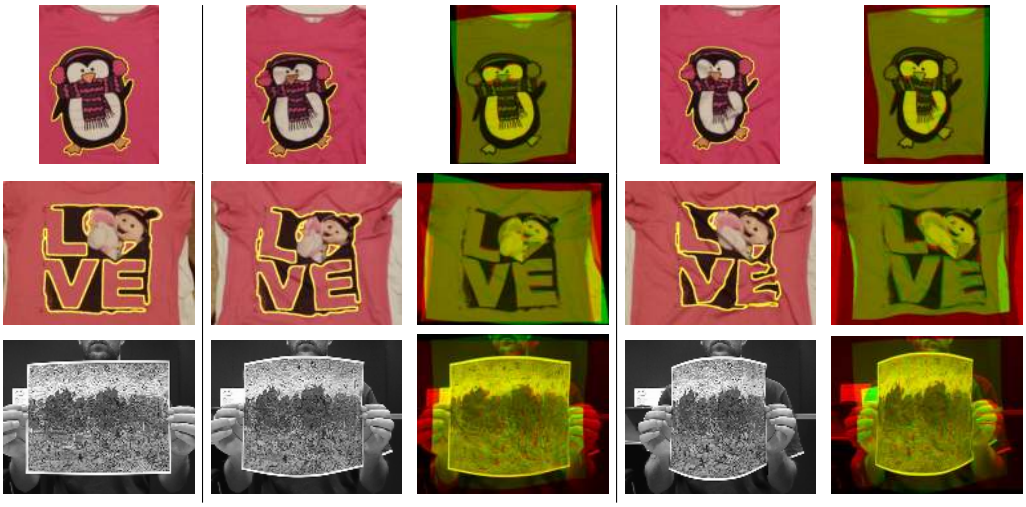


Figure 4.14: Samples from the real experiments. The first two rows contain images of shirts with textured patches. Each block contains three images: *template*, *observation* and an overlay of the transformed *template* over the *observation*. The images have been acquired with an ordinary camera. The last row shows samples from the bending paper dataset obtained from [9]. The first column contains the *template*, then we show two different stages of the bending with the *observations* and the overlay images.

polygons, then iteratively estimate the occluded regions using simple boolean operators while solving a system of equations in the least-square sense. Our method compares favorably to two recent registration algorithms [6, 44] and the algorithm has been tested on real images, too.

Second, we have proposed an algorithm for non-rigid registration of image patches using a similar algebraic framework [Sánta and Kato, 2016b]. The main aim is to handle covariant functions, *e.g.* intensity information, to reduce the ambiguity of the deformable registration. While any parametric model could be used within the framework, in this work we focused on affine model for linear and TPS model for deformable registration. Experimental results show the capabilities of the proposed approach both on synthetic and real images. Moreover, the algorithm compares favorably to two recent state of the art methods published in [11] and [12].

Further research aims to increase the robustness against radiometric deformations by investigating the applicability of region-based descriptors. In the recent years, several dense descriptors have been proposed for solving various related tasks, *e.g.* spectral methods [12], or application specific descriptors obtained by learning methods [14]. The main benefit of using such functions over the grayscale values is the robustness against the higher radiometric deformations and the capability of handling multimodal cases [14]. Another interesting way is to combine the proposed methods and increase the robustness of the covariant function based algorithm against higher occlusions.

Chapter 5

Ad-hoc Mobile Camera Network Calibration

In this chapter, we present a novel approach for calibrating ad-hoc camera networks, composed of a set of smartphones with integrated cameras [Sánta and Kato, 2013b]. In the last couple of years, smartphones have become very popular. Introduced as a new development platform, yielded many new possible applications. The main advantages of smartphones are the availability of a powerful embedded processor, networking capabilities and multiple types of sensors (*e.g.* GPS, accelerometer, gyroscope). Using a set of mobile phones, a distributed, highly scalable intelligent camera network can be created, which is desirable in many tasks of vision based applications, like localization, reconstruction, object recognition and tracking or HDR imaging of dynamic events. Many of these applications require a calibrated camera system. The topic of calibrating camera networks have been reviewed in Section 2.2. In the following, we will observe the applicability of these methods on this so called *smart camera network*.

Since mobile phones typically have built-in fixed focus cameras, their intrinsic parameters can be easily calibrated using various algorithms and simple calibration patterns [74] or self-calibration approaches [75, 76]. Therefore, in this work, we will focus on the estimation of extrinsic camera parameters (*i.e.* pose estimation) of such ad-hoc mobile camera networks.

Calibration of other types of (non ad-hoc) smart camera networks, which are typically used in surveillance applications, has been studied in the literature. However, there are substantial differences compared to ad-hoc mobile camera networks. First of all, surveillance camera systems are designed with optimal coverage of a particular scene, hence the network topology and individual camera positions in the 3D world are known a priori – at least with a good approximation. Furthermore, such cameras have fixed position and orientation (or known range of orientations in case of PTZ cameras), thus pose estimation can be solved during installation by taking pictures of specific calibration targets. The main problems related to the localization of multi-camera systems or vision based wireless sensor networks [79, 80, 133–136] are determining the neighboring cameras from the overlapping areas (*e.g.* via vision graph construction [79, 80, 137, 138] or using sensor data), metric calibration and pose estimation of the camera network [80, 134]. In such environments, these problems have to be solved with limited computation and communication between the nodes, because of the limited power supplies.

Smartphones are less limited and they also have better cameras, hence a wider range of

algorithms can be used. In the case of ad-hoc camera networks, pose has to be estimated from the actual images taken by the individual mobile cameras and special calibration patterns are usually not available.

5.1 The Calibration Framework

The aim of the proposed method is to localize the network in a three-dimensional (3D) scene with respect to a 3D structure [Sánta and Kato, 2013b]. In other words, the origin of the estimated network will be attached to a well-defined 3D location. Since our initial assumption excludes the availability of a calibration pattern, we also have to reconstruct such structure using the camera images. A proper review on 3D reconstruction is far beyond the scope of the current work, herein we will focus on the major tasks only.

The proposed approach has three main steps:

1. Estimation of the relative pose of the cameras within the network w.r.t. an arbitrary *main camera*.

2. Computation of the absolute pose between the *main camera* and a 3D structure.

3. Estimation of the relative scale factors of the previous two coordinate systems yielding a fully calibrated network w.r.t. the 3D scene.

The steps are visualized in Figure 5.1. Note that, the first two steps are independent, therefore, they could be estimated in parallel and synchronized only in the last step.

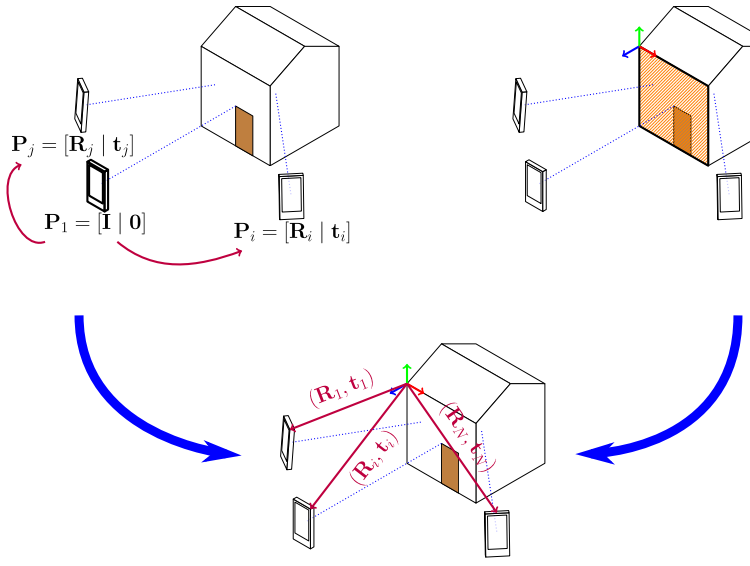


Figure 5.1: The proposed pipeline for camera network calibration.

In the following, we will assume that the method gets the initial network topology as a *vision graph* from Equation (2.9). Establishing such structure in a mobile environment is also an interesting topic. A recent method has been published in [81], where the vision graph is constructed by combining visual and sensor information (*e.g.* GPS).

5.1.1 Relative Pose of the Cameras Within the Network

The first step of the proposed method built on a standard relative pose estimation pipeline, which is described in Section 2.2. As we mentioned above, we will choose an arbitrary *main camera*, which will be the origin in the current phase. Since our method is distributed, we can assume without loss of generality, that $i = 1$ is the *main camera*. Now, let us determine the relative pose of the cameras w.r.t. a *main camera* [Sánta and Kato, 2013b].

According to the review from Section 2.2.1, this problem can be easily solved by estimating the essential matrices acting between the *main camera* and the other cameras. For that purpose, ASIFT [29] features are extracted by each mobile in a distributed, parallel manner, and then the descriptors are sent to the *main camera*, which subsequently computes correspondences and estimates the essential matrices [Sánta and Kato, 2013b]. The necessary amount of communication depends on the size of the adopted descriptor, it is typically between 64 and 128 for ASIFT features. Thus M_i extracted descriptors yields $\mathcal{O}(M_i C)$ bytes communication for the i^{th} camera, where C is the size of a descriptor vector. In our experiments, M_i was at most 500 and $C = 64$. Each packet has to be sent only once to the *main camera* [Sánta and Kato, 2013b]. Of course, the size of a packet could be reduced using compression methods described in [79, 80, 138], but this is beyond the scope of the current work.

Since the intrinsic parameters of the cameras are known, we send the normalized coordinates of the extracted feature point coordinates as $\mathbf{K}_i^{-1}\bar{\mathbf{x}}_j^i$. Following the review in Section 2.2.1, the essential matrix $\mathbf{E}_i \in \mathbb{R}^{3\times 3}$ can be determined by several methods. In the current work, we use the *Normalized 8-point algorithm* taking into account that $\mathbf{E}_i$ is rank 2 with two identical singular values [72] satisfying

$$\hat{\mathbf{x}}^{i^T}\mathbf{E}_i\hat{\mathbf{x}} = 0, \tag{5.1}$$

for all corresponding normalized point pairs $\bar{\mathbf{x}}$ of the *main camera* and $\bar{\mathbf{x}}^i$ of the i^{th} camera. Recall that, the main advantage of this method over other approaches is that it returns only one solution for the essential matrix. However, we have to use at least 8 point correspondences [Sánta and Kato, 2013b].

Assuming that the *main camera* matrix is of a canonical form $\hat{\mathbf{P}}_1 = [\mathbf{I} \mid \mathbf{0}]$, the other (normalized) camera matrices can be determined from the essential matrices [72]:

$$\hat{\mathbf{P}}_i = [\mathbf{R}_i \mid \mathbf{R}_i\mathbf{t}_i] \qquad i = 2, \ldots, N, \tag{5.2}$$

where $\mathbf{R}_i \in \mathbb{SO}(3)$ is the relative rotation and $\mathbf{t}_i \in \mathbb{R}^3$ is the relative translation of the camera $\mathbf{P}_i$ w.r.t. $\mathbf{P}_1$.

Once we have the pair-wise relative poses, we can reconstruct several 3D points by triangulating the extracted landmarks for each pairs of cameras. One of the most convenient measures for describing the accuracy of an extracted camera pose is the reprojection error of the triangulated points, estimated as the Euclidean distance between the original and the reprojected landmark positions. While it is necessary for a good pose to have a minimal reprojection error, the landmark based measurements are usually challenged by the inaccuracy and matching ambiguities of the landmark extraction process. For this particular case, it means that a small reprojection value does not correspond to a good pose, although it can be used to eliminate the trivially inferior results. In the current approach, an upper bound has been given for the maximal acceptable reprojection error and the landmarks have

been filtered accordingly. In our experiments, we used a 10 pixels limit. If the number of acceptable landmarks is smaller than 11, the pose has been marked as bad. This threshold corresponds to the minimal number of points for a simple bundle adjustment, considering two cameras only.

Now, each camera is calibrated w.r.t. the *main camera* $\mathbf{P}_1$, but all of them is determined up to a unique scale [Sánta and Kato, 2013b]. To make these scale factors consistent, we will determine the relative scale between each camera pair $(\hat{\mathbf{P}}_1, \hat{\mathbf{P}}_i), i = 2, \ldots, N$, by making use of the *Two point algorithm* described in Section 2.2.2. Herein, the cameras are added iteratively to the network, while we estimate the relative scales using triangulated pairs of points [73]. The estimation is based on the formula from Equation (2.14). Note that using this method we need correspondences from three cameras only to estimate the scale factor of any camera in the network [Sánta and Kato, 2013b].

The last step is *bundle adjustment* [72], which simultaneously refines camera parameters and 3D point coordinates by minimizing the overall reprojection error of the reconstructed 3D points:

$$\min_{\bar{\mathbf{X}}_j^i, \mathbf{P}_i} \sum_{i=1}^{N} \sum_{j=1}^{n_i} \|\mathbf{P}_i \bar{\mathbf{X}}_j^i - \bar{\mathbf{x}}_j^i\|^2, \tag{5.3}$$

where $\mathbf{P}_i \bar{\mathbf{X}}_j^i$ is the back projection of $\bar{\mathbf{X}}_j^i$ in the i^{th} camera, while $\bar{\mathbf{x}}_j^i$ denotes the true pixel coordinates. For further information, please refer to Section 2.2.3. Scale estimation and bundle adjustment are computed in the main mobile only, thus there is no communication in these steps [Sánta and Kato, 2013b].

5.1.2 Localizing the Camera Network in the 3D Scene

In the following, we will investigate how to relate the camera network to the 3D scene. According to our initial assumption 3D structures are not available at beginning of the pipeline, therefore we have to reconstruct one or more element of the 3D scene first. After that, the world coordinate system could be attached to any of those structures.

In general, we need at least two calibrated cameras to obtain a metric reconstruction of 3D objects [72]. Then using reliable geometric correspondences between the camera images; points, planes or even complex surfaces can be reconstructed [139]. Since the camera network has been calibrated in the previous step, we only have to deal with establishing of geometric correspondences. The point-wise correspondences used for camera calibration could be good candidates for such task, however attaching the origin of the coordinate system to a single point makes the calibration unreliable and introduces accuracy problems as well. Moreover, since a single point does not have any orientation, we have to choose additional points to give the major axes of the system, which makes the whole process more ambiguous.

Using planes instead of points, however, leads to a more stable approach. A plane inherently defines two of the three major axes and from the projected patches in the camera images very accurate geometric correspondences could be obtained via affine or projective homographies. Such homographies established by using 3 or 4 coplanar feature points (for affine and projective transformations, respectively) [72] or by making use of a patch based registration algorithm as discussed in Chapter 4. However, these approaches both assume the availability of reliable point- or patch-wise correspondences between the camera images, which is hard to extract in a highly repetitive, urban environment (*e.g.* flats, windows, brick walls, see Figure 5.3). This ambiguity caused by the regular structures is a great challenge

for classic registration methods, therefore we present a new approach, where the regularity is handled as an advantage instead of a drawback.

First of all, in this special case it is possible to obtain 3D structures by using one camera only [140–142]. This is achieved through handling the repetitive patterns as Transform Invariant Low-rank Texture (TILT) patches [142].

Let us consider a 2D texture as a function $\mathcal{I}_0(x_1, x_2) : \mathbb{R}^2 \to \mathbb{R}$. According to the formal definition from [142], $\mathcal{I}_0$ is low-rank if the family of one-dimensional function $\{\mathcal{I}_0(x_1, z) \mid z \in \mathbb{R}\}$ spans a finite low-dimensional linear subspace as

$$\dim(\text{span}\{\mathcal{I}_0(x_1, z) \mid z \in \mathbb{R}\}) \leq k, \tag{5.4}$$

where k is a small positive integer. In practice, we are dealing with discrete images, thus the $\mathcal{I}_0$ texture is sampled on a finite discrete grid of size $m \times n$, which could be considered as a matrix with real values [142]. Therefore, the formal definition can be interpreted as

$$\text{rank}(I_0) \leq k \ll \min(m, n), \tag{5.5}$$

where I_0 is the discretized matrix.

In most cases, we observe a transformed version of $\mathcal{I}_\prime$, *e.g.* in our case, it will be deformed by a projective homography. Let us denote the discrete image of $\mathcal{I}_\prime$ in the i^{th} camera as I_i. Hence, there is a $\mathbf{H}_i$ planar homography, which aligns I_0 to I_i as

$$I_i = \mathbf{H}_i(I_0). \tag{5.6}$$

In practice, image patches could be challenged by various types of noise and degradations, thus Equation (5.6) might not be true. The degradations could be expressed as a matrix addition:

$$I_i = \mathbf{H}_i(I_0) + \mathbf{S}_i, \tag{5.7}$$

where $\mathbf{S}_i$ contains all differences between the true and the projected low-rank patch. As long as $\mathbf{S}_i$ is sparse, we can reconstruct I_0 from I_i by solving the following problem [142]

$$\min_{I_0, \mathbf{S}_i, \mathbf{H}_i} \quad \text{rank}(I_0) + \gamma \|\mathbf{S}_i\|_0 \quad \text{s. t.} \quad \mathbf{H}_i^{-1}(I_i) = I_0 + \mathbf{S}_i. \tag{5.8}$$

Such robust rank minimization problem could be solved by making use of the Augmented Lagrange Multipliers (ALM) method as described in [142]. Using the extracted elements, we can define a TILT feature as

$$T = (I_i, \mathbf{H}_i, \mathbf{S}_i), \tag{5.9}$$

where I_i is the deformed patch in camera i, while $\mathbf{H}_i$ and $\mathbf{S}_i$ denote the extracted homography and sparse error matrices, respectively. Once a TILT feature is extracted, we can easily relate the camera i w.r.t. I_0. In the current approach, we use I_0 as the 3D structure for the origin of the final coordinate system [Sánta and Kato, 2013b].

Using these results, the second step of the pipeline will have the following sub-steps:

1. Extract a low-rank candidate patch from the image of the *main camera*.
2. Apply the TILT algorithm from [142] to establish a TILT feature in form of Equation (5.9).
3. Relate the *main camera* using the extracted homography.

Since the steps are executed only in the main mobile, there is no need for communication. However, the scale of the extracted coordinate system will be independent from the scale of the network's coordinate system, thus we have to synchronize them in a subsequent step. In the following, we will refer to the currently estimated system as *plane coordinate system*. While Step 2 is straightforward, Step 1 and 3 need further elaboration. In the current approach, we select the low-rank candidate patch by user interaction (selecting its bounding box). While it is adequate for our experiments, we have to mention that automatic detection of repetitive and/or symmetric patterns is also possible [141, 143].

In Step 3, we assume that the world coordinate system is attached to the top left corner of I_0 and it corresponds to the $Z = 0$ plane. This means that the relative pose of $\mathbf{P}_1$ can be factorized from the extracted $\mathbf{H}_1$ homography by making use of a method from Sturm [144].

Since all points in I_0 has $Z = 0$ third coordinate, the planar homography $\mathbf{H}_1$ is composed (up to scale) from the relative camera matrix as

$$\mathbf{H}_1 = \mathbf{K}_1[\mathbf{R}_{3\times 2} \mid \mathbf{R}\mathbf{t}], \tag{5.10}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the relative pose of $\mathbf{P}_1$ w.r.t. I_0. Since $\mathbf{K}_1$ is known, we can write:

$$\mathbf{M} = \mathbf{K}_1^{-1}\mathbf{H}_1, \tag{5.11}$$

from which the first two columns of the rotation matrix $\mathbf{R}$ can be factorized using Singular Value Decomposition [144]: Get the SVD decomposition of the first two columns of $\mathbf{M}$, as $\mathbf{M}_{3\times 2} = \mathbf{U}_{3\times 2}\mathbf{\Sigma}_{2\times 2}\mathbf{V}_{2\times 3}^T$. Then the first two columns are $\mathbf{R}_{3\times 2} = \mathbf{U}\mathbf{V}^T$. The third column of $\mathbf{R}$ is the cross product of the first two columns because $\mathbf{R}$ is orthonormal. Finally, the relative position $\mathbf{t}$ will be the third column of $\alpha\mathbf{R}^T\mathbf{M}$, where

$$\alpha = \frac{\mathrm{trace}(\mathbf{R}_{3\times 2}^T\mathbf{M}_{3\times 2})}{\mathrm{trace}(\mathbf{M}_{3\times 2}^T\mathbf{M}_{3\times 2})}, \tag{5.12}$$

and $\mathrm{trace}(\mathbf{A}_{n\times n}) = \sum_{i=1}^{n} a_{ii}$ is the simple matrix trace [144].

5.1.3 Registering the Camera Network with the Extracted Plane

In the previous sections, the relative pose of the *main camera*, as well as the camera network pose, is determined up to a free scale factor independently, hence they are not guaranteed to be in the same scale. The last step of our algorithm is to determine the relative scale of the plane coordinate system with respect to the network coordinate system. Combining the extracted elements will provide a consistent calibration of the camera network in the world coordinate frame, where the origin is attached to a 3D plane.

The task could be formalized as a pattern matching problem, where the aim is to locate a *template* patch in one or more images. As discussed above, the degrees of freedom of such problems and the repetitive nature of TILT features make practically impossible to solve the problem in general [141].

Fortunately, the space of possible transformations can be reduced by making use of the extracted homography of the TILT features. In [141], the pattern matching problem is described as determination of four parameters (two translation and two scaling values). This is achieved by first extracting a set of TILT features in each input image. Then, after merging the coplanar regions, each remaining patch and camera image are rectified using the

extracted homographies. The rectification eliminates the possible rotations, skewing and perspective distortions, thus we only have to deal with translation and scaling. The parameters are obtained by maximizing the normalized cross correlation between the rectified images and the transformed patch [141]. While this approach relies only on the TILT features, the repetitive nature of the features could lead to false matchings. Nevertheless, the rectification of the whole camera images can be impractical on a mobile device.

Further reductions can be achieved if we assume the availability of the camera poses. In such case, the problem reduces only to find a single parameter [Sánta and Kato, 2013b]. Intuitively, this parameter describes the distance between the *main camera* and the rectified plane w.r.t. the coordinate system of the camera network. Let us assume that, we determined a TILT feature $T = (I_1, \mathbf{H}_1, \mathbf{E}_1)$ in the image of $\mathbf{P}_1$, where $\mathbf{H}_1$ is a planar homography between I_1 and I_0. The aim is to find I_k, which will be the projection of I_0 in the image of P_k by estimating the relative scale of $\mathbf{H}_1$ [Sánta and Kato, 2013b].

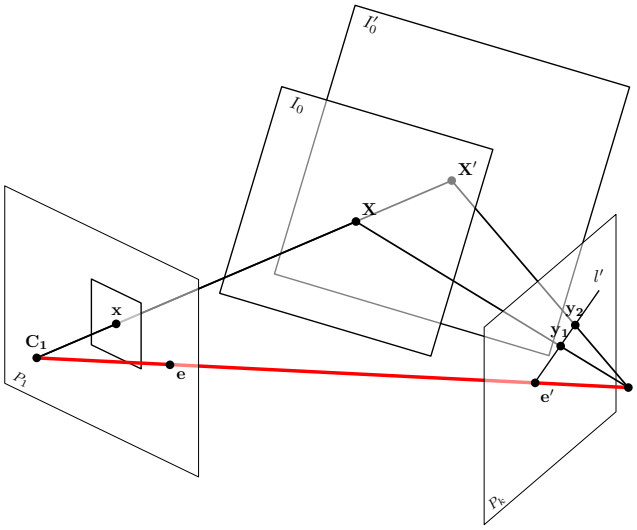


Figure 5.2: Illustration of the scale estimation.

Observe that, due to the homogeneous division for all $\beta \neq 0$ value

$$I_1 = \mathbf{H}_1(I_0) = \beta \mathbf{H}_1(I_0). \tag{5.13}$$

This equation describes the scale ambiguity of the perspective projections, which is caused by the unknown physical size of I_0. However, when two cameras are available the behavior of the β scale factor can be measured in the second camera, as Figure 5.2 shows it. Consider a fixed point $\bar{\mathbf{x}}$ of I_1, *e.g.* the centroid of the low-rank pattern, which is projected back to a 3D point $\mathbf{H}_1^{-1}\bar{\mathbf{x}}$. Intuitively, β will be a scale along this projection ray [Sánta and Kato, 2013b]. In the camera $\mathbf{P}_k$, the image of this ray will be the epipolar line l' corresponding to $\bar{\mathbf{x}}$, therefore, according to the epipolar constraint, the image of the 3D point $\mathbf{H}_1^{-1}\bar{\mathbf{x}}$ in $\mathbf{P}_k$ must be on l'. Changing β will move this image point along l' (see Figure 5.13) [Sánta and Kato, 2013b].

Let us denote the homography acting between I_0 and $\mathbf{P}_k$ by $\mathbf{H}_k$. Since $\mathbf{P}_k$ sees the same I_0 pattern as $\mathbf{P}_1$, there is only one β for which $\bar{\mathbf{y}} = \mathbf{H}_k \beta \mathbf{H}_1^{-1}\mathbf{x}$ also holds. Since $\mathbf{H}_k$ is not available, further investigation is needed to estimate the pursued scale factor. According to

the initial assumption, we have the relative pose between the $\mathbf{P}_1$ and $\mathbf{P}_k$ as

$$\hat{\mathbf{P}}_1 = [\mathbf{I} \mid \mathbf{0}] \qquad \hat{\mathbf{P}}_k = [\mathbf{R}_k \mid \mathbf{t}_k]. \tag{5.14}$$

Moreover, the relative pose of $\mathbf{P}_1$ w.r.t. I_0 is also available using Equation (5.10):

$$\hat{\mathbf{P}}_1 = [\mathbf{R}_1 \mid \mathbf{R}_1 \beta \mathbf{t}_1]. \tag{5.15}$$

Observe that, the β parameter acts only on the translational part. Combining these two equations, the relative pose of $\mathbf{P}_k$ w.r.t. I_0 can be written as

$$\hat{\mathbf{P}}_k^{\beta} = \mathbf{R}_k[\mathbf{R}_1 \mid (\frac{1}{\beta}\mathbf{t}_k + \mathbf{R}_1\mathbf{t}_1)]. \tag{5.16}$$

Using $\mathbf{P}_k^{\beta}$, the planar homography from I_0 to the image plane of the k^{th} camera can be written as

$$\mathbf{H}_k^{\beta} = \mathbf{K}_k[(\mathbf{R}_k\mathbf{R}_1)_{3\times 2} \mid \frac{1}{\beta}\mathbf{R}_k\mathbf{t}_k + \mathbf{R}_k\mathbf{R}_1\mathbf{t}_1]. \tag{5.17}$$

Finally, the planar homography from the first to the k^{th} camera is $\mathbf{H}_{1\to k}^{\beta} = \mathbf{H}_k\mathbf{H}_1^{-1}$.

In order to find the β scale factor, we can simply use a similarity metric based registration method [2], *e.g.* minimization of *mutual information*. We can also estimate the possible bounds of β based on the fact that transformed bounding box of I_1 to I_k has to fall within the image frame of $\mathbf{P}_k$. Making use of this interval, we can easily initialize and speed up registration via a *branch-and-bound* algorithm: initially, the whole interval is a candidate interval. Then divide the candidate interval into smaller sub-intervals and the similarity metric is computed for each sub-interval centroid as a candidate β value. Then the procedure continues iteratively with the sub-interval giving the best alignment [Sánta and Kato, 2013b].

The algorithm runs on the mobile of $\mathbf{P}_k$, hence the extracted low-rank pattern I_1 (in our experiments, it was at most 256×256 pixels) and the necessary matrices $\mathbf{H}_1$ and $\mathbf{P}_1$ have to be sent at the beginning by the *main camera*. After the estimation, the extracted β is sent back by the k^{th} mobile. To achieve greater stability, the scale factor estimation can be solved simultaneously for all cameras seeing the pattern I_0. The median of these estimated scale factors is taken to filter out potential outliers and errors [Sánta and Kato, 2013b].

Algorithm 5.1 Pseudo code of the calibration framework.

Input: N images and internal calibration matrices $\mathbf{K}_i$ from N cameras
Output: The $\mathbf{P}_i$ camera matrices calibrated to the scene

1: Choose a *main camera*, $\mathbf{P}_1$.
2: Calibrate all cameras to $\mathbf{P}_1$by estimating essential matrices and relative scale factors λ_i as described in Section 5.1.1.
3: Determine the planar homography to a low-rank pattern I_0 seen by the *main camera* as described in Section 5.1.2.
4: Synchronize the scales of the homography and the calibrated camera system estimated in the previous two steps using the method of Section 5.1.3.

5.2 Experimental Results

In order to quantitatively evaluate the performance of the proposed method, a synthetic dataset of 1230 test cases has been created. Each case contains five cameras with randomly

Figure 5.3: Sample low-rank patterns from the database.

generated positions and randomly created calibration matrices (of course with overlapping views), and the cameras took a virtual picture of a 3D plane containing a low-rank pattern chosen randomly from a database of 31 patterns (examples can be found in Figure 5.3). A randomly generated point set has also been coded into these images to simulate point correspondences. Camera parameters were randomly chosen according to the following rules: for intrinsics, the focal length is chosen from $[1000; 1300]$ with zero skew and a principal point set to the image center; for extrinsics, rotation angles were between $[-\pi/4; \pi/4]$, and the translation was between $[-10; 10]$ meters (corresponding to typical imaging conditions in urban areas). Some test cases can be found in Figure 5.4.

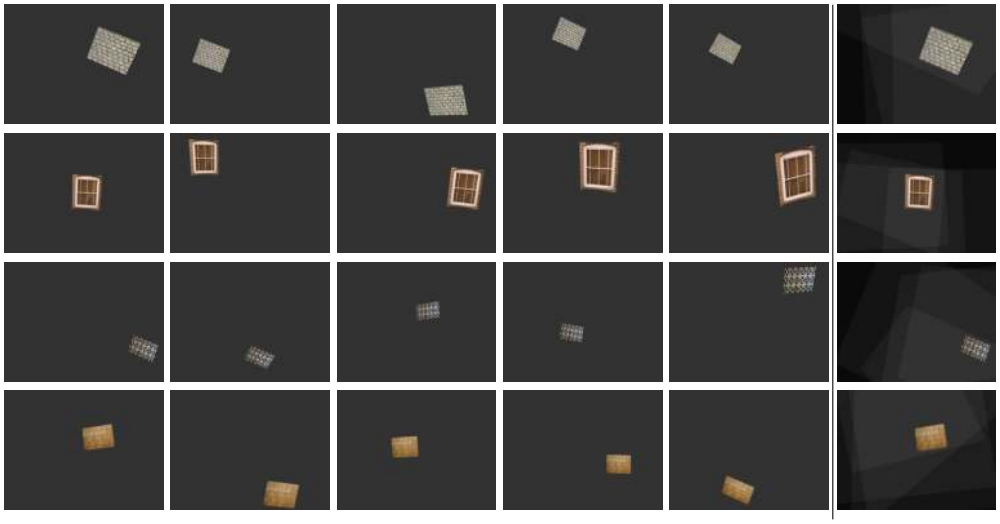


Figure 5.4: Images from the synthetic tests. For each row, the first five pictures show the images of the low-rank pattern as seen by the random cameras, while the last one contains the overlayed reprojected frames using the estimated camera matrices.

The algorithms were implemented in MATLAB. The TILT problem solver for the *low-rank* pattern alignment was obtained from `http://perception.csl.illinois.edu/matrix-rank/tilt.html`. Using this library, an affine alignment has been estimated, then used as an initialization to the homography solver. For the *Normalized 8-point algorithm*, the *RANSAC* [35] version of the algorithm implementation of [145] has been used. For bundle adjustment, we used a modified version of *Generic Sparse Bundle Adjustment* from Lourakis

et al. [87], written in C++. Finally, for the homography scale estimation in Section 5.1.3, we used a simple *mutual information* based registration algorithm using the internal solvers of MATLAB [2].

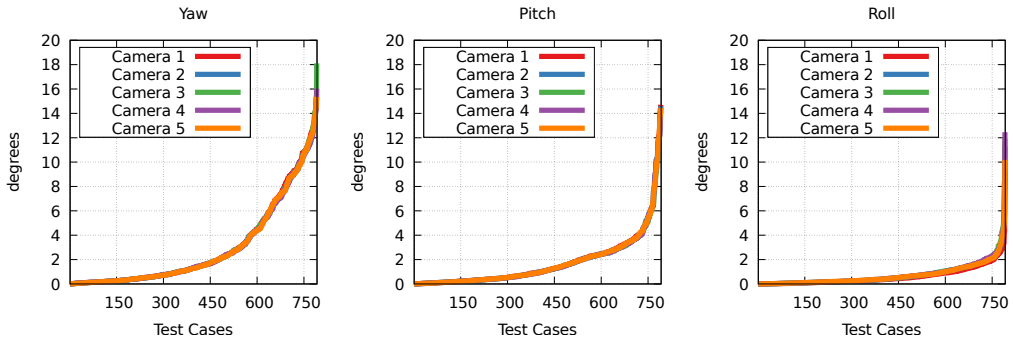


Figure 5.5: Quantitative results of rotation angle estimation.

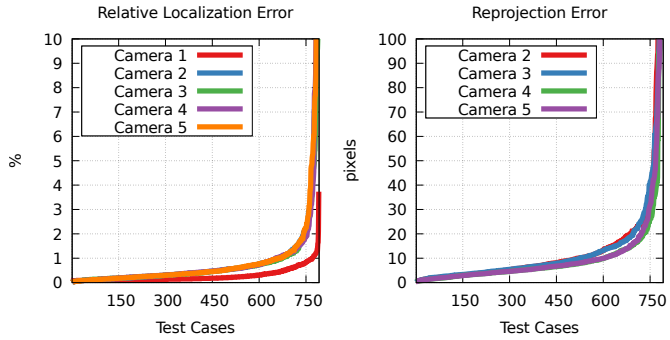


Figure 5.6: The camera position error and reprojection error of the centroid of the *low-rank* pattern.

	Runtime (sec)			
	TILT Solver	8-point algorithm	Bundle adjustment	Homography registration
m	2.83	0.093	0.40	4.17
μ	3.14	0.10	0.41	7.75
σ	1.31	0.03	0.12	68.14

Table 5.1: Runtime statistics on the synthetic images (m – median, μ – mean and σ – standard deviation).

Quantitative results on the synthetic dataset can be found in Figure 5.5 and Figure 5.6. The first diagram of Figure 5.5 contains the error of the *yaw* angle (rotation around the z-axis), the second contains the error of the *pitch* angle (rotation around the y-axis) and the third diagram shows the error of the *roll* angle (rotation around the x-axis). For the majority of the test cases, the difference is below 2 degrees. Figure 5.6 shows the camera position error and the reprojection error of the centroid of the *low-rank* pattern. The position error is computed as the Euclidean distance of the estimated and the original camera positions, and the error in the plot is given in percentages with respect to the original camera distance. Considering that translation is of the order of several meters, the actual error for almost all of the test cases was a few centimeters only. Since all of the frames contained the same *low-rank* plane in the synthetic tests, we could compute the reprojection error for each camera using the $\mathbf{H}_{1\to i}$ $(i = 2, \ldots, N)$ homographies. The reprojection error is defined as the distance

between the true and the projected centroids of the planes. Note that, the reprojection error is always 0 for the first camera. The running time of our algorithm can be found in Table 5.1.

As we mentioned in Section 5.1.3, the homography scale factor β can be determined simultaneously from all cameras seeing the low-rank pattern. As expected, scale estimation became more stable (deviation and average of the errors are considerably lower) when the median of these estimated scales was taken. We show the average gain of using this variant of the algorithm in Table 5.2, where the values were calculated as the average differences for each camera. However, the main advantage of this approach is its robustness against occlusions, which is quite common in real-life situations (see *e.g.* the third image set in Figure 5.8).

	Reprojection error (pixels)	Relative distance (%)
m	1.03	0.12%
μ	27.85	2.47%
σ	54.17	4.46%

Table 5.2: The average gain of solving the homography scale estimation for all cameras (m – median, μ – mean and σ – standard deviation).

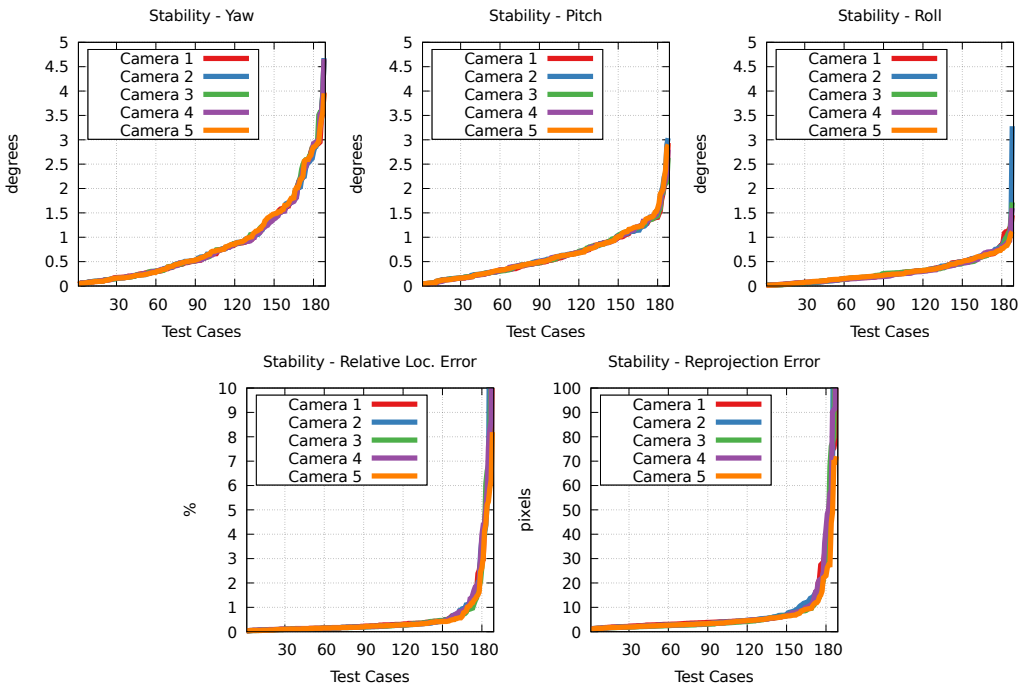


Figure 5.7: Stability test with varying *main cameras*. Each plot shows the average of absolute distances from the medians of each configurations.

Another interesting topic is the robustness of the algorithm when the *main camera* is changed. Since our approach is distributed, the choice of the *main camera* should be arbitrary, and the calibration results must not be influenced by this choice. Figure 5.7 shows how the algorithm behaves within a particular network of five cameras when the *main camera* is changed. The plots show the average of absolute distances from the medians for each *main camera* configurations. The charts for each case are running together and errors are low, showing that calibration results are not affected seriously by the choice of the *main camera*. Of course, the middle camera is usually the best choice, as it minimizes the distortion with the other cameras.

Figure 5.8: Results on real images. For every test case, the last image shows the overlayed back projected camera images using the estimated camera matrices. The low-rank patterns used for calibration are marked on the main camera image as well as on the result.

5.2.1 Results on Real Data

The proposed method has been tested on real images as well. The photos have been taken in urban environment by five different smartphones. The point correspondences were extracted using ASIFT [27] features and the bounding boxes of low-rank pattern were picked by hand. The main camera was always the central camera for these test cases. Our results on this dataset can be found in Figure 5.8. The first three sets show good calibrations, while the last row presents a less precise estimate, which is mainly due to the errors in establishing point-wise correspondences.

Usually, a real scene contains low-rank patterns repetitively (see Figure 5.8 for examples), and as we described above, the matching of these patterns with conventional methods is quite difficult. The homography scale estimator proposed in Section 5.1.3 proved to be quite robust, which is a major advantage of our solution.

5.3 Conclusion and Future Work

A camera network pose estimation algorithm is proposed, which was inspired by the recent works on low-rank rectification. First, the relative pose of the camera network is determined, then the relative pose to a real 3D plane with a low-rank pattern is estimated. The calibration algorithm is designed in a distributed way: each mobile is processing its own image and sends only the minimum amount of data towards other mobiles. Although, for many steps of the algorithm it is necessarily to pick a *main camera*, the choice of this particular camera is arbitrary and has no major influence on the quality of the final estimates. Robustness and stability of the algorithm have been tested on a large synthetic dataset as well as on real images taken in urban environments. While the presented results are encouraging, we see several points where the algorithm could be further improved, *e.g.* investigate the possibilities of automatic detection of the low-rank patterns, or increase the accuracy of the homography estimator.

Chapter 6

Conclusions

This thesis work summarized the author's research on various topics of computer vision. These are all related to *image registration*, which is a particularly interesting problem in itself. Unfortunately, many of the *gold standard* techniques rely on heuristic assumptions or manual interactions. Automatized approaches, however, are able to reduce the overall time consumption and increase the accuracy. The presented research intended to increase automation in the observed areas by developing new ideas and algorithms.

The first topic addressed a novel registration framework for aligning 3D objects without established point-wise correspondences. The basic idea is to set up a system of non-linear equations which solution directly provides the parameters of the aligning transformation modeled by a parametric transformation model. The framework has successfully utilized the advantages of various 3D data representations, such as volumetric images and triangular surface meshes. The efficiency and the robustness of the proposed approach have been demonstrated on large synthetic and real datasets. Our method compares favorably to two recent 3D matching algorithms [5, 6].

In the next topic, we have addressed two challenges of registering images using the 2D variant of the algebraic framework from the previous chapter. First, we have proposed a novel algorithm in order to deal with higher occlusions for affine deformations. The basic idea is to represent the shapes as polygons, then iteratively estimate the occluded regions using simple boolean operators and solve a system of non-linear equations in the least-squares sense. The robustness against higher segmentation error has been validated on a large synthetic dataset, and the method outperformed two recent registration algorithms [6, 44] in terms of alignment accuracy.

Thereafter, we have proposed an algorithm for deformable registration of image patches using the same framework. The main aim is to handle covariant functions, *e.g.* intensity information, to reduce the ambiguity of binary registration. While any parametric model could be used within the framework, we focused on affine and TPS models in this work. Experimental results show the capabilities of the proposed approach both on synthetic and real images. The presented polynomial equation based approach compares favorably to the linear one published in [130]. For elastic deformations, the proposal has surpassed two state of the art methods published in [11] and [12].

Finally, a camera pose estimation algorithm is proposed which was inspired by the recent works on low-rank texture rectification. First, the relative pose of the camera network is determined, then the relative pose to a real 3D plane with a low-rank pattern is estimated. The calibration algorithm is designed in a distributed way: each mobile is processing its

own image and sends only the minimum amount of data towards other mobiles. Although, for many steps of the algorithm it is necessary to pick a *main camera*, the choice of this particular camera is arbitrary and has no major influence on the quality of final estimates. The robustness and stability of the algorithm have been tested on a large synthetic dataset as well as on real images taken in urban environments.

Appendix A

Summary in English

Computer vision is a field of analyzing and interpreting visual objects by making use of various computational tools. The fundamental aim is to create *automatized* systems to efficiently handle multiple tasks. In almost every computer vision process an important preparatory step is *image registration*, which is a particularly interesting problem in itself. The main goal of this task is to estimate mappings between different observations of the same scene. Unfortunately, many of the *gold standard* techniques rely on heuristic assumptions or manual interactions. On the contrary, automatized approaches are able to reduce the overall time consumption and increase the accuracy. This work presents my research on developing solutions for various problems of image registration.

A.1 Key Points of the Thesis

In the following, I summarized my results into two main thesis groups. In the first one, I present my findings on registering 3D objects, while in the second one my results on 2D shape registration are shown. In Table A.1, the connections between the thesis points and the corresponding publications are displayed.

I.) **Registration of 3D Objects**
Inspired by the already published 2D registration framework [7], the method can be extended for registering 3D objects. The basic idea is to set up a system of non-linear equations whose solution directly provides the parameters of the aligning transformation modeled by a parametric transformation model.

(a) When considering general 3D surfaces as input objects, the basic integrals of the equations will be surface integrals. I derived two recursive numerical schemes (an exact and an approximate) to efficiently estimate surface integrals over triangular surface meshes. I tested the proposed framework on a large synthetic dataset using thin plate spline (TPS) transformation model. I empirically verified the robustness of the method against segmentation errors and compared the results to two recent registration framework [5, 6]. Finally, I showed practical application of the method for aligning 3D facial scans.

(b) Registering 3D volumetric objects can be realized by making use of volumetric integrals as well. For this purpose, I investigated two object representations: a

voxel based and a closed triangular surface based approach. For voxel representation, following the theoretical result from [7], I derived an efficient numerical scheme for estimating the integrals in the case of polynomial transformations. For triangular surface mesh representation, following the general surface approach, I gave an efficient numerical scheme based on tetrahedrons in the case of TPS transformations. I demonstrated the efficiency of the methods on a large synthetic dataset. I made experiments to verify the robustness of the methods against segmentation errors and model overfitting. I showed practical applications of the method for registering lung CT scans and brain surfaces.

II.) **Registration of 2D Shapes**

This thesis group summarize my results on registering binary and grayscale images. In the first topic, I dealt with the alignment of occluded binary shapes, which is a common problem for registering images taken in less controlled environments. In the second topic, I dealt with the ambiguity of the inner parts of shapes, when they are registered using non-rigid transformation models with higher degrees of freedom. Finally, I developed a method for calibrating ad-hoc camera networks.

(a) The affine registration methods for binary shapes published in [122, 127] can be adapted to handle occluded shapes by appropriately choosing the integration domains. In the proposed approach, I represented the shapes as polygons and determined iteratively the occluded areas. Then, using these areas as integration domains, I estimated the best affine transformation between the shapes. I demonstrated the efficiency of the method on a large synthetic dataset. I compared the results to two recent registration methods [6, 44]. I showed practical applicability of the method on images taken in urban environments containing static and dynamic occlusions.

(b) The general framework from [7] can be further regularized by making use of grayscale images. For this purpose, I developed a formalism, where the geometric and intensity information are used in a coupled system of equations. I verified empirically the efficiency of the method on a large synthetic dataset. I showed experimentally the robustness of the method against additive zero-mean Gaussian noise. I compared the results to two recent registration method [11, 12]. Finally, I showed practical applicability of the framework on real images and on a public dataset.

(c) Recent results on Transform Invariant Low-rank Textures (TILT) [142] can be used for calibrating camera networks. I developed an algorithm which is able to determine the absolute pose of a camera network by making use of planar homographies extracted from TILT features. I verified the efficiency and stability of the proposed method on a large synthetic dataset and real images taken with mobile cameras.

	I		II		
	a	b	a	b	c
[Sánta and Kato, 2012a]		•			
[Sánta and Kato, 2012b]		•			
[Sánta and Kato, 2013a]		•			
[Sánta and Kato, 2016a]	•				
[Sánta and Kato, 2018]	•	•			
[Sánta and Kato, 2014]			•		
[Sánta and Kato, 2016b]				•	
[Sánta and Kato, 2013b]					•

Table A.1: The connection between the thesis points and publications.

Appendix B

Summary in Hungarian

A *számítógépes látás* témakörében vizuális objektumok számítógépes értelmezésével és vizsgálatával foglalkozunk. A cél olyan automatizált rendszerek kidolgozása amelyek képesek hatékonyan megvalósítani az élőlények vizuális érzékelését. A legtöbb számítógépes látásbeli problémában egy fontos előkészítő lépés a *képregisztráció*, amely önmagában is érdekes terület. Itt a fő célunk, hogy leképzéseket határozzunk meg az aktuálisan vizsgált színtér különböző megfigyelései között. Sajnos a jelenleg alkalmazott eljárások nagyban támaszkodnak különböző heurisztikákra és emberi interakcióra. Ezzel szemben az automatizált módszert lecsökkenthetjük a megoldáshoz szükséges időt és növelhetjük az eredmények pontosságát. A dolgozatban összefoglaltam a kutatási eredményeim a képregisztráció területéről.

B.1. Az eredmények tézisszerű összefoglalása

A dolgozat eredményeit két fő téziscsoportban foglaltam össze, ahol az elsőben 3D objektumok, míg a másodikban 2D alakzatok regisztrációjával foglalkozom. A téziscsoportok és az elfogadott publikációim közötti kapcsolatot a B.1 táblázatban prezentálom.

I.) **Háromdimenziós objektumok regisztrációja**
A korábban publikált 2D keretrendszer által inspirálva [7], a módszer kiterjeszthető 3D objektumok regisztrációjára. A javasolt módszerben a regisztrációs problémát egy megfelelően konstruált nem-lineáris egyenletrendszer megoldására vezetjük vissza, ahol az ismeretlenek megadják a keresett transzformációs modell paramétereit.

   (a) Az egyenletekben szereplő integrálok általános 3D felszínek esetén felszíni integrálok lesznek. Ezek hatékony kiszámítására levezettem két rekurzív numerikus formulát (egy egzakt és egy approximációs) háromszögfelszíni hálós reprezentáció esetén. A javasolt keretrendszert nagy méretű szintetikus adathalmazon teszteltem, vékony fémlemez spline transzformációt (TPS) használva. Empirikusan igazoltam a módszer robusztusságát szegmentálási hibákkal szemben, és az eredményeket összehasonlítottam két korábban publikált módszer eredményeivel [5, 6]. Végül a keretrendszer valós alkalmazhatóságát különböző emberekről készített 3D arcfelvételek illesztésével mutattam be.

   (b) Térfogati objektumok illesztése esetén az integrálok felírhatók volumetrikus integrálokként is. Ehhez kétféle reprezentációt vizsgáltam: a voxel alapú és a zárt háromszögfelszíni hálókkal megadott objektumokat. A voxel alapú megoldásnál

a [7] cikkben publikált elméleti eredmény felhasználásával levezettem egy hatékony numerikus számítási eljárást polinomiális transzformációk alkalmazásakor. A zárt felszíni hálós megoldásban az általános felszínekre felírt egzakt számítási formulát követve egy rekurzív, tetraéder alapú megoldást vezettem le TPS transzformációkra. A módszerek hatékonyságát nagy méretű szintetikus teszthalmazon igazoltam. A módszer robusztusságát szegmentálási és modell túlillesztési hibákkal szemben kísérleti úton igazoltam. A kapott eredményeket összehasonlítottam két korábban publikált módszer eredményeivel [5, 6]. A módszer valós alkalmazhatóságának igazolására tüdő CT felvételek és agyi felszínek regisztrációja került bemutatásra.

II.) **Kétdimenziós alakzatok regisztrációja**
A téziscsoport összefoglalja az eredményeim bináris és többszintű képek regisztrációjában. Első témámban kitakart és hiányos bináris alakzatok regisztrációjával foglalkoztam, ami egy gyakori probléma valós környezetben készített képek illesztésénél. Ezt követően a bináris alakzatok nem-lineáris regisztrációja során tapasztalható alulhatározottság problémájára javasoltam egy megoldást. Végül kidolgoztam egy ad-hoc kamerahálózatok kalibrációjának meghatározására alkalmas algoritmust.

(a) A [122, 127] cikkekben bemutatott, bináris alakzatok affin illesztésére alkalmas keretrendszer adaptálható kitakart alakzatok regisztrációjára, az integrálási tartomány alkalmas megválasztásával. A javasolt algoritmusban az alakzatokat poligonokként reprezentálva, iterációnként meghatároztam a kitakart területeket. Majd ezen területeket integrálási tartományként használva meghatároztam a legjobb affin transzformációt az alakzatok között. A módszer hatékonyságát nagy méretű szintetikus adathalmazon igazoltam. Az eredményeket összehasonlítottam két korábban publikált módszer eredményeivel [6, 44]. A javasolt módszer valós alkalmazhatóságát városi körülmények között készített statikus és dinamikus takarásokat tartalmazó képeken mutattam be.

(b) A [7] cikkben bemutatott módszer tovább regularizálható szürkeárnyalatos képek alkalmazásával. Ehhez kidolgoztam egy olyan formalizmust, amelyben a geometriai és intenzitás információk egy közös egyenletrendszerben jelennek meg. A módszer hatékonyságát empirikus úton igazoltam nagy elemszámú szintetikusan generált halmazon. Kísérleti úton megmutattam az eljárás robusztusságát különböző szórással generált normális eloszlású intenzitás zaj mellett. Összehasonlítottam a kapott eredményeket két korábban publikált módszer eredményeivel [11, 12]. Végül valós körülmények között készített képekkel és egy publikusan elérhető tesztadatbázison igazoltam a módszer alkalmazhatóságát.

(c) Az alacsony rangú textúrákkal (TILT) [142] kapcsolatos eredmények felhasználhatók kamerahálózatok kalibrációjának meghatározására. Kidolgoztam egy algoritmust, amely TILT jellemzőkből kinyert 2D homográfiák segítségével meghatározza egy kamerahálózat és az egyes kamerák abszolút helyzetét a világ koordinátarendszerben. A javasolt algoritmus hatékonyságát és stabilitását nagy elemszámú szintetikus halmazon és mobil kamerákkal készített valós képekkel is igazoltam.

	I		II		
	a	b	a	b	c
[Sánta and Kato, 2012a]		•			
[Sánta and Kato, 2012b]		•			
[Sánta and Kato, 2013a]		•			
[Sánta and Kato, 2016a]	•				
[Sánta and Kato, 2018]	•	•			
[Sánta and Kato, 2014]			•		
[Sánta and Kato, 2016b]				•	
[Sánta and Kato, 2013b]					•

B.1. táblázat. A tézispontokhoz kapcsolódó publikációk.

Publications

Refereed Articles

[Sánta and Kato, 2018] Z. Sánta and Z. Kato, "Elastic Alignment of Triangular Surface Meshes", *International Journal on Computer Vision*, 2018, Accepted.

Refereed Conference Papers

[Sánta and Kato, 2016a] Z. Sánta and Z. Kato, "3D Face Alignment without Correspondences", in *Proceedings of the ECCV Workshop on 3D Face Alignment in the Wild*, Amsterdam, Netherlands, Oct. 2016, pp. 521–535. DOI: 10.1007/978-3-319-48881-3_36.

[Sánta and Kato, 2016b] Z. Sánta and Z. Kato, "An Algebraic Framework for Deformable Image Registration", in *Proceedings of International Conference on Pattern Recognition*, Cancun, Mexico, Dec. 2016, pp. 3792–3797. DOI: 10.1109/ICPR.2016.7900225.

[Tanács *et al.*, 2015] A. Tanács, A. Majdik, L. Hajder, J. Molnár, Z. Sánta, and Z. Kato, "Collaborative Mobile 3D Reconstruction of Urban Scenes", in *Proceedings of the ACCV Workshop on Intelligent Mobile and Egocentric Vision*, C. V. Jawahar and S. Shan, Eds., ser. Lecture Notes in Computer Science, vol. 9010, Singapore: Springer International Publishing, 2015, pp. 486–501, ISBN: 978-3-319-16633-9. DOI: 10.1007/978-3-319-16634-6_36.

[Sánta and Kato, 2014] Z. Sánta and Z. Kato, "Affine Alignment of Occluded Shapes", in *Proceedings of International Conference on Pattern Recognition*, IAPR, Stockholm, Sweden: IEEE, Aug. 2014, pp. 2155–2160. DOI: 10.1109/ICPR.2014.375.

[Sánta and Kato, 2013a] Z. Sánta and Z. Kato, "Correspondence-Less Non-Rigid Registration of Triangular Surface Meshes", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Portland, Oregon: IEEE, Jun. 2013, pp. 2275–2282. DOI: 10.1109/CVPR.2013.295.

[Sánta and Kato, 2013b] Z. Sánta and Z. Kato, "Pose Estimation of Ad-hoc Mobile Camera Networks", in *Proceedings of International Conference on Digital Image Computing: Techniques and Applications*, Hobart, Tasmania, Australia: IEEE, Nov. 2013, pp. 88–95. DOI: 10.1109/DICTA.2013.6691514.

[Sánta and Kato, 2012a] Z. Sánta and Z. Kato, "A Unifying Framework for Non-linear Registration of 3D Objects", in *Proceedings of IEEE International Conference on Cognitive Infocommunications*, Kosice, Slovakia: IEEE, Dec. 2012, pp. 547–552. DOI: 10.1109/CogInfoCom.2012.6422041.

[Sánta and Kato, 2012b] Z. Sánta and Z. Kato, "Elastic Registration of 3D Deformable Objects", in *Proceedings of International Conference on Digital Image Computing: Techniques and Applications*, Fremantle, Western Australia: IEEE, Dec. 2012. DOI: 10.1109/DICTA.2012.6411674.

Bibliography

[1] L. G. Brown, "A survey of image registration techniques", *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992. DOI: 10.1145/146370.146374.

[2] B. Zitová and J. Flusser, "Image registration methods: A survey", *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003, ISSN: 0262-8856. DOI: 10.1016/S0262-8856(03)00137-9.

[3] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration", *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998. DOI: 10.1016/s1361-8415(01)80026-8.

[4] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable Medical Image Registration: A Survey", *IEEE Transactions on Medical Imaging*, vol. 32, no. 7, pp. 1153–1190, Jul. 2013, ISSN: 0278-0062. DOI: 10.1109/TMI.2013.2265603.

[5] B. Jian and B. Vemuri, "Robust Point Set Registration Using Gaussian Mixture Models", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 33, no. 8, pp. 1633–1645, Aug. 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.223.

[6] A. Myronenko and X. Song, "Point Set Registration: Coherent Point Drift", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.46.

[7] C. Domokos, J. Nemeth, and Z. Kato, "Nonlinear Shape Registration without Correspondences", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 34, no. 5, pp. 943–958, 2012. DOI: 10.1109/TPAMI.2011.200.

[8] Y. Tian and S. G. Narasimhan, "Globally Optimal Estimation of Nonrigid Image Distortion", *International Journal on Computer Vision*, vol. 98, no. 3, pp. 279–302, 2012, ISSN: 0920-5691. DOI: 10.1007/s11263-011-0509-0.

[9] M. Salzmann, R. Hartley, and P. Fua, "Convex Optimization for Deformable Surface 3-D Tracking", in *Proceedings of International Conference on Computer Vision*, Oct. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409031.

[10] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework", *International Journal on Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000011205.11775.fd.

[11] B. Glocker, A. Sotiras, N. Komodakis, and N. Paragios, "Deformable Medical Image Registration: Setting the State of the Art with Discrete Methods", *Annual Review of Biomedical Engineering*, vol. 13, no. 1, pp. 219–244, 2011. DOI: 10.1146/annurev-bioeng-071910-124649.

[12] H. Lombaert *et al.*, "Spectral Log-Demons: Diffeomorphic Image Registration with Very Large Deformations", *International Journal on Computer Vision*, vol. 107, no. 3, pp. 254–271, 2014. DOI: 10.1007/s11263-013-0681-5.

[13] F. Michel, M. Bronstein, A. Bronstein, and N. Paragios, "Boosted metric learning for 3D multi-modal deformable registration", in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, Mar. 2011, pp. 1209–1214. DOI: 10.1109/ISBI.2011.5872619.

[14] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 3594–3601. DOI: 10.1109/CVPR.2010.5539928.

[15] Q. Xie *et al.*, "Metric-Based Pairwise and Multiple Image Registration", in *Proceedings of European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8690, Springer International Publishing, 2014, pp. 236–250, ISBN: 978-3-319-10604-5. DOI: 10.1007/978-3-319-10605-2_16.

[16] M. Holden, "A Review of Geometric Transformations for Nonrigid Body Registration", *IEEE Transactions on Medical Imaging*, vol. 27, no. 1, pp. 111–128, 2008. DOI: 10.1109/TMI.2007.904691.

[17] X. Yang, Z. Xue, X. Liu, and D. Xiong, "Topology preservation evaluation of compact-support radial basis functions for image registration", *Pattern Recognition*, vol. 32, no. 8, pp. 1162–1177, 2011, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2011.03.004.

[18] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, 2nd. Springer Publishing Company, Incorporated, 2009, ISBN: 978-1-85233-617-2.

[19] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer London, 2011. DOI: 10.1007/978-1-84882-935-0.

[20] A. A. Goshtasby, *Image Registration: Principles, Tools and Methods*, 1st ed., ser. Advances in Computer Vision and Pattern Recognition. Springer-Verlag London, 2012, ISBN: 978-1-4471-2458-0. DOI: 10.1007/978-1-4471-2458-0.

[21] G. K. L. Tam *et al.*, "Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid", *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1199–1217, 2013, ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.310.

[22] Y. Lipman and I. Daubechies, "Conformal Wasserstein Distances: Comparing Surfaces in Polynomial Time", *Advances in Mathematics*, vol. 227, no. 3, pp. 1047–1077, 2011, ISSN: 0001-8708. DOI: 10.1016/j.aim.2011.01.020.

[23] M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 1704–1711. DOI: 10.1109/CVPR.2010.5539838.

[24] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended Intrinsic Maps", *ACM Transactions on Graphics*, vol. 30, no. 4, 79:1–79:12, Jul. 2011, ISSN: 0730-0301. DOI: 10.1145/2010324.1964974.

[25] I. H. Jermyn, S. Kurtek, E. Klassen, and A. Srivastava, "Elastic Shape Matching of Parameterized Surfaces Using Square Root Normal Fields", in *Proceedings of European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, A. Fitzgibbon *et al.*, Eds., vol. 7576, Springer Berlin Heidelberg, 2012, pp. 804–817, ISBN: 978-3-642-33714-7. DOI: 10.1007/978-3-642-33715-4_58.

[26] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: A survey", *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2015.02.008.

[27] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, ISSN: 1077-3142. DOI: 10.1016/j.cviu.2007.09.014.

[29] J. Morel and G. Yu, "ASIFT: A New Framework for Fully Affine Invariant Image Comparison", *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 438–469, 2009. DOI: 10.1137/080732730.

[30] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features", in *Proceedings of European Conference on Computer Vision*, Fiorenze, Italy, 2012. DOI: 10.1007/978-3-642-33783-3_16.

[31] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces", in *Proceedings of British Machine Vision Conference*, Bristol, UK, 2013. DOI: 10.5244/c.27.13.

[32] J. Matas, O. Chum, U. Martin, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions", in *Proceedings of British Machine Vision Conference*, 2002, pp. 384–393. DOI: 10.5244/C.16.36.

[33] F. L. Bookstein, "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 11, no. 6, pp. 567–585, 1989. DOI: 10.1109/34.24792.

[34] L. Zagorchev and A. Goshtasby, "A Comparative Study of Transformation Functions for Nonrigid Image Registration", *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 529–538, Mar. 2006, ISSN: 1057-7149. DOI: 10.1109/TIP.2005.863114.

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Magazine Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[36] X. Yang and K.-T. Cheng, "Local Difference Binary for Ultrafast and Distinctive Feature Description", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 36, no. 1, pp. 188–194, Jan. 2014, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.150.

[37] J. Ma *et al.*, "Robust L_2E Estimation of Transformation for Non-Rigid Registration", *IEEE Transactions on Signal Processing*, vol. 63, no. 5, pp. 1115–1129, Mar. 2015, ISSN: 1053-587X. DOI: 10.1109/TSP.2014.2388434.

[38] D. W. Scott, "The L2E method", *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 45–51, 2009, ISSN: 1939-0068. DOI: 10.1002/wics.4.

[39] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 373–380. DOI: 10.1109/CVPR.2009.5206748.

[40] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992. DOI: 10.1109/34.121791.

[41] H. Chui and A. Rangarajan, "A New Point Matching Algorithm for Non-rigid Registration", *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, Feb. 2003, ISSN: 1077-3142. DOI: 10.1016/S1077-3142(03)00009-2.

[42] A. Myronenko, X. B. Song, and M. Á. Carreira-Perpiñán, "Non-rigid point set registration: Coherent Point Drift", in *Proceedings of the Advances in Neural Information Processing Systems*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds., Vancouver, British Columbia, Canada: MIT Press, Dec. 2006, pp. 1009–1016.

[43] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces", *International Journal on Computer Vision*, vol. 13, no. 2, pp. 119–152, Oct. 1994. DOI: 10.1007/BF01427149.

[44] J. Zhu *et al.*, "Robust affine iterative closest point algorithm with bidirectional distance", *Computer Vision, IET*, vol. 6, no. 3, pp. 252–261, 2012, ISSN: 1751-9632. DOI: 10.1049/iet-cvi.2011.0178.

[45] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The Trimmed Iterative Closest Point algorithm", in *Proceedings of International Conference on Pattern Recognition*, vol. 3, 2002, pp. 545–548. DOI: 10.1109/ICPR.2002.1047997.

[46] J. Yang, H. Li, and Y. Jia, "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally", in *Proceedings of International Conference on Computer Vision*, Dec. 2013, pp. 1457–1464. DOI: 10.1109/ICCV.2013.184.

[47] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[48] J.-P. Thirion, "Image matching as a diffusion process: An analogy with Maxwell's demons", *Medical Image Analysis*, vol. 2, no. 3, pp. 243–260, 1998. DOI: 10.1016/S1361-8415(98)80022-4.

[49] X. Pennec, P. Cachier, and N. Ayache, "Understanding the "Demon's Algorithm": 3D Non-rigid Registration by Gradient Descent", in *Medical Image Computing and Computer-Assisted Intervention*, ser. Lecture Notes in Computer Science, C. Taylor and A. Colchester, Eds., vol. 1679, Springer Berlin Heidelberg, 1999, pp. 597–605, ISBN: 978-3-540-66503-8. DOI: 10.1007/10704282_64.

[50] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, "Non-parametric Diffeomorphic Image Registration with the Demons Algorithm", in *Medical Image Computing and Computer-Assisted Intervention*, ser. Lecture Notes in Computer Science, N. Ayache, S. Ourselin, and A. Maeder, Eds., vol. 4792, Springer Berlin Heidelberg, 2007, pp. 319–326, ISBN: 978-3-540-75758-0. DOI: 10.1007/978-3-540-75759-7_39.

[51] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 6, no. 6, pp. 721–741, Nov. 1984, ISSN: 0162-8828. DOI: 10.1109/TPAMI.1984.4767596.

[52] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer, 2009, vol. 26, ISBN: 978-1-84800-279-1. DOI: 10.1007/978-1-84800-279-1.

[53] B. Glocker *et al.*, "Dense Image Registration through MRFs and Efficient Linear Programming", *Medical Image Analysis*, vol. 12, no. 6, pp. 731–741, 2008, Special issue on information processing in medical imaging 2007, ISSN: 1361-8415. DOI: 10.1016/j.media.2008.03.006.

[54] A. Yezzi, L. Zöllei, and T. Kapur, "A variational framework for integrating segmentation and registration through active contours", *Medical Image Analysis*, vol. 7, no. 2, pp. 171–185, 2003. DOI: 10.1016/s1361-8415(03)00004-5.

[55] J. Kybic and J. Borovec, "Automatic simultaneous segmentation and fast registration of histological images", in *Proceedings of International Symposium on Biomedical Imaging*, Apr. 2014, pp. 774–777. DOI: 10.1109/ISBI.2014.6867985.

[56] T. Gass, G. Székely, and O. Goksel, "Simultaneous Segmentation and Multiresolution Nonrigid Atlas Registration", *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2931–2943, Jul. 2014, ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2322447.

[57] J. Pokrass, A. M. Bronstein, and M. M. Bronstein, "Partial Shape Matching without Point-Wise Correspondence", *Numerical Mathematics: Theory, Methods and Applications*, vol. 6, pp. 223–245, 2012. DOI: 10.1017/S1004897900001203.

[58] S. Korman, D. Reichman, G. Tsur, and S. Avidan, "FasT-Match: Fast Affine Template Matching", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 2331–2338. DOI: 10.1109/CVPR.2013.302.

[59] S. M. Seitz *et al.*, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, New York, NY: IEEE Computer Society, 2006, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

[60] G. Jin *et al.*, "3D Surface Reconstruction and Registration for Image Guided Medialization Laryngoplasty", in *Proceedings of the International Conference on Advances in Visual Computing*, Lake Tahoe, NV: Springer-Verlag, 2006, pp. 761–770, ISBN: 978-3-540-48628-2. DOI: 10.1007/11919476_76.

[61] T. Weise, B. Leibe, and L. Van Gool, "Fast 3D Scanning with Automatic Motion Compensation", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383291.

[62] F. Blais, M. Picard, and G. Godin, "Accurate 3D acquisition of freely moving objects", in *Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, Sep. 2004, pp. 422–429. DOI: 10.1109/TDPVT.2004.1335269.

[63] L. Rineau and M. Yvinec, "3D Surface Mesh Generation", in *CGAL User and Reference Manual*, 4.0, CGAL Editorial Board, 2012.

[64] Y. Zeng *et al.*, "Dense non-rigid surface registration using high-order graph matching", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 382–389. DOI: 10.1109/CVPR.2010.5540189.

[65] Y. Zeng *et al.*, "A Generic Deformation Model for Dense Non-rigid Surface Registration: A Higher-Order MRF-Based Approach", in *Proceedings of International Conference on Computer Vision*, Dec. 2013, pp. 3360–3367. DOI: 10.1109/ICCV.2013.417.

[66] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A Tensor-Based Algorithm for High-Order Graph Matching", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 33, no. 12, pp. 2383–2395, Dec. 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.110.

[67] L. Torresani, V. Kolmogorov, and C. Rother, "Feature Correspondence Via Graph Matching: Models and Global Optimization", in *Proceedings of European Conference on Computer Vision*, D. Forsyth, P. Torr, and A. Zisserman, Eds., ser. Lecture Notes in Computer Science, vol. 5303, Springer Berlin Heidelberg, 2008, pp. 596–609, ISBN: 978-3-540-88685-3. DOI: 10.1007/978-3-540-88688-4_44.

[68] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 5, pp. 1168–1172, Jan. 2006, ISSN: 1091-6490. DOI: 10.1073/pnas.0508601103.

[69] J. Sun, M. Ovsjanikov, and L. Guibas, "A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion", *Computer Graphics Forum*, vol. 28, no. 5, pp. 1383–1392, 2009, ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01515.x.

[70] M. Aubry, U. Schlickewei, and D. Cremers, "The Wave Kernel Signature: A Quantum Mechanical Approach to Shape Analysis", in *Proceedings of International Conference on Computer Vision*, Nov. 2011, pp. 1626–1633. DOI: 10.1109/ICCVW.2011.6130444.

[71] M. M. Bronstein and A. M. Bronstein, "Shape Recognition with Spectral Distances", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 33, no. 5, pp. 1065–1071, 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.210.

[72] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, 2004, ISBN: 0521540518.

[73] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I The First 30 Years and Fundamentals", *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec. 2011, ISSN: 1070-9932. DOI: 10.1109/MRA.2011.943233.

[74] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[75] B. Triggs, "Proceedings of European Conference on Computer Vision", H. Burkhardt and B. Neumann, Eds., Springer Berlin Heidelberg, 1998, ch. Autocalibration from planar scenes, pp. 89–105, ISBN: 978-3-540-69354-3. DOI: 10.1007/BFb0055661.

[76] M. Pollefeys, R. Koch, and L. V. Gool, "Self-Calibration and Metric Reconstruction Inspite of Varying and Unknown Intrinsic Camera Parameters", *International Journal on Computer Vision*, vol. 32, no. 1, pp. 7–25, 1999, ISSN: 1573-1405. DOI: 10.1023/A:1008109111715.

[77] N. Jiang, Z. Cui, and P. Tan, "A Global Linear Method for Camera Pose Registration", in *Proceedings of International Conference on Computer Vision*, Dec. 2013, pp. 481–488. DOI: 10.1109/ICCV.2013.66.

[78] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 26, no. 6, pp. 756–770, Jun. 2004, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.17.

[79] D. Devarajan, R. J. Radke, and H. Chung, "Distributed metric calibration of ad hoc camera networks", *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 380–403, Aug. 2006, ISSN: 1550-4859. DOI: 10.1145/1167935.1167939.

[80] D. Devarajan, Z. Cheng, and R. Radke, "Calibrating Distributed Camera Networks", *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1625–1639, 2008, ISSN: 0018-9219. DOI: 10.1109/JPROC.2008.928759.

[81] L. Kovács, "Processing Geotagged Image Sets for Collaborative Compositing and View Construction", in *Proceedings of International Conference on Computer Vision Workshops*, Dec. 2013, pp. 460–467. DOI: 10.1109/ICCVW.2013.67.

[82] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal graphs for efficient structure from motion", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587678.

[83] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[84] L. Kneip, H. Li, and Y. Seo, "UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability", in *Proceedings of European Conference on Computer Vision*, Springer International Publishing, 2014, pp. 127–142, ISBN: 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_9.

[85] P. F. Sturm and B. Triggs, "A Factorization Based Algorithm for Multi-Image Projective Structure and Motion", in *Proceedings of European Conference on Computer Vision*, London, UK, UK: Springer-Verlag, 1996, pp. 709–720, ISBN: 3-540-61123-1. DOI: 10.1007/3-540-61123-1_183.

[86] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment – A Modern Synthesis", in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372, ISBN: 978-3-540-44480-0. DOI: 10.1007/3-540-44480-7_21.

[87] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment", *ACM Transactions on Mathematical Software*, vol. 36, no. 1, pp. 1–30, 2009. DOI: 10.1145/1486525.1486527.

[88] M. Brückner, F. Bajramovic, and J. Denzler, "Experimental Evaluation of Relative Pose Estimation Algorithms", in *Proceedings of International Conference on Computer Vision Theory and Applications*, vol. 2, Jan. 2008, pp. 431–438. DOI: 10.5220/0001073704310438.

[89] V. Rodehorst, M. Heinrichs, and O. Hellwich, "Evaluation of relative pose estimation methods for multi-camera setups", in *International Archives of Photogrammetry and Remote Sensing (ISPRS'08)*, 2008, pp. 135–140.

[90] T. Werner and T. Pajdla, "Cheirality in Epipolar Geometry", in *Proceedings of International Conference on Computer Vision*, vol. 1, Jul. 2001, pp. 548–553. DOI: 10.1109/ICCV.2001.937564.

[91] O. Chum, T. Werner, and J. Matas, "Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint", in *Proceedings of International Conference on Pattern Recognition*, vol. 1, Aug. 2004, pp. 112–115. DOI: 10.1109/ICPR.2004.1334020.

[92] M. Havlena, A. Torii, J. Knopp, and T. Pajdla, "Randomized structure from motion based on atomic 3D models from camera triplets", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 2874–2881. DOI: 10.1109/CVPR.2009.5206677.

[93] S. Agarwal *et al.*, "Building Rome in a day", in *Proceedings of International Conference on Computer Vision*, Sep. 2009, pp. 72–79. DOI: 10.1109/ICCV.2009.5459148.

[94] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: A Factorization Method", *International Journal on Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992, ISSN: 1573-1405. DOI: 10.1007/BF00129684.

[95] K. Glashoff and M. M. Bronstein, "Structure from Motion Using Augmented Lagrangian Robust Factorization", in *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, Oct. 2012, pp. 379–386. DOI: 10.1109/3DIMPVT.2012.27.

[96] J.-M. Frahm *et al.*, "Building Rome on a Cloudless Day", in *Proceedings of European Conference on Computer Vision*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 368–381, ISBN: 978-3-642-15561-1. DOI: 10.1007/978-3-642-15561-1_27.

[97] R. Hagege, "Explicit Estimation of Functions Deformations", PhD thesis, Ben-Gurion University, 2009.

[98] M. A. Audette, F. P. Ferrie, and T. M. Peters, "An algorithmic overview of surface registration techniques for medical imaging", *Medical Image Analysis*, vol. 4, no. 3, pp. 201–217, 2000, ISSN: 1361-8415. DOI: 10.1016/S1361-8415(00)00014-1.

[99] G. Wahba, *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990. DOI: 10.1137/1.9781611970128.

[100] J. Mitra *et al.*, "Spectral clustering to model deformations for fast multimodal prostate registration", in *Proceedings of International Conference on Pattern Recognition*, IAPR, Tsukuba Science City, Japan: IEEE, Nov. 2012, pp. 2622–2625.

[101] J. Mitra *et al.*, "A spline-based non-linear diffeomorphism for multimodal prostate registration", *Medical Image Analysis*, vol. 16, no. 6, pp. 1259–1279, Aug. 2012. DOI: 10.1016/j.media.2012.04.006.

[102] G. Peyré and L. Cohen, "Geodesic Methods for Shape and Surface Processing", in *Advances in Computational Vision and Medical Image Processing*, ser. Computational Methods in Applied Sciences, J. M. R. S. Tavares and R. M. N. Jorge, Eds., vol. 13, Springer Netherlands, 2009, pp. 29–56, ISBN: 978-1-4020-9085-1. DOI: 10.1007/978-1-4020-9086-8_2.

[103] G. Christensen, "Consistent Linear-Elastic Transformations for Image Matching", in *Proceedings of Information Processing in Medical Imaging*, Springer-Verlag, 1999, pp. 224–237. DOI: 10.1007/3-540-48714-X_17.

[104] H. Guo, A. Rangarajan, and S. Joshi, "Diffeomorphic Point Matching", in *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds., Springer US, 2006, pp. 205–219, ISBN: 978-0-387-28831-4. DOI: 10.1007/0-387-28831-7_13.

[105] V. Camion and L. Younes, "Geodesic interpolating splines", in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. Lecture Notes in Computer Science, M. Figueiredo, J. Zerubia, and A. Jain, Eds., vol. 2134, Springer Berlin Heidelberg, 2001, pp. 513–527, ISBN: 978-3-540-42523-6. DOI: 10.1007/3-540-44745-8_34.

[106] J. M. Pozo, M.-C. Villa-Uriol, and A. F. Frangi, "Efficient 3D Geometric and Zernike Moments Computation from Unstructured Surface Meshes", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 33, no. 3, pp. 471–484, Mar. 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.139.

[107] P. Koehl, "Fast Recursive Computation of 3D Geometric Moments from Surface Meshes", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 34, no. 11, pp. 2158–2163, 2012. DOI: 10.1109/TPAMI.2012.23.

[108] M. Lourakis, *Levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++*, [web page] http://www.ics.forth.gr/~lourakis/levmar/, Jun. 2004.

[109] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques", *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, Apr. 2003, ISSN: 1077-2626. DOI: 10.1109/TVCG.2003.1196006.

[110] L. Rineau and M. Yvinec, "A generic software design for Delaunay refinement meshing", *Computational Geometry*, vol. 38, no. 1–2, pp. 100–110, 2007, ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2006.11.008.

[111] J. R. Shewchuk, "Delaunay Refinement Algorithms for Triangular Mesh Generation", *Computational Geometry*, vol. 22, no. 1–3, pp. 21–74, 2002, 16th ACM Symposium on Computational Geometry, ISSN: 0925-7721. DOI: 10.1016/S0925-7721(01)00047-5.

[112] A. S. Bryant and R. J. Cerfolio, "The Maximum Standardized Uptake Values on Integrated FDG-PET/CT Is Useful in Differentiating Benign From Malignant Pulmonary Nodules", *The Annals of Thoracic Surgery*, vol. 82, pp. 1016–1020, 2006. DOI: 10.1016/j.athoracsur.2006.03.095.

[113] G. Postelnicu, L. Zollei, R. Desikan, and B. Fischl, "Geometry Driven Volumetric Registration", in *Information Processing in Medical Imaging*, ser. Lecture Notes in Computer Science, N. Karssemeijer and B. Lelieveldt, Eds., vol. 4584, Springer Berlin Heidelberg, 2007, pp. 675–686, ISBN: 978-3-540-73272-3. DOI: 10.1007/978-3-540-73273-0_56.

[114] C. Creusot, N. Pears, and J. Austin, "3D face landmark labelling", in *Proceedings of the ACM workshop on 3D object retrieval*, ACM Press, 2010. DOI: 10.1145/1877808.1877815.

[115] ——, "A Machine-Learning Approach to Keypoint Detection and Landmarking on 3D Meshes", *International Journal on Computer Vision*, vol. 102, no. 1-3, pp. 146–179, Mar. 2013. DOI: 10.1007/s11263-012-0605-9.

[116] P. Nair and A. Cavallaro, "3D Face Detection, Landmark Localization, and Registration Using a Point Distribution Model", *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 611–623, 2009. DOI: 10.1109/tmm.2009.2017629.

[117] P. Perakis, G. Passalis, T. Theoharis, and I. A. Kakadiaris, "3D Facial Landmark Detection & Face Registration", CGL Technical Report, Tech. Rep. TP-2010-01, 2010.

[118] A. Savran *et al.*, "Bosphorus Database for 3D Face Analysis", in *Biometrics and Identity Management*, Springer Science + Business Media, 2008, pp. 47–56. DOI: 10.1007/978-3-540-89991-4_6.

[119] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction", in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06, Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70, ISBN: 3-905673-36-3. DOI: 10.2312/SGP/SGP06/061-070.

[120] P. Alliez, L. Saboret, and G. Guennebaud, "Surface Reconstruction from Point Sets", in *CGAL User and Reference Manual*, 4.4, CGAL Editorial Board, 2000.

[121] H.-J. Zimmermann, *Fuzzy Set Theory – and Its Applications*. Springer Science & Business Media, 2001.

[122] A. Tanács, J. Lindblad, N. Sladoje, and Z. Kato, "Estimation of linear deformations of 2D and 3D fuzzy objects", *Pattern Recognition*, vol. 48, no. 4, pp. 1391–1403, 2015, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.10.006.

[123] M. Lindenbaum and A. Bruckstein, "On Recursive, O(N) Partitioning of a Digitized Curve into Digital Straight Segments", *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 15, no. 9, pp. 949–953, Sep. 1993, ISSN: 0162-8828. DOI: 10.1109/34.232082.

[124] B. R. Vatti, "A Generic Solution to Polygon Clipping", *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, Jul. 1992, ISSN: 0001-0782. DOI: 10.1145/129902.129906.

[125] M. H. Singer, "A general approach to moment calculation for polygons and line segments", *Pattern Recognition*, vol. 26, no. 7, pp. 1019–1028, 1993, ISSN: 0031-3203. DOI: 10.1016/0031-3203(93)90003-F.

[126] J. Flusser, B. Zitova, and T. Suk, *Moments and Moment Invariants in Pattern Recognition*. Wiley Publishing, 2009, ISBN: 9780470699874.

[127] C. Domokos and Z. Kato, "Parametric Estimation of Affine Deformations of Planar Shapes", *Pattern Recognition*, vol. 43, no. 3, pp. 569–578, 2010. DOI: 10.1016/j.patcog.2009.08.013.

[128] ——, "Affine Shape Alignment Using Covariant Gaussian Densities: A Direct Solution", *Journal of Mathematical Imaging and Vision*, vol. 51, no. 3, pp. 385–399, 2015, ISSN: 0924-9907. DOI: 10.1007/s10851-014-0530-3.

[129] N. Geva, R. Hagege, and J. Francos, "Parametric modeling and linear estimation of elastic deformations", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2011, pp. 1301–1304. DOI: 10.1109/ICASSP.2011.5946650.

[130] R. Hagege and J. M. Francos, "A Random Sets Framework for Error Analysis in Estimating Geometric Transformations – A First Order Analysis", in *International Symposium on Information Theory and Its Applications*, Dec. 2008, pp. 1–4. DOI: 10.1109/ISITA.2008.4895442.

[131] R. Hagege and J. M. Francos, "Parametric Estimation of Affine Transformations: An Exact Linear Solution", *Journal of Mathematical Imaging and Vision*, vol. 37, no. 1, pp. 1–16, May 2010, ISSN: 0924-9907. DOI: 10.1007/s10851-009-0188-4.

[132] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code", *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, Apr. 1983, ISSN: 0090-6778. DOI: 10.1109/TCOM.1983.1095851.

[133] J.-M. Frahm, K. Köser, and R. Koch, "Pose estimation for multi-camera systems", in *Pattern Recognition*, Springer, 2004, pp. 286–293. DOI: 10.1007/978-3-540-28649-3_35.

[134] S. Soro and W. Heinzelman, "A survey of visual sensor networks", *Advances in Multimedia*, vol. 2009, 2009. DOI: 10.1155/2009/640386.

[135] R. Tron and R. Vidal, "Distributed computer vision algorithms through distributed averaging", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 57–63. DOI: 10.1109/CVPR.2011.5995654.

[136] I. Junejo and H. Foroosh, "Optimizing PTZ camera calibration from two images", *Machine Vision and Applications*, pp. 1–15, Feb. 2011, ISSN: 0932-8092. DOI: 10.1007/s00138-011-0326-z.

[137] X. Dong and M. C. Vuran, "Vision Graph Construction in Wireless Multimedia Sensor Networks", in *Proceedings of IEEE Global Telecommunications Conference*, IEEE, 2010, pp. 1–5. DOI: 10.1109/glocom.2010.5683899.

[138] W. Stone and Y. Sekercioglu, "Analysis of distributed smart camera calibration accuracy using minimal data transmission", in *Proceedings of IEEE International Conference on Communication Systems (ICCS)*, 2010, pp. 655–661. DOI: 10.1109/ICCS.2010.5686597.

[139] J. Molnár and D. Chetverikov, "Quadratic Transformation for Planar Mapping of Implicit Surfaces", *Journal of Mathematical Imaging and Vision*, vol. 48, no. 1, pp. 176–184, 2014. DOI: 10.1007/s10851-012-0407-2.

[140] Z. Zhang, Y. Matsushita, and Y. Ma, "Camera calibration with lens distortion from low-rank textures", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2321–2328. DOI: 10.1109/CVPR.2011.5995548.

[141] H. Mobahi, Z. Zhou, A. Y. Yang, and Y. Ma, "Holistic 3D reconstruction of urban structures from low-rank textures", in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov. 2011, pp. 593–600. DOI: 10.1109/ICCVW.2011.6130297.

[142] Z. Zhang, A. Ganesh, X. Liang, and Y. Ma, "TILT: Transform Invariant Low-Rank Textures", *International Journal on Computer Vision*, vol. 99, no. 1, pp. 1–24, 2012, ISSN: 0920-5691. DOI: 10.1007/s11263-012-0515-x.

[143] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual Place Recognition with Repetitive Structures", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 883–890. DOI: 10.1109/cvpr.2013.119.

[144] P. Sturm, "Algorithms for plane-based pose estimation", in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, 706–711 vol.1. DOI: 10.1109/cvpr.2000.855889.

[145] P. D. Kovesi, *MATLAB and Octave Functions for Computer Vision and Image Processing*, [web page] http://www.peterkovesi.com/matlabfns/, Mar. 2017.