INSTITUT FÜR INFORMATIK

Monocular Pose Estimation Based on Global and Local Features

Marco Antonio Chavarria Fabila

Bericht Nr. 0922
November 2009

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Monocular Pose Estimation Based on Global and Local Features

Marco Antonio Chavarria Fabila

Bericht Nr. 0922
November 2009

e-mail: mc@ks.informatik.uni-kiel.de

Dieser Bericht gibt den Inhalt der Dissertation wieder, die der Verfasser im Januar 2007 bei der Technischen Fakultät der Christian-Albrechts-Universität zu Kiel eingereicht hat.
Datum der Disputation: 05. Juni 2009.

1. Gutachter	Prof. Gerald Sommer (Kiel)
2. Gutachter	Prof. Bodo Rosenhahn (Hannover)

Datum der mündlichen Prüfung: 05.06.2009.

ABSTRACT

The presented thesis work deals with several mathematical and practical aspects of the monocular pose estimation problem. Pose estimation means to estimate the position and orientation of a model object with respect to a camera used as a sensor element. Three main aspects of the pose estimation problem are considered. These are the model representations, correspondence search and pose computation. Free-form contours and surfaces are considered for the approaches presented in this work. The pose estimation problem and the global representation of free-form contours and surfaces are defined in the mathematical framework of the conformal geometric algebra (CGA), which allows a compact and linear modeling of the monocular pose estimation scenario. Additionally, a new local representation of these entities is presented which is also defined in CGA. Furthermore, it allows the extraction of local feature information of these models in 3D space and in the image plane. This local information is combined with the global contour information obtained from the global representations in order to improve the pose estimation algorithms. The main contribution of this work is the introduction of new variants of the iterative closest point (ICP) algorithm based on the combination of local and global features. Sets of compatible model and image features are obtained from the proposed local model representation of free-form contours. This allows to translate the correspondence search problem onto the image plane and to use the feature information to develop new correspondence search criteria. The structural ICP algorithm is defined as a variant of the classical ICP algorithm with additional model and image structural constraints. Initially, this new variant is applied to planar 3D free-form contours. Then, the feature extraction process is adapted to the case of free-form surfaces. This allows to define the correlation ICP algorithm for free-form surfaces. In this case, the minimal Euclidean distance criterion is replaced by a feature correlation measure. The addition of structural information in the search process results in better conditioned correspondences and therefore in a better computed pose. Furthermore, global information (position and orientation) is used in combination with the correlation ICP to simplify and improve the pre-alignment approaches for the monocular pose estimation. Finally, all the presented approaches are combined to handle the pose estimation of surfaces when partial occlusions are present in the image. Experiments made on synthetic and real data are presented to demonstrate the robustness and behavior of the new ICP variants in comparison with standard approaches.

ACKNOWLEDGMENT

I thank to all the people who in a way or another helped me in the elaboration of this thesis work. First at all, I thank to my supervisor Prof. Dr. Gerald Sommer for giving me the opportunity to collaborate in the Cognitive System Group. I thank him for supporting me to obtain my scholarship, for suggesting me this interesting research topic, for his advice and comments to solve the different research problems and for supporting the conferences I visited. Especially, I want to thank him for his patience also in the difficult moments.

Furthermore, I thank all the members and ex-members of the Cognitive System Group for their help and support. Especially to Bodo Rosenhahn and Oliver Granert with whom I had the first discussions about geometric algebras and the pose estimation problem. That helped me a lot to understand the main theoretical and practical aspects of the pose estimation problem. I thank the help and support of Nils Siebel, Christian Perwass, Yohannes Kassahun, Di Zang, Florian Hoppe, Herward Prehn and Christian Gebken who always were ready to answer my questions and to share valuable comments, which in some way or another are reflected in this work. The friendly and pleasant atmosphere in the group made the research work very stimulating. I also thank to my colleagues Sven Buchholz, Lennart Wietzke and Stephan Zeitschel for proofreading the different chapters of this thesis work and for suggesting me improvements of the contents and redaction. I appreciate the help of the technical staff, Henrik Schmidt, Gerd Diesner and the secretaries of the group Françoise Maillard and Armgard Wichmann. A special thank to the staff of the International Center of the Christian Albrechts University for their help during my stay at the University of Kiel.

I thank the Mexican Council of Science and Technology (CONACYT) for giving me the scholarship (number of registry: 144161) to support of my doctoral studies. In a similar way, I thank the German Service of Academic Exchange (DAAD) for giving me the opportunity me to participate in a language course, where I started to learn more about the culture of the German people.

Last, but not at least, I thank my family for their unconditional support and encouragement. Thanks to my parents Martin and Gloria, to my sister Diana and to my brother Efrain. I dedicate this thesis work to my family and especially to my beloved wife Caterina Maria Carrara, whose encouraging and loving words during these years gave me the strength to accomplish this work.

CONTENTS

Chapter 1

INTRODUCTION

Computer vision and robotic systems are among the most innovative and important research topics with an enormous potential of practical applications. No other technical innovation has developed so fast than computers in the recent years. Since the first personal computer developed in the seventy years, every new generation of computers is faster, cheaper and more compact. Hence, efficient applications based on computer systems are available nowadays for almost every institution, industry and individuals in almost all aspects of modern life. This has also allowed the use of robots in a wide variety of applications, e.g. in entertainment systems, industrial applications and missions to explore other planets among others. Robots are used in those situations where the work is too dangerous, hard and repetitive and in cases where humans can not perform a certain task with high accuracy. In this context, a big challenge for research has been to emulate in robots the basic human behavior to solve problems. That means, the way that humans perceive information and take decisions to solve a specific problem. In this framework, the perception-action cycle defines the way that robots interact with their environment to solve a certain task. Once that all possible informations of the environment have been acquired and interpreted in the perception phase, a set of decisions are taken in the action phase.

Let us concentrate on the perception phase. A description about the size and structure of the robot's environment must be generated. On the other hand, the location of the robot and any other objects must be determined in order to plan a possible action strategy. In order to obtain this information, modern robot systems are equipped with a large variety of sensor elements. The most adequate sensors are chosen depending on the specific task to be solved. Infrared or ultrasonic sensors are commonly used for robot navigation applications. Thermal and humidity sensors are also used to work in industrial environments. Modern sensors based on GPS (global positioning systems) are used for robot localization and navigation. Although these systems are able to deliver very precise measurements, they are limited to a specific parameter of the world.

In contrast to the above mentioned sensor elements, the use of visual information has become more popular in the last years. The most important way to get information about the world is through vision. Vision allows humans to perceive a huge amount of information, e.g. shape, color, dimension, etc. It is the principle to con-

struct more general and abstract interpretations of the world. That means, humans learn and take decisions basically from visual information. Because of that, the aim is to simulate the human eye as sensor element by video cameras. Similarly, the function of the human brain as visual processing center is simulated by a computational system. The processing and interpretation of visual information has become an important research topic in the fields of artificial intelligence and cognitive systems. Among the most common applications are controlling processes, detection of events, organization of information, modeling of objects and environments and human-machine interaction.

As mentioned before, one of the main tasks that a robot system must solve to interact with its environment is to determine its location and the location of additional objects. This is known as "Pose Estimation Problem". Roughly speaking, it is defined in [49] as follows:

Definition 1.1 *Pose estimation is defined as the transformation (rigid body motion) to map an object model from its initial position to the actual position in agreement with the sensor data.*

According to the last definition, the pose estimation problem is classified depending on how model and sensor data are defined as 2D-2D, 3D-3D and 2D-3D, see [28]. If sensor and model data are defined in the image plane, the problem is defined as 2D-2D. In this case, a rigid body motion in the image plane is obtained from the pose computation. If model and sensor data are defined in 3D space, the problem is considered as 3D-3D and the computed rigid body motion is also defined in 3D space. The pose estimation problem is defined as 2D-3D if 3D information is recovered from the image and the rigid body motion is computed in 3D space. Many authors refer to the pose estimation problem with different names, although the idea of computing a rigid body motion with respect to sensor data remains the same. When the pose is computed for a sequence of images, it is known as 2D or 3D tracking respectively [61]. The pose estimation problem is also known as 3D registration [64] when two clouds of points in 3D, or a cloud of points and a given model are matched.

In the context of this work, the monocular 2D-3D pose estimation for 3D free-form contours and surfaces is considered in the mathematical framework of geometric algebras. Recently, geometric algebras have been introduced in computer vision as a problem adaptive algebraic language in case of modeling geometry related problems, see [55, 102]. The main advantage of geometric algebras over standard representation methods is that they allow linear and compact symbolic representations of higher order entities. In the context of the algebra, a higher order entity can be interpreted as a subspace of a given vector space. For example, lines and planes can be considered higher order entities since they represent one and two dimensional subspaces of $\mathbb{R}^3$. Several geometric concepts and operations defined in different algebras are unified in geometric algebra. For instance, the principle of

duality in projective geometry, Lie algebras and Lie groups, Plücker representation of lines, real numbers and quaternions among others. Because of the last properties, the use of geometric algebra allows to define linear operators acting on these different geometric entities. Furthermore, it is also possible to define the image formation process in the case of projective cameras, see [9, 33, 85].

As it was discussed by Rosenhahn in [87], several aspects must be considered in order to define the pose estimation problem. First, a mathematical framework must be defined in order to model the geometry and mathematical spaces involved in the problem. Based on that mathematical framework, rigid body motions and the respective free-form models must be accordingly described. Additionally, operators to define the best way to fit 3D object data to image information are needed. Finally, image features must be extracted (originally, points, lines and planes) according to the available model object and pose scenario. The main contribution of Rosenhahn's work was the unification of all these concepts in the mathematical framework of the conformal geometric algebra (CGA). Roughly speaking, the conformal space (on which the conformal geometric algebra is defined) is a vector space generated as a geometrical embedding of the Euclidean space by a stereographic projection [44]. It turned out that the conformal geometric algebra is especially useful because its ability of handling stratified geometrical spaces, which is an essential property for the definition of the pose estimation problem, see [92].

In a further work, Rosenhahn extended the pose estimation approaches in CGA for model objects like circles, ellipses, spheres, free-form contours and surfaces, see [89, 90, 91]. To achieve that, the idea of twist rotations plays a significant role to represent contours and surfaces. Initially, twists were used to model rigid body motions in CGA. By exploiding their similarity with the exponential representation of the Fourier transform of closed contours, a set of twist descriptors (geometrically similar to a set of coupled rotations) are obtained. Then, it was possible to define the pose estimation constraints in CGA of free-form objects.

It is worth to mention that geometric algebras have been applied to other computer vision problems. The image formation process based on catadioptric and stereographic projections has been defined in [107]. Then, the 2D-3D pose estimation problem with catadioptric cameras was formulated for applications to mobile robots, see [46]. In the case of multidimensional signal theory, a local geometrical modeling of image signals was proposed to develop new edge and corner detectors [110, 111]. The theoretical basis for neuronal network applications with geometric algebras has been introduced in [21].

1.1 Motivation

The aim of the present work is to propose mathematical and practical approaches to improve the monocular pose estimation problem. In a first instance, the monocular

pose estimation problem is divided in the following three main tasks: acquisition of data from the image, correspondence search and pose computation. In the acquisition step, feature information is extracted from the image and from the object model. This feature information is used to find correspondence pairs (model and image). Finally, the correspondences are used to define constraints in order to compute the pose parameters. In this work, several strategies are proposed to obtain compatible feature information from image and model data. The main contributions are focused on the development of correspondence search strategies which use such feature information. The efficient integration of these tasks may result in good quality pose estimation algorithms in terms of robustness, convergence behavior and time performance.

For the pose estimation problem, finding proper correspondences is one of the major challenges to solve. In general, the correspondence search problem is defined as follows:

Definition 1.2 *Given two sets of points $\mathcal{A}$ and $\mathcal{B}$, correspondence pairs of "similar" points must be found. Additionally, the functional f measures the similarity of points with respect to their feature information. Therefore, for every point $x_i \in \mathcal{A}$, its corresponding point $y_j \in \mathcal{B}$ is the one which follows the criterion*

$$\mathrm{corr}(x_i, \mathcal{B}) \quad = \quad \max_{\mathrm{j}=1,\ldots,n} \left\{ f\left(x_i, y_j\right) \right\}. \tag{1.1}$$

According to the last definition, two points form a correspondence pair if their similarity measure is maximal. Analog to the classification of the pose estimation problem, the correspondence search can be also classified depending on how model and sensor data are defined, i.e. between image data (2D-2D), data in Euclidean space (3D-3D) and between image and Euclidean space data (2D-3D). In general, the classical variant of ICP (iterative closest point) algorithm [13] is the most common approach which is used in most of the scenarios. It uses the Euclidean distance between model and sensor data as correspondence search criterion.

If the correspondences are correct, any pose estimation algorithm will deliver a correct pose within ceratin limits. In real applications, false correspondences due to occlusions, noise or other perturbations are present during the estimation process. Therefore, a correspondence search criterion must be defined which makes possible to find better correspondences. To achieve that, new variants of the ICP algorithm are presented in this work which combine Euclidean distance with additional global and local feature information. The choice of the best strategy to define a correspondence search criterion based on feature information depends in general on the following considerations:

1. **Global and local object representations**. For applications in robotics and computer vision, it is desirable to gain as much information as possible about

the objects of interest in order to solve a certain task. The object representation must satisfy certain mathematical and geometric properties. Ideally, it must be able to define the object by a complete, unique and if possible compact representation. In most of the cases, a compromise between these properties must be made for practical applications. One important question is how to represent such objects (free-form contours and surfaces), which eventually must be detected and segmented in the images. On the other hand, the concept and representation of an object can be regarded as global or local. Each representation offers certain advantages in a mathematical point of view and for its implementation. Intuitively, the global concept refers to representations which involve the complete model and therefore useful information can only be extracted if the complete model is available. On the other hand, a local representation does not need the complete object to approximate patches or regions of interest of the objects. As mentioned before, a global representation of free-form contours and surfaces was proposed by Rosenhahn. Hence, a new local representation in CGA is proposed which is able to deliver local feature information.

2. **Global and local features**. Feature based approaches are well known in the context of computer vision problems. Evermore, the concept of fusing global and local approaches can be also applied to the correspondence search problem. Global features describe a complete object or image information in a way which can be useful to certain applications. Sensitiveness to occlusions, clutter and the need of segmentation approaches are the main drawbacks of global feature based algorithms. In contrast to that, local features do not need a complete segmentation procedure and they are robust against occlusions and clutter. Since local features describe structures of different nature and sometimes the locally described region may have different ranges, the combination of several kinds of local features into standard techniques may be difficult. According to these properties, local and global feature approaches provide essentially independent structural information. Such information can be combined in order to get a better description of model objects and image data. In the case of 3D free-form contour and surfaces, it is only possible to obtain global and local features from contour segments or complete contours in 3D space or in the image plane according to its global and local model representations. Then, several methods are proposed in this work to obtain adequate contour features which can be used in the correspondence search problem.

3. **Tracking assumption**. In the context of the pose estimation problem, the tracking assumption defines the inherent convergence limits within which a pose estimation algorithm is able to compute a correct pose. In some applications, sequences of pictures are previously captured. Then, the conditions to get proper correspondences and a correct pose are in general in advance predetermined. That means, the tracking assumption condition is considered. A model object is considered to be under tracking assumption conditions if its position

and orientation difference with respect to the image data is small enough to avoid convergence to a local minimum. Other possible scenario is when the system has to locate the objects in order to make decisions, for example for robot navigation or object manipulation with a robot arm. In these cases, the information of the image has to be processed, the decision must be made and a new information must be acquired to continue the loop (perception action cycle). During this cycle, unexpected changes in the robot work environment or rapid movements of robot or objects may cause larger displacements between model and image. In these cases, the tracking assumption is not met any more. In both of the described situations, the application of a different correspondence search strategies is needed. Therefore, the proposed solutions to the correspondence search problem must be robust against the tracking assumption.

1.2 Literature Overview

In this section, a review of the most representative contributions in the literature regarding to the main aspects of the correspondence search problem in context of the pose estimation problem are presented.

Model Representations

Point and line information are commonly used to represent model objects for the monocular pose estimation problem. In consequence, the same information is extracted from the image plane, see [3, 51, 71, 86]. In contrast to the model representation of free-form objects in CGA, most of the typical object representations presented in the literature use more intuitive geometrical concepts than formal mathematical representations to describe these entities. In general, the term free-form object [24, 38, 60, 112] is applied to such contours or surfaces which can not be easily described by an explicit parametric function or by a set of planar and regular patches. Human faces and organs, car models, ships, airplanes, terrain maps and industrial parts among others are typical examples of objects modeled as free-form objects (see Besl [13] and Stein and Medioni [105]).

A review of free-form object representations and their applications to computer vision problems was presented by Cambell and Flynn in [24]. Surfaces can be implicitly modeled as the zero-set of an arbitrary function [106]. In general, these functions are defined by a polynomial of certain degree. One main disadvantage of this approach is that the coefficients of the polynomial are in general not invariant to pose transformations. Surfaces are also defined by implicit functions called superquadrics where a surface is defined by volumetric primitives [101]. In this case, the simplest primitive is the sphere. Then more complex surfaces are generated by

modifying and adding extra parameters to the basic superquadric equation like tapering, twisting and bending deformations. Due to the basis equations to generate the curves, superquadrics are able to model coarse shapes of models. Because of that, superquadrics lack of the ability to describe fine details of models and therefore they can not deliver good quality local structural features. Other possibility is to model surfaces as generalized cylinders [1, 81]. Once that the main orientation axis of the 3D points are obtained, a set of contours are constructed in the planes normal to this axis defining the boundary of the surface. It is clear that this approach is well suitable to define elongated objects. Finally, other common object representation important for our purposes is based on polygonal meshes [24]. A mesh consists of a set of triangles, rectangles or any polygonal patches which define a surface. Due to the possibility of defining the resolution of the polygonal patches, complex free-form objects can be modeled with a relatively good accuracy.

Local and Global Contour Features

In the case of free-form contours and surfaces, local features are commonly computed from the curvature of contour segments or surface patches respectively, see [14, 53]. In the case of the image plane, digital 8-neighbor connected curves are obtained. Then, an angular measure based on the reference point p_i and its neighbors p_{i+k} and p_{i-k} is used to estimate the curvature in [52]. According to that, the curvature is defined with respect to changes in the orientations of the tangent of a point p_i. The approach presented in [73] approximates the digital curve at the interest point by second order polynomials. With this local approximation, the curvature is computed by applying derivatives of the curve. The complete digital curve is approximated by cubic B-splines in [77] in order to compute the curvature with respect to the approximated curve. Osculating circles are used in [31] to compute the curvature of discrete contour points. In this case, a circle is constructed with three contour points where the curvature is defined by the length of its radius.

In the case of images, segmentation procedures must be applied in order to extract contour information and eventually to compute the local features. To achieve that, approaches based on the monogenic scale space for two-dimensional signal processing introduced by Felsberg and Sommer are used, see [40, 41]. The monogenic signal is a generalization of the analytic signal [43] which delivers feature information of image points in terms of local amplitude, phase and orientation. The local amplitude encodes information about local energy (presence of structure) while the phase refers to the local structure (symmetry of the signal). That means, additional local structure is used to describe contours in the image plane.

Global features can be computed from closed contours. Global position and orientation are computed by different approaches depending of the kind of object models, sensor devices and pose estimation scenarios. Among the most common approaches are the principal component analysis (PCA) as proposed in [20]. In this

case, the main orientation axes correspond to the distribution directions derived from the eigenvectors and eigenvalues of the covariance matrix computed with the set of points in 3D. For image contours, the main distribution axes can be extracted from the elliptic Fourier descriptors as defined in [58, 63].

Correspondence Search and ICP Algorithm

The ICP algorithm was originally used for correspondence search by Besl and McKay [13]. Because of its relatively low computation complexity, variants of the classical ICP algorithm are applied for tracking and pose estimation applications where real time performance is expected. From this original idea, several variations and improvements have been made. Chen and Medioni [29] use the sum of square distance between scene and model points in their ICP variant. An extension of this work was made by Dorai and Jain [35], where an optimal uniform weighting of points is used. Other variants define as error measure the absolute difference for each coordinate component of the point pairs [108]. Zhang [112] uses a modified ICP algorithm which includes robust statistics and adaptive thresholding to deal with the occlusion problem. Instead of a point-point distance metric, correspondences are found by the minimal distance between points and tangent planes of surfaces in the approach proposed by Dorai et al. [35]. In order to accelerate the search process, Benjemaa and Schmitt use z-buffers of surfaces [10] where the search is translated to a specific area of the z-buffers. A comparison of variants of the ICP algorithm is presented in [94], where the different variants are applied to align artificially generated 3D meshes.

Other variants of the ICP algorithm perform a selection of corresponding points. Once that all the correspondences are found, only a certain number of points are selected by a given criterion. Chetverikov et al. developed the trimmed ICP algorithm [30] based on the Least Trimmed Squares (LTS) approach [6]. In this case, correspondence pairs are sorted by their least square errors and only a certain number of them are used to compute the pose. This selection process is repeated in all steps of the algorithm. A hierarchical control point selection is used in the work of Zinsser et al. [113]. The iteration process is divided in $h + 1$ hierarchy levels, where only every $2^h - i$ points are used as control points for each level. Once that the algorithm converges for a set of control points, the pose is computed with the control points of the next level. A similar idea is used in the ICP algorithm proposed by Masuda and Yokoya [75] where control points are chosen randomly to define correspondences. Once that the pose is computed, it is evaluated with a given error measure. This is repeated in an internal loop until the pose with the minimal error is found. Then, this optimal pose is applied to the model for the next iteration. The last ICP variants are useful for incomplete sensor data especially for 2D-2D and 3D-3D pose estimation scenarios. Most of these above cited methods seem to be robust and suitable for real time applications. Since all of them are based on information obtained from a single point, the tracking assumption must be considered in order to ensure an ef-

ficient performance and to guarantee that the algorithms do not converge to a local minimum.

The ICP algorithm is combined with additional features obtained from relatively complex image processing approaches like optical flow [19, 88] or bounded Hough transform [98]. In these cases, the algorithms show a better performance in comparison with the normal ICP algorithm in a monocular pose estimation scenario. The tracking assumption can be slightly overcome but the computational cost increases considerably. In some cases, computation times up to 2 minutes per frame are reported.

An evaluation of 2D-3D correspondences is made in the work of Liu [66]. Additional rigid body motion constraints are defined which determine the necessary conditions for a point par to form a good correspondence. A further extension of this work exploits the collinearity and closeness (point-line and point-point) constraints without the need of any feature extraction [65, 67]. The approach proposed by Shang et al. [99] uses known information about the limits of the object velocity and the image frame rate to reduce the transformation space between every frame. Then, the tracking is transformed into a classification problem.

In the work of Najafi et al. [80], appearance models (more realistic models which include colors, textures and the effect of light reflections) are used to find correspondences. Panin and Knoll [83, 84] presented a system that combines active contours [57] and appearance models. This system uses a global pose estimation module based on a robust feature matching procedure. A reference image of the object is used as model to compare and globally match the features with the current image of the sequence. On the other hand, a local contour tracking is used to compute the exact pose. After every frame, the coherence of the computed pose is checked by the normalized cross correlation index. Once that the reference image (model) is rendered onto the image plane at the computed pose, the cross correlation index between image and reference is computed. If it is below a given reference value, the computed pose is considered a good initialization for the next frame. Otherwise, the global pose estimation is used. A quite similar idea is used by Ladikos et al. [59] where feature and template-based tracking systems are combined. In this case, the 3D objects are modeled by a set of planar textured patches. The system switches to feature tracking if the cross correlation index between model patches and image are above a threshold value.

The most relevant approach for this thesis work is the contribution of Sharp et al. [100], where additional invariant features like curvature, moment invariants [95, 50] and spherical harmonics [23] are used to define a variant of ICP algorithm for 3D registration. The particularity of this method is that an extended similarity measure function is defined as a combination of point and feature Euclidean distance. Although this combination improves the correspondence search, it can not be applied to the monocular pose estimation since curvature and moments are not invariant under perspective projection.

1.3 Contributions and Outlook

Once that the general concepts of the correspondence search problem have been presented, the main contributions of this work are summarized. In order to find correspondences between model and image points, two strategies are possible:

1. Reconstruct 3D information from the 2D image feature information in order to solve the correspondence problem in 3D space.
2. Project the 3D contour or surface points of the object model onto the image plane and translate the correspondence search problem completely to 2D.

Essentially, information in 3D space is different than the information in the image plane. 3D features are in general not invariant against perspective projections. If only one image is used to compute the pose, 3D information can not be completely reconstructed from the image. If the 3D models are projected onto the image plane, there are more possibilities to exploit more image information and to find equivalent features from the projected models. Because of that, the second strategy is followed in order to develop the new variants of the ICP algorithm. A schematic diagram of the general strategy is presented in figure 1.1.

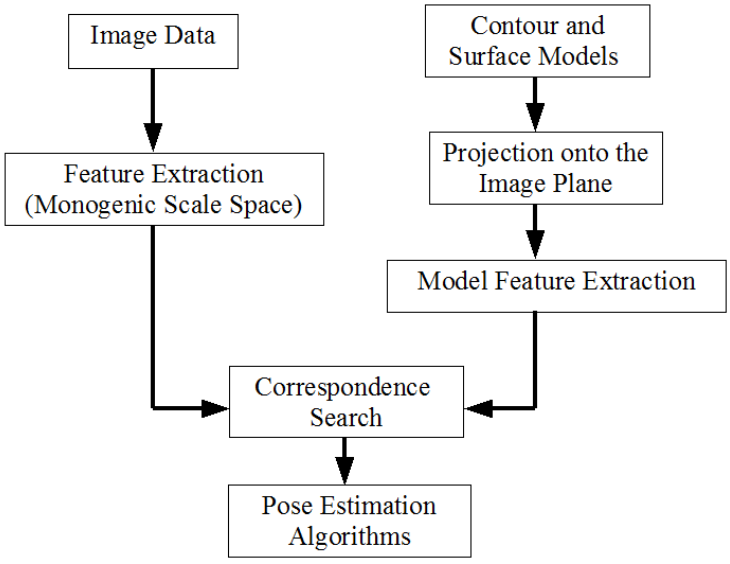

Fig. 1.1: General strategy to translate the correspondence problem from the 3D Euclidean space to the 2D image plane.

The present thesis work is organized as follows:

In chapter 2, the pose estimation problem is introduced in the framework of geometric algebras. Initially, the basic definitions and properties of geometric algebras are presented. The associated algebras of Euclidean, projective and conformal

space are defined. Basic geometric entities like points, lines and planes as well as the image formation process are defined in projective geometric algebra. On the other hand, rigid body motions are modeled in conformal geometric algebra. Then, the interaction of these entities is described in order to define the pose estimation constraints.

The use of twist rotations as global and local contour and surface generator elements is discussed in chapter 3. In a first instance, the concepts of free-form models and Fourier contour and surface approximations are introduced. Then, they are coupled with the pose estimation constraints in CGA. Additionally, contour and surface approximations are obtained from the quaternion valued Fourier transform and Hartley transform which offer an alternative to the complex valued Fourier approximations for practical applications. For instance, they allow the extraction of global orientation information. By following the idea of twist rotations, a new local representation of free-form contours and surfaces in the framework of the conformal geometric algebra is proposed in this chapter (published in [25]). This new representation allows to generate and describe the vicinity of a model point, larger segments or even the complete model by a pivot point and a set of concatenated twist rotations. Therefore, it can be used in the pose minimization constraints defined by Rosenhahn. Additionally, this local representation allows to extract local features information from contour segments.

One of the central contributions of this work is presented in chapter 4. In order to extract feature information of the image, the monogenic scale space is used to detect local structure and edge information at different scales. Then, a contour search algorithm is applied to extract iteratively contours from the image at different scales. Based on the geometrical properties of the proposed local model representation in the image plane, a set of analog local features to that of the monogenic scale space are extracted. From the local orientation of model points, a local feature defined as transition index is presented. It represents an artificially generated structural feature analog to the local phase. Additional to these features, model and image contours are segmented according to their structure in straight, convex and concave segments. Additional constraints are defined with these features in order to define the structural ICP algorithm (published in [26]) for the case of planar 3D free-form contours.

The procedures for the contour feature extraction were extended for the case of free-form surfaces in chapter 5. Under some considerations, the local features are extracted from the projected silhouette of the model with respect to the image plane. Then, the structural ICP algorithm can be applied. In contrast to the normal ICP variants, the feature information of larger contour segments is used to define profile vectors. Then, each contour point has an associated profile vector that describes its structural information. Other variant of ICP algorithm is presented in this chapter where the Euclidean distance is replaced by a feature measurement based on the correlation matrix defined by profile vectors of model and image points (published in [27]). For cases where the displacement between model and image data

is too large (non-tracking assumption conditions), the local orientation information is combined with the global orientation obtained from the Hartley transform in 2D. Instead of a normal pre-alignment approach, a simple feature pre-alignment in the image plane is done in order to place the model under tracking-assumption conditions. A set of experiments done with artificially generated data and real images are presented in chapter 6. Those experiments focus on testing of the main properties of the new variants of ICP algorithm: robustness against the tracking assumption and convergence behavior. The new ICP variants were extensively compared with the classical variants of the ICP algorithm.

In chapter 7, the problem of the pose estimation of free-form surfaces with partial occlusions under non-tracking assumption conditions is discussed. Additional positional features are combined with the correlation ICP algorithm as outlier elimination criterion. In this case, the angular position of silhouette points with respect to its global main orientation is used. Several combinations of correspondence search criteria and outlier elimination are analyzed which have several properties and behaviors. A system that selects the best combination according to the global position, orientation and absolute pose error is presented.

Finally, the discussion is presented in chapter 8. Several possibilities are mentioned in order to improve the presented algorithms and to apply them to more general scenarios of the pose estimation problem.

Chapter 2

POSE ESTIMATION IN THE LANGUAGE OF GEOMETRIC ALGEBRA

Geometric algebras have become an important mathematical framework to model geometrically related problems by focusing on the geometrical interpretation of algebraic entities. Prior to the introduction of the Clifford algebra, Hamilton (1805-1865) proposed quaternions as a generalization of complex numbers and applied them to model rotations in 3D space. Other important contribution is the exterior (also called outer) algebra developed by Grassman (1809-1877). In this case, the outer product of vectors is an algebraic construction that generalizes certain features of the cross product to higher dimensions. Those ideas were taken by Clifford (1845-1879) in order to develop his Clifford algebra. Essentially, Clifford made a slight modification of the exterior product (he introduced the geometric product), which allowed him to define quaternions and Grassman's exterior algebra into the same algebraic framework.

In this chapter, an introduction to geometric algebras and the definition of the pose estimation problem in this mathematical framework are given. For a more general and detailed introduction, the reader should refer to [8, 15, 37, 85, 92, 93, 102]. Since this chapter serves as an introduction, the reader may also refer to the above mentioned bibliography for details about the presented definitions and properties. The notation and definitions used in this chapter are based on the last cited works. Initially, the geometric algebra of Euclidean space $\mathbb{R}^3$ as well as its relation with the algebra of quaternions is presented. Then, the corresponding algebras of projective and conformal spaces are introduced. The image formation process and basic geometrical entities (points, lines and planes) are defined in projective geometrical algebra. On the other hand, rigid body motions and operations defining points-line and point-plane incidence relations are defined in conformal geometric algebra. Finally, all these concepts are used to define two different pose estimation constraints. They are based on minimization in 3D space and a projective variant based on minimization in the image plane respectively.

2.1 Definition of Geometric Algebra

In this section, the definition of geometric algebra, its elements (multivectors) and its associated product (geometric product) are presented. A geometric algebra denoted by $\mathcal{G}_{p,q,r}$ is a linear space spanned by 2^n, with $n = p + q + r$, basis elements constructed from a vector space $\mathbb{R}^{p,q,r}$. This linear space is defined by subspaces called blades which represent elements called multivectors as higher grade entities.[1] The indices p, q, r refer to the signature of the corresponding basis vectors. The vector space is formed by p, q and r basis vectors which square to $1, -1$ and 0, respectively. According to that, a proper signature can be chosen to define spaces with certain geometrical properties.

2.1.1 Geometric Product

The product operation associated with the geometric algebra is the geometric product. For two vectors $\mathbf{a}$ and $\mathbf{b}$, the geometric product is defined by

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}. \tag{2.1}$$

Then, the geometric product is constructed by the combination of an inner "$\cdot$" and an outer product "$\wedge$". Let us consider the case of two orthonormal elements $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^{p,q,r}$ of a given vector space. In general, the geometric product of these elements leads to a scalar $\mathbf{e}_i\mathbf{e}_j := 1 \in \mathbb{R}$ if $i = j$. The last implies that the vectors are the same. Otherwise, the geometric product leads to a new entity called bivector (grade two) representing the space spanned by these two vectors: $\mathbf{e}_{ij} = \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i$ if $i \neq j$. These elements are the basis elements of geometric algebra. Then, geometric algebras are expressed on the basis of graded elements.

A general multivector $\mathbf{M}$ is formed by a linear combination of elements of different grades as

$$\mathbf{M} = \sum_{i=0}^{n} \langle \mathbf{M} \rangle_i, \tag{2.2}$$

where each element $\langle \mathbf{M} \rangle_i$ represents each part of the multivector with respect to its grade i. According to that, $\langle \mathbf{M} \rangle_0$ is the scalar part of $\mathbf{M}$, $\langle \mathbf{M} \rangle_1$ and $\langle \mathbf{M} \rangle_2$ are the vector and bivector parts respectively. If an element is constructed as the outer product of k independent vectors, it is called a blade of grade k. Then, the blade of the vectors $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_k$ is defined as

$$\mathbf{A}_{\langle k \rangle} := \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge ... \wedge \mathbf{e}_k. \tag{2.3}$$

By using the properties of the geometric product (see [85]), it is possible to express the inner and outer product of two vectors in terms of their geometric product.

[1] Note that vectors are considered as first grade entities.

For two vectors $\mathbf{a}, \mathbf{b} \in \langle \mathcal{G}_{p,q} \rangle_1$, its respective inner and outer products are defined as

$$\mathbf{a} \cdot \mathbf{b} := \frac{1}{2}(\mathbf{ab} + \mathbf{ba}) \tag{2.4}$$

$$\mathbf{a} \wedge \mathbf{b} := \frac{1}{2}(\mathbf{ab} - \mathbf{ba}). \tag{2.5}$$

In order to clarify the operations used in the examples presented during this chapter, some rules for the computation of the inner product between blades are presented. According to [85], the inner product of a vector $\mathbf{x}$ (grade 1) with a blade $\mathbf{A}_{\langle k \rangle}$ (with grade $k \geq 1$) results in a blade of grade $k-1$. The last can be expressed in the next equation:

$$\begin{aligned} \mathbf{x} \cdot \mathbf{A}_{\langle k \rangle} &= (\mathbf{x} \cdot \mathbf{e}_1)(\mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4 \wedge \ldots \wedge \mathbf{e}_k) \\ &\quad -(\mathbf{x} \cdot \mathbf{e}_2)(\mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4 \wedge \ldots \wedge \mathbf{e}_k) \\ &\quad +(\mathbf{x} \cdot \mathbf{e}_3)(\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_4 \wedge \ldots \wedge \mathbf{e}_k) \\ &\quad -\ldots \\ &= \sum_{i=1}^{k} (-1)^{i+1} (\mathbf{x} \cdot \mathbf{e}_i)[\mathbf{A}_{\langle k \rangle} \backslash \mathbf{e}_i], \end{aligned} \tag{2.6}$$

where $[\mathbf{A}_{\langle k \rangle} \backslash \mathbf{e}_i]$ is the blade $\mathbf{A}_{\langle k \rangle}$ without the vector $\mathbf{e}_i$.

A more general operation is the inner product of two blades $\mathbf{A}_{\langle k \rangle}$ and $\mathbf{B}_{\langle k \rangle}$ (with $0 < k \leq l \leq n$). It can be computed as

$$\mathbf{A}_{\langle k \rangle} \cdot \mathbf{B}_{\langle l \rangle} = \mathbf{e}_1 \cdot \left(\mathbf{e}_2 \cdot (\ldots \cdot (\mathbf{e}_\mathrm{k} \cdot \mathbf{B}_{\langle l \rangle})) \right), \tag{2.7}$$

with the resulting blade of grade $l - k$. In general, the inner product reduces the grade of a blade while the outer product increases its grade.

A generalization of the geometric product of vectors can be done for the product of general multivectors. Thus, the geometric product as the combination of inner and outer product of two vectors is extended by applying the properties or the geometric product (see [85]). For two general multivectors $\mathbf{A}, \mathbf{B} \in \mathcal{G}_3$, the definition of equations (2.4) and (2.5) is used to define the geometric product as

$$\begin{aligned} \mathbf{AB} &= \frac{1}{2}(\mathbf{AB} + \mathbf{BA}) + \frac{1}{2}(\mathbf{AB} - \mathbf{BA}) \\ &:= \mathbf{A} \overline{\times} \mathbf{B} + \mathbf{A} \underline{\times} \mathbf{B}, \end{aligned} \tag{2.8}$$

where the analog operators of the inner and outer products for the case of general multivectors are called commutator $\overline{\times}$ and anticommutator $\underline{\times}$ product respectively.

Additional to the inner and outer product, other important operations with blades are defined in the algebra. One of these operations is the reverse $\widetilde{\mathbf{A}}_{\langle k \rangle}$ of a blade $\mathbf{A}_{\langle k \rangle} = \mathbf{e}_1 \wedge \mathbf{e}_2 \ldots \wedge \mathbf{e}_{k-1} \wedge \mathbf{e}_\mathrm{k}$, defined as

$$\widetilde{\mathbf{A}}_{\langle k \rangle} = \mathbf{e}_k \wedge \mathbf{e}_{k-1} \wedge \ldots \wedge \mathbf{e}_2 \wedge \mathbf{e}_1. \tag{2.9}$$

Because of the associative and noncommutative properties of the outer product, the reordering of vectors in a blade changes only its sign. Then, the reverse can be computed as

$$\widetilde{\mathbf{A}}_{\langle k \rangle} = (-1)^{k(k-1)/2} \mathbf{A}_{\langle k \rangle}. \tag{2.10}$$

The reverse operation is used to define the inverse of a blade as

$$\mathbf{A}_{\langle k \rangle}^{-1} := \frac{\widetilde{\mathbf{A}}_{\langle k \rangle}}{\|\mathbf{A}_{\langle k \rangle}\|^2}, \quad \text{with} \quad \|\mathbf{A}_{\langle k \rangle}\|^2 \neq 0. \tag{2.11}$$

An important element of each algebra is the unit pseudoscalar, denoted by $\mathbf{I}$, which is defined by the unit blade of highest grade. The inverse of the pseudoscalar is used to define the dual of a blade. For a blade $\mathbf{A}$ of grade r, the dual $\mathbf{A}^*$ is defined as follows

$$\mathbf{A}^* := \mathbf{A}\mathbf{I}^{-1}. \tag{2.12}$$

If n is the dimension of the complete space, the grade of the dual blade $\mathbf{A}^*$ will be $n - r$. Thus, the dual of a blade results in a blade complementing the whole space.

An operation that evaluates the intersection of subspaces represented by blades is the "meet". The generalization of this operation for general multivectors is the basis to describe the camera perspective projection process and to define point-line and point-plane incidence equations. The meet operation for two arbitrary blades $\mathbf{A}, \mathbf{B}$, denoted by the operator "$\vee$", is defined as

$$\mathbf{A} \vee \mathbf{B} := (\mathbf{A}\mathbf{J}^{-1} \wedge \mathbf{B}\mathbf{J}^{-1})\mathbf{J}, \tag{2.13}$$

where $\mathbf{J}$ is the result of the "join" operator of the blades.

For two blades $\mathbf{A}$ and $\mathbf{B}$, the join $\mathbf{J} = \mathbf{A} \dot{\wedge} \mathbf{B}$ is defined as the pseudoscalar of the space spanned by the blades $\mathbf{A}$ and $\mathbf{B}$ (see [85] for more details). According to the last definition, the join is an adapted version of the pseudoscalar. Instead of considering the pseudoscalar $\mathbf{I}$ of the entire space, a "reduced" pseudo scalar $\mathbf{J}$ is constructed. For example, the join of the vector $\mathbf{e}_2$ with the bivector $\mathbf{e}_2 \wedge \mathbf{e}_3$ is simply the bivector $\mathbf{e}_2 \wedge \mathbf{e}_3$. It is clear that these elements are defined in the subspace spanned by $\mathbf{e}_2 \wedge \mathbf{e}_3$. Since the vector $\mathbf{e}_1$ does not play any role in this example, it is not considered in the definition of the join for this particular example.

2.2 Euclidean Geometric Algebra

In order to facilitate the reader to follow the interaction of elements in the different spaces, the following notation will be used in next sections. Points in Euclidean space will be represented with the notation $\mathbf{x} \in \mathbb{R}^n$. The embedding of points in projective space with $\mathbf{X} \in \mathbb{R}^{n+1} \backslash \{0\}$, while elements of conformal space will be denoted $\underline{\mathbf{X}} \in \mathbb{R}^{n+1,1} \backslash \{0\}$.

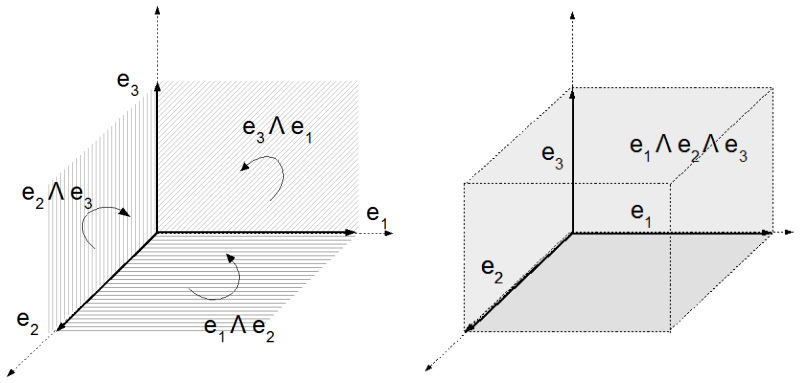

Fig. 2.1: Left: outer product of two vectors expanding an oriented plane segment. Right: outer product of three basis vectors expanding an oriented segment of the complete space.

The Euclidean space, denoted by $\mathbb{R}^3$, is spanned by the orthonormal basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. Its associated geometric algebra $\mathcal{G}_3$ is spanned by the following $2^3 = 8$ basis elements

$$\mathcal{G}_3 = \text{span}\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}, \mathbf{e}_{123}\}. \tag{2.14}$$

As can be seen, the algebra is spanned by one scalar, there vectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, three bivectors $(\mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12})$ and a trivector $(\mathbf{e}_{123})$. This trivector is also known as unit pseudoscalar $(\mathbf{I}_E)$. It squares to -1 and commutes with all other elements of the algebra.

A multivector $\mathbf{M} \in \mathcal{G}_3$ is the combination of all these basis elements:

$$\mathbf{M} = \mathrm{a}_0 + \mathrm{a}_1\mathbf{e}_1 + \mathrm{a}_2\mathbf{e}_2 + \mathrm{a}_3\mathbf{e}_3 + \mathrm{a}_4\mathbf{e}_{23} + \mathrm{a}_5\mathbf{e}_{31} + \mathrm{a}_6\mathbf{e}_{12} + \mathrm{a}_7\mathbf{e}_{123}. \tag{2.15}$$

The geometrical interpretation of these elements can be seen in figure 2.1. The outer product of two vectors defines the area of the parallelogram formed by them. Then, the outer product represents the oriented segment of a plane spanned by these two vectors (the orientation is represented by the arrows in the figure). In a similar way, the outer product of three basis elements represents the oriented unit volume and therefore a segment of the complete space.

Geometrically, the inner and outer products serve as operators that increase or decrease the grade of a given blade or multivector. While the outer product of two blades results in a blade of higher grade, the inner product decreases its grade. By following the definition of equation (2.6), the result of the inner product of the bivector $\mathbf{e}_1 \wedge \mathbf{e}_2$ with the vector $\mathbf{e}_1$ is

$$\mathbf{e}_1 \cdot (\mathbf{e}_1 \wedge \mathbf{e}_2) = (\mathbf{e}_1 \cdot \mathbf{e}_1)\mathbf{e}_2 - (\mathbf{e}_1 \cdot \mathbf{e}_2)\mathbf{e}_1 = \mathbf{e}_2.$$

Since $\mathbf{e}_1 \cdot \mathbf{e}_1 = 1$ and $\mathbf{e}_1 \cdot \mathbf{e}_2 = 0$ are scalars, it is clear that the outer product of a vector with a bivector results in a vector. As can be seen in the example, this operation

can be interpreted as a "subtraction" of the subspace spanned by $\mathbf{e}_1$ from the space spanned by $\mathbf{e}_1 \wedge \mathbf{e}_2$.

It has been shown in [92] that the dual operation of a blade defined in equation (2.12) plays an important roll to construct an algebraic representation of points, lines and planes based on their geometrical representations. This will be described in the next sections. Let us consider one of the examples presented in [85] in order to clarify how the dual works in Euclidean geometric algebra. As described before, the bivector $\mathbf{e}_2 \wedge \mathbf{e}_3$ describes a plane. By the definition of equation (2.12) and according to the inner product rules of equations (2.6) and (2.7), it follows that

$$\begin{aligned}
(\mathbf{e}_2 \wedge \mathbf{e}_3)^* &= (\mathbf{e}_2 \wedge \mathbf{e}_3) \cdot \mathbf{I}_E^{-1} \\
&= (\mathbf{e}_2 \wedge \mathbf{e}_3) \cdot (\mathbf{e}_3 \wedge \mathbf{e}_2 \wedge \mathbf{e}_1) \\
&= \mathbf{e}_2 \cdot [\mathbf{e}_3 \cdot (\mathbf{e}_3 \wedge \mathbf{e}_2 \wedge \mathbf{e}_1)] \\
&= \mathbf{e}_2 \cdot [(\mathbf{e}_3 \cdot \mathbf{e}_3)(\mathbf{e}_2 \wedge \mathbf{e}_1) - (\mathbf{e}_3 \cdot \mathbf{e}_2)(\mathbf{e}_3 \wedge \mathbf{e}_1) + (\mathbf{e}_3 \cdot \mathbf{e}_1)(\mathbf{e}_3 \wedge \mathbf{e}_2)] \\
&= \mathbf{e}_2 \cdot [\mathbf{e}_1 \wedge \mathbf{e}_2] \\
&= \mathbf{e}_1. \qquad (2.16)
\end{aligned}$$

It is clear that the vector $\mathbf{e}_1$ is the dual element of the plane $\mathbf{e}_2 \wedge \mathbf{e}_3$. On the other hand, let us remember how to represent a plane in the Euclidean space $\mathbb{R}^3$. According to the Hessian representation of a plane, it is described by the normal unit vector of the plane and the distance from the origin to the plane. As can be seen in the last example, the result of the dual operation provides the normal vector $\mathbf{e}_1$ which is also used by the Hessian representation. The Hessian representation describes a plane only in $\mathbb{R}^3$. In contrast to that, the bivector or its dual always describe a plane independent of the dimension of the space they are embedded in. This simple example gives a hint of how the dual operation can provide an algebraic element which can be used to represent a geometrical entity. As it will we shown in the next sections, the dual operator is used to find dual representations of points, lines and planes which can be embedded from projective to conformal space and viceversa.

2.2.1 Quaternions

In this section, the ideas presented in [85] to define an isomorphism between quaternions and geometric algebras are summarized. In the next chapter, quaternions will be used as an alternative to model 3D free-form objects as a combination of rotations in the context of geometric algebras.

Quaternions are a non-commutative extension of the complex numbers. They are defined by the combination of a scalar with three imaginary components $\mathbf{i}$, $\mathbf{j}$ and

$\mathbf{k}$ satisfying the following rules

$$\begin{aligned} \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1, \qquad & \mathbf{ij} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{i}, \quad \mathbf{ki} = \mathbf{j} \\ \mathbf{ij} = -\mathbf{ji}, \quad \mathbf{jk} = -\mathbf{kj}, \quad \mathbf{ki} = -\mathbf{ik}, \qquad & \mathbf{ijk} = -1. \end{aligned} \tag{2.17}$$

A general quaternion $\mathbf{q}$ has the form $\mathbf{q} = \mathrm{q}_0 + \mathrm{q}_1\mathbf{i} + \mathrm{q}_2\mathbf{j} + \mathrm{q}_3\mathbf{k}$ with $\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3 \in \mathbb{R}$. A quaternion which does not have a scalar component is called a pure quaternion $\mathbf{p} = \mathrm{q}_1\mathbf{i} + \mathrm{q}_2\mathbf{j} + \mathrm{q}_3\mathbf{k}$. The complex conjugate of a quaternion $\mathbf{q}$ is denoted by $\mathbf{q}^* = \mathrm{q}_0 - \mathrm{q}_1\mathbf{i} - \mathrm{q}_2\mathbf{j} - \mathrm{q}_3\mathbf{k}$. Similar to the complex numbers, a quaternion can be also expressed in exponential form as

$$\mathbf{q} \;=\; r\exp(\theta\hat{\mathbf{p}}), \tag{2.18}$$

where $r = \sqrt{\mathbf{q}\mathbf{q}^*}$, $\theta = \mathrm{atan}(\frac{\sqrt{\mathbf{p}\mathbf{p}^*}}{\mathrm{q}_0})$ and $\hat{\mathbf{p}} = \frac{\mathbf{p}}{\sqrt{\mathbf{p}\mathbf{p}^*}}$. The exponential representation of a unit quaternion can be interpreted as a rotation operation. A vector represented by the pure quaternion $\mathbf{a}$ is rotated by the following operation

$$\mathbf{b} \;=\; \mathbf{r}\quad\mathbf{a}\quad\mathbf{r}^*, \tag{2.19}$$

with $\mathbf{r} = \exp\left(\frac{1}{2}\theta\hat{\mathbf{p}}\right)$ and $\hat{\mathbf{p}}$ the unit quaternion representing the rotation vector. Then, the vector $\mathbf{a}$ is rotated by an angle θ around the vector represented by $\hat{\mathbf{p}}$.

Since quaternions are formed by a combination of a scalar component and three imaginary components, an isomorphism can be defined between quaternions $\mathbb{H}$ and geometric algebra. The key idea of this isomorphism is that bivectors of $\mathcal{G}_3$ have the same properties as the imaginary quaternion units $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$. Thus, the imaginary units can be identified with the following bivectors:

$$\mathbf{i} \to \mathbf{e}_{23}, \quad \mathbf{j} \to \mathbf{e}_{12}, \quad \mathbf{k} \to \mathbf{e}_{31}. \tag{2.20}$$

Then, quaternions can be considered as elements of the subalgebra $\mathcal{G}_3^+ \subset \mathcal{G}_3$ spanned by the basis vectors:

$$\mathcal{G}_3^+ = \mathrm{span}\{1, \mathbf{e}_{23}, \mathbf{e}_{12}, \mathbf{e}_{31}\}. \tag{2.21}$$

Since the subalgebra $\mathcal{G}_3^+$ is isomorphic to the quaternion algebra $\mathbb{H}$, all properties of equation (2.17) are valid for the bivectors defining $\mathcal{G}_3^+$. By using the geometric product, it ha been shown in [85] that the known relations between the imaginary units are preserved,

$$\begin{aligned} \mathbf{ij} \to (\mathbf{e}_2\mathbf{e}_3)(\mathbf{e}_1\mathbf{e}_2) = (\mathbf{e}_3\mathbf{e}_1) \to \mathbf{k} \quad &, \quad (\mathbf{e}_3\mathbf{e}_1)^2 = -1 \\ \mathbf{jk} \to (\mathbf{e}_1\mathbf{e}_2)(\mathbf{e}_3\mathbf{e}_1) = (\mathbf{e}_2\mathbf{e}_3) \to \mathbf{i} \quad &, \quad (\mathbf{e}_2\mathbf{e}_3)^2 = -1 \\ \mathbf{ki} \to (\mathbf{e}_3\mathbf{e}_1)(\mathbf{e}_2\mathbf{e}_3) = (\mathbf{e}_1\mathbf{e}_2) \to \mathbf{j} \quad &, \quad (\mathbf{e}_1\mathbf{e}_2)^2 = -1. \end{aligned} \tag{2.22}$$

Let us consider the rotation operation of equation (2.19) with the unit quaternion $\hat{\mathbf{p}} = x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$ defining the rotation axis. According to the basis identification of equation (2.20), this unit quaternion can be directly related to the unit bivector $\mathbf{n} = x_1\mathbf{e}_{23} + x_2\mathbf{e}_{12} + x_3\mathbf{e}_{31} \in \mathcal{G}_3^+$. Then, the analog rotation operation called "rotor" of equation (2.19) can be written as

$$\mathbf{R} = \exp\left(-\frac{1}{2}\theta\mathbf{n}\right) \in \mathcal{G}_3^+. \tag{2.23}$$

Notice that $\hat{\mathbf{p}}$ represents a rotation vector in $\mathbb{H}$ while $\mathbf{n}$ represents a plane in $\mathcal{G}_3^+$. That means, the rotation is performed around a rotation axis in $\mathbb{H}$ while the same rotation is performed with respect to a plane in $\mathcal{G}_3^+$.[2] Although $\mathbf{n}$ and $\hat{\mathbf{p}}$ are different geometrical entities defined in different algebras, there is a relation between their parameters. To make this idea clear, the dual operation is applied to the bivector $\mathbf{n}$. Similar to the example of equation (2.16), the dual of each basis element of $\mathbf{n}$ are

$$(\mathbf{e}_2\mathbf{e}_3)^* = \mathbf{e}_1, \quad (\mathbf{e}_1\mathbf{e}_2)^* = \mathbf{e}_3, \quad (\mathbf{e}_3\mathbf{e}_1)^* = \mathbf{e}_2.$$

Therefore, the dual $\mathbf{n}^* = x_1\mathbf{e}_1 + x_2\mathbf{e}_3 + x_3\mathbf{e}_2$ corresponds to the rotation vector represented by the unit quaternion $x_1\mathbf{i}+x_2\mathbf{j}+x_3\mathbf{k}$ according to the identification of equation (2.20). If quaternions are used to represent vectors in Euclidean space, each imaginary unit $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ is identified with the three x, y and z axes respectively. In $\mathcal{G}_3$ these axes are denoted by $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$. Notice that the axes $\mathbf{e}_2$ and $\mathbf{e}_3$ of $\mathbf{n}^*$ are changed. Then, the corresponding change of axes produces the change of sign in the rotation of equation (2.23).

One advantage of using rotors over quaternions is that rotors can be defined in any dimension. While rotations can be only applied to vectors in quaternion algebra $\mathbb{H}$, rotations in $\mathcal{G}_3^+$ can be applied to blades of any grade e.g. lines and planes.

2.3 Geometric Algebra of Projective Space

Essential for the pose estimation problem is the representation of basic geometric entities like points, lines, planes as well as their interaction with the image formation process. This is not possible in Euclidean geometric algebra $\mathcal{G}_3$. Therefore, the use of projective geometry is required.

The projective space $\mathbb{PR}^n$ (with dimension $\mathbb{R}^{n+1}\backslash\{0\}$ excluding the origin) is defined by all elements $\mathbf{X}$ and $\mathbf{Y}$, which form a set of equivalence classes $\sim$ such that

$$\forall \mathbf{X}, \mathbf{Y} \in \mathbb{PR}^n : \mathbf{X} \sim \mathbf{Y} \Leftrightarrow \exists \lambda \in \mathbb{R}\backslash\{0\} : \mathbf{X} = \lambda\mathbf{Y}. \tag{2.24}$$

[2] This fact is the principle for the construction of rotations in geometric algebra that will be explained in detail in the next sections.

The projective space is illustrated in figure 2.2 for $\mathbb{R}^2$. All the points of the line defined by $\mathbf{X}$ and the origin represent the same point in the projective plane. Thus, the entire equivalence class is collapsed into a single object, for this example in a single point. The basis elements of projective space are the orthonormal vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_n, \mathbf{e}_{n+1}\}$, where the basis vector $\mathbf{e}_{n+1}$ is known as homogeneous component.

The operators P and P^{-1} are defined in [85], which denote the transformations of an element from Euclidean ($\mathbf{x} \in \mathbb{R}^n$) to projective space ($\mathbf{X} \in \mathbb{PR}^n$) and vice versa as

$$P(\mathbf{x}) = \mathbf{x} + \mathbf{e}_{n+1} \in \mathbb{PR}^n \tag{2.25}$$

$$P^{-1}(\mathbf{X}) = \frac{1}{\mathbf{X} \cdot \mathbf{e}_{n+1}} \sum_{i=1}^{n} (\mathbf{X} \cdot \mathbf{e}_i)\mathbf{e}_i \in \mathbb{R}^n. \tag{2.26}$$

As can be seen in the last equation, the homogeneous component of X is always the unity. These transformations are illustrated in figure 2.2.

Then, the Euclidean space $\mathbb{R}^3$ is embedded in projective space $\mathbb{PR}^3$ by adding a homogeneous component denoted by $\mathbf{e}_-$, with the property that $\mathbf{e}_-^2 = -1$. The corresponding geometric algebra of projective space $\mathcal{G}_{3,1}$ is spanned by the following $2^4 = 16$ elements

$$\begin{aligned} \mathcal{G}_{3,1} = \; & \text{span}\{1, \mathbf{e}_-, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{12}, \mathbf{e}_{-1}, \mathbf{e}_{-2}, \mathbf{e}_{-3}, \\ & \mathbf{e}_{123}, \mathbf{e}_{-23}, \mathbf{e}_{-31}, \mathbf{e}_{-12}, \mathbf{e}_{-123}\}, \end{aligned} \tag{2.27}$$

with the pseudoscalar $\mathbf{I}_P := \mathbf{e}_{-123}$ with the property that with $\mathbf{e}_{-123}^2 = -1$.

The outer product is important in the definition of the projective geometric algebra. It has been discussed in [87] that the outer product is used to define the equivalence class of equation (2.24). Since $\mathbf{X} \wedge \mathbf{X} = 0$, it is clear that $\mathbf{X} \wedge \lambda\mathbf{X} = 0 \forall \lambda \in \mathbb{R} \backslash \{0\}$. Hence, all vectors $\mathbf{X}$ represent a point $\mathbf{A}$ if $\mathbf{A} \wedge \mathbf{X} = 0$.

2.3.1 Points, Lines and Planes in Projective Geometric Algebra

The idea of representing a point in projective space is shown in the left graphic of figure 2.2 for $\mathbb{R}^2$. Because of the homogeneous representation of $\mathbf{X} \in \mathbb{PR}^2$, all points along this vector excluding the origin represent the same point $\mathbf{x} \in \mathbb{R}^2$. A point $\mathbf{x} = \mathrm{x}_1\mathbf{e}_1 + \mathrm{x}_2\mathbf{e}_2 + \mathrm{x}_3\mathbf{e}_3 \in \mathcal{G}_3$ is represented in projective geometric algebra $\mathcal{G}_{3,1}$ just by adding the homogeneous coordinate $\mathbf{e}_-$ as

$$\mathbf{X} = \mathbf{x} + \mathbf{e}_- = \mathrm{x}_1\mathbf{e}_1 + \mathrm{x}_2\mathbf{e}_2 + \mathrm{x}_3\mathbf{e}_3 + \mathbf{e}_-. \tag{2.28}$$

The right graphic of figure 2.2 shows the idea of representing lines. Let us consider two points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^2$ and their respective homogeneous vectors $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{PR}^2$.

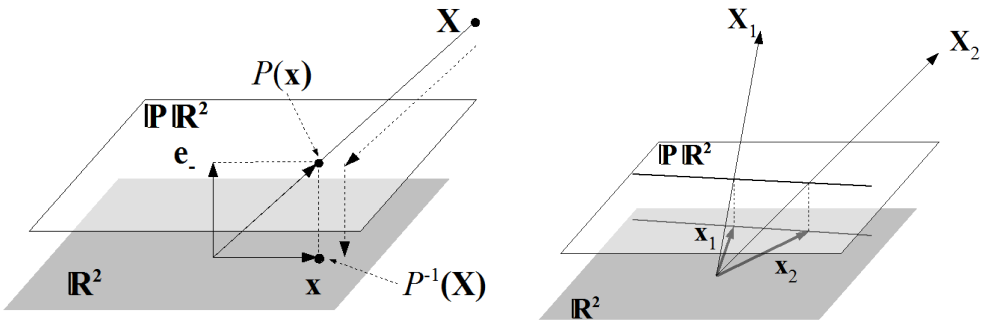

Fig. 2.2: Left: example of the embedding of an Euclidean vector in projective space and the opposite projection of a homogeneous vector to Euclidean space. Right: representation of a line in projective space by two homogeneous vectors.

Then, a plane is spanned by these homogeneous vectors (see [85]). The points that result from the intersection of this plane with the projective plane define a line in $\mathbb{PR}^2$. As can be seen in the figure, this line does not pass through the origin. Finally, the orthographic projection of the intersection gives a line in $\mathbb{R}^2$.

In order to define lines and planes in projective geometric algebra, the outer product of points is used as defined [92]. Let us consider the case of a line $\mathbf{L} \in \mathcal{G}_{3,1}$. It is obtained by the outer product of two points $\mathbf{X}_1, \mathbf{X}_2$. This operation in the algebra to obtain a line is summarized as described in [92] as follows

$$\begin{aligned} \mathbf{L} &= \mathbf{X}_1 \wedge \mathbf{X}_2 \\ &= (\mathbf{x}_1 + \mathbf{e}_-) \wedge (\mathbf{x}_2 + \mathbf{e}_-) \\ &= \mathbf{x}_1 \wedge \mathbf{x}_2 + (\mathbf{x}_1 - \mathbf{x}_2)\mathbf{e}_- \\ &= \mathbf{m} - \mathbf{r}\mathbf{e}_-, \end{aligned} \tag{2.29}$$

where the parameters $\mathbf{m}$ (moment) and $\mathbf{r}$ (direction) are same parameters used by the Plücker representation of a line [15].

Similarly, a plane in projective space is represented by the outer product of three points $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3 \in \mathcal{G}_{3,1}$. This operation is summarized as presented in [92] as follows

$$\begin{aligned} \mathbf{P} &= \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \\ &= (\mathbf{x}_1 + \mathbf{e}_-) \wedge (\mathbf{x}_2 + \mathbf{e}_-) \wedge (\mathbf{x}_3 + \mathbf{e}_-) \\ &= \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 + (\mathbf{x}_1 - \mathbf{x}_2) \wedge (\mathbf{x}_1 - \mathbf{x}_3)\mathbf{e}_- \\ &= d\mathbf{I}_E + \mathbf{n}\mathbf{e}_-. \end{aligned} \tag{2.30}$$

with the pseudoscalar of Euclidean space $\mathbf{I}_E = \mathbf{e}_{123}$.

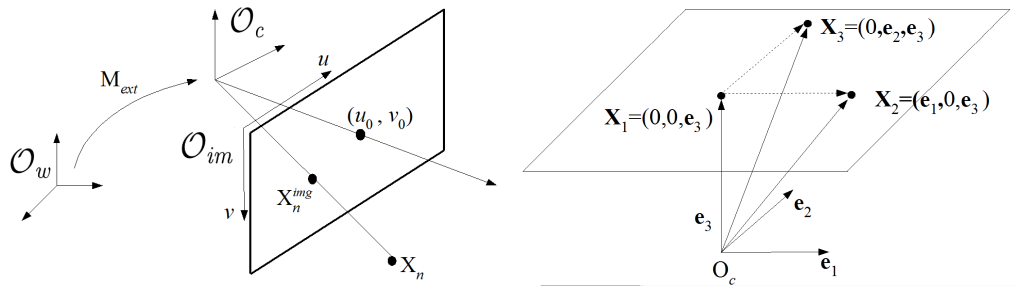

Fig. 2.3: Left: coordinate systems involved in the complete camera model. Right: definition of the image plane according to the normalized camera model.

In this case, the parameters $\mathbf{n}$ (normal of the plane) and $d \in \mathbb{R}$ (distance to the origin) are the parameters used in the Hesse representation of planes. In comparison with the elements of the algebra of Euclidean space $\mathcal{G}_3$, algebraic linear representations are gained from the basic geometrical concept of construction of lines and planes: two points define a line and three points a plane.

2.3.2 Camera Model in Projective Geometric Algebra

The pinhole camera model defines the perspective projection of spatial points onto the image plane. More detailed information about the pinhole camera model can be found in [37]. For a given number of world points and their corresponding image points, the calibration matrix defining the projection is computed. Additionally, it can be can decomposed in a set of external and internal parameters (known as complete camera model, see [45]). As shown in figure 2.3, this set of parameters defines the relation between the different coordinate systems involved in the projection process: world $\mathcal{O}_w$, camera $\mathcal{O}_c$ and image $\mathcal{O}_{im}$. In the next equations, the corresponding origin points of these coordinate systems are denoted as $\mathbf{O}_w$, $\mathbf{O}_c$ and $\mathbf{O}_{im}$ respectively. The external parameters are encoded in the matrix $\mathbf{M}_{ext}$ which defines the rigid body motion needed to represent a point in the camera coordinate system. From this matrix, the location of the optical center of the camera $\mathbf{O}_c$ can be obtained. The internal parameters are defined by the scale factors fk_u, fk_v in the u and v image axes directions and the coordinates of the principal point in the image (u_0, v_0). The principal point is the point where the optical axis intersects the image plane (note that this point may not necessary be the center of the image). Based on these parameters, a point $\mathbf{X}_n \in \mathbb{PR}^3$ is projected onto the image plane in two steps. First, it is transformed to the camera coordinate system $\mathbf{X}'_n = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 + \mathbf{e}_-$ by the rigid body motion $\mathbf{M}_{ext}$. Finally, it is projected onto the image plane by the internal parameters to the point

$$\mathbf{X}_{\mathrm{n}}^{img} = \left(\frac{fk_u x_1 + u_0}{x_3}\right)\mathbf{e}_1 + \left(\frac{fk_u x_2 + v_0}{x_3}\right)\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_- \in \mathcal{G}_{3,1}. \tag{2.31}$$

The camera model is constructed and normalized in such a way, that the image center intersects the optical axis at the point with coordinates $(0, 0, 1)$ in the camera coordinate system $\mathcal{O}_c$ (notice that its origin point corresponds to $(0, 0, \mathbf{e}_3)$). Since the u and v image axes are parallel to the basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ respectively, the image plane is spanned in projective space by the points

$$\begin{aligned} \mathbf{X}_1 &= \alpha\mathbf{e}_3 + \mathbf{e}_- && \in \mathcal{G}_{3,1} \\ \mathbf{X}_2 &= \alpha(\mathbf{e}_1 + \mathbf{e}_3) + \mathbf{e}_- && \in \mathcal{G}_{3,1} \\ \mathbf{X}_3 &= \alpha(\mathbf{e}_2 + \mathbf{e}_3) + \mathbf{e}_- && \in \mathcal{G}_{3,1}, \end{aligned} \tag{2.32}$$

with the factor $\alpha \geq 1 \in \mathbb{R}$. Notice that the exact position of the image plane can be found since the internal and external parameters are known.

It is possible to define the perspective projection of equation (2.31) as the result of intersection of the image plane and a line defined by the point and the optical center. Then, the image plane is defined according to equation (2.30) by the outer product of the points $\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$ as

$$\begin{aligned} \mathbf{P}_{im} &= \mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3 \\ &= \mathbf{e}_3 \wedge (\mathbf{e}_1 + \mathbf{e}_3) \wedge (\mathbf{e}_2 + \mathbf{e}_3) + (\mathbf{e}_3 - (\mathbf{e}_1 + \mathbf{e}_3)) \wedge (\mathbf{e}_3 - (\mathbf{e}_2 + \mathbf{e}_3)) \wedge \mathbf{e}_- \\ &= -\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3 + (\mathbf{e}_1 \wedge \mathbf{e}_2)\mathbf{e}_- \\ &= -\mathbf{I}_E + (\mathbf{e}_1 \wedge \mathbf{e}_2)\mathbf{e}_-. \end{aligned} \tag{2.33}$$

Since the optical center is considered as the origin of the coordinate system, it has the form $\mathbf{O}_c = 0 + \mathbf{e}_- \in \mathcal{G}_{3,1}$. The line defined from the optical center and the point $\mathbf{X}_n = \mathbf{x}_n + \mathbf{e}_- \in \mathcal{G}_{3,1}$ is computed according to equation (2.29) by

$$\begin{aligned} \mathbf{L}_n &= \mathbf{x}_\mathrm{n} \wedge 0 + (\mathbf{x}_\mathrm{n} - 0)\mathbf{e}_- \\ &= \mathbf{x}_\mathrm{n}\mathbf{e}_-. \end{aligned} \tag{2.34}$$

According to equation (2.13), the intersection of this line and the image plane is computed by the meet operation "$\vee$"

$$\begin{aligned} \mathbf{X}_n^{img} &= \mathbf{L}_n \vee \mathbf{P}_{im} \\ &= (\mathbf{O}_c \wedge \mathbf{X}_n) \vee (\mathbf{X}_1 \wedge \mathbf{X}_2 \wedge \mathbf{X}_3) \\ &= (-\mathbf{I}_E + (\mathbf{e}_1 \wedge \mathbf{e}_2)\mathbf{e}_-) \vee (\mathbf{x}_\mathrm{n}\mathbf{e}_-) \\ &= p_1\mathbf{e}_1 + p_2\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_-. \end{aligned} \tag{2.35}$$

To find its corresponding pixel coordinates in the image, the internal camera parameters are used as it was shown in equation (2.31) to scale the point coordinates as follows

$$\mathbf{X}_n^{img} = (fk_u p_1 + u_0)\mathbf{e}_1 + (fk_v p_2 + v_0)\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_-. \tag{2.36}$$

Important for the pose estimation problem is to recover information in $\mathbb{R}^3$ from information of the image plane. The basic entity that can be recovered is an optical ray. According to equation (2.29), a line can be represented as the outer product of two points in projective space. If $\mathbf{X} \in \mathcal{G}_{3,1}$ is a point in projective space, a line throw these points can be calculated as the outer product $\mathbf{L} = \mathbf{O}_c \wedge \mathbf{X}$. With the scale factors fk_u, fk_v and the coordinates of the image center (u_0, v_0), the coordinates of an image point (u, v) can be re-scaled to get the coordinates of that point in projective space as follows

$$\mathbf{X} = \frac{u - u_0}{fk_u}\mathbf{e}_1 + \frac{v - v_0}{fk_v}\mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_- \in \mathcal{G}_{3,1}. \tag{2.37}$$

Then, the optical ray may be directly computed with equation (2.29). Similarly, a plane can be defined with the optical center and the image points $\mathbf{x}_1 = (u_1, v_2)$ and $\mathbf{x}_2 = (u_2, v_2)$. Once that these points have been re-scaled to $\mathbf{X}_1 \in \mathcal{G}_{3,1}$ and $\mathbf{X}_2 \in \mathcal{G}_{3,1}$, the plane is computed by

$$\mathbf{P} = \mathbf{O}_c \wedge \mathbf{X}_1 \wedge \mathbf{X}_2 \in \mathcal{G}_{3,1}. \tag{2.38}$$

2.4 Conformal Geometric Algebra

Several spaces and its associated algebras have been discussed in the last sections. It has been shown that the projective geometric algebra is useful for the construction of basic geometrical entities (points, lines and planes) and to define the image formation process. In order to formulate the pose estimation problem, it is necessary to relate rigid body motions with these entities from an algebraic point of view. Although it is possible to represent rotations as linear operations in Euclidean space, translations do not have this property. This may be a problem in order to define a rigid body motion as linear operation in the algebra.[3] In order to get a linear representation of combined rotations and translations, the dual quaternion algebra [8] has been used. Despite of that, the duality concept needed to define projective geometry can not be applied in this algebra. To overcome these problems, it is necessary to perform a further geometric embedding. In this section, the embedding process which is presented in [87] and [85] is introduced.

The Euclidean space is embedded in a higher dimensional space (conformal space). The resulting conformal geometric algebra (CGA) is a general algebra where the Euclidean and projective geometric algebras are subsets of it. Additionally, it has the property of containing an artificially generated null space (constructed from a Minkowski space [62]). Because of that, it is possible to change between the elements of the null space and the non-null space. As it will be shown later, this property is important because it allows to change the representation of elements from projective to conformal algebra and vice versa.

[3] Let be $\mathbf{x}$ and $\mathbf{y}$ two vectors of $\mathcal{G}_3$. A rotation $\mathbf{R}$ follows the linearity property $\mathbf{R}(\mathbf{x}+\mathbf{y}) = \mathbf{R}(\mathbf{x}) + \mathbf{R}(\mathbf{y})$. The translation $\mathbf{T}(\mathbf{x}+\mathbf{y}) \neq \mathbf{T}(\mathbf{x}) + \mathbf{T}(\mathbf{y})$ is not linear. Therefore, their combination is not linear.

The geometrical embedding of Euclidean space into conformal space is done by using a special kind of projection called stereographic projection (commonly used to generate maps). It is called conformal since the projection is done by preserving the angles between projected curves. The main idea is illustrated in figure 2.4 for the projection of a one dimensional point, where the vectors $\{\mathbf{e}_1, \mathbf{e}_+\}$ define an orthonormal basis. Let us imagine that a ray of light coming from the reference point

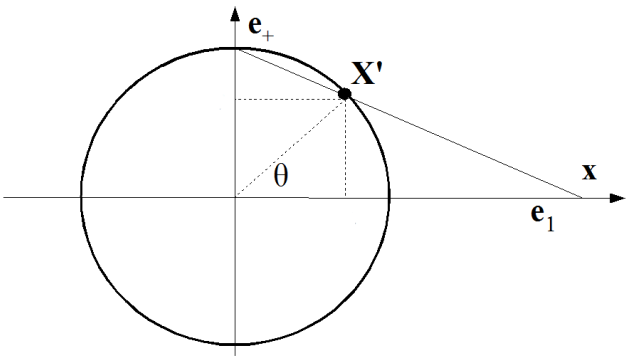

Fig. 2.4: Example of a stereographic projection of a 1D point.

$\mathbf{e}_+$ incides with the point on the unit sphere $\mathbf{X}'$. Then, its shadow is projected onto the $\mathbf{e}_1$ axis at the point $\mathbf{x} \in \mathbb{R}^1$. As presented in [87], a point $\mathbf{X}' = \cos(\theta)\mathbf{e}_1 + \sin(\theta)\mathbf{e}_+$ is projected onto the axis $\mathbf{e}_1$ as

$$\mathbf{X}' \Rightarrow \mathbf{x} = \left(\frac{\cos(\theta)}{1 - \sin(\theta)}\right)\mathbf{e}_1 + 0\mathbf{e}_+. \tag{2.39}$$

The opposite projection of an Euclidean point $x\mathbf{e}_1$ onto the unit circle (see [87]) is performed by the following operation

$$\mathbf{x} \Rightarrow \mathbf{X}' \;=\; \frac{2x}{x^2+1}\mathbf{e}_1 + \frac{x^2-1}{x^2+1}\mathbf{e}_+. \tag{2.40}$$

Let us notice that the extra basis vector $\mathbf{e}_+$ with properties that $\mathbf{e}_+^2 = \mathbf{e}_1^2 = 1$ has been added as an additional basis vector to the one-dimensional Euclidean space. A point projected on the point $\mathbf{e}_+$ represents the point at infinity. If the point is projected on $-\mathbf{e}_+$, it represents the origin.

In general, the conformal space is denoted by $\mathbb{K}^n$ and represents the space $\mathbb{R}^{n+1}$. Similar to the homogenization of the Euclidean space, the conformal space is also homogenized by adding the component $\mathbf{e}_-$ with negative signature $\mathbf{e}_-^2 = -1$. That means, $\mathbb{K}^n$ is embedded in projective space $\mathcal{P}\mathbb{K}^n$, which is represented by the space $\mathbb{R}^{n+1,1}\backslash\{0\}$. Then, a point $\mathbf{X}' = \mathrm{a}\mathbf{e}_1 + \mathrm{b}\mathbf{e}_+$ is embedded as

$$\mathcal{P}(\mathbf{X}') \;=\; \mathrm{a}\mathbf{e}_1 + \mathrm{b}\mathbf{e}_+ + \mathbf{e}_-. \tag{2.41}$$

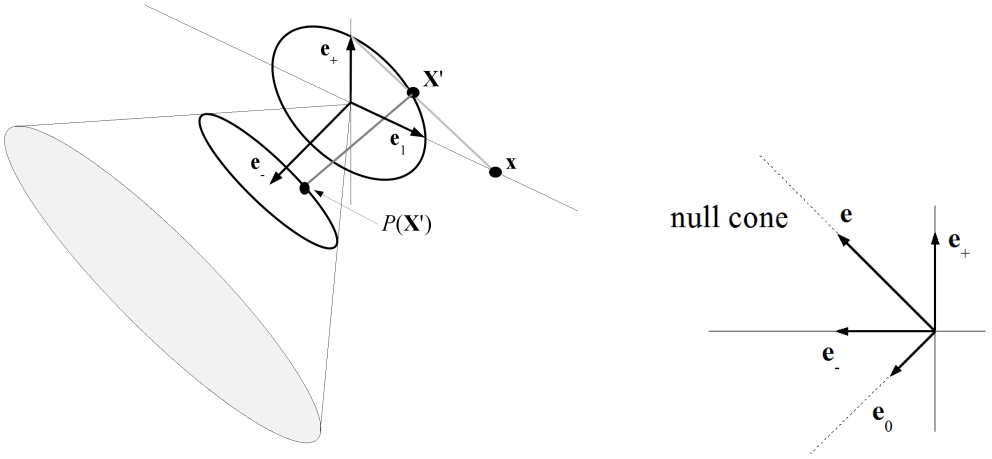

Fig. 2.5: Example of the homogenization of the conformal space by adding the component $\mathbf{e}_-$ (left). Additional null basis of the null cone (right).

This is illustrated in figure 2.5. As can be seen in the figure, the embedded point lies on a unit circle. A direct consequence that follows from the addition of a homogeneous component is that the embedded points square to zero ($(\mathbf{X}')^2 = 0$). The mathematical prove of the last property can be consulted in [85]. Then, all vectors in $\mathcal{P}\mathbb{K}^n$ that result from the embedding of a vector in Euclidean space square to zero and they are called null vectors. For the example presented in this section, the set of points in $\mathcal{P}\mathbb{K}^1$ which satisfy this condition lie on a null cone as it is can be seen in figure 2.5. That means, only the vectors of the null cone can be projected back to the Euclidean space and only this vectors have a geometrical meaning back in Euclidean space.

2.4.1 Definition of the Conformal Geometric Algebra

Once that the concept of orthographic protection has been briefly introduced, the basis elements of conformal geometric algebra and their main properties are introduced in this section. The conformal vector space derived from $\mathbb{R}^3$ is denoted as $\mathbb{R}^{4,1}$. In addition to the basis elements of Euclidean space, the basis elements described in last section are considered in the algebra. Then, the basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_+, \mathbf{e}_-\}$ span the conformal space $\mathbb{R}^{4,1}$. Therefore, the corresponding algebra denoted by $\mathcal{G}_{4,1}$ contains $2^5 = 32$ elements.

It has been mentioned before that only the elements of the null cone can be projected back to Euclidean space. Therefore, two additional basis vectors associated

with the null cone (see figure 2.5) are defined as

$$\begin{aligned} \mathbf{e}_0 : &= \tfrac{1}{2}(\mathbf{e}_- - \mathbf{e}_+) \\ \mathbf{e} : &= (\mathbf{e}_- + \mathbf{e}_+). \end{aligned} \tag{2.42}$$

Similar to the algebras of Euclidean and Projective spaces, the conformal algebra has its pseudoscalar element defined by $\mathbf{I}_C = \mathbf{e}_{+-123} = \mathbf{E}\mathbf{I}_E$. In this case $\mathbf{I}_E = \mathbf{e}_{123}$ denotes the pseudoscalar of $\mathcal{G}_3$ and $\mathbf{E} := \mathbf{e} \wedge \mathbf{e}_0 = \mathbf{e}_+ \wedge \mathbf{e}_-$.

According to the stereographic projection model and after a proper homogenization and scaling of the elements (see [87]), points in Euclidean space $\mathbf{x} \in \mathbb{R}^3$ are embedded into the null cone as

$$\underline{\mathbf{X}} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathbf{e} + \mathbf{e}_0. \tag{2.43}$$

The last representation of a point in will be used in this and the next chapters. Finally, the main mathematical properties of the basis elements in the algebra are summarized as follows

$$\begin{array}{ccc} \mathbf{e}_0^2 = \mathbf{e}^2 = 0 & \mathbf{e} \cdot \mathbf{e}_0 = -1 & \mathbf{E} = \mathbf{e}_+\mathbf{e}_- \\ \mathbf{E}\mathbf{e} = -\mathbf{e} & \mathbf{E}\mathbf{e}_0 = \mathbf{e}_0 & \mathbf{E}^2 = 1 \\ \mathbf{e}_+\mathbf{E} = \mathbf{e}_- & \mathbf{e}_-\mathbf{E} = \mathbf{e}_+ & \mathbf{e}_+\mathbf{e} = \mathbf{E} + 1 \\ \mathbf{e}_-\mathbf{e} = -(\mathbf{E}+1) & \mathbf{e} \wedge \mathbf{e}_- = \mathbf{E} & \mathbf{e}_+ \cdot \mathbf{e} = 1. \end{array} \tag{2.44}$$

2.4.2 Geometric Entities in Conformal Geometric Algebra

It is clear that points are the are the basic geometric entities which can be described in Euclidean space. In contrast to that, spheres are the basic entities of the conformal geometric algebra $\mathcal{G}_{4,1}$. In fact, a point in CGA is a degenerate sphere. A detailed introduction about the derivation of basic geometric entities in CGA can be consulted in [36, 85, 102]. In this section, the basic principle of constructing spheres, points, lines and planes in CGA is presented.

In this and the next sections, the notation $\underline{\mathbf{X}}$ will be used to denote elements of $\mathcal{G}_{4,1}$. One of the advantages of using conformal geometric algebras is their ability to represent geometric entities in a linear and compact way. Let us start with the case of a sphere. Spheres can not be represented in a compact and linear way in Euclidean space[4]. In contrast to that, it has been shown in [87] that a sphere $\underline{\mathbf{S}} \in \mathcal{G}_{4,1}$ can be represented in CGA as

$$\underline{\mathbf{S}} = \mathbf{p} + \frac{1}{2}(\mathbf{p} - p^2)\mathbf{e} + \mathbf{e}_0, \tag{2.45}$$

[4] In fact, a sphere in $\mathbb{R}^3$ can be only characterized by the constraint equation $(\mathbf{x} - \mathbf{p})^2 - p^2 = 0$, where $\mathbf{x} \in \mathcal{G}_3$ is a point on the sphere and $\mathbf{p} \in \mathcal{G}_3$ is the center of the sphere.

where $\mathbf{p} \in \mathcal{G}_3$ is the center of the sphere and $p \in \mathbb{R}, p \geq 0$ is its radius.

If the radius $p = 0$, the equation is reduced to the point representation of equation (2.43). That means, a point can be seen as a degenerate sphere. In a similar way, a plane can eventually be regarded as a sphere with infinite radius. The intersection of two spheres define a circle, while the intersection of three spheres results in a point pair.

As it was shown in the examples of sections 2.2 and 2.2.1, dual representations of certain entities were found which represented interesting geometrical properties. A similar idea is used in [87] to define dual representations of the entities derived from spheres in CGA. If $\underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}, \underline{\mathbf{D}} \in \mathcal{G}_{4,1}$ are four points on a sphere, its dual is defined by the outer product of these points as

$$\underline{\mathbf{S}}^* = \underline{\mathbf{A}} \wedge \underline{\mathbf{B}} \wedge \underline{\mathbf{C}} \wedge \underline{\mathbf{D}}. \tag{2.46}$$

Similarly, the dual representation of a circle $\underline{\mathbf{Z}}^* \in \mathcal{G}_{4,1}$ is defined by the outer product of three points

$$\underline{\mathbf{Z}}^* = \underline{\mathbf{A}} \wedge \underline{\mathbf{B}} \wedge \underline{\mathbf{C}}. \tag{2.47}$$

As can be seen in the last equations, the dual is used to represent these entities based on points contained on them. This property is also valid to get dual representations of points, lines and planes useful for the formulation of the pose estimation problem. Initially, the dual representation of a point $\underline{\mathbf{X}} \in \mathcal{G}_{4,1}$ is constructed with the point at infinity $\mathbf{e}$ (see [36, 102]) as

$$\begin{aligned}
\underline{\mathbf{X}}^* &= \mathbf{e} \wedge \underline{\mathbf{X}} \\
&= \mathbf{e} \wedge (\mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathbf{e} + \mathbf{e}_0) \\
&= \mathbf{e} \wedge \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathbf{e} \wedge \mathbf{e} + \mathbf{e} \wedge \mathbf{e}_0 \\
&= \mathbf{e} \wedge \mathbf{x} + \mathbf{E} = \mathbf{e}\mathbf{x} + \mathbf{E}.
\end{aligned} \tag{2.48}$$

A line can be interpreted as a circle passing through the point at infinity $\mathbf{e}$. By using the dual representation of a circle of equation (2.47), the dual of a line is computed with the points $\underline{\mathbf{X}}_1, \underline{\mathbf{X}}_2 \in \mathcal{G}_{4,1}$ as

$$\begin{aligned}
\underline{\mathbf{L}}^* &= \mathbf{e} \wedge \underline{\mathbf{X}}_1 \wedge \underline{\mathbf{X}}_2 \\
&= (\mathbf{e} \wedge \underline{\mathbf{X}}_1) \wedge (\mathbf{x}_2 + \frac{1}{2}\mathbf{x}_2^2\mathbf{e} + \mathbf{e}_0) \\
&= (\mathbf{e} \wedge \underline{\mathbf{X}}_1) \wedge \mathbf{x}_2 + (\mathbf{e} \wedge \underline{\mathbf{X}}_1) \wedge \frac{1}{2}\mathbf{x}_2^2\mathbf{e} + (\mathbf{e} \wedge \underline{\mathbf{X}}_1) \wedge \mathbf{e}_0 \\
&= (\mathbf{e} \wedge \mathbf{x}_1 + \mathbf{E}) \wedge \mathbf{x}_2 + (\mathbf{e} \wedge \mathbf{x}_1 + \mathbf{E}) \wedge \mathbf{e}_0 \\
&= \mathbf{e} \wedge \mathbf{x}_1 \wedge \mathbf{x}_2 + \mathbf{E} \wedge \mathbf{x}_2 - \mathbf{E} \wedge \mathbf{x}_1 \\
&= \mathbf{e} \wedge \mathbf{x}_1 \wedge \mathbf{x}_2 + (\mathbf{x}_2 - \mathbf{x}_1) \wedge \mathbf{E} \\
&= \mathbf{e}\mathbf{m} + \mathbf{r}\mathbf{E},
\end{aligned} \tag{2.49}$$

where the vectors $\mathbf{m}$ and $\mathbf{r}$ are the moment and direction of the Plücker representation of lines. This is a similar representation to the representation of lines in projective space of equation (2.29).

Analog to lines, planes can be defined from the dual representation of spheres by the outer product of three points and the point at infinity $\mathbf{e}$. By following a similar procedure as in equation (2.49), the dual of a plane is defined with de points $\underline{\mathbf{X}}_1, \underline{\mathbf{X}}_2 . \underline{\mathbf{X}}_3 \in \mathcal{G}_{4,1}$ by

$$\begin{aligned} \underline{\mathbf{P}}^* &= \mathbf{e} \wedge \underline{\mathbf{X}}_1 \wedge \underline{\mathbf{X}}_2 \wedge \underline{\mathbf{X}}_3 \\ &= (\mathbf{e} \wedge \underline{\mathbf{X}}_1 \wedge \underline{\mathbf{X}}_2) \wedge (\mathbf{x}_3 + \frac{1}{2}\mathbf{x}_3^2\mathbf{e} + \mathbf{e}_0) \\ &= \mathbf{e} \wedge \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 + \mathbf{E}(\mathbf{x}_2 - \mathbf{x}_1) \wedge (\mathbf{x}_3 - \mathbf{x}_1) \\ &= \mathbf{e}\mathbf{I}_E\, d + \mathbf{E}\mathbf{n}, \end{aligned} \tag{2.50}$$

with $\mathbf{I}_E = \mathbf{e}_{123}$. Similar to the representation of planes in projective space (equation (2.30)), the dual of a plane is represented by its Hesse form with the normal $\mathbf{n}$ and the distance to the origin $d \in \mathbb{R}$.

2.4.3 Rigid Body Motions

Other important entities which can be defined in conformal geometric algebra as linear operations are rigid body motions. To achieve that, the concept of conformal transformation is used[5]. This section summarizes the most relevant contributions of [62, 79, 85, 87], which describe how conformal transformations can be used to define rigid body motions[6].

It has been mentioned before that only elements of the null cone can be related to elements in Euclidean space. Because of that, it is important that rigid body motions in CGA can be applied to elements of the null cone. Then, the result may be another element of the null cone. As can be seen in [87], elements of the null cone have the property of been invariant under conformal transformations. In consequence, a conformal transformation on the null cone has a respective transformation in Euclidean space.

On the other hand, transformations in $\mathcal{G}_3$ can be defined as a combination of reflections as basic operations as shown in [54]. By following this principle, rotations can be defined as a combination of reflection operations also in $\mathcal{G}_{4,1}$. This is shown in figure 2.6 for the 2D case. It is clear that a rotation of the point $\mathbf{a}$ with respect to the plane is exactly the same rotation for its projected point on the sphere $\underline{\mathbf{a}}$. On the

[5] A conformal transformation is defined as the product $\sigma\underline{\mathbf{X}'} = G\underline{\mathbf{X}}G^{-1}$, where G is a versor and σ a scalar. The conformal transformations are: reflections, inversions, rotations, translations, transversions, dilatation and involution operations.

[6] Let us remember that rigid body motions correspond to the special Euclidean transformation group $SE(3)$, which are contained in the group of conformal transformations.

other hand, the translation of the point $\mathbf{a}$ to $\mathbf{a}'$ corresponds to a special rotation of $\underline{\mathbf{a}}$ to $\underline{\mathbf{a}}'$ on the unit sphere. That means, rotations are basically the same operations in $\mathcal{G}_3$ and $\mathcal{G}_{4,1}$. Since translations can be regarded as special rotations, the combined rigid body motions (rotation plus rotation) can be described in $\mathcal{G}_{4,1}$ as linear operations.

In a first instance, the construction of a rotor as a combination of reflections is presented in $\mathcal{G}_3$. Then, the combination of rotors and translators are represented in $\mathcal{G}_{4,1}$ where rigid body motions are defined by twist rotations [79]. An example of

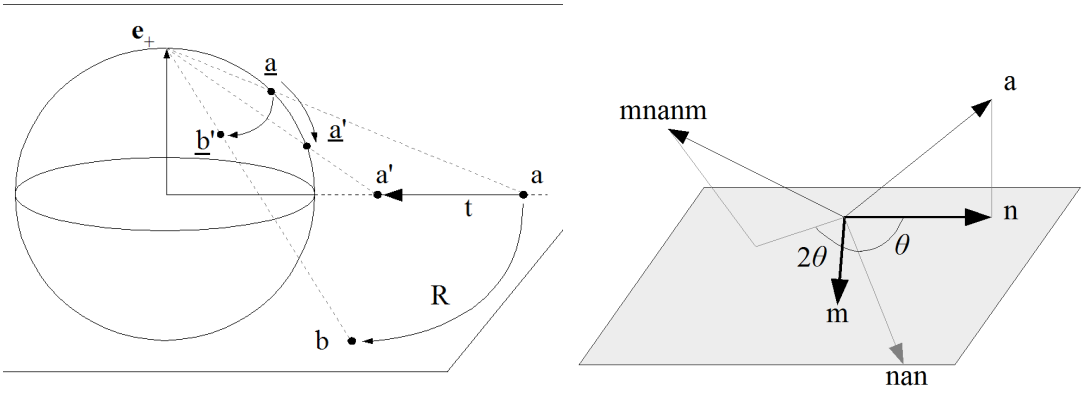

Fig. 2.6: Example of the translation and rotation of a point in the plane and its corresponding transformation in the sphere (left). Combination of reflections to form a rotation (right).

the reflection of a vector is shown in figure 2.6. As it was shown in [85], a vector $\mathbf{a} \in \mathcal{G}_3$ can be reflected with respect to a reference vector $\mathbf{n} \in \mathcal{G}_3$ by simply applying the product $\mathbf{nan}$. In fact, rotations are defined as a combination of two consecutive reflection operations. As can be seen in figure 2.6, the reflection of the vector $\mathbf{a}$ with respect to the vector $\mathbf{n}$ followed by a reflection with respect to $\mathbf{m}$ results in a rotation of $\mathbf{a}$ with respect to the plane spanned by $\mathbf{m} \wedge \mathbf{n}$. Notice that the vector is rotated by an angle of 2θ, with θ defining the angle between the vectors $\mathbf{m}$ and $\mathbf{n}$. Thus, this rotation can be written as

$$\mathbf{b} = (\mathbf{mn})\mathbf{a}(\mathbf{nm}). \tag{2.51}$$

From the definition of the geometric product, it was shown in [85] that the reverse of the product $(\mathbf{nm})^{\sim} = \mathbf{mn}$. Then, equation (2.51) can be rewritten as

$$\mathbf{b} = \mathbf{R}\mathbf{a}\tilde{\mathbf{R}}, \quad \text{with} \quad \mathbf{R} := \mathbf{mn}, \quad \mathbf{R}\tilde{\mathbf{R}} = 1. \tag{2.52}$$

The defined operator $\mathbf{R} \in \mathcal{G}_3$ is called rotor. By expanding this rotor by the definition of the geometric product, it follows that

$$\mathbf{R} = \mathbf{mn} \quad = \quad \mathbf{m} \cdot \mathbf{n} + \mathbf{m} \wedge \mathbf{n}$$

$$\begin{aligned} &= \cos\theta + \sin\theta(\mathbf{m} \wedge \mathbf{n}) \\ &= \exp(\theta \mathbf{l}), \end{aligned} \tag{2.53}$$

where $\mathbf{l} \in \mathcal{G}_3$ is a bivector which represents the normalized version of $\mathbf{m} \wedge \mathbf{n}$ with the property that it squares to -1. $\mathbf{R}$ represents a clockwise rotation by an angle 2θ with respect to the pane $\mathbf{l}$. In order to make a counter clockwise rotation of θ degrees and therefore a mathematically positive rotation, the rotor is written as

$$\mathbf{R} = \exp\left(-\frac{\theta}{2}\mathbf{l}\right). \tag{2.54}$$

In the following, this exponential representation will be used to express rotors in conformal space. Since the Euclidean space was initially embedded in the conformal space, it can be seen that the unit bivector $\mathbf{l} \in \mathcal{G}_3$ is contained in $\mathcal{G}_{4,1}$. Therefore, the rotor $\mathbf{R} \in \mathcal{G}_3$ is also contained in $\mathcal{G}_{4,1}$. Additionally, an operation called translator is defined as a special rotation acting at infinity by using the point at infinity $\mathbf{e}$. These basic rotation and translation operations are represented in exponential form by

$$\mathbf{R} = \exp\left(-\frac{\theta}{2}\mathbf{l}\right) \in \mathcal{G}_{4,1} \tag{2.55}$$

$$\mathbf{T} = \left(1 + \frac{\mathbf{et}}{2}\right) = \exp\left(\frac{\mathbf{et}}{2}\right) \in \mathcal{G}_{4,1}, \tag{2.56}$$

with the vector $\mathbf{t} \in \mathcal{G}_3$ representing the translation vector.

By combining rotors and translators, rigid body motions are defined as $\mathbf{M} := \mathbf{TR}$. This operation is called motor and it is defined as the consecutive application of a rotation followed by a translation. Thus, the rigid body motion of a point $\underline{\mathbf{X}} \in \mathcal{G}_{4,1}$ is done by the operation $\underline{\mathbf{X}'} = \mathbf{M}\underline{\mathbf{X}}\widetilde{\mathbf{M}}$.

It has been proved in [79] that rigid body motions can be defined as a rotation around a line in space combined by a translation along this line. This is known as "screw" operation, see figure 2.7. In a first instance, a general rotation of a point around a line is defined as follows. Initially, the point $\underline{\mathbf{X}} \in \mathcal{G}_{4,1}$ is translated by $\mathbf{T}$ with the distance vector between the line and the origin. At this position, a normal rotation as defined in equation (2.55) is applied. Finally, the point is translated back to its original position by $\widetilde{\mathbf{T}}$. This is known as "twist" operation and it is summarized in the following motor representation

$$\begin{aligned} \mathbf{M} &= \mathbf{TR}\widetilde{\mathbf{T}} \\ &= \left(1 + \frac{\mathbf{et}}{2}\right) \exp\left(-\frac{\theta}{2}\mathbf{l}\right) \left(1 - \frac{\mathbf{et}}{2}\right) \\ &= \exp\left(-\frac{\theta}{2}(\mathbf{l} + \mathbf{e}(\mathbf{t} \cdot \mathbf{l}))\right). \end{aligned} \tag{2.57}$$

Once that the rotation around the line has been done, a translation along this line is performed by applying the translator $\mathbf{T}_{\mathbf{dn}}$, where d represents the amount of the translation. The complete operation is summarized as follows

$$\begin{aligned} \mathbf{M}_s &= \mathbf{T}_{\mathbf{dn}}\mathbf{T}\mathbf{R}\tilde{\mathbf{T}} \\ &= \exp\left(\frac{\mathbf{edn}}{2}\right)\exp\left(-\frac{\theta}{2}(\mathbf{l}+\mathbf{e}(\mathbf{t}\cdot\mathbf{l}))\right) \\ &= \exp\left(-\frac{\theta}{2}\left(\mathbf{l}+\mathbf{e}(\mathbf{t}\cdot\mathbf{l}-\frac{d}{\theta}\mathbf{n})\right)\right). \end{aligned} \tag{2.58}$$

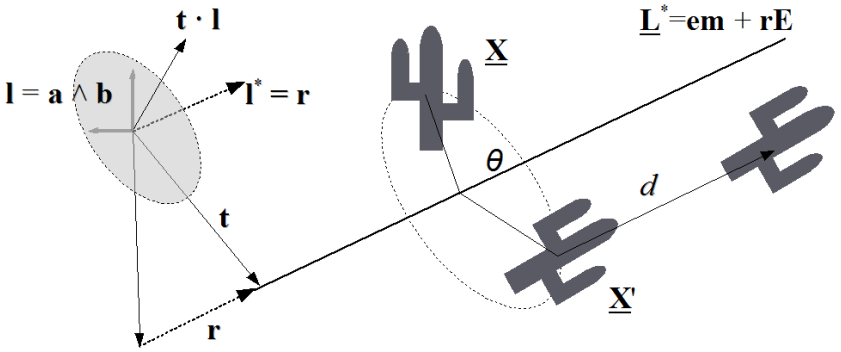

Fig. 2.7: General screw motion defined as a general rotation around a line. It is shown that the parameters of a general rotation around a line correspond to the parameters of this line.

One property of the motor representation of equation 2.57 is that the parameters of the exponential part encode the parameters of a line. A formal prove of this property can be seen in [87]. The geometrical interpretation of this property can be seen in figure 2.7. Let us remember that the dual representation of a line is denoted as $\underline{\mathbf{L}}^* = \mathbf{em}+\mathbf{rE}$. The rotor $\mathbf{R}$ is so defined that the direction vector $\mathbf{r}$ of $\underline{\mathbf{L}}^*$ corresponds to the dual of the rotation plane $\mathbf{l}^*$. Note that the rotation plane is perpendicular to the line $\underline{\mathbf{L}}^*$. If we recall the example of the dual of a bivector presented in section 2.2.1, it can be seen that the parameters of the vector $\mathbf{r}$ correspond to that of $\mathbf{l}^*$ and therefore to the rotation plane defined by the bivector $\mathbf{l}$. On the other hand, the inner product of the bivector $\mathbf{l}$ and the vector $\mathbf{t}$ results in a vector contained in the plane $\mathbf{l}$. It is also perpendicular to the line $\underline{\mathbf{L}}^*$ and it contains the parameters of the moment $\mathbf{m}$ of the line.

2.5 Pose Estimation in the Language of Geometric Algebra

Until this point, all entities involved in the pose estimation problem were described (points, lines, planes, rigid body motions and camera model) as well as the different

spaces and algebras where they are defined. In this section, the interaction between elements of those spaces needed for the formulation of pose estimation constraints is presented. This allows to use the advantages of each algebra for each step of the pose estimation problem. For a set of model points and their corresponding sensor measurements, the pose is computed by minimizing the average Euclidean distance between model and extracted image data. In that sense, appropriate expressions are needed to model such distance measures. Finally, the final constraint equations are presented for two different pose estimation scenarios shown in figure 2.8. The pose estimation based on 3D minimization constraints was presented in [92]. In this approach, 3D information is projectively reconstructed from the image data and compared with model information. The second scenario describes a projective variant based on 2D minimization constraints in the image plane [2, 4]. In this case, model points are projected onto the image plane where they are compared with data extracted from the image. One major difference of this method is that it integrates the perspective projection geometry (in terms of the camera parameters) within the minimization task.

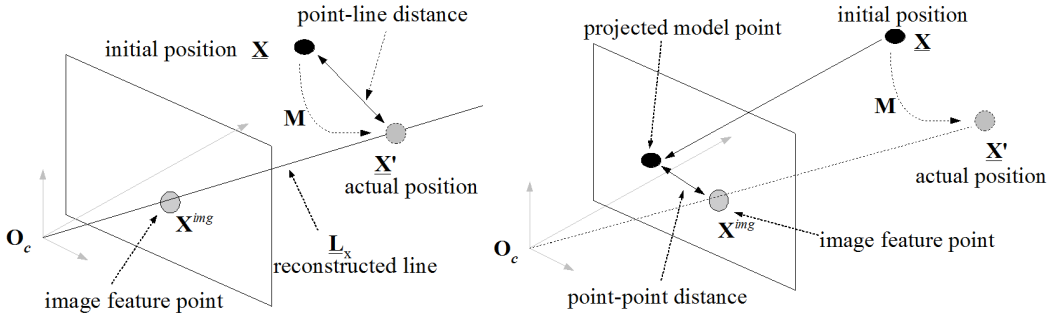

Fig. 2.8: Pose estimation constraints for the 3D case (left). Projective variant of the pose estimation (right).

2.5.1 Change of Representations of Geometric Entities

In this section, the change of representation of geometrical entities (points, lines and planes) between the different spaces will be presented. Depending of the used pose estimation strategy, these entities must be changed from projective to conformal space and vice versa. It has been discussed in last sections that the algebras of Euclidean and projective space are subalgebras of the conformal geometric algebra. Then, the change of an element from one to the other space is done by applying a special operator.

First, the interaction of a point $\mathbf{x} \in \mathcal{G}_3$ in Euclidean space with the corresponding point $\mathbf{X} \in \mathcal{G}_{3,1}$ in projective space as defined in [62] is described as

$$\mathbf{x} \in \mathcal{G}_3 \rightarrow \mathbf{x} + \mathbf{e}_- = \mathbf{X} \in \mathcal{G}_{3,1} \tag{2.59}$$

$$\mathbf{X} \in \mathcal{G}_{3,1} \rightarrow \frac{(\mathbf{X} \wedge \mathbf{e}_-) \cdot \mathbf{e}_-}{\mathbf{X} \cdot \mathbf{e}_-} = \mathbf{x} \in \mathcal{G}_3. \tag{2.60}$$

By applying an operator called projective conformal extension (defined in [87]), an entity is changed from projective to conformal representation. According to the embedding described in section 2.4, this is only valid for entities that are contained in the null cone. To change the representation of elements $\mathbf{A}_p \in \{\mathbf{X}, \mathbf{L}, \mathbf{P}\} \in \mathcal{G}_{3,1}$ to the corresponding dual representations in conformal algebra $\underline{\mathbf{A}}_c^* \in \{\underline{\mathbf{X}}^*, \underline{\mathbf{L}}^*, \underline{\mathbf{P}}^*\} \in \mathcal{G}_{4,1}$, the following operation is applied

$$\underline{\mathbf{A}}_c^* = \mathbf{e} \wedge \mathbf{A}_p. \tag{2.61}$$

To make clear how the projective conformal extension operator works, let us consider the following example for a point $\mathbf{X} = \mathbf{x} + \mathbf{e}_- \in \mathcal{G}_{3,1}$

$$\begin{aligned} \underline{\mathbf{X}}^* \in \mathcal{G}_{4,1} &= \mathbf{e} \wedge (\mathbf{x} + \mathbf{e}_-) \\ &= \mathbf{e} \wedge \mathbf{x} + \mathbf{e} \wedge \mathbf{e}_- \\ &= \mathbf{e}\mathbf{x} + \mathbf{E}, \end{aligned} \tag{2.62}$$

which corresponds to the dual representation of a point of equation (2.48). The same operation is valid for lines $\mathbf{L} \in \mathcal{G}_{3,1}$ and planes $\mathbf{P} \in \mathcal{G}_{3,1}$

$$\begin{aligned} \mathbf{e} \wedge \mathbf{L} &= \mathbf{e} \wedge (\mathbf{e}_-\mathbf{r} + \mathbf{m}) = \mathbf{E}\mathbf{r} + \mathbf{e}\mathbf{m} = \underline{\mathbf{L}}^* \in \mathcal{G}_{4,1} \\ \mathbf{e} \wedge \mathbf{P} &= \mathbf{e} \wedge (\mathbf{e}_-\mathbf{n} + d\mathbf{I}_E) = \mathbf{E}\mathbf{n} + \mathbf{e}d\mathbf{I}_E = \underline{\mathbf{P}}^* \in \mathcal{G}_{4,1}. \end{aligned} \tag{2.63}$$

It can be clearly seen in the last equations that the parameters of points, lines and planes defined in the projective space (equations (2.28), (2.29) and (2.30) respectively) are preserved in the dual representation in conformal space. Only the basis vectors were changed by the projective conformal extension operators.

The opposite operation is done by applying the conformal projective split (defined in [87]). An element $\underline{\mathbf{A}}_c^* \in \{\underline{\mathbf{X}}^*, \underline{\mathbf{L}}^*, \underline{\mathbf{P}}^*\} \in \mathcal{G}_{4,1}$ is changed to its projective representation $\mathbf{A}_p \in \{\mathbf{X}, \mathbf{L}, \mathbf{P}\} \in \mathcal{G}_{3,1}$ by applying

$$\mathbf{A}_p = \mathbf{e}_+ \cdot \underline{\mathbf{A}}_c. \tag{2.64}$$

The following example makes this clear. According to equation (2.6) and the properties of the basis vectors in CGA of equation (2.44), the inner product of a

point $\underline{\mathbf{X}}^* = \mathbf{ex} + \mathbf{E} \in \mathcal{G}_{4,1}$ with $\mathbf{e}_+$ is computed as

$$\begin{aligned} \mathbf{e}_+ \cdot \underline{\mathbf{X}}^* &= \mathbf{e}_+ \cdot (\mathbf{ex} + \mathbf{E}) \\ &= \mathbf{e}_+ \cdot \mathbf{ex} + \mathbf{e}_+ \cdot \mathbf{E} \\ &= \mathbf{x} + \mathbf{e}_+ \cdot (\mathbf{e}_0 \wedge \mathbf{e}_-) \\ &= \mathbf{x} + (\mathbf{e}_+ \cdot \mathbf{e}_0)\mathbf{e}_- - (\mathbf{e}_+ \cdot \mathbf{e}_-)\mathbf{e}_0 \\ &= \mathbf{x} + \mathbf{e}_- \in \mathcal{G}_{3,1}. \end{aligned} \tag{2.65}$$

2.5.2 Incidence Equations

Constraint equations to model incidence of entities in the framework of the dual-quaternion algebra were developed by W. Blaschke [15]. In [93], the commutator $\underline{\times}$ and anti-commutator $\overline{\times}$ products (see section 2.1.1) are used to express collinearity and coplanarity of points, lines an planes. One advantage of this formulation is that Euclidean distance measures can be expressed and computed without adding extra non-linear operators. In consequence, the final constraint equations are also linear.

As an illustrative example[7], let us consider the case of two points $\underline{\mathbf{X}}_1 = \mathbf{ex}_1 + \mathbf{E}$ and $\underline{\mathbf{X}}_2 = \mathbf{ex}_2 + \mathbf{E}$. Their commutator product is given by

$$\begin{aligned} \underline{\mathbf{X}}_1 \underline{\times} \underline{\mathbf{X}}_2 &= \tfrac{1}{2}(\underline{\mathbf{X}}_1\underline{\mathbf{X}}_2 - \underline{\mathbf{X}}_2\underline{\mathbf{X}}_1) \\ &= \tfrac{1}{2}((\mathbf{ex}_1 + \mathbf{E})(\mathbf{ex}_2 + \mathbf{E}) - (\mathbf{ex}_2 + \mathbf{E})(\mathbf{ex}_1 + \mathbf{E})) \\ &= \tfrac{1}{2}((\mathbf{ex}_1\mathbf{ex}_2 + \mathbf{ex}_1\mathbf{E} + \mathbf{Eex}_2 + \mathbf{E}^2) - (\mathbf{ex}_2\mathbf{ex}_1 + \mathbf{ex}_2\mathbf{E} + \mathbf{Eex}_1 + \mathbf{E}^2)) \\ &= \tfrac{1}{2}(-\mathbf{x}_1\mathbf{e} + \mathbf{ex}_1 - \mathbf{ex}_2 + \mathbf{x}_2\mathbf{e}) \\ &= (\mathbf{x}_2 - \mathbf{x}_1)\mathbf{e}. \end{aligned} \tag{2.66}$$

It is clear that the product $\underline{\mathbf{X}}_1 \underline{\times} \underline{\mathbf{X}}_2$ will be zero when the two points are the same. Otherwise, the magnitude of vector $\mathbf{x}_2 - \mathbf{x}_1$ gives the distance between these points.

The point-line and point-plane incidence equations defined in [93] are summarized as follows. A point $\mathbf{X} = \mathbf{E} + \mathbf{ex} \in \mathcal{G}_{4,1}$ is collinear with a line $\mathbf{L} = \mathbf{Er} + \mathbf{em} \in \mathcal{G}_{4,1}$ if the following constraint is satisfied

$$0 = \underline{\mathbf{X}} \quad \underline{\times} \quad \underline{\mathbf{L}} = -(\mathbf{m} - \mathbf{x}\underline{\times}\mathbf{r})\mathbf{e}. \tag{2.67}$$

To make clear the geometrical meaning of the expression $\mathbf{m} - \mathbf{x}\underline{\times}\mathbf{r}$, it is necessary to remember again the Plücker representation of a line [15]. The moment $\mathbf{m}$ of a line

[7] Note that only dual representations of points, lines and planes will be used for the next formulations. The *-sign will be omitted to simplify the notation.

is computed by the outer product of its direction vector $\mathbf{r}$ with any point $\mathbf{x}'$ on the line. If the point $\mathbf{x}$ belongs to the line, it is clear to see that the result of the product will be the same moment $\mathbf{m}$ and the difference will become zero. Otherwise, the magnitude of the resulting bivector $\mathbf{m} - \mathbf{x}\underline{\times}\mathbf{r}$ will describe the distance from the point $\mathbf{x}$ to the line.

Similarly, a point $\underline{\mathbf{X}} = \mathbf{e}\mathbf{x} + \mathbf{E} \in \mathcal{G}_{4,1}$ is coplanar to a plane $\underline{\mathbf{P}} = \mathbf{E}\mathbf{n} + \mathbf{e}d\mathbf{I}_E \in \mathcal{G}_{4,1}$ if

$$0 = \underline{\mathbf{X}} \quad \underline{\times} \quad \underline{\mathbf{P}} = (d\mathbf{I}_E - (\mathbf{x}\overline{\times}\mathbf{n}))\mathbf{e}. \tag{2.68}$$

The anti-commutator product of the normal $\mathbf{n}$ and the vector $\mathbf{x}$ results in a trivector $d'\mathbf{I}_E$ which represents the orthogonal projection of $\mathbf{x}$ onto $\mathbf{n}$. The factor d' is the distance form the origin to the projected point along the normal $\mathbf{n}$. If $\mathbf{x}$ belongs to the plane, then $d = d'$. Otherwise, the difference $d - d'$ is the distance from the point $\mathbf{x}$ to the plane.

Since $\mathbf{e}^2 = 0$, the products $\mathbf{X}\underline{\times}\mathbf{L}$ and $\mathbf{X}\underline{\times}\mathbf{P}$ are elements of the null cone. As it was described in the last sections, if an element is contained in the null cone, it also has a geometrical meaning in the Euclidean space. Thus, these incidence equations are changed to projective space by applying the conformal projective split. By following the definition of projective space (see equation (2.24)), a scaling parameter is applied to express a distance measure in Euclidean space. The complete operation is summarized as follows

$$\begin{aligned} -(\mathbf{m} - \mathbf{x}\underline{\times}\mathbf{r})\mathbf{e}\cdot\mathbf{e}_+ &= \mathbf{m} - \mathbf{x}\underline{\times}\mathbf{r} &\rightarrow\ & \lambda_l\mathbf{m} - \mathbf{x}\underline{\times}(\lambda_l\mathbf{r}) \\ \mathrm{d}\mathbf{I}_E - (\mathbf{x}\overline{\times}\mathbf{n})\mathbf{e}\cdot\mathbf{e}_+ &= d\mathbf{I}_E - (\mathbf{x}\overline{\times}\mathbf{n}) &\rightarrow\ & \lambda_p d\mathbf{I}_E - \mathbf{x}\overline{\times}(\lambda_p\mathbf{n}), \end{aligned} \tag{2.69}$$

with $\lambda_l = \frac{1}{\|\mathbf{r}\|} \in \mathbb{R}$ and $\lambda_p = \frac{1}{\|\mathbf{n}\|} \in \mathbb{R}$.

2.5.3 Pose Estimation Constraints

Once that the interaction between spaces and the incidence equations have been presented, the pose estimation constraints can be constructed. For both pose estimation constraints described in this section, the initial position of the model points and the complete camera model are assumed to be known. The pose estimation scenario for the 3D point-line minimization constraint is shown in figure 2.8. The aim is to compute the unknown motor $\mathbf{M} \in \mathcal{G}_{4,1}$ that transforms the model point $\underline{\mathbf{X}} \in \mathcal{G}_{4,1}$ to its actual position $\underline{\mathbf{X}}' \in \mathcal{G}_{4,1}$ by minimizing the distance from that point to the reconstructed line $\underline{\mathbf{L}} \in \mathcal{G}_{4,1}$.

From its initial position, the model point is rotated and translated by the unknown motor $\mathbf{M} \in \mathcal{G}_{4,1}$ by the operation

$$\underline{\mathbf{X}}' = \mathbf{M} \quad \underline{\mathbf{X}} \quad \tilde{\mathbf{M}}. \tag{2.70}$$

In this position, an image is acquired and the corresponding feature point $\mathbf{X}^{img} \in \mathcal{G}_{3,1}$ is extracted. Then, the optical ray $\mathbf{L}_{\mathbf{x}} \in \mathcal{G}_{3,1}$ is computed with the optical center $\mathbf{O}_c \in \mathcal{G}_{3,1}$ by

$$\mathbf{L}_{\mathbf{x}} = \mathbf{O}_c \quad \wedge \quad \mathbf{X}^{img}. \tag{2.71}$$

Since this line is defined in $\mathcal{G}_{3,1}$, the projective conformal extension operator $\mathbf{e}\wedge$ (see equation (2.61)) must be applied to transform it to $\mathcal{G}_{4,1}$. Once that the line is transformed, the commutator product is applied to express the collinearity of the translated point and the reconstructed line as

$$\underline{\mathbf{X}}' \quad \underline{\times} \quad \mathbf{L}_{\mathbf{x}} = (\mathbf{M} \quad \underline{\mathbf{X}} \quad \tilde{\mathbf{M}}) \quad \underline{\times} \quad \mathbf{e} \wedge (\mathbf{O}_c \quad \wedge \quad \mathbf{X}^{img}). \tag{2.72}$$

In order to get the complete constraint equation and the distance measurement back to Euclidean space, it is first multiplied by the conformal projective split operator $\mathbf{e}_+$ and then scaled with the factor $\lambda \in \mathbb{R}$ (see equations (2.64) and (2.69)). Then, the final point-line constraint equation as defined in [87] has the form

$$\lambda \left(\left(\mathbf{M} \quad \underline{\mathbf{X}} \quad \tilde{\mathbf{M}} \right) \quad \underline{\times} \quad \mathbf{e} \quad \wedge \left(\mathbf{O}_c \quad \wedge \quad \mathbf{X}^{img} \right) \right) \cdot \mathbf{e}_+ = 0. \tag{2.73}$$

It is also possible to define point-plane and line-plane constraint equations, see [87]. In those cases, the optical rays are replaced by 3D planes reconstructed from the image plane. If the object model is defined by a set of 3D lines, the equation is known as line-plane constraint. Once that each line is transformed by the operation, $\mathbf{M}\underline{\mathbf{L}}\widetilde{\mathbf{M}}$, the commutator product with the reconstructed plane is computed and the pose constraint is obtained as described before.

The second pose estimation strategy is also shown in figure 2.8. Instead of minimizing the Euclidean distance in 3D space, the problem is translated onto the image plane. The unknown pose parameters are computed by minimizing the Euclidean distance between detected feature image points and projected model points. As it was described in section 2.3.2, the projection of world points onto the image plane is done by the meet operation in $\mathcal{G}_{3,1}$. If a complete camera model is used, the image plane $\mathbf{P}_{img} \in \mathcal{G}_{3,1}$ is defined according to equation (2.33).

Similar to the first strategy, the model point $\underline{\mathbf{X}} \in \mathcal{G}_{4,1}$ is translated by the unknown motor $\mathbf{M} \in \mathcal{G}_{4,1}$ to its actual position $\underline{\mathbf{X}}' \in \mathcal{G}_{4,1}$. At this position, the line between this point and the optical center is computed by

$$\mathbf{L} = \mathbf{O}_c \quad \wedge \quad (\mathbf{M} \quad \underline{\mathbf{X}} \quad \tilde{\mathbf{M}}) \cdot \mathbf{e}_+. \tag{2.74}$$

Note that the operator $\mathbf{e}_+$ is applied to transform the translated point to projective space. Then, the projection of the point is computed as the intersection of the image plane and the line (see equation (2.35)) as

$$\mathbf{X}^p = \mathbf{P}_{img} \quad \vee \quad (\mathbf{O}_c \wedge \underline{\mathbf{X}}') \quad = \quad \mathbf{P}_{img} \quad \vee \quad \left(\mathbf{O}_c \wedge (\mathbf{M}\underline{\mathbf{X}}\tilde{\mathbf{M}}) \cdot \mathbf{e}_+ \right). \tag{2.75}$$

Finally, the constraint equation in the image plane is defined by expressing the distance between the extracted image point and the projected model point,

$$\begin{aligned} 0 &= \left\{ \mathbf{X}^{img} \quad - \quad \mathbf{X}^{p} \right\} \quad \lambda \\ &= \left\{ \mathbf{X}^{img} \quad - \quad [\mathbf{P}_{img} \quad \vee \quad (\mathbf{O}_c \wedge \underline{\mathbf{X}}')] \right\} \quad \lambda \\ &= \left\{ \mathbf{X}^{img} \quad - \quad \left[\mathbf{P}_{img} \quad \vee \quad \left(\mathbf{O}_c \wedge (\mathbf{M}\underline{\mathbf{X}}\tilde{\mathbf{M}}) \cdot \mathbf{e}_+ \right) \right] \right\} \quad \lambda. \end{aligned} \tag{2.76}$$

In this case, only the rigid body motion of the model point is defined in $\mathcal{G}_{4,1}$. The projection of the transformed point and the distance measure are modeled in projective space. Thus, it is not necessary to apply the commutator product to express incidence for this particular case. If needed, a point-line constraint can be constructed by expressing point-line collinearity in the image plane. Despite of that, only the 3D point-line constraint defined in equation (2.73) and the minimization constraint of equation (2.76) in the image plane are considered in the experimental part of this work.

2.5.4 Linearization of the Pose Parameters

Until now, the equations of both pose minimizations constraints were described. All parameters (object model, camera model and extracted image features) are known, while the unknown motor $\mathbf{M}$ containing the motion parameters has to be found. Since the motor is expressed by its exponential representation, its pose parameters are also encoded in an exponential function. According to the Taylor series power expansion of the exponential function[8], the pose parameters are part of the non-linear polynomial. In this case, the numerical computation of the pose parameters by a standard method may be difficult and time consuming.

As mentioned in the last sections, all the operations involved in the pose estimation constraints are described as linear operators. Therefore, the unknown motor must be also represented or approximated in a linear way to facilitate the computation of the pose parameters. It has been shown in [93], that linear equations are found with respect to the generators of the motor. If the motor $\mathbf{M} = \exp\left(-\frac{\theta}{2}(\mathbf{l} + \mathbf{em})\right)$ is approximated by its Taylor series power expansion the following expression is obtained:

$$\mathbf{M} \quad = \quad 1 - \tfrac{\theta}{2}(\mathbf{l} + \mathbf{em}) + \tfrac{(-\frac{\theta}{2}(\mathbf{l}+\mathbf{em}))^2}{2!} + \tfrac{(-\frac{\theta}{2}(\mathbf{l}+\mathbf{em}))^3}{3!} + \cdots \tag{2.77}$$

The idea is to use only the first order approximation of the last equation to approximate the motor (first two terms of last equation). Then, the rigid body motion

[8] The Taylor series power expansion of the exponential function is defined as: $\exp(\mathbf{x}) = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{x}^n = 1 + \mathbf{x} + \frac{\mathbf{x}^2}{2!} + \frac{\mathbf{x}^3}{3!} + \cdots$

of a point $\mathbf{X}$ is approximated in the following way

$$\begin{aligned} \mathbf{M}\underline{\mathbf{X}}\widetilde{\mathbf{M}} &\approx \left(1-\tfrac{\theta}{2}(\mathbf{l}+\mathbf{em})\right)(\mathbf{ex}+\mathbf{E})\left(1+\tfrac{\theta}{2}(\mathbf{l}+\mathbf{em})\right) \\ &\approx \mathbf{E}+\mathbf{e}(\mathbf{x}-\theta(\mathbf{l}\cdot\mathbf{x})-\theta\mathbf{m}), \end{aligned} \tag{2.78}$$

It has been discussed in [93] that the last approximation implies a mapping of the global rigid body transformation to its generator elements. Therefore, the search of the unknown motion parameters is not done with respect to the Lie group SE(3) but with respect to the Lie algebra se(3).

The approximation of equation (2.78) can be directly used in any of the constraints equations described in last section. For example, if $\mathbf{L} \in \mathcal{G}_{3,1}$ is a reconstructed line from the image plane, the constraint takes the form

$$\lambda\left((\quad \mathbf{E}+\mathbf{e}(\mathbf{x}-\mathbf{l}'\cdot\mathbf{x}-\mathbf{m}')\quad)\quad \underline{\times}\quad \mathbf{e}\wedge\mathbf{L}\right)\cdot\mathbf{e}_+ = 0, \tag{2.79}$$

where $\mathbf{l}' := \theta\mathbf{l}$ and $\mathbf{m}' := \theta\mathbf{m}$. The unknown motion parameters $\mathbf{l}', \mathbf{m}'$ and therefore the complete constraint equation is linear as result of the approximation. For a given set of point-line correspondences, this linear equation can be solved for the motion parameters by applying any standard numeric approach like Householder or QR decomposition methods [47]. Finally, the respective motor $\mathbf{M}$ can be evaluated by applying the Rodriguez formula [79].

This kind of approximation is known as gradient decent method. A detailed analysis of the convergence of this method and a comparison with other approaches are presented in [87]. Roughly speaking, the pose parameters are found in an iterative process, where the computed pose changes incrementally until it converges by fulfilling a given criterion.

2.6 Summary

An introduction to geometric algebras has been presented in this chapter. Once that the geometric algebra of Euclidean space was defined, its relation with other algebras was shown. Then, it was shown that the Euclidean space $\mathbb{R}^3$ is embedded in projective space $\mathbb{PR}^3$ resulting on the algebra of projective space $\mathcal{G}_{3,1}$. In this algebra, the image formation process and basic geometric entities (points, lines and planes) are defined. A stereographic projection is used to perform a second embedding to define the conformal geometric algebra $\mathcal{G}_{4,1}$. This allows to represent rigid body motions as linear multiplicative operations. In this algebra, additional operators to define the interaction between elements of projective and conformal space are defined. Finally, all these concepts are combined to define the 3D and projective pose estimation constraints.

Based on the subspace structure of geometric algebra, the interaction of entities in Euclidean, projective and conformal spaces is described in a very natural way.

In comparison with the classical vector algebra, geometrical entities defined in different spaces are unified in geometric algebra. This allows to model the described strategies to solve the pose estimation estimation problem and to obtain linear relations which can be easily implemented in practice. The presented definitions and properties of the different elements serve as background for the definition of the global and local model representations of free-form models presented in the next chapter.

Chapter 3

GLOBAL AND LOCAL REPRESENTATION OF CONTOURS AND SURFACES

The mathematical framework of the pose estimation problem in CGA has been introduced in the last chapter for cases where the model is represented by a set of points or lines. The next step is to extend the pose estimation constraints for extended model objects like circles, ellipses, spheres, free-form contours and surfaces. Therefore, suitable representations in CGA of those entities are needed. To achieve that, the properties of twist rotations are used. In that context, the relation between curves generated from combinations of twists rotations and Fourier descriptors have been proposed by Rosenhahn et al., see [89, 90, 91]. The properties of the Fourier transform have been used to find a mathematical equivalence between Fourier descriptors and twist rotations. This leads to a global twist-based representation of model objects. For the case of free-form models, the Fourier transform is applied to the parametric representation of free-form contours and surfaces to obtain their corresponding Fourier descriptors. For practical applications, this allows to obtain low-frequency approximations of models used during the iterative process to avoid the local minimum problem.

The idea of twist rotations is essential to define the pose estimation of free-form models from a geometrical and mathematical point of view. On the other hand, it will serve as a basis to define the local representation of such entities. Because of that, the first part of this chapter focuses on describing the relationship between twist rotations and free-form models. Initially, a brief introduction of the basic definitions and properties of free-form models is presented in this chapter. Then, Rosenhahn's idea of using twist rotations to represent extended model objects and free-form contours and surfaces is introduced. It will be shown that twist representations of free-forms contours and surfaces can be also obtained from quaternion and real-valued Hartley transforms, which have some advantages for practical implementations of the algorithms. Once that free-form models are represented in CGA, the combination of the pose estimation constraints with the twist representation of free-form models is described. With a practical example of the pose estimation of a free-form contour model, the effect of using low-frequency approximations during the iterative process is analyzed. One disadvantage of using low-frequency approximations is that the local structure of contour and surface models is lost during the

first iterations.

Let us remember that the aim of the approaches presented in this work is to combine global and local information to improve the pose estimation problem. Because of that, a new local representation of free-form objects in CGA is proposed in this chapter. The twist rotation concept is used to locally approximate the vicinity of contour and surface points. Since segments of contours and surfaces are represented by the action of concatenated twist rotations over a starting point, this local representation is compatible with the pose estimation constraints in CGA. Furthermore, it allows to extract local information which complements the global information obtained from the twist-based global representations.

3.1 Definition and Properties of Free-Form Model Objects

Computer models of free-form objects have been used in a wide range of applications including computer graphics, visualization, robotics and computer vision. In this section, a general overview of the definition and properties of free-form objects is given. Then, examples of parametric contours and surfaces are presented since they will be used to get the twist representation presented in the next sections.

Several authors have defined free-form objects in similar ways. Campbell defines in [24] a free-form object as a composition of one or more nonplanar, nonquadric surfaces.[1] According to Besl [12], a free-form surface has a well defined surface normal which is continuous in almost every point of the surface except at vertices, edge and peak points. The choice of a specific kind of model depends on its specific properties and the application context where it will be applied. Ambiguity, conciseness and uniqueness are the main properties of free-form object representation defined by Brown [18]. Ambiguity refers to the ability of the model to represent the complete object. A model representation is said to be concise if an object is compactly and efficiently described by a given model. Uniqueness is used to determine if there is more than one way to represent the same object given the construction methods of this particular representation. Some computer graphics applications require complete models to generate realistic synthetic images of the represented objects. In contrast to that, an efficient execution of the application is more important for computer vision applications like navigation, recognition or tracking. Therefore, visual fidelity may be sacrificed in those cases. In the context of the monocular pose estimation problem, object models may be represented in 3D space and in the image plane. Because of that, the choice of a given model must also consider adequate techniques for extracting compatible global and local features from objects and input images.

In the following, a description of parametric representations of contours and surfaces is given which are later used to get a twist representation of those entities. The

[1] Roughly speaking, a quadric surface is a surface defined as the locus or zeros of a quadratic polynomial.

simplest and more natural way to describe a 3D free-form contour or surface is by a set of chained points or by a set of parametrical functions (one function for every coordinate axis, see [112]). This representation is also known as CAD (computer aided design) model and is widely used in computer graphics and computer vision applications.

Thus, 3D contours and surfaces are represented by

$$\mathbf{c}(\phi) = \sum_{\mathrm{i}=1}^{3} f^i(\phi)\mathbf{e}_\mathrm{i} \tag{3.1}$$

$$\mathbf{c}(\phi_1, \phi_2) = \sum_{\mathrm{i}=1}^{3} f^i(\phi_1, \phi_2)\mathbf{e}_\mathrm{i}. \tag{3.2}$$

A 3D contour is generated by a set of three functions $f^i(\phi) : \mathbb{R} \rightarrow \mathbb{R}$, while a surface is defined by three functions $f^i(\phi_1, \phi_2) : \mathbb{R}^2 \rightarrow \mathbb{R}$ acting on the different basis vectors $\mathbf{e}_i$. This corresponds to 1D and 2D manifolds embedded in 3D respectively. This reduction of the domain where the model is defined has some advantages for feature computation procedures. Instead of computing features directly in 3D space (which in some cases is not straight forward), a larger variety of standard mathematical tools can be directly applied for the analysis of parametrical functions. For example, 1D and 2D Fourier transforms can be used to get spectral representations of contours and surfaces as it will be shown later. An example of a 3D surface and its corresponding parametric functions is shown in figure 3.1. More complex objects can be represented by the combination of several surface models.

Note that the model of figure 3.1 is considered to be a simplified model since it does not cover all surfaces of the object. Despite of that, the model is constructed in such a way that it is possible to represent all edges of the original object. Then, it is possible to generate adequate artificial images for any arbitrary pose where the projected model edges represent the projected edges of the original object. As it was explained before, this is the key point to extract compatible image and model features derived from edge and contour information. On the other hand, this reduced model is also suitable for visualization purposes and for the experimental validation of the proposed correspondence search approaches for pose estimation presented in the next chapters.

3.2 Coupled Twist Rotations as basic Contour and Surface Generators

The representation of points, lines and planes was discussed in the last chapter as elements of conformal geometric algebra (CGA). These elements were also used to

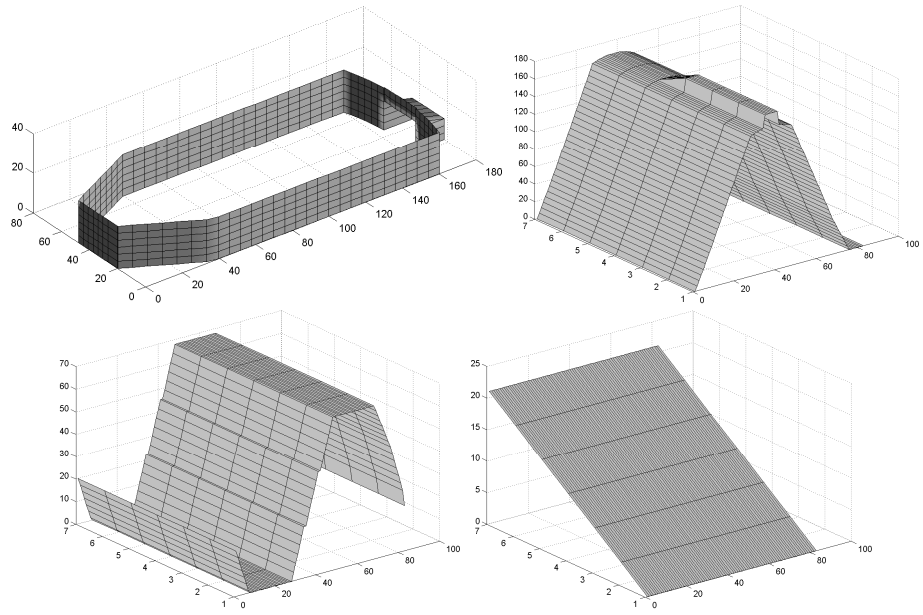

Fig. 3.1: Surface model (upper left) and its corresponding parametric functions.

define the pose estimation constraints in 3D and in the image plane. Those constraints were defined for cases where the model object is defined by a set of points or eventually lines. It is also possible to define the pose estimation constraints for extended object concepts like circles, ellipses and spheres. Then, suitable representations of these entities must be found in CGA in order to incorporate them in the pose equations. To achieve that, twist rotations are used in [93] as basic contour and surface generator elements to define these extended objects. In this section, the principles for the construction of curves and surfaces by coupled twist rotations are briefly introduced.

In order to generate a circle, a twist rotation and a starting point are needed. The idea is shown in figure 3.2. The twist rotation is defined with respect to the rotation axis $\underline{\mathbf{L}}_\phi$. Let us remember that the parameters of the dual representation of a line $\underline{\mathbf{L}}_\phi = \mathbf{em} + \mathbf{rE}$ correspond to the parameters of a rotation around this line, see section 2.4.3 and [87]. Then, the motor which performs a rotation around this line can be directly derived by

$$\begin{aligned} \mathbf{M}_\phi &= \exp\left(-\frac{\phi}{2}(\underline{\mathbf{L}}_\phi \mathbf{I}_c)\right) \\ &= \left(-\frac{\phi}{2}(\mathbf{r}' + \mathbf{em}')\right), \end{aligned} \tag{3.3}$$

with $\mathbf{I}_c = \mathbf{E}\mathbf{I}_E = (\mathbf{e}_+\mathbf{e}_-)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$.

The circle is defined by the orbit of the initial point $\underline{\mathbf{X}}_p$ resulting from the action

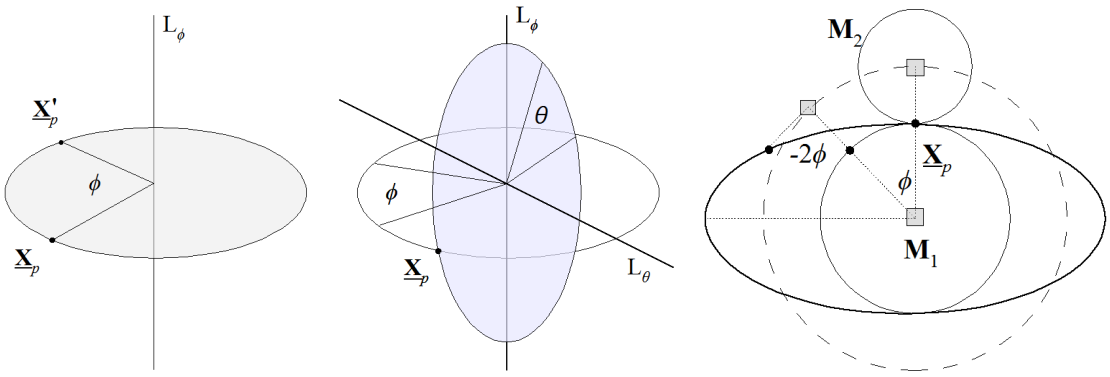

Fig. 3.2: A circle generated by a single twist rotation and a starting point (left). Two coupled rotations with perpendicular rotation axes to generate a sphere (middle). Example of an ellipse generated by two couples twist rotations (right).

of a single motor. All points of the circle are simply generated as follows:

$$\underline{\mathbf{X}}'_p = \quad \mathbf{M}_\phi \underline{\mathbf{X}}_p \widetilde{\mathbf{M}}_\phi \quad : \phi \in [0, \ldots, 2\pi]. \tag{3.4}$$

A sphere is defined by two coupled twists with perpendicular rotation axes as can be seen in figure 3.2. The corresponding motors with respect to each rotation axes have the form

$$\mathbf{M}_\phi = \exp\left(\frac{\phi}{2}(\mathbf{r}'_\phi + \mathbf{e}\mathbf{m}'_\phi)\right) \tag{3.5}$$

$$\mathbf{M}_\theta = \exp\left(\frac{\theta}{2}(\mathbf{r}'_\theta + \mathbf{e}\mathbf{m}'_\theta)\right). \tag{3.6}$$

An example of a generated sphere is shown in figure 3.3. The points on the sphere result from the action of these motors by

$$\underline{\mathbf{X}}'_p = \quad \mathbf{M}_\theta \mathbf{M}_\phi \underline{\mathbf{X}}_p \widetilde{\mathbf{M}}_\phi \widetilde{\mathbf{M}}_\theta \quad : \phi \in [0, \ldots, 2\pi], \theta \in [0, \ldots, 2\pi]. \tag{3.7}$$

A sphere is the basic element which can be constructed by the action of two twist rotations. More general curves and surfaces can be generated by coupling two or three twist rotations in a proper way. Those curves are commonly known as cycloidal curves. Roughly speaking, cycloidal curves are generated from circles rolling on circles or lines. In fact, an ellipse is an example of a cycloidal curve generated by two coupled twists. Several examples of such curves can be consulted in [68]. According to that, the idea for the construction of an ellipse is shown in figure 3.2. In this case, two twist rotations with parallel rotation axes are coupled to generate the ellipse. While the rotation axis of $\mathbf{M}_1$ is fixed, $\mathbf{M}_2$ is rotated by $\mathbf{M}_1$. The initial point $\underline{\mathbf{X}}_p$ is first rotated by an angle of -2ϕ around $\mathbf{M}_2$ and then by ϕ around $\mathbf{M}_1$.

In general, a curve generated by two coupled twists (with parallel rotation axes) is defined as:

$$\underline{\mathbf{X}}'_p = \quad \mathbf{M}^2_{\lambda_2\phi}\mathbf{M}^1_{\lambda_1\phi}\underline{\mathbf{X}}_p\widetilde{\mathbf{M}}^1_{\lambda_1\phi}\widetilde{\mathbf{M}}^2_{\lambda_2\phi} \quad : \lambda_1, \lambda_2 \in \mathbb{R}, \phi \in [0, \ldots, 2\pi], \tag{3.8}$$

where the scalars λ_1, λ_2 define the ratio of angular frequencies between the twist and the initial point $\underline{\mathbf{X}}_p$. Several examples are shown in figure 3.3. Ellipses are generated if $\lambda_1 = -2$ and $\lambda_2 = 1$. If $\lambda_1 = \lambda_2 = 1$, the resulting curve is a cardioid and $\lambda_1 = 2, \lambda_2 = 1$ leads to nephroids among others.

Finally, a third twist rotation perpendicular to the plane in which the curve is defined is added. This results in a 3D surface generated by the following combination of twists:

$$\underline{\mathbf{X}}^{\phi_1,\phi_2}_p = \quad \mathbf{M}^3_{\lambda_3\phi_2}\mathbf{M}^2_{\lambda_2\phi_1}\mathbf{M}^1_{\lambda_1\phi_1}\underline{\mathbf{X}}_p\widetilde{\mathbf{M}}^1_{\lambda_1\phi_1}\widetilde{\mathbf{M}}^2_{\lambda_2\phi_1}\widetilde{\mathbf{M}}^3_{\lambda_3\phi_2}, \tag{3.9}$$

with $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ and $\phi_1, \phi_2 \in [0, \ldots, 2\pi]$. In this case, the surface is parameterized by two rotation angles which correspond to a parametrical representation of a surface as described in section 3.1. The combination of the motors $\mathbf{M}^1_{\lambda_1\phi_1}, \mathbf{M}^2_{\lambda_2\phi_1}$ defines the corresponding contour in the plane. The resulting surface is generated by the action of the motor $\mathbf{M}^3_{\lambda_3\phi_2}$ for all angles ϕ_2. Examples of a 3D sphere, an ellipsoid and a hypocycloid are shown in figure 3.3.

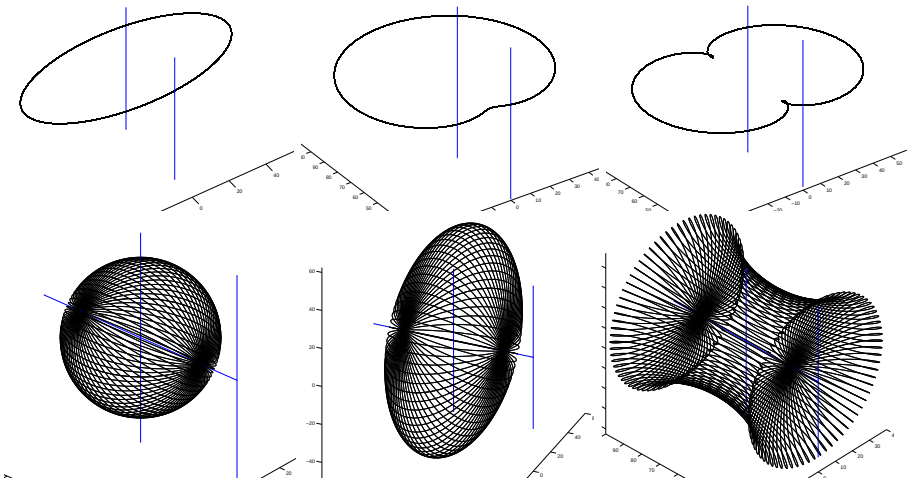

Fig. 3.3: Example of twist generated curves and surfaces.

A large variety of contours and surfaces can be modeled by the combination of a relatively small number of twist rotations. Complex models can be represented by the combination of lines, planes, circles, spheres, etc., where each element must be constructed separately. Despite of that, the representation of models is still limited to objects which can be described by the combination of these geometric entities. In fact, cycloidal curves and surfaces are the intermediate stage for the representation

of 3D free-from contours and surfaces in CGA which are introduced in the next section.

3.3 Twist based Free-form Model Representations

The modeling of a curve or surface by a set of coupled twists was described in the last section. The next step is to find the combination of twist rotations needed to approximate a general 3D curve or surface. This problem is related to Fourier descriptors which are widely used for object recognition applications, see [5, 97, 109]. It has been proved that a set of coupled twists modeling general rotations is equivalent to a sum over a set of rotors which act on different phase vectors. This can be interpreted as a Fourier series expansion which leads to a trigonometric interpolation of a set of contour points. Therefore, Fourier descriptors can be used to model 3D contours and surfaces within the context of the pose estimation problem in conformal geometric algebra.

In this section, the coupling of twist rotations and Fourier descriptors is introduced to describe 3D free-form contours and surfaces. Then, alternative variants to find suitable twist model representations based on the quaternion Fourier transform and the real-valued Hartley transform in 3D and in the image plane are presented. It will be shown that these variants are mathematically consistent with the concept of twist generated curves. On the other hand, the real-valued Hartley transform in the image plane can be used to obtain global orientation information of projected contour and silhouette models. The reader should refer to [89, 90, 91] to get a more detailed introduction to the concepts presented in this section.

3.3.1 Free-form 3D Contours

Let us consider the case of a planar 2D curve to recall the definition of the Fourier series expansion. For a parametric representation of a closed curve $\mathbf{f}(t) \in \mathbb{R}^2$, its Fourier series representation is defined as:

$$\mathbf{f}(t) = \frac{1}{M} \sum_{u=-\infty}^{u=\infty} \mathbf{F}(u) \exp\left(\frac{2\pi i u t}{M}\right), \tag{3.10}$$

where $\mathbf{F}(u)$ are the Fourier coefficients, $u \in \mathbb{Z}$ the frequency, i the imaginary unit, with $i^2 = -1$, and M the length of the curve. The idea is to find a relation between this Fourier representation and a twist generated curve. In a first instance, the case of a curve generated by two coupled twists is considered. According to equation (3.8), a curve generated by two twist rotations is defined as $\underline{\mathbf{X}}_p^{\phi} = \mathbf{M}^2_{\lambda_2\phi}\mathbf{M}^1_{\lambda_1\phi}\underline{\mathbf{X}}_p\widetilde{\mathbf{M}}^1_{\lambda_1\phi}\widetilde{\mathbf{M}}^2_{\lambda_2\phi}$. Since equation (3.10) is only valid in Euclidean space, the curve is initially repre-

sented in Euclidean space[2] as

$$\mathbf{x}_p^{\phi} = \mathbf{R}^2_{\lambda_2\phi}\left((\mathbf{R}^1_{\lambda_1\phi}(\mathbf{x}_c - \mathbf{t}_1)\widetilde{\mathbf{R}}^1_{\lambda_1\phi} + \mathbf{t}_1) - \mathbf{t}_2\right)\widetilde{\mathbf{R}}^2_{\lambda_2\phi} + \mathbf{t}_2. \tag{3.11}$$

After some algebra and reordering of the terms, the last equation can be rewritten as

$$\mathbf{x}_p^{\phi} = \mathbf{p}_0 + \mathbf{V}^1_{\lambda_1\phi}\mathbf{p}_1\widetilde{\mathbf{V}}^1_{\lambda_1\phi} + \mathbf{V}^2_{\lambda_2\phi}\mathbf{p}_2\widetilde{\mathbf{V}}^2_{\lambda_2\phi}, \tag{3.12}$$

with $\mathbf{p}_0 = \mathbf{t}_2$, $\mathbf{p}_1 = \mathbf{t}_1 - \mathbf{t}_2$, $\mathbf{p}_2 = \mathbf{x}_c - \mathbf{t}_1$, $\mathbf{V}^1_{\lambda_1\phi} = \mathbf{R}^2_{\lambda_2\phi}$, $\mathbf{V}^2_{\lambda_2\phi} = \mathbf{R}^2_{\lambda_2\phi}\mathbf{R}^1_{\lambda_1\phi}$ and $\lambda_i = 2\pi u_i/M$.

If a vector $\mathbf{x}$ lies in the rotation plane of a rotor, the relation $\mathbf{R}\mathbf{x} = \mathbf{x}\widetilde{\mathbf{R}}$ is valid (see [90]). The square of a rotor is equal to a rotor of twice the angle if both rotors are defined in the same rotation plane, then $\widetilde{\mathbf{V}}^i_{\phi}\widetilde{\mathbf{V}}^i_{\phi} = \widetilde{\mathbf{V}}^i_{2\phi}$. Based on the last properties, the curve can be rewritten by expressing the rotors in their exponential form

$$\begin{aligned}\underline{\mathbf{x}}_p^{\phi} &= \mathbf{p}_0 + \mathbf{p}_1\widetilde{\mathbf{V}}^1_{2\phi} + \mathbf{p}_2\widetilde{\mathbf{V}}^2_{2\phi} \\ &= \mathbf{p}_0 + \mathbf{p}_1\exp\left(\frac{2\pi u_1\phi}{M}\mathbf{l}\right) + \mathbf{p}_2\exp\left(\frac{2\pi u_2\phi}{M}\mathbf{l}\right).\end{aligned} \tag{3.13}$$

This representation of a curve is equivalent to the Fourier series expansion of equation (3.10). The imaginary unit has been replaced by the unit bivector $\mathbf{l}$ representing the rotation plane and the Fourier coefficients by the phase vectors $\mathbf{p}_i$ which are also defined in the plane. If we express every exponential function as $\mathbf{R}_{k,\phi} = \exp\left(\frac{2\pi k\phi}{M}\mathbf{l}\right)$, any closed plane curve $\mathbf{c}(\phi)$ can be expressed as a series expansion

$$\mathbf{c}(\phi) = \lim_{n\to\infty}\sum_{k=-n}^{n}\mathbf{p}_k\exp\left(\frac{2\pi k\phi}{M}\mathbf{l}\right) = \lim_{n\to\infty}\sum_{k=-n}^{n}\mathbf{R}_{k,\phi}\mathbf{p}_k\widetilde{\mathbf{R}}_{k,\phi}. \tag{3.14}$$

According to the last equation, there is a unique set of phase vectors $\{\mathbf{p}_0, \ldots, \mathbf{p}_k\}$ that describe the curve. In [90], they are also called Fourier descriptors of a closed curve. The set of phase vectors are generator elements of the curve, since it is described as the combination of twist rotations acting on them.

Let us remember that the last procedure is only valid for planar curves. For the case of a general non-planar curve in 3D space, the phase vectors are computed separately for each signal $f^i(\phi)$ of the parametrical representation of a contour of equation (3.1). This is equivalent to project the 3D curve onto each plane defined by $\mathbf{e}_{12}$, $\mathbf{e}_{23}$ and $\mathbf{e}_{31}$ respectively. On the other hand, discrete curves representing object models or image extracted contours are used for practical applications. If a finite

[2] The motor $\mathbf{M}^1_{\lambda_1\phi}$ represents a general rotation defined by a translation of the point $\mathbf{x}_c$ by the vector $-\mathbf{t}_1$ followed by a rotation $\mathbf{R}^1_{\lambda_1\phi}$ and finally a translation $\mathbf{t}_1$.

number of twists are considered, the Fourier series expansion becomes the inverse discrete Fourier transform. Then, every parametric function $f^i(\phi)$ is approximated as

$$f^i(\phi) = \sum_{k=-N}^{N} \mathbf{p}_k^i \exp\left(\frac{2\pi k\phi}{2N+1}\mathbf{l}_i\right), \tag{3.15}$$

where the phase vectors are computed by the discrete Fourier transform as

$$\mathbf{p}_k^i = \frac{1}{2n+1}\sum_{j=-N}^{N} f^i(j)\exp\left(-\frac{2\pi kj}{2N+1}\mathbf{l}_i\right). \tag{3.16}$$

This leads to the following representation of a general 3D curve

$$\begin{aligned}\mathbf{c}(\phi) &= \sum_{i=1}^{3}\sum_{k=-N}^{N} \mathbf{p}_k^i \exp\left(\frac{2\pi k\phi}{2N+1}\mathbf{l}_i\right)\mathbf{e}_i \\ &= \sum_{i=1}^{3}\left(\sum_{k=-N}^{N} \mathbf{R}_{k,\phi}^i \mathbf{p}_k \widetilde{\mathbf{R}}_{k,\phi}^i\right)\mathbf{e}_i. \end{aligned}\tag{3.17}$$

where $\mathbf{l}_1 = \mathbf{e}_{12}$, $\mathbf{l}_2 = \mathbf{e}_{23}$ and $\mathbf{l}_3 = \mathbf{e}_{31}$.

3.3.2 Free-form 3D Surfaces

Similar to the case of 3D contours, the parametric representation of a surface introduced in equation (3.1) is used to get the corresponding Fourier descriptors. Then, a parametric surface is defined by three functions $f^i(\phi_1, \phi_2) : \mathbb{R}^2 \rightarrow \mathbb{R}$ over each base vector $\mathbf{e}_i$. Each function f^i is orthogonal to one of the planes spanned by the bivectors $\mathbf{e}_{12}$, $\mathbf{e}_{23}$ and $\mathbf{e}_{31}$. Thus, the 2D discrete Fourier transform can be applied to each function $f^i(\phi_1, \phi_2)$ to compute the phase vectors as follows

$$\begin{aligned}\mathbf{p}_{k_1,k_2}^i &= \frac{1}{(2N_1+1)(2N_2+1)} \\ &\sum_{n_1=-N_1}^{N_1}\sum_{n_2=-N_2}^{N_2} f^i(n_1,n_2)\exp\left(-\frac{2\pi k_1 n_1}{2N_1+1}\mathbf{l}_i\right)\exp\left(-\frac{2\pi k_2 n_2}{2N_2+1}\mathbf{l}_i\right),\end{aligned}\tag{3.18}$$

where the imaginary unit has been replaced by three different rotation planes defined by the bivectors $\mathbf{l}_i$, with the property $\mathbf{l}_i^2 = -1$. The Fourier descriptors of each parametric function are contained in the phase vectors $\mathbf{p}_{k_1,k_2}^i$ which lie on the plane spanned by $\mathbf{l}_i$.

Once that the Fourier descriptors are known, the surface can be approximated as a series expansion as

$$\begin{aligned}\mathbf{c}(\phi_1,\phi_2) &= \sum_{i=1}^{3}\left(\sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2}\mathbf{p}^i_{k_1,k_2}\exp\left(\frac{2\pi k_1\phi_1}{2N_1+1}\mathbf{l}_i\right)\exp\left(\frac{2\pi k_2\phi_2}{2N_2+1}\mathbf{l}_i\right)\right)\mathbf{e}_i \\ &= \sum_{i=1}^{3}\left(\sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2}\mathbf{R}^{1,i}_{k_1,\phi_1}\mathbf{R}^{2,i}_{k_2,\phi_2}\mathbf{p}^i_{k_1,k_2}\widetilde{\mathbf{R}}^{2,i}_{k_2,\phi_2}\widetilde{\mathbf{R}}^{1,i}_{k_1,\phi_1}\right)\mathbf{e}_i. \end{aligned} \tag{3.19}$$

Each parametric function is defined with the phase vectors as generator elements under the concatenated action of the twists rotations $\mathbf{R}^{1,i}_{k_1,\phi_1}$ and $\mathbf{R}^{2,i}_{k_2,\phi_2}$.

3.3.3 Quaternion Fourier Descriptors for 3D Contours and Surfaces

The quaternion Fourier transform (QFT) has been used for image processing applications including spectral analysis of color images, see [96]. In this case, each of the three components of the image (RGB) are encoded in a pure quaternion element and the 2D quaternion Fourier transform is applied. Since 3D contours and surfaces are represented by a set of three parametric functions, the discrete QFT can be also applied to them [7]. Furthermore, the relation between Quaternion Fourier transform and Clifford algebras has been studied in [56]. Since the quaternion algebra is isomorphic to the rotor subalgebra $\mathcal{G}_3^+ \in \mathcal{G}_3$, the QFT also delivers a twist representation of contours and surfaces.

In a first instance, let us consider the case of 3D contours. According to the approach presented in [7], the parametric functions must be arranged in a pure quaternion function as: $\mathbf{f}_{\phi_1} = f^1(\phi_1)\mathbf{i} + f^2(\phi_1)\mathbf{j} + f^3(\phi_1)\mathbf{k}$. In [96], the discrete quaternion Fourier transform of $\mathbf{f}_{\phi_1}$ has been defined by

$$\mathbf{q}_{k_1} = \frac{1}{N_1}\sum_{n_1=-N_1}^{N_1}\mathbf{f}_{n_1}\exp\left(-\frac{2\pi k_1 n_1}{N_1}\bar{\mathbf{h}}\right), \tag{3.20}$$

where $\bar{\mathbf{h}}$ is a unit pure quaternion with $\bar{\mathbf{h}}^2 = -1$. In section 2.2.1, the imaginary quaternion units were identified with the following bivectors $\mathbf{i} \to \mathbf{e}_{23}$, $\mathbf{j} \to \mathbf{e}_{12}$ and $\mathbf{k} \to \mathbf{e}_{31}$, with $\mathbf{e}_{23}, \mathbf{e}_{12}, \mathbf{e}_{31} \in \mathcal{G}_3$. The equivalent operation in $\mathcal{G}_3^+ \in \mathcal{G}_3$ can be done if the pure quaternion containing the parametric functions is rewritten as

$$\mathbf{f}_{\phi_1} = f^1(\phi_1)\mathbf{e}_{23} + f^2(\phi_1)\mathbf{e}_{12} + f^3(\phi_1)\mathbf{e}_{31}.$$

Similarly, the pure unitary quaternion is replaced with the bivector $\mathbf{h} = \mathbf{e}_{23} + \mathbf{e}_{12} + \mathbf{e}_{31}$, with $\bar{\mathbf{h}} = \mathbf{h}/\|\mathbf{h}\|$, representing the rotation plane. Instead of computing three separate sets of phase vectors (one for each projection of the curve onto the

plane $\mathbf{e}_{ij}$), only one set of quaternion phase vectors is computed. Furthermore, the obtained quaternion phase vectors represent 3D generator elements of the curve which are not necessarily defined in the rotation plane $\bar{\mathbf{h}}$. Once that the quaternion phase vectors are known, the contour is approximated as series expansion by

$$\begin{aligned} c(\phi_1) &= \sum_{k_1=-N_1}^{N_1} \mathbf{q}_{k_1} \exp\left(\frac{2\pi k_1 \phi_1}{N_1}\bar{\mathbf{h}}\right) \\ &= \sum_{k_1=-N_1}^{N_1} \mathbf{R}_{k_1,\phi_1}\mathbf{q}_{k_1}\widetilde{\mathbf{R}}_{k_1,\phi_1} \\ &= f^1(\phi_1)\mathbf{e}_{23} + f^2(\phi_1)\mathbf{e}_{12} + f^3(\phi_1)\mathbf{e}_{31}. \end{aligned} \tag{3.21}$$

This twist representation can be considered to be more general than the one obtained from the Fourier transform. In contrast to equation (3.17), the twist rotation $\mathbf{R}_{k_1,\phi_1}$ is defined with respect to the general plane $\bar{\mathbf{h}}$ in 3D instead of each plane $\mathbf{e}_{ij}$. Then, the action of this rotation over the 3D phase vectors approximates the curve directly in 3D.

In order to apply the quaternion Fourier transform to 3D surfaces, each 2D parametric function is arranged in a pure quaternion as

$$\mathbf{f}_{\phi_1,\phi_2} = f^1(\phi_1,\phi_2)\mathbf{e}_{23} + f^2(\phi_1,\phi_2)\mathbf{e}_{31} + f^3(\phi_1,\phi_2)\mathbf{e}_{12}.$$

Then, the quaternion phase vectors are computed by applying a 2D discrete quaternion Fourier transform as

$$\mathbf{q}_{k_1,k_2} = \frac{1}{N_1 N_2}\sum_{n_1=-N_1}^{N_1}\sum_{n_2=-N_2}^{N_2} \mathbf{f}_{n_1,n_2}\exp\left(-\frac{2\pi k_1 n_1}{N_1}\bar{\mathbf{h}}\right)\exp\left(-\frac{2\pi k_2 n_2}{N_2}\bar{\mathbf{h}}\right). \tag{3.22}$$

Finally, the surface is approximated as a series expansion

$$\begin{aligned} \mathbf{c}(\phi_1,\phi_2) &= \sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2} \mathbf{q}_{k_1,k_2}\exp\left(\frac{2\pi k_1\phi_1}{N_1}\bar{\mathbf{h}}\right)\exp\left(\frac{2\pi k_2\phi_2}{N_2}\bar{\mathbf{h}}\right) \\ &= \sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2} \mathbf{R}^1_{k_1,\phi_1}\mathbf{R}^2_{k_2,\phi_2}\mathbf{q}_{k_1,k_2}\widetilde{\mathbf{R}}^2_{k_2,\phi_2}\widetilde{\mathbf{R}}^1_{k_1,\phi_1} \\ &= f^1(\phi_1,\phi_2)\mathbf{e}_{23} + f^2(\phi_1,\phi_2)\mathbf{e}_{12} + f^3(\phi_1,\phi_2)\mathbf{e}_{31}. \end{aligned} \tag{3.23}$$

3.3.4 Real valued Hartley Approximations

Ralph Hartley (1942) developed a symmetrical real valued Fourier transform known in the literature as Hartley transform. In contrast to the Fourier transform, the integral kernel of the Hartley transform is a real kernel defined by the "cas" function

as $\mathrm{cas}(t) = \cos(t) + \sin(t)$. Originally, the discrete Hartley transform (DHT) was proposed in [16]. Its properties have been studied to develop fast and efficient algorithms for applications in computer science [17, 22]. The real and imaginary parts of the Fourier transform can be directly obtained as the even and odd parts of the DHT. That means, Fourier contour and surface descriptors can be obtained directly from the DHT without explicitly calculating the Fourier transform. Therefore, the Hartley transform is a suitable substitute of the Fourier transform.

Let us consider the case of a 3D curve. The corresponding Hartley descriptors of a 3D contour are obtained by applying the DHT to the parametric representation of a contour of equation (3.1). Then, the corresponding Hartley descriptors for each function $f^i(\phi)$ are computed as follows

$$\mathbf{p}_k^i = \frac{1}{2N+1} \sum_{n=-N}^{N} f^i(n) \left(\cos\left(\frac{2\pi kn}{2N+1}\right) + \sin\left(\frac{2\pi kn}{2N+1}\right)\right). \tag{3.24}$$

With this set of Hartley descriptors, the contour is approximated as

$$\mathbf{c}(\phi) = \sum_{i=1}^{3} \left[\sum_{k=-N}^{N} \mathbf{p}_k^i \left(\cos\left(\frac{2\pi k\phi}{2N+1}\right) + \sin\left(\frac{2\pi k\phi}{2N+1}\right)\right)\right] \mathbf{e}_i. \tag{3.25}$$

The Hartley descriptors of last equation are also known as elliptic Fourier descriptors. They have been used in the literature for recognition of 2D closed contours, see [58, 63]. These descriptors have a quite different geometrical interpretation than the complex or quaternion Fourier descriptors. Instead of describing closed curves as a set of coupled circles as described in last sections, closed curves are described with sets of coupled ellipses. In order to clarify this idea, let us consider the case of 2D contours in the image plane with the following parametric representation

$$\mathbf{c}(\phi) = f_p^1(\phi)\mathbf{e}_1 + f_p^2(\phi)\mathbf{e}_2. \tag{3.26}$$

The last parametric functions that define a closed contour parameterized within the range $0 \leq \phi < 2\pi$ can be expressed by Fourier series expansions in matrix form as

$$\begin{aligned} \begin{bmatrix} f_p^1(\phi) \\ f_p^2(\phi) \end{bmatrix} &= \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} a_k & b_k \\ c_k & d_k \end{bmatrix} \begin{bmatrix} \cos(k\phi) \\ \sin(k\phi) \end{bmatrix} \\ &= \mathbf{m}_0 + \sum_{k=1}^{\infty} \mathrm{E}_k \begin{bmatrix} \cos(k\phi) \\ \sin(k\phi) \end{bmatrix}, \end{aligned} \tag{3.27}$$

where the coefficients are defined as:

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f_p^1(\phi)\, d_\phi \qquad b_0 = \frac{1}{2\pi}\int_0^{2\pi} f_p^2(\phi)\, d_\phi$$

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f_p^1(\phi)\cos(k\phi)\,d_\phi \qquad b_k = \frac{1}{2\pi}\int_0^{\pi} f_p^1(\phi)\sin(k\phi)\,d_\phi$$
$$c_k = \frac{1}{\pi}\int_{k=0}^{2\pi} f_p^2(\phi)\cos(k\phi)\,d_\phi \qquad d_k = \frac{1}{\pi}\int_0^{\pi} f_p^2(\phi)\sin(k\phi). \tag{3.28}$$

Let us notice that the last coefficients correspond to the even and odd parts of the Hartley transform of equation (3.24).

The parameters of each ellipse can be obtained from the set of coefficients a_k, b_k, c_k, d_k arranged in the matrix E_k of equation (3.27). The vector $\mathbf{m}_0$ is the center of gravity of the contour. An example to illustrate the approximation of a curve with elliptic descriptors is shown in figure 3.4. If only one elliptic descriptor is used, the contour is approximated by the ellipse E_1. The center of the ellipse E_2 revolves along the orbit defined by E_1. Then, the contour is approximated by the concatenated rotations of an initial point of the curve along the orbits of E_2 and E_1. As more descriptors are used, a better approximation of the contour is obtained.

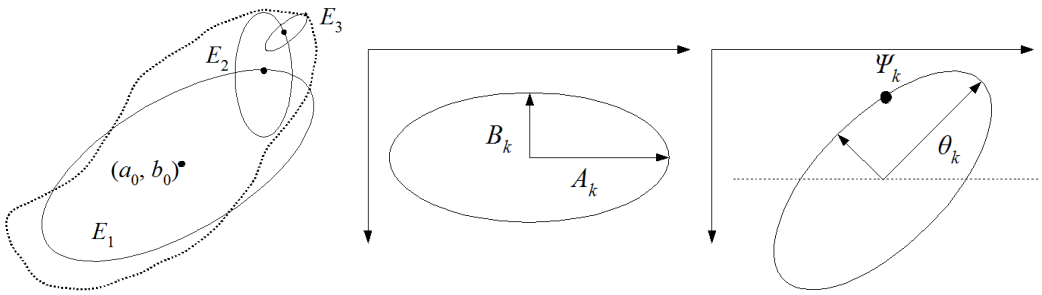

Fig. 3.4: Left: approximation of a 2D curve with elliptic descriptors. Middle and right: geometrical features of the elliptic twist descriptors: A_k and B_k are the major and minor axis respectively, θ_k orientation of the major axes and ψ_k the phase angle.

By applying the singular value decomposition to the matrix E_k, the parameters of each ellipse can be recovered as follows

$$\text{svd}\,(E_k) \;=\; \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \begin{bmatrix} A_k & 0 \\ 0 & B_k \end{bmatrix} \begin{bmatrix} \cos(\psi_k) & -\sin(\psi_k) \\ \sin(\psi_k) & \cos(\psi_k) \end{bmatrix}. \tag{3.29}$$

An example of the geometrical interpretation of such parameters is shown in figure 3.4. The scalars A_k and B_k are the lengths of the major and minor axes respectively. The angle θ_k stands for the orientation of the major axis with respect to the x image axis and ψ_k is the phase angle. In this context, phase refers to the position of the initial point of the contour with respect to the major axis. As can be seen in

figure 3.4, the first ellipse can be considered a global rough approximation of the curve with the major and minor axes describing the main distribution directions of the contour points.[3] Then, the point (a_0, b_0) together with the angle θ_k are considered the global position and orientation of the contour in the image plane. This global information is relevant for the approaches presented in the next chapters since it will be used to define new strategies to solve the correspondence search problem in the image plane.

3.3.5 Time Performance Comparison

The presented Fourier, quaternion Fourier and Hartley transforms deliver suitable twists descriptors of contours and surfaces with their corresponding low-frequency approximations. The question now is, which of these variants is better suited for practical applications. As an example, a comparison of the computation times for the presented Fourier, quaternion valued Fourier transform and Hartley transform was done for the power socket model in 3D as it is shown in figure 3.5.

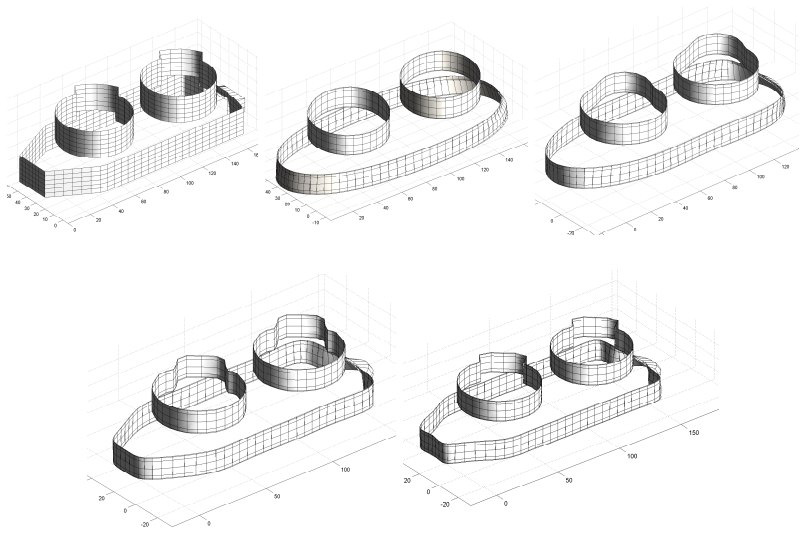

Fig. 3.5: Low-frequency approximations of the power socket model.

This model consists of three different surfaces with a total number of 1043 points. For practical pose estimation applications, low-frequency approximations must be computed during the iterative process. The set of descriptors are computed and stored off-line for the complete model. Only the computation times of the inverse transformations were compared in this experiment. In this case, discrete versions of the transforms were implemented (better computation times can be achieved with

[3] This is analog to compute the eigenvectors of a covariance matrix defined with the 2D contour points.

Transform	2 descriptors	5 descriptors	10 descriptors	15 descriptors
Fourier	6 ms	56 ms	158 ms	242 ms
Quaternion	5 ms	45 ms	139 ms	210 ms
Hartley	3 ms	31 ms	95 ms	142 ms

Tab. 3.1: Time comparison of the low-frequency approximation of the power socket model for 2, 5, 10 and 15 descriptors.

the use of fast algorithms). Low-frequency approximations were computed with 2, 5, 10 and 15 descriptors for every inverse transform.

A direct comparison of the computation times is shown in table 3.1. All variants were tested on a Linux based system with a 3 GHz Intel Pentium 4 processor. As expected, the fastest approximations are obtained with the Hartley transform since no complex calculations are needed. Although the quaternion variant uses special functions to perform quaternion multiplications, the computation time is even lower than that of the Fourier approximations. Let us remember that only one quaternion inverse Fourier transform is needed to approximate the surface, while three inverse transforms are needed in the complex valued Fourier transform case. This simple time performance comparison of the different variants shows the convenience of using the real valued Hartley transform for practical applications. Better time performance can be achieved by applying fast Fourier transform algorithms.

3.4 Pose Estimation of Twist Generated Models

The combination of the pose estimation constraints (see section 2.5.3) with the twist based model representations is described in this section. Since all elements have been compactly represented in their respective algebra, the interpretation of the constraint equations is quite intuitive and simple. Although several constraints have been defined, see [92], only the 3D point-line constraint is used in this section to describe the pose estimation of twist generated models. The same principle is used to define 3D point-plane constraints or 2D projective constraints. Finally, the use of Fourier approximations for a practical correspondence search and pose estimation example is also presented in this section.

Let us recall the point-line constraint equation in conformal space

$$0 = \left(\mathbf{M} \quad \underline{\mathbf{X}} \quad \widetilde{\mathbf{M}}\right) \quad \underline{\times} \quad \mathbf{e} \quad \wedge \left(\mathbf{O}_c \quad \wedge \quad \mathbf{X}^{img}\right). \tag{3.30}$$

The model point $\underline{\mathbf{X}}$ is transformed by the unknown motor $\mathbf{M}$ by the operation $\mathbf{M}\underline{\mathbf{X}}\widetilde{\mathbf{M}}$. Then, the collinearity of the transformed point is computed with respect

to the reconstructed line $\underline{\mathbf{L}} = \mathbf{e} \wedge (\mathbf{O}_c \wedge \mathbf{X}^{img})$ by the operator $\underline{\times}$. Finally, the pose is found by finding the motor that minimizes this collinearity measure. In order to adapt this constraint equation for extended model objects, the model point $\underline{\mathbf{X}}$ is replaced in the equation by a given twist generated entity in a proper way.

To define the constraint equation for two and three-twists generated models (planar curves and surfaces), equations (3.8) and (3.9) are directly substituted in the point-line constraint equation as

$$0 = \left(\mathbf{M}(\mathbf{M}^2_{\lambda_2\phi}\mathbf{M}^1_{\lambda_1\phi}\underline{\mathbf{X}}_p\widetilde{\mathbf{M}}^1_{\lambda_1\phi}\widetilde{\mathbf{M}}^2_{\lambda_2\phi})\widetilde{\mathbf{M}}\right) \quad \underline{\times} \quad \underline{\mathbf{L}} \tag{3.31}$$

$$0 = \left(\mathbf{M}(\mathbf{M}^3_{\lambda_3\phi_2}\mathbf{M}^2_{\lambda_2\phi_1}\mathbf{M}^1_{\lambda_1\phi_1}\underline{\mathbf{X}}_p\widetilde{\mathbf{M}}^1_{\lambda_1\phi_1}\widetilde{\mathbf{M}}^2_{\lambda_2\phi_1}\widetilde{\mathbf{M}}^3_{\lambda_3\phi_2})\widetilde{\mathbf{M}}\right) \quad \underline{\times} \quad \underline{\mathbf{L}}. \tag{3.32}$$

In this case, the last equations express the rigid body motion $\mathbf{M}$ of a curve or surface incident to a reconstructed optical ray. The last equations consider the initial point $\underline{\mathbf{X}}_p$ and the generator twists of the curve or surface respectively. Since the generator elements are also twist rotations, they can be directly incorporated in a linear way in the constraint equations. Thus, the unknown parameters of the last equations are the motor $\mathbf{M}$ and the motors $\widetilde{\mathbf{M}}^1_{\lambda_1\phi_1}$, $\widetilde{\mathbf{M}}^2_{\lambda_2\phi_1}$ and $\widetilde{\mathbf{M}}^3_{\lambda_3\phi_2}$ which define the curve.

In a similar way, the point-line constraint equation is defined for free-form contours and surfaces as

$$0 = \left(\mathbf{M} \quad \underline{\mathbf{M}}_{3D} \quad \widetilde{\mathbf{M}}\right) \quad \underline{\times} \quad \mathbf{e} \quad \wedge \left(\mathbf{O}_c \quad \wedge \quad \mathbf{X}^{img}\right), \tag{3.33}$$

where the approximated model $\underline{\mathbf{M}}_{3D}$ is defined by

$$\underline{\mathbf{M}}_{3D} = \mathbf{e} \wedge \left(\left(\sum_{i=1}^{3}\left(\sum_{k=-n}^{n}\mathbf{R}^i_{k,\phi}\mathbf{p}_k\widetilde{\mathbf{R}}^i_{k,\phi}\right)\mathbf{e}_i\right) + \mathbf{e}_-\right)$$

for the case of 3D free-form contours and

$$\underline{\mathbf{M}}_{3D} = \mathbf{e} \wedge \left(\left(\sum_{i=1}^{3}\left(\sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2}\mathbf{R}^{1,i}_{k_1,\phi_1}\mathbf{R}^{2,i}_{k_2,\phi_2}\mathbf{p}^i_{k_1,k_2}\widetilde{\mathbf{R}}^{2,i}_{k_2,\phi_2}\widetilde{\mathbf{R}}^{1,i}_{k_1,\phi_1}\right)\mathbf{e}_i\right) + \mathbf{e}_-\right)$$

for 3D free-form surfaces.

In a first instance, every contour or surface point is approximated by the action of the twist rotations over the phase vectors. In this case, the contour and surface approximations of equations (3.17) and (3.19) are defined in Euclidean space. Therefore, the approximated contour or surface point is first embedded in projective space by adding the vector $\mathbf{e}_-$ and finally in conformal space by applying the operator $\mathbf{e}\wedge$. Once that the approximated point is defined in conformal space, the motor $\mathbf{M}$ can be applied. Finally, the product of the transformed curve or surface with the corresponding reconstructed optical ray is applied to define the constraint.

3.4.1 Correspondence Search with Model Approximations

For practical applications, correspondences must be found between extracted image features and model information. Once that feature information (point, lines or planes) have been extracted from the image, their corresponding model points must be found in order to construct the constraint equation and to compute the unknown pose parameters. Usually, the iterative closest point (ICP) algorithm is used in this cases to find image-model correspondences, see [13]. For each image point, the closest model point (in terms of the Euclidean distance) is used to form a correspondence pair. If the model has relatively complex structure, it is possible that wrong correspondence pairs affect drastically the computed pose. In the worst of the cases, the minimization algorithm converges to a local minimum. One of the main advantages of the representation of free-form models derived from Fourier or quaternion Fourier descriptors is that low-frequency approximations of contours and surfaces can be obtained. This property is used in order to avoid the local minimum problem up to a certain limit, see [91].

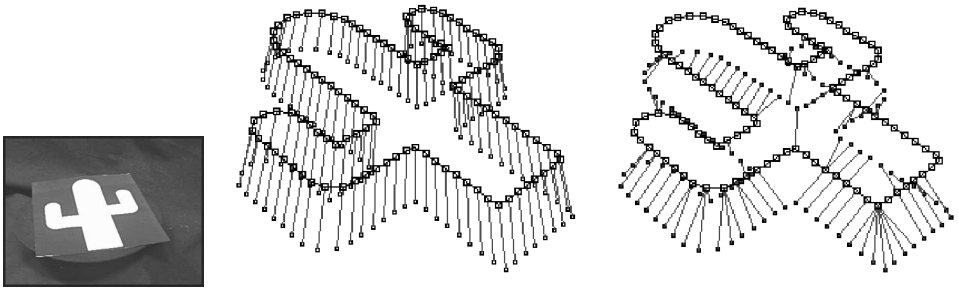

Fig. 3.6: Examples of exact correspondences for the cactus model (left figure) and real correspondences obtained by the ICP algorithm (left picture).

An example of correspondences between a 3D contour model and contour points from the image is shown in figure 3.6. Model and image points are presented separately for a better visualization of the correspondences. An exact correspondence set can be seen in the middle picture. In this ideal example, the algorithm will converge within a few number of iterations to the real pose. However, this may not be the case for real applications. The correspondence set found with the ICP algorithm (minimal distance criterion) is shown in the right picture. As can be seen, many of the correspondences do not correspond to the ideal ones. Therefore, it may take more iterations to the algorithm to converge to the actual pose or it may not converge at all. This fact is discussed theoretically and experimentally in more detail in the next chapters.

An example of the use of low-frequency approximations of the model during the iterative minimization process is shown in figure 3.7. The number of descriptors

used to approximate the model increases with the number of iterations. A rough approximation of the model is used to get correspondences in the first iteration. With this correspondence set, a first approximation of the pose is computed. As more Fourier descriptors are used, the detail level of the model increases. Therefore, the correspondences found in the next iterations are closer to the ideal correspondences and the pose is computed more accurately.

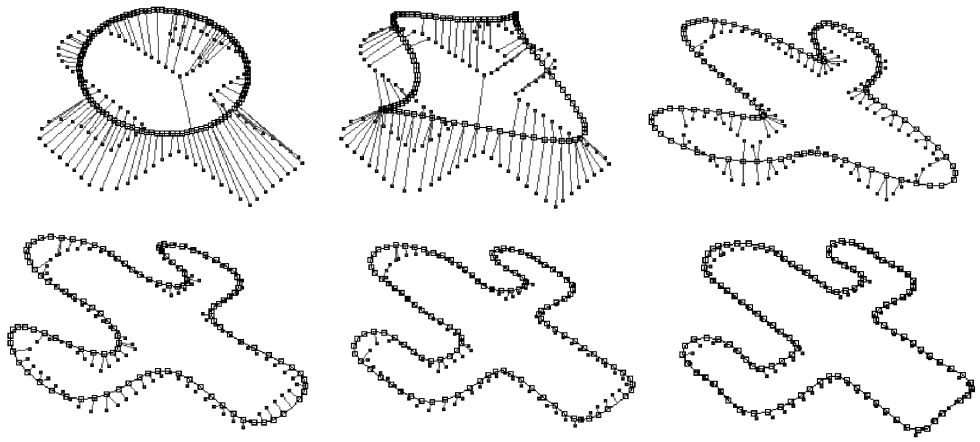

Fig. 3.7: Examples of a convergence sequence by using low-frequency approximations of the cactus model during the iterative process.

As it was mentioned before, this approach is applied to models with a relatively complex structure ass it has been discussed in [91]. When the contour model is defined by line or slight curved segments, the use of such approximations is not necessary. Although the use of low-frequency approximations offers an alternative to avoid the local minimum problem, it also has some drawbacks for practical applications. In a first instance, the computation time increases considerably as more Fourier descriptors are used to approximate the model during the iterations. The use of low-frequency approximations during the first iterations implies a loss of local structure of the model. In that case, no additional information derived from local structure (for example local orientation and curvature) can be used to improve the correspondence search or to decide if a correspondence pair should be rejected as outlier. Then, the correspondence search is limited to the minimal Euclidean distance criterion of the normal ICP algorithm. Because of that, the tracking assumption must be considered in these cases. That mens, the difference between the initial and the actual position of the object must be small enough to ensure that the corresponding closest point is a good correspondence.

The correspondence search problem can be a very complex problem, which has not a general solution for all possible scenarios. Low-frequency approximations of the model may be suitable for certain scenarios and for certain object models. On the other hand, the use of local structure will increase the probability to find

better correspondences as it will be shown in the next chapters. Therefore, the approaches presented in this the next chapters consider model objects without the use of low-frequency approximations. Then, the local structure is preserved during the complete correspondence search process.

3.5 Local Model Representation in Conformal Geometric Algebra

The idea of coupling twist rotations and Fourier descriptors to get a geometrical and mathematical representation of free-form models in CGA was introduced in the last sections. Fourier based twist representations are considered to be global model representations, since the model generator elements (phase vectors) are computed by considering the complete model information.

It is desirable to have a local representation that complements the information obtained from global models. In that sense, a new local model representation of free-form contours and surfaces is presented in this section. In a first instance, this representation describes the local geometry of contours and surfaces by preserving the central idea of a twist model representation in the framework of the CGA. This makes possible to incorporate this local model in the pose estimation constraints as shown in the last sections. Secondly, the local model allows to extract local structural information which will be used to improve the correspondence search problem. The local neighborhood of contour points is described by a set of local motors (a motor operation defines a general rotation around a given axis). The geometry involved in the process of finding the parameters of the local motors is closely related to the concept of osculating circles which is initially introduced. Then, a local motor describing locally a contour point is derived from the parameters of the osculating circle. Once that a set of local motors are obtained for the complete contour, contour segments are described by the action of concatenated local motors over an initial point. Finally, some examples of the description of contour and surface segments are presented.

3.5.1 Osculating Circle

The concept of osculating circle has been used in boundary and contour analysis as a tool to extract local feature information from planar contours, see [31, 53]. The basic idea is shown in figure 3.8. Given a contact point of a curve, a tangent line at this point can be computed. If a circle is constructed in such a way that its tangent is the same as the tangent of the curve at this point, this circle is called the osculating circle of the point. According to that, it is easy to see that both curve and circle have the same orientation and curvature at the contact point. Furthermore, the circle can be considered as an infinitesimal approximation of the curve at the contact point.

By using three points of a curve in 2D, the parameters of the osculating circle,

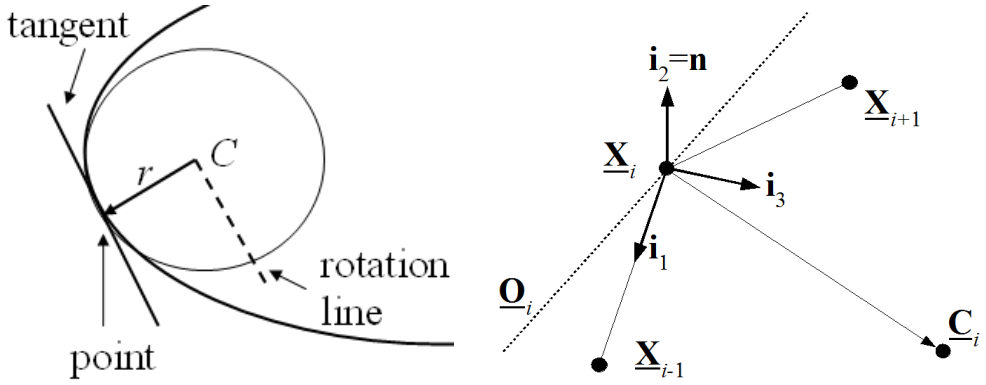

Fig. 3.8: Left: osculating circle of a curve point. Right: definition of the local coordinate system needed to compute the parameters of the circle in 3D.

radius r and coordinates of the center (x, y) can be computed by applying planar geometry. According to the definition of the osculating circle, the curvature $|k(\phi)|$ at the contact point is simply computed as the inverse of the radius $|k(\phi)| = \frac{1}{r}$. According to this, the osculating circle can be interpreted as local descriptor of a curve since it represents its local features.

3.5.2 Construction of the Local Motor

The question is now, how to use the parameters of the osculating circle to obtain the parameters of a twist rotation describing the local vicinity of the contact point. Several possibilities to describe circles in CGA were mentioned in last chapter. A circle can be constructed by the intersection of two spheres or by its dual representation defined by the outer product of three points on the circle. Despite of that, these descriptions do not deliver information about the parameters of the circle (center and radius) and therefore about their local features. On the other hand, these representations describe a circle as a complete entity. It is not possible to describe only a section of the circle. Let us remember that only the vicinity of the point in the osculating circle is part of the contour. Then, only this infinitesimal section must be approximated. Other important property of the local model representation in CGA is the ability to approximate more than a point, a contour segment or the complete contour. This is not possible with the osculating circle or with the equations of the circle in CGA. As it was described in section 3.2, twist rotations can be used as contour and surface generator elements. If a point is rotated by a given angle around an axis, the perimeter of a circle is constructed. The same idea is used to construct the local motor. Instead of approximating the complete perimeter, the motor approximates only the section defined by the contact point and its left and right neighbors.

Let us notice that computing the circle parameters as shown in equation (**??**) is valid only when the curve is defined in 2D. On the other hand, we want to construct

local motors defining curves and surfaces for general 3D curves. Then, the problem is first translated to 2D and then the corresponding motor parameters are recovered in 3D as follows:

- Select three points on the curve, the contact point and its left and right neighbors.
- Construct the plane defined by these points.
- Define a local coordinate system in this plane where the center and the radius of the osculating circle are computed.
- With the normal vector to the plane and the center of the circle define the rotation axis of the local motor back in 3D.

Figure 3.8 shows the scenario for the local motor construction. According to equation (2.50), the local plane related to the contact point $\underline{\mathbf{X}}_i$ is computed by the outer product of the point at infinity $\mathbf{e}$ and the points $\underline{\mathbf{X}}_{i-1}, \underline{\mathbf{X}}_i, \underline{\mathbf{X}}_{i+1} \in \mathcal{G}_{4,1}$

$$\begin{aligned} \mathbf{P}_i^{loc} &= \mathbf{e} \wedge \underline{\mathbf{X}}_{i-1} \wedge \underline{\mathbf{X}}_i \wedge \underline{\mathbf{X}}_{i+1} \\ &= \mathbf{E}\mathbf{n}_i + \mathbf{e}d\mathbf{I}_E \in \mathcal{G}_{4,1}, \end{aligned} \tag{3.34}$$

with $\mathbf{n}_i$ the normal of the plane and d its distance to the origin.

For simplicity, the following operations will be presented in Euclidean space. Once that the parameters of the circle are found, the corresponding motor will be constructed in $\mathcal{G}_{4,1}$. In order to map the points (3D) onto the plane (2D), a local coordinate system must be defined. If the contour points are considered in Euclidean space, $\mathbf{x}_i, \mathbf{x}_{i-1}, \mathbf{x}_{i+1} \in \mathbb{R}^3$, and the origin is fixed at the point $\mathbf{x}_i$, the local basis vectors are given by:

$$\begin{aligned} \mathbf{i}_1 &= \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|} \\ \mathbf{i}_2 &= \mathbf{n}_i \\ \mathbf{i}_3 &= \frac{\mathbf{i}_1 \times \mathbf{i}_2}{\|\mathbf{i}_\mathrm{i} \times \mathbf{i}_2\|}. \end{aligned} \tag{3.35}$$

The process to find the parameters of the circle is presented in figure 3.9. Once that the local coordinate system is defined, it is possible to get the coordinates of the points in the plane defined by the basis vectors $\mathbf{i}_1$ and $\mathbf{i}_3$. Then, the points in the local coordinate system have the following coordinates:

$$\begin{aligned} \mathbf{x}_{i-1}^{loc} &= [-\|\mathbf{x}_i - \mathbf{x}_{i-1}\|, 0, 0] \\ \mathbf{x}_i^{loc} &= [0, 0, 0] \\ \mathbf{x}_{i+1}^{loc} &= [\|\mathbf{x}_{i+1} - \mathbf{x}_i\| \cos\beta, \|\mathbf{x}_{i+1} - \mathbf{x}_i\| \sin\beta, 0]. \end{aligned} \tag{3.36}$$

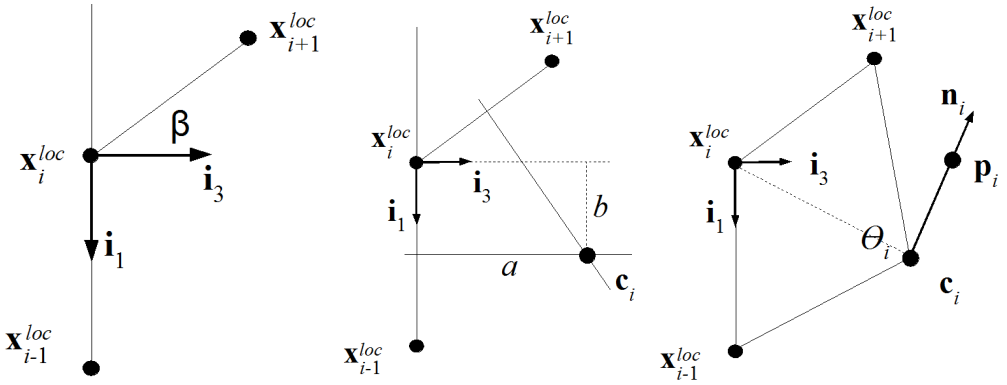

Fig. 3.9: Process to compute the parameters of the circle in the local plane defined by the basis vectors $\mathbf{i}_1$ and $\mathbf{i}_3$.

As can be seen in figure 3.9, the $\mathbf{i}_2$ component of the three points is zero. Therefore, basic planar geometry can be directly applied to find the circle parameters. The problem is reduced to the classical problem of finding the circle that passes trough three points. Basically this is done in two steps. The first step is to find the perpendicular bisectors of the segments defined by $\overline{\mathbf{x}_i^{loc}\mathbf{x}_{i-1}^{loc}}$ and $\overline{\mathbf{x}_i^{loc}\mathbf{x}_{i+1}^{loc}}$ respectively. The second step is to find the intersection point of these bisectors, which corresponds to the center of the circle. Once that the $\mathbf{i}_1, \mathbf{i}_3$ coordinates of the center (b, a) are found, the radius is directly obtained by subtracting the coordinates of the contact point $\mathbf{x}_i^{loc}$. Then, the center and radius of the osculating circle at the point $\mathbf{x}_i^{loc}$ are defined by

$$\begin{aligned} \mathbf{c}_i &= b\mathbf{i}_1 + a\mathbf{i}_3 \\ \mathbf{r}_i &= \mathbf{c}_i - \mathbf{x}_i^{loc}. \end{aligned} \tag{3.37}$$

The osculating circle is still defined in the local coordinate system. It is necessary to find the coordinates of the center $\mathbf{c}_i$ in 3D space. According to the definition of the local coordinate system of equation (3.35), each basis vector $\mathbf{i}$ defines a direction in 3D space. Then, the coordinates of the center in 3D are simply computed by adding $\mathbf{c}_i$ to the point $\mathbf{x}_i$ in its original 3D coordinates as

$$\mathbf{c}_i^{3D} = \mathbf{x}_i + b\mathbf{i}_1 + a\mathbf{i}_3. \tag{3.38}$$

The rotation axis of the motor is obtained directly from center of the circle and any other point in the direction of the normal vector $\mathbf{n}$. If $\underline{\mathbf{C}}_i \in \mathcal{G}_{4,1}$ is the center of the circle embedded in conformal space and $\underline{\mathbf{P}}_i \in \mathcal{G}_{4,1}$ a point in direction of the normal vector, the rotation axis is defined by the line passing through $\underline{\mathbf{C}}_i$ and $\underline{\mathbf{P}}_i$. By applying equation (2.49), the dual representation of a line is obtained as

$$\begin{aligned} \underline{\mathbf{L}}_i^{loc} &= \mathbf{e} \wedge \underline{\mathbf{C}}_i \wedge \underline{\mathbf{P}}_i \\ &= \mathbf{E}\mathbf{r}_i + \mathbf{e}\mathbf{m}_i, \end{aligned} \tag{3.39}$$

with $\mathbf{r}_i$ and $\mathbf{m}_i$ defined as the direction and moment of the line, respectively.

As can be seen in figure 3.9, the angle θ_i is the rotation angle defined by the segment $\overline{\mathbf{x}_{i-1}\mathbf{c}_p\mathbf{x}_{i+1}}$. It was shown in [87] that the parameters of a line correspond to the parameters of a motor performing a general rotation around this line. Then, the local motor is obtained as

$$\begin{aligned} \mathbf{M}_i &= \exp\left(\pm\frac{\theta_i}{4}(\underline{\mathbf{L}}_i^{loc}\mathbf{I}_c)\right) \\ &= \exp\left(\pm\frac{\theta_i}{4}(\mathbf{r}_i' + \mathbf{e}\mathbf{m}_i')\right), \end{aligned} \tag{3.40}$$

with $\mathbf{I}_c = \mathbf{E}\mathbf{I}_E = (\mathbf{e}_+\mathbf{e}_-)\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$. According to the definition of a motor of equation (2.57), a rotation from the point $\mathbf{x}_{i-1}$ to $\mathbf{x}_{i+1}$ is performed with $\frac{\theta}{2}$. Sine a rotation from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$ is needed, the angle $\frac{\theta}{4}$ is used. The next neighbor point in clockwise or counter clockwise direction can be obtained by rotating the contact point $\underline{\mathbf{X}}_i$ by $+\frac{\theta_i}{4}$ or $-\frac{\theta_i}{4}$, respectively. In cases where the points $\underline{\mathbf{X}}_{i-1}$, $\underline{\mathbf{X}}_i$ and $\underline{\mathbf{X}}_{i+1}$ define a straight line segment, the computed motor becomes a pure translation.

3.5.3 Local Motors as Contour and Surface Descriptors

Once that a set of local motors describing a 3D contour is obtained, we proceed to describe how they can be used to approximate segments of contour and surface models. In a first instance, let us consider the case of 3D contours. A contour segment is generated by the action of a set of concatenated motors. In fact, this is analog to kinematic chains used to model robot manipulators. For a given 3D contour of length n, a set of local motors $D_{loc} = \{\mathbf{M}_{\theta_1}, \ldots, \mathbf{M}_{\theta_n}\}$ associated to every contour point is computed. The next neighbor in clockwise direction of a reference point $\underline{\mathbf{X}}_r \in \mathcal{G}_{4,1}$ is approximated as

$$\underline{\mathbf{X}}_{r+1} = \mathbf{M}_{\theta_r}\underline{\mathbf{X}}_r\widetilde{\mathbf{M}}_{\theta_r}. \tag{3.41}$$

Then, all points of a contour segment of length $s \leq n$ are approximated by

$$\underline{\mathbf{X}}_{r+j} = \mathbf{M}_{\theta_j}\cdots\mathbf{M}_{\theta_1}\underline{\mathbf{X}}_r\widetilde{\mathbf{M}}_{\theta_1}\cdots\widetilde{\mathbf{M}}_{\theta_j} \quad : j \in [1, \ldots, s]. \tag{3.42}$$

This local description of contours and surfaces is based on the combination of twist rotations in conformal space. Therefore, the approximated local segments can be directly combined with the pose estimation constraints as shown in section 3.4. Some examples of contour segments approximated by local motors are shown in figure 3.10. In the first example, the corresponding local motors for several contour points are shown. The next figures show the set of local motors used to approximate larger contour segments. In the last row, an example of a contour containing straight lines is shown. In this case, local motors approximate straight segments as a pure translation.

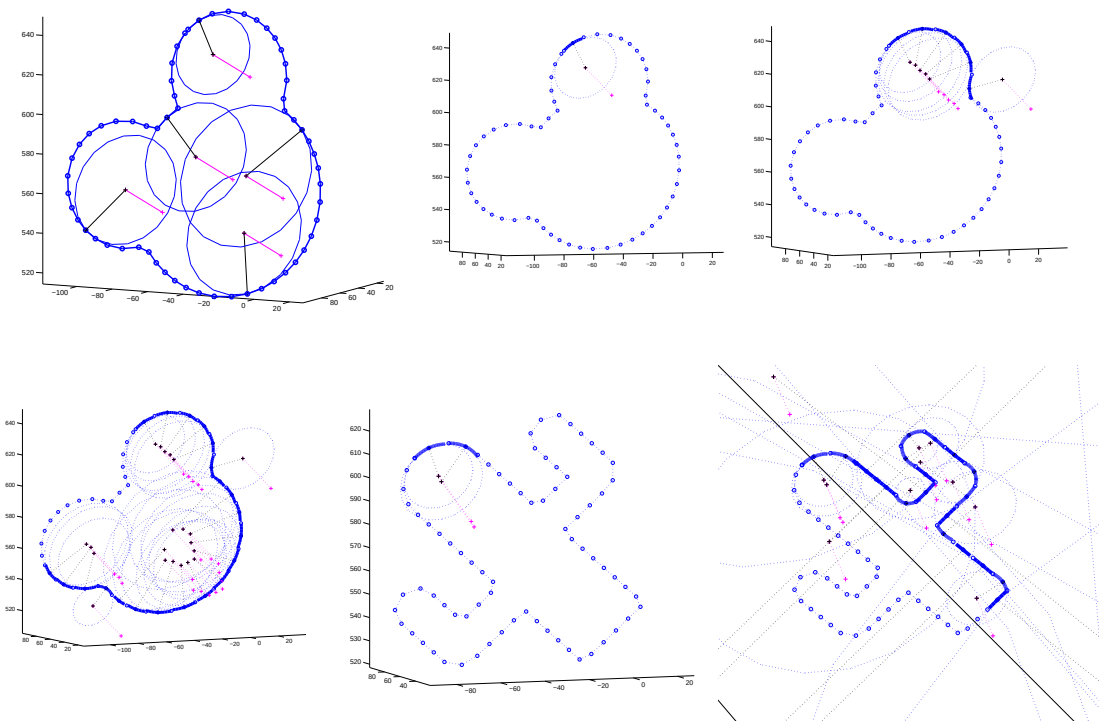

Fig. 3.10: Examples of the local motors which are used to approximate contour segments of the mouse and cactus models.

According to the geometrical construction of the local motors, it is possible to obtain local information describing the contour segments. For every contour point $\underline{\mathbf{X}}_i$, the following local features are directly extracted from its corresponding local motor:

$$F_i^{3D} = \{\underline{\mathbf{O}}_i, \pm\frac{\theta_i}{4}, \|\mathbf{r}_i\|\}, \tag{3.43}$$

where the line $\underline{\mathbf{O}}_i \in \mathcal{G}_{4,1}$ represents the local orientation of the contour segment in 3D. The angle $\pm\frac{\theta_i}{4}$ is the amount of rotation needed to approximate the neighbor points and $\|\mathbf{r}_i\|$ is the radius of curvature. This local information will be used in the next chapters to extract additional structural information. Local structure combined with the Euclidean distance search criterion of the ICP algorithm will be used to improve the correspondence search problem.

Similar to the contour approximation, a segment of a surface is approximated by two local motors. According to the parametric representation of a surface of equation (3.2), a surface point is connected with its four neighbor points. In order to approximate this local neighborhood, two motors with perpendicular rotation axes are constructed as can be seen in figure 3.11. For every surface point, its corresponding local motors are defined as $\{\mathbf{M}_{\theta_1}^{\phi_1}, \mathbf{M}_{\theta_2}^{\phi_2}\}$. Each motor approximates the neighbor

points in the directions defined by ϕ_1 and ϕ_2 respectively. If the point $\underline{\mathbf{X}}_{\phi_1,\phi_2}$ is used as a reference, a neighbor point is approximated by

$$\underline{\mathbf{X}}_{\phi_1+i,\phi_2+j} = \mathbf{M}^{\phi_2}_{j\theta_2}\mathbf{M}^{\phi_1}_{i\theta_1}\underline{\mathbf{X}}_{\phi_1,\phi_2}\widetilde{\mathbf{M}}^{\phi_1}_{i\theta_1}\widetilde{\mathbf{M}}^{\phi_2}_{j\theta_2}, \tag{3.44}$$

where the rotation angles of the local motors are defined by the terms $i\theta_2$ and $j\theta_1$. For example, to approximate the point $\underline{\mathbf{X}}_{\phi_1+1,\phi_2}$ $(i = 1, j = 0)$, only the rotation $\mathbf{M}^{\phi_1}_{i\theta_1}$ in the direction ϕ_1 is applied for this case. The action of both rotations approximates the complete surface patch around the point $\underline{\mathbf{X}}_{\phi_1,\phi_2}$.

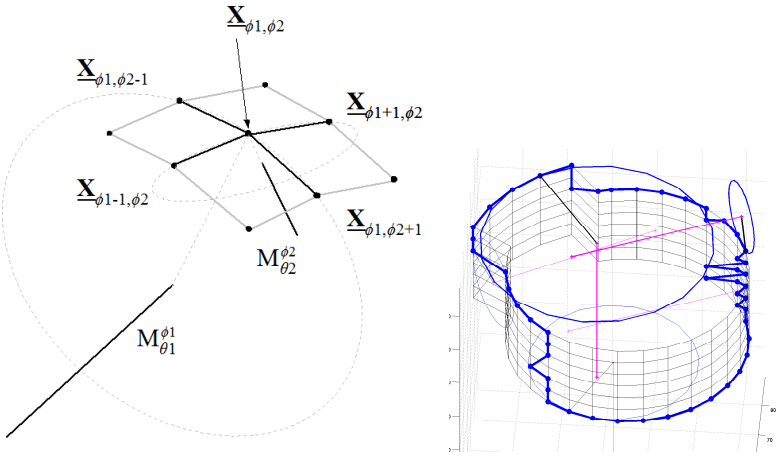

Fig. 3.11: Local motor construction of a point on a surface and its neighbor points (left picture). Example of a 3D silhouette extracted from a surface model and some of its corresponding local motors (right picture).

For some applications, it is convenient to extract 3D contours from surface models. In the context of the pose estimation problem, the 3D contour defining the silhouette of a surface model with respect to the image plane is extracted. The surface pose is computed by aligning its 3D silhouette and the image contour information, see [89]. Than means, the model is reduced from the complete 3D surface to a non-planar 3D contour. According to that, only the local motors and local features of the 3D silhouette must be computed to describe it. An example of an extracted silhouette and some of its local motors is shown in figure 3.11.

It has been shown that the construction of local motors delivers local feature information of 3D contour segments. Similarly, local feature information must be obtained from contours extracted from the image. Then, osculating circles must be obtained from digital contour segments in order to obtain the corresponding local motors. Essentially, the same process discussed in the last section is used to construct local motors for contours in the image plane. As can be seen in figure 3.12, the parameters of the osculating circle are computed from three points in the image. To construct the local motor, the rotation axis is defined by the optical ray passing through the optical center of the camera and the center of the circle.

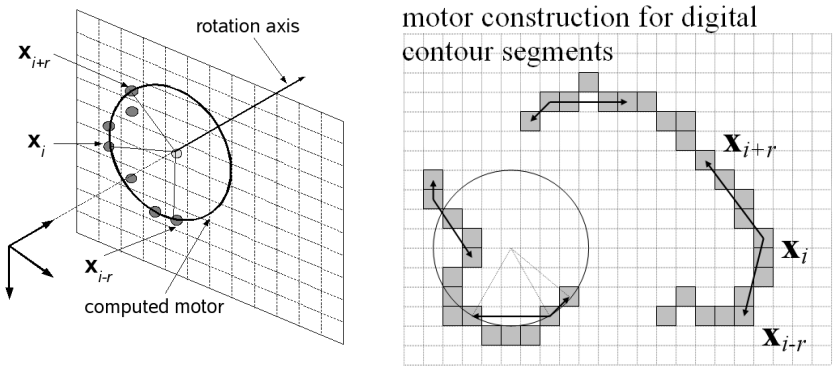

Fig. 3.12: Examples of the construction of local motor from digital contours in the image plane.

3.6 Summary

In this chapter, several mathematical and practical aspects of the twist-based representation of free-form models in CGA were introduced. Complex, quaternion Fourier transforms and Hartley transforms were used to gain global twist representations of free-form surfaces which have different geometrical properties. It was shown that the real valued Hartley transform offers an alternative to the complex valued Fourier transform to obtain model approximations for practical applications. Furthermore, a new local representation of free-form contours and surfaces based on twist rotations was presented. The local vicinity of contour points is approximated by a twist rotation, while larger segments are approximated by concatenated twist rotations over a starting point. Since both model representations were defined in the framework of CGA, they can be combined with the pose estimation constraints. In addition to a mathematical consistent description, it was shown that global and local representations deliver features describing the model objects.

In general, it can not be said that a certain model representation is the best for all possible pose estimation scenarios. For practical applications, the choice of describing a model by global or local model features depends on many factors. When complete model and image data are available and the model displacement is small enough, the use of global features make sense. On the other hand, local features are more suitable when only segments of the image contour information can be extracted because of the presence of occlusions. The question is, how to use global and local models and their derived features to find correspondences between model and image points in order to define the pose estimation constraints. This central part of the pose estimation problem will be discussed in the next chapters.

Chapter 4

STRUCTURAL ICP ALGORITHM

All elements involved in the pose estimation problem have been described in the last chapters. The different pose estimation scenarios based on 3D point-line and 2D projective constraints were described in the framework of geometric algebra. Additionally, global and local representations of free-form contour and surfaces were used to extend the pose estimation constraints to these entities. Since these model representations are able to deliver global and local contour features, it is desirable to use this information to improve the correspondence search problem for practical applications.

In the present chapter, the Structural ICP Algorithm is presented for the case of 3D planar contour models. In a first instance, the classical ICP algorithm is introduced for the pose estimation scenarios described in the last chapters. Then, the main idea of the description levels of model objects in context of the correspondence search problem is discussed. For the image feature extraction step, the monogenic signal response proposed by Felsberg and Sommer [41] is described. This approach allows to extract local structural information of image signals at different scales, which is used in a multi-scale contour extraction algorithm. A local feature extraction approach based on the local contour representation presented in chapter 3 is introduced. The local features computed by this approach are geometrically analog to that of the monogenic signal. Based on these local features, model points are segmented and labeled according to their local structure. Finally, the structural ICP algorithm is presented as a combination of the Euclidian distance search criterion and structural information of image and model.

4.1 ICP Algorithm

The ICP algorithm has been used in a large variety of robotic and computer vision applications and for different pose estimation scenarios. The general concept for any tracking and pose estimation algorithm is illustrated in figure 4.1. As input data, the ICP algorithm requires a set of entities defining the model and a set of data measurements acquired by a sensor device. In a first instance, every sensor measurement must be related to its corresponding model entity by a given corre-

spondence search criterion. The obtained correspondences are the input elements of the minimization constraints used to compute the pose parameters.

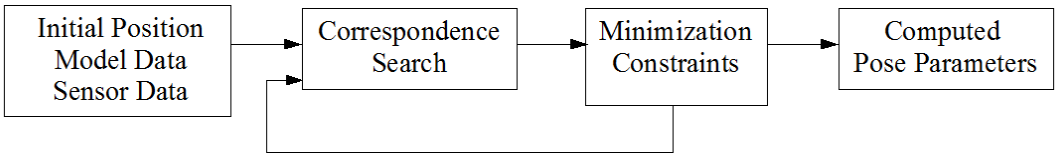

Fig. 4.1: General concept of the ICP algorithm.

The general definition of the ICP algorithm proposed by Besl and McKay [13] is given in this section. Based on this definition, the different elements of the ICP algorithm are identified with the pose estimation scenarios used in this work. Let $\mathcal{M} \in \mathbb{R}^n$ and $\mathcal{S} \in \mathbb{R}^n$ be two sets representing model and sensor elements defined as

$$\begin{aligned} \mathcal{M} &= \{m_i\} \;:\; i = 1 \dots n_{\mathcal{M}} \\ \mathcal{S} &= \{s_i\} \;:\; i = 1 \dots n_{\mathcal{S}}, \end{aligned} \tag{4.1}$$

where $n_{\mathcal{M}}$ and $n_{\mathcal{S}}$ are the size of model and sensor sets respectively. In an ideal pose estimation scenario, two main assumptions are made regarding these sets. The number of elements is the same $n_{\mathcal{M}} = n_{\mathcal{S}}$. Additionally, each element of the set $\mathcal{S}$ corresponds to only one element of the set $\mathcal{M}$. These ideal assumptions are commonly made to analyze the minimization constraints without considering the correspondence search step, see [70].

For real applications, correspondences must be found by a given criterion. In the case of the ICP algorithm, a measure function d is defined to compute the distance between two elements of $\mathcal{M}$ and $\mathcal{S}$ according to a given metric. For a given sensor element $s_j \in \mathcal{S}$, its closest neighbor of the set $\mathcal{M}$ is computed by

$$d(s_j, \mathcal{M}) = \min_{i=1,\dots,n_{\mathcal{M}}} \{d(s_j, m_i)\}, \tag{4.2}$$

where the measure function d is selected depending on the geometric entities that define model and sensor elements. As it was shown in the last chapters, point-point, point-line and line-plane metrics are commonly used in many pose estimation algorithms. For all elements of $\mathcal{S} = \{s_i\}_{i=1}^{n_{\mathcal{M}}}$, their corresponding closest elements are arranged in the set $\mathcal{X} = \{x_i\}_{i=1}^{n_{\mathcal{S}}}$, with $x_i \in \mathcal{M}$. Since the set $\mathcal{S}$ is taken as a reference, the size of both sets is always $n_{\mathcal{S}}$.

In the case of the monocular pose estimation problem, the sensor elements can be defined as points, lines or planes. Let us consider the case that s_i and their corresponding closest elements x_i are defined as points in 3D Euclidean space. In this

case, the unknown pose parameters $\mathbf{R}$ and $\mathbf{t}$ are computed by minimizing the following general equation

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{n_{\mathcal{S}}} \|s_i - (\mathbf{R}(x_i) + \mathbf{t})\|^2. \tag{4.3}$$

The iterative closest point algorithm can be summarized as follows. Given the initial position of the model set $\mathcal{M}_1$ and the acquired sensor elements $\mathcal{S}$, repeat for $j = 1$ to m iterations

Iterative Closest Point Algorithm.

1. For each element of $\mathcal{S}$, find its closest element of $\mathcal{M}_j$ and form the correspondence set $\{\mathcal{S}, \mathcal{X}\}$.
2. With this set, compute the pose parameters $\mathbf{R}, \mathbf{t}$ with the minimization task $E(\mathbf{R}, \mathbf{t})$.
3. With the computed pose parameters, actualize the position of the model set $\mathcal{M}_{j+1}$.
4. Compute the average error between sensor and actualized model elements $E_{mean} = d(\mathcal{S}, \mathcal{M})$.
5. If the error $E_{mean} < thres$ or $j = m$ exit, else goto 1.

As it was mentioned before, the last definition of the ICP algorithm is valid for general pose estimation scenarios. Thus, it is convenient to link the ICP algorithm with the pose estimation scenarios presented in the last chapters. The minimization task $E(\mathbf{R}, \mathbf{t})$ corresponds to one of the minimization constraints in conformal geometric algebra defined in section 2.5.3. To facilitate the reader following the interaction between world and image entities, points in the image plane are denoted by lower case letters $\mathbf{x}$. Elements in Euclidean and conformal space are denoted with capital letters $\mathbf{X}$ and $\underline{\mathbf{X}}$ respectively.

Let us consider the case of finding point-line correspondences in 3D space as shown in figure 4.2. In this case, the model set is defined by $\mathcal{M} = \{\underline{\mathbf{X}}_i\}_{i=1}^n \in \mathcal{G}_{4,1}$ with n model points. From each image point $\mathbf{x}^{img} \in \mathcal{G}_{2,1}$, an optical ray is computed and embedded in conformal space by the operation $\underline{\mathbf{L}}_i = \mathbf{e} \wedge (\mathbf{O}_c \wedge \mathbf{x}^{img})$. Then, the set of sensor elements is defined as $\mathcal{S} = \{\underline{\mathbf{L}}_i\}_{i=1}^n \in \mathcal{G}_{4,1}$.

According to the point-line incidence relation of equation (2.67), the distance between a point and a line is represented as $(\underline{\mathbf{X}} \quad \underline{\times} \quad \underline{\mathbf{L}})$ where $\underline{\times}$ is the commutator product. Then, the closest model point to each reconstructed optical ray is computed by

$$d(\underline{\mathbf{L}}, \mathcal{M}) = \min_{i=1,\ldots,n} \{\underline{\mathbf{L}} \quad \underline{\times} \quad \underline{\mathbf{X}}_i\}. \tag{4.4}$$

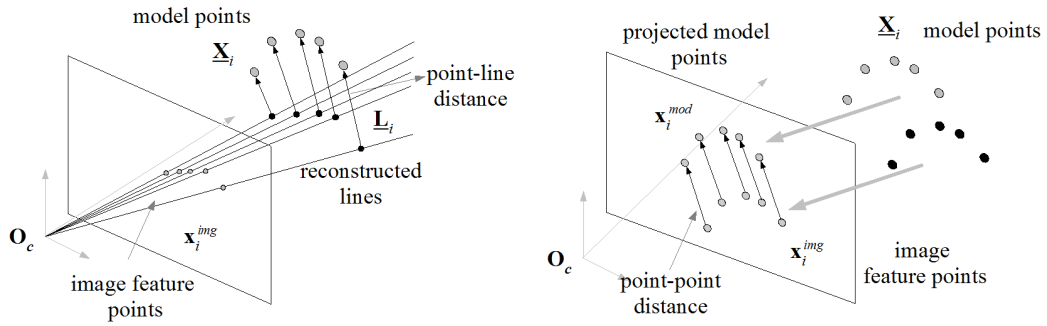

Fig. 4.2: Finding correspondences in 3D space by a point-line distance criterion (left image) and in the image plane by a 2D point-point distance criterion (right image).

Once that the correspondence set $\{\underline{\mathbf{L}}_i, \underline{\mathbf{X}}_i\}_{i=1}^n$ is found, the corresponding n minimization constraint equations are defined as

$$0 = \lambda \left(\left(\mathbf{M} \quad \underline{\mathbf{X}}_i \quad \widetilde{\mathbf{M}} \right) \quad \underline{\times} \quad \underline{\mathbf{L}}_i \right) \cdot \mathbf{e}_+ \quad : \quad i = 1 \ldots n. \tag{4.5}$$

where the motor $\mathbf{M} \in \mathcal{G}_{4,1}$ encodes the pose parameters.

The scenario of the correspondence search problem in the image plane is shown in the right picture of figure 4.2. According to the projective pose constraints defined in equation (2.76), the model points are projected onto the image plane. To simplify the notation, the projection operation of a model point $\underline{\mathbf{X}}_i \in \mathcal{G}_{4,1}$ will be written as $\mathcal{P}(\underline{\mathbf{X}}_i) \in \mathcal{G}_{2,1}$. Then, the model set is defined as $\mathcal{M} = \{\mathcal{P}(\underline{\mathbf{X}}_i)\}_{i=1}^n \in \mathcal{G}_{2,1}$. For each detected image point $\mathbf{x}^{img}$, its closest projected model point is obtained as

$$d(\mathbf{x}^{img}, \mathcal{M}) = \min_{i=1,\ldots,n} \left\{ \|\mathbf{x}^{img} - \mathcal{P}(\underline{\mathbf{X}}_i)\|^2 \right\}. \tag{4.6}$$

Finally, the minimization constraints are defined with the correspondence set of image and projected points $\{\mathbf{x}_i^{img}, \mathcal{P}(\underline{\mathbf{X}}_i)\}_{=1}^n$ as

$$0 = \lambda \left\{ \mathbf{x}_i^{img} \quad - \quad \mathcal{P}(\mathbf{M}\underline{\mathbf{X}}_i\widetilde{\mathbf{M}}) \right\} \quad : \quad i = 1 \ldots n. \tag{4.7}$$

4.1.1 Exact and Well Conditioned Correspondences

Besl and McKay proposed the ICP algorithm for registration of 3D shapes. This was possible since they assumed that the sensor data are approximately aligned with the model data. The last implies that the initial pose is close enough to the global minimum to ensure a proper convergence behavior. In the case of the monocular pose estimation, this is possible if the motion of an object between two images is

small enough (tracking assumption condition). This occurs if the images are taken in a short time period or when the object moves sufficiently slow. Once that the definition of the ICP has been given, it is convenient to make some general remarks about the tracking assumption and the correspondence search problem when the closest distance criterion is used.

Any variant of pose estimation algorithm will deliver a correct pose if an exact set correspondences set is used. An example of exact correspondences for 2D objects can be seen in figure 4.3. In the upper left example, the arrows show the exact correspondences of points of the square. The upper right figure shows an example of exact correspondences for a 2D contour. When the correspondence set is correct, only a few iterations are needed for the algorithm to converge to the correct pose.

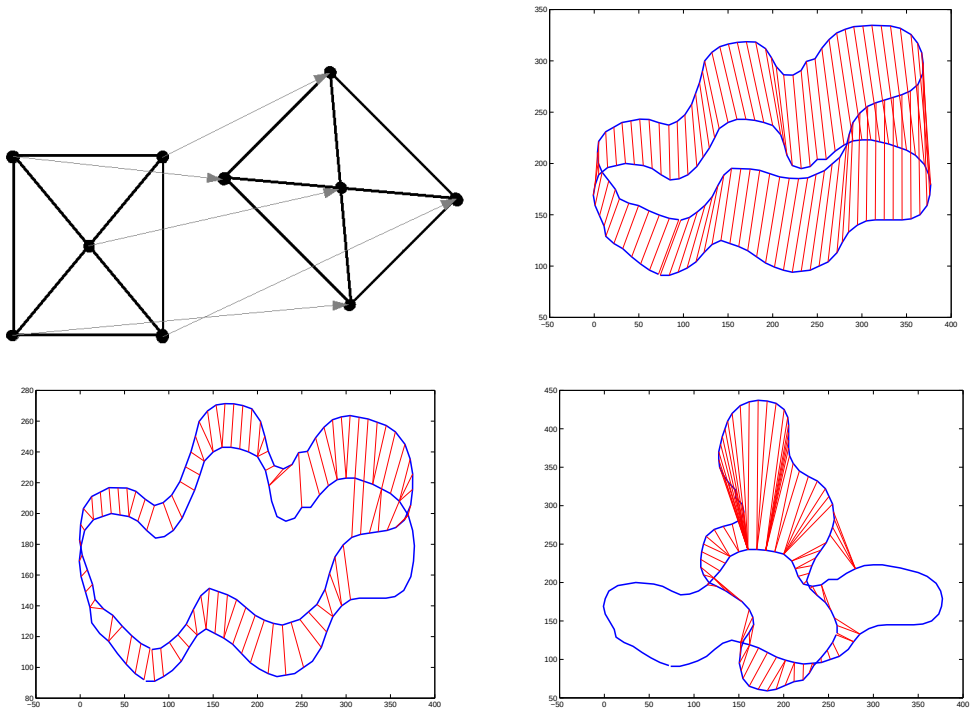

Fig. 4.3: Upper row: examples of exact correspondence sets of 2D objects. Lower row: example of well (left) and bad (right) conditioned correspondences for a 2D contour.

Ideally, a correspondence search algorithm should find correspondences close to the exact ones. For each possible pose estimation scenario, the correspondence search problem can be well conditioned or not. In general, the complexity of models and the initial pose are the main factors that determine if a correspondence set is well conditioned. Figure 4.3 shows several examples of correspondences for 2D models. The lower left picture shows an example of well conditioned correspon-

dences. For most of the contour points, their initial correspondences are relatively close to the exact ones. Then, the correspondence set is considered to be well conditioned. Because of that, the computed pose with this correspondence set will be also well conditioned for the next iterations. This has the effect that a larger number of well correspondences are found in the following iterations. Eventually, the correct pose will be computed for this scenario. An example of extreme bad-conditioned correspondences is shown in the lower right picture of figure 4.3. If these correspondences are used, there is a high probability that the computed pose would be bad conditioned for the next iterations. The algorithm may need a relatively large number of iterations to reach the correct pose. In the worst of the cases, it may converge to a local minimum.

4.2 Description Levels of Contour Models

The examples presented in the last section consider only the Euclidean distance criterion to find correspondences between contour points. In order to improve the correspondence search problem, more information about the nature and structure of models elements and image data must be considered. Then, the general idea of describing contour models at different levels based on their local feature information is presented in this section. In this context, the level of description refers to the size of a segment around a contour point needed to define its features.

One important question regarding contour features is how ”local” or ”global” they should be for a given task, e.g. for the correspondence search problem. The general idea is shown in figure 4.4. The inverted triangle represents the description levels of a point according to its features. On the other hand, the triangle of the right represents the amount of possible correspondence candidates of a contour point at each description level. Let us suppose the case of an ideal system with the ability to get ”all” possible information from image and model points. In this case, an image point is represented at the highest possible level of description as shown in the figure. Since all information of the point is known, the only correspondence candidate is the exact corresponding model point. Notice that such an ideal scenario is one of the major challenges for computer and robot vision applications.

The lower vertex of the left triangle represents the lowest level of description of a point. In this case, it is described only by its position in 3D or 2D Euclidean space. If a scene point is described at this lowest level, all model points are possible correspondence candidates. Then, the most suitable point is selected by applying a specific criterion (e.g. minimal Euclidean distance for the ICP algorithm). As the description level increases, more feature information is used to describe the point. Since more information about the point is available, a correspondence search criterion based on that information can be used. The possible correspondence candidates of an image point are all model points described by the same feature information. Therefore, the amount of possible correspondence candidates decreases

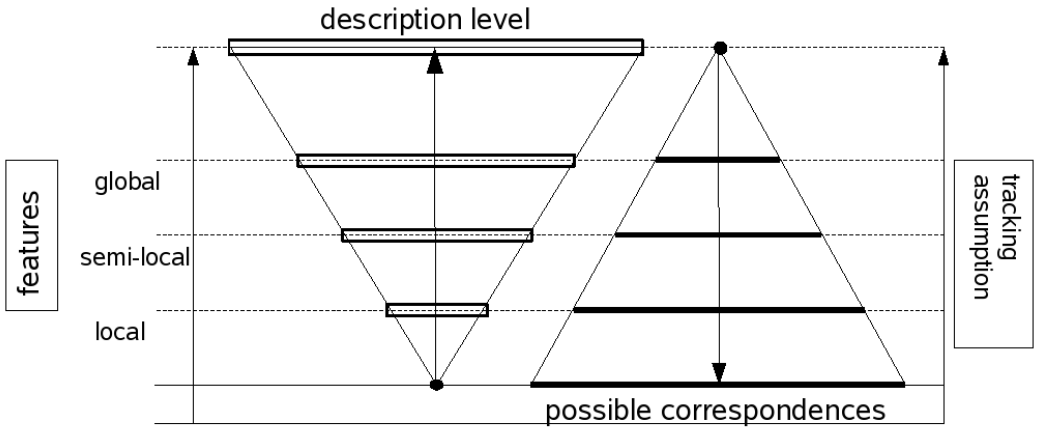

Fig. 4.4: Relationship between the description level of contour points according to their feature information and their possible correspondence candidates.

and the probability to find the correct correspondences increases. At the same time, the dependence of the correspondence search criterion on the tracking assumption decreases.

Let us consider the example of figure 4.5 for the case of contour models. The basic geometric entity which can be described is a point. From this point, only its 3D or 2D position is available. Thus, a single point does not deliver more information about its structure. Instead, according to the local representation of free-form contours introduced in section 3.5.3, contour segments can be approximated by a set of concatenated local motors. Furthermore, local orientation and curvature is directly obtained from this representation.

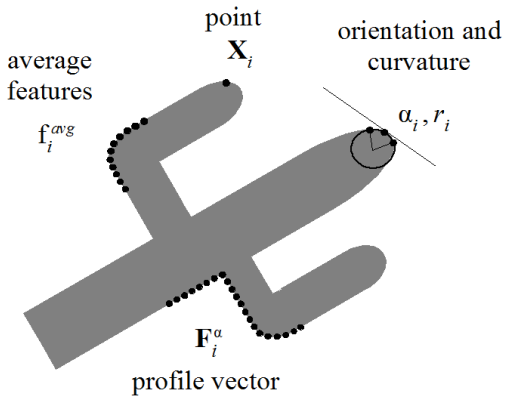

Fig. 4.5: Feature description levels of contour points.

The notation used to refer the description of contour points at different levels is

introduced. In a first instance, the set of local features obtained from the vicinity of a contour point is denoted as

$$\mathbf{f}_i^{loc} = \{\mathbf{X}_i, \alpha_i, k_i\}, \tag{4.8}$$

where $\mathbf{X}_i, \alpha_i$ and k_i stand for the 3D or 2D spatial position of the point, orientation and curvature respectively.

At the next description level, a larger contour segment around a given point is considered to compute its features. As can be seen in figure 4.5, the local vicinity of a point is extended in order to compute "semi-local" features. The term semi-local is used since these features are a combination of the local features obtained within the segment. For a segment of length n around a contour point x_i, the semi-local features are defined by

$$\mathbf{f}_i^{avg} = \left\{ \frac{1}{n}\sum_{i=1}^{n} \alpha_i, \frac{1}{n}\sum_{i=1}^{n} k_i \right\}, \tag{4.9}$$

where, they represent the average orientation and curvature around a contour point.

Other possibility is to use the local features to define "profile" feature vectors. According to that, the orientation and curvature profile vectors for a contour point $\mathbf{X}_i$ are defined as

$$\mathbf{F}_i^{\alpha} = \{\alpha_{i-n} \ldots \alpha_i \ldots \alpha_{i+n}\} \tag{4.10}$$

$$\mathbf{F}_i^{k} = \{k_{i-n} \ldots k_i \ldots k_{i+n}\}. \tag{4.11}$$

As can be seen in figure 4.5, profile vectors can be used to describe contour segments of different lengths. That means, the description level of a contour point can be extended from the local vicinity of a model point up to the complete contour length.

4.3 Image Feature Extraction

So far the ICP algorithm in context of the monocular pose estimation problem has been introduced. The object models used for the pose estimation are represented by free-form contours with their corresponding local features. Therefore, it is necessary to detect local feature information of contours in the image. To achieve that, image analysis approaches based on local signal representations can be used [48]. The main idea of such approaches is to assign structural and geometrical information around an image point. One of this approaches is the monogenic signal. From the monogenic signal response, features describing the local structure of the image signal are obtained (local amplitude and phase) as well as geometric feature as local orientation. The foundations of the monogenic signal can be consulted in [39, 41, 104].

In this section, a brief introduction to the concepts of amplitude and phase of 1D signals is given. An extension of this concepts to 2D signals is used to define the monogenic signal. If the monogenic signal is defined at different scale values, the monogenic scale space is obtained. Then, an approach to detect image edge information at different scales known as "phase congruency" is presented. Finally, contour segments are extracted from the image by a multi-scale contour extraction algorithm. The phase concept plays an important role for the definition of the structural ICP algorithm. As it will be shown later, it will be used to define additional correspondence search constraints.

In order to clarify the concepts of amplitude and phase, lets us consider the definition of the analytic signal for 1D signals $f : \mathbb{R} \longrightarrow \mathbb{R}$, see [43]. The analytic signal $f_A : \mathbb{R} \longrightarrow \mathbb{C}$ is defined as the combination of the original signal with its Hilbert transform as follows

$$f_A(x) = f(x) + i\left(\frac{1}{\pi x} * f(x)\right) = f(x) + if_H(x), \tag{4.12}$$

where $\frac{1}{\pi x}$ refers to the Hilbert convolution kernel. The components $f(x)$ and $f_H(x)$ are phase shifted by $-\frac{\pi}{2}$, that means they are in quadrature phase relation, see [39]. In fact, $f_A(x)$ is a complex valued function which can be rewritten by the Euler formula as the exponential function $f_A = A(x)\exp\{p(x)i\}$, with the local amplitude and phase defined respectively by

$$A(x) = \sqrt{f^2(x) + f_H^2(x)} \tag{4.13}$$

$$p(x) = \arg(f_A(x)). \tag{4.14}$$

By means of the analytic signal, the local structure of a point is characterized by the phase response. The local phase provides information about the kind of structure contained in the signal. An example of typical phase responses for a 1D signal is shown in figure 4.6. If the local energy is zero, it is not possible to make a phase analysis. In the case that its value exceeds a given threshold, the local phase delivers information about the local symmetry of the signal. A peak corresponds to the phase value $p(x) = 0$ and a pit to $p(x) = \pi$. Descending or increasing slopes have the phase responses $p(x) = \pi/2$ and $p(x) = -\pi/2$ respectively. Peak and Pit indicate even symmetry, while slopes indicate odd symmetry.

4.3.1 Monogenic Signal

The monogenic signal is a vector function composed by three components derived from the extension of the analytical signal for the multidimensional case. For a 2D image signal $\mathbf{f}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$, the monogenic signal is defined as the combination of the original signal with its Riesz transform (composed by an x-component and an y-component)

$$\mathbf{f}_M = \mathbf{f}(\mathbf{x}) + (h_2 * \mathbf{f})(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{f}_R(\mathbf{x}), \tag{4.15}$$

where $h_2 = \frac{\mathbf{x}}{2\pi|\mathbf{x}|^3}$ is the Riesz convolution kernel.

The real part $\mathbf{f}(\mathbf{x})$ is known as the even component of the monogenic signal, while the Riesz transform $\mathbf{f}_R(\mathbf{x})$, known as image flow, is considered the odd component. Similar to the analytic signal, even and odd parts are in quadrature relation. Thus, amplitude and phase are similarly obtained as shown in equation 4.13.

In practice, Poisson and conjugate Poisson filter kernels are used as can be seen in [41]. If these Poisson kernels are applied to the original image signal $\mathbf{f}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$, the Poisson scale-space and its harmonic conjugate Poisson scale-space are formed as

$$p(\mathbf{x}; s) = (\mathbf{f} * P)(\mathbf{x}) \quad \text{with} \quad P(\mathbf{x}) = \frac{s}{2\pi(|\mathbf{x}|^2 + \mathrm{s}^2)^{3/2}} \tag{4.16}$$

$$\mathbf{q}(\mathbf{x}; s) = (\mathbf{f} * Q)(\mathbf{x}) \quad \text{with} \quad Q(\mathbf{x}) = \frac{\mathbf{x}}{2\pi(|\mathbf{x}|^2 + \mathrm{s}^2)^{3/2}}, \tag{4.17}$$

where s denotes the scale value and $P(\mathbf{x})$ and $Q(\mathbf{x})$ the Poisson and conjugate Poisson kernels respectively.

The monogenic scale space is formed by combining the Poisson scale-space with its harmonic conjugate scale-space at all scale values s as shown in figure 4.6 [1] . If the scale parameter is set to zero, the conjugate Poisson kernel corresponds to the Riesz kernel. That means, the monogenic signal is obtained at the lowest scale value. In fact, the monogenic scale-space can be interpreted as the combination of the monogenic signals at all scales, where the monogenic signals are formed by the smoothed original signals $p(\mathbf{x}; s)$ and the image flow $\mathbf{q}(\mathbf{x}; s)$. One important property of the monogenic scale-space is that the smoothed signal and the image flow preserve the quadrature phase relation for all scale values.

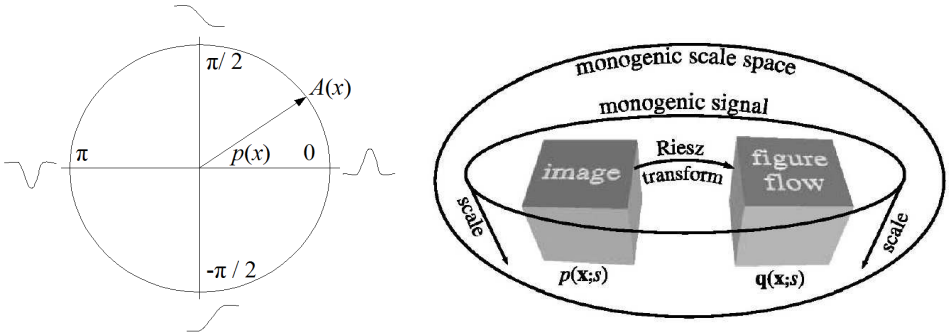

Fig. 4.6: Typical phase responses for a 1D signal (left figure). Monogenic scale space defined by the combination of Poisson and conjugate Poisson filter responses (right figure).

[1] Image taken from [41].

From the filter responses of equations (4.16) and (4.17), features describing the local structure are computed in terms of the local amplitude and phase as follows

$$A(\mathbf{x}; s) = \sqrt{|\mathbf{q}(\mathbf{x}; s)|^2 + |p(\mathbf{x}; s)|^2} \tag{4.18}$$

$$\mathbf{r}(\mathbf{x}; s) = \frac{\mathbf{q}(\mathbf{x}; s)}{|\mathbf{q}(\mathbf{x}; s)|} \arctan\left(\frac{|\mathbf{q}(\mathbf{x}; s)|}{p(\mathbf{x}; s)}\right). \tag{4.19}$$

The amplitude $A(\mathbf{x}; s)$ is related to the local energy of the signal. That means, high amplitude responses indicate the presence of local structure. Phase and orientation information are combined in the phase vector $\mathbf{r}(\mathbf{x}; s)$. As described before, the phase describes the local symmetry of the 2D image signal. For instance, edges in the image are considered to have odd local symmetry, while lines have even local symmetry. The orientation information corresponds to direction of the highest signal variance. An example of the phase and orientation responses for a real image is

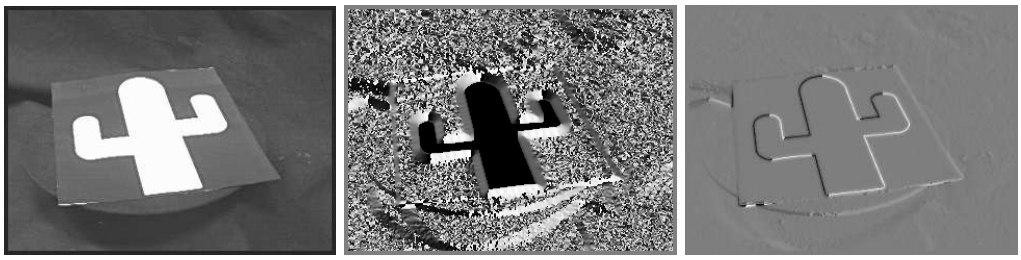

Fig. 4.7: Example of the monogenic signal response of an image. Original image (left), local orientation (center) and phase (right).

shown in figure 4.7. In the right image, the phase response can be seen. Note that the minimal and maximal phase responses correspond to edge points of the original image. As it was mentioned before, an edge point has locally odd symmetry with phase values $\pi/2$ and $-\pi/2$ respectively. This can be interpreted as a change from a high to a low gray value or viceversa in the image signal. Thus, the local phase vector characterizes the local gray level transition of an image signal. Finally, the orientation response is shown in the center image of the figure.

The local features of an edge point obtained from the monogenic scale-space at a given scale value s are summarized as

$$F_i^{im} = \{x^{im}, y^{im}, \alpha_i^{im}, \| r_i^x \|, \| r_i^y \|\}, \tag{4.20}$$

where x^{im}, y^{im} are its Euclidean coordinates, α_i^{im} is the local orientation and $\| r_i^x \|, \| r_i^y \|$ are the phase responses in x and y directions. As described in section 4.3.1, the monogenic signal is defined by an x- component and an y-component. Therefore, two phase responses are obtained in the direction of these components.

4.3.2 Phase Congruency for Edge Detection

Scale-space techniques have been widely investigated and used for image processing applications. In a similar way that a linear scale-space is constructed with Gaussian kernels, the linear scale-space based on Poisson kernels presented in [41] can be used for edge detection and image restoration applications. Edge detection by applying quadrature filters can be performed in two different ways: either by detecting local maxima of the amplitude information or by detecting points of stationary phase in scale-space. The first option is convenient for cases where the image has relatively high contrasted edges, like the example of figure 4.7. The second approach is called "phase congruency". Roughly speaking, this method is based on comparisons of the local phase at certain different scales.

For the local phase vector $\mathbf{r}(\mathbf{x};\mathbf{s})$ of the monogenic scale-space representation of equation (4.19), points that satisfy the condition

$$\partial_s \mathbf{r}(\mathbf{x};s) = 0 \tag{4.21}$$

are known as points of differential phase congruency. Those points correspond to image edges, see [41].

The question is how to approximate this expression in suitable way for practical applications. One possibility may be to compute the derivative by applying classical approximations based on finite differences. Due to the unification of scale-space and phase-based approaches in the context of the monogenic scale-space, it was proved in [41] that the scale derivative can be directly computed as

$$\partial_s \mathbf{r}(\mathbf{x};\mathbf{s}) = \frac{p(\mathbf{x};s)\partial_s \mathbf{q}(\mathbf{x};\mathbf{s}) - \mathbf{q}(\mathbf{x};\mathbf{s})\partial_{\mathbf{s}} p(\mathbf{x};s)}{|p(\mathbf{x};s)|^2 + |\mathbf{q}(\mathbf{x};\mathbf{s})|^2}. \tag{4.22}$$

In comparison with a finite differences approach, this method allows a higher accuracy and a faster computation of the derivative. In order to find the points of phase congruency, the solution of equation (4.21) must be found. In this case, it is only necessary to find the zeros of the two components of the numerator in equation (4.22). Then, the expression to solve becomes

$$p(\mathbf{x};s)\partial_s \mathbf{q}(\mathbf{x};\mathbf{s}) - \mathbf{q}(\mathbf{x};\mathbf{s})\partial_{\mathbf{s}} p(\mathbf{x};s) = 0. \tag{4.23}$$

Once that the derivatives are computed, the zeros of last equation can be detected by applying a linear regression approach. Finally, the edges at a given scale are detected by removing those zeros where the slope of last equation is smaller than a given threshold value. An example of a real image and the corresponding detected edges at different scales is shown in figure 4.8. Since only high contrasted edges are detected at high scale values, several segments of the object are not detected. Those segments appear gradually as the scale decreases. At the lowest scales, almost all the object edges are detected as well as many noisy (in the sense that they do not belong to the object) edges caused by non-uniform illumination.

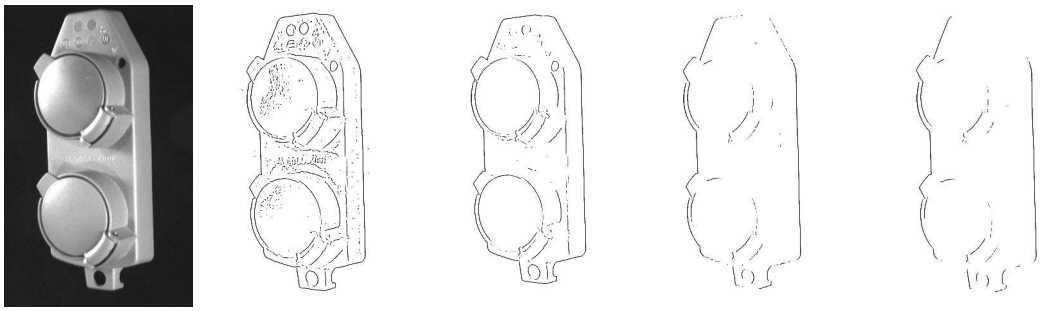

Fig. 4.8: Detected edges of an image model at different scales.

4.3.3 Multi-Scale Contour Extraction

Once that edges are detected in the image, contour search algorithms are usually applied to extract contour segments of different sizes. Let us consider the case of the image shown in figure 4.8 and its corresponding detected edges at different scales. All edges of the object including noisy edges are detected at the lowest scale. In contrast to that, only high contrasted edges are detected at higher scales. Then, the choice of a given scale value where contour segments of the object can be extracted without the presence of noisy edges is not easy. Instead of selecting a given scale to extract the edge segments, the detected edges are analyzed at different scales. The

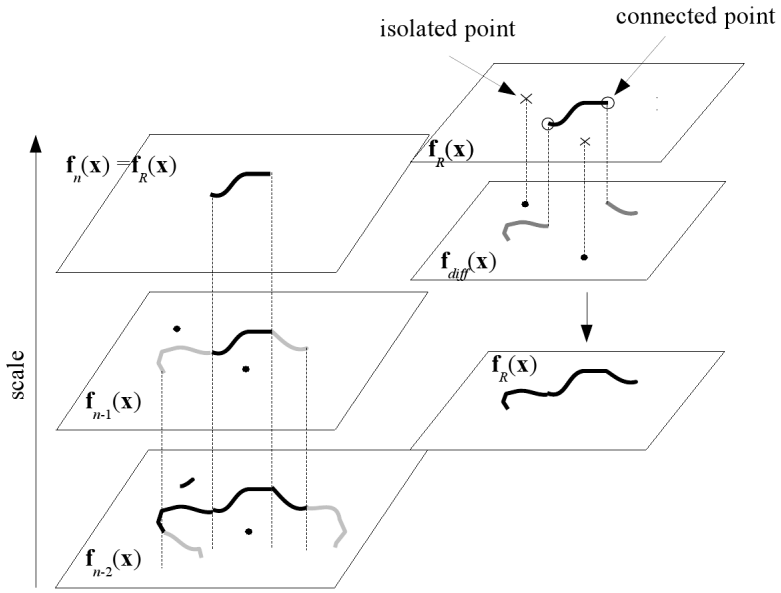

Fig. 4.9: Idea of finding the image edges by taking the highest scale edges as a reference.

set of images containing the detected edges at n scale values is defined as

$$I = \{\mathbf{f}_1(\mathbf{x}), \ldots, \mathbf{f}_n(\mathbf{x})\}\,. \tag{4.24}$$

The idea of the edge extraction algorithm at different scales is shown in figure 4.9. The image $\mathbf{f}_n(\mathbf{x})$ containing the detected edges at the highest scale value is used as a reference, $\mathbf{f}_R(\mathbf{x}) = \mathbf{f}_n(\mathbf{x})$. In order to find the edges that appear at the next lower scale $n-1$, the image difference is computed as

$$\mathbf{f}_{diff}(\mathbf{x}) = \mathbf{f}_{n-1}(\mathbf{x}) - \mathbf{f}_n(\mathbf{x}). \tag{4.25}$$

With this substraction operation, the common edges of both images are removed. Therefore, only the edges at the lower scale are present in $\mathbf{f}_{diff}(\mathbf{x})$. The next step is to determine if the points in the difference image are connected with the points of the reference image. For every point in $\mathbf{f}_{diff}(\mathbf{x})$, a neighborhood of a given range around this point is analyzed in the reference image $\mathbf{f}_R(\mathbf{x})$. If more than one point is found, it is considered to be connected to the previous detected edges and it is added to the reference image. If no pixel is found, the edge is considered to be isolated. This procedure is iteratively repeated for all the images at all scales.

The result for the image of figure 4.8 can be seen in figure 4.10. For this example, a set of images containing the detected edges at ten different scale levels were used. The image of the left is the reference image $\mathbf{f}_R(\mathbf{x})$. As the following images are processed, the most representative edges of the object are gradually extracted and most of the noisy edges of the lower scales are eliminated. The picture of the right shows the result after processing all images, where the outer contour of the object and partial information of the inner contours are obtained.

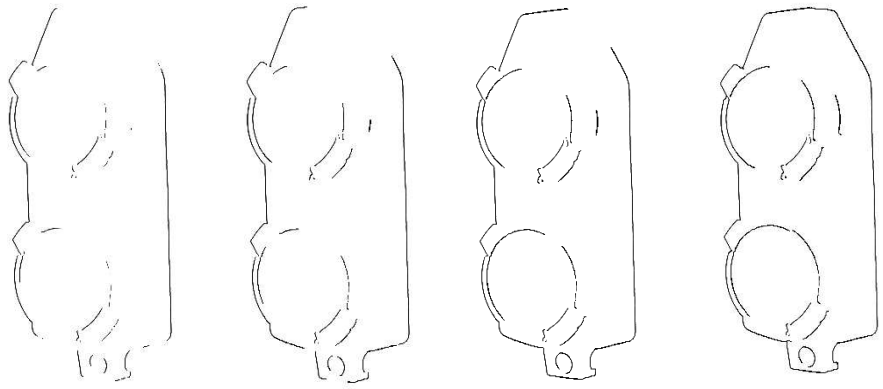

Fig. 4.10: Result of the multi-scale edge detection algorithm.

A proper extraction of the outer contour is essential for the new variants of the ICP algorithm presented in the next chapters, since it allows the computation of

global features in the image plane (see section 3.3.4). On the other hand, the inner contour information can be used if the inner contours are also represented in the object model, see [87].

As described in this section, the monogenic scale space approaches are only used as a tool to extract contour information from the images. Hence, the concept of scale-space is not directly used in the correspondence search strategies presented in this and the next chapters.

4.4 Model Feature Compatibility

According to the pose estimation scenarios described in section 4.1, the correspondence search problem can be defined in 3D space or in the image plane. In order to use additional feature information in the correspondence search problem, compatible model and image features must be obtained. The main question is, in which space it is possible to find better compatible features. A first option is to extract 3D features from image information and to compare them with model 3D features. If we consider that only one image is used, the reconstruction of 3D features from image information is restricted to lines and planes. The curvature of a 3D contour segment can not be reconstructed from the curvature of image segments. On the other hand, the concept of phase obtained from the monogenic signal is only valid for image signals. The second possibility is to project the 3D model onto the image plane to find 2D features in order to compare them with the image features. In the case of 3D contour models, local orientation and curvature are obtained from the local model representation. This information is not invariant under perspective projection. Thus, it can not be directly used as correspondence search criterion as proposed in [100]. Despite of that, features derived from orientation and curvature of projected contour segments are used to improve the correspondence search problem.

In this section, it is shown that the projection of model contour segments onto the image plane allows to find compatible local features to that of the image plane. In a first instance, an analog structural feature to the monogenic phase response called ”transition index” is presented. It defines an artificially generated grey level transition similar to the monogenic phase. Then, the local curvature of image and contour models is used to compute semi-local structure. If a neighborhood around a given point is considered, it is decided if that point belongs to a straight, concave or convex segment. That means, contour points are segmented according to their structure. Finally, several examples of this segmentation process are presented.

4.4.1 Transition Index

It is necessary to get compatible features between the monogenic phase and local model features in the image plane. The most natural solution would be to project the object model onto the image plane and to compute directly the monogenic signal response. Let us remember that the monogenic phase response encodes structural information corresponding to the local symmetry of the image signal. On the other hand, the local representation of contour models presented in section 3.5 and its corresponding local features are derived from sets of discrete points. Thus, the corresponding projected model points are also sets of discrete image points. This implies that the local structure of the projected points can not be obtained from the monogenic signal, since a single image point does not have local symmetry at all.

A direct computation of the monogenic signal would be only possible if an artificial image is generated for each iteration of pose estimation algorithm. In practice, this will increase significatively the computation time. Instead of that, an analog feature to the phase response can be easily derived from the local orientation of model points. Let us remember that phase and orientation are encoded in the phase vector, see equation (4.19). That means, the local orientation can be obtained for points whose phase response correspond to edges. In the case of contour models, only the local orientation is directly computed from the local contour representation. Based on the local orientation, the transition index is defined as an artificially generated transition value of a projected contour segment. It simulates the gray level transition of the image signal encoded in the monogenic phase response.

In order to define the transition index, the following assumptions are made with respect to the 3D contour models and their local features:

- The orientation computed from three contour points in 3D space is preserved after the projection of these points onto the image plane. That means, the projected model points must allow to compute local orientation in the image plane. If two or more 3D points of a segment are projected onto the same image point, it is not possible to compute the local 2D orientation. Therefore, only planar 3D contour models are considered for this initial approach. An extreme case occurs when the planar contour is perpendicular to the image plane. Then, all points are projected to a line onto the image and no feature extraction is possible.

- In the case of an object modeled by a 3D free-form contour, every point represents implicity an edge of the real object. The phase response encodes information about the local structure of edges (even or odd symmetry) that depends on the gray value level of the real object and background. Then, the problem is reduced to identify to which kind of edge the model points correspond (increasing or decreasing slope as shown in figure 4.6). To achieve that, the gray level of the model object and background are considered to be known.

Let us remember the local feature set obtained from the local representation of 3D contours introduced in section 3.5. As can be seen in figure 4.11, the following local features are obtained from a 3D model point $\underline{\mathbf{X}}_i \in \mathcal{G}_{4,1}$

$$F_i^{3D} = \{\underline{\mathbf{O}}_i, \pm\frac{\theta_i}{2}, \|\mathbf{r}_i\|\}, \tag{4.26}$$

where $\underline{\mathbf{O}}_i \in \mathcal{G}_{4,1}$ is the local orientation line. It represents the orientation of the contour segment in the 3D coordinate system as the tangent line of the point $\underline{\mathbf{X}}_i$. The rotation angle θ_i describes the amount of rotation needed to approximate this segment and $\|\mathbf{r}_i\|$ is the radius of curvature.

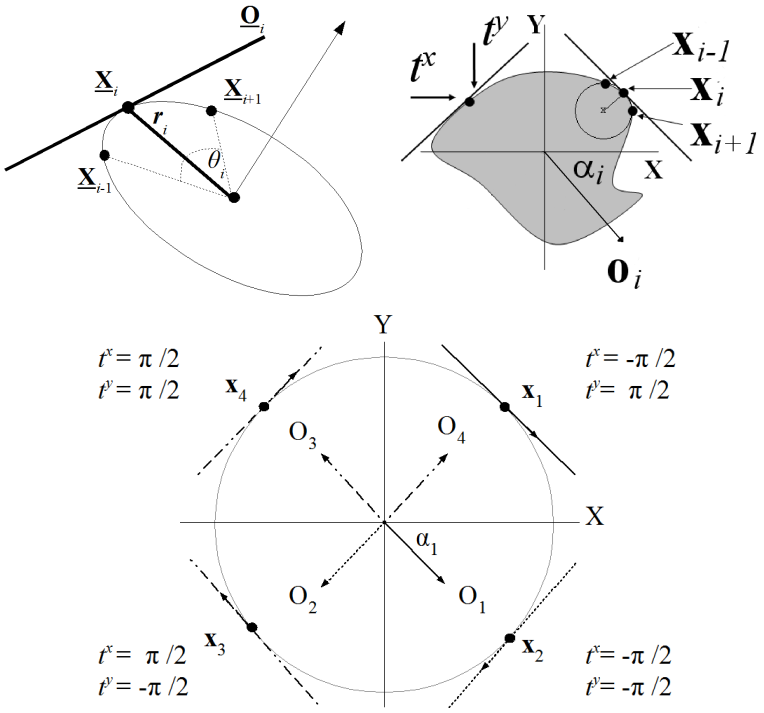

Fig. 4.11: Upper row: features obtained from the local motor in 3D (left figure) and its corresponding local features in the image plane (right figure). Lower row: example of the assignation of the transition indexes for different image points with respect to their local orientation.

Once that the 3D model points are projected onto the image plane, the corresponding local motor is computed with the points $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1} \in \mathcal{G}_{2,1}$. The last is also shown in the upper right picture of figure 4.11. From the local motor construction in the image plane, the orientation line and radius of curvature are also obtained. As mentioned before, only the orientation information is needed to define the transition index. From the orientation line, the normalized orientation vector $\mathbf{O}_i$ with orientation angle α_i is obtained.

The upper right picture of figure 4.11 shows the principle to assign the transition values t^x and t^y. For the example presented in the figure, the model corresponds to a dark object in a white background. Then, the gray value transition t^x for the point $\mathbf{x}_i$ goes from black to white in the x direction. If black corresponds to a grey value of 0 and white to 255, this can be interpreted as an increasing slope. In the y direction, the transition goes from white to black corresponding to a decreasing slope. The bottom figure shows examples of the transition values in both directions for four points $\mathbf{x}_1, \ldots, \mathbf{x}_4$. Depending on the orientation α_i, the transition values are assigned according to the following criterion

$$t_i^x, t_i^y \Rightarrow \begin{cases} t_i^x = +\frac{\pi}{2} & t_i^y = +\frac{\pi}{2} & : & \alpha_i \in [0, \frac{\pi}{2}) \\ t_i^x = +\frac{\pi}{2} & t_i^y = -\frac{\pi}{2} & : & \alpha_i \in [\frac{\pi}{2}, \pi] \\ t_i^x = -\frac{\pi}{2} & t_i^y = -\frac{\pi}{2} & : & \alpha_i \in [-\pi, -\frac{\pi}{2}] \\ t_i^x = -\frac{\pi}{2} & t_i^y = +\frac{\pi}{2} & : & \alpha_i \in (-\frac{\pi}{2}, 0). \end{cases} \tag{4.27}$$

The transition index takes the values $\frac{\pi}{2}$ and $-\frac{\pi}{2}$ since those values are obtained from the monogenic phase response. In case that the model is defined by a white object on a dark background, the transition values of last equation take the opposite sign. An example of the computation of the transition index for the cactus model is shown in figure 4.12. The pictures show a comparison of the phase response $\parallel r_i^x \parallel$ computed from equation (4.19) with the computed transition index t^x from the projected cactus model.

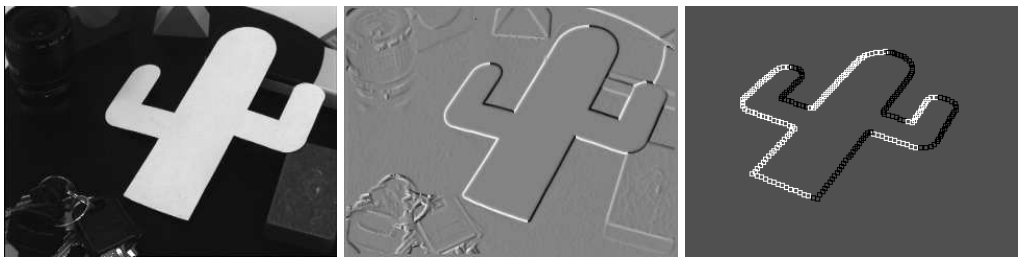

Fig. 4.12: Comparison of the monogenic phase response and the transition index. Left picture: original image of the cactus. Middle image: monogenic phase response in x direction. Right image: computed transition index for all model points in x direction.

Other possible scenario has been considered for the computation of the transition index. Suppose a gray object moving on a background with some dark and white regions as shown in the artificially generated image of figure 4.13. In this case, it is not possible to compute the transition values directly from equation (4.27). The transition index of a model point is different if the background is white or dark. For these scenarios, the same principle used in the background subtraction techniques

[76] is applied. A background image is used as a reference to compute the transition index. In a first instance, the model points are projected and the transition index is computed assuming a white background by equation 4.27. Then, the average gray value level of a region of size n around the point $\mathbf{x}_i$ is computed in the background image $\mathbf{f}_{ref}(\mathbf{x})$ as

$$\text{avg}(\mathbf{x}_i) \quad = \quad \frac{1}{(2n+1)^2} \sum_{j=-n}^{n} \sum_{i=-n}^{n} \mathbf{f}_{ref}(u+i, v+j), \tag{4.28}$$

where u, v are the image coordinates of the point $\mathbf{x}_i$. If G_{mod} is the gray value of the contour model, the transition values change their sign according to the next criterion

$$t_i^x, t_i^y \Rightarrow \left\{ \; t_i^x = -t_i^x, \quad t_i^y = -t_i^y \quad : \quad avg(\mathbf{x}_i) < G_{mod}. \right. \tag{4.29}$$

This criterion simply changes the sign of the transition indexes if the average gray level of the reference image is smaller than the grey value of the contour model. A comparison of the monogenic phase response and the transition index for this scenario is shown in figure 4.13. For this example, the cactus model was projected onto the background image to generate an artificial image. The monogenic phase response of this artificial image in the x direction is shown in the lower left picture. In order to compare the phase and the transition index with respect to the reference background image, the transition values where projected onto the phase response image of the background as shown in the lower right image.

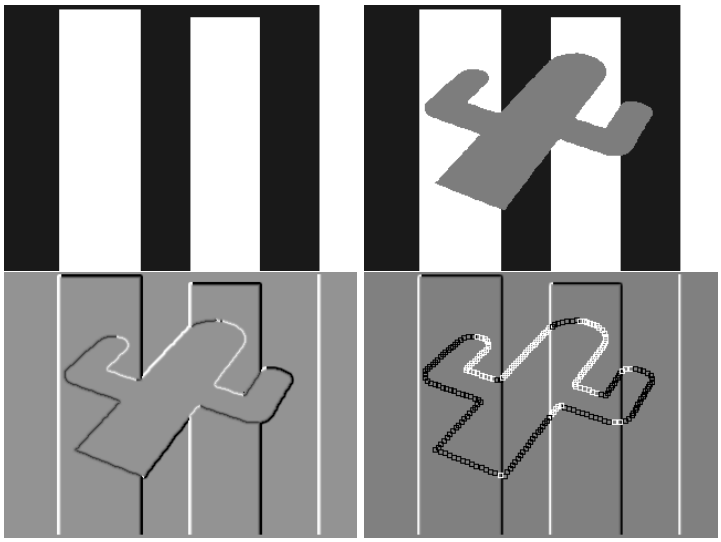

Fig. 4.13: Upper row: background image (left) and artificially projected cactus model (right). Lower row: phase response of the artificial image (left) and transition index projected onto the background phase response (right).

As can be seen in the presented examples, the transition index is easily computed without the need of generating complete artificial images of the projected model. This will allow to compare the transition index and the phase information extracted from the image during each iteration of the ICP algorithm without a considerable increase of the computation time. In normal tracking and pose estimation estimation scenarios, the color of object and background does not change drastically during a sequences of images. Then, the assumptions made to compute the transition index are consistent for the most standard scenarios. As mentioned before, only planar 3D contours are considered for this initial approach. The computation of the transition index for projected non-planar and surface models will be discussed in the next chapter where extra considerations are made for those cases.

4.4.2 Contour Segmentation

As mentioned in section 4.2, semi-local features are computed from model and image contour segments which describe their average local orientation and curvature. Although these features are not invariant in a monocular pose estimation scenario, they can be used to classify contour segments according to their structure. In this case, contour segments can be classified in straight, concave or convex segments [34, 69]. Since 3D planar contour models are considered, it can be assumed that the projection of a 3D straight segment results in a 2D straight segment. The same is assumed for concave and convex segments. This is valid for a relatively large number of poses except in the case that the contour model is perpendicular to the image plane. Therefore, the semi-local structure of models is preserved after the perspective projection under this assumption. In order to perform the contour classification, the local curvature of contour segments is used.

Let us remember that only 3D or projected contour models may be approximated by twist rotations in order to define the pose estimation constraints (see section 3.4). Local curvature of contour models is directly obtained from the local motor representation introduced in section 3.5.3. According to the pose constraints, there is no need to compute local motors from image contours. On the other hand, no local curvature is obtained from the monogenic signal response. Then, a simplified method is applied in order to obtain local curvature from image contours.

The idea of computing the curvature for a point $\mathbf{x}_i$ and its two neighbors is shown in figure 4.14. Once that the points are defined in a local coordinate system[2] constructed by the basis vectors $\mathbf{i}_1$ and $\mathbf{i}_3$, the curvature is simply defined as the rate of change of the angle between the vectors $\mathbf{x}_i - \mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1} - \mathbf{x}_i$. Then, the bending angle β_i representing the curvature and curvature vector $\mathbf{k}_i$ are computed

[2] Note that this local coordinate system is the same as the one used to define the local motor for 3D contours as described in section 3.5.2.

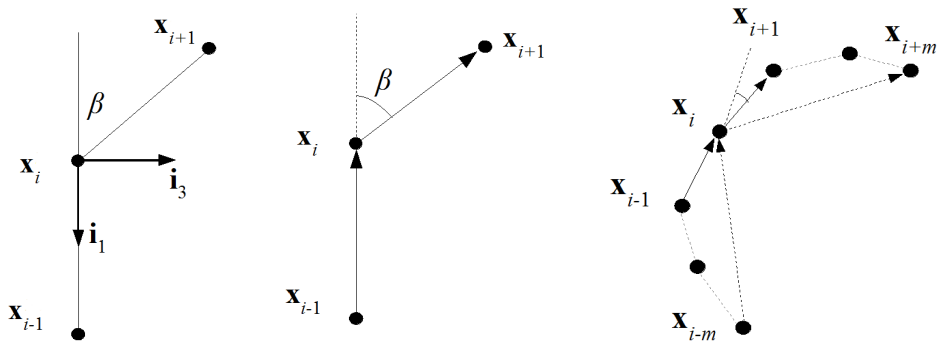

Fig. 4.14: Local curvature defined by the bending angle β (left and center) and extended neighborhood to compute robust features (right).

as follows

$$\beta_i = \arccos \frac{(\mathbf{x}_i - \mathbf{x}_{i-1}) \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i)}{\|(\mathbf{x}_i - \mathbf{x}_{i-1})\| \|(\mathbf{x}_{i+1} - \mathbf{x}_i)\|} \tag{4.30}$$

$$\mathbf{k}_i = (\mathbf{x}_i - \mathbf{x}_{i-1}) \times (\mathbf{x}_{i+1} - \mathbf{x}_i). \tag{4.31}$$

In this case, the curvature vector takes the form $\mathbf{k}_i = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$, where x_3 changes its sign when the three points correspond to a concave or convex segment. Based on the bending angle and the sign of x_3, the structure index s_i is defined according to the criterion

$$s_i := \begin{cases} 1 & : \ \text{sign}(x_3) > 0 \quad \wedge \quad \beta_i > thres \\ -1 & : \ \text{sign}(x_3) < 0 \quad \wedge \quad \beta_i > thres \\ 0 & : \ \beta_i \leq thres, \end{cases} \tag{4.32}$$

where $thres$ is a given threshold value. $s_i = 1$ means local convexity, while $s_i = -1$ stands for local concavity. If the bending angle β_i has a value closed to zero, the point is considered to be a part of a straight line.

An extended feature calculation allows to compute more robust features, especially in the image plane where noise is present and digital contours are extracted. As can be seen in figure 4.14, the bending angle and curvature vector are computed not only from the two neighbor points around $\mathbf{x}_i$. This neighborhood is extended to larger segments of a given range m. By taking the point $\mathbf{x}_i$ as a reference, the average feature values are computed as follows:

$$\beta_i = \frac{1}{m} \sum_{j=1}^{m} \arccos \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|} \tag{4.33}$$

$$\mathbf{k}_i = \frac{1}{m} \sum_{\text{j=1}}^{m} \mathbf{v}_1 \times \mathbf{v}_2, \tag{4.34}$$

with $\mathbf{v}_1 = \mathbf{x}_i - \mathbf{x}_{i-j}$ and $\mathbf{v}_2 = \mathbf{x}_{i+j} - \mathbf{x}_i$.

It is important to select an appropriate range value m to compute the average features. Examples of the segmentation of the cactus and puzzle models for different range values m are presented in figure 4.15. The contours were extracted from the test images to visualize the segmentation results. At lower range values, some points are clearly wrong classified for both models. Several points that are part of concave or convex segments are classified as straight lines. For range values around $m = 10$, the result of the classification shows better results. At larger ranges ($m = 20$), some straight line segments of the cactus model are classified as concave. That means, large range values may cause the lost of the structure of certain segments.

In the next experiment, the robustness of the segmentation was tested in the presence of noise. Different levels of Gaussian white noise were added to the extracted contours. Some examples of noisy contours and their corresponding bending angle profiles are shown in figure 4.16. The graphics show the distortion of the bending angle profiles for different levels of noise. The angle β is multiplied by the factor $\text{sign}(x_3)$ (this is done to visualize the effect of the conditions $\text{sign}(x_3) > 0$ and $\text{sign}(x_3) < 0$ of equation (4.33)). Both lines represent the positive and negative threshold values which determine if a point is considered concave or convex.

The behavior of the classification of noisy contours was analyzed at several range values. For a given range m, a reference vector $\mathbf{s}_{ref} = \{s_1 \cdots s_n\}$ containing the computed structure indexes without noise is defined according to equation (4.32). Similarly, a vector containing the noisy classification indexes is defined as $\mathbf{s}_m = \{s_1 \cdots s_n\}$. This was done for noise levels from $\sigma = 0$ to $\sigma = 2$. By using the vector $\mathbf{s}_{ref}$ as a ground truth, the error is computed as

$$E(\mathbf{s}_{ref}, \mathbf{s}_m) = \|\mathbf{s}_{ref} - \mathbf{s}_m\|^2. \tag{4.35}$$

The results are shown in figure 4.17. As expected, the segmentation is more sensitive to noise at lower range values for all models. At larger ranges, the error remains relatively constant for the different noise levels. An adequate range value for all contour models can be defined between $m = 10$ and $m = 15$. Within these range values, the classification is performed without loosing important structure of the segments. This simple experiment shows that the presence of noise always adds a certain level of error in the segmentation. In practice, the same range value must be used to segment contour models and contours extracted from the image.

4.5 Structural ICP Algorithm

The procedures to obtain compatible local features from contour models and extracted image contours were presented in the last sections. Image contour segments

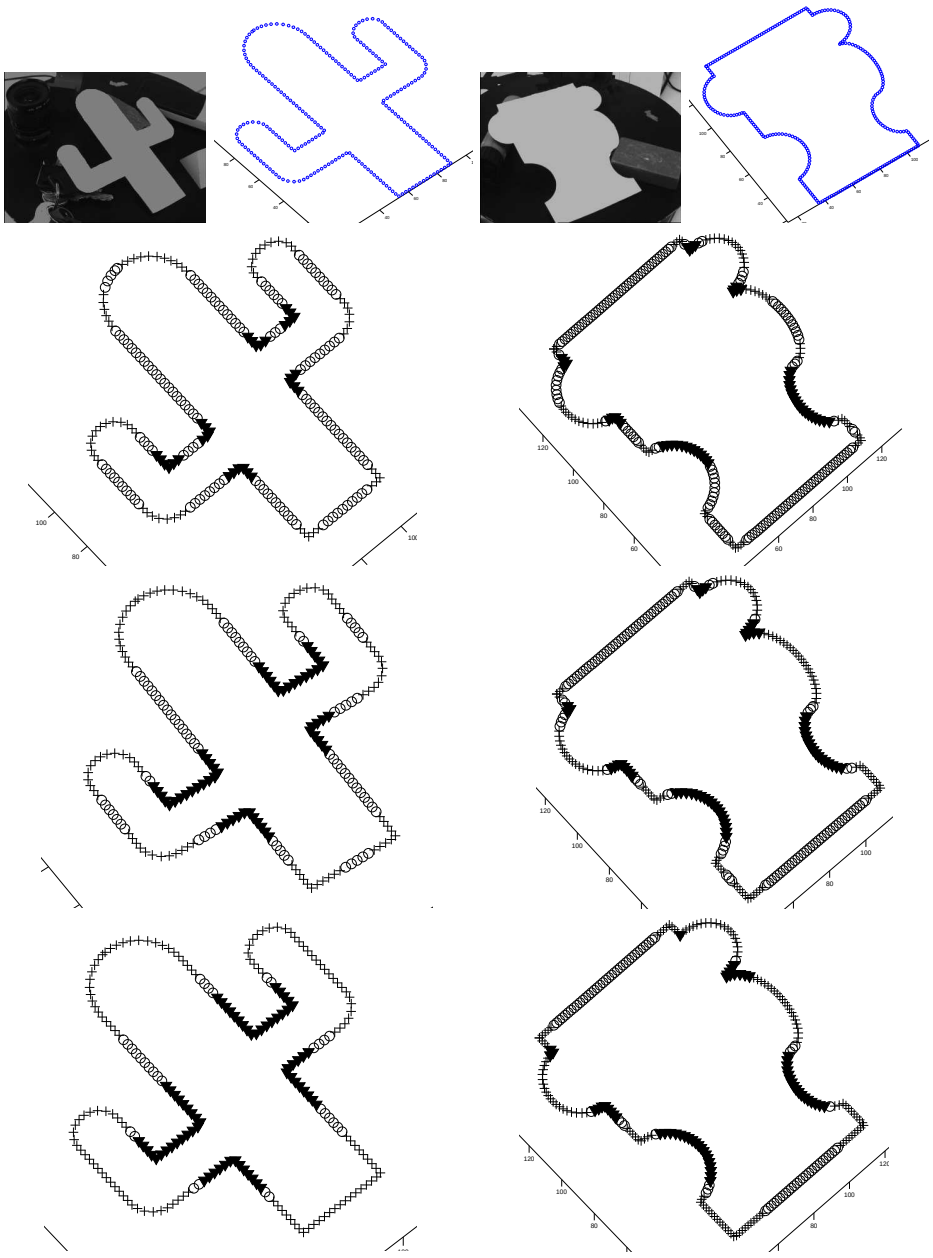

Fig. 4.15: Examples of the contour segmentation procedure. Top row: original images of the cactus and puzzle. From second to fourth rows: segmented models for range values $m = 5$, $m = 10$ and $m = 20$ respectively.

and their local feature information are extracted from the monogenic scale space. On the other hand, compatible features are obtained from contour models. As discussed in section 4.1, the correspondences are computed in accordance with a given similarity criterion. Then, a new variant of ICP algorithm is presented in this section. This variant is called the "structural iterative closest point algorithm", see [26]. It is

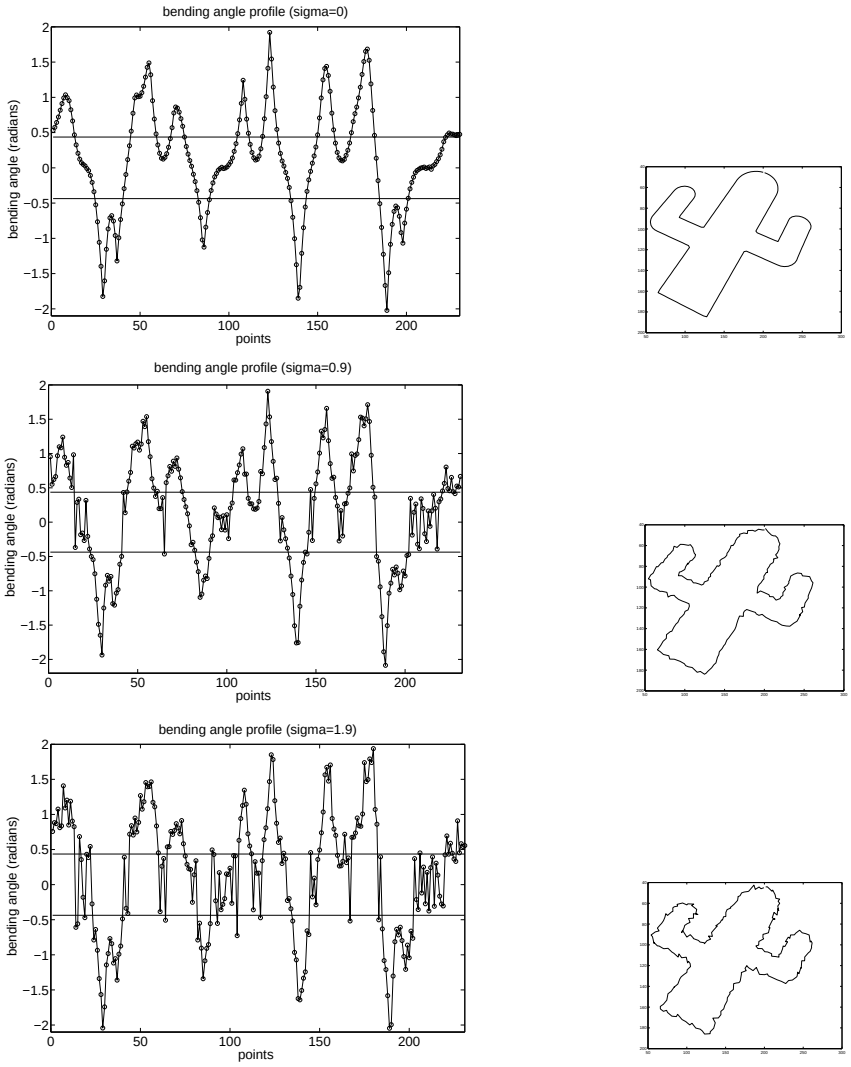

Fig. 4.16: Noisy profiles and contour examples for noise levels: sigma=0 (upper), sigma=0.9 (middle) and sigma=1.9 (lower).

an adapted version of the normal ICP algorithm for the monocular pose estimation problem, where the local structure of model and image points is combined with the Euclidean distance to define new correspondence search constraints. Initially, the structural constraints are defined with the sets of model and image contour features obtained in the last sections. Finally, the structural ICP algorithm is defined for the 3D-2D and projective pose estimation constraints defined in section 2.5.3.

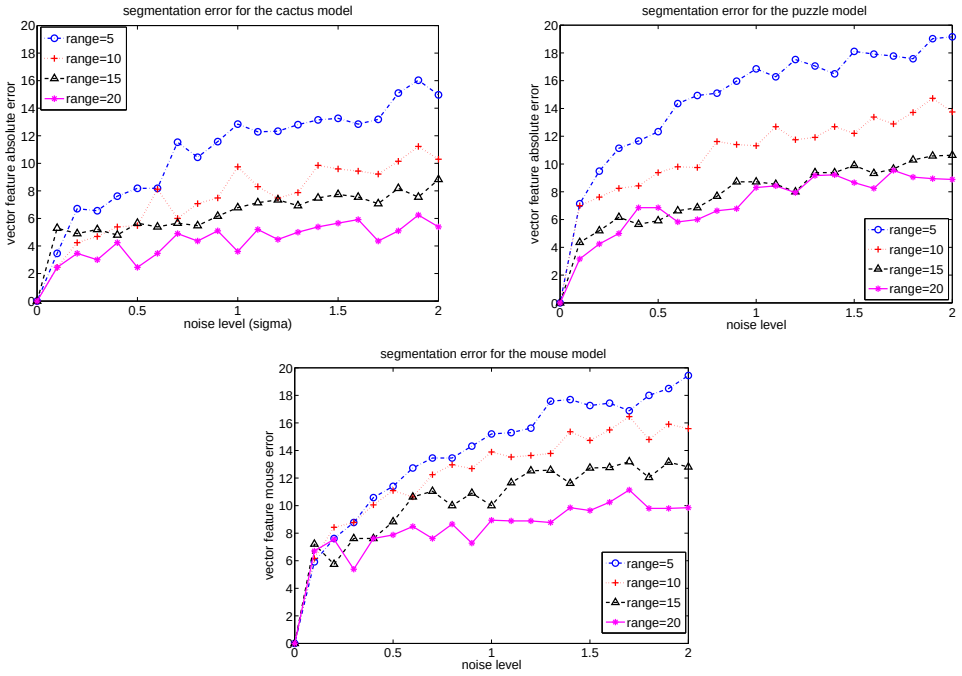

Fig. 4.17: Results of the segmentation error experiment for the cactus (upper left figure), puzzle (upper right figure) and mouse (lower figure).

4.5.1 Structural Constraints

According to the notation introduced in section 4.1, the group of model points in 3D space is defined as $\mathcal{M}^{3D} = \{\underline{\mathbf{X}}_i\}_{i=1}^{n}$. Once that each model point is projected onto the image plane, the group of projected model points $\mathcal{M}^{2D} = \{\mathbf{x}_i^{mod}\}_{i=1}^{n}$ is formed. On the other hand, the group of image detected points is denoted as $\mathcal{S}^{2D} = \{\mathbf{x}_j^{img}\}_{i=1}^{m}$. For every projected model point $\mathbf{x}_i^{mod}$ and image point $\mathbf{x}_j^{img}$, their respective structural feature sets are defined as

$$\mathbf{f}_i^{2Dm} = \{\alpha_i, t_i^x, t_i^y, \mathbf{k}_i^{2Dm}, \beta_i^{2Dm}\} \tag{4.36}$$

$$\mathbf{f}_j^{2Dp} = \{\phi_j, \parallel r_j^x \parallel, \parallel r_j^y \parallel, \mathbf{k}_j^{2Dp}, \beta_j^{2Dp}\}, \tag{4.37}$$

where the pair (α_i,ϕ_i) corresponds to the local orientation of model and image points and the pairs ($t_i^x, \parallel r_i^x \parallel$), ($t_i^y, \parallel r_i^y \parallel$) denote the phase-transition indexes in x and y directions. The features related to the semi-local structure of the contour segments are the curvature vectors ($\mathbf{k}_i^{2Dm}$, $\mathbf{k}_i^{2Dp}$) and the bending angles (β_i^{2Dm}, β_i^{2Dp}).

Based on these features, three main structural constraints are defined for a point pair $(\mathbf{x}_i^{mod}, \mathbf{x}_j^{img})$. The phase-transition index constraint is defined as follows

$$C_1 = \begin{cases} 1 & \text{if } \parallel r_j^x \parallel = t_i^x \quad \text{and} \quad \parallel r_j^y \parallel = t_i^y \\ 0 & \text{otherwise.} \end{cases} \tag{4.38}$$

The following constraints are defined according to the segmentation criterion of equation 4.32. The second constraint is called straightness constraint and it is defined with the local bending angles $\beta_i^{2\mathrm{Dm}}$ and $\beta_j^{2\mathrm{Dp}}$ by

$$C_2 = \begin{cases} 1 & \text{if } \beta_i^{2\mathrm{Dm}} < t \quad \text{and} \quad \beta_j^{2\mathrm{Dp}} < t \\ 0 & \text{otherwise,} \end{cases} \tag{4.39}$$

where t is a threshold value that determines if a point is selected as a straight line (see figure 4.16). Finally, the concavity-convexity constraint is defined from the sign of the $\mathbf{e}_3$ component of the vectors $\mathbf{k}_i^{2Dm} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$ and $\mathbf{k}_j^{2\mathrm{Dp}} = y_1\mathbf{e}_1 + y_2\mathbf{e}_2 + y_3\mathbf{e}_3$ (see equation 4.31) by

$$C_3 = \begin{cases} 1 & \text{if } \mathrm{sign}(\mathrm{x}_3) = \mathrm{sign}(\mathrm{y}_3) \quad \text{and} \quad C_2 = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.40}$$

Notice that the condition $C_2 = 0$ was added in the last constraint. It simply ensures that the constraint will be fulfilled only on those cases where the image and model points are not previously considered to be part of straight segments.

Once that the constraints are defined, their combination with the Euclidean distance criterion is quite simple and easy to interpret. Then, for each image detected point $\mathbf{x}_j^{img} \in \mathcal{S}^{2D}$, its corresponding model point is found with the combination of the Euclidean distance and the structural constraints defined as follows

$$\begin{aligned} d(\mathbf{x}_i^{img}, \mathcal{M}^{2D}) \quad &= \quad \min_{j=1\cdots n} \left\{ d(\mathbf{x}_i^{img}, \mathbf{x}_j^{mod}) \right\} \\ &\quad \text{if} \quad C_1 \wedge (C_2 | C_3) = 1, \end{aligned} \tag{4.41}$$

where the symbols $\wedge$ and $|$ stand for the "and" and "or-exclusive" logical operations respectively. Only one of the constraints C_2 or C_3 can be equal to one at the same time. That means, two corresponding points must be part of a straight line segment $C_2 = 1$ or part of a concave or convex segment $C_3 = 1$.

The correspondence set $\{\mathcal{S}^{2D}, \mathcal{M}^{2D}\}$ is formed with the last equation. With the addition of the structural constraints, the correspondence search process is restricted to subgroups of elements of $\mathcal{M}^{2D}$ and $\mathcal{S}^{2D}$ with the same structure. Therefore, corresponding points will be the closest points which have the same semi-local structure.

Figure 4.18 shows the general idea of the minimal Euclidean distance criterion combined with the structural constraints. The upper left figure shows the case where only the minimal distance is considered. It is evident that many of the correspondence pairs are not well conditioned. In the upper right figure, the same example is shown with the structural constraints C_2 and C_3. A similar example is shown for the cactus model with the phase-transition index constraint C_1.

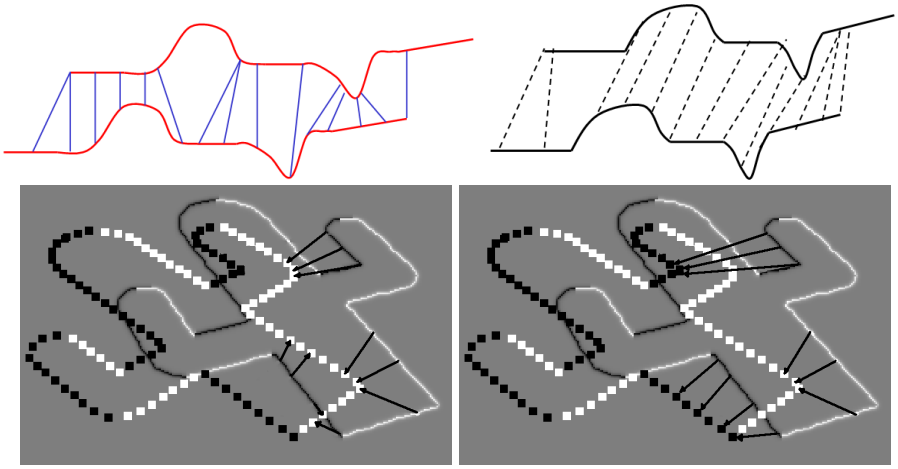

Fig. 4.18: Example of correspondence pairs for normal (upper left) and structural (upper right) ICP variants. Correspondence example for the cactus model by considering only the Euclidean distance (lower left) and with the phase-transition constraint (lower right). The lines show the corresponding pairs of image and model.

Figure 4.19 shows the result of the segmentation of two artificially generated contours. The pictures show the segmentation of both contours according to its structure (concavity, convexity and straightness) as well as the transition indexes in x and y directions. With this segmentation results, correspondences were found by combining the Euclidean distance with each constraint C_1, C_2 and C_3. The computed correspondences with the minimal Euclidean distance criterion are shown in the upper left picture of figure 4.20. In the next picture, the correspondences with the combination of the concavity-convexity constraints are shown. For this position of the contours, the correspondences with C_2 and C_3 are more similar than the correspondences with the minimal distance criterion. Let us notice that the structure of this contour is relatively complex. Therefore, the closest point with the same concavity-convexity structure is not enough to improve significatively the correspondences for this example. According to the segmented contours shown in figure 4.19, it can be clearly seen that closest points with the same concave, convex or straight structure result in bad conditioned correspondences.

The lower left picture of figure 4.20 shows the combination of the minimal distance with the phase-transition index constraint C_1. As mentioned before, the corresponding point in this case is the closest point with the same phase and transition indexes. As can be seen in the example, better conditioned correspondences are found with the addition of the constraint C_1. According to the classification of figure 4.19, closest points with the same phase and transition indexes tend to be better conditioned. Finally, the lower right picture shows the correspondences with the combination of all constraints. This simple example shows the advantage to com-

bine all constraints in the correspondence search process.

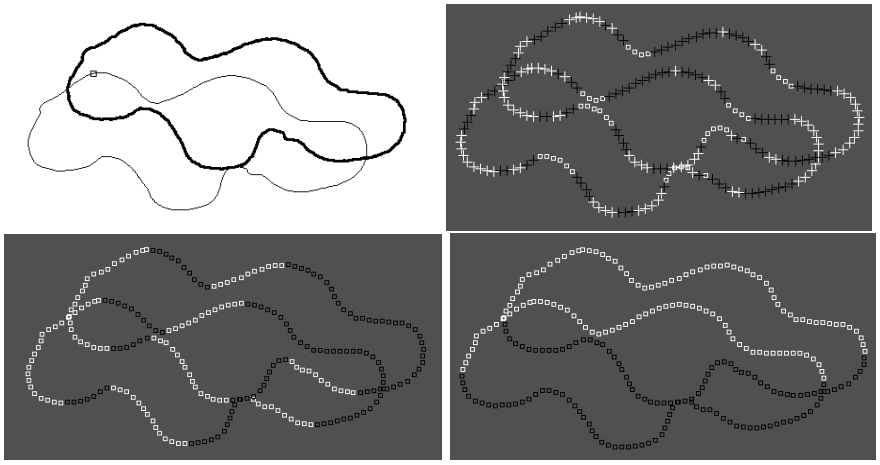

Fig. 4.19: Upper row: model and image detected contours (left) and result of the segmentation according to the concavity-convexity constraint (right). Lower row: transition indexes in the x (left) and y directions (right).

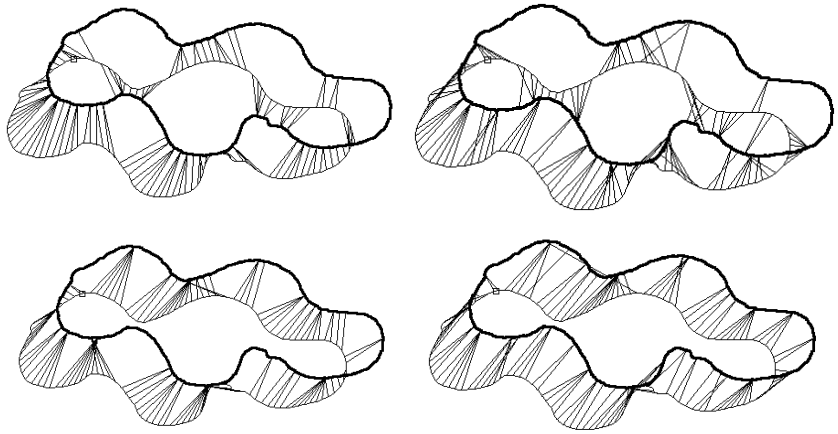

Fig. 4.20: Correspondences with the minimal distance criterion (upper left). Correspondences with the combination of minimal distance and the concavity-convexity constraint (upper right). Combination of minimal distance and phase-transition index constraints (lower left). Combination of minimal distance, phase-transition index and concavity-convexity constraints.

4.5.2 Contour based Structural ICP Algorithm

A correspondence set of image and projected model points $\mathcal{C}^{2\mathrm{D}} = \{\mathbf{x}_i^{img}, \mathbf{x}_i^{mod}\}_{i=1}^n$ is obtained with the combination of the minimal Euclidean distance criterion and the structural constraints. Although this set has been obtained in the image plane, the correspondence sets needed to compute the pose according to the minimization constraints of section 4.1 are directly obtained. The idea is shown in figure 4.21. Let us remember that the point $\mathbf{x}_i^{mod}$ is the projection of the point $\underline{\mathbf{X}}_i^{mod}$. Then, a 2D-3D point correspondence set $\mathcal{C}_1 = \{\mathbf{x}_i^{img}, \underline{\mathbf{X}}_i^{mod}\}_{i=1}^n$ is directly obtained. With the corresponding image point $\mathbf{x}_i^{img}$, an optical ray is computed by $\underline{\mathbf{L}}_i = \mathbf{e} \wedge \mathbf{O}_c \wedge \mathbf{x}_j^{img}$ and the point-line correspondence set $\mathcal{C}_2 = \{\underline{\mathbf{L}}_i, \underline{\mathbf{X}}_i^{mod}\}_{i=1}^n$ is defined.

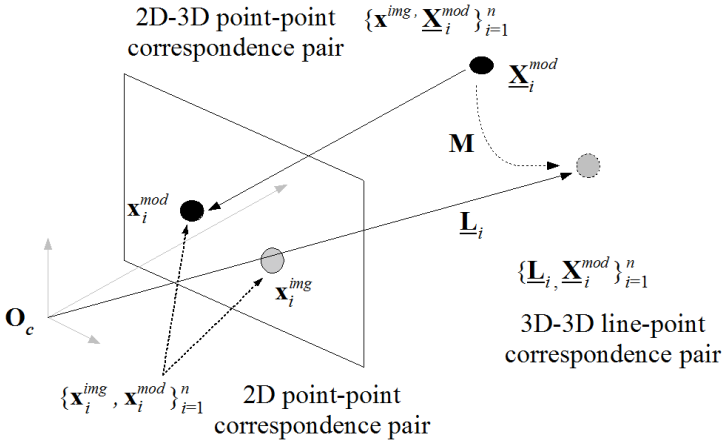

Fig. 4.21: Recovering the 2D-3D point-point and 2D-3D point-line correspondence sets from the correspondences obtained in the image plane.

Similar to the classical ICP algorithm, the structural ICP variant is summarized for the different pose estimation scenarios as follows: given the model and image sets $\mathcal{M}^{3D}, \mathcal{S}^{2D}$ and the local features obtained from the image points $\mathcal{S}^{2D}$, repeat for $j = 1$ to n iterations

Structural Iterative Closest Point Algorithm.

1. Project the model points $\mathcal{M}^{3D}$ onto the image plane to obtain the set $\mathcal{M}^{2D}$ and its corresponding local features.
2. For each point of $\mathcal{S}^{2D}$, find the corresponding projected model point of $\mathcal{M}^{2D}$ with the search criterion of equation (4.41).
3. With the correspondence set $\mathcal{C}_1$, compute the pose parameters M with the 2D-3D constraint

$$0 = \lambda \Big\{ \mathbf{x}^{img} \quad - \quad \mathcal{P}(\mathbf{M}\underline{\mathbf{X}}_i\widetilde{\mathbf{M}}) \Big\},$$

or compute $\mathbf{M}$ with the correspondence set $\mathcal{C}_2$ and the projective constraint

$$0 = \lambda \left(\begin{pmatrix} \mathbf{M} & \underline{\mathbf{X}}_i & \widetilde{\mathbf{M}} \end{pmatrix} \quad \underline{\times} \quad \underline{\mathbf{L}}_i \right) \cdot \mathbf{e}_+.$$

4. Actualize the position of the model points $\mathcal{M}^{3D}$ with the computed pose parameters $\mathbf{M}$.

5. Compute the positional error in the image plane $err = d(\mathcal{S}^{2D}, \mathcal{M}^{2D})$ between the projection of the model and the image points. If $err < thres$, exit; else goto 1.

4.5.3 Correspondence Search Direction

The structural ICP algorithm was defined to find correspondences by taking the image set $\mathcal{S}^{2D}$ as a reference with respect to the model set $\mathcal{M}^{2D}$. It is also possible to find correspondences in the opposite direction [94]. By taking a model point as a reference, its corresponding image point is found. Then, correspondence sets in image-model and model-image directions are defined by

$$d(\mathbf{x}_i^{img}, \mathcal{M}^{2D}) \rightarrow \mathcal{C}^I = \{\mathbf{x}_i^{img}, \mathbf{x}_i^{mod}\} \tag{4.42}$$

$$d(\mathbf{x}_i^{mod}, \mathcal{S}^{2D}) \rightarrow \mathcal{C}^D = \{\mathbf{x}_i^{mod}, \mathbf{x}_i^{img}\}. \tag{4.43}$$

Depending on the search direction, different correspondence sets are obtained. An example is shown in figure 4.22. The fine line represents the detected image points $\mathcal{S}^{2D}$ and the rough line represents the projected model points $\mathcal{M}^{2D}$. Correspondence sets were found as described in the last equations with the combination of Euclidean distance criteria and structural constraints. As can be seen in the lower left picture, the correspondence set $\mathcal{C}^I$ is better conditioned than the set $\mathcal{C}^D$ shown in the lower right picture. In practice, it is not possible to determinate if the computed correspondences will be better conditioned in a specific direction. Therefore, the use of any of these correspondence sets may not necessary ensure a better behavior of the algorithm for a given pose scenario. Despite of that, the change of the search direction during the iterations may increase the probability to find better conditioned correspondences in some cases.

The choice of the correspondence set can be made according to the measured error $err = d(\mathcal{S}^{2D}, \mathcal{M}^{2D})$. If the error increases with respect to the computed error of the last iteration, the correspondence set is changed and the pose parameters are computed again.

4.6 Summary

In this chapter, the structural ICP algorithm was presented for the pose estimation of planar free-form contours. Sets of compatible model and contour image features

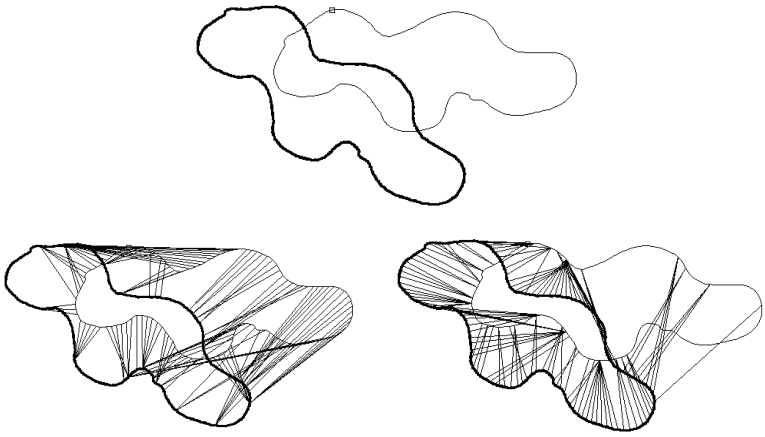

Fig. 4.22: Upper figure: model contour (rough line) and image contour (fine line). Lower right: correspondences in image-model direction. Lower left: correspondences in model-image direction.

have been obtained. This compatibility is gained by computing features based on local orientation and curvature. Since the correspondence problem has been completely translated onto the image plane, it is possible to use additional structural features that are only available for image contours. In that sense, the local orientation of projected model contours was used to find a compatible feature to the image phase response. On the other hand, contours are classified according to their semi-local structure in straight, concave and convex segments depending on their average local curvature. For the case of planar contours, it was assumed that the semi-local structure is preserved after perspective projection up to some limits. This allowed to define additional constraints which were used in the definition of the structural ICP algorithm.

The presented examples consider relatively large displacements between model and image contours. In this case, the use of the concavity-convexity constraint is not enough to obtain a significant improvement of the correspondence sets with respect to the normal ICP algorithm. With the addition of the phase-transition index constraint, better conditioned correspondence sets are obtained. As mentioned in section 4.2, the structural features allow to describe contours on a higher level than the simple Euclidean distance. Then, the probability to find better conditioned correspondences with the structural constraints increases in comparison to the minimal distance criterion. For the structural ICP algorithm, the model objects were restricted to planar 3D contours. Under some considerations, the structural ICP algorithm can be applied to free-form surfaces as will be shown in the next chapter.

Chapter 5

CORRELATION ICP ALGORITHM

All variants of the ICP algorithm use the Euclidean distance as main correspondence search criterion. Therefore, the search process is limited within a region where the correct correspondence pairs may be found (tracking assumption condition). This fact implies a natural limitation of the algorithms. Despite of that, better conditioned correspondences are found with the structural ICP algorithm. Furthermore, it is possible to combine global and local features of model and image contours in order to improve the pose estimation algorithms. As mentioned in the last chapter, the more information is available about model and image data, the better is the probability to find better correspondences. If more feature information is used to describe contour segments, their description level increases as it was discussed in section 4.2. Contour information (spatial 2D or 3D position) is not sufficient to describe the structure of contour segments and to use this information to improve significatively the correspondence search problem. A natural way to overcome this problem is to upgrade the algorithms from contour-based to feature-based approaches. Instead of considering the position of points, a direct comparison of their local and global features is done in order to find correspondences.

In this chapter, the correlation based ICP algorithm is presented for the pose estimation of free-form surface models. Initially, the silhouette-based pose estimation algorithm is described. Instead of considering the complete surface model, its 3D silhouette with respect to the image plane is used to compute features, find correspondences and compute the pose. The correlation matrix is used to define a similarity measure between model and image feature information. One option is to combine the Euclidean distance criterion with the feature correlation as an extra structural constraint. A second variant replaces the Euclidean distance with the correlation measure. In the second part of this chapter, the adaptation of the classical pre-alignment approaches for the monocular pose estimation is introduced. Finally, a technique to simplify the pre-alignment is presented. This is achieved by using a combination of global and local features obtained from projected model information and extracted image contours.

5.1 Pose Estimation of Free-form Surface Models

The pose estimation constraints for free-form surfaces have been defined in section 3.4. However, suitable model-image correspondences must be found in order to define these pose estimation constraints. A typical pose estimation example of a free-form surface is shown in figure 5.1. Surface models are more complex than free-form contours. As can be seen in the right picture of the figure, all surface points are projected onto the image containing all detected edge points. Trying to find suitable image-model correspondences is not a trivial task if only the position of surface model and image points is available. Notice that the edge points corresponding to the contour of the object in the image (middle picture) can be easily extracted. Then, the correspondence problem can be simplified if the points which define the silhouette of the surface model are determined.

Fig. 5.1: Surface model of the power socket (left). Captured image (middle). Projected surface points and detected image contour segments (right).

Instead of considering all surface points, correspondences are found between image points and the 3D silhouette of the surface model with respect to the image plane. With this correspondence set, the pose of the complete surface model is computed. In this section, the process to extract 3D silhouettes from surface models is initially described. Finally, the ”silhouette-based ICP algorithm” see [89], is introduced.

5.1.1 Silhouette Extraction

A surface model can be interpreted as a set of 3D free-form contours. In that sense, the external 3D contour of the surface with respect to the image plane is known as 3D silhouette. Figure 5.2 shows an example of a surface model in 3D space and its projection onto the image plane. As can be seen in the figure, only the node points of the surface are part of the model. The lines that join every node point allow a better visualization in 3D space or in the image plane. Such lines generate a set of ”virtual” contours. They are called virtual because they are not implicitly part of

the surface model. In that sense, the marked contour shown in the right picture of figure 5.2 defines the virtual 2D silhouette SIL^{2D}.

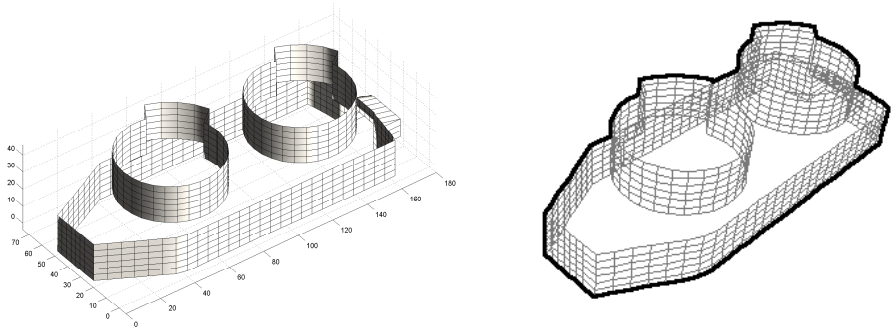

Fig. 5.2: Surface model in 3D space (left) and virtual 2D silhouette defining the contour of the projected model (right).

The group containing all 3D surface model points is defined as $\mathcal{M}^{3D}$ and the corresponding 2D projected points is denoted by $\mathcal{M}^{2D}$. If this virtual silhouette is used as a reference, the 3D silhouette with respect to the image plane is defined as

$$\mathrm{SIL}^{3D} = \{\underline{\mathbf{X}}_i \in \mathcal{M}^{3D} | \mathcal{P}(\underline{\mathbf{X}}_i) \in \mathrm{SIL}^{2D}\}, \tag{5.1}$$

where the operation $\mathcal{P}(\underline{\mathbf{X}}_i)$ stands for the perspective projection of the model point onto the image plane. The problem of finding the 3D silhouette is reduced to find the projected surface points which are part of the virtual 2D silhouette SIL^{2D}.

In order to extract the 3D silhouette, the surface model is projected onto a reference image IM_{ref}. For every model point $\underline{\mathbf{X}}_i \in \mathcal{M}^{3D}$ with coordinates in 3D space (x_i, y_i, z_i), its corresponding projected point is denoted as $\mathbf{x}_i \in \mathcal{M}^{2D}$ with image coordinates (u_i, v_i). As can be seen in figure 5.3, every projected model point is marked on the reference image with a different gray level value. This will serve as a reference to select only the projected points which are part of the 2D silhouette. Simultaneously to the projection onto the reference image, the 3D coordinates of every node point are stored in three 2D arrays $\mathrm{X}^{m\times n}$, $\mathrm{Y}^{m\times n}$ and $Z^{m\times n}$ as follows

$$\begin{aligned} X[u_i][v_i] &= x_i \\ Y[u_i][v_i] &= y_i \\ Z[u_i][v_i] &= z_i, \end{aligned} \tag{5.2}$$

where the dimension of the arrays (m, n) is the same as the dimension of the reference image.

The coordinates of each projected point (u_i, v_i) define the position in the array where each 3D coordinate is stored. These arrays can be interpreted as three different images, where the gray value of each point has been replaced by each 3D

coordinate x, y and z respectively. Once that the model has been completely projected onto the auxiliary image, a contour search algorithm is applied to the reference image IM_{ref} as can be seen in figure 5.3. This algorithm recovers all points of the virtual 2D silhouette SIL_{im}^{2D}. Let us remember that only the marked pixels in this virtual silhouette are part of the 3D silhouette. As the algorithm follows the contour, it detects the 2D coordinates of all marked pixels (u_i, v_i). Since the corresponding 3D coordinates of these points are stored in the arrays, they are directly recovered from equation (5.2).

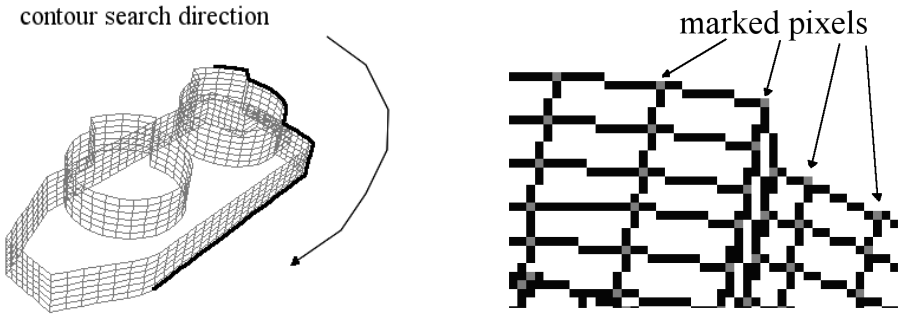

Fig. 5.3: Left image: reference image used to detect the 2D virtual silhouette by a contour search algorithm. Right image: marked pixels which correspond to the 3D silhouette points.

Several examples of extracted silhouettes for different poses are presented in figure 5.4. The left column shows the surface model in 3D. At these poses, the surface model was projected onto the image plane and its corresponding 3D silhouette was extracted. As can be seen in the middle column, some regions of the surface model are almost perpendicular to the image plane. Thus, several surface points are projected onto the same point of the virtual 2D silhouette. Because of that, these points are also detected as part of the 3D silhouette. In these cases, the extracted silhouette results in an irregular 3D contour. The extracted silhouettes are shown in the right column of figure 5.4. Let us notice that the power socket model used for this example is constructed by three different surfaces. Hence, the segments of the silhouette belong to different surface parts. For segments of the same surface part, the sampling distance between silhouette points is in general constant. At points where the silhouette changes from one to other surface part, the sampling distance is not regular anymore (see marked points of right column).

5.1.2 Silhouette-Based ICP Algorithm

In this and the next sections, the same notation introduced in sections 4.1 and 4.5.2 will be used. Then, the silhouette based ICP algorithm used in [89] is summarized.

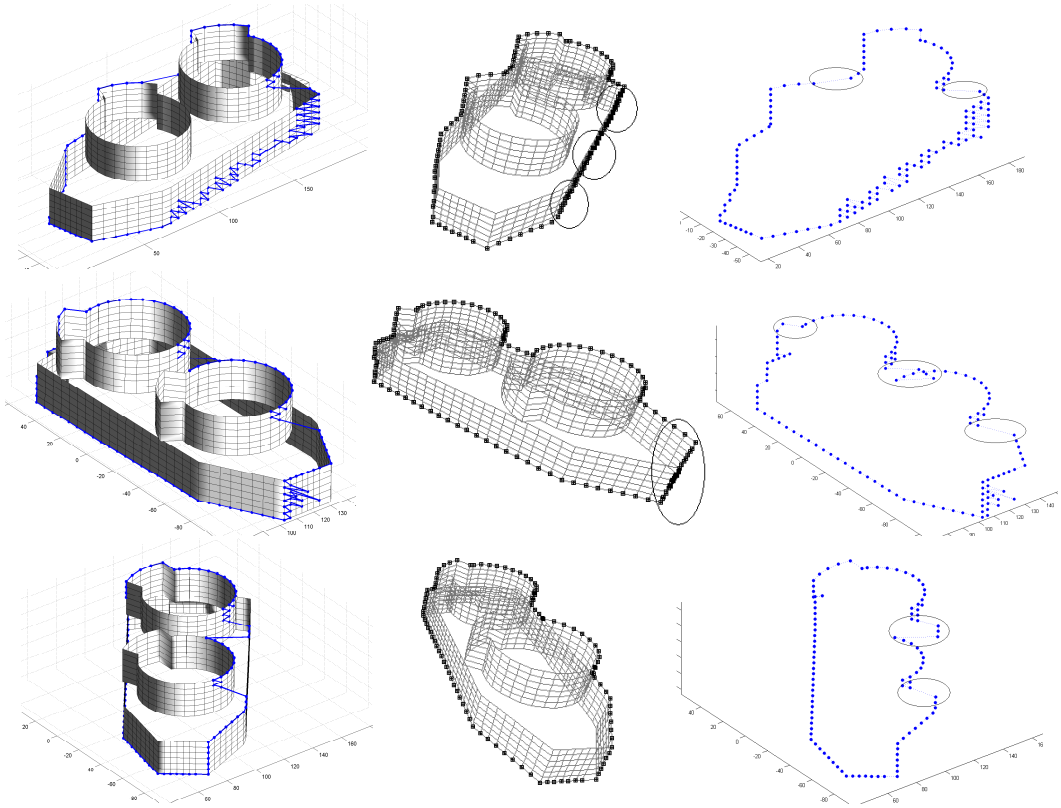

Fig. 5.4: Examples of the silhouette extraction of a surface model at different poses (left column) with respect to the camera plane. Projected silhouettes onto the image plane (middle column). Extracted silhouettes (right column).

Given the set $\mathcal{M}^{3D} = \{\underline{\mathbf{X}}_i\}_{i=1}^{n}$ containing all surface model points and the set of image detected points $\mathcal{S}^{2D} = \{\mathbf{x}_i\}_{i=1}^{m}$, repeat for $j = 1$ to n iterations

Silhouette-based ICP Algorithm

1. Generate the reference image IM_{ref} with the model points $\mathcal{M}^{3D}$ and extract the silhouette points SIL^{3D}.

2. With all image points $\mathbf{x}_i \in \mathcal{S}^{2D}$, define the group $\mathcal{S}^{3D} = \{\underline{\mathbf{L}}_i\}_{i=1}^{n}$ containing the reconstructed optical rays $\underline{\mathbf{L}}_i$.

3. For every optical ray $\underline{\mathbf{L}}_j \in \mathcal{S}^{3D}$, find its corresponding 3D silhouette point $\underline{\mathbf{X}}_i \in \mathrm{SIL}^{3D}$ with the search criterion

$$d(\underline{\mathbf{L}}_j, \mathrm{SIL}^{3D}) = \min_{i=1,\ldots,n} \left\{ \underline{\mathbf{L}}_j \quad \underline{\times} \quad \underline{\mathbf{X}}_i \right\}.$$

4. With the correspondence pairs $\{\underline{\mathbf{L}}_i, \underline{\mathbf{X}}_i\}_{i=1}^{n}$, find the pose parameters $\mathbf{M}$ with

the 2D-3D minimization constraint

$$0 = \lambda \left(\left(\mathbf{M} \quad \underline{\mathbf{X}}_i \quad \widetilde{\mathbf{M}} \right) \quad \underline{\times} \quad \underline{\mathbf{L}}_i \right) \cdot \mathbf{e}_+ .$$

5. Actualize the position of the surface model points $\mathcal{M}^{3D}$ with the computed pose parameters $\mathbf{M}$.

6. Compute the average error $err = d(\mathcal{S}^{3D}, \mathrm{SIL}^{3D})$ between the reconstructed optical rays and the silhouette points. If $err < thres$, exit; else goto 1.

As can be seen in the last algorithm, the pose is computed during the iterations only with the 3D silhouette. With this pose, the position of the complete surface model is actualized in the next iterations. Therefore, this algorithm can be interpreted as a partial reduction of the pose estimation of free-form surfaces to the pose estimation of free-form contours.

5.2 Correlation ICP Algorithm

Similar to the case of the correspondence problem for planar contours, the aim is to find correspondences between projected surface model points and detected image contours by considering their local features. As described in the last section, the 3D silhouette of the surface model with respect to the image plane is used to find correspondences and compute the pose with the classical ICP algorithm. Because of the irregularity of the extracted 3D silhouettes, it is not possible to extract local and semi-local features as in the case of 3D planar contours. To overcome this problem, local features are extracted from the projected surface model with respect to its 2D virtual silhouette. As shown in figure 5.5, this makes possible to compute local and averaged local features at different description levels as described in section 4.2. According to this description levels, profile vectors containing the feature information of larger contour segments are defined.

A similarity measure between profile vectors is used to define the correlation ICP algorithm which is presented in this section. Initially, several examples of local and semi-local features computed from projected 3D silhouettes and real images are presented. Then, the idea of using the correlation matrix as similarity criterion and its application to find correspondences in two possible scenarios are introduced. Finally, the correlation ICP algorithm is summarized for the pose estimation of surface models with the 2D-3D and projective minimization constraints.

5.2.1 Feature Computation with the Virtual 2D Silhouette

The extracted 3D silhouette of a surface model is a 3D closed contour which is in general not planar. Let us consider the example of figure 5.6. In this case, some

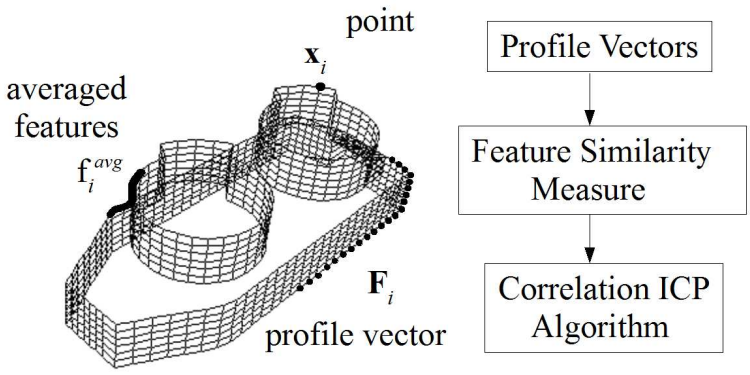

Fig. 5.5: Description level of a projected surface model with respect to its virtual silhouette.

points of the 3D silhouette are part of a straight line which is perpendicular to the image plane. Therefore, all these points are projected onto the same point in the image plane and no local structure can be computed. Similarly, it is also possible that convex and concave segments would be projected as straight segments. The same effect can be seen in the extracted silhouettes of figure 5.4. Since the semi-local structure (convexity, concavity and straightness) is not preserved under the perspective projection anymore, the assumption made in the last chapter for planar contours it is not valid for 3D silhouettes.

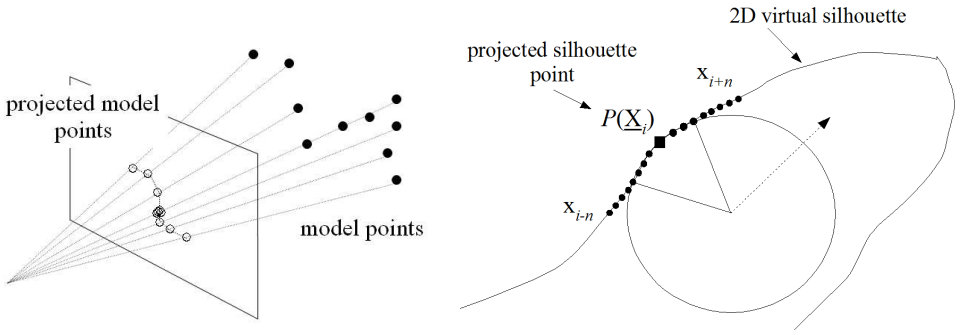

Fig. 5.6: Lost of structure due to the perspective projection (left). Projected model points used as a reference on the virtual silhouette (right).

Because of that, it is not possible to compute local and semi-local features directly from the projected model points. Therefore, some considerations have to be done to compute compatible image and silhouette features. As mentioned in section 5.1, a contour search algorithm is used to detect the projected silhouette points in the reference image. This algorithm delivers a virtual 2D silhouette SIL^{2D} in the image plane. Let us remember that the model is defined in such a way that a virtual silhouette is obtained for every possible pose. On the other hand, every projected

silhouette point belongs to the virtual silhouette. Instead of computing the local and semi-local features directly from the projected silhouette points, they are computed with respect to the virtual silhouette. This idea is shown in figure 5.6. For every 3D silhouette point $\underline{\mathbf{X}}_i \in \mathrm{SIL}^{3D}$, a contour segment in the 2D virtual silhouette is defined as

$$\mathbf{v}_i = \{\mathbf{x}_{i-n}, \cdots, \mathcal{P}(\underline{\mathbf{X}}_i), \cdots, \mathbf{x}_{i-n}\}, \quad \mathbf{x}_i \in \mathrm{SIL}^{2D}, \tag{5.3}$$

where n is the range of the segment and $\mathcal{P}(\underline{\mathbf{X}}_i)$ denotes the projection of the silhouette point onto the image plane.

Local and semi-local features can be computed from the segment $\mathbf{v}_i$ according to the procedures presented in section 4.4. For a projected 3D silhouette point $\mathcal{P}(\underline{\mathbf{X}}_i)$ and a detected image point $\mathbf{x}_j$, the following sets of local features are obtained

$$F_i^m = \left\{\alpha_i, t_i^x, t_i^y, s_i^{mod}\right\} \tag{5.4}$$

$$F_j^p = \left\{\beta_j, \parallel r_j^x \parallel, \parallel r_j^y \parallel, s_j^{img}\right\}, \tag{5.5}$$

where α_i, β_j are the local orientations of projected silhouette and image points respectively. The pairs $t_i^x, \parallel r_j^x \parallel$ and $t_i^y, \parallel r_j^y \parallel$ stand for the transition index and phase responses in x and y directions respectively. Finally, the structure coefficients s_i^{mod}, s_j^{img} define if projected and image points are part of straight, concave or convex segments.

In the examples of figure 5.7, the projected silhouette points were segmented according to the local structure with respect to the 2D silhouette. The upper row shows the projected surface model, the segmentation of the points in concave, convex and straight segments and the transition indexes in x and y directions. A contour extracted from a real image of the power socket model is shown in the middle row. The bottom rows shows the segmentation of the image contour and its corresponding phase responses. As can be seen in these examples, computing features with respect to the 2D silhouette will deliver suitable features. This makes possible to apply the structural ICP algorithm for the pose estimation of surface models. Additionally, the local features can be used to define the profile vectors of contour points needed for the definition of the correlation ICP algorithm.

5.2.2 Correlation as Similarity Criterion

Once that vectors containing local feature information are constructed, it is necessary to have a measure function to define a correspondence search criterion. A well known technique to find the statistical similarity between two variables is the computation of their correlation matrix. In the case of model and image contour segments, the correlation measures the similarity of these segments in terms of their local features. In contrast to the Euclidean distance measure, the correlation coefficient is invariant under linear transformations of the data. Two segments with

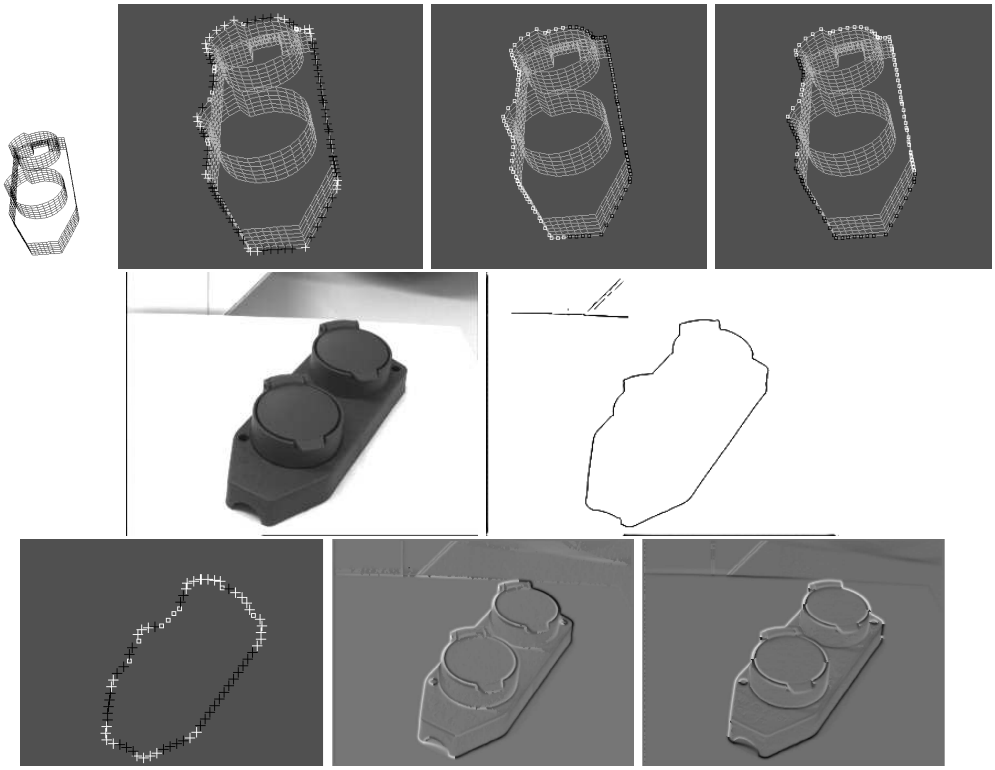

Fig. 5.7: Upper row: segmentation of a projected surface according to its semi-local structure (left) and transition indexes in x (middle) and y (right) directions. Middle row: real image of the power socket (left) and its extracted contour (right). Bottom row: semi-local structure of the image contour (left) and phase responses in x (middle) and y (right) directions.

identical feature structure, but different magnitude, will have a strong correlation after rotation and translation in the image plane.

Two profile vectors defined by the local orientation of model points α_i and image contours β_i are denoted by

$$\mathbf{o}_i^{img} = \{\beta_{i-n}, \ldots, \beta_i, \ldots, \beta_{i+n}\} \tag{5.6}$$

$$\mathbf{o}_j^{mod} = \{\alpha_{i-n}, \ldots, \alpha_i, \ldots \alpha_{i+n}\}\,. \tag{5.7}$$

The similarity of these vectors is computed by the correlation matrix as follows

$$\mathrm{corr}(\mathbf{o}_i^{img}, \mathbf{o}_j^{mod}) = \frac{\mathrm{cov}(\mathbf{o}_i^{img}, \mathbf{o}_j^{mod})}{\sqrt{V_{img}\, V_{mod}}}, \tag{5.8}$$

where $\mathrm{cov}(\mathbf{o}_i^{mod}, \mathbf{o}_j^{img})$ is the covariance matrix and V_{mod}, V_{img} are the respective variances of image and model local features. The correlation may vary in a range of $-1 \leq \mathrm{corr}(\mathbf{o}_i^{mod}, \mathbf{o}_j^{img}) \leq 1$, where -1 indicates perfect negative correspondence, 0 indicates no correspondence and 1 indicates perfect correspondence. An example of

two profile vectors containing the local orientation information of contour segments can be seen in figure 5.8. In the left graphic, the profiles are not strongly correlated, while in the right graphic the correlation is higher.

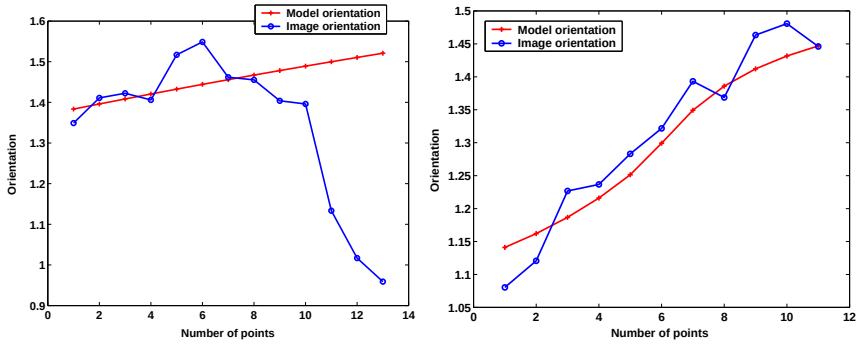

Fig. 5.8: Example of two not correlated (left graphic) and strongly correlated (right graphic) orientation profiles.

When the semi-local structure of contour segments is computed in the image plane (concavity, convexity and straightness), the neighborhood around each projected point is constructed by considering pixels around this point with respect to the virtual 2D silhouette. In the case of the profile vectors, larger neighborhoods are used to describe the structure (in terms of the local features) of a projected model point. The use of larger profile vectors implies that the description level of contour segments is extended from local features or semi-local structure to a feature structural description (see section 4.2).

In practice, contour points are sampled every s pixels from the projected model point to the left and right directions as illustrated in figure figure 5.9. The respective local features of these sampled points are taken to construct the profile vectors. This allows to describe larger contour segments without increasing the dimension of the corresponding profile vectors. Notice that larger profile vectors would increase the computation times of the correlation matrix. Thus, for an image or projected model point $\mathbf{x}_i$, its profile vector of range n is defined by $\mathbf{o} = \{\mathbf{x}_{i-(ns)}, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_{i-(ns)}\}$.

5.2.3 Correspondence Search Criteria based on Feature Correlation

The first possibility to use the feature correlation to find correspondences is to combine the Euclidean distance criterion with the correlation measure as an additional structural constraint. Thus, for an image point $\mathbf{x}_i^{img} \in \mathcal{S}^{2D}$ with its corresponding profile vector $\mathbf{o}_i^{img}$, its corresponding silhouette point $\mathbf{x}_j^{sil} = \mathcal{P}(\underline{\mathbf{X}}_j) \in \mathrm{SIL}^{2D}$ is found

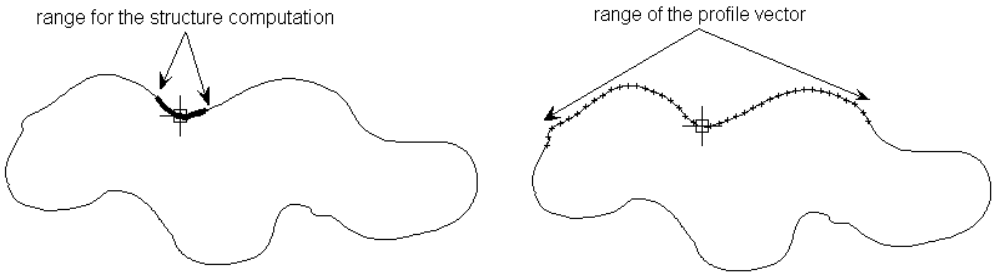

Fig. 5.9: Range of the neighborhood used for the computation of the semi-local structure (left) and to define the profile vectors (right).

by the next criterion

$$d(\mathbf{x}_i^{img}, \mathrm{SIL}^{2D}) = \min_{j=1\cdots n} \left\{ d(\mathbf{x}_i^{img}, \mathbf{x}_j^{sil}) \right\} \quad \text{if} \quad \mathrm{corr}(\mathbf{o}_i^{img}, \mathbf{o}_j^{sil}) > thres, \tag{5.9}$$

where the vector $\mathbf{o}_i^{sil}$ denotes the profile vector of the silhouette point $\mathbf{x}_j^{sil}$.

As can be seen in the last equation, the corresponding point will be the closest point whose correlation measure will be larger than a given threshold value. Similar to the normal variants of the ICP algorithm, the main correspondence search criterion is the Euclidean distance. This variant is useful in pose estimation scenarios where occlusions are present in the image. An example of this pose scenario is shown in figure 5.10. Under these conditions, several image contour segments have more similar feature profiles than the model segments. Then, a model point may have several possible correspondence candidates. Because of that, it is necessary to include the minimal distance criterion. The last implies that this variant may be applied for scenarios under tracking assumption conditions until certain limits.

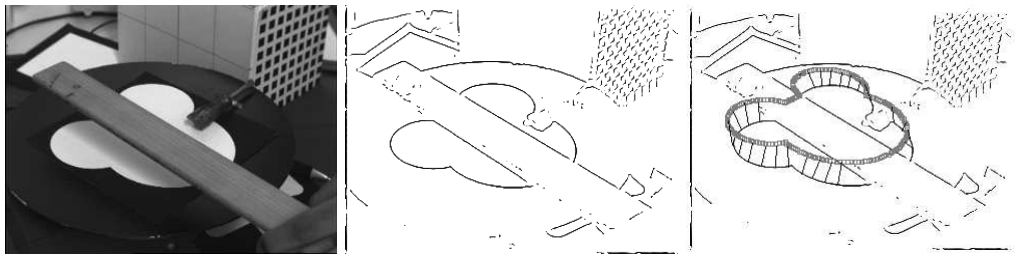

Fig. 5.10: Example of a contour model with occlusions (left) and the extracted image contour segments (middle). Correspondences by combining Euclidean distance with the correlation constraint (right).

In the second variant, the Euclidean distance is replaced by the correlation mea-

sure as follows

$$\begin{aligned} \mathrm{corr}(\mathbf{o}_i^{img}, \mathrm{SIL}^{2D}) &= \max_{j=1,\cdots,n} \left\{ \mathrm{corr}(\mathbf{o}_i^{img}, \mathbf{o}_j^{sil}) \right\} \\ &\quad \text{if} \quad C_1 \wedge (C_2 | C_3) = 1, \end{aligned} \tag{5.10}$$

where the constraints C_1, C_2 and C_3 are the structural constraints defined in section 4.5.1. According to that, two image and silhouette points form a correspondence pair if the correlation of their respective profile vectors is maximum. Additionally, it is possible to combine the structural constraints (phase-transition index, straightness, concavity and convexity) with the correspondence measure. Similar to the case of the structural ICP variant, the addition of these extra features will increase the probability to find better conditioned correspondence sets.

In the next examples, all correspondence search criteria (minimal distance, structural and correlation) were compared. Additionally, the correspondences with the feature invariant ICP algorithm, see [100], are obtained. Each variant was initially applied to projected 3D contours as shown in figure 5.11. It is evident that the computed correspondences with the minimal distance and the feature invariant criteria are bad conditioned. The feature invariant ICP algorithm uses the combination of Euclidean distance plus feature distance $d = d_E + \alpha d_f^1 + \alpha d_f^2$. In this case, the feature distances d_f^1 and d_f^2 are defined by the Euclidean distance measure of vectors containing local curvature and invariant moments respectively. Since these features are not invariant under perspective projection, the obtained correspondences are still bad conditioned.

Better correspondences are found with the structural variant. Despite of that, some correspondences are still bad conditioned. The best correspondences are found with the correlation criterion. A similar comparison was done to find correspondences between the projected surface model and the image contour of figure 5.12. The top row shows the correspondences with the minimal distance and feature invariant criteria. The correspondences found with the structural and correlation criteria are shown in the bottom row of the figure. As can be seen in the pictures, the best correspondences are found with the correlation variant.

An important parameter in the definition of the orientation profile vectors is the range of the neighborhood around each contour point. The correct choice of this parameter will have a direct effect on the quality of the computed correspondences. Since the image contour was artificially generated for these examples, the exact correspondences between the projected silhouette points s_i and image contour points p_i are known. As shown in figure 5.13, a vector containing the positions of the image contour points $p_1, p_2, p_3, \ldots, p_n$ is defined as $\mathbf{p}_{ini} = [1, 2, 3, \ldots, n]$. According to the computed correspondences, the sequence of this points may change to $p_1, p_3, p_2, \ldots, p_n$ and the vector containing the new point positions $\mathbf{p}_{end} = [1, 3, 2, \ldots, n]$ is defined. Finally, the error in the correspondence set is defined by the Euclidean distance of these vectors

$$\mathrm{E}(\mathbf{p}_{ini}, \mathbf{p}_{end}) = \|\mathbf{p}_{ini} - \mathbf{p}_{end}\|^2. \tag{5.11}$$

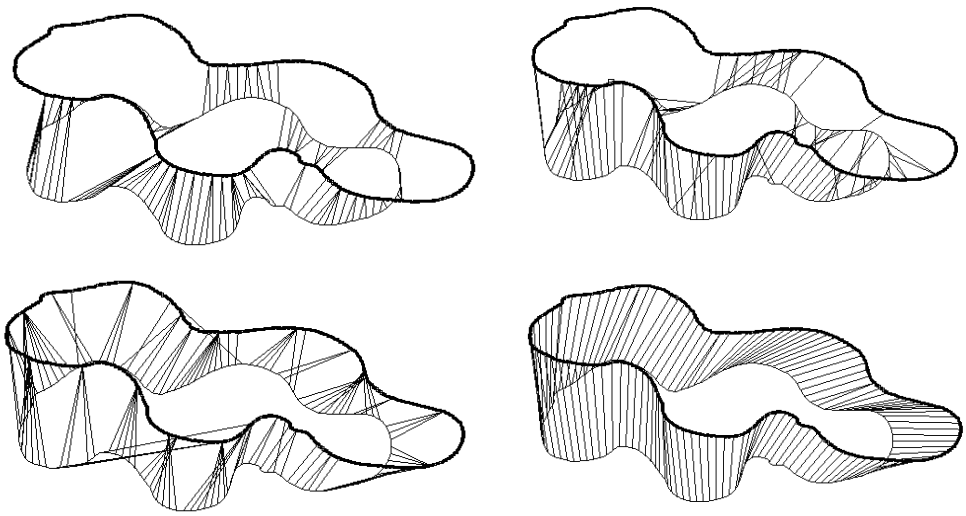

Fig. 5.11: Upper row: correspondences with the Euclidean distance criterion (left) and with the invariant features variant (right). Lower row: correspondences with the structural (left) and correlation (right) correspondence search variants.

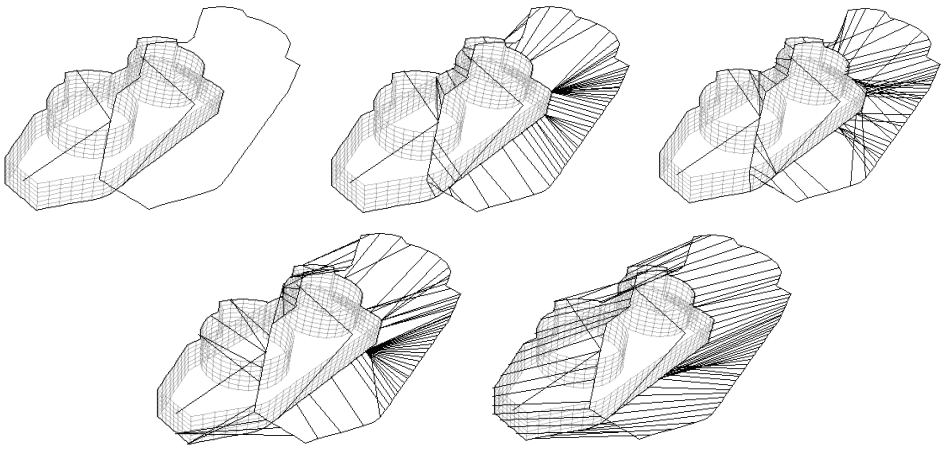

Fig. 5.12: Correspondences of a surface model and an artificially generated image contour. Upper row: correspondences with the minimal distance (middle) and invariant feature criteria (right). Lower row: correspondences with the structural (left) and correlation variants (right).

The left graphic of figure 5.13 shows the error of the correspondence computation when the range of the neighborhood was varied from 0 to 40 points. It can be clearly seen that the error is considerably reduced for range values larger than 20 pixels. In the next experiment, different levels of gaussian noise were added to the

contour points coordinates. The correspondences were computed with profile vectors of ranges 20 and 30 points respectively. The result of this experiment is shown in the right graphic of figure 5.13. Larger variations of the error are obtained for smaller range values. Although larger range values would increase the robustness against noise, the computation time of the correspondence would be also increased. From the results of both experiments, it can be concluded that the optimal range values to define the profile vectors are between 20 and 30 points.

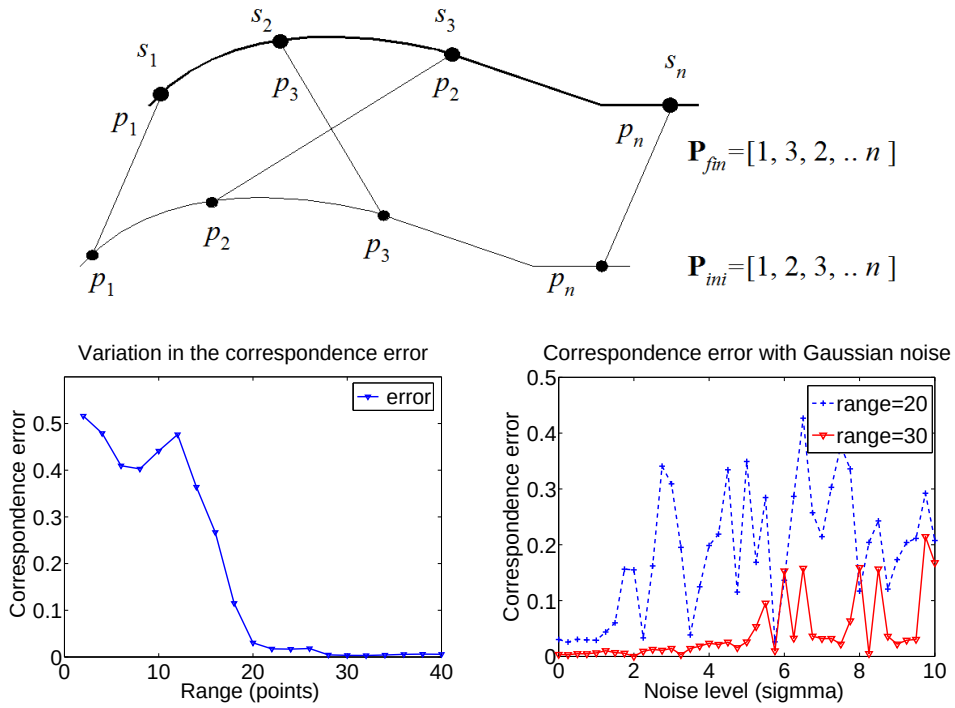

Fig. 5.13: Construction of the vectors for the computation of the correspondence error (upper figure), correspondence error for range values from 0 to 40 points (lower left figure) and correspondence error in the presence of Gaussian noise (lower right figure).

5.2.4 Silhouette-based Correlation ICP Algorithm

Once that the feature computation with respect to the virtual silhouette and the correlation search criterion have been described, the correlation ICP algorithm is summarized for the 3D-2D point-line and projective point-point minimization constraints of section 2.5.3. Similar to the silhouette-based ICP algorithm presented in section 5.1.2, the sets $\mathcal{M}^{3D} = \{\underline{\mathbf{X}}_i\}_{i=1}^{n}$ and $\mathcal{S}^{2D} = \{\mathbf{x}_i\}_{i=1}^{m}$ containing surface and image detected points are defined as input of the algorithm. The local features of the image points are computed and the set containing their respective profile vectors

$\mathcal{V}_{img} = \{\mathbf{o}_j^{img}\}_{j=1}^{n}$ is defined. If the point-line minimization constraint is used, the set containing the reconstructed optical rays $\underline{\mathbf{L}}_i$ from every image point is defined by $\mathcal{S}^{3D} = \{\underline{\mathbf{L}}_i\}_{i=1}^{n}$. For $j = 1$ to n iterations the following steps are repeated:

Silhouette-based Correlation ICP Algorithm

1. Extract the 3D silhouette $\mathrm{SIL}^{3D} \in \mathcal{M}^{3D}$ and the virtual silhouette SIL^{2D}.
2. Compute the local features of the projected silhouette points $\mathcal{P}(\underline{\mathbf{X}}_i)$ with respect to SIL^{2D} and define the set of profile vectors $\mathcal{V}_{sil} = \{\mathbf{o}_i^{sil}\}_{i=1}^{\mathrm{n}}$.
3. For every image point $\mathbf{x}_j \in \mathcal{S}^{2D}$ with profile vector $\mathbf{o}_i^{img} \in \mathcal{V}_{img}$, find its corresponding projected silhouette point $\mathcal{P}(\underline{\mathbf{X}}_i)$ with the search criterion

$$\mathrm{corr}(\mathbf{o}_i^{img}, \mathcal{V}_{sil}) = \max_{j=1,\cdots,n} \left\{\mathrm{corr}(\mathbf{o}_i^{img}, \mathbf{o}_j^{sil})\right\},$$

   and define the correspondence sets $\{\underline{\mathbf{L}}_i, \underline{\mathbf{X}}_i\}_{i=1}^{n}$ or $\{\mathbf{x}_i, \underline{\mathbf{X}}_i\}_{i=1}^{n}$.
4. With the defined correspondence sets, find the pose parameters $\mathbf{M}$ with the 2D-3D minimization constraint

$$0 = \lambda \left(\begin{pmatrix} \mathbf{M} & \underline{\mathbf{X}}_i & \widetilde{\mathbf{M}} \end{pmatrix} \quad \underline{\times} \quad \underline{\mathbf{L}}_i \right) \cdot \mathbf{e}_+$$

   or with the projective pose minimization constraint

$$0 = \lambda \left\{ \mathbf{x}_i \quad - \quad \mathcal{P}(\mathbf{M}\underline{\mathbf{X}}_i\widetilde{\mathbf{M}}) \right\}.$$

5. Actualize the position of the surface model points $\mathcal{M}^{3D}$ with the computed pose parameters $\mathbf{M}$.
6. Compute the average error $err = d(\mathcal{S}^{2D}, \mathrm{SIL}^{2D})$ between image points and silhouette points. If $err < thres$, exit; else goto 1.

5.3 Pre-Alignment of 3D Surfaces

Pre-alignment approaches are commonly used in the context of 3D surface registration problems. Sensor information is matched to a model object by applying variants of the classical ICP algorithm in most of the cases. If the distance between sensor and model data is too large, the ICP algorithm can not be applied anymore. In this case, the problem is divided in two steps. First, a rough approximation of the pose is computed by aligning model and sensor data in 3D space. Once that model and sensor data are aligned, the classical ICP algorithm can be applied to find the exact pose. The last implies that a pre-alignment step is used to gain tracking assumption conditions in order to ensure the computation of the real pose by applying the

ICP algorithm. In practice, the computation of the pre-alignment and the exact pose imply the estimation of two rigid body motions by two minimization processes.

When a pre-alignment approach is needed, global features of model and sensor data are used. In general, the global orientation and position of a set of 3D points are defined by its major and minor distribution axes and their respective centers of mass, see [20]. For specific pose estimation scenarios, thinning algorithms in 3D are applied to sensor and model points in order to reduce them to a 3D skeleton, see [78]. The sets of arms and joints defining the skeleton are used to find the needed correspondences to compute the rough pose.

In this section, the pre-alignment problem for the monocular pose estimation is introduced. According to the correspondence search strategies and pose estimation constraints defined in this and the last chapters, a variation of the classical pre-alignment approaches is introduced for the monocular pose estimation of surface models.

5.3.1 Pre-alignment for the Monocular Pose Estimation

Classical pre-alignment approaches are in general proposed for pose scenarios where model and sensor data are defined in 3D space. For the monocular pose estimation problem, some considerations must be done to apply these approaches. Let us remember that 3D-2D and projective pose minimization constraints are available. On the other hand, the correspondence search problem for contour and surface models can be solved in 3D space or in the image plane. Similarly, finding correspondences to perform the pre-alignment can be done either in 3D space or in the image plane.

The first variant is shown in figure 5.14. In this case, it is assumed that the major and minor distribution axes of the model in 3D space are known. Similarly, the main orientation axes of the detected image contour are obtained. Then, a plane $\mathbf{P}_m$ is reconstructed from the image major axis as it is shown in the figure. The rough pose is computed by aligning the model major axis to its corresponding reconstructed plane. Correspondence pairs are formed between points lying along the main axis of the model and the plane $\mathbf{P}_m$. With this set, a 3D point-plane minimization constraint can be applied to compute the rough pose, see [87]. If only the major axis is aligned, the pre-alignment is computed with certain uncertainty. Let us remember that normal pre-alignment approaches align three distribution axes in 3D. Nevertheless, this is not possible for the monocular pose estimation problem since only two axes are obtained from the image plane. If the minor axis of the image contour is also considered to reconstruct a second plane, a better alignment is obtained.

The pre-alignment can be also done in the image plane. The main idea is shown in figure 5.15. Once that the 3D silhouette of the surface is projected onto the image plane, its main axes are obtained. Similar to the pre-alignment in 3D space, the rough pose is computed by the alignment of the image and silhouette main axes. In

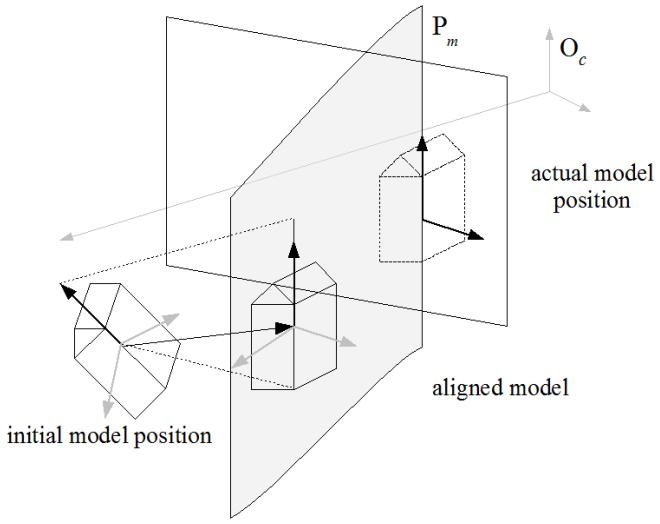

Fig. 5.14: Alignment in 3D space. The model is aligned to a plane which is reconstructed from the image.

this case, 2D point-point correspondences are defined with points along both axes. Finally, the rough pose is computed by the projective pose estimation constraints.

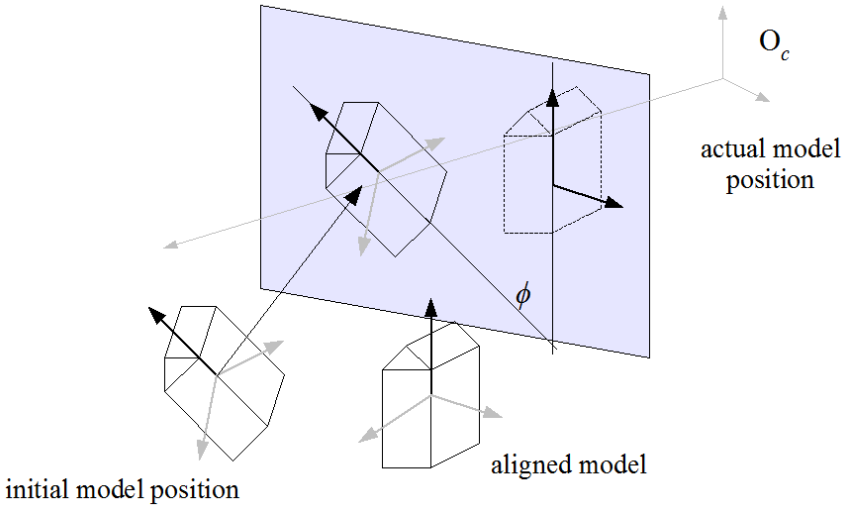

Fig. 5.15: Alignment in the image plane. The model is projected onto the image plane, where a 2D alignment is performed.

In both cases, the pre-alignment is done with respect to projected or extracted data from the image plane. In consequence, the computed motor $\mathbf{M}_r = \exp\left(-\frac{\theta}{2}\underline{\mathbf{L}}\right)$ describing the rough pose is formed by rotation and translation with respect to the

image plane. As can be seen in figure 5.16, the rotation axis of the motor $\underline{\mathbf{L}}$ is perpendicular to the image plane. On the other hand, the translational component of the motor is parallel to the image plane.

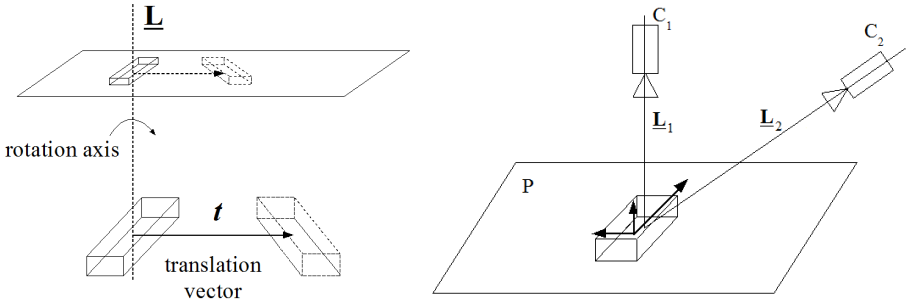

Fig. 5.16: Rotation axis and translation vector of the computed rough pose with respect to the image plane (left) and different positions of the camera with respect to the 3D surface model (right).

When the surface model moves parallel to the image plane, the result of the alignment with respect to the image may be better conditioned. To clarify that, let us suppose that the surface moves within a plane P as shown in the right picture of figure 5.16. At the camera position C_1, the computed rough pose results in a rigid body motion around the rotation axis $\underline{\mathbf{L}}_1$ which is perpendicular to image plane and to the plane P. At this position, the rough pose corresponds to the real movement of the surface model as long as the model is within the view range of the camera. At the position C_2, the model is aligned by a rotation around the axis $\underline{\mathbf{L}}_2$. This rotation aligns the main axes of the model with the main axes of the image contour. Nevertheless, it rotates the surface model outside of the plane P. Therefore, the computed alignment does not correspond to the real movement of the surface anymore.

Let us remember that it is only possible to reconstruct planes from the image or to project the silhouette onto the image plane. Both cases imply that the pre-alignment is done with respect to only two orientation axes. Therefore, the pre-alignment for the monocular pose estimation is performed with larger uncertainty than the pre-alignment in 3D space.

5.4 Feature Alignment in the Image Plane

As it was already mentioned in the last section, the pose estimation problem is divided in two steps for larger model displacements. In both steps, correspondences must be determined by a given strategy. Similarly, different pose minimization constraints must be applied for each step. Then, the pose estimation algorithm must switch between the pre-alignment and the fine pose estimation. Although this may

not directly imply a loss of quality of the computed pose, it is desirable to use only one pose estimation constraint. In this section, a strategy to simplify the pre-alignment computation is presented. In a first instance, the pre-alignment algorithm is reduced to one single iteration by constructing exact correspondences from projected silhouette points. This is done by considering the global position and orientation of contours in the image plane derived from the extracted distribution axes, see section 3.3.4. Finally, the correlation correspondence search criterion is combined with this global orientation in order to simplify the pre-alignment problem to a 2D feature alignment.

5.4.1 Exact Correspondences for Pre-alignment

For projected silhouette points $\mathcal{M}^{2D}$ and image points $\mathcal{S}^{2D}$, the following global features are obtained

$$G^{sil} = \left\{\theta^{sil}, \mathbf{m}^{sil}\right\} \tag{5.12}$$
$$G^{img} = \left\{\theta^{img}, \mathbf{m}^{img}\right\}, \tag{5.13}$$

with the global orientation angles of the projected silhouette and the image contour denoted as θ^{sil} and θ^{img} respectively. The global positions are defined by the respective centers of mass $\mathbf{m}^{sil}$ and $\mathbf{m}^{img}$. With this set of global features, the angle ϕ is defined as the difference of the projected silhouette and image contour orientations. On the other hand, the translation vector $\mathbf{t}_r$ is defined with the respective centers of mass.

The first step in order to simplify the pre-alignment computation is to reduce the number of iterations needed to compute the rough pose. As it was discussed in section 4.1.1, only one iteration is needed to compute the pose if an exact correspondence set is available. To achieve that, the projected silhouette points are simply rotated with respect to the image plane by the angle ϕ and translated by the vector $\mathbf{t}_r$. This is done for all projected points $\mathbf{x}_i \in \mathcal{M}^{2D}$ as

$$\mathbf{x}'_i = \mathbf{T}(\phi, \mathbf{t}_r)\mathbf{x}_i, \tag{5.14}$$

where the transformation $\mathbf{T}(\phi, \mathbf{t}_r)$ is defined by a homogeneous matrix representing a combined rotation and translation in the image plane. As shown in figure 5.17, the correspondence set is directly formed with the points $\{\mathbf{x}_i, \mathbf{x}'_i\}$ in the image plane. Then, the rough pose can be computed with one of the pose minimization constraints described in the last chapters.

Some examples of the pre-alignment with exact correspondences are presented in figure 5.18. The left column shows the initial position of the projected surface model and the detected image contour. The pictures also show the extracted main orientation axes. As can be seen in the middle column, each silhouette point is rotated and translated to define the exact correspondences. The computed rough

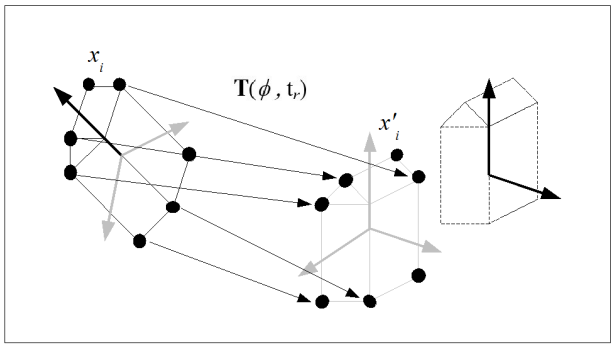

Fig. 5.17: Exact correspondence set for a projected surface model.

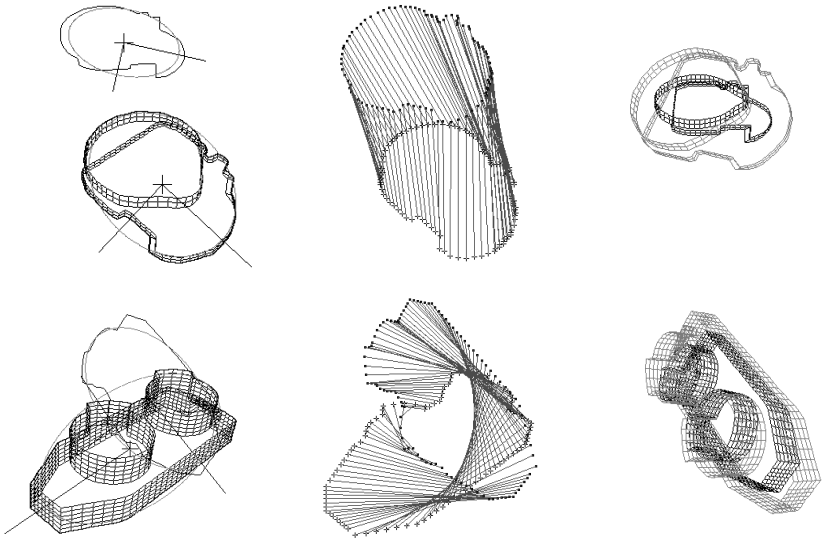

Fig. 5.18: Left column: initial position of the surface model with respect to the detected image contour. Middle column: sets of exact correspondences. Right column: computed rough pose.

pose is shown in the right column. It turned out that the rough pose computed with exact correspondences is practically the same as the pose computed by the alignment of the main axes with the approaches described in section 5.3. There is no significant difference between these variants for the presented examples. Despite of that, the pre-alignment computation has been reduced to only one iteration.

5.4.2 Local Orientation Alignment

The idea of the 2D orientation alignment is similar to that of the procedure described in the last section. This can be seen in figure 5.19, where the dotted object represents the projected surface model and the solid object is the detected contour in the image. Once that the corresponding main axes are computed, the 2D transformation matrix of equation (5.14) is defined with the global positions and orientations $\{\mathbf{m}^{mod}, \theta^{mod}\}$ and $\{\mathbf{m}^{img}, \theta^{img}\}$. Instead of using an exact set of correspondences, the projected silhouette points $\mathbf{x}_i$ are aligned (rotated and translated in 2D) by the matrix $\mathbf{T}(\phi, \mathbf{t}_r)$. With the aligned points $\mathbf{x}'_i$, the local orientation is computed and a set of aligned orientation profile vectors $\overline{\mathbf{o}}^{sil}$ is defined. Finally, the correlation criterion of equation (5.10) is applied with the aligned profile vectors as

$$\mathrm{corr}(\mathbf{o}_i^{img}, \mathrm{SIL}^{3D}) \;=\; \max_{j=1,\cdots,n} \left\{ \mathrm{corr}(\mathbf{o}_i^{img}, \overline{\mathbf{o}}_j^{sil}) \right\}. \tag{5.15}$$

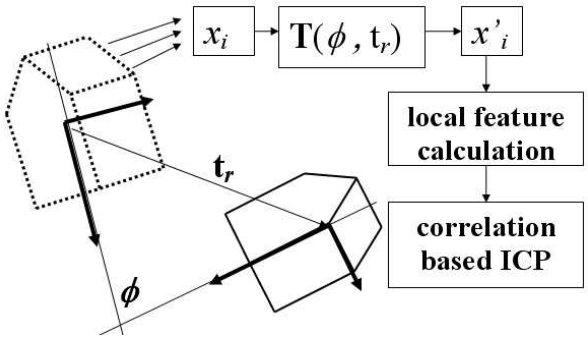

Fig. 5.19: 2D feature alignment based on the difference of global orientation angles and position in the image plane.

A comparison of the computed correspondences with the simple correlation criterion and with the feature alignment is shown in the top row of figure 5.20. The effect of the alignment in the orientation profiles can be seen in figure 5.21. In the graphic of the left, the superposition of silhouette and image orientation profiles is shown. If only the correlation criterion is applied, the computed correspondences are bad conditioned and in some cases are completely wrong. When the projected silhouette points are aligned, the orientation profiles are globally more similar (see right graphic of figure 5.21). Therefore, a better conditioned correspondence set is obtained. Finally, the computed rough poses with exact correspondences and with the feature alignment are shown in the lower row of figure 5.20. It can be clearly seen that a better rough pose is obtained with the orientation alignment.

In contrast to the classical pre-alignment approaches, the rotation and translation components of the transformation matrix $\mathbf{T}(\phi, \mathbf{t}_r)$ are directly computed from

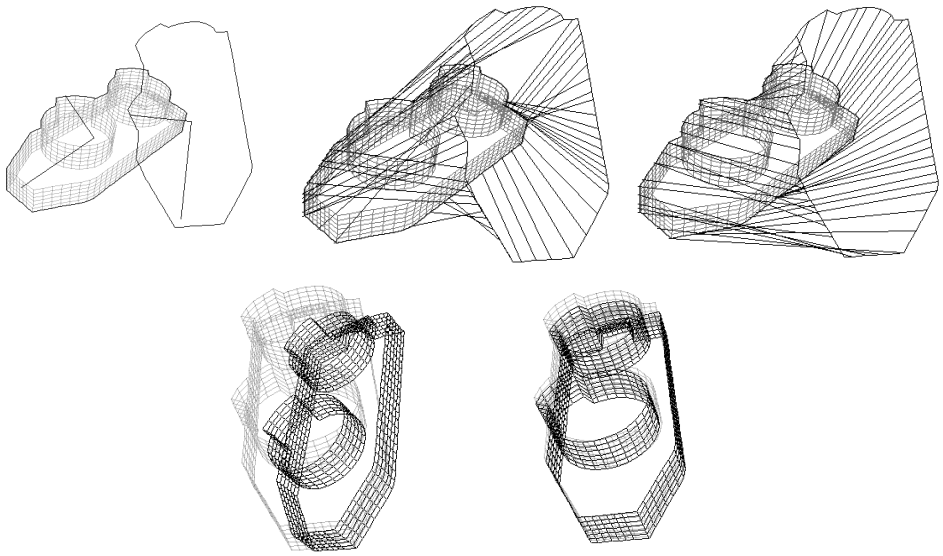

Fig. 5.20: Examples of correspondences between an object model and an image contour. Upper row: initial position of the surface model and image contour (left), correspondences with the simple correlation criterion (middle) and correspondences with aligned orientation profiles (right). Lower row: alignment results with exact correspondences (middle) and with aligned orientation profiles (right).

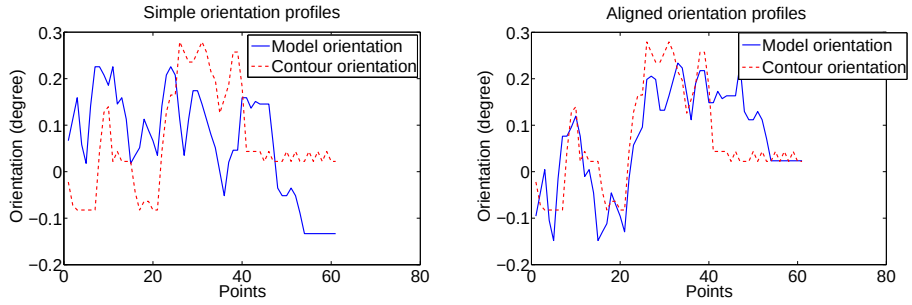

Fig. 5.21: Orientation profiles of two corresponding points with simple correlation (left) and with aligned orientation profiles (right).

the global orientation and position differences. In this approach, the pre-alignment is done only for the first iteration of the correlation ICP algorithm. Then, there is no need to switch to a different pose estimation constraint as done by the classical pre-alignment variants described in section 5.3. Conceptually, this approach can be considered as a combination of local and global features to solve the correspondence search problem.

5.5 Summary

In this chapter, the correlation based ICP algorithm was presented for the pose estimation of free-form surfaces. Instead of using the complete surface model, only the extracted 3D silhouette is considered. As a consequence, the pose estimation of free-form surfaces is reduced to the pose estimation of free-form contours for each iteration of the algorithm. A virtual 2D silhouette is generated for the visualization of the surface in the image plane and it is used for the silhouette extraction process. Since the global and local features are computed with respect to this virtual silhouette, it is possible to apply the structural and correlation ICP algorithms for non-planar free-form 3D contours and consequently for free-form surfaces. The minimal Euclidean distance search criterion of the classical variants of the ICP algorithm has been replaced with a feature correlation measure. A direct consequence of the use of the feature correlation is that better correspondence sets are obtained. Therefore, a better pose is computed in comparison to the normal ICP algorithm.

For cases where the tracking assumption is not considered, the pre-alignment problem for the monocular pose estimation was discussed. Global position and orientation features of projected silhouette and detected image contours were combined with local orientation information to simplify the pre-alignment step. To achieve that, a direct 2D feature alignment is done for the first iteration of the correlation ICP algorithm. This has the advantage that better correspondence sets are obtained also in the cases where a pre-alignment step is needed. In contrast to the classical pre-alignment approaches, no extra pose estimation algorithm is needed to compute the pre-alignment.

Chapter 6

EXPERIMENTS

So far the local model representations for 3D free-form contours and surfaces have been discussed. The global and local feature extraction procedures were introduced and finally, the proposed variants of the ICP algorithm were presented. In this chapter, several experiments to analyze the main properties of the pose estimation algorithms in a monocular pose estimation scenario are presented. Initially, the experimental setup used to generate a ground truth pose and artificial images for the different tests is presented. Then, the structural and correlation ICP algorithms are compared with the classical ICP algorithm. The behavior of the pose estimation algorithms during the iterative process and their robustness are analyzed for image sequences of different contour and surface models. An experiment of a pose scenario is presented where the camera is mounted on a robot arm. In this case, the purpose is to recover the position of the camera as it moves around the object. In a further experiment, the 2D pre-alignment approach is compared with the classical pre-alignment approaches for cases where the tracking assumption is not considered. Finally, examples of the application of the proposed algorithms to sequences of real images are presented.

6.1 Experimental Setup to Generate Artificial Images

In order to perform an analysis and a suitable comparison of the behavior of the proposed ICP algorithms, a ground truth pose must be generated. If the object and camera models are known, a synthetic image of the object in any desired position and environment can be created. Since only contour-based features are needed for the proposed variants of the ICP algorithm, the generation of an image where contour information can be extracted will be sufficient to perform representative experiments. Several contour and surface models with different characteristics and sizes have been used for the experiments as shown in figure 6.1. Models relatively rich in structure are used. For example, the cactus and planar puzzle models consist of several concave, convex and straight segments. On the other hand, contour models with less structure are also considered, e.g. the mouse model. The triangle and house models are defined by a single surface, while the power socket, motor part

and 3D puzzle models are defined by several surfaces.

The idea of generating a ground truth pose in 3D can be seen in figure 6.1. Once that the main distribution axes are extracted from the different contour and surface models, twist rotations are defined around each axis as well as translations along them. Then, the ground truth pose is defined as a combination of twist rotations and it is applied to the model. At this actual position, the model is projected onto the image plane to generate artificial images as shown in figure 6.2. Notice that these images allow to apply the monogenic scale space approaches described in section 4.3.1 in order to extract local image features. Therefore, it is possible to apply the structural or correlation ICP algorithms with these artificially generated images. Finally, the computed pose is compared with the ground truth for each iteration and at the end of the pose estimation algorithm. This allows to analyze the convergence behavior and the quality of the final pose.

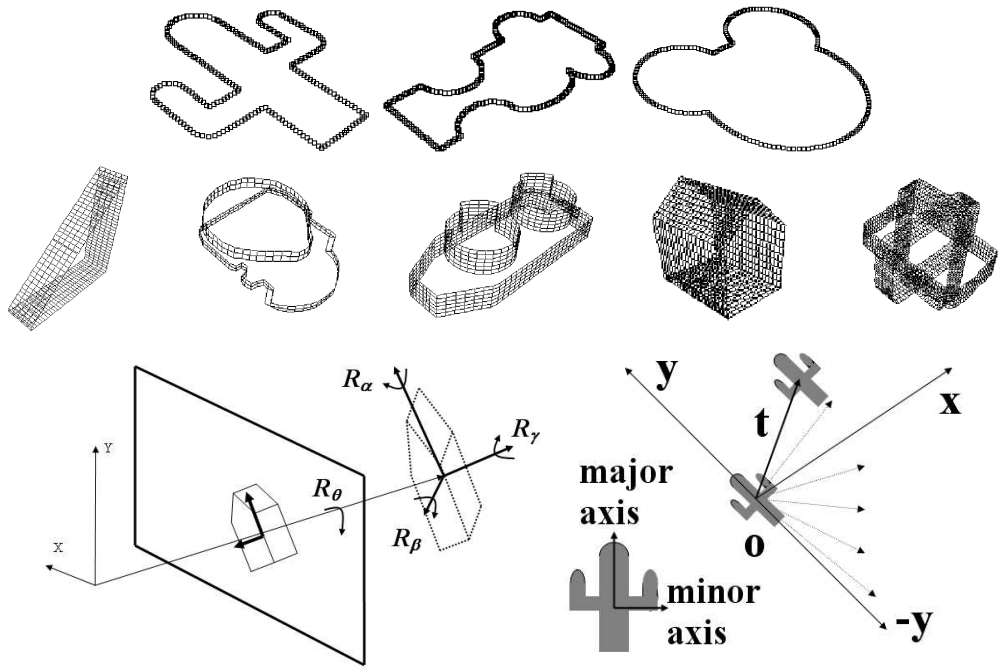

Fig. 6.1: Upper row: examples of contour models (cactus, puzzle and mouse). Middle row: examples of surface models (triangle, motor part, power socket, house and 3D puzzle). Lower row: setup for the experiments to generate the ground truth poses (rotation plus rotation).

6.2 Comparisons With the Classical ICP Algorithm

In this section, the experimental results regarding convergence behavior and robustness against the tracking assumption are presented. In a first instance, the computa-

Fig. 6.2: Examples of artificially generated images of a contour model (left image) and a surface (right image) model.

tion of the pose is analyzed for bad-conditioned correspondence sets when different pose estimation constraints are used. The structural ICP variant was compared with the classical ICP algorithm for the case of 3D free-form contour models. Similarly, the convergence and robustness of the correlation ICP algorithm are analyzed for the case of 3D surface models. Finally, experiments showing the robustness of these algorithms in the presence of noise and missing contours are presented.

6.2.1 Pose under Bad-conditioned Correspondences

This first experiment shows the behavior of the different pose estimation constraints introduced in the last chapters for the case of bad-conditioned correspondences. The two principal pose estimation algorithms described in section 2.5.3 were compared: the 2D-3D [93] (minimization in 3D space) and projective constraints [4] (minimization in the image plane). Additionally, the pose estimation algorithm called "orthogonal iteration" algorithm [71] was considered. Instead of the gradient decent methods used by the other variants, this algorithm performs a minimization in 3D space based on a singular value decomposition (SVD). The setup of this experiment is shown in figure 6.3. Initially, the pose was computed with a set of exact correspondences. Then, random false correspondences were added to the set as can be seen in the top row of figure 6.3. The pose was computed with the different pose constraints without searching the correct correspondences during the iterations.

The graphics show that the error of the pose increases as more bad-conditioned correspondences are added to the set. According to the graphics, the orthogonal iteration algorithm is more sensitive to bad correspondences in comparison with the other algorithms. On the other hand, the 2D-3D and projective constraints are less sensitive to bad correspondences. In both graphics, the absolute error of the projective constraint is smaller than the error computed with the 2D-3D variant. These results show that the minimization in the image plane is more robust against bad-conditioned correspondences than the other variants.

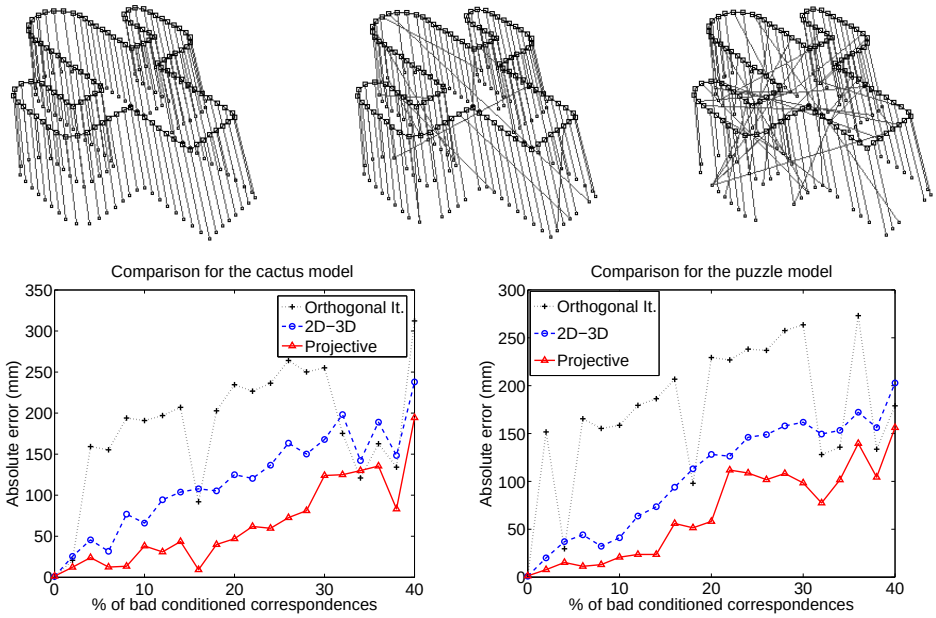

Fig. 6.3: Upper figures: examples of the generated bad-conditioned correspondences of the cactus model. Lower figures: comparison of the computed pose for the different pose estimation algorithms for the cactus (left) and puzzle (right) models.

As discussed in section 2.5.4, the motor $\mathbf{M} \in \mathcal{G}_{4,1}$ expressing the rigid body motion is approximated by its Taylor series power expansion in order to define linear equations with respect to the motor parameters. According to that, a linear system of equations of the form $A\mathbf{x} = \mathbf{y}$ is obtained, where the vector $\mathbf{x}$ contains the pose parameters. In the case of the 2D-3D pose constraint, the pose parameters are found by minimizing a 3D point-line error measure. According to the linearized equations, see [87], the dimension of the matrix A increases in three additional rows for every correspondence pair. For the projective variant, a 2D point-point error measure in the image plane is used. Therefore, the dimension of A increases only two rows for every correspondence pair, see [4]. With the same number of correspondences, the dimension of the linear system derived with the 2D-3D constraint is larger than the one of the projective constraint. Numerical solutions of linear systems based on standard approaches of like Householder or QR decomposition methods [47] are able to solve systems of higher dimensions in a proper and stable way. Despite of that, these approaches are more sensitive to numerical errors as the dimension of the system increases.

Let us remember that the algorithm converges to a local minimum in the presence of bad-conditioned correspondences. As the linear system is defined, each bad-conditioned correspondence implies that bad-conditioned rows are also added to the matrix A and to the vector $\mathbf{y}$. For the same number of bad-conditioned corre-

spondences, a reduced number of bad-conditioned rows are added to A in the case of the projective constraint (two instead of three for the 2D-3D constraint). In consequence, the uncertainty in the pose computation is smaller with the projective pose estimation constraints.

6.2.2 Convergence Behavior

In the sequence of images of figure 6.4, the convergence behavior of the classical ICP algorithm is compared with the structural variant. Each image shows the resulting pose and the correspondence pairs (denoted by the lines) for certain iterations. The normal variants of the ICP algorithm consider only the Euclidean distance as correspondence search criterion. Thus, many bad-conditioned correspondences are found in the first iterations and therefore the convergence is slower. In some cases, the algorithm does not converge at all. The structural variant also considers semi-local features (concavity, convexity and phase-transition index) as extra correspondence search criteria, see section 4.5.1. Although some bad-correspondences are also found in the first iterations, the use of the structural constraints increases the probability to find better conditioned correspondences during the iterations. Therefore, the convergence rate of the algorithm is increased in every iteration.

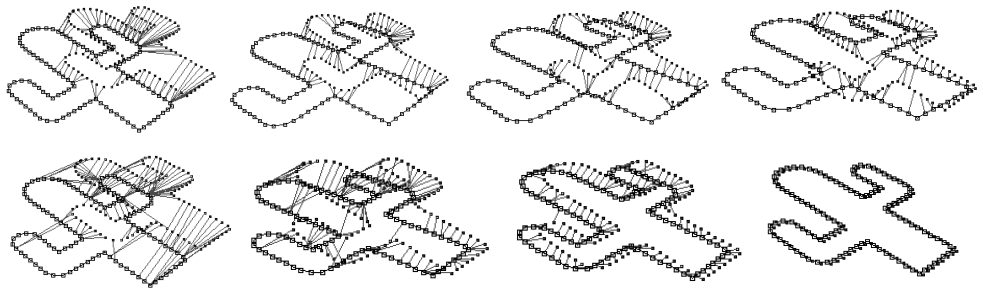

Fig. 6.4: Convergence sequence for normal (first and third rows) and structural (second and fourth rows) ICP variants applied to the cactus model.

The same comparison was made for the case of surface models. In the sequences of figure 6.5, an example of the convergence is presented when the displacement of the model and image data is not large. In this case, the initial position of the model is denoted by the gray surface and the ground truth by the dark surface. As can be seen in the example, the normal ICP algorithm does not converges to the correct pose. On the other hand, the structural ICP algorithm converges to the correct pose. The same behavior can be observed for the 3D puzzle model, although it has a more complex structure than the power socket model. For the presented examples, the ICP algorithm does not converge even if the pose is computed for a larger number of iterations. It can be considered that the tracking assumption is not met for these

examples. Then, the structural ICP algorithm is more robust against the tracking assumption than the normal ICP.

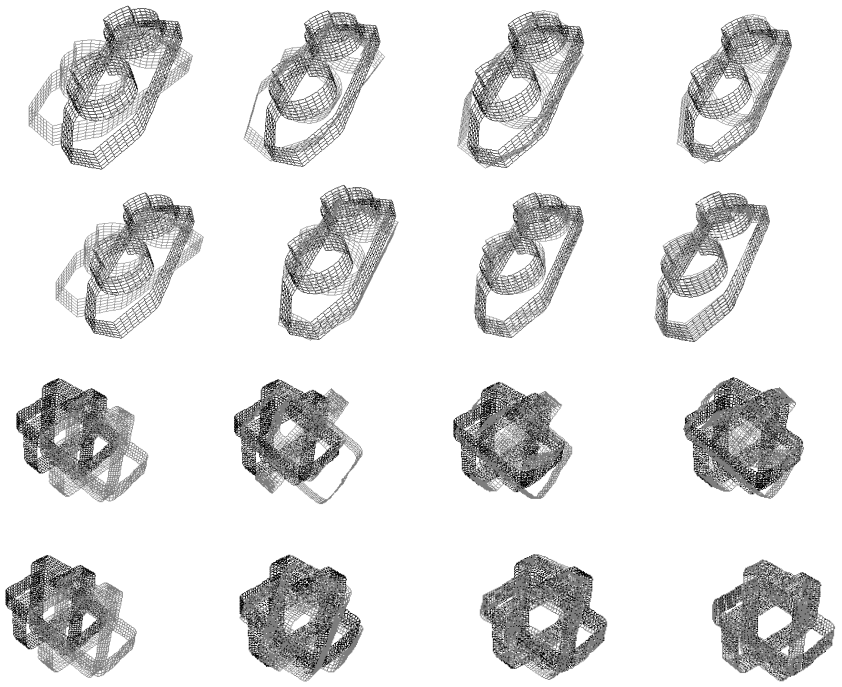

Fig. 6.5: Convergence sequence for normal (top row) and structural (bottom row) ICP variants applied to surface models.

A quantitative comparison of the convergence behavior can be seen in figure 6.6. For these experiments, a rotation around the axis perpendicular to the planar contour was defined. Then, the ground truth pose was defined by a rotation around this axis. The ground truth was defined small enough to ensure that both variants of the ICP algorithm converge to the real pose. Once that the correspondence sets were found, the pose was computed with two different pose estimation algorithms: the 2D-3D [93] and projective [4] ones.

The graphics show the absolute error of the computed pose for every iteration when the correspondence search directions defined in section 4.5.3 are used: model to image, image to model and combined search. The error was measured by computing the average Euclidean distance in 3D between the model points at the ground truth position and the model position for every iteration. The graphics of the left column show the results for the 2D-3D minimization approach, while the results for the projective algorithm are shown in the right column. As expected from the experiments of the last section, the computed pose errors are in general smaller for the projective algorithm. The structural ICP algorithm improves the convergence behavior of the algorithms. Less iterations are needed for the algorithm to converge

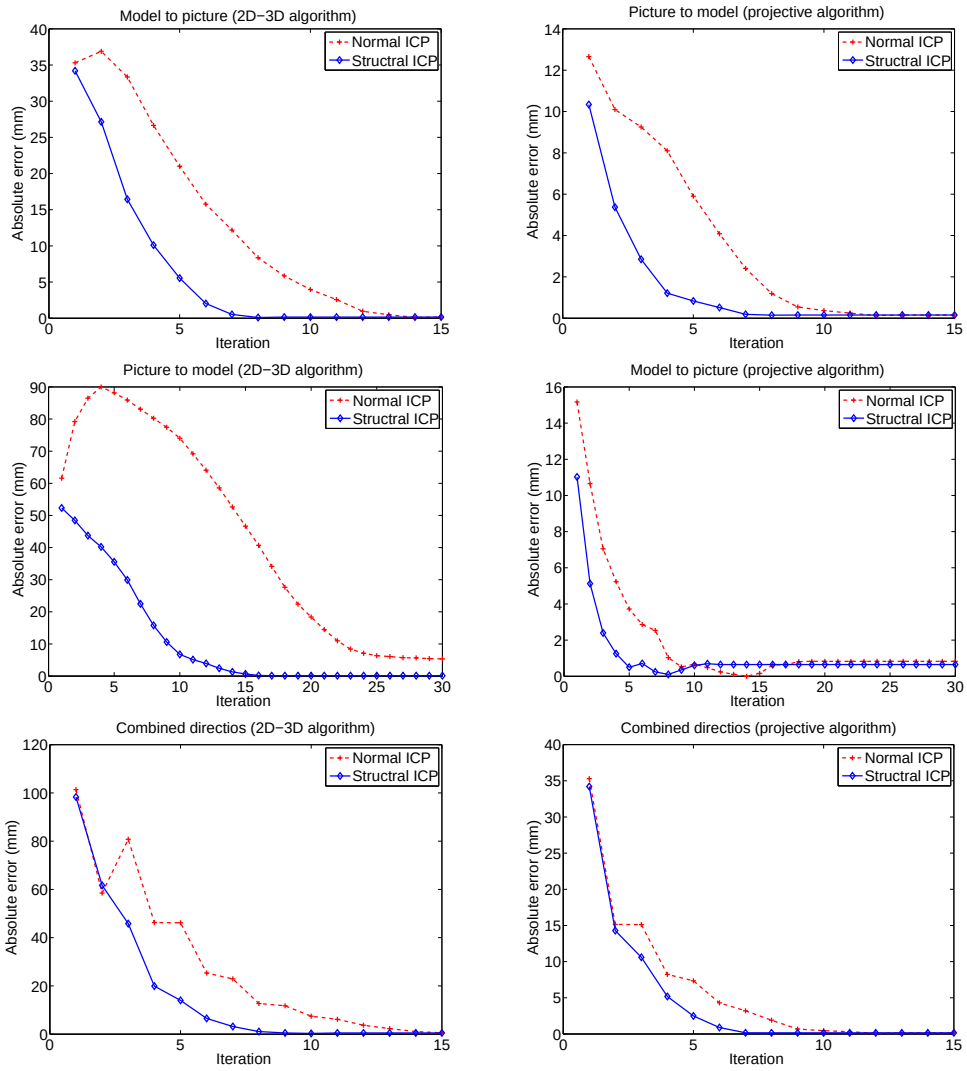

Fig. 6.6: Comparison of the convergence behavior for the 2D-3D algorithm (left column) and for the projective algorithm (right column).

to the real pose than the classical ICP variant. As can be seen in the graphics, the convergence during the iterations differs depending on the correspondence search direction (model to picture or vice versa). For a given direction, the correspondence search may be better or worse conditioned. The graphics of the lower row show the comparison when the search direction is switched for every iteration.

An interesting effect can be seen in the example of combined search directions for the 2D-3D algorithm. For the third iteration, the correspondence search is bad-conditioned and therefore the pose error increases with respect to the previous iteration. In the next iteration, the search direction is changed. Better conditioned correspondences are found and the error decreases. If the search direction is switched, a better convergence behavior can be achieved for both ICP variants and both pose

estimation algorithms. Nevertheless, the structural ICP variant still shows a better performance.

In the next experiment, the convergence behavior of the correlation ICP algorithm was compared against the classical and structural variants for the case of surface models. It was already shown that the projective algorithm delivers a better pose than the other variants. Then, only this variant is considered for this experiment. At the initial position of the surface model, its main orientation axes were extracted in 3D. Each one defines the rotation axes α, β and γ as can be seen in figure 6.1. The model was rotated by -30 degrees around the γ axis to generate the ground truth pose. In this new position, the corresponding artificial image was generated and the pose was computed with the three variants of the ICP algorithm.

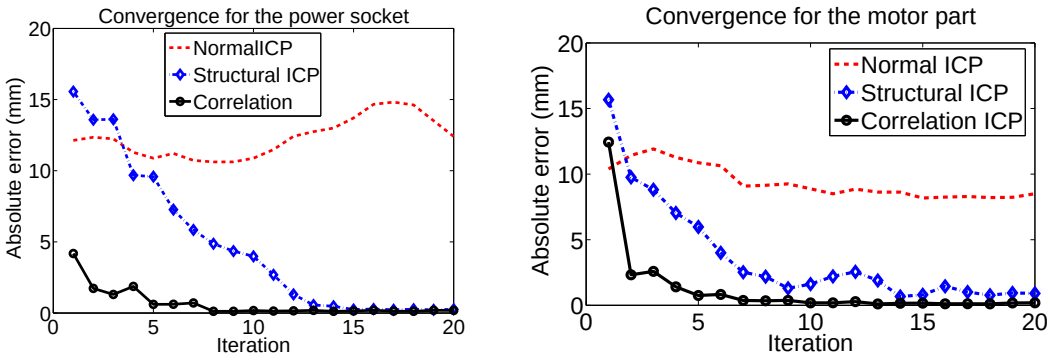

Fig. 6.7: Convergence behavior comparison for the power socket (left) and motor part (right) models.

The graphics of figure 6.7 show a comparison of the convergence behavior for the motor and power socket models. Because of the rotation around the different axes, the local structure of the 2D silhouette changes with respect to the initial position. For this example, the normal ICP algorithm was not able to converge anymore to the ground truth pose as can be seen in the graphics. In contrast to that, the structural ICP algorithm allows to recover the correct pose. Then, the extra structural information provide enough well-conditioned correspondences during the iterations to overcome the change of the silhouette caused by the rotation. By considering the semi-local structure (concavity, convexity and phase-transition index), the correspondences are in general well-conditioned but not very precise. However, the algorithm needs more iterations depending on the amount of the rotation (around 15 for these examples). On the other hand, the number of iterations needed to converge to the ground truth pose is significatively reduced with the correlation ICP algorithm (around 7 iterations). If the orientation profiles of contour segments are considered, better correspondences are found during all iterations and the algorithm converges faster than the structural ICP variant.

6.2.3 Tracking Assumption

The next experiments were made to test the robustness of the new ICP algorithms against the tracking assumption. Different tests have been done to compare the convergence ranges of the algorithms. That mens, the amount of rotations and translations with respect to the initial position within which the algorithms are able to converge to the ground truth pose. In a first instance, the case of pure translations was tested. As can be seen in figure 6.1, the contour model was translated to all directions in the plane where it is defined. For every position, the absolute pose error of the classical and structural ICP algorithms was compared. Similar to the convergence experiments, the projective and 2D-3D pose estimation algorithms were used. The comparison results for the puzzle model are shown in the graphics of figure 6.8. Analog to the experimental results of the convergence case, the structural ICP algorithm allows larger translations in both x and y directions. This can be observed for both pose minimization algorithms. Furthermore, the robustness is significatively improved with the combination of the structural ICP algorithm and the projective pose estimation.

An interesting effect can be seen in the graphic corresponding to the translation along the x direction with the projective algorithm. In general, the structural ICP algorithm allows a larger translation range (up to 40 mm against 12 mm with the classical ICP). There is an uncertainty region from 40 mm to 74 mm within which the error of the structural variant is still significatively reduced. Despite of that, the algorithm does not converges to the correct pose. After 74 mm, the algorithm tends to converge again. For this region, the semi-local structure is not enough to generate well-conditioned correspondences and to compute the correct pose. Let us notice that only one search direction was used for this experiment. One possibility to avoid such uncertainty regions is to switch the search direction in case that the error increases during the iterations.

The 2D graphics of figure 6.9 show the convergence regions of the structural ICP algorithm when translations in all directions are applied. This graphics show that the convergence region of the algorithm depends on the symmetry of the objects and on their complexity in terms of their semi-local structure. For the cactus model, the algorithm is more sensitive to translations in the y direction. This corresponds to translations in the major axis direction (see figure 6.1), while relatively large translations are allowed in the x direction (minor axis direction). The same effect can be seen for the puzzle model. Since both models are larger in their major axis directions, more segments with the same local structure can be found along them. Model and image points are initially perfectly aligned. Therefore, the algorithm finds almost perfect correspondences for smaller translations. As the model moves along the major axis, the pose error tends to increase. This uncertainty is smaller along the minor axis, since there are less points with the same semi-local structure in this direction. Thus, the convergence region is larger.

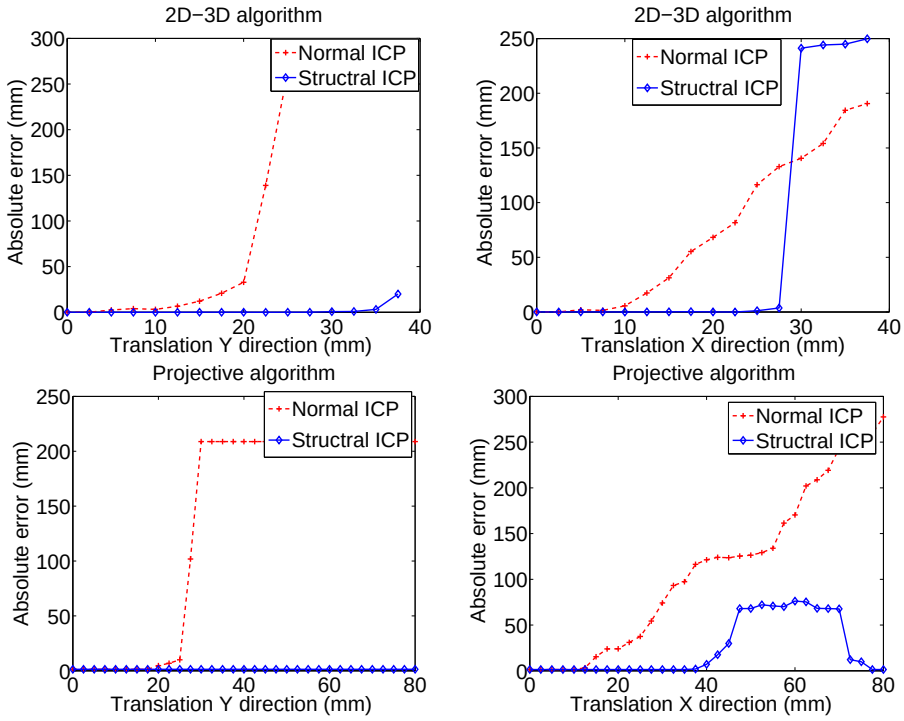

Fig. 6.8: Comparison of the robustness against translations for the contour case with the 2D-3D (upper graphics) and projective algorithms (bottom graphics).

The puzzle and cactus models are complex objects with enough structure to deal with relatively large translations. In contrast with these contour models, many segments of the mouse model have the same local structure. The effect on the convergence region can be seen in the bottom graphic of figure 6.9. Many correspondence pairs found with the structural ICP algorithm are similar to the correspondences found with the normal ICP algorithm. Therefore, the region of convergence is smaller and more irregular than the one of the other contour objects. Consequently, the error increases considerably for large translations.

In order to test the robustness against rotations, the ground truth pose was generated by rotating the contour model from zero to 50 degrees around its z axis, which is the perpendicular axis to the plane in which the contour is defined. As can be seen in figure 6.10, the pose error with the structural ICP algorithm is minimal for rotations up to 30 degrees for the 2D-3D algorithm and up to 40 degrees for the projective one. The convergence regions of the classical ICP algorithm are smaller in both cases. This shows that the structural ICP variant allows larger model rotations than the normal ICP.

The last experiment was made to compare the robustness of the correlation ICP algorithm with the other variants for surface models. In this experiments, the power socket model was used. If the surface model is translated along the different axes,

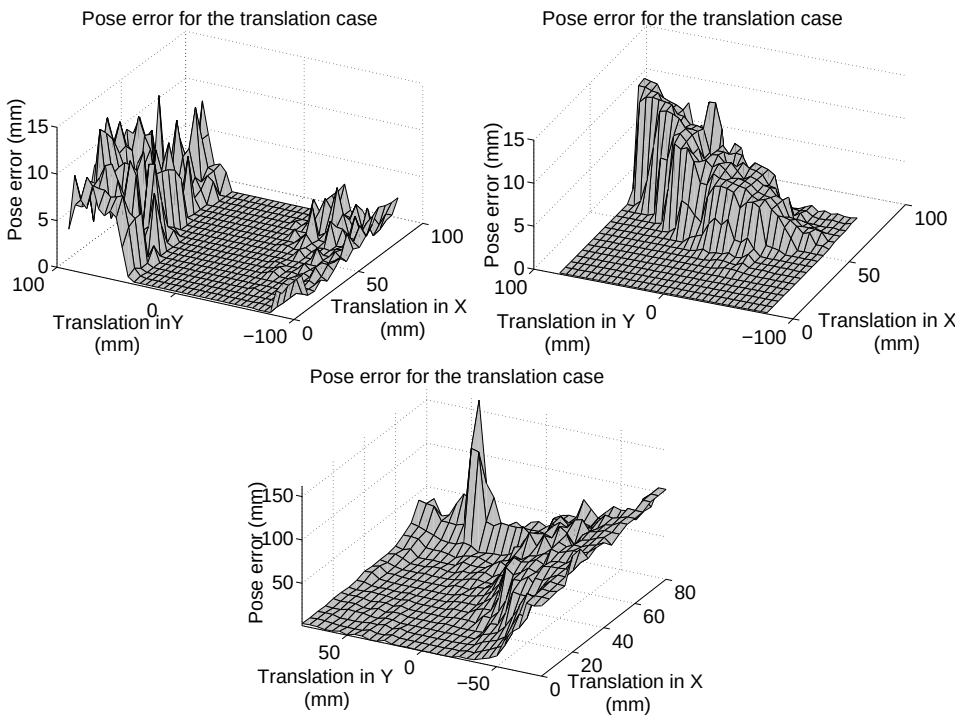

Fig. 6.9: Pose error for the translation case for the cactus model (top left), for the puzzle model (top right) and for the mouse model (bottom).

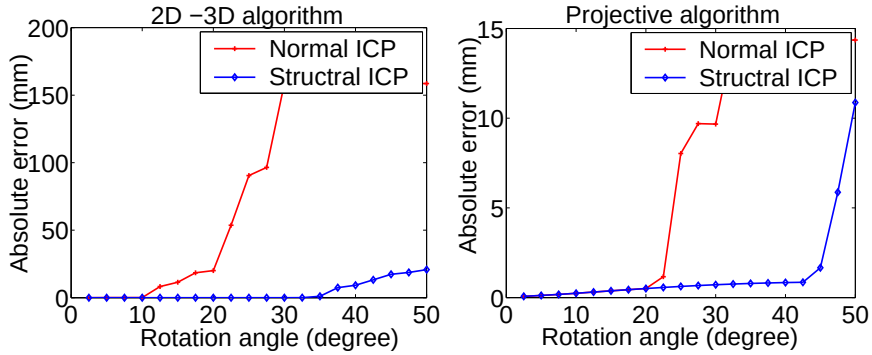

Fig. 6.10: Comparison of the robustness against rotations for the contour case with the 2D-3D (left) and projective (right) algorithms.

its 2D silhouette does not changes significatively with respect to the image plane. Thus, similar results as the ones obtained for the contour case can be obtained. As it was already mentioned in the last chapter, the structure is computed with respect to the 2D silhouette of the surface model. In the case of a rotation, the 2D silhouette and its corresponding structure change with respect to the image plane. Because of that, it is interesting to analyze the ranges (maximal rotation ranges) within which

the algorithms are able to converge to the ground truth pose. Thus, only the rotation case is considered for these experiments.

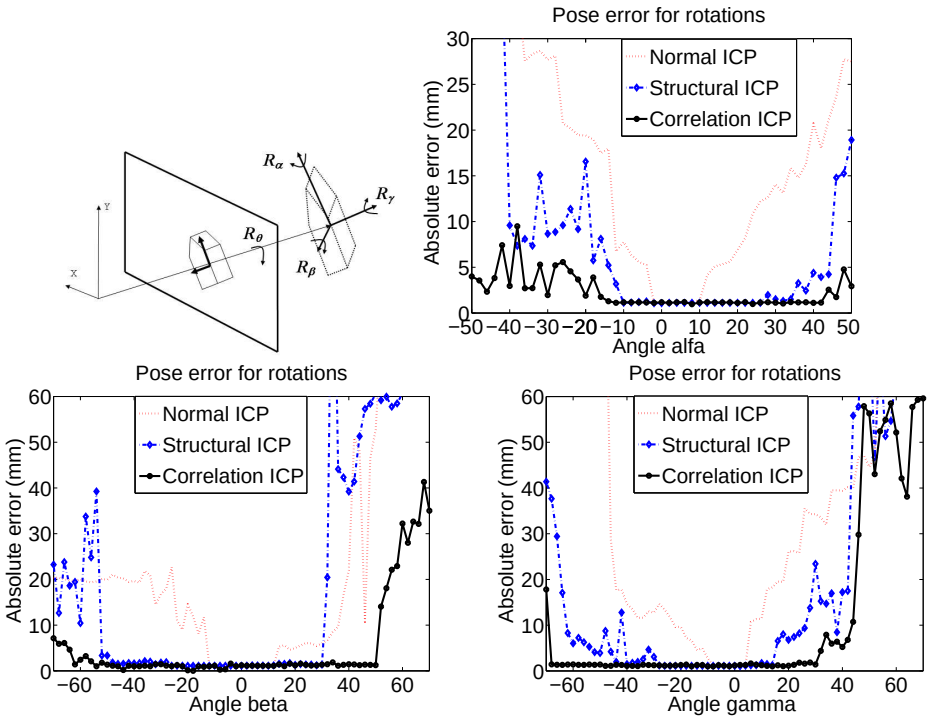

Fig. 6.11: Setup for the experiment for the 3D surface case with the correlation based ICP algorithm (upper left). Convergence ranges for rotations around the α (upper right), β (bottom left) and γ (bottom right) axes.

Figure 6.11 shows the comparison of the convergence ranges for the power socket model when the classical, structural and correlation ICP algorithms are applied. Rotations from -60 to 60 degrees around the β and γ axes were applied to the model to generate the ground truth. The rotation range around the α axis goes from -50 to 50 degrees. In the case of rotations around α and γ, the extracted silhouette changes drastically with respect to the image plane as the angle increases. Therefore, the region where the algorithms converge is smaller. The β axis is oriented in direction to the image plane (not exactly perpendicular). Rotations performed around this axis does not cause larger changes in the model silhouette and therefore, the convergence region is larger than the other cases.

In general, the classical variant of the ICP algorithm converges for relatively small rotations around all axes. These results show that the use of the classical ICP algorithm implies a strong dependence to the tracking assumption in the case of free-form surfaces. Similar to the results of the 3D contour experiments, the structural ICP variant shows larger convergence ranges for all cases. An even better performance is achieved with the correlation ICP algorithm. In some cases, the ranges

are significatively increased as can be seen in the graphic corresponding to the rotation around the γ axis. Although the power socket model is symmetric with respect to its major distribution axis, its projected 2D silhouette is not symmetric in the image plane. Because of that, the convergence regions in the graphics are not symmetric for positive and negative rotation angles.

6.3 Pose with Noisy and Missing Contours

The next experiments show the behavior of the different ICP algorithms in the presence of noisy contours and in the case where incomplete contours are considered. In order to have a better comparison, the rigid body motion used to generate the ground truth ensures the convergence of all the variants of the ICP algorithm. As described in section 5.2.2, a range value of 30 pixels was used to compute the semi-local features of image contours and projected silhouettes for this experiment. The same number of points are used to define the profile vectors with a sampling step of 5 pixels.

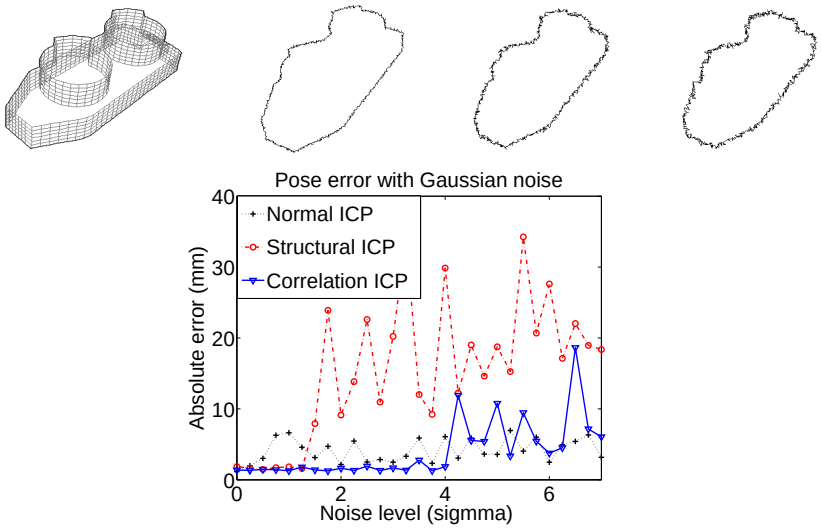

Fig. 6.12: Upper row: power socket model and image contours with different levels of noise (from right to left $\sigma = 1$, $\sigma = 4$ and $\sigma = 7$ respectively). Lower row: comparison of the computed pose with noisy contours for the normal, structural and correlation ICP variants.

Once that the image contours were extracted, Gaussian noise was added as can be seen in the examples of figure 6.12. Local and semi-local features were extracted from the noisy contours. By adding Gaussian noise to the contours, the local structure is evidently distorted. As discussed in section 4.4.2, the robustness against noise depends on the range of the neighborhood used to compute the features around a

point. Larger contour segments allow to obtain more robust features and better correspondences. Therefore, the pose computation is more robust as shown in the graphic of figure 6.12. The graphic shows that the correlation ICP variant is more robust against noise than the classical and structural ICP variants. Since larger contour segments are described by the profile vectors, the pose can be computed with larger levels of noise. In the case of the normal ICP algorithm, the error of the estimated pose does not show a larger variation when the level of noise is increased. The normal ICP algorithm finds the closest image point to every model point without considering the structure. On the other hand, the Gaussian noise distorts all points in x and y directions. Thus, the algorithm always finds a correspondence point (a distorted point) relatively close to the real correspondence.

For some pose estimation scenarios, it is not always possible to use the complete contour extracted from the image. For example, when partial occlusions are present in the image. In these cases, some segments of the detected contour must be eliminated as outliers. The next experiment was done to test the ability of the ICP algorithms to compute a correct pose if incomplete contours are considered. Once that the image contour was extracted, a segment of it was deleted. With the remaining points, semi-local features and profile vectors were extracted. The pose was computed with the normal, structural and correlation ICP variants. The results are shown in figure 6.13 for the power socket and motor part models. Similar to the results of the experiment with noisy contours, the correct pose can be still recovered with the structural and correlation ICP algorithms for larger missing segments.

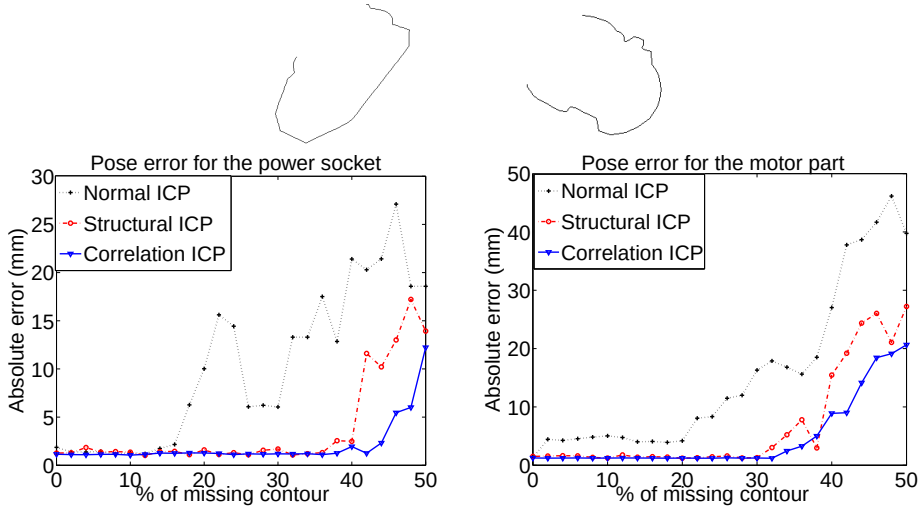

Fig. 6.13: Results of the missing contour experiment for the power socket (left) and motor part (right) models.

For points close to the begin and end sections of the partial contour, it is not possible to consider contour segments of the same length than segments in the middle section of the contour. The semi-local features and profile vectors must be computed

from smaller segments. Therefore, the features are less robust in such sections. Because of that, the difference of the pose error between the structural and correlation ICP algorithms is quite similar for this experiment.

6.4 Pose with a Robot Arm

In this section, an experiment is presented where the silhouette-based pose estimation algorithm was applied in the case that the camera in mounted on a robot arm. This experiment was made in order to include the silhouette-based pose estimation algorithm in the final demonstrator of the VISATEC project. The VISATEC project (Vision-based Integrated Systems Adaptive to Task and Environment with Cognitive abilities) was developed in collaboration with the Christian Albrechts University of Kiel (Germany), the University Linköping (Sweden) and ISRA Vision System AG (Germany) (see [103]). The aim of this project was to develop a system with cognitive capabilities able to recognize objects by using visual information in an optimal manner depending on the specific task and environment. Several new techniques for feature extraction, object recognition and classification applications were developed during the project. These new algorithms, as well as all the libraries for the control of the robot and video cameras, were integrated in the software infrastructure PACLlib [82] (Perception Action Libraries).

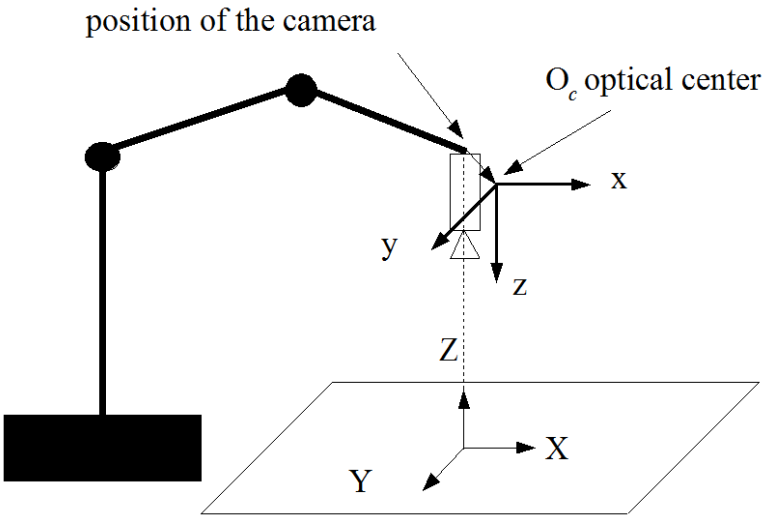

Fig. 6.14: Scenario of the experiment for the final demonstrator of the VISATEC project.

As can be seen in figure (6.14), the camera was mounted on the robot arm where its initial position is known. Initially, the camera was aligned with the world Z axis. In this position, the camera was calibrated with respect to the world coordinate

system. All camera parameters were extracted from the calibration matrix (internal plus external) including the coordinates of the projection center $\mathbf{O}_c$ (see [45]). The object was placed in the origin of the world coordinate system and a sequence of images was acquired. Notice that the algorithm computes the pose of the object with respect to the world coordinate system.

For this experiment, the camera was moved around the object with the trajectory being on a half-sphere (with respect to the world coordinate system). That means, the camera was initially rotated around the Y axis (pitch rotation) from 0 to 30 degrees with a step of 5 degrees. Then, it was rotated around the Z axis (yaw rotation) from 0 to 360 degrees with the same step width. In this particular case, the camera moves in the space while the object remains static. In terms of the pose estimation, getting the pose with respect to the camera or the world coordinate system is an equivalent problem under some considerations. Let us notice that the camera and end effector move simultaneously. Then, the position of the optical center also moves describing the trajectory within the half-sphere.

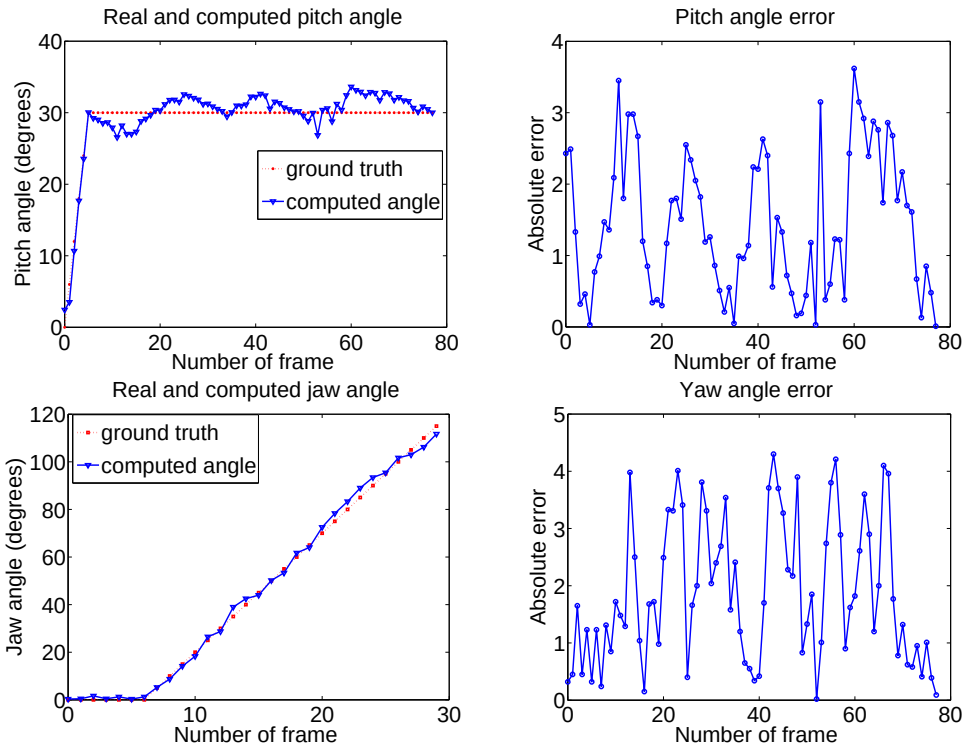

Fig. 6.15: Pose results for the test done with the motor part sequence. Comparison of the computed pitch and jaw angles with the ground truth (left column) and absolute errors for each frame (right column).

The external camera parameters give the orientation and position of the optical center with respect to the world coordinate system. These parameters define a transformation between the world coordinate system and the camera coordinate system.

As the pose is computed for each frame, the external parameters must be updated according to that movement in order to get the new position of the optical center. Finally, the rotation angles (pitch and yaw) are calculated by decomposing the updated external parameters matrix in its pitch, roll and jaw components. In figure (6.15), the comparison between the real camera movement and the estimated pose is shown. The graphics show the real and estimated angles (pitch and yaw respectively) for every frame as well as the absolute error. For this experiment, the mean errors of the pitch and jaw angles were 1.45 and 1.88 degrees respectively.

In the tested sequences, the camera was rotated by 5 degrees between every frame. This small rotation step ensures that also the normal ICP algorithm converges to the real pose. However, this will restrict the movements of the robot arm in a normal application. With the structural and correlation ICP variants, the robot arm has a larger range of movements (see results of figure 6.7). The errors of the pitch and jaw angles computed with the silhouette-based algorithm are larger in a range from 0.6 to 1.3 degrees than the errors reported by the algorithms of the VISATEC system. The experimental results obtained by the algorithms of the VISATEC project are the results of the combination of several approaches in a more complex system with cognitive abilities. Our results show that the quality of the pose computation is acceptable considering the conditions of the experiment, the size of the tested objects and the fact that only contour feature information is considered.

6.5 Pre-alignment Experiments

The experiment presented in this section shows the quality of the alignment computation when the correlation ICP algorithm is used to compute a rough pose to get tracking assumption conditions. In this case, the 2D feature pre-alignment approach proposed in section 5.4.2 has been compared with the standard pre-alignment algorithm (see [20] and [78]) based on the PCA (Principal Component Analysis). These approaches have been adapted for the monocular pose estimation problem in order make suitable comparisons. The set up for this experiment is shown in figure 6.16 for the motor part and power socket models. The model was initially translated and rotated from 0 to 50 degrees around the γ axis to generate the ground truth pose. As can be seen in the left column of the figure, this rotation is too large that even the correlation ICP algorithm does not converge to the correct pose.

As it was discussed in section 5.4.2, only one iteration is needed to compute the rough pose with the 2D feature alignment. Thus, the computed pose after the first iteration of the correlation ICP algorithm is compared with the pre-alignment based on the principal component analysis. The middle column of figure 6.16 shows the result of the pre-alignment with the PCA approach, while the right column shows the result with the correlation ICP algorithm. In contrast to the alignment in 3D space, only two principal axes can be extracted and aligned in the image plane. This loss of information has the effect that, although the major and minor axes are

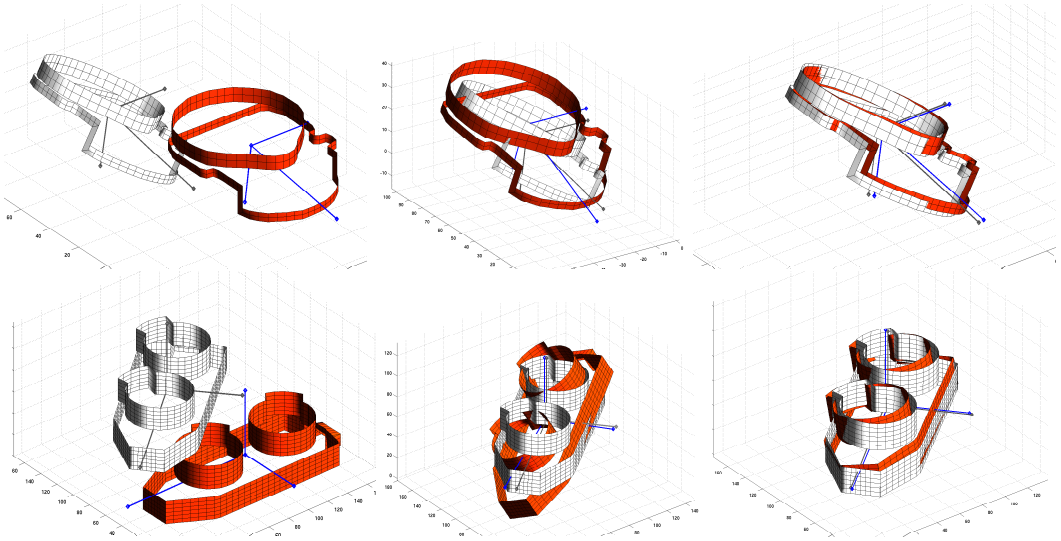

Fig. 6.16: Pre-alignment comparison of PCA and 2D feature alignment. Initial position (left column), PCA pre-alignment result (middle column) and computed pose after the first iteration of the correlation ICP algorithm (right column).

Rotation (degree)	Absolute error (mm)		Major axis difference (degree)	
	PCA	CORR + 2D Align.	PCA	CORR + 2D Align.
0	14.9172	3.1682	10.2715	1.0845
10	15.8307	3.4555	10.9774	1.1382
20	17.0352	3.4002	11.6436	1.5795
30	18.6300	2.7695	11.8687	1.1854
40	20.5412	6.6615	11.6005	3.9551
50	22.6003	5.2424	11.2610	7.0052

Tab. 6.1: Error comparison for the pre-alignment step (motor part).

roughly aligned, the error in 3D is significantly larger. On the other hand, the combination of the correlation ICP algorithm and the 2D feature alignment results in a better computation of the pre-alignment. Tables 6.1 and 6.2 show the absolute error and the difference angle of the major orientation axes in 3D for the motor part and power pocket models respectively.

Rotation (degree)	Absolute error (mm)		Major axis difference (degree)	
	PCA	CORR + 2D Align.	PCA	CORR + 2D Align.
0	12.4174	6.9135	4.0596	1.5261
10	13.8881	7.2764	2.7424	0.2738
20	14.5451	6.3372	7.0082	1.1097
30	17.0028	6.4017	8.0718	0.8097
40	21.8211	11.5085	3.8095	0.8627
50	26.6258	12.3943	4.2856	2.0066

Tab. 6.2: Error comparison for the pre-alignment step (power socket).

6.6 Experiments on Real Images

In this section, examples of the computed pose with the proposed ICP variants are presented for real images. In a first instance, experiments on image sequences under the tracking assumption are presented. The correlation ICP algorithm is used to handle occlusions in the image. With a simple interpolation technique, the computation of the pose parameters can be improved in cases where more model point than image points are available. Finally, the result of the pose computed with the structural and correlation ICP algorithms is presented for sequences of different model objects.

Different modules were implemented in the platform PAClib (Perception and Action Libraries, [82]) under C++ in a Linux based system with a 3 Ghz. Pentium 4 processor. Several object oriented libraries were developed to handle and visualize 3D contours and surfaces in the image, for the image processing and feature extraction. Similarly, libraries were written to find correspondences with the different search criteria presented in this work and to define the pose minimization constraints. For each image sequence, the camera was calibrated with the pattern shown in figure 6.17. Six reference points must be manually selected by the user in order to recover all the corner points of the pattern and to construct the 2D-3D point correspondences needed for the calibration. The point $P2$ of the calibration pattern defines the origin of the world coordinate system. Once that the camera is calibrated, the initial position of the contour or surface model is defined as shown in the upper pictures of the figure. For image sequences where the tracking assumption is considered, the computed pose for one image is used as initial position for the next image. As shown in the examples of figure 6.18, the model displacement between each frame is small. Then, the classic ICP variant is sufficient in this cases to compute the pose for the complete sequence.

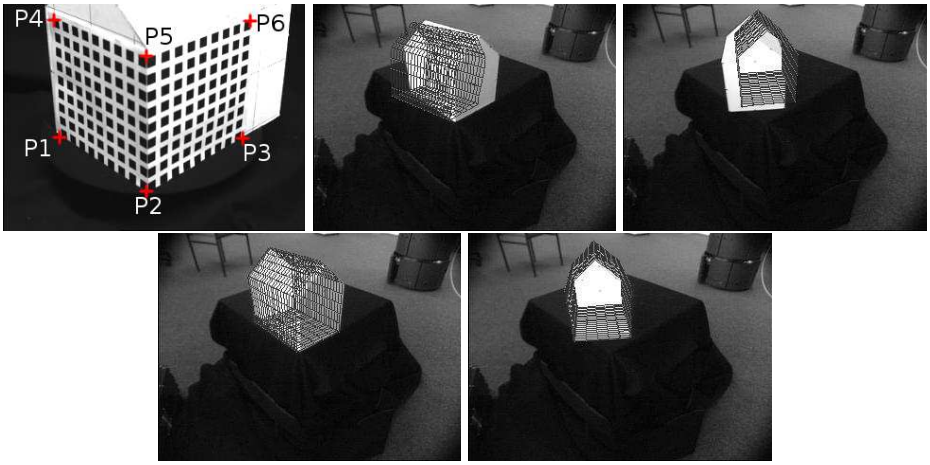

Fig. 6.17: Pattern used for the camera calibration and examples of the initial conditions for the sequences (upper pictures). Computed pose for these examples (lower pictures).

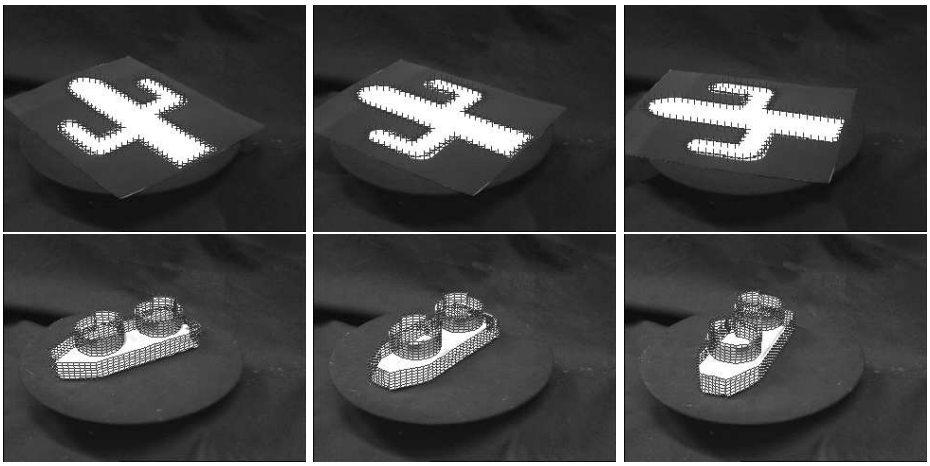

Fig. 6.18: Classical sequences of pose estimation with free-form contours and surfaces.

6.6.1 Occlusion under Tracking Assumption

In this section, several experiments to test the pose estimation of free-form contours with general occlusions in the image are presented. An example of this pose scenario is shown in the upper pictures of figure 6.20. The pictures show the initial position of the projected mouse model with respect to the real 3D mouse contour. It is evident that the image contour can not be extracted as a closed contour for these examples. The local structure of the extracted contour segments corresponds partially to the real contour and to the additional occluding objects. The main difficulty for this scenario is to decide which segments should be considered as part of the

model and which ones eliminated as outliers. An example of this scenario can be seen in figure 6.19 for a syntectic image with an occluding object. If the correspondences are found with the normal ICP algorithm, the only possibility to eliminate possible outliers is by considering the distance from model to image points. Point pairs whose Euclidean distance is larger than a certain threshold value can be eventually eliminated. Nevertheless, the correct choice of the threshold may be in practice difficult for these scenarios. A proper threshold can be selected in cases where the model is nearly aligned to the image contour and the occlusion appears for the next sequences. For this experiment, the occlusion is present since the beginning of the sequence and the same initial position is considered for all the images.

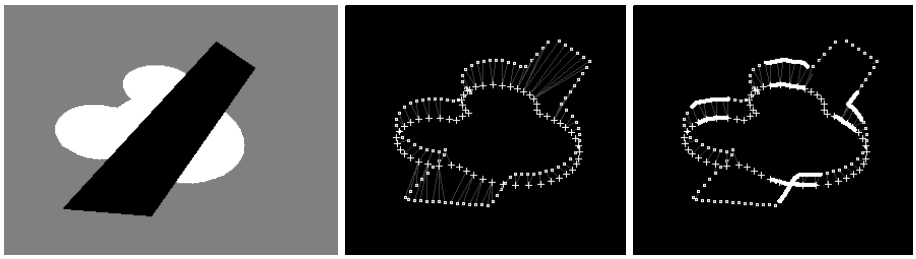

Fig. 6.19: Syntectic generated image of the mouse model with an occluding object (left). Correspondences with the minimal Euclidean distance criterion (middle). Possible correspondences with the Euclidean distance plus correlation criterion (right).

Due to the large number of possible contour segments in the image, the correspondences are found in directions from the projected model points to the image points. The Euclidean distance combined with the feature correlation defined in section 5.2.3 was used as correspondence search criterion. Since it is not possible to get the complete image contour, smaller segments are considered to define the profile vectors. An example can be seen in the right image of figure 6.19. As discussed in section 4.2, the description level of contours decreases if smaller segments are considered. In consequence, the correspondence search is more dependent on the tracking assumption.

The result of this experiment can be seen in figure 6.20. The initial position for some images is shown in the upper row. Some examples of the detected image segments can be seen in the pictures of the middle row. Only the segments of similar orientation profiles are considered to form correspondence pairs, while the non-correlated contour segments are eliminated as outliers. For a given model point, the closest image point with similar local structure (in terms of the local orientation profiles) is selected as correspondence pair. Finally, the result of the pose estimation is presented in the bottom row of figure 6.20. Despite of the amount of occlusion and the complexity of the occluded objects (e.g. the hand in the right column), the algorithm is able to find the correct pose.

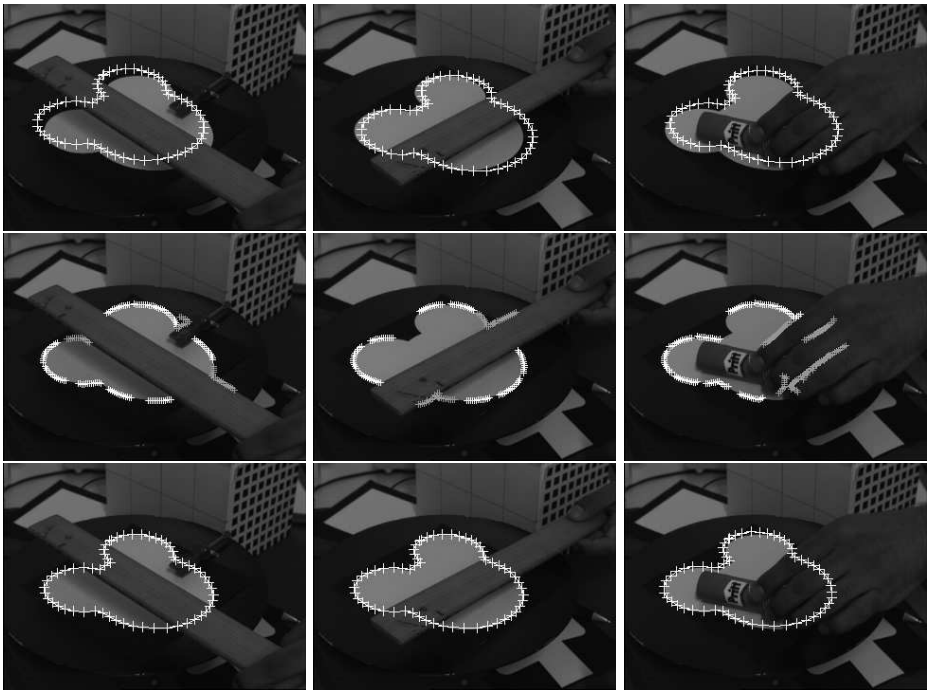

Fig. 6.20: The top images show the initial pose, the middle row shows the detected segments and the bottom row shows the pose results.

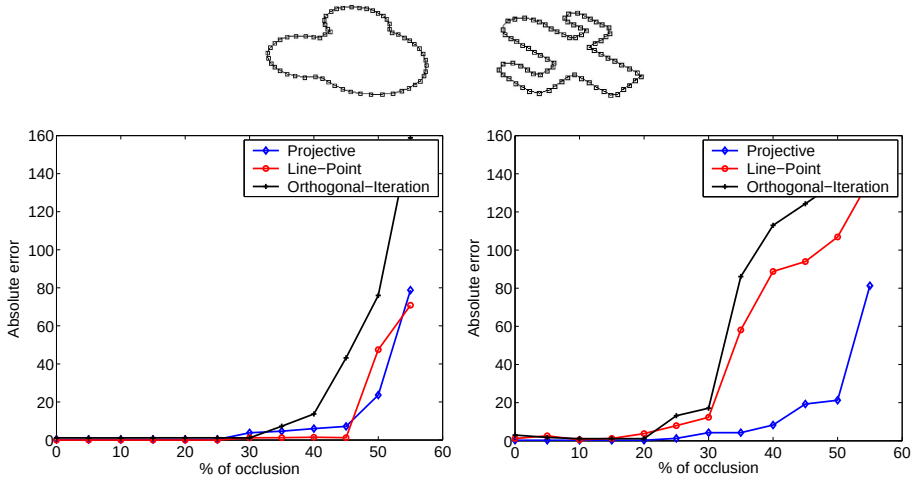

Fig. 6.21: Absolute error in millimeters for the mouse model (left) and the cactus model (right) while increasing the occlusion rate.

In a further experiment, occlusions were simulated in a syntectic image as shown in figure 6.19. Once that the correspondences were found, the pose was computed with the minimization constraints used in the last experiments: 2D-3D [93], projective [4] and orthogonal iteration [71]. As can be seen in figure 6.21, the 2D-3D and

orthogonal iteration algorithms have an acceptable error at about 30% of occlusion for the mouse model and about 20% in case of the cactus model. In cases where the contour model is rich in structure, it is possible that several segments of the model have the same orientation profiles than the occluding object. Therefore, the correspondence search algorithms are more sensitive to errors and they are less robust against occlusions for the cactus model. Notice that the mouse contains mostly curved segments. If the occluded object is defined by straight segments, the correspondences can be found more properly. Since the mouse model is poor in structure, the error is similar for all pose minimization constraints. In contrast to that, the algorithm is more robust with the projective constraint for the cactus model since this constraint is better conditioned in the image plane.

6.6.2 Pose with Interpolated Model Points

An important factor that affects the quality of the pose parameters in real pose scenarios is the number of available image and model points. For a pose scenario similar to that of the figure 6.20, the size of the extracted contours segments can be larger than the projected model points. In cases where more image points than model points are available (most real applications), several image points correspond to a single model point as can be seen in the upper left picture of figure 6.22. The middle and right pictures show an example of the computed pose with these correspondence sets for the next iterations. Image and model points are matched with respect to the image plane during the iterations. Despite of that, the pose converges to a local minimum region where the computed pose is slightly shifted from the ground truth.

If the computed pose is decomposed in its corresponding rotation and translation components, the error becomes more evident that the perceptible error in the image plane. This can be seen in the comparison presented in table 6.3. To overcome this effect, it is desirable to have the same number of image and model points to find the correspondences. Once that an image segment of a given length is extracted, the respective closest model points are interpolated in the image plane to obtain a segment of the same length. For every point of this image segment, it is possible to construct only one correspondence pair with the interpolated model points. Table 6.3 shows the computed pose parameters for the example of figure 6.22. A notorious difference in the pose parameters can be seen for the correspondences without interpolation, while this error is reduced if interpolated model points are used.

6.6.3 Pose with the Structural and Correlation ICP Algorithms

For the experiments presented in this section, the new variants of the ICP algorithm are used in combination with the projective minimization constraint. Examples of

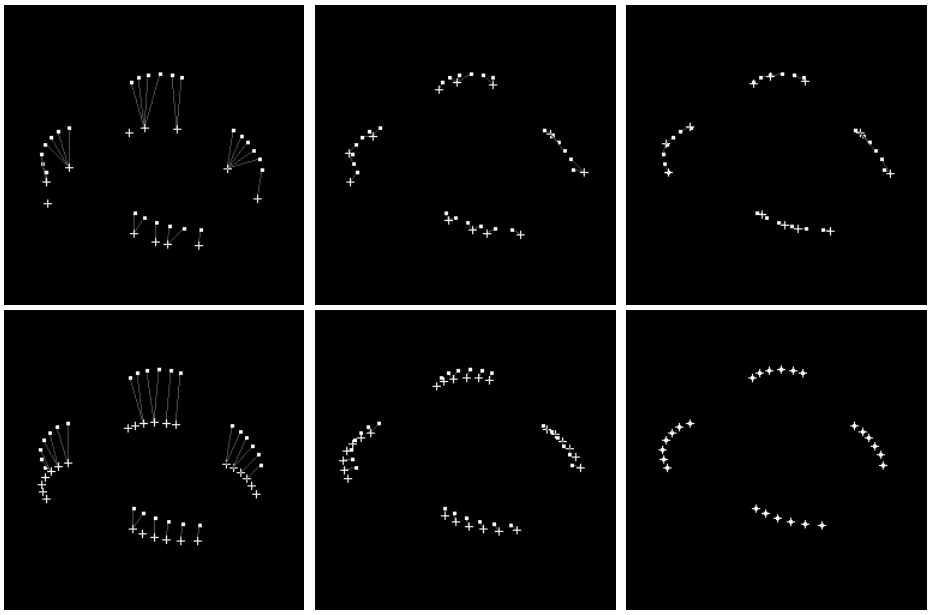

Fig. 6.22: Upper row: correspondences of contour segments without interpolation. Lower row: examples of correspondences with interpolated model points.

Pose parameters	$\alpha[^\circ]$	$\beta[^\circ]$	$\gamma[^\circ]$	$x[mm]$	$y[mm]$	$z[mm]$
Ground truth	10	-10	0	0	0	5
Without interpolation	10.7500	-6.5833	1.4971	-5.1075	-11.7413	14.0040
With interpolation	9.9976	-10.0092	-0.0151	0.1728	-0.1844	5.0172

Tab. 6.3: Comparison of the pose parameters of interpolated and non-interpolated model points.

the different sequences with the contour and surface models are shown. In contrast to the sequences where the tracking assumption is considered, the same initial position of the model is used for all the images. In a first instance, several examples of the pose estimation of free-form contours with the structural ICP algorithm are presented. For this examples, the different 3D contour models (puzzle, cactus and mouse) were used. Images of 230×240 pixels were acquired for these sequences. On the other hand, the contour models consist of 110, 95 and 90 points for the mouse, cactus and puzzle model respectively.

Figure 6.23 shows the initial position of the model and the computed pose for the different sequences. The average computing time per frame was 225 milliseconds for the image processing module. Due to the relatively large displacement of the object, more iterations are needed by the algorithm to converge and therefore the computational time increases. For these image sequences, the average computation

time (image processing plus pose estimation) was 2.65 seconds.

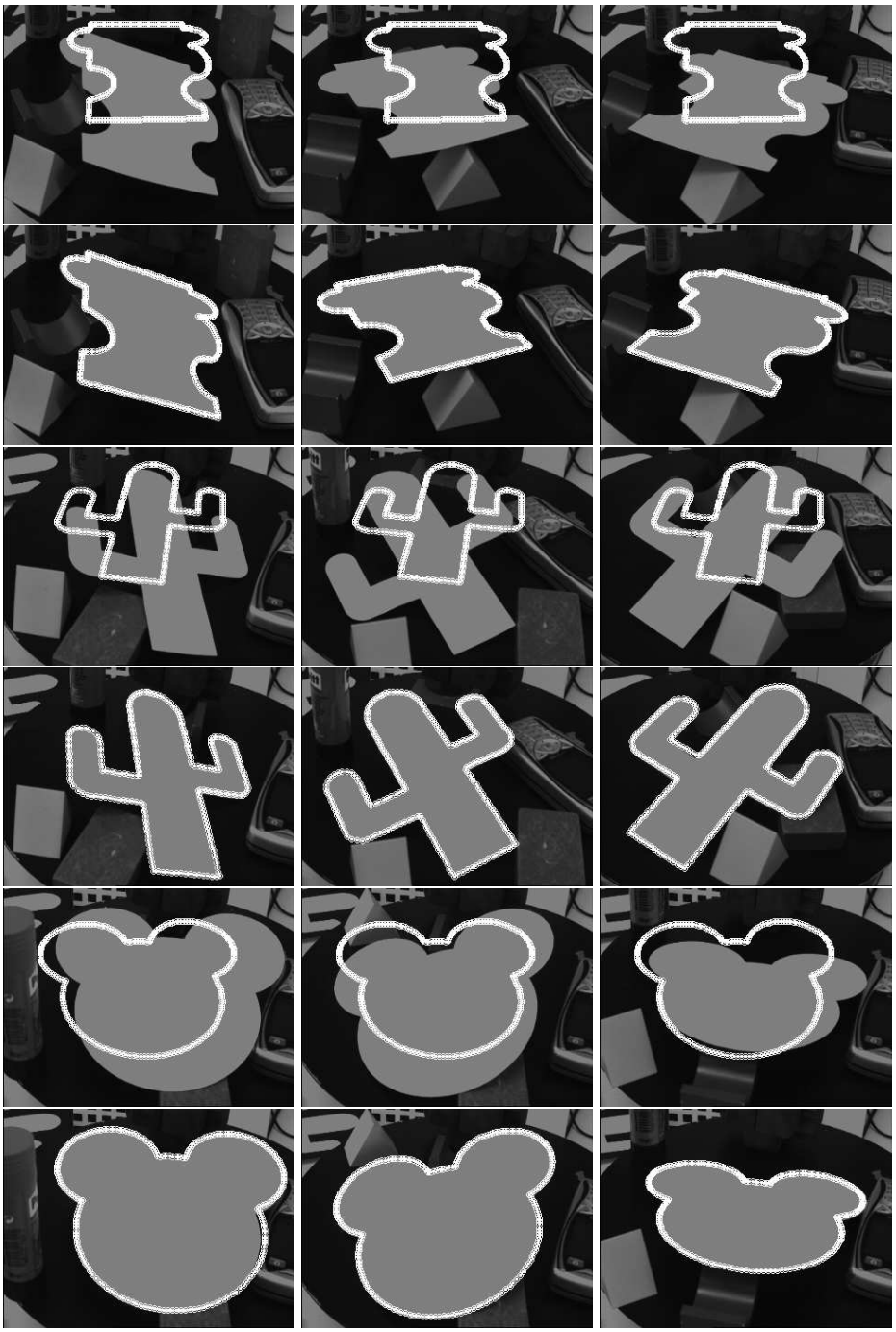

Fig. 6.23: Upper pictures: initial position of the contour model for all sequences. Lower pictures: results of the pose estimation.

Examples of the experiments where the correlation criterion is combined with the 2D feature pre-alignment are shown in figure 6.24 for different surface models.

Similar to the contour experiments, the initial position of the surface models and the computed poses can be seen in the figure. In this case, images of 620×540 pixels were acquired for the sequences. In contrast to the 3D contour examples, larger images were used for these sequences. Examples with the triangle, power socket and motor part models are presented. The triangle model is defined by one surface with 612 points. On the other hand, the motor part and power socket models are defined by two and three surfaces with 369 and 1043 points respectively.

Module	Range=30	Range=20	Range=10
Image Processing	92		
Sillhouethe Extraction	44		
Global Feature	76	23	11
Local Feature	58	41	20
Pose Calculation (10 it)	4739	4430	4149

Tab. 6.4: Computation times for the motor part model.

The computation time was measured when the profile orientation vectors are defined with ranges of 10, 20 and 30 pixels (see section 5.2.1). The average computing time per frame for the image processing module (contour extraction, global and local feature extraction) was 364 milliseconds and the complete pose computation process (image processing plus pose estimation) was 4.73 seconds for 10 iterations. The computation times of the different modules for different ranges can be seen in table 6.4. Let us remember that the image processing and feature extraction of the image contours are done only once at the beginning of the algorithm. The global features are computed only for the first iteration while the local and semi-local features are computed for all iterations. Real time computation times for each frame are not reached with the proposed new variants of the ICP algorithm. Despite of that, the reported times can be considered a good tradeoff considering the robustness of the algorithms.

In real images, the presence of noise or slight illumination changes may produce a higher uncertainty in the feature computation and therefore in the correspondences search. In order to eliminate false correspondence points as outliers, the Euclidean distance criterion defined in [74] is used. Correspondence pairs are rejected if their point-to-point distance is larger than 2.5 times the standard deviation of the complete correspondence set. A deeper analysis of use of different outlier elimination criteria is presented in the next chapter, which is the basis for the pose estimation with partial occlusions.

6.7 Summary

Experiments to test the properties and robustness of the new variants of the ICP algorithm were presented in this chapter for the monocular pose estimation. The experiments were made to test the convergence behavior of the algorithms and their robustness against the tracking assumption. Additionally, the algorithms were tested in the presence or noise, missing contours and occlusions. It has been shown that the presented algorithms perform efficiently for artificial and real pose estimation scenarios. The new variants of the ICP algorithm improve the pose computation process for the different minimization constraints considered in the experiments. Furthermore, a better performance is obtained in combination with the projective pose estimation constraints. Let us remember that the feature extraction process, correspondence search and minimization constraints are essentially defined in the image plane. This proves that translating all steps of the pose estimation problem to the image plane allows to obtain better-conditioned correspondences and therefore a more robust pose computation.

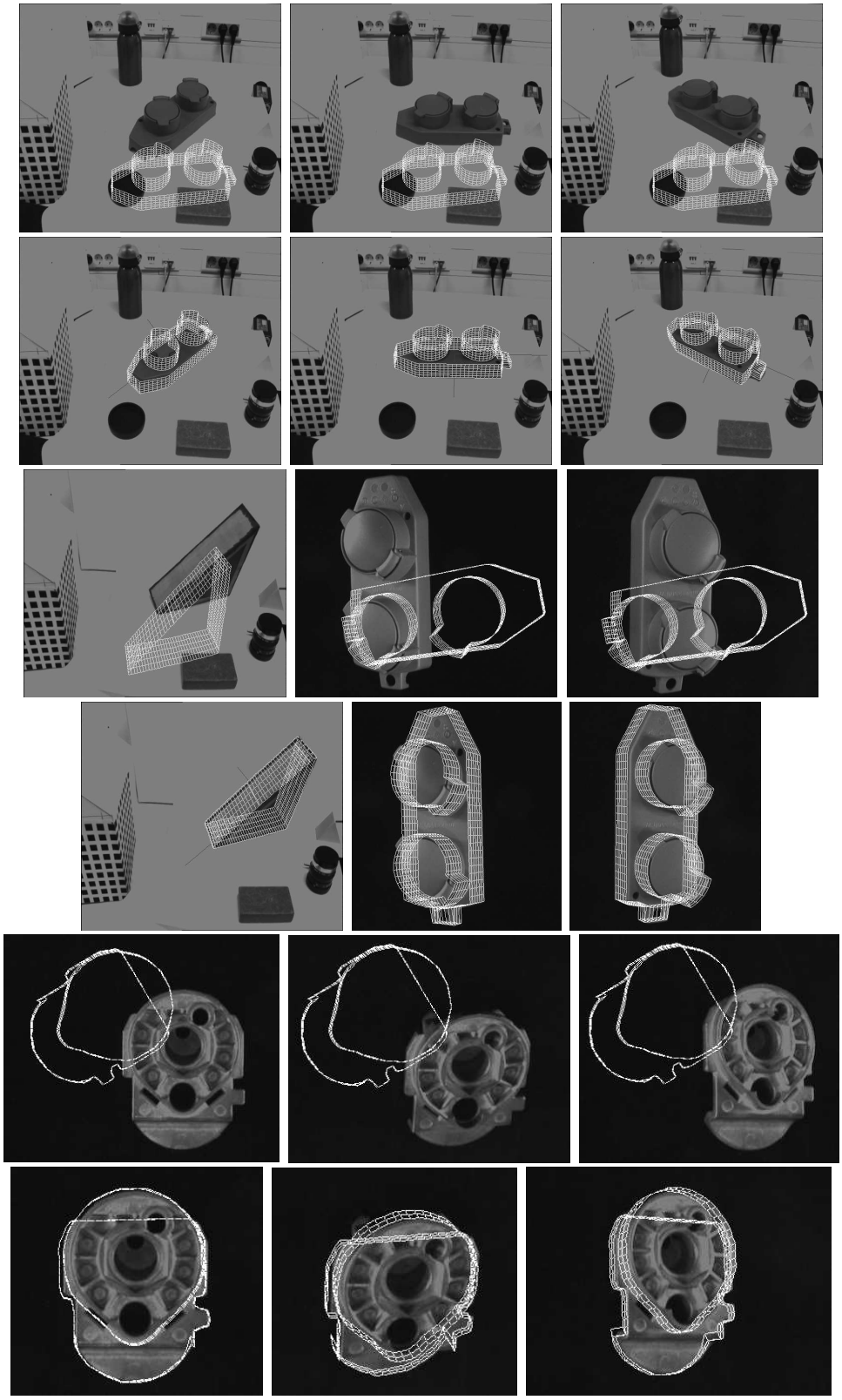

Fig. 6.24: Upper pictures: initial position of the surface model for all sequences. Lower pictures: results of the pose estimation.

Chapter 7

COMBINATION OF THE NEW ICP VARIANTS

Two new variants of the ICP algorithm have been presented in the last chapters. In order to find correspondences, local, semi-local and global information have been combined with the minimal Euclidean distance and maximal correlation criteria. All these possible variants offer advantages for certain pose estimation scenarios, e.g. large displacements, general and partial occlusions. From the experiments presented in the last chapter, it can be concluded that it is not possible to have only one general correspondence search criterion for all possible pose estimation scenarios. This statement is valid especially in the case that only contour information is used.

One of the major challenges of tracking and pose estimation applications is to develop autonomous and robust systems, which have self-adaptation capabilities at the beginning and during the tracking sequences. Examples of these systems which use a combination of strategies for the pose estimation problem can be seen in [59, 80, 83, 84]. Ideally, these systems may be able to perform an automatic initialization and re-initialization of their initial parameters. That means, the system may be able to choose a proper initial position of the object model at the beginning of the tracking sequences. Eventually, this initial position can be updated if the tracking is lost during the sequences. On the other hand, the tracking system may not need the same correspondence search criterion during all the sequence. Therefore, it is desirable to establish several criteria to define which correspondence search approach should be used. The same idea can be applied to choose the most appropriate search criterion also during the iterative process for each single image.

The above cited methods are the combination of several standard and novel techniques based on more complete model objects than free-from contours and surfaces. Despite of that, it is possible to use these ideas to combine the presented variants of the ICP algorithm to solve the correspondence problem when only contour information is available. In this chapter, the combination of the different ICP variants is presented for the pose estimation of surfaces with partial occlusions when the tracking assumption is not considered. In a first instance, the combination of the correlation ICP algorithm with an outlier elimination criterion is introduced. Then, a pose estimation strategy is presented that selects the most adequate combination

of correspondence search and outlier elimination criteria. Finally, several experiments with synthetic and real images are shown to test the presented strategies.

7.1 Combination of Correspondence and Outlier Elimination Criteria

The general concept of the combination of correspondence search and outlier elimination strategies is shown in figure 7.1. By following the notation used in the last chapters, the sets of projected model $\mathcal{M}^{2D}$ and image contour $\mathcal{S}^{2D}$ points define the input of the algorithm. Additionally, the sets $\mathbf{F}_{mod}$ and $\mathbf{F}_{img}$ correspond to the model and image features respectively. Additionally to the Euclidean distance C_1 and feature correlation criteria C_2, the correspondence search criterion C_3 based on a positional feature is defined. Since these variants use local, semi-local or global features, they deliver different correspondence sets and therefore different poses. It has been shown in the last chapters that better correspondences are obtained if more information from model and image contours is used. At the beginning or during the image sequence, the algorithm selects the most appropriate search criterion depending on the initial conditions for a given image.

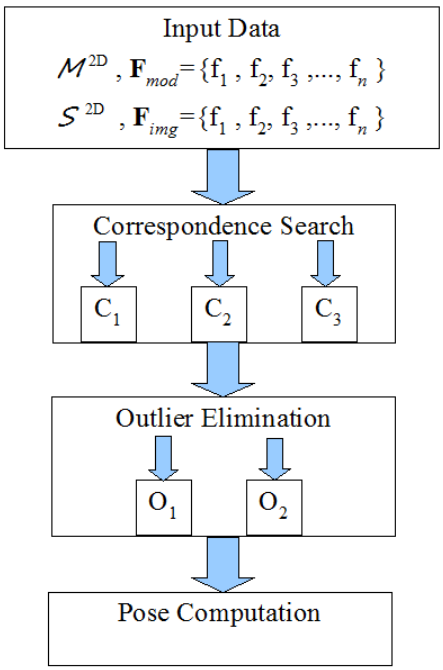

Fig. 7.1: General concept for the combination of the ICP variants.

Let us consider the case when the tracking assumption is not considered at the beginning of the algorithm. In this case, the algorithm must decide if a pre-alignment

step is needed to get tracking assumption conditions. If only contour information is used, the only possibility to make this decision is by comparing the position and orientation of the projected model with respect to the contour extracted from the image. Once that the correspondence set is obtained, the next step is to decide if these correspondences are correct or not. In practice, there is no direct method to evaluate if a correspondence set is completely correct or not. Despite of that, the Euclidean distance between model and image points can be used to detect possible outliers. This outlier elimination criterion is denoted as O_1. Additionally, a positional elimination criterion O_2 is also considered. It relates two contour points according to their angular position with respect to the main orientation of the complete contour. Similar to the correspondence search criteria, the use of O_1 or O_2 depends on the initial conditions of the sequence.

In order to define the correspondence search C_3 and the outlier elimination criterion O_2, a positional feature of contour points is used as can be seen in figure 7.2. Since the principal axes and centers of mass have been computed from the elliptical Fourier descriptors (see section 3.3.4), it is possible to describe each contour point by its angular position. That means, the angle between the principal axis and the vector formed with the center of mass and the point itself. In fact, the angular position can be considered as a global feature since it is obtained from the global position and orientation of the contour.

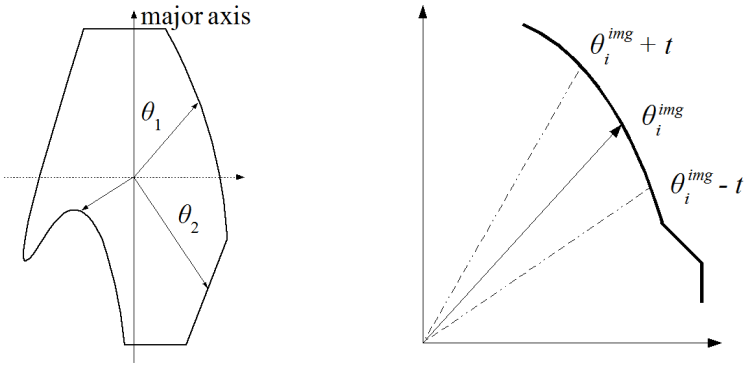

Fig. 7.2: Angular position of contour points with respect to the major axis (left). Interval within which the corresponding image point should be defined (right).

The correspondence search criterion C_3 is introduced as follows. For an image point $\mathbf{x}_i^{img} \in \mathcal{S}^{2D}$ with angular position θ_i^{img}, its corresponding model point $\mathbf{x}_j^{mod} \in \mathcal{M}^{2D}$ is defined by

$$\mathrm{ang}(\mathbf{x}_i^{img}, \mathcal{M}^{2D}) = \min_{j=1,\cdots,n} \{\| \theta_i^{img} - \theta_j^{mod} \|\}, \tag{7.1}$$

where θ_j^{mod} denotes the angular position of the model point $\mathbf{x}_j^{mod}$. According to this criterion, two points correspond to each other if the difference between their

angular positions is minimal. Eventually, this criterion can be also combined with the structural constraints defined for the structural ICP algorithm.

Once that a correspondence set $\{\mathbf{x}_i^{img}, \mathbf{x}_i^{mod}\}_{i=1}^n$ has been found, the Euclidean distance is used to select or reject possible correspondence pairs. Then, the outlier elimination criterion O_1 is defined by

$$d(\mathbf{x}_i^{img}, \mathbf{x}_j^{mod}) > a(\sigma), \tag{7.2}$$

where σ is the standard deviation of the distance of the complete correspondence set and a is a scaling factor. Correspondence pairs are rejected if their distance is larger than the standard deviation multiplied by the scaling factor. In practice, this factor is set between 1 and 2.5, see [74].

Finally, a correspondence pair is rejected by the criterion O_2 according to their respective angular positions θ_i^{img} and θ_j^{mod} if the following condition is fulfilled

$$\parallel \theta_i^{img} - \theta_j^{mod} \parallel > t, \tag{7.3}$$

where the value t stands for a threshold value that defines the interval within which the correct correspondence pair must be defined, seen figure 7.2.

7.2 Pre-alignment With Partial Occlusions

In this section, the combination of the correspondence search and outlier elimination criteria is applied for the pose estimation problem in the presence of partial occlusions. Several possibilities to combine the criteria defined in the last section are defined. Then, experiments are presented to test the behavior of the algorithms during the iterations and with respect to the final pose.

One drawback of the presented variants of the ICP algorithm is the high sensitivity of the correspondence search process to partial occlusions. An example of the power socket with an occluded object is shown in figure 7.3. It is clear that this kind of occlusion distorts the local and global structure of the detected image contour. The additional concave, convex and straight segments cause bad-conditioned correspondences if the structural ICP algorithm is applied. This correspondences will not be corrected in the next iterations since there are no model segments that correspond to the occluded object. An example of correspondences computed with the correlation ICP algorithm and the 2D feature alignment is shown in the lower row of the figure. It can be seen that many bad-conditioned correspondences are found since the profile vectors containing local orientation are distorted. The use of a simple outlier elimination criterion based on the Euclidean distance is not enough for the structural and correlation ICP algorithms to handle these occlusions.

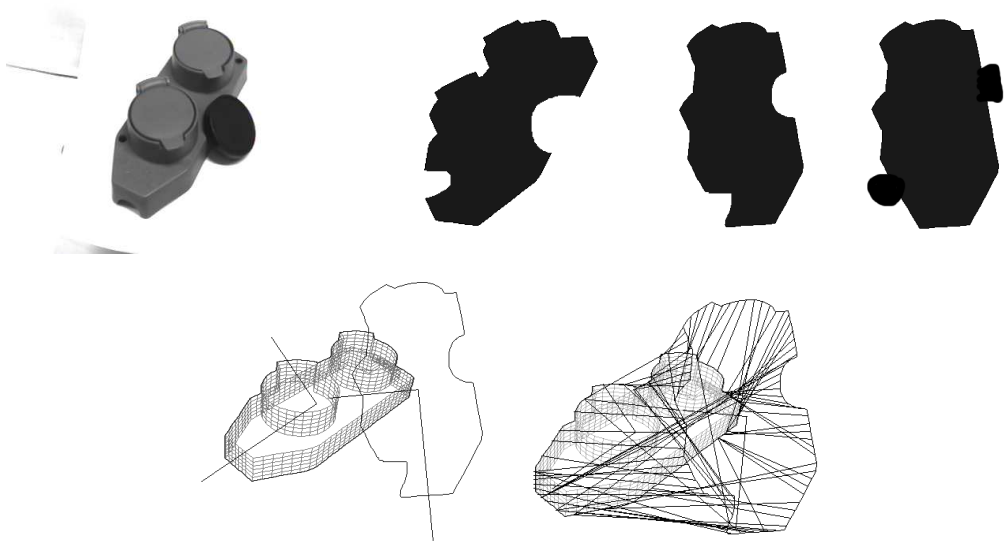

Fig. 7.3: Upper row: examples of partial occlusions of the power socket model and examples of artificially generated image contours with occlusions. Lower row: model and detected image contours (right) and correspondences with the correlation criterion with the 2D feature correlation (right).

7.2.1 Possible Combinations

Several possibilities to combine the presented correlation and outlier elimination criteria are considered to handle partial occlusions. They are defined as follows

- M0: $\mathrm{dist}(\mathbf{x}_i^{img}, \mathcal{M}^{2D})$ with $(S_i^{img} = S_j^{mod})$ and O_2.
- M1: $\mathrm{corr}(\mathbf{o}_i^{img}, \mathcal{M}^{2D})$ with $(S_i^{img} = S_j^{mod})$ and O_1.
- M2: $\mathrm{corr}(\mathbf{o}_i^{img}, \mathcal{M}^{2D})$ with $(S_i^{img} = S_j^{mod})$ and O_2.
- M3: $\mathrm{ang}(\mathbf{x}_i^{img}, \mathcal{M}^{2D})$.
- M4: $\mathrm{ang}(\mathbf{x}_i^{img}, \mathcal{M}^{2D})$ with $(S_i^{img} = S_j^{mod})$ and O_1.
- M5: $\mathrm{ang}(\mathbf{x}_i^{img}, \mathcal{M}^{2D})$ with $(S_i^{img} = S_j^{mod})$ and O_2.

The variant M0 stands for the structural ICP algorithm. The variants M1 and M2 use the correlation search constraint in combination with the structural constraints defined in section 4.5.1. In order to simplify the notation, the indexes S_i^{img} and S_j^{mod} are used to denote the semi-local structure (concavity, convexity, straightness and phase-transition index) of image and model respectively. Thus, the condition

$S_i^{img} = S_j^{mod}$ implies that model and image points have the same semi-local structure. In order to eliminate correspondences, the variant M1 uses the Euclidean distance criterion O_1 while M0 and M2 use the angular position criterion O_2. Only the angular position is used to find correspondences with the variant M3 without considering any outlier elimination criterion. Finally, the variants M4 and M5 use the angular position and local structure to find correspondences in combination with O_1 and O_2 respectively.

7.2.2 Experimental Results

The purpose of the next experiments is to determine which one of the combinations defined in the last section is able to deliver a better pre-aligned pose. Therefore, only the result of the computed pose after the first iteration is presented in the first experiment. For these experiments, partial occlusions were simulated in the images as can be seen in the examples of figure 7.3.

A comparison of the correspondences and computed poses in 3D space is shown in figure 7.4. The initial and the actual position (ground truth) of the power socket model are shown in the upper row. In the following rows, correspondences and computed poses in 3D are shown for the variants M1 to M5. The worst-conditioned correspondences are obtained with the variant M1 (this variant corresponds to the correlation search criterion in combination with the 2D feature alignment). Therefore, the computed pose is completely wrong as can be seen in the figure. A better pose is computed with the variants M3 and M4. The use of the angular position as correspondence search criterion (variant M3 without outlier elimination) allows to find better conditioned correspondences. Although the algorithm does not eliminate the wrong correspondences that are part of the occluded segments, the computed pose is acceptable.

In contrast to the last variants, the correspondences of the occluded segments are better eliminated with the addition of the angular position as elimination criterion by the variants M2 and M5. Although the variant M2 reduces significatively the number of correspondences, there are still enough well-conditioned correspondences to compute an acceptable pose. On the other hand, a larger number of correspondences are found by the variant M5 while the correspondences of occluded segments are better eliminated. Therefore, the best correspondences and pre-alignment poses have been obtained with the variants M2 and M5. A numerical comparison of the absolute error in 3D and the difference of the main orientation axes of the surface model is shown in table 7.1. It can be seen that all variants are able to align the main axes with an acceptable error rate. Despite of that, the absolute error differs in every case (let us remember that the alignment is done with respect to the image plane).

Several of the presented variants seem to deliver an acceptable pre-alignment pose. Despite of that, the convergence behavior may differ if the pose is computed

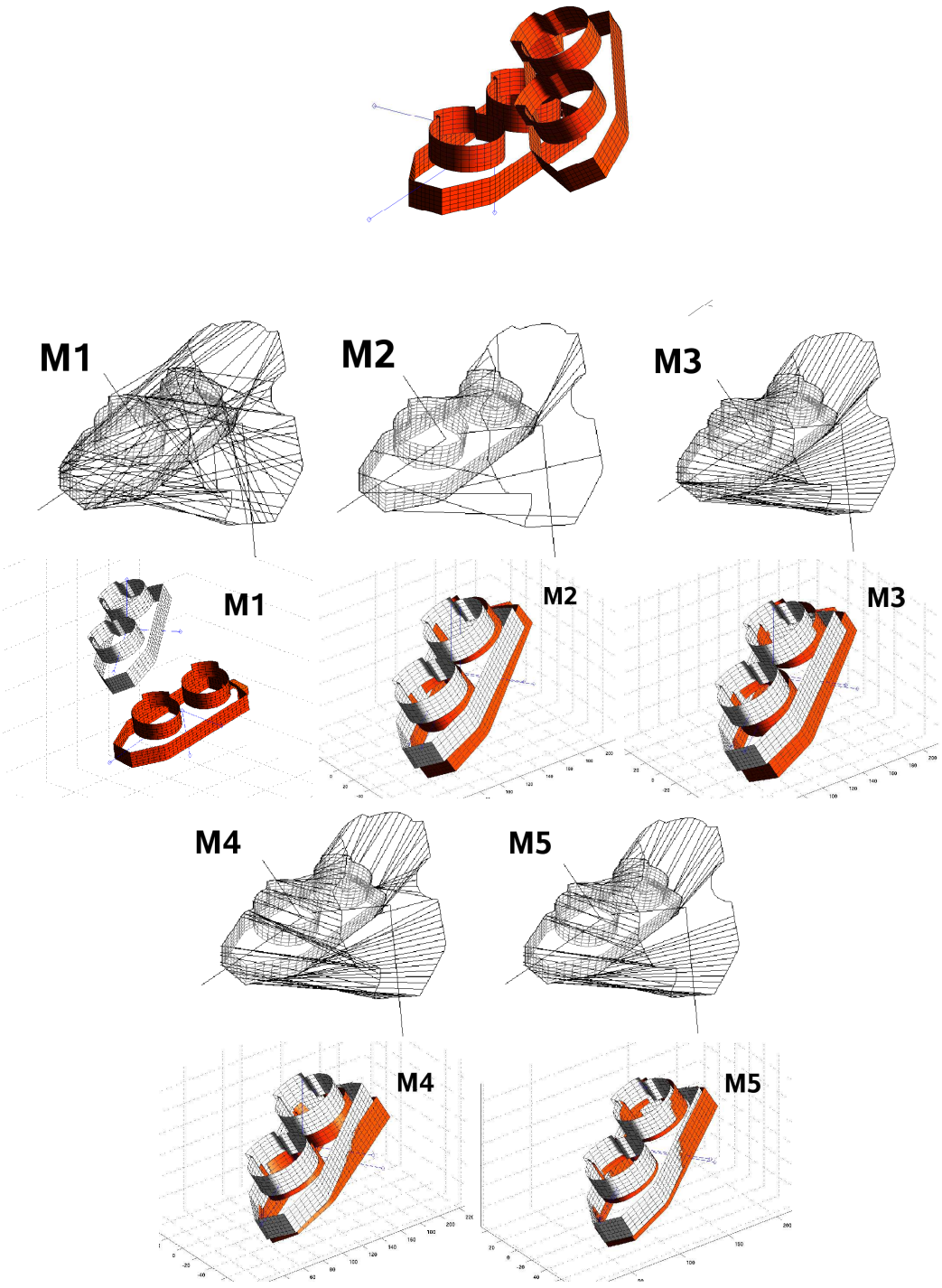

Fig. 7.4: Initial and actual position of the surface in 3D (fist row). Computed correspondences in the image plane (second row) and computed pose in 3D (third row).

Method	Absolute error (mm)	Major axis difference (degree)
M1	265.04	38.09
M2	21.75	1.1438
M3	20.4702	0.5980
M4	18.2343	4.7797
M5	7.6833	1.7230

Tab. 7.1: Error comparison of the pre-alignment with occlusions for the different pre-alignment variants.

for a larger number of iterations. A comparison of the variants M2 to M5 for 10 iterations is shown in figure 7.5. The error computed with the variants M3 and M4 remains relatively constant during the iterations. Although the computed correspondences with the angular position are close to the correct correspondences, the addition of the structural constraints is not enough to ensure a proper convergence. In the case of the variant M5, the error is even smaller than the error computed with the other variants only for the first two iterations. At the third iteration, the error increases similarly to the other variants. This particular phenomenon appears depending on the complexity of the object and its position with respect to the image contour. In this case, the angular position and Euclidean distance criteria refer only to position information. The algorithm may not ensure a stable convergence in the presence of occlusions if the local structure is not considered by the correspondence search criteria. Therefore, the angular position is a suitable option for the pre-alignment but it is not necessarily well-conditioned as a correspondence search criteria for all the algorithm. As can be seen in figure 7.5, the variant M2 (correlation search criterion plus angular position as outlier elimination) performs better and more stable during the iterations.

An important parameter that affects the behavior of pose computation is the range defined by the threshold value t, see equation (7.3). In the next experiment, the threshold value was varied from 10 to 50 degrees and the pose was computed for several iterations. The results of this experiment are shown in figure 7.6. If t is chosen too large, the algorithm does not converge to the real pose. For smaller range values, a better convergence is achieved. Smaller range values will ensure a better elimination of the correspondences formed with the occluded contour segments. On the other hand, the number of correspondences used to compute the pose are considerably reduced at smaller range values. The right graphic of figure 7.6 shows the number of correspondences for every iteration. From initially 96 pairs, the number of correspondences is considerably reduced. As the algorithm converges to the real pose, the number of correspondences tend to be similar for each case (around

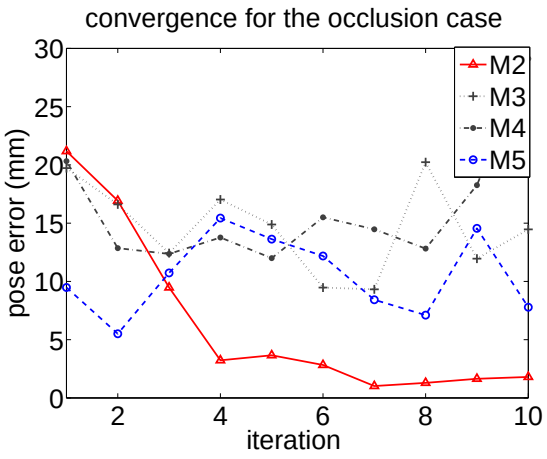

Fig. 7.5: Convergence of the different variants for the occlusion case.

20 pairs for all cases). In terms of convergence and number of correspondences, an optimal behavior of the algorithm is achieved with range values between 20 and 30 degrees.

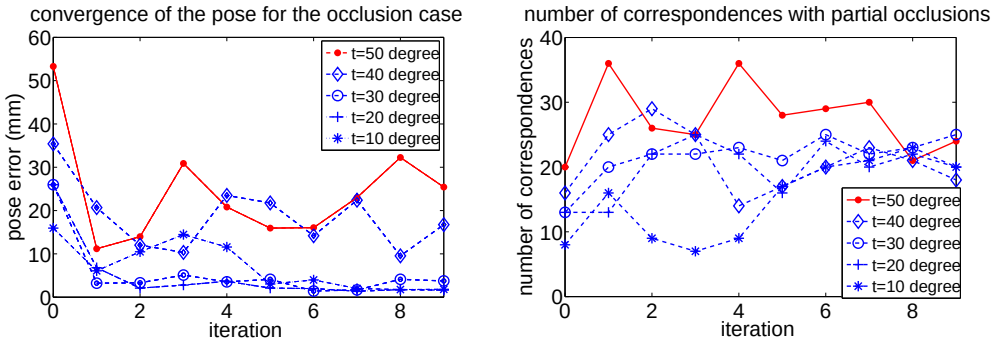

Fig. 7.6: Left graphic: convergence behavior of the algorithm for the occlusion case with different range values t. Right graphic: number of correspondences for each iteration.

7.3 Selective Pose Estimation for Real Images

Once that the different combinations of correspondence search and outlier elimination criteria have been introduced, a pose estimation strategy that performs a selective pose estimation is presented in this section. The selection is done depending on the initial conditions for each frame and the error during the iterations.

In real pose estimation and tracking applications, the choice of a specific variant

for every possible scenario may be in practice not trivial. Due to large movements of the objects between every captured image, the feature alignment approach may be required. In the case of small movements, it may be sufficient to apply the correlation ICP or even the structural ICP variant. Although the correlation search criterion combined with the angular position as outlier elimination performs more stable, it may be convenient to switch between the different combinations depending on the conditions of a particular sequence.

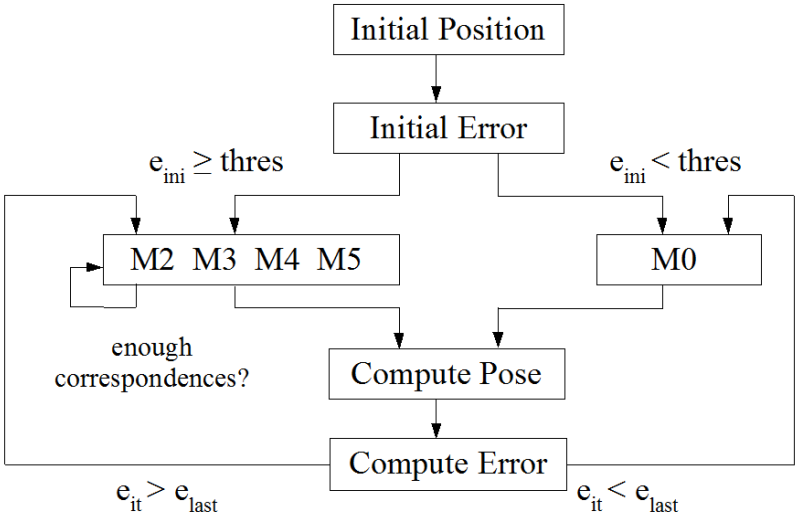

Fig. 7.7: General diagram for the selective pose estimation system.

As can be seen in figure 7.7, the initial error is used to decide which correspondence search criteria must be selected for the first iteration. This initial error is computed in terms of the absolute distance error between the detected image contour and the 2D silhouette of the model. Similarly, the difference between the extracted major distribution axes is used as an estimation of the initial error. If this error is smaller than a given threshold value, the pose can be computed with the structural ICP variant in combination with the angular position as outlier elimination (M0). Otherwise, one of the variants combined with the feature pre-alignment should be used. If a pre-alignment is needed, the most robust variant is initially chosen for the first iteration (M2).

Depending of the amount of possible occlusions, the variant M2 may not deliver enough correspondences to find the pose in extreme cases. In this case, a different variant is selected to find enough correspondences. The order in which the variants are selected depends on their robustness. As can be seen in the examples of figure 7.4, better-conditioned correspondences are found with M5, M4 and M3 respectively. In the worst of the cases, the pre-alignment would be computed with the variant M3 and the error is computed again. This is repeated at least for the first two iterations to ensure the best possible pre-alignment pose. If the error is smaller

than the initial pose error, the system switches to the structural variant M0 for the next iterations.

When occlusions are present, the risk to converge to a local minimum is larger. Therefore, the error computed in the last iteration $\mathrm{e}_{\mathrm{last}}$ is compared with the error of the actual iteration e_{it}. If this error increases, the pose is computed with a more robust variant combined with a change of the correspondence search direction as defined in section 4.5.3. In the case that the error increases with respect to the last error or it remains relatively constat, the algorithm converges to a local minimum. That means, the initial position of the model is out of the convergence rage that the structural and correlation ICP variants are able to handle.

7.3.1 Examples of Surfaces with Partial Occlusions

In this section, several examples of the pose estimation of free-form surfaces with partial occlusions are presented. The system described in the last section was tested with sequences of the motor part and power socket. For these examples, several occluding objects were drawn in the images as shown in the upper row of figure 7.8. This results in the addition of extra concave and convex segments to the extracted image contours as can be seen in the second row of the figure. Since a pre-alignment is needed for these examples, the correspondences were initially obtained with the variant M2. In some cases, some of the correspondences of the distorted contour segments are not eliminated in the first iteration of the algorithm. This can be seen in the marked regions of the images. Despite of that, the algorithm is able to eliminate these bad-conditioned correspondences during the next iterations. An example of a sequence of the power socket model is shown in figure 7.9. In this case, the occlusions in the images are generated by adding several squares randomly at two different positions of the contour. The same initial position was used for all images of the sequence. As can be seen in the pictures, the correct pose is computed for all images.

Figure 7.10 shows some examples of the sequences where the algorithm is not able to deliver a correct the pose. That means, cases where the initial position of the model is bad-conditioned or out of the convergence region for this pose scenario. Essential for the feature pre-alignment is the proper extraction of the main orientation axes in the image plane. In the first two rows, examples are shown where the orientation of the projected model drastically differs with respect to the orientation of the image contour. Since the global orientation can not be properly recovered, the feature alignment fails and bad-conditioned correspondences are obtained. In the examples of the next two rows, the main orientation and correspondences are consistently computed. Despite of that, the obtained pose at the end of the iterations is not correct. In both examples, the rotation between model and image is out the converge region of the algorithm. This results in very similar sets of local features for such poses. Therefore, the algorithm may converge to a wrong pose (local

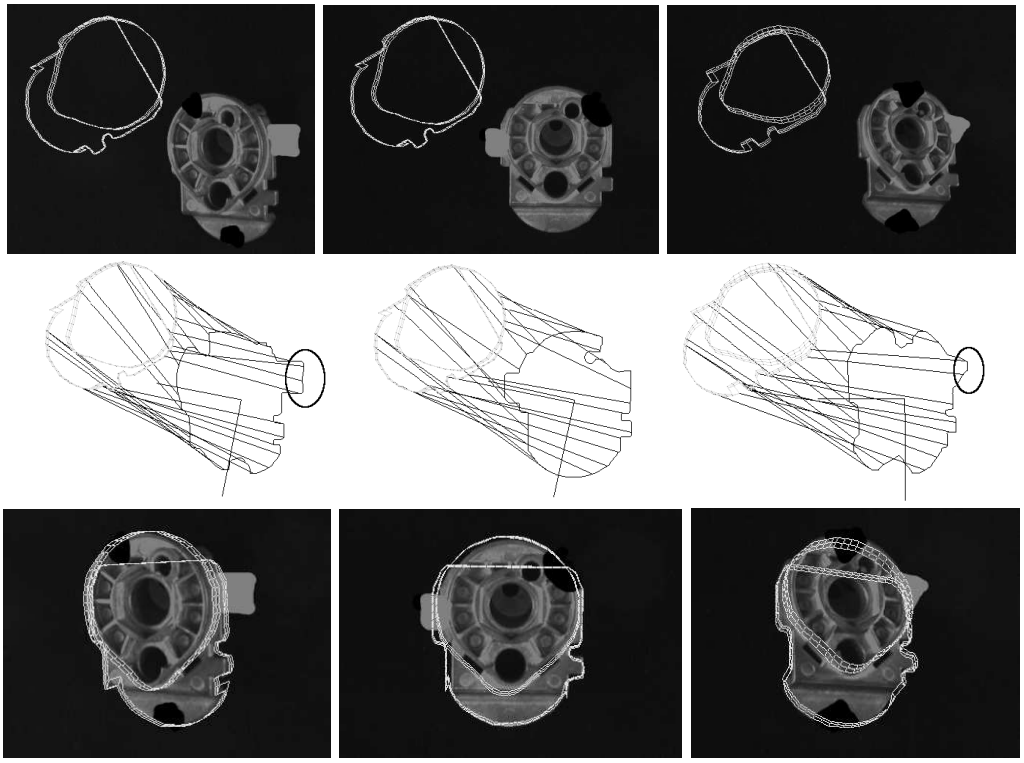

Fig. 7.8: Examples of the initial position of the model with respect to the image data with occlusions (upper row). Obtained correspondences (middle row) and computed poses (lower row).

minimum).

The last row of figure 7.10 shows an example of a large rotation. Let us remember that the minimization process based on a gradient decent method would be able to compute a correct pose up to rotations about 180 degrees, see section 2.5.4. Nevertheless, this is valid only when exact correspondences are available. In practice, the gradient decent minimization is more sensitive to bad-conditioned correspondences at larger rotations. Although the rotation is smaller than 180 degrees, the computed pre-alignment pose is still not optimal. Such larger rotations rarely appear in a normal tracking sequence. If necessary, the pre-alignment can be done in two steps if the major axes difference is larger than 90 degrees.

7.4 Summary

In this chapter, a selective pose estimation strategy was presented for the pose estimation problem of free-form surfaces with partial occlusions. The presented examples show that the correlation search criterion, combined with a distance outlier

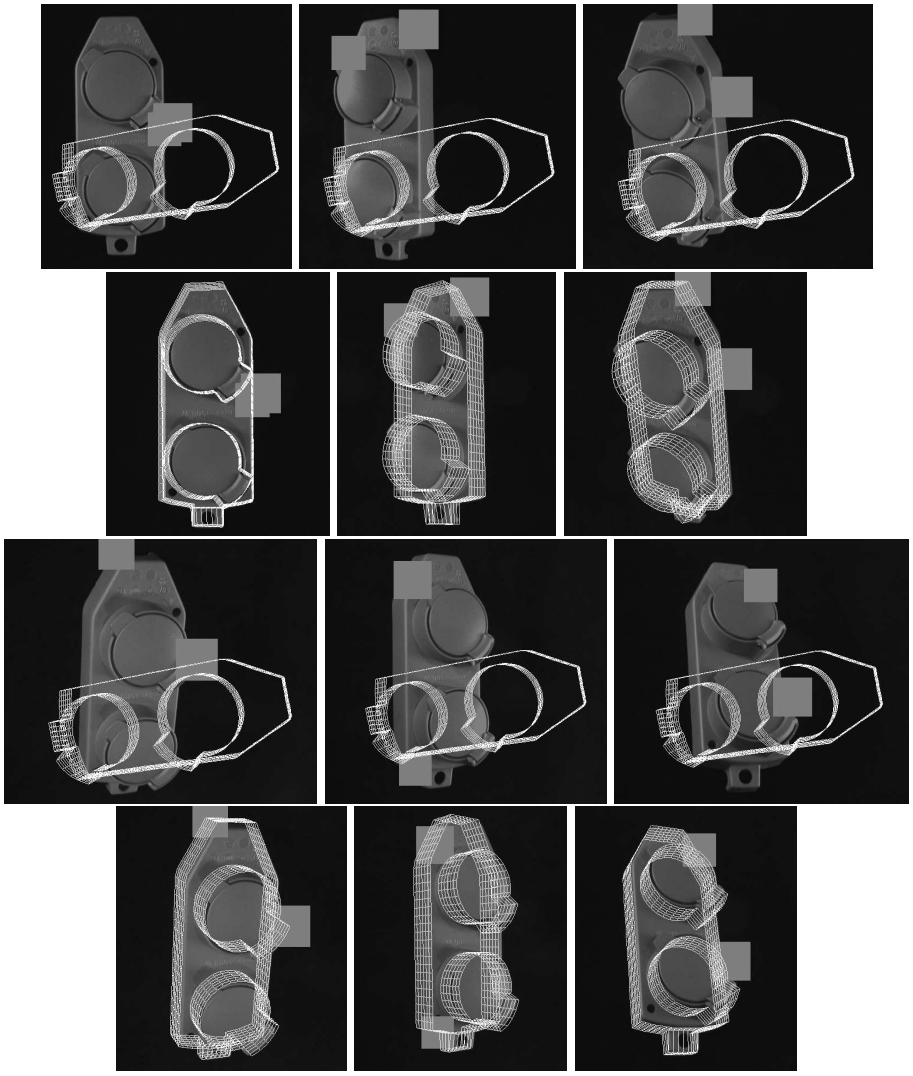

Fig. 7.9: Examples with the sequence of the power socket model with partial occlusions. Initial position of the model (first and third rows) and computed pose (second and fourth rows).

minimization criterion is not sufficient to obtain an adequate pre-alignment pose. To overcome this problem, the angular position of contour points was used as outlier elimination criterion. Since this feature is computed with respect to the main orientation axis (global orientation) of contour and projected model points, it can be considered as a global positional feature. Several combinations of correspondence search and outlier elimination criteria were analyzed. It turned out that the best pre-alignment pose is computed with the correlation criterion combined with the angular position as outlier elimination. Because of the robustness of the correlation criterion, it is sufficient to apply it only at the beginning of the algorithm in most of the cases. The structural ICP algorithm can be applied for the rest of the iterations

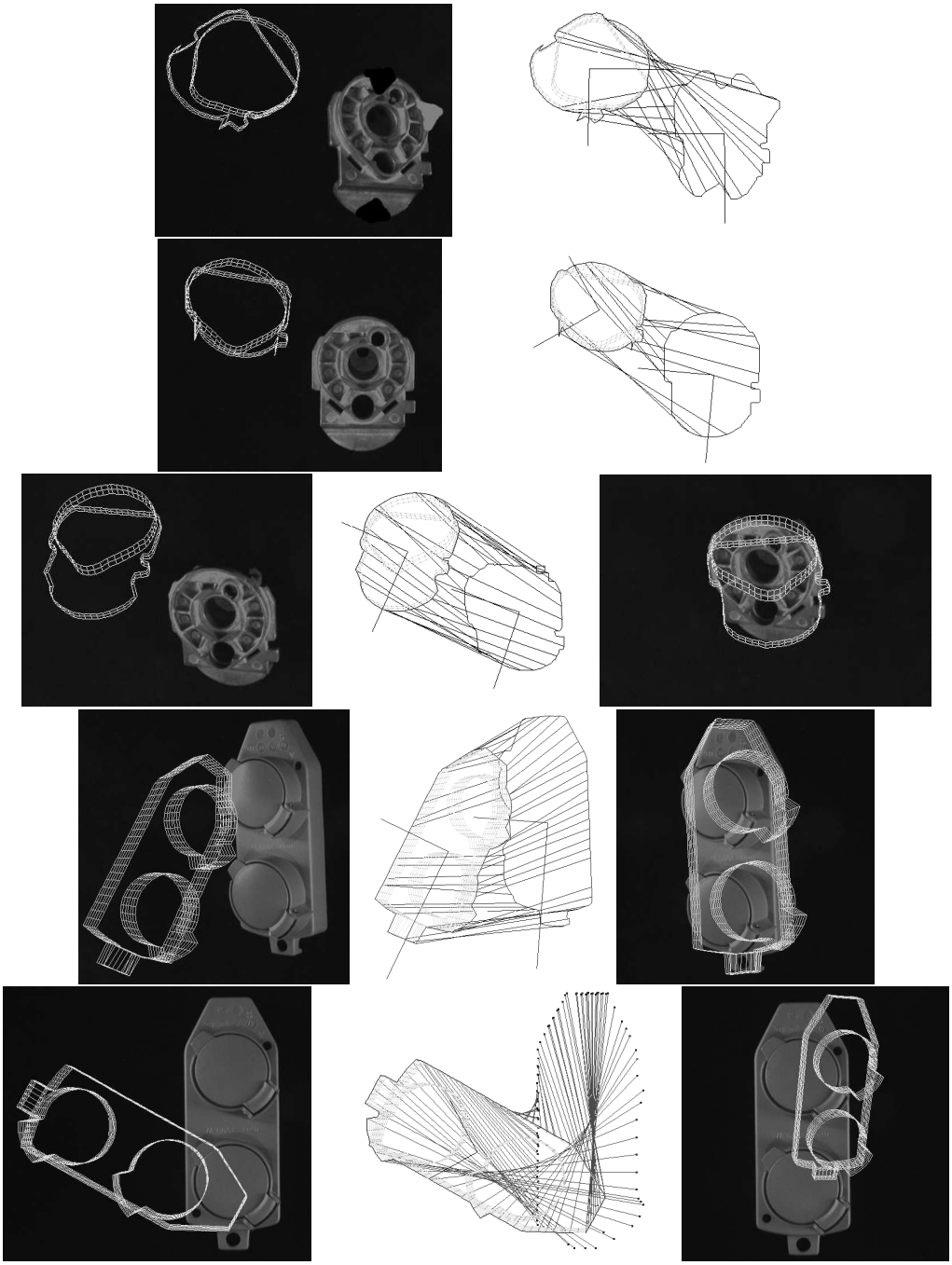

Fig. 7.10: Examples of initial positions of the surface model where the pose computation with pre-alignment fails.

to find the fine pose.

The presented pose estimation strategy performs efficiently for the tested image sequences. Depending on the amount of occlusions, the most robust combination

may eliminate too many correspondences. Thus, the pose can not be computed anymore. In such cases, the system selects another combination to get a sufficient number of correspondences and to compute an acceptable pre-alignment. If the error during the iterations increases, the system changes the correspondence criterion and the search direction to avoid the local minimum problem. The proposed algorithms use the main orientation of projected model and image contours. Therefore, the operational limits of the presented algorithms are defined by those initial positions where the main orientation axes can be computed in a proper way.

Chapter 8

CONCLUSION

Several mathematical and practical techniques have been presented to solve the correspondence problem based on local and global features for the monocular pose estimation problem. In this context, the main contributions of this work are a new local representation of contours and surfaces and different variants of the ICP algorithm. The first and more evident conclusion derived from this work is that there is no general solution for the correspondence search problem. Each variant offers certain advantages over the other ones for a given pose estimation scenario.

A local representation of free-form contours and surfaces was presented in chapter 2. Since contour segments are approximated as a combination of local motor rotations, they can be combined with the pose estimation constraints in CGA. As the local motors approximate contour and surface segments, structural information is computed from them. Therefore, the proposed local representation offers a suitable mathematical and geometrical description of such entities.

The monogenic signal was used to obtain local features from the image (amplitude, phase and orientation) and to define a contour detection algorithm in section 4.3.3. It allows to extract low and high contrast contour segments of the objects more efficiently and it avoids noise until certain limits. Similar local features were obtained from the contour model points. This was achieved by the introduction of the transition index in section 4.4.1. First, the contour model points are projected onto the image plane and their corresponding local motors are constructed. Based on the local orientation, the transition index is computed as an analog feature to the local monogenic phase. Additionally to these features, the motors also allow a direct computation of the local curvature. This was combined with a strategy to classify the extracted image contour segments and projected model segments in concave, convex and straight segments (semi-local features).

With the sets of image and model features, new variants of ICP algorithms were presented. The Euclidean distance criterion of the classical ICP was combined with additional structural constraints derived from local features in order to define the structural ICP algorithm in section 4.5. For the second variant, the Euclidean distance was replaced by a feature correlation measure as it was described in section 5.2.3. For cases where the tracking assumption is not considered, a new strategy

for the pre-alignment was presented in section 5.4.2. Global orientation information obtained from the Hartley transform and the local orientation are combined in an approach that simplifies the classical pre-alignment procedure. In this case, a 2D feature alignment is done in combination with the correlation ICP algorithm.

According to the experimental results presented in chapter 6, the proposed ICP algorithms perform better than the classical variants. In general, the algorithms are robust against the tracking assumption and in the presence of noise and missing contour information. Furthermore, a better convergence behavior is achieved in comparison with the classical ICP variants. Translating the minimization constraints, feature extraction and correspondence search problem to the image plane results in an important improvement of the monocular pose estimation algorithm. Similarly, the proposed feature alignment approach performed better than the classical pre-alignment based on the principal component analysis (PCA).

The presence of partial occlusions causes a significant distortion in the local and semi-local features. Therefore, the structural and correlation ICP variants are sensitive to partial occlusions. To overcome this problem, several outlier elimination strategies were presented in section 7.2.1. In this case, the angular position of each contour point with respect to its mayor distribution axis was used as an additional feature. Once that the correspondences are found by the correlation ICP algorithm, corresponding points that are outside of a range defined by the angular position are eliminated as outliers.

In section 7.3, a system which integrates several possible correspondence search strategies for the case of the pose estimation with partial occlusions was presented. The selection of the most adequate strategy for the initialization of the pose computation (in order to place the model under tracking assumption conditions) was chosen depending on the distance and orientation between the image contour and the projected model. The presented experiments show the natural limits within which the proposed algorithms are able to find adequate correspondences and therefore a correct pose. Since global and local features are considered by the algorithms, these limits are defined by the range of the poses within which the computation of local and global features is possible with only one image.

8.1 Further Extensions and Applications

The possible extensions of this work are focused on the following problems. First, the presented algorithms can be adapted for articulated objects like robotic arms or human models. In this case, local deformations should be considered that allow to recover more realistic model movements and poses. Secondly, more complete and realistic model objects can be eventually used. Since a large variety of efficient rendering techniques are available, structural information extracted from model and image textures can be used to complement and extend the proposed correspondence

search constraints.

For the presented experiments, only one camera was used. The addition of a stereo or a multiple camera system will consequently increase the converge regions of the algorithms. In these cases, the object can be observed from different view angles. A possible extension for navigation applications is the use of catadioptric camera systems. Because of the catadioptric projection, the structure of 3D models is distorted with respect to the image. Despite of that, the model can be eventually projected onto the image by a catadioptric projection model and the features can be computed from the resulting distorted contour segments.

State of the art tracking and pose estimation systems are intended to work in real time and under general scenarios. In order to improve the computation times of the algorithms and reach real time performance, the standard nearest-neighbor search strategy used in the presented experiments must be further optimized. This can be achieved by the incorporation of search strategies like multidimensional KD-trees [11] which reduce considerably the complexity of the search process.

The use of an algorithm for fitting model to experimental data called Random Sample Consensus (RANSAC, see [42]) has been recently used for tracking and pose estimation applications (see for example [32, 59, 72]). It offers an efficient alternative to fit observed data to a given model in the presence of outliers. The incorporation of RANSAC to the proposed correspondence search and outlier elimination criteria may allow to handle more general occlusions in different pose scenarios.

Because of the variety of possible pose estimation scenarios, complexity of object models and acquisition devices, the ICP algorithms are in most of the cases restricted to a specific application. Since there is no general solution for the correspondence problem, the use of different tracking strategies and the integration of systems based on different kinds of feature information are needed. This represents the natural tendency on the research about this topics, see [57, 59, 80, 83]. It is necessary to mention that more work has to be done in order to develop a complete fully-optimized system based on the presented ICP variants. Despite of that, the approaches presented in this work have the potential to be incorporated in these kind of complex and more robust systems.

BIBLIOGRAPHY

[1] G.J. Agin and T.O. Binford. Computer description of curved objects. *IEEE Transactions on Computers*, 25(4):439–449, 1976.

[2] O. Ait-Aider, P. Hoppenot, and E. Colle. Adaptation of Lowe's camera pose recovery algorithm to mobile robot self-localisation. *Robotica*, 20(4):385–393, 2002.

[3] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 282–296, London, UK, 2002. Springer-Verlag.

[4] H. Araujo, R. Carceroni, and Ch. Brown. A fully projective formulation to improve the accuracy of Lowe's pose-estimation algorithm. *Computer Vision and Image Understanding*, 70(2):227–238, 1998.

[5] K. Arbter, W. E. Snyder, H. Burhardt, and G. Hirzinger. Application of affine-invariant Fourier descriptors to recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):640–647, 1990.

[6] A. C. Atkinson and T.-C. Cheng. Computing least trimmed squares regression with the forward search. *Statistics and Computing*, 9(4):251–263, 1999.

[7] P. Bas, N. Le Bihan, and J.M. Chassery. Color image watermarking using quaternion Fourier transform. In *Conference on Acoustics, Speech, and Signal Processing*, pages 521–524, 2003.

[8] E. Bayro-Corrochano, K. Daniilidis, and G. Sommer. Motor algebra for 3D kinematics: The case of hand-eye calibration. *Journal of Mathematical Imaging and Vision*, 13:79–100, 2000.

[9] E. Bayro-Corrochano and B. Rosenhahn. A geometric approach for the analysis and computation of the intrinsic camera parameters. *Pattern Recognition*, 35:169–186, 2002.

[10] R. Benjemaa and F. Schmitt. Fast global registration of 3D sampled surfaces using a multi-z-buffer technique. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, page 113, Washington, DC, USA, 1997. IEEE Computer Society.

[11] J. L. Bentley. K-D trees for semidynamic point sets. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, New York, NY, USA, 1990.

[12] P. Besl. The free-form surface matching problem. *Machine Vision for Three-Dimensional Scenes, Freeman H. (Editor)*, pages 25–71, Academic Press, San Diego, 1990.

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

[14] P. J. Besl and R.C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Computer Vision, Graphics, and Image Processing*, 33(1):33–80, 1986.

[15] W. Blaschke. *Kinematik und Quaternionen, Mathematische Monographien 4*. Deutsche Verlag der Wissenschaften, 1960.

[16] R.N. Bracewell. Discrete Hartley transform. *Journal of Optical Society of America*, 73(12):1832–1835, 1983.

[17] R.N. Bracewell, O. Buneman, H. Hao, and J. Villasenor. Fast two-dimensional Hartley transform. In *Proceedings of the IEEE*, volume 74, pages 1282 – 1283, Sept. 1986.

[18] C.M. Brown. Some mathematical and representational aspects of solid modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4):444–453, July 1981.

[19] T. Brox, B. Rosenhahn, D. Cremers, and H. Seidel. High accuracy optical flow serves 3-D pose tracking: Exploiting contour and flow based constraints. In A. Leonardis, H. Bischof, and A. Pinz, editors, *9th European Conference on Computer Vision, ECCV 2006, May 2006, Graz, Austria*, number 3952 in LNCS, pages 98–111. Springer-Verlag, Berlin Heidelberg, 2006.

[20] D. Brujic and M. Ristic. Analysis of free form surface registration. In *International Conference of Image Processing. Lausanne, Switzerland*, pages 393–396, 1996.

[21] S. Buchholz and G. Sommer. Introduction to neural computation in Clifford algebra. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 291–314. Springer-Verlag, Heidelberg, 2001.

[22] O. Buneman. Conversion of FFT's to fast Hartley transforms. *SIAM Journal on Scientific and Statistical Computing*, 7(2):624–638, April 1986.

[23] G. Burel and H. Hénocq. Three-dimensional invariants and their application to object recognition. *Signal Process.*, 45(1):1–22, 1995.

[24] R. J. Campbell and P. J. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81(2):166–210, 2001.

[25] M. Chavarria and G. Sommer. Local representation of 3D free-form contours for pose estimation. In *18th International Conference on Pattern Recognition, ICPR 2006, August 20-24, Hong-Kong*, pages 751–754, Washington, DC, USA, 2006. IEEE Computer Society.

[26] M. Chavarria and G. Sommer. Structural ICP algorithm for pose estimation. In *2nd International Conference on Computer Vision Theory and Applications, VISAPP 2007, March 8-11, Barcelona, Spain*, volume 2, pages 341–346, 2007.

[27] M. Chavarria and G. Sommer. Correlation ICP algorithm for pose estimation based on local and global features. In *3nd International Conference on Computer Vision Theory and Applications, VISAPP 2008, Funchal Madeira, Portugal*, volume 2, pages 528–534, 2008.

[28] H.H. Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):530–541, 1991.

[29] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

[30] D. Chetverikov, D. Svirko, D. Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*, pages 545–548, Washington, DC, USA, 2002. IEEE Computer Society.

[31] D. Coeurjolly, S. Miguet, and L. Tougne. Discrete curvature based on osculating circle estimation. In *IWVF-4: Proceedings of the 4th International Workshop on Visual Form*, pages 303–312, London, UK, 2001. Springer-Verlag.

[32] Carlo Colombo, Dario Comanducci, and Alberto Del Bimbo. Robust iris localization and tracking based on constrained visual fitting. *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, 0:454–460, 2007.

[33] A. Dell'Acqua, A. Sarti, and S. Tubaro. Three-view camera calibration using geometric algebra. In *International Conference on Image Processing, ICIP 2003*, volume 1, pages 305–308, 2003.

[34] Y. Dong and G.R. Hillman. Three-dimensional reconstruction of irregular shapes based on a fitted mesh of contours. *Image and Vision Computing*, 19(3):165–176, 2001.

[35] Ch. Dorai, J. Weng, and A. Jain. Optimal registration of object views using range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1131–1138, 1997.

[36] L. Dorst. *Geometric Algebra for Computer Science: An Object-oriented Approach to Geometry*. Morgan Kaufmann Publishers. In Series: The Morgan Kaufmann Series in Computer Graphics, 2007.

[37] O. Faugeras. *Three-dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[38] J. Feldmar and N. Ayache. Rigid, affine and locally affine registration of free-form surfaces. *Int. Journal of Computer Vision*, 18(2):99–119, 1996.

[39] M. Felsberg. Low-level image processing with the structure multivector. Technical Report Number 0203, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, März 2002.

[40] M. Felsberg and G. Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, December 2001.

[41] M. Felsberg and G. Sommer. The monogenic scale-space: A unifying approach to phase-based image processing in scale-space. *J. Math. Imaging Vis.*, 21(1):5–26, 2004.

[42] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[43] D. Gabor. Theory of communication. *Journal of the IEEE*, (93):429–457, 1946.

[44] J. Gallier. *Geometric Methods and Applications for Computer Science and Engineering*. Springer-Verlag, London, UK, 2000.

[45] S. Ganapathy. Decomposition of transformation matrices for robot vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 130–139, 1984.

[46] Ch. Gebken, A. Tolvanen, Ch. Perwass, and G. Sommer. Perspective pose estimation from uncertain omnidirectional image data. In *International Conference on Pattern Recognition (ICPR)*, volume I, pages 793–796, 2006.

[47] G.H. Golub, F. Charles, and V. Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[48] G.H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, Dordrecht, 1995.

[49] W. E. L. Grimsom. *Object Recognition by Computer*. The MIT Press, Cambridge, 1990.

[50] X. Guo. Three-dimensional moment invariants under rigid transformation. In *CAIP '93: Proceedings of the 5th International Conference on Computer Analysis of Images and Patterns*, pages 518–522, Budapest, Hungary. Springer-Verlag, Berlin, Heidelberg, 1993.

[51] R. M. Haralick, C. N. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the 3-point perspective pose estimation problem. 13(3):331–356, December 1994.

[52] S. Hermann and R. Klette. Multigrid analysis of curvature estimators. In *Proc. Image and Vision Computing New Zealand*, pages 434–443, 2003.

[53] S. Hermann and R. Klette. A comparative study on 2D curvature estimators. In *International Conference on Computing Theory and Applications, ICCTA '07*, pages 584–589, 2007.

[54] D. Hestenes. Invariant body kinematics. I: Saccadic and compensatory eye movements. *Neural Networks*, 7(1):65–77, 1994.

[55] D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus*. D. Riedel Publ. Comp. Dordrecht, 1984.

[56] E.M.S. Hitzer. Quaternion Fourier transform on quaternion fields and generalizations. In *Proceedings of Function Theories in Higher Dimensions. Advances in Applied Clifford Algebras*, volume 3, pages 497–517, 2007.

[57] M.l Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 4(1):321–331, 1987.

[58] F.P. Kuhl and Ch. R. Giardina. Elliptic Fourier features of a closed contour. *Computer Graphics and Image Processing*, 18(1):236–258, 1982.

[59] A. Ladikos, S. Behimane, and N. Navab. A real-time tracking system combinig template-based and feature-based approaches. In *2nd International Conference on Computer Vision Theory and Applications, VISAPP 2007, March 8-11, Barcelona, Spain*, pages 325–332, Portugal, 2007.

[60] S. Lavallee and R. Szeliski. Recovering the position and orientation of free-form objects from image contours using 3D distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):378–390, 1995.

[61] V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[62] H. Li, D. Hestenes, and A. Rockwood. Generalized homogeneous coordinates for computational geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 27–59. Springer-Verlag, Heidelberg, 2001.

[63] Ch. S. Lin and Ch. L. Hwang. New forms of shape invariants from elliptic Fourier descriptors. *Pattern Recognition*, 20(5):535–545, 1987.

[64] Y. Liu and W. Heidrich. Interactive 3D model acquisition and registration. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 115–122, Washington, DC, USA, 2003. IEEE Computer Society.

[65] Y. H. Liu. Constraints for closest point finding. *Pattern Recognition Letters*, 29(7):841–851, 2008.

[66] Y. H. Liu and Y. Wang. Evaluating 3D-2D correspondences for accurate camera pose estimation from a single image. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 703–708, 2003.

[67] Y.H. Liu. Improving ICP with easy implementation for free-form surface matching. *Pattern Recognition*, 37(2):211–226, February 2004.

[68] E. H. Lockwood. *A Book of Curves*. Cambridge University Press, 1967.

[69] R.E. Loke, M.M. Bayer, D.G. Mann, and J.M.H. du Buf. Diatom recognition by convex and concave contour curvature. In *Oceans 2002 MTS/IEEE*, volume 4, pages 2457–2465, Oct. 2002.

[70] A. Lorusso, D. W. Eggert, and R. B. Fisher. A comparison of four algorithms for estimating 3-D rigid transformations. In *BMVC '95: Proceedings of the 1995 British conference on Machine vision (Vol. 1)*, pages 237–246, Surrey, UK, UK, 1995. BMVA Press.

[71] Ch. P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.

[72] L. Lu, X. Dai, and G. D. Hager. Efficient particle filtering using RANSAC with application to 3D face tracking. *Image Vision Computing*, 24(6):581–592, 2006.

[73] M. Marji. *On the Detection of Dominant Points on Digital Planar Curves*. PhD thesis, Wayne State University, Detroit, Michigan, 2003.

[74] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-D model construction. In *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, page 879, Washington, DC, USA, 1996. IEEE Computer Society.

[75] T. Masuda and N. Yokoya. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding*, 61(3):295–307, 1995.

[76] A. McIvor. Background subtraction techniques. In *Proc. of Image and Vision Computing, Auckland, New Zealand*, 2000.

[77] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic b-splines. In *Proc. Robotics and Automation*, volume 3, pages 764–769, 1986.

[78] V. Murino, L. Ronchetti, U. Castellani, and A. Fusiello. Reconstruction of complex environments by robust pre-aligned ICP. In *Proc. Third Int. Conf. on 3-D Digital Imaging and Modeling*, pages 187–194, 2001.

[79] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introducion to Robotic Manipulation*. CRC Press, 1994.

[80] H. Najafi, Y. Genc, and N. Navab. Fusion of 3D and appearance models for fast object detection and pose estimation. In *Asian Conferenceon Computer Vision Volume II*, pages 415–426, 2006.

[81] R. Nevatia and T.O. Binford. Description and recognition of curved objects. *Artificial Inteligence*, 8(1):77–98, 1977.

[82] PACLib. *(Homepage of the Kieler Perception-Action-Components (PAC) Library) http://www.ks.informatik.uni-kiel.de/ paclib/.*

[83] G. Panin and A. Knoll. Fully automatic real-time 3D object tracking using active contour and appearance models. *Journal of Multimedia (JMM)*, 1(1):62–70, 2006.

[84] G. Panin, A. Ladikos, and A. Knoll. An efficient and robust real-time contour tracking system. In *ICVS '06: Proceedings of the Fourth IEEE International Conference on Computer Vision Systems*, page 44, Washington, DC, USA, 2006. IEEE Computer Society.

[85] C. Perwass. *Geometric Algebra with Applications in Engineering*. Series: Geometry and Computing, Vol. 4, Springer-Verlag, Berlin, Heidelberg, 2009.

[86] T.Q. Phong, R. Horaud, A. Yassine, and P.D. Tao. Object pose from 2D to 3D point and line correspondences. *International Journal of Computer Vision*, 15(3):225–243, July 1995.

[87] B. Rosenhahn. Pose estimation revisited. Technical Report Number 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, September 2003.

[88] B. Rosenhahn, T. Brox, D. Cremers, and H. Seidel. A comparison of shape matching methods for contour based pose estimation. In *11th International Workshop on Combinatorial Image Analysis (IWCIA)*. Berlin, Germany, LNCS Springer-Verlag, 2006.

[89] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of free-form surface models. In B. Michaelis and G. Krell, editors, *25. Symposium für Mustererkennung, DAGM 2003, Magdeburg*, volume 2781 of *LNCS*, pages 574–581. Springer-Verlag, Berlin, 2003.

[90] B. Rosenhahn, C. Perwass, and G. Sommer. Free-form pose estimation by using twist representations. *Algorithmica*, 38:91–113, 2004.

[91] B. Rosenhahn, C. Perwass, and G. Sommer. Pose estimation of 3D free-form contours. *Int. Journal of Computer Vision*, 62(3):267–289, 2005.

[92] B. Rosenhahn and G. Sommer. Pose estimation in conformal geometric algebra, part I: The stratification of mathematical spaces. *Journal of Mathematical Imaging and Vision*, 22:27–48, 2005.

[93] B. Rosenhahn and G. Sommer. Pose estimation in conformal geometric algebra, part II: Real-time pose estimation using extended feature concepts. *Journal of Mathematical Imaging and Vision*, 22:49–70, 2005.

[94] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling*, pages 145–152, Quebec City, Canada, 2001.

[95] F. A. Sadjadi and E. L. Hall. Three-dimensional moment invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 2(2):127–136, 1980.

[96] S.J. Sangwine. The problem of defining the Fourier transform of a colour image. In *International Conference on Image Processing. ICIP 98*, pages 171–175, 1998.

[97] M. Sarfraz. Object recognition using Fourier descriptors: Some experiments and observations. In *CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, pages 281–286, Washington, DC, USA, 2006. IEEE Computer Society.

[98] L. Shang, P.Jasiobedzki, and M. Greenspan. Discrete pose space estimation to improve ICP-based tracking. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 523–530, Washington, DC, USA, 2005. IEEE Computer Society.

[99] Limin Shang, Piotr Jasiobedzki, and Michael Greenspan. Model-based tracking by classification in a tiny discrete pose space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):976–989, 2007.

[100] G. Sharp, S. Lee, and D. Wehe. ICP registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102, 2002.

[101] F. Solina and R. Bajcsy. Recovery of parametric models from range images: the case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(2):131–147, 1990.

[102] G. Sommer, editor. *Geometric Computing with Clifford Algebras*. Springer-Verlag, Heidelberg, 2001.

[103] G. Sommer, G. Granlund, O. Granert, M. Krause, K. Nordberg, Ch. Perwass, R. Söderberg, F. Vikstén, and M. Chavarria. Visatec final report. Technical report, Dept. EE, Linköping University, May 2005. http:www.visatec.info.

[104] G. Sommer and D. Zang. Parity symmetry in multi-dimensional signals. *Communications in Pure and Applied Analysis*, 6(3):829–852, 2007.

[105] F. Stein and G. Medioni. Structural indexing: Efficient 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145, 1992.

[106] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[107] A. Tolvanen, C. Perwass, and G. Sommer. Projective model for central catadioptric cameras using Clifford algebra. In *27. Symposium für Mustererkennung, DAGM 2005, Wien, 29.8.-2.9.005*, volume 3663 of *LNCS*, pages 192–199. Springer-Verlag, Berlin, Heidelberg, 2005.

[108] E. Trucco, A. Fusiello, and V. Roberto. Robust motion and correspondence of noisy 3-d point sets with missing data. *Pattern Recogn. Lett.*, 20(9):889–898, 1999.

[109] M.F. Wu and H.T. Sheu. Representation of 3D surfaces by two-variable Fourier descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):858–863, 1998.

[110] D. Zang and G. Sommer. The monogenic curvature scale-space. In R. Reulke, U. Eckardt, B. Flach, U. Knauer, and K. Polthier, editors, *11th International Workshop on Combinatorial Image Analysis, IWCIA'06, Berlin*, volume 4040 of *LNCS*, pages 320–332. Springer-Verlag, Berlin, Heidelberg, 2006.

[111] D. Zang and G. Sommer. Signal modeling for two-dimensional image structures. *Journal of Visual Communication and Image Representation*, 18(1):81–99, 2007.

[112] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. Journal of Computer Vision*, 13(2):119–152, 1994.

[113] T. Zinsser, L. Schmidt, and H. Niemann. A refined ICP algorithm for robust 3-D correspondence estimation. In *International Conference on Image Processing, ICIP 2003*, volume 2, pages 695–698, 2003.