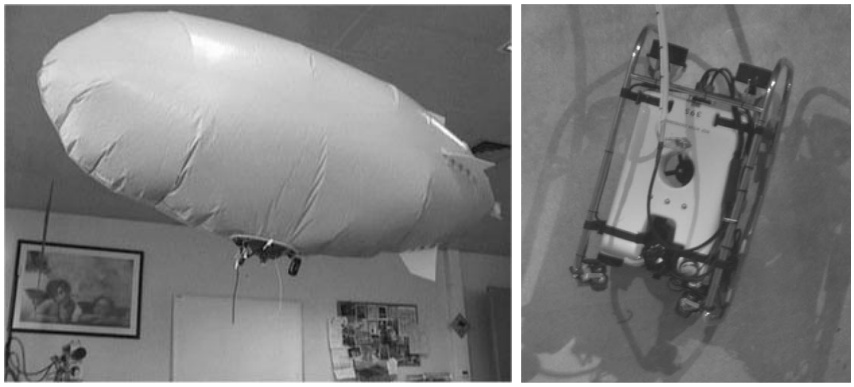
UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO



VISION BASED STATION KEEPING AND DOCKING FOR FLOATING ROBOTS

Sjoerd van der Zwaan
(Licenciado)

Dissertação para a obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador: *Doutor José Alberto Rosado dos Santos Victor*

Presidente: *Doutor João José dos Santos Sentieiro*

Vogais: *Doutor Henrik I Christensen*
Doutor José Alberto Rosado dos Santos Victor
Doutor Carlos Jorge Ferreira Silvestre

Maio de 2001

Agradecimentos

O trabalho apresentado nesta tese foi desenvolvido ao longo de um ano e três meses, no laboratorio de Visão do Instituto de Sistemas e de Robotica em Lisboa, sendo ainda integrado no projecto NARVAL[1]. A ajuda que recebi e a experiência que ganhei durante este periodo deveu-se a um conjunto de pessoas a quem gostava de agradecer aqui.

Em primeiro lugar, quero agradecer ao Prof. José Santos-Victor, por me ter convidado trabalhar com ele, pela excelente orientação, por me ter introduzido aos hábitos, cultura e bom espírito do Vislab e pelo amizade desenvolvida ao longo dos anos.

Ao Alexandre Bernardino, cujo disponibilidade, amizade e ajuda foram fundamentais na elaboração desta tese.

A todas as pessoas ligados ao Vislab e ao Instituto de Sistemas e Robótica, Pólo Lisboa, com quem tive a oportunidade de trabalhar. Em particular, quero agradecer a ajuda, amizade e convívio de Nuno Gracias, José António Gaspar (o optimista bem informado), António Bastos, Luis Jordão, Niall Winters, Etienne Grossmann, Pedro Soares, João Maciel, César Silva e Javier Mínguez (companheiro no Japão).

Aos meus colegas Carlos Marques e José Moreira, por terem partilhado os momentos altos e baixos do curso de Mestrado e pela amizade.

A toda a equipa do NARVAL, pelo suporte financeiro do projecto e pelos exce-

[1]ESPRIT-LTR Project 30185, NARVAL - Navigation of Autonomous Robots via Active Environmental Perception, http://gandalf.isr.ist.utl.pt/narval/index.htm

lentes e únicos condições de trabalho oferecidos. Em especial, agradeço a amizade, ajuda e convívio dos colegas Jonas Hornstein (Skål!), Matteo Perrone e Stefan Rolfes.

A Fundação de Ciência e de Tecnologia, pelo apoio financeiro através da bolsa do Mestrado, Praxis XXI/BM/20759/99.

Finalmente quero agradecer aos meus pais, por me terem sempre apoiado nas minhas decisões, e a Sónia, por tudo!

Acknowledgements

The work presented in this thesis was completed in the Robot and Computer Vision Lab of the Institute of Systems and Robotics, Lisbon, over a period of 1 year and three months. This work was also carried out within the NARVAL[2] project. The experience I gained during this period is mainly due to a group of people that I would like to thank.

Firstly, I would like to thank Prof. José Santos-Victor, for his excellent guidance and supervision, for having introduced me to the habits, culture and positive spirit of the Robot and Computer Vision Lab, and for his friendship.

To Alexandre Bernardino, for his fundamental contributions to this thesis, his patience and his friendship.

To all the other members of the Lab and the Institute of Systems and Robotics (ISR-Lisbon), with whom I have had the opportunity and pleasure to work with. I would especially like to thank for their support and friendship: Nuno Gracias, António Gaspar, António Bastos, Luis Jordão, Niall Winters, Etienne Grossmann, Pedro Soares, João Maciel, César Silva and Javier Mínguez.

To Carlos Marques and José Moreira, for their friendship and for sharing all the ups and downs of post-graduate life.

To the NARVAL team, the project's financial supporters and for the excellent and unique working conditions provided. In particular, I would like to thank the

[2]ESPRIT-LTR Project 30185, NARVAL - Navigation of Autonomous Robots via Active Environmental Perception, http://gandalf.isr.ist.utl.pt/narval/index.htm

team-members Jonas Hornstein, Matteo Perrone and Stefan Rolfes for their support and friendship.

To the Portuguese Foundation of Science and Technology, for the financial support provided by the scholarship PraxisXXI/BM/20759/99.

Finally, I would like to thank my parents, who always supported me in my decisions and to Sónia, for everything!

Resumo

Nesta tese desenvolvem-se metodologias baseadas em visão para a estabilização e acostagem de robots flutuante. Devido à perturbações exteriores, esta tarefa é fundamental por permitir estabilizar o veículo relativamente a um sistema de coordenadas externo. Utiliza-se um sistema de visão para medir as deformações na imagem quando o veículo se desloca relativamente à estação de acostagem. Usa-se um superfície planar como plano de referência para o seguimento visual de qualquer região texturada, baseado em transformações projectivas. O sistema de seguimento integra fluxo óptico e métodos baseados em correlação que, recorrem a modelos de movimento para amostrar as deformações de imagem expectáveis ao longo do tempo. Estes modelos são actualizados para se adaptarem ao movimento do robot. A informação do sistema de seguimento é utilizado para calcular homografias entre imagens e para alinhar a imagem actual com uma imagem de referência inicial. Estas transformações revelam o movimento real da câmara no espaço 3D e permitem reconstruir a trajectória da câmara. Esta informação é usada para o controlo dos robots. Estuda-se o controlo visual baseado em medidas 2D ou 3D. As estratégias de controlo resultantes são ilustradas com experiências reais com um dirigível e um robot submarino.

Palavras-chave: Seguimento visual, fluxo óptico, transformações projectivas planares, controlo baseado em visão, robots submarinos, dirigíveis, estabilização e acostagem.

Abstract

This thesis describes a method of vision based station keeping and docking for floating robots. Due to the motion disturbances in the environment, these tasks are important to keep the vehicle stabilized relative to an external reference frame. A vision-based tracking system is used to measure the image deformations, as the vehicle moves with respect to a docking station. A planar surface is chosen as a reference plane which allows visual tracking of a naturally textured region, based on planar projective transformations. The tracker integrates optic flow computation within the region with a correlation based method using motion models that sample the expected image deformations over time. These models are updated by the robot history of motion so as to accommodate predicted/future camera motions. The information provided by the tracker is then used to calculate inter-image homographies that register the current viewed image with some initial reference image. These transformations reveal the real camera motion in 3D space and allow to reconstruct the camera trajectory. This information can then be used to control the robots. Both 3D visual servoing as image based visual servoing approaches are studied. The resulting control strategies are illustrated with real experiments with a blimp and an underwater robot.

Key-words: visual tracking, optic flow, planar projective motion models, visual servoing, underwater robots, blimp, station keeping, docking.

Contents

Chapter 1

Introduction

So far, in the field of robotics, a lot of effort and progress has been made on visual navigation of mobile robots. Most cases however deal with the navigation of wheeled mobile bases that are restricted to move on the ground plane. More complex systems deal with what we call floating robots that move in 3D space. Typical examples are unmanned aerial vehicles or underwater robots.

Research on the utilization of unmanned aerial vehicles has grown with an increasing interest on robotic airships, also known as blimps or lighter-than-air vehicles. The motivation behind it is that airships outperform airplanes and helicopters in low-speed, low altitude applications, having an enormous potential for tasks like environmental and traffic monitoring, climate research, transportation, etc. Several references can be found in literature about airship modeling for autonomous navigation [5, 10, 17].

In marine research an increase of interest on the utilization of autonomous underwater vehicles (AUV's) and remote operated underwater vehicles (ROV's) is noted during the last decade. Such underwater vehicles can be inserted into a wide variety of applications related to underwater management, monitoring and manipulation tasks. Some examples are map building of the ocean floor, pipe-line inspection, manipulation tasks in the off-shore industry, etc. A recent shift of interest is towards the use of underwater robots in coastal regions, which are socially and economically

important areas. These regions are at shallow waters and thus affected by surface phenomena such as waves, tides, weather conditions, air and water interaction and commercial and recreational navigation. Therefore, they are characterized by fast dynamics and high concentration of energy, giving rise to strong perturbations. This demands strong requirements on controlling any vehicles in these areas. In literature, various references can be found for modeling and control of underwater vehicles. A complete textbook on these topics can be found in [8]. Other references deal with robust control of vehicle motion in shallow waters [2, 24].

The work presented in this thesis is integrated in the NARVAL[1] project, for which one of the main goals is the design and implementation of reliable navigation systems of limited cost for mobile robots in unstructured environments. For demonstration, two experimental setups were used, namely: (i) an airborne blimp and (ii) a remote operated underwater vehicle (ROV). The problem addressed is that of station keeping and docking with the indoor blimp and ROV, based on visual input. Since the kinematics and dynamics of an underwater vehicle are a reasonably approximated by those of an airship, we use the blimp as a test-bed for experiments in a laboratory environment.

Maintaining a vehicle fixed at a given position and orientation is an important behavior, required for many tasks. To do so, the floating vehicles need to sense and actively control their position and orientation to avoid drift caused by unknown currents and forces. For underwater robots, classical sensors for position and orientation include magnetic compasses, depth-sensors and translational motion sensors (i.e. accelerometers or Doppler velocity logs). The main drawbacks of using magnetic compasses is that they have a slow update rate and do not perform well in the presence of metallic structures such as ships. Translational motion sensors are integrating sensors (odometry) and are thus likely to accumulate errors (drift). As-

[1]ESPRIT-LTR Project 30185, NARVAL - Navigation of Autonomous Robots via Active Environmental Perception, http://gandalf.isr.ist.utl.pt/narval/index.htm

suming that the tasks of docking and station keeping are defined relative to short range regions in the environment, one can use vision in order to extract information about the vehicle's pose and use this information for visual servoing.

Various references can be found on automatic vision based station keeping for underwater vehicles and blimps. In [23], optical flow information is used to reconstruct the vehicle 3D motion so as to realize station keeping. However, not all 3D motions are observable from the optical flow of feature points in the image. Especially, feature displacements cannot be used to determine the absolute range without other knowledge of the scene structure. In [22], the problems related to this poor conditioning are pointed out and an alternative approach is proposed by assuming that certain motions can be sensed by using other sensing modalities.

More recently, the station keeping problem has been addressed within the framework of visual servoing architectures [27, 6, 20, 18], which can be mainly classified into *3D visual servoing* and *image based servoing*. In the first case, image information is interpreted so as to reconstruct the 3D scene or motions and this information is then used for navigation. With image based visual servoing, image feature parameters are directly measured in the image plane and regulated to some desired value. Visual servoing techniques usually deal with regulating a kinematic error function to zero. The problem of how to introduce dynamics into the image plane is addressed in [32] for the case of station keeping with a blimp. Applications of image based visual servoing for station keeping for underwater robots can be found in [25] and [19]. However, in both cases the emphasis is on the visual servoing part and the visual tracking is assumed to be perfect. This is not always realistic for real implementations, since in a marine environment, images in general contain non-uniform lighting, low contrast, marine snow and thus lack of necessary features.

Some of the most reliable methods for tracking apply numerical optimization to minimize the error between a reference image and a target region, subject to a pa-

rameterized deformation model. Often, the L_2 norm is used to measure the error, and the approach is referred to as sum-of-squared-difference (SSD) method. Some variants of this method deal with tracking simple (affine or translational) deformations [12] and more complex projective planar [29, 21] and piecewise bilinear transformations over large images regions. A more general derivation of the SSD algorithm, described in [9], uses optimization techniques in the specific context of image registration. Motivated by the latter approach, we developed a tracking system based on full planar projective transformations, thus accommodating a wide set of possible image deformations.

1.1 Objectives of the thesis

In this thesis the major emphasis is put on the visual processing part, requiring both robust and real time tracking of image features, which is necessary to allow visual servoing. The approach adopted determines camera motion from the registration between the current live image and an initial reference image. Assuming that an image patch with sufficient texture is initially identified by the user, the temporal changes of this image patch induced by the vehicle's motion are tracked based upon planar projective transformations. To accomplish this goal, a set of motion vectors is used that sample the search space for expected image deformations. With this method, we avoid exhaustive search on the parameter space, allowing high tracking frequencies required for real time applications. To enhance robustness, optical flow information is integrated, providing the tracker with an initial estimate of the current transformation parameters.

The tracked image region is then used as a visual landmark and its deformation parameters convey information about the camera motion, the vehicle motion and scene structure. Under the planarity assumption, it follows that it is possible to

obtain a scaled Euclidean reconstruction of the relative camera trajectory in 3D space from the image registration [7]. No further assumptions are made with respect to the landmark, so that any textured region in the image plane can be a candidate, as long as it is locally planar in 3D space. This provides a means of self-localizing the robot relative to any naturally textured scene in an unstructured environment.

Various methods for vision based station keeping are then proposed and formulated into a visual servoing approach. Both the *3D visual servoing* approach and the *image based visual servoing* approach will be discussed and compared. To allow station keeping over a wider range, we also introduce the use of image stabilization with an on-board pan and tilt camera and discuss how this should be integrated in the various control strategies. In addition, the vehicle non-holonomic constraints are taken into account in the design of the control laws. All methods are illustrated by data obtained from real experiments with the blimp and ROV.

1.2 Structure of the thesis

This thesis is organized as follows. Chapter 2 introduces the fundamental aspects of projective geometry required to understand the work done on visual tracking and motion analysis. Chapter 3 describes and discusses the tracking system developed for this work. In Chapter 4, the blimp and the underwater robot used for experiments are specified and modeled. Chapter 5 introduces the various visual servoing techniques used for realizing station keeping and docking. Chapter 6 shows experimental results of station keeping and docking test, obtained from real data for both the blimp an the underwater robot. Finally, in Chapter 7, conclusions are drawn and future work is indicated.

Chapter 2

Projective Geometry and Transformations

This chapter summarizes the fundamental aspects of projective geometry required to understand the work done on visual tracking and motion analysis in this thesis. This chapter is included for the sake of completeness of the thesis. The reader familiar with these concepts might skip reading this chapter. A thorough discussion of these matters can be found in [7, 13].

One important reason to study projective geometry is that most system with lenses, whether it is a biological one like the human visual system or machine vision, can be modeled by a system that performs a perspective transformation. Under this transformation, points of the world are projected onto the retinal or image plane, involving a non-linear mapping (perspective distortion) when represented in the usual Euclidean space. In the projective space, this mapping can be represented with linear equations by using the homogeneous representation of points and lines, as will be seen in what follows. Another reason to study projective geometry is that it provides an intuitive insight into the geometry of imaging systems.

2.1 The projective plane

In a n-dimensional projective space, $\mathbb{P}^n$, a point is represented by a $n+1$ coordinate vector $\mathbf{x} = (x_1, ..., x_{n+1})$, where at least one of the x_i is non-zero. This is the homogeneous representation of a point and it is defined up to a scale factor. This means that any two coordinates $\mathbf{x} = (x_1, ..., x_{n+1})$ and $\mathbf{x}' = (x'_1, ..., x'_{n+1})$ in the projective space represent the same point if and only if $\mathbf{x'_i} = k\mathbf{x_i}$ for any non-zero k. The space $\mathbb{P}^2$ is known as the projective plane.

Homogeneous representation of points and lines. A point $\mathbf{x} = (x, y)$ in $\mathbb{R}^2$, lies on the line $\mathbf{l}$ represented by the vector $\mathbf{l} = (a, b, c)$ if and only if the equation $ax + by + c = 0$ is satisfied. This can be written as the inner product of the vectors $(x, y, 1).(a, b, c)^\top = 0$. However, this condition is satisfied for any vector $k(x, y, 1)$ with $k \neq 0$, which means that the class of vectors given by (kx, ky, k) represent the same point (x, y) in $\mathbb{R}^2$. The class of equivalent vectors related by such a scaling factor is known as a *homogeneous vector*. Any arbitrary vector of the type (x_1, x_2, x_3) in $\mathbb{P}^2$ can thus be seen as a homogeneous representation of a point $(x_1/x_3, x_2/x_3)$ in $\mathbb{R}^2$. Similarly, different choices of a, b and c specify different lines. This representation is not unique since the vector $k(a, b, c)$ represents the same line for any non-zero k. Therefore, a line is also represented by a homogeneous vector. The set of equivalent classes of vectors in $\mathbb{R}^3$ forms the projective space $\mathbb{P}^2$, represented by the set of homogeneous vectors.

The use of homogeneous coordinates allows to obtain simple linear expressions when dealing with intersections of lines and lines passing through points. In the projective plane, we can write the equation of a line as:

$$\mathbf{x}^\top \mathbf{l} = 0 \tag{2.1}$$

This equation states that the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if the inner

product of the homogeneous vector representation is equal to zero. The intersection of two lines $\mathbf{l} = (a, b, c)$ and $\mathbf{l}' = (a', b', c')$ is given by the point:

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' \tag{2.2}$$

From the triple scalar product identity one obtains $(\mathbf{l} \times \mathbf{l}')^T\mathbf{l} = (\mathbf{l} \times \mathbf{l}')^T\mathbf{l}' = 0$. Considering the vector $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$ as representing a point, it can be verified that $\mathbf{x}^\top\mathbf{l} = 0$ and also $\mathbf{x}^\top\mathbf{l}' = 0$. From equation (2.1), it follows that the point $\mathbf{x}$ lies on both lines $\mathbf{l}$ and $\mathbf{l}'$ and therefore is the intersection of the two lines.

Similarly, it is also possible to obtain an expression for the line joining two points. Considering two points $\mathbf{x}$ and $\mathbf{x}'$, their vector cross-product can be thought of as representing the line:

$$\mathbf{l} = \mathbf{x} \times \mathbf{x}' \tag{2.3}$$

From the triple scalar product identity $\mathbf{x}^T(\mathbf{x} \times \mathbf{x}') = \mathbf{x}'^T(\mathbf{x} \times \mathbf{x}') = 0$, it can be seen that both points $\mathbf{x}$ and $\mathbf{x}'$ lie on this line.

Points and lines at infinity. Two lines $\mathbf{l} = (a, b, c)$ and $\mathbf{l}' = (a, b, c')$ that differ only in their third element are parallel lines in $\mathbb{R}^2$. From equation (2.2) we observe that their point of intersection in the projective plane is at $\mathbf{x} = (c' - c).(b, -a, 0)$. This point does not correspond to any finite point in $\mathbb{R}^2$, since it implies a division by zero. Intuitively, this suggests that parallel lines in $\mathbb{R}^2$ meet at infinity. In the projective plane however, the intersection of parallel lines is given by a homogeneous vector of the type $(x_1, x_2, 0)$. Points in the projective plane for which the last element is equal to zero era called *ideal points* or *points at infinity*.

The only line that satisfies the condition $\mathbf{x}^\top\mathbf{l} = 0$ for any x_1, x_2 is given by $\mathbf{l}_\infty = (0, 0, c)$. This is the *line at infinity*.

The concepts of ideal points and the line at infinity simplify the intersection properties of points and lines. In the projective plane $\mathbb{P}^2$, any two lines intersect at a single point and any two points are joined by one single line. This is not true for $\mathbb{R}^2$,

where parallel lines do not intersect and the points at infinity are not included. The projective space $\mathbb{P}^2$ can thus be thought of as $\mathbb{R}^2$, augmented by points at infinity.

2.2 Planar projective transformations

The image plane of a camera can be modeled as a projective plane. In Figure 2.1 we illustrate the case when a camera is observing a plane in the 3D space. Points in the observed space are projected onto the image plane according to a perspective projection model.

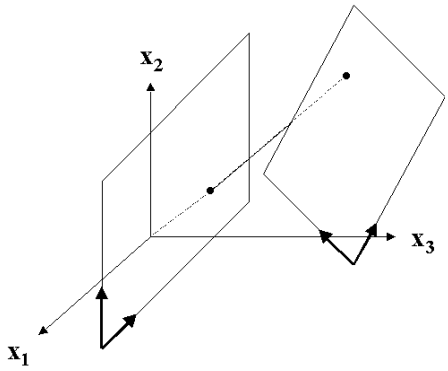


Figure 2.1: Perspective projection defines a mapping of points on one plane to another plane

If a coordinate system is defined on each plane and points are represented in homogeneous coordinates, then the mapping that relates points on the image plane to the points on the world plane is given by a projectivity.

Definition 2.1 (Projectivity). *A projectivity is an invertible mapping, h, from $\mathbb{P}^2$ to itself such that three points $\mathbf{x_1}, \mathbf{x_2}$ and $\mathbf{x_3}$ lie on the same line if and only if $h(\mathbf{x_1}), h(\mathbf{x_2})$ and $h(\mathbf{x_3})$ do.*

Projectivities preserve the collinearity property (a line one a plane is mapped to a line in the other plane) but in general do not preserve parallel lines. Synonyms

of a projectivity are collineation or homography. A projectivity can be written algebraically as a linear transformation on homogeneous coordinates in the projective space $\mathbb{P}^2$. These transformations are planar projective transformations and are represented by a non-singular 3×3 matrix H such that:

$$\mathbf{x}' = H\mathbf{x}, \quad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2.4}$$

where $\mathbf{x}'$ and $\mathbf{x}$ are homogeneous coordinate vectors of points on the image plane and world plane, respectively. The homography H is defined up to a scale factor and called a homogeneous matrix, thus having eight degrees of freedom.

Under perspective imaging, it is common to use inhomogeneous coordinates instead of homogeneous because they can be measured directly from the image plane. Let the inhomogeneous coordinates of a pair matching points $\mathbf{x}$ and $\mathbf{x}'$ in the world and image plane be (x,y) and (x',y') respectively. The projective transformation that maps the points (x, y) from the world plane to the image points (x', y') can then be written in inhomogeneous form as:

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.5}$$

2.3 Homography estimation

In this section we discuss algorithms for estimating homographies from point-to-point correspondences on planes. Given a set of points $\mathbf{x_i}$ and a corresponding set of points $\mathbf{x'_i}$ in $\mathbb{P}^2$, the estimation problem is to compute a 3×3 matrix H such that $\mathbf{x'_i} = H\mathbf{x_i}$ for each i. A minimum of four point-to-point correspondences is required to estimate the homography. This is the *minimum solution*. If more than four point correspondences are given, then these correspondences may not be fully compatible due to small errors in the point measurements resulting from image noise. It is thus necessary to find the best possible estimate given the data set. This can be realized

by minimizing some algebraic or geometric cost function. The most commonly used approach uses an algebraic distance to be minimized with least-square techniques, resulting into low computational requirements. Geometric distance functions result into non-linear optimization problems but, in general, allow to define more intuitive cost-functions. The rest of this section focuses on least-square type algorithms.

The minimal solution. Given a set of four 2D point-to-point correspondences, $\mathbf{x_i} \leftrightarrow \mathbf{x'_i}$, the homography H can be determined up to a unique solution. Consider the equation $\mathbf{x'_i} = H\mathbf{x_i}$ for each correspondence to be expressed in terms of a vector cross-product as $\mathbf{x'_i} \times H\mathbf{x_i} = 0$. If the j-th row of H is denoted by $\mathbf{h_j}^\top$, we have:

$$H\mathbf{x_i} = \begin{bmatrix} \mathbf{h_1}^\top \mathbf{x_i} \\ \mathbf{h_2}^\top \mathbf{x_i} \\ \mathbf{h_3}^\top \mathbf{x_i} \end{bmatrix} \tag{2.6}$$

Writing $\mathbf{x'_i} = (x'_i, y'_i, w'_i)^\top$, the cross-product is given by:

$$\mathbf{x'_i} \times H\mathbf{x_i} = \begin{bmatrix} y'_i \mathbf{h_3}^\top \mathbf{x_i} - w'_i \mathbf{h_2}^\top \mathbf{x_i} \\ w'_i \mathbf{h_1}^\top \mathbf{x_i} - x'_i \mathbf{h_3}^\top \mathbf{x_i} \\ x'_i \mathbf{h_2}^\top \mathbf{x_i} - y'_i \mathbf{h_1}^\top \mathbf{x_i} \end{bmatrix} = 0 \tag{2.7}$$

Since $\mathbf{h_j}^\top \mathbf{x_i} = \mathbf{x_i}^\top \mathbf{h_j}$ for $j = 1, ..., 3$, this gives a set of three equations in the entries of H, which may be written in the form:

$$\begin{bmatrix} \mathbf{0}^\top & -w'_i \mathbf{x_i}^\top & y'_i \mathbf{x_i}^\top \\ w'_i \mathbf{x_i}^\top & \mathbf{0}^\top & -x'_i \mathbf{x_i}^\top \\ -y'_i \mathbf{x_i}^\top & x'_i \mathbf{x_i}^\top & \mathbf{0}^\top \end{bmatrix} \begin{bmatrix} \mathbf{h_1} \\ \mathbf{h_2} \\ \mathbf{h_3} \end{bmatrix} = 0 \tag{2.8}$$

This equation has the form $A_i \mathbf{h} = 0$, where A_i is a 3×9 matrix determined from a point correspondence and $\mathbf{h}$ a 9×1 vector containing the entries of H. This system is linear in $\mathbf{h}$, whereas the elements of A_i are quadratic in the known point coordinates. The system described by equation (2.8) contains three linear equations for each correspondence but only two are linearly independent. Stacking the four 2×9 matrices obtained from each correspondence into a single 8×9 matrix, A, one ends up with a system of eight equations with eight unknowns ($\mathbf{h}$ is defined up

to a scale vector), of the type $A\mathbf{h} = 0$. To avoid the obvious solution $\mathbf{h} = 0$, an additional constraint such as $\|h\| = 1$ needs to be imposed. To preserve the linear independency of the system, no more than two points can be collinear.

Equation (2.8) holds for any coordinate representation of the type (x'_i, y'_i, w'_i) and since the homogeneous coordinate vector is defined up to a scale vector, it is possible to choose $w'_i = 1$ and set x'_i and y'_i equal to the coordinates as measured from the image.

The overdetermined solution. If more than four point correspondences are given, then the linear system $A\mathbf{h} = 0$ becomes overdetermined. If the position of points is exact, then a unique solution exists. However, since points measurements are always corrupted with noise, there will be no exact solution. This requires the definition of a cost function to be minimized so as to obtain the best possible estimate of $\mathbf{h}$ in a least square sense. To avoid the obvious solution $\mathbf{h} = 0$, an additional constraint such as $\|\mathbf{h}\| = 1$ needs to be imposed.

A possible cost function could be used to minimize the norm $\|A\mathbf{h}\|$ subject to the additional constraint of $\|h\| = 1$. This is known as the *Direct Linear Transformation* algorithm (*DLT*-algorithm) [13].

The *DLT*-algorithm:

- Given $n \geq 4$ point correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x'_i}$ compute the matrix A_i from equation (2.8) for each correspondence . Only the first two rows need to be used.

- Stack the n 2×9 matrices into a single $2n \times 9$ matrix A.

- Obtain the singular value decomposition of A. The unit singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. If $A = UDV^T$

with D diagonal with positive entries arranged in descending order down the diagonal, then $\mathbf{h}$ is given by the last column of V.

- The matrix H is determined from $\mathbf{h}$.

The inhomogeneous solution. An alternative solution is to turn the homogeneous matrix H into a inhomogeneous representation by imposing $h_9 = 1$. Equation (2.8) then turns into:

$$\begin{bmatrix} 0 & 0 & 0 & -x_i w'_i & -y_i w'_i & -w_i w'_i & x_i y'_i & y_i y'_i \\ x_i w'_i & y_i w'_i & w_i w'_i & 0 & 0 & 0 & -x_i x'_i & -y_i x'_i \end{bmatrix} \tilde{\mathbf{h}} = \begin{bmatrix} -w_i y'_i \\ w_i x'_i \end{bmatrix} \quad (2.9)$$

Using four or more point correspondences, these equations result into a system of the type $M\tilde{\mathbf{h}} = \mathbf{b}$. This can be solved for the minimum case into a unique solution and for the overdetermined case by minimizing the norm of $\parallel M\tilde{\mathbf{h}} - \mathbf{b} \parallel$.

One drawback of turning to the inhomogeneous solution is that not all true solutions can be reached. For example, those solutions that require h_9 to be zero (which involves mappings of finite points on one plane to infinite points on the other plane) can not be obtained. This suggests that the solution becomes unstable for those situations for which $h_9 \approx 0$.

Normalized *DLT*-algorithm. Image coordinates are sometimes given with the origin at the top-left of the image or alternatively with the origin at the image center. Unfortunately, the *DLT*-algorithm is not invariant to the selection of the origin and scaling in the coordinate frame. to avoid ill-conditioned solutions, a normalization transform to the data need to be applied before using the *DLT*-algorithm.

The following normalization procedure is proposed in [13]. In a first step, the coordinates in each image are translated so as to bring the centroid of the cloud of all points to the coordinate origin. In a second step, the coordinates are scaled so that the average distance of all points to the origin is equal to $\sqrt{2}$. This can be

realized by applying the following transformation to the set of points in each image:

$$T = \begin{bmatrix} k & 0 & -kx_c \\ 0 & k & -ky_c \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.10}$$

where k is a scaling factor and (x_c, y_c) are the coordinates of the centroid of the cloud of points in each image. Applying the *DLT*-algorithm to the normalized data set gives an estimate of the homography $\tilde{H}$, relating the normalized point coordinates in one image to the other. The mapping between the originally measured coordinates in both images is given by $\hat{H} = T'^{-1}\tilde{H}T$, were T and T' are the normalization transformations corresponding to each image data set.

In Figure 2.2, the performance between the *DLT*-algorithm, the normalized *DLT*-algorithm and the inhomogeneous solution are compared for 5 point-to-point correspondences. The experiment simulates an identity transformation were points from a reference image are mapped to the same points in the target image. The real homography thus is given by the 3×3 identity matrix. To compare the various algorithms, 100 trials were made with each point being subject to 0.2 pixel Gaussian noise in the target image. Mapping an arbitrary image point from the reference image to the target image with the estimated homography is a way of visualizing the effects of error propagation. Ideally, this point is projected to the same coordinates in the target image. The 100 projections obtained from the estimated homographies are displayed in the figure for the various algorithms.

The experiment illustrates that the *DLT*-algorithm performs worst if not using normalization. The inhomogeneous solution performs as well as the normalized *DLT*-algorithm and therefore can be used as an alternative for those solutions that do not require h_9 to be close to zero.

(a) (b) (c)

Figure 2.2: Comparison between the performance of (a) the DLT-algorithm, (b) the normalized DLT-algorithm and (c) the inhomogeneous solution.

2.4 The perspective camera model

A camera performs a mapping between 3D-points in the world and a 2D image plane. The most widely used camera model is the pinhole model, illustrated in Figure 2.3.

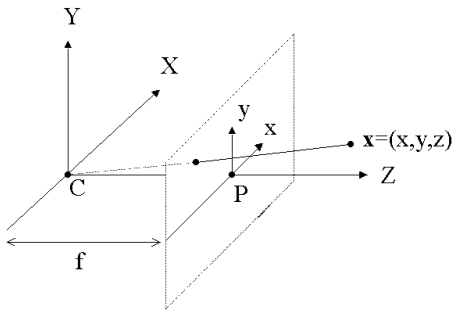


Figure 2.3: The pinhole camera model

The 3D point $\mathbf{X} = (X, Y, Z)$ is mapped to the 2D point $(fX/Z, fY/Z)$ on the image plane. This point is obtained from the intersection of the image plane with the ray passing through the 3D point and the camera center C, defining a non-linear mapping. In this model, the Euclidean coordinate system placed at the center of projection is the *the camera coordinate frame*, the plane $z = f$ is the *image plane* and f is the *camera focal distance*. The line containing the camera center, perpen-

dicular to the image plane is called the *principal axis* of the camera, which coincides with the z-axis of the camera coordinate frame. The intersection of the principal axis with the image plane is called the *principal point*.

Perspective projection with homogeneous coordinates. If the world and image coordinates are expressed as homogeneous vectors, the 3D to 2D mapping of points in the world to the image plane can be expressed as a linear mapping. Introducing the notation $\mathbf{X_{cam}} = (X, Y, Z, 1)$ as the homogeneous representation of world points expressed in the camera reference frame and $\mathbf{x} = (fX, fY, Z)$ as the homogeneous 3-vector of points in the image plane, the projection equations of the pinhole camera can be written as:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.11}$$

$$\mathbf{x} = P\mathbf{X_{cam}} \tag{2.12}$$

The 3×4 homogeneous matrix P is called the *camera projection matrix*.

Intrinsic parameters. In general, the image coordinate system is not placed at the principal point, (p_x, p_y) and we need to translate the coordinates of every image point to account for this effect. Also, image coordinates are often given in pixels so that the effect of unequal scaling factors in each direction need to be accounted for. Under these considerations, the mapping from 3D world coordinates to 2D pixel coordinates is given by the following projection matrix:

$$\mathbf{x} = K[I \mid \mathbf{0}]\mathbf{X_{cam}} \tag{2.13}$$

The matrix K is called the *camera calibration matrix*, given by:

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

were $\alpha_x = fm_x$, $\alpha_y = fm_y$, m_x is the number of pixels in the x-direction per unit distance, m_y the number of pixels in the y-direction per unit distance and $(x_0, y_0) = (m_x p_x, m_y p_y)$ are the coordinates of the principal point expressed in pixels. The parameters contained into K specify the internal camera structure and are called the camera *intrinsic* parameters.

Camera rotation and orientation. In general, points in space are not given in the camera reference frame but in some world reference frame. The two coordinate frames are related by a rotation and a translation. If $\tilde{\mathbf{X}}$ is an inhomogeneous vector representing the coordinates of a point in the world frame and $\tilde{\mathbf{X}}_{\mathbf{cam}}$ represents the same point in the camera frame, then:

$$\tilde{\mathbf{X}}_{\mathbf{cam}} = R(\tilde{\mathbf{X}} - \mathbf{t}), \tag{2.15}$$

where R is the 3×3 rotation matrix describing the orientation of the camera coordinate frame in the world frame and $\mathbf{t}$ is the position of the camera center in the world coordinate frame. Substituting $\mathbf{X}_{\mathbf{cam}}$ in equation (2.12) by $\mathbf{X}$ (which is the homogeneous representation of $\tilde{\mathbf{X}}$), the camera projection becomes:

$$\mathbf{x} = KR[I \mid -\mathbf{t}]\mathbf{X} \tag{2.16}$$

This equation describes the mapping between 3D points, expressed in an external world coordinate system and points in the camera image plane. The camera projection matrix is given by:

$$P = K[R| - R\mathbf{t}], \tag{2.17}$$

The parameters in $R, \mathbf{t}$ are called the camera *extrinsic* parameters, since they specify the camera external position and orientation in the world frame.

Normalized coordinate system. A coordinate system attached to the camera such that the projection of a point $M = (X, Y, Z)$ is the point $m = (x, y)$ determined

by the relations: $X/x = Y/y = Z$ is called the normalized coordinate system and is illustrated in Figure 2.4. Note that (x, y) are metric image coordinates which can be related to the real image coordinates in pixels by an affine transformation determined by the camera intrinsic parameters.

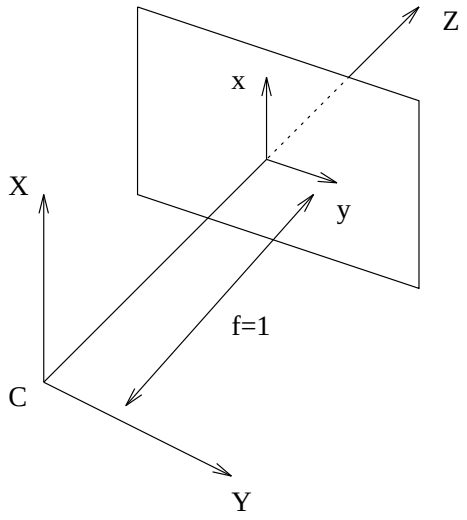


Figure 2.4: The normalized camera coordinate system

The normalized camera coordinate system allows to ignore the intrinsic parameters of the camera and to think in terms of ideal systems. This is especially useful for camera motion analysis. If the coordinates in the image plane are transformed according to:

$$\tilde{\mathbf{x}} = K^{-1}\mathbf{x}, \tag{2.18}$$

where $\tilde{\mathbf{x}}$ are the new transformed image coordinates, then the camera projection matrix for the normalized coordinate system turns into $P = [R| - R\mathbf{t}]$.

Chapter 3

Tracking of Image Regions

The problem addressed in this thesis consists in controlling a floating robot based on the visual information captured by an on-board camera. When the robot is at a short distance from the environment, vision is a powerful means to extract information about the vehicle's ego-motion and can thus be used for closed loop control of the robot. The motion of the robot (and camera) in the 3D space induces motion in the video stream. In general, these image deformations (transformations) are dependent on both the camera motion and scene structure. Hence, measuring these temporal image deformations can provide information of the camera position and orientation.

The tracking system described in this chapter aims at providing estimates of image transformations, over time. As the robot moves in 3D, the image formed by the on-board camera can be distorted according to a fairly general set of deformations. We assume that the scene can be approximated locally by a planar surface. As a consequence, the image motion can be described by a planar projective transformation.

Since visual cues are later used to control the robot in a visual feedback loop, the tracker design should take the computational requirements into account and avoid introducing large delays in the control loop due to image processing. We use a region-based tracking method, whereby a region selected in the reference image is

tracked over time and used as a visual landmark. No further assumptions are made with respect to the landmark, so that any textured region in the image plane can be a candidate, as long as it belongs to a plane in the 3D space. This provides a means of self-localizing the robot relative to any naturally textured scene and increases the applicability of the system, since it does not require deploying artificial items in the environment.

Some of the most reliable methods for region-based tracking apply numerical optimization to minimize the error between a reference region and a target region, subject to a parameterized deformation model. Often, the L_2 norm is used to measure the error, and the approach is referred to as the *sum-of-squared-difference* (*SSD*) method. A general derivation of the *SSD*-algorithm is described in [9] and uses optimization techniques in the specific context of image registration. The algorithm uses a set of motion (deformation) models applied to the reference region that sample the search space for expected image deformations. These deformations can be stored into a pre-calculated database and the main advantage is that most computations can be done off line. This database is also referred to as the *difference template* [9]. Motivated by the latter approach, we developed a tracking system based on full planar projective transformations, thus accommodating a wide set of possible image deformations. Additionally, the tracker adapts the difference template in accordance with the history of robot motions, while maintaining the computational loads minimal.

To enhance robustness of the tracking system, optical flow information is used to provide the method with an initial estimate. The advantages are two-fold: (i) on providing an initial estimate to the difference template matching, only small adjustments remain to be made. As only residual registration errors persist, we use a dense sampling of image motion models in a small neighborhood of the reference image, thus increasing the precision of the method. (ii), since the optic flow information

keeps track of average motion in the image plane, it provides a means for tracking when no other method is at hand (e.g. in those cases for which the landmark is out of sight).

3.1 Motion estimation using optic flow

In this section we describe an algorithm for tracking image regions based on optic flow information, assuming that the image of points move according to a planar motion model [28, 23]. Such a model can be approximated by an affine motion model, estimated from measurements of the spatiotemporal image derivatives [15]. Keeping track of this motion provides an initial estimate for the mapping of regions in between two successive images.

Estimation of affine motion in the image plane. If a camera is moving in 3D while observing a static environment, then the motion of points in the image plane induced by the camera rigid motion is given by the motion field (Appendix A):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -1-x^2 & y \\ 1+y^2 & xy & -x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.1}$$

In this equation, the vector $\mathbf{t} = (t_x, t_y, t_z)$ is the camera translational velocity, the vector $\omega = (\omega_x, \omega_y, \omega_z)$ is the camera rotational velocity and (u, v) denote the optical flow that we assume to be equal to the motion field.

We assume that the camera observes a plane in 3D-space, described by:

$$aX + bY + cZ + 1 = 0 \tag{3.2}$$

Using the equations of normalized perspective projection, we can rewrite the equation of the plane as a function of image coordinates:

$$\frac{1}{Z} = -ax - by - c \tag{3.3}$$

Under the planar assumption, the optic flow becomes constrained. Substituting equation (3.3) into equation (3.1) the optic flow for points belonging to the plane is given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} ax+by+c & 0 & -ax^2-bxy-cx \\ 0 & ax+by+c & -axy-by^2-cy \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -1-x^2 & y \\ 1+y^2 & xy & -x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3.4}$$

This equation can be parameterized as follows:

$$\begin{aligned} u &= u_0 + u_x x + u_y y + u_{xy} xy + u_{xx} x^2 \\ v &= v_0 + v_x x + v_y y + v_{xy} xy + v_{yy} y^2 \end{aligned}$$

This is the planar motion model of the optic flow, for a moving camera observing a plane. Since quadratic terms in general give rise to poor estimates in the presence of image noise, it is preferable to omit these terms, giving rise to the affine motion model [26]:

$$\begin{aligned} u &= u_0 + u_x x + u_y y \\ v &= v_0 + v_x x + v_y y \end{aligned} \tag{3.5}$$

This affine motion model is a good approximation of the planar motion model, whenever the viewing angle between the camera principal axis and the plane normal is not too large [23]. To estimate the parameters of affine motion model, consider the optical flow constraint equation [15] (Appendix A):

$$E_x u + E_y v + E_t = 0, \tag{3.6}$$

where E_x and E_y are the spatial derivatives in the image along the x- and y-axis and E_t is the temporal derivative. These quantities can be measured from successive images. Substituting the affine model given by equation (3.5) into equation (3.6) yields:

$$u_0 E_x + u_x E_x x + u_y E_x y + v_0 E_y + v_x E_y x + v_y E_y y = -E_t \tag{3.7}$$

Packing all the model parameters in a single vector, this equation can be written as a linear combination of the spatiotemporal derivatives and the parameter vector:

$$\begin{bmatrix} E_x & E_x x & E_x y & E_y & E_y x & E_y y \end{bmatrix} \theta = -E_t, \tag{3.8}$$

with $\theta = [u_0 \; u_x \; u_y \; v_0 \; v_x \; v_y]^T$. The parameter vector θ can be estimated with a minimum of six measurements of spatiotemporal derivatives. The overdetermined case can be resolved in a least-square sense. A robust estimate is obtained by including measurements at each image point, assuming that the plane viewed by the camera covers the whole image.

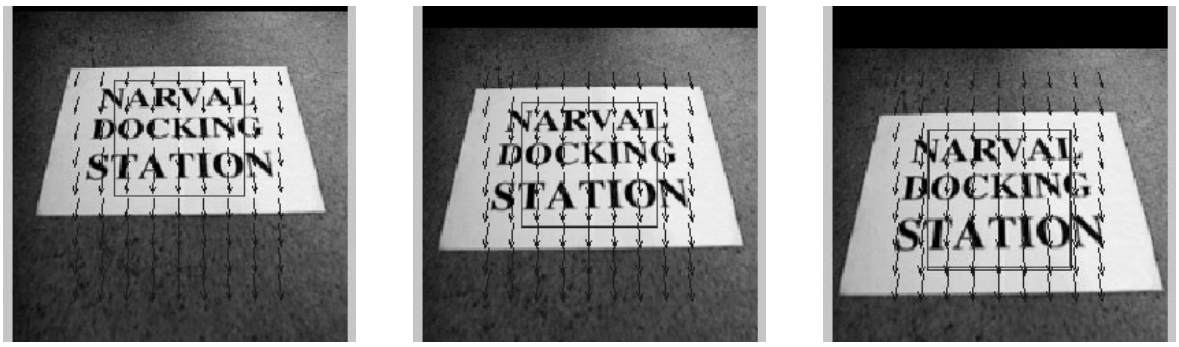
Tracking of image regions based on optic flow. Using the previous described method, it is possible to adjust an affine model from a sequence of images. For each new image, the affine flow provides an estimate of the velocity of the image points. Points in the previous image can be mapped to the current image by integrating this optic flow information. Figure 3.1 shows the affine flow field obtained from a sequence of translated images.

If the point (x_i, y_i) denotes the coordinates of a point in one image, then its new coordinates (x'_i, y'_i) in the subsequent image may be obtained by integrating the local component of the affine motion over time according to:

$$x'_i = x_i + u \tag{3.9}$$

$$y'_i = y_i + v \tag{3.10}$$

The components u and v of the affine flow are calculated at each image point, (x_i, y_i), from the estimated parameters of the affine model, as given in equation (3.5). It should be noted however that errors in the estimated components u and v



(frame 1) (frame 12) (frame 25)

Figure 3.1: Tracking an image patch based on affine flow measurements; the camera views a plane under an angle so that a perspective distortion is observable; for each frame in the sequence, the affine flow is sampled and super-imposed for the case of pure camera translation at constant speed.

of the affine flow are accumulated while integrating. This can be observed in Figure 3.1 from the difference between the real and estimated positions of the tracked window. The last frame shows an offset in estimated window corner coordinates. The corresponding $L2$-norm of the difference between the corner coordinates of the real and estimated window position over time is illustrated in Figure 3.2.

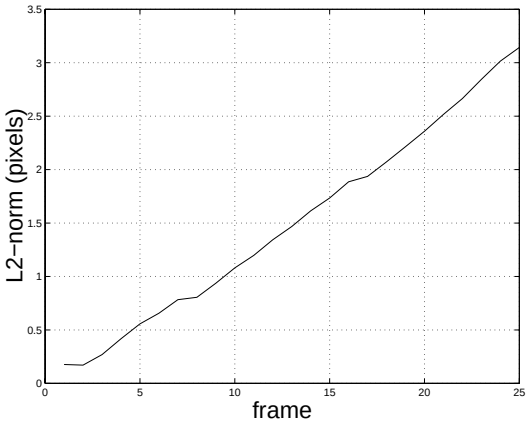


Figure 3.2: L2-norm from the difference between the real and estimated coordinates of the tracked window; the estimate is obtained from the affine flow measurements.

The integration of tracking errors over time calls for the need of an additional

method so as to reset these errors. This will be the topic of the next section.

3.2 Difference template matching

Given a reference image I_0 and a target image I_1, the registration problem is that of computing a transformation $(x', y') = \mathcal{T}(x, y)$, that maps points (x, y) in the reference image to points (x', y') in the target image. Usually, these transformations, are parameterized as a function of a vector $\mathbf{q}$: $\mathcal{T}(x, y) = \mathcal{T}_{\mathbf{q}}(x, y)$. This transformation is on image coordinates and therefore defines an image warping that maps pixel intensity values from I_1 back to I_0:

$$\mathcal{W}_{\mathbf{q}}(I_1) \mapsto I_0 : \quad I_0(x, y) = I_1\big(\mathcal{T}_{\mathbf{q}}(x, y)\big) \tag{3.11}$$

This is illustrated in Figure 3.3.

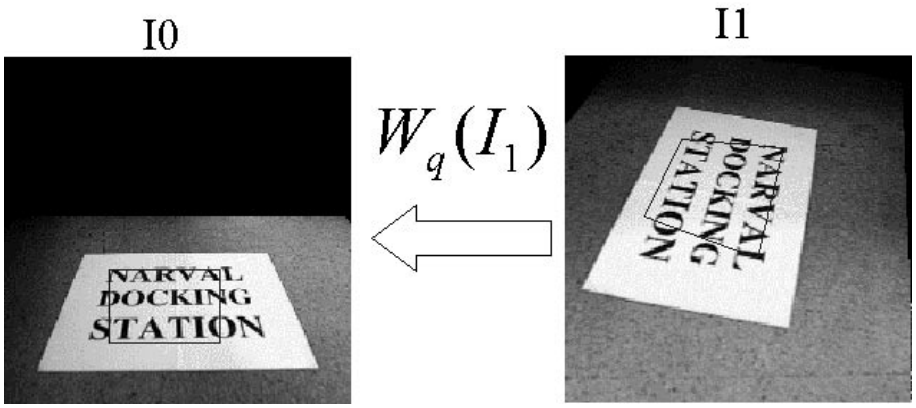


Figure 3.3: The problem of image registration is to find the transformation $\mathcal{W}$ that warps a selected region in the target image, I_1, onto a selected region of the reference image I_0. Image points that belong to a plane in 3D space are related by a 2D projective planar transformation that can be parameterized as a function of a vector $\mathbf{q}$.

We continue assuming that the target to track can be locally approximated as a planar surface. Planar motions cannot be adequately modeled by simple image

transforms, like affine or translational. A projective planar transformation is the exact motion model when a camera rotates about its eyepoint or if the image surface is planar. As discussed in Section 2.2, the 2D projective transformation, H, has eight degrees of freedom, $h_1, h_2, \cdots, h_9$. Considering the inhomogeneous solution and parameterizing the entries of H by a vector $\mathbf{q}$, image points of two images of a plane are related by:

$$x' = \frac{q_1 x + q_2 y + q_3}{q_7 x + q_8 y + 1}, \quad y' = \frac{q_4 x + q_5 y + q_6}{q_7 x + q_8 y + 1} \tag{3.12}$$

As suggested in [14], instead of representing the projective transformation by the eight coefficients of the homography, the positional offsets of the four corners of some image region is used to define the various motion models. This is illustrated in Figure 3.4, where four examples of motion sampling vectors $\Delta\mathbf{q_i}$ are given. The coordinates (x_i, y_i) of the four corners of the quadrangle define eight degrees of freedom and thus completely describe planar projective transforms. The corresponding homography can be easily obtained from point-to-point correspondences between the reference quadrangle and the deformed quadrangle.

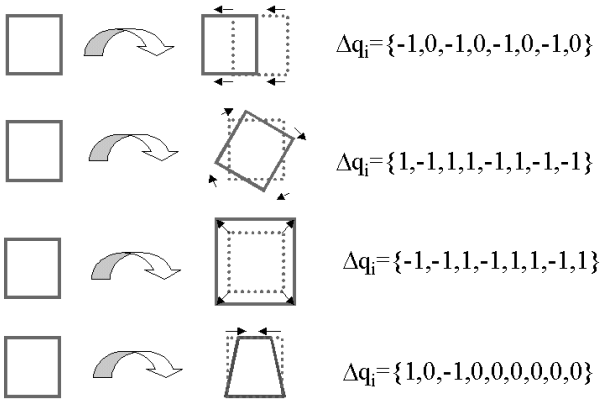


Figure 3.4: Planar projective transformations can be parameterized by the positional offset of the four corners of an quadrangle. From top to down: translation, rotation, scaling and perspective distortion.

To register two image regions, one needs to minimize a suitable error criterion,

such as the sum-of-squared-differences ($L2$-error criterion). If the real transformation is given by the parameter vector $\mathbf{q}$, then the minimization problem is defined as finding an estimate $\hat{\mathbf{q}}$ that minimizes the residue of the sum of squared differences between the pixel intensity values of the two image regions to be registered. Writing images as column vectors, the parameterized $L2$ error function is given by:

$$e(\mathbf{q}) = \frac{1}{2} \parallel \mathcal{W}_{\mathbf{q}}(I_1) - I_0 \parallel^2, \tag{3.13}$$

where the image warping $\mathcal{W}_{\mathbf{q}}(I_1)$ is specified by the homography parameterized with the vector $\mathbf{q}$, mapping I_1 to I_0 according to equation (3.11). This error function can be minimized with usual gradient descent methods. Discrete approximations to the partial derivatives of $I_1\big(\mathcal{T}_{\mathbf{q}}(x,y)\big)$ can be computed as:

$$\frac{I_1\big(\mathcal{T}_{\delta\mathbf{e}_i}(x,y)\big) - I_1(x,y)}{\delta}, \tag{3.14}$$

where $\mathbf{e}_i$ is the i'th basis vector over the parameter space of vector $\mathbf{q}$ and δ is an adequate value for discretization, depending on the shape of the error function. However, for an 8 dimensional space this method has a weak interpolation capability, i.e. it only searches the space effectively in the coordinate directions, and within a range determined by the parameter δ. Furthermore, minimizing equation (3.13) using an exhaustive search on the parameter space would be impractical.

At each time instant, we assume that an initial estimate $\mathbf{q}_0$ of the transformation parameters is available. In our case, this initial estimate is obtained with the affine motion estimate, discussed before. As a consequence, I_0 and I_1 can be approximately registered, such that only small errors remain. The transform parameters need to be adjusted by a vector $\Delta\mathbf{q}$ such that I_1 can be transformed into a better approximation of I_0. This defines a mapping of I_1 onto I_0 by a composition of two image warps: (i) a warping of I_1 to an image I_1' defined by the initial estimate $\mathbf{q}_0$ and subsequently

(ii) a warping of I_1' to I_0 by the adjustments to the transformation parameters, $\Delta\mathbf{q}$. Figure 3.5 visualizes this composition of image warps.

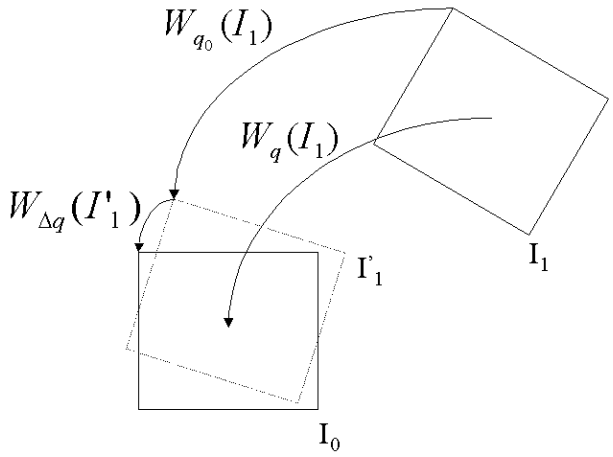


Figure 3.5: Mapping I_1 to I_0 by the composition of an initial estimate of the transformation parameters and the adjustments to these parameters.

From Figure 3.5 it follows that $\mathcal{W}_q(I_1) = \mathcal{W}_{\Delta\mathbf{q}}\big(\mathcal{W}_{\mathbf{q_0}}(I_1)\big)$ so that I_0 can be obtained from warping I_1 according to:

$$\mathcal{W}_{\Delta\mathbf{q}}\big(\mathcal{W}_{\mathbf{q_0}}(I_1)\big) \mapsto I_0 : \quad I_0(x,y) = I_1\big(\mathcal{T}_{\mathbf{q_0}}(\mathcal{T}_{\Delta\mathbf{q}}(x,y))\big) \tag{3.15}$$

Substituting this composition into equation (3.13), the new error criterion can be seen as a function of $\Delta\mathbf{q}$:

$$e(\Delta\mathbf{q}) = \frac{1}{2} \parallel \mathcal{W}_{\Delta\mathbf{q}}\big(\mathcal{W}_{\mathbf{q_0}}(I_1)\big) - I_0 \parallel^2, \tag{3.16}$$

which can be rewritten to obtain the following equivalent error function:

$$e(\Delta\mathbf{q}) = \frac{1}{2} \parallel \mathcal{W}_{\mathbf{q_0}}(I_1) - \mathcal{W}_{\Delta\mathbf{q}}^{-1}(I_0) \parallel^2 \tag{3.17}$$

The image obtained from warping I_1 according to $\mathcal{W}_{\mathbf{q_0}}$ is given by $I_1'(x,y) = I_1\big(\mathcal{T}_{\mathbf{q}_0}(x,y)\big)$ and is approximately registered with I_0 by the initial estimate, $\mathbf{q}_0$. The image obtained from warping I_0 with $\mathcal{W}_{\Delta\mathbf{q}}^{-1}$ is given by $\bar{I}_0(\mathbf{\Delta q}) = I_0\big(\mathcal{T}_{\Delta\mathbf{q}}^{-1}(x,y)\big)$

and accounts for small adjustments applied to I_0 in order to match I_0 with I_1' and further optimize the registration between I_0 and I_1.

To minimize this error function, we define a set of m motion vectors $\{\mathbf{\Delta q}_i : i \in (1 \dots m)\}$ to sample the parameter space, instead of performing exhaustive search. Each $\mathbf{\Delta q_i}$ transforms the reference image I_0 into an image $\bar{I}_0(\mathbf{\Delta q_i}) = I_0\big(\mathcal{T}^{-1}_{\mathbf{\Delta q_i}}(x, y)\big)$ that contains image deformations expected to be observed over time. With the set of motion vectors, the parameter vector $\mathbf{\Delta q}$ can be expressed as a linear combination of the various $\mathbf{\Delta q}_i$:

$$\mathbf{\Delta q} = \sum_{i=1}^{m} k_i \mathbf{\Delta q}_i \tag{3.18}$$

This decomposition is not unique since $\{\mathbf{\Delta q}_i : i \in (1 \dots m)\}$ is in general a set of redundant vectors. The choice of the motion vectors $\mathbf{\Delta q_i}$ is discussed later in this chapter. Now, the image space can be considered as a function of the parameter vector $\mathbf{k} = [k_1 \dots k_m]^T$. The new parameterization is given by:

$$\hat{I}_0(\mathbf{k}) = \bar{I}_0(\sum_{i=1}^{m} k_i \mathbf{\Delta q}_i) \tag{3.19}$$

For small deviations about $\mathbf{k} = \mathbf{0}$ we have the first order approximation:

$$\hat{I}_0(\mathbf{k}) \approx I_0 + \sum_{i=1}^{m} \frac{\partial \hat{I}_0}{\partial k_i} k_i, \tag{3.20}$$

where discrete approximations of each partial derivative can be expressed as:

$$\frac{\partial \hat{I}_0}{\partial k_i} = \bar{I}_0(\mathbf{\Delta q}_i) - I_0 = B_i \tag{3.21}$$

In [9], the set of vectors B_i are denoted *Difference Templates* and are also used for image registration, but they are justified in a different form. Substituting equation (3.21) into equation (3.20), the new parameterization is then given by:

$$\hat{I}_0(\mathbf{k}) \approx I_0 + B\mathbf{k}, \tag{3.22}$$

where B is the partial derivatives matrix: $B = [B_1 \dots B_m]$. Substituting the new parameterization given by equation (3.22) into the error function given by equation (3.17), this error criterion now becomes a function of the vector $\mathbf{k}$ and takes the form:

$$e(\mathbf{k}) = \frac{1}{2} \parallel D - B\mathbf{k} \parallel^2, \tag{3.23}$$

where D is the difference image $I_1' - I_0$. The least square solution for $\mathbf{k}$ can be determined by minimizing:

$$min_{\mathbf{k}} \parallel D - B\mathbf{k} \parallel^2 \quad \Rightarrow \quad \mathbf{k} = (B^T B)^{-1} B^T D \tag{3.24}$$

After determining $\mathbf{k}$, the solution for $\Delta\mathbf{q}$ can be calculated from equation (3.18). Using the updated estimates of the parameter vector $\Delta\mathbf{q}$, this process can proceed in subsequent iterations.

An example. Figure 3.6 (a) shows a reference image with a selected image region of interest (ROI). The image contained within this ROI, corresponds to I_0 in equation (3.16). The problem is to register the reference image with the current target image for each temporal iteration. Based on an initial estimate, it is possible to predict the location of the reference ROI in the target image up to some small errors that remain to be adjusted. Figure 3.6 (b) shows the current target image with the initial estimate of the ROI position (note that small errors remain).

From the initial estimate of the transform parameters, it is possible to warp the target image approximately back onto the reference image. The region bounded by the vertices of the reference ROI in this warped image is I_1', which can be obtained from $I_1'(x, y) = I_1\big(\mathcal{T}_{\mathbf{q_0}}(x, y)\big)$, $(x, y) \in I_0$.

The image regions I_0 and I_1' together with their difference $D = I_1' - I_0$ are illustrated in Figure 3.7. The two regions are approximately registered based on the initial motion estimate, but small adjustments remain to be made.

(a) (b)

Figure 3.6: (a) Reference image with an identified ROI; (b) current target image with the ROI position predicted from some initial estimate.

Figure 3.7: Image regions I_0, I_1' and the difference between them, $D = I_1' - I_0$

To register I_1' with I_0, the method proposes to "explain" the observed difference (D) by a linear combination of the elements in the pre-calculated difference template, as stated in equation (3.18). The difference templates are calculated from the set of motion (deformation) models applied to the reference region and sample the search space for expected image deformations. Figure 3.8 shows some typical image deformations (translation, rotation, scaling and perspective distortion) applied to the reference image I_0. The figure also shows the resulting difference templates B_i, as stated in equation (3.21).

The solution $\mathbf{k}$ given by equation (3.24) states that the observed image difference D is best approximated by the image obtained from the linear combination of the difference templates B_i, according to the elements of $\mathbf{k}$. The question of which difference templates are to be included in the set will be discussed in the following paragraph. Once obtained $\mathbf{k}$, the solution for $\Delta\mathbf{q}$ can be calculated according to equation (3.18).

With the adjusted estimate of the transform parameters, the updated coordinates (x_i, y_i) of the four corners of the ROI in the target image can now be calculated according to: $(x_i', y_i') = \mathcal{T}_{\mathbf{q_0}}\big(\mathcal{T}_{\Delta\mathbf{q}}(x_i, y_i)\big)$, where (x_i, y_i) is a corner coordinate in the reference image and (x_i', y_i') the corresponding corner coordinate in the target image. The projection of the updated ROI position in the target image is illustrated in Figure 3.9 together with the initial estimate.

Discussion and implementation of the method. The reason for rewriting equation (3.16) into the form of equation (3.17) is that it allowed to derive a new parameterization of the error criterion (equation (3.22)) that uses the difference templates, B_i. The main advantage to do so is that upon identifying the reference ROI in the reference image, the set of difference templates B_i can be computed *a priori*, that is off-line. To obtain the solution for $\mathbf{k}$ as in equation (3.24), it follows

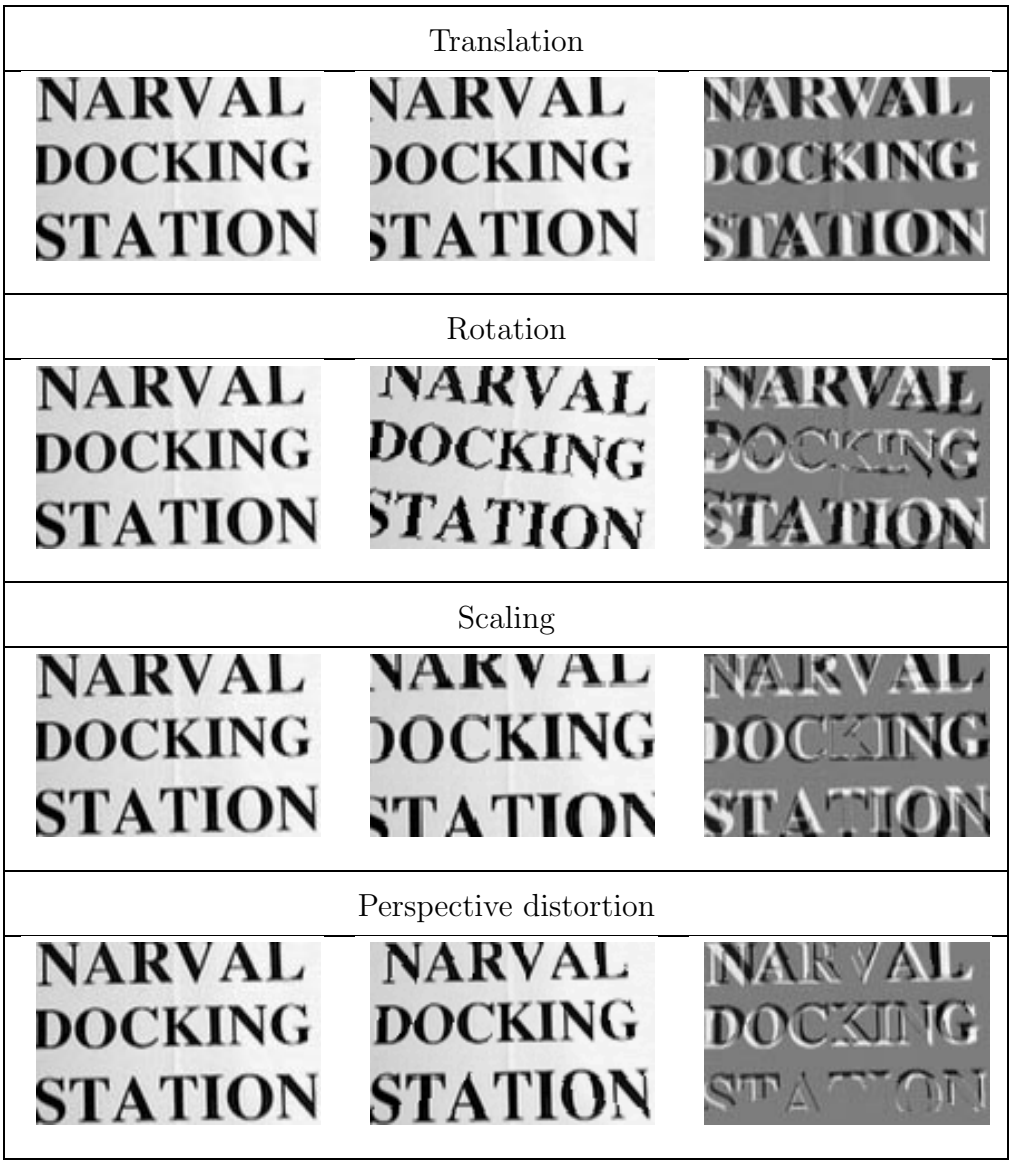


Figure 3.8: Reference ROI (I_0), expected image deformations ($\bar{I}_0(\mathbf{\Delta q_i})$) and their corresponding difference images (B_i).

Figure 3.9: ROI position in target image as predicted by the difference template method. Also the initial prediction was super-imposed.

that most computational requirements go out with the off-line computation of the matrix $(B^T B)^{-1}$. The only on-line computation of the algorithm implies simple matrix computations and an image warp $\mathcal{W}_{\mathbf{q_0}}(I_1) \mapsto I_1'$. This makes the method very well-suited for real time tracking applications.

Another advantage of this method in relation to standard local gradient descent methods is the ability to customize the set of motion models according to the kind and range of expected image deformations. Also, with the increase of computational power, it is easy to add new sample vectors to improve estimation results.

The choice of the sample vectors is of great importance for the performance of the algorithm. The following design choices need to be taken into account:

- the number of sample vectors;
- the directions of the sample vectors;
- the range of the sample vectors;
- the number of iterations.

As proposed in [1], the best performance is obtained by creating three sets of difference templates to cover translational, affine and planar projective motion mod-

els. This is motivated by the fact that large deformations may be estimated more robustly using a simpler motion model, which is refined in subsequent iterations. Starting with the most constrained set of vectors, the translational set will center the template with the current image, but further deformations need to be estimated. In a second phase, affine deformations will be adjusted, estimating most of the rotation, scaling and shear in the transformation. Finally, the planar projective set is used so to estimate remaining perspective distortions.

Since we are interested in real time tracking, computation time is important. In practice, a fixed computation time is desired, which has an upper bound given by the minimal required frequency of tracking (which is application dependent). Once such a frequency is defined, a trade-off has to be established between the number of sampling vectors and the number of iterations to do with the set. Using a higher number of sampling vectors, allows to estimate transformations over an wider range, but requires more online computation time. An increase in computation time results into a lower frame rate which implies larger inter-image deformations, imposing some trade-offs. The various sets of sampling vectors used in this work were established after many experiments and are shown in Appendix B. Each set includes 48 motion vectors that sample the parameter space uniformly in each direction, but over non-uniform increasing range. With this choice, the algorithm needs to iterate only once for the translational, affine and planar projective sampling sets.

Performance of the difference template tracker. Some experiments were conducted to characterize the performance of the method. Camera motions in 3D are simulated, such that they produce sudden increasing image deformations. We consider increasing translations, rotations and scaling in the image plane. The performance of the tracking algorithm is measured by comparing the L2 norm of the real and estimated positional offset of the quadrangle corner coordinates. In

Figure 3.10 these norms are plotted for increasing deformations.

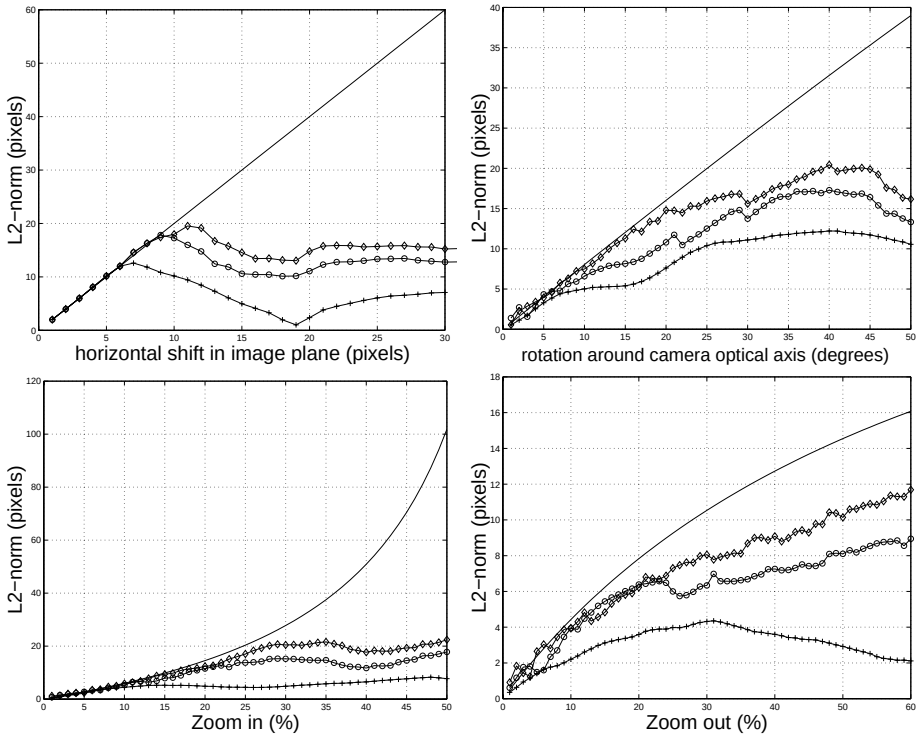


Figure 3.10: Performance of the algorithm for estimating translation. rotation, zoom in and zoom out in the image plane. The continuous line shows ground-truth values, $+$ - first iteration with translational set, o - second iteration with affine set, $\diamondsuit$ - last iteration with planar projective set.

The algorithm is applied by subsequently iterating once with the translational set, the affine set and the planar projective set of motion models. From Figure 3.10 it follows that in the case of translational deformations, the algorithm is able to accurately estimate translations up to an inter-image shift of 8 pixels.

3.3 Adaptive templates

The choice of the motion models that are included in the difference template method greatly determines the performance of the algorithm, as they sample the search space

for expected image deformations. Ideally, the choice of motion sampling vectors should be adapted to the camera motion.

Assuming that robot motion is smooth, then the induced image deformations over time are also smooth. Hence, recently estimated transformations point out into the **direction** and **range** of expected image deformations in near future. Including these past transformations into the set of motion sampling vectors, one expects an increase on performance for the algorithm. Introducing new sampling vectors into the existing partial derivatives matrix B (in equation (3.23)) requires the on-line computation of B^TB and $(B^TB)^{-1}$ for obtaining the solution given by equation (3.24), which would slow down the tracking frequency significantly.

In this section, we describe a method to substitute a motion sampling vector in an existing difference template database by taking advantage of the information already contained in the pre-calculated matrices B^TB and $(B^TB)^{-1}$, requiring a low computational effort. Then, an heuristic is used for deciding *which* sampling vector is to be replaced by *what* other sampling vector.

Substituting a sampling vector in the template database. Consider the structure of the partial derivatives matrix B, containing all difference templates B_i. Writing each B_i as a column vector, the partial derivatives matrix is given by:

$$B_{(l\times m)} = [B_1 \quad B_2 \quad \cdots \quad B_m], \tag{3.25}$$

where l is the number of pixels contained in the selected image region and m is the number of sampling vectors included in the set. The matrix B^TB has the form:

$$B^TB = \begin{bmatrix} B_1^T \\ \vdots \\ B_m^T \end{bmatrix} \begin{bmatrix} B_1 & \cdots & B_m \end{bmatrix} = \begin{bmatrix} B_1^TB_1 & \cdots & B_1^TB_m \\ \vdots & \ddots & \vdots \\ B_m^TB_1 & \cdots & B_m^TB_m \end{bmatrix}_{(m\times m)} \tag{3.26}$$

To calculate B^TB it is thus necessary to do $m\times m$ multiplications of two difference images. If the size of each difference image is $(p\times r)$, the calculation of B^TB implies

$m \times m \times p \times r$ multiplications, requiring a great deal of computational power.

Upon substitution of a sampling vector, a difference template B_i needs to be substituted by a new difference template B'_i, where $i \in \{1, 2, \ldots, m\}$. The new partial derivatives matrix B' differs only from B by its i^{th} entry:

$$B'_{(1\times m)} = [B_1 \cdots B'_i \cdots B_m], \tag{3.27}$$

The corresponding new matrix $B'^T B'$ has the form:

$$B'^T B' = \begin{bmatrix} B_1^T \\ \vdots \\ B_i'^T \\ \vdots \\ B_m^T \end{bmatrix} \begin{bmatrix} B_1 & \cdots & B'_i & \cdots & B_m \end{bmatrix} = \begin{bmatrix} B_1^T B_1 & \cdots & B_1^T B'_i & \cdots & B_1^T B_m \\ \vdots & \ddots & \vdots & & \vdots \\ B_i'^T B_1 & \cdots & B_i'^T B'_i & \cdots & B_i'^T B_m \\ \vdots & & \vdots & \ddots & \vdots \\ B_m^T B_1 & \cdots & B_m^T B'_i & \cdots & B_m^T B_m \end{bmatrix}_{(m\times m)} \tag{3.28}$$

The problem is how to calculate $(B'^T B')^{-1}$ by taking advantage of the information stored in the already pre-calculated matrix $(B^T B)^{-1}$. This is done in two steps. First we show how to obtain $B'^T B'$ from $B^T B$ and in a second step, their inverse matrices will be related.

Step 1:

Since the entries of $B^T B$ and $B'^T B'$ only differ on the i^{th} column and row, the new $B'^T B'$ can be obtained from the pre-calculated $B^T B$ simply by updating the $2 \times m$ new entries for the i'th row and column, requiring only $p \times r \times 2 \times m$ multiplications.

Without loss of generality, in the remaining part of this section we assume that i is at the last row and column, which can always be achieved with permutation matrices. In this case both $B^T B$ and $B'^T B'$ contain the same information at the entries of their upper $(m-1) \times (m-1)$ block. Writing:

$$B^T B = \begin{bmatrix} E_{(m-1)\times(m-1)} & F_{(m-1)\times 1} \\ G_{1\times(m-1)} & H_{1\times 1} \end{bmatrix} \tag{3.29}$$

it follows that $B'^T B'$ can be written as:

$$B'^T B' = \begin{bmatrix} E_{(m-1)\times(m-1)} & F'_{(m-1)\times 1} \\ G'_{1\times(m-1)} & H'_{1\times 1} \end{bmatrix} \tag{3.30}$$

This representation allows to relate the inverse matrices of $B^T B$ and $B'^T B'$ in a convenient way, as will be seen in the next step.

Step 2:

Now the goal is to obtain $(B'^T B')^{-1}$ at a low computational cost from the already pre-calculated matrix $(B^T B)^{-1}$. Writing $B^T B$ as a block matrix and supposing that the inverse of its upper-left block exists, it follows [16] that the inverse of $B^T B$ can be written as:

$$(B^T B)^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}^{-1} = \begin{bmatrix} E^{-1} + E^{-1}FS^{-1}GE^{-1} & -E^{-1}FS^{-1} \\ -S^{-1}GE^{-1} & S^{-1} \end{bmatrix}, \tag{3.31}$$

where $S = H - GE^{-1}F$ is a scalar. This inverse is also a block matrix for which the upper left block, $E^{-1} + E^{-1}FS^{-1}GE^{-1}$ has size $(m-1 \times m-1)$. Note the fact that, to obtain the inverse $(B^T B)^{-1}$, one only needs to calculate the inverse of matrix E and scalar S. Now, since matrices $B^T B$ and $B'^T B'$ share the same matrix block E, the inverse of $B'^T B'$ can be obtained at a cheap computational cost from the inverse of $B^T B$ by taking advantage of the shared E^{-1}. The matrix block E^{-1} can be recovered from $(B^T B)^{-1}$ by writing:

$$(B^T B)^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}^{-1} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & m_{22} \end{bmatrix} \tag{3.32}$$

With this representation, E^{-1} is given by:

$$E^{-1} = M_{11} - M_{12}.m_{22}^{-1}.M_{21} \tag{3.33}$$

Once extracted E^{-1}, it can be used to calculate the inverse of $(B'^T B')'^{-1}$ according

to:

$$(B'^{T}B')^{-1} = \begin{bmatrix} E^{-1} + E^{-1}F'S'^{-1}G'E^{-1} & -E^{-1}F'S'^{-1} \\ -S'^{-1}G'E^{-1} & S'^{-1} \end{bmatrix}, \tag{3.34}$$

with $S' = H' - G'E^{-1}F'$. For each template substitute, the updated pseudo inverse of the new partial derivatives matrix B' is obtained by this equation. Upon periodically substituting one sampling vector in the database, all pre-calculated information can be updated at a relative low computational effort.

Substitute heuristic. Upon periodically substituting a difference image in the partial derivatives matrix, still the problem remains of deciding which sampling vector is to be replaced by what new sampling vector. Assuming smooth motion, the best new motion sampling vector is given by the last detected motions in the image plane, which is obtained from the current estimate $\Delta\mathbf{q}$.

The decision of which sampling vector to remove is not trivial, since it can destroy the ability of the database to sample deformations in all directions. Such a situation occurs when image deformations occur in a single direction over a relative long time span, e.g. during translation. Upon periodically substituting in the entire database, the database will end up sampling only in the direction of translation. For this reason, it is decided to maintain the original sampling set, $\{\mathbf{\Delta q}_i : i \in (1 \dots 48)\}$, fixed and have a smaller complementary set in which expected image deformations are substituted based on the last estimated value of the transformation.

To accommodate the substitute sampling vectors, an additional set of 8 sampling vectors is added to the existing translational, affine and planar projective sampling sets (each of which includes 48 motion vectors). This way one expects an increase in performance of the difference template registration method while at the same time a redundant base is preserved, sampling all possible image deformations.

A simple heuristic is used to decide which sampling vector in the complementary sets is to be substituted. Based upon the values of the last 8 elements in vector

k (equation (3.24)), the sample vector with the lowest weight k_i, $i \in \{49 \cdots 56\}$ is substituted.

Performance with adaptive sampling vectors. Doing the same experiment as illustrated in Figure 3.10, the performance with substitution of motion vectors is given in Figure 3.11.

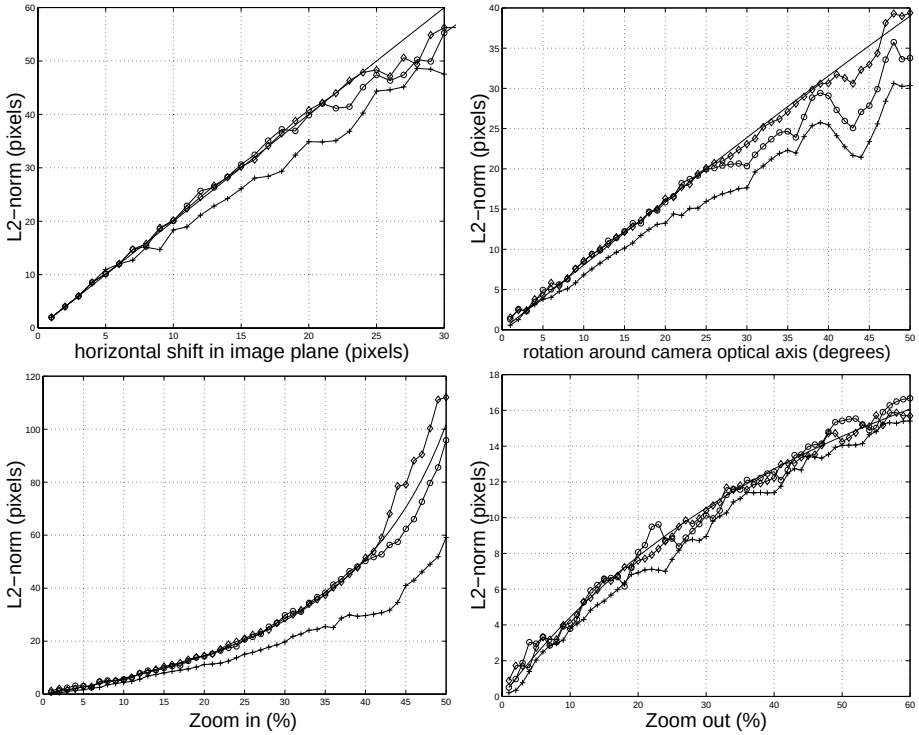


Figure 3.11: Performance of the adaptive algorithm for estimating translation, rotation, zoom in and zoom out in the image plane. The continuous line shows ground-truth values, + - first iteration with translational set, *o* - second iteration with affine set, $\diamondsuit$ - last iteration with planar projective set.

During the experiment, the motion vectors are substituted iteratively, based on the previous estimate. Since past values are integrated, this experiment shows the ability of the algorithm to track the visual landmark when undergoing an acceleration in the image plane, rather then showing the absolute range of detectable deformations, as was the case in Figure 3.10. However, comparing Figures 3.11 and

3.10 shows that this version of the algorithm is able to estimate the various transformations in the image plane over a much wider range in the presence of acceleration. We therefore conclude that the adaptive component contributes to the robustness of the tracking system.

3.4 Outline of the tracking algorithm

The implementation of the tracker algorithm is resumed in the following tables.

Initialization/Off-line Operations
1. Identify the textured image region to be tracked, I_0. 2. Construct the translational, affine and planar projective sampling sets: - translational set: $\{\mathbf{\Delta q}_i : i \in (1 \dots 48)\}_1$ - affine set: $\{\mathbf{\Delta q}_i : i \in (1 \dots 48)\}_2$ - planar projective set: $\{\mathbf{\Delta q}_i : i \in (1 \dots 48)\}_3$ 3. Compute the corresponding partial derivatives matrices and pseudo inverses: - translational set: $\{B, B^T B, (B^T B)^{-1}\}_1$ - affine set: $\{B, B^T B, (B^T B)^{-1}\}_2$ - planar projective set: $\{B, B^T B, (B^T B)^{-1}\}_3$

Run-time/On-line Operations.

1. Compute the optic flow in the image: (u, v)
and the corresponding transform estimate: $\mathcal{T}_{(u,v)}(x, y)$

2. Compute the current initial estimate from the composition of the optic flow estimate and the previous transform estimate, $\mathcal{T}_{\hat{\mathbf{q}}_{t-1}}(x, y)$:

$$\mathcal{T}_{\mathbf{q}_0}(x, y) = \mathcal{T}_{\hat{\mathbf{q}}_{t-1}}\big(\mathcal{T}_{(u,v)}(x, y)\big)$$

3. Refine transform estimate with the translational motion sampling set

- compute $I_1^{'} = I_1(\mathcal{T}_{\mathbf{q}_0}(x, y))$ using the current initial estimate
- compute the difference image, $D = I_1^{'} - I_0$
- estimate vector $\mathbf{k_1}$ and parameter update $\Delta\mathbf{q}_1$
- update the current transform estimate: $\mathcal{T}_{\hat{\mathbf{q}}}(x, y) = \mathcal{T}_{\mathbf{q}_0}\big(\mathcal{T}_{\Delta\mathbf{q}_1}(x, y)\big)$.

4. Refine transform estimate with the affine motion sampling set

- use $\hat{\mathbf{q}}$ as the initial estimate: $\mathcal{T}_{\mathbf{q}_0} = \mathcal{T}_{\hat{\mathbf{q}}}$
- compute $I_1^{'} = I_1(\mathcal{T}_{\mathbf{q}_0}(x, y))$ using the current initial estimate
- compute the difference image, $D = I_1^{'} - I_0$
- estimate vector $\mathbf{k}_2$ and parameter update $\Delta\mathbf{q}_2$
- update the current transform estimate: $\mathcal{T}_{\hat{\mathbf{q}}}(x, y) = \mathcal{T}_{\mathbf{q}_0}\big(\mathcal{T}_{\Delta\mathbf{q}_2}(x, y)\big)$.

5. Refine transform estimate with the perspective motion sampling set

- use $\hat{\mathbf{q}}$ as the initial estimate: $\mathcal{T}_{\mathbf{q}_0} = \mathcal{T}_{\hat{\mathbf{q}}}$
- compute $I_1^{'} = I_1(\mathcal{T}_{\mathbf{q}_0}(x, y))$ using the current initial estimate
- compute the difference image, $D = I_1^{'} - I_0$
- estimate vector $\mathbf{k}_3$ and parameter update $\Delta\mathbf{q}_3$
- update the current transform estimate: $\mathcal{T}_{\hat{\mathbf{q}}}(x, y) = \mathcal{T}_{\mathbf{q}_0}\big(\mathcal{T}_{\Delta\mathbf{q}_3}(x, y)\big)$.

6. Substitute the composition $\Delta\mathbf{q} = \Delta\mathbf{q}_1 \circ \Delta\mathbf{q}_2 \circ \Delta\mathbf{q}_3$ in the translational, affine and projective motion sampling sets

With this algorithm, we were able to successfully track an image patch undergoing planar projective transformation. Figure 3.12 shows results of tracking an identified region over time.

Figure 3.12: Tracking an image patch over time. This test was conducted with the aerial blimp.

Chapter 4

Robot Modeling

The work presented in this thesis is integrated in the NARVAL project, for which one of the main goals is the design and implementation of reliable navigation systems of limited cost for mobile robots in unstructured environments. For demonstration, two experimental setups were used: (i) an airborne blimp and (ii) a remotely operated underwater vehicle (ROV).

In this chapter, these robots are specified and modeled. Both robots have the same degrees of freedom, since they both float in 3D space. The design of the blimp is such that the geometric arrangement of the thrusters defines the same controllable degrees of freedom as in the ROV. For these reasons, the blimp kinematic and dynamic relations are a reasonable approximation of those of the submarine, having different time constants. Therefore, this chapter only introduces the blimp dynamic model which also applies to the ROV. The blimp can be used as a test-bed for experiments in a laboratory environment. The only major difference is the fact that since water is more viscous than air, the control will be simpler in that case, when compared to the blimp. On the other hand, visual processing is expected to be more difficult for the underwater case, requiring high-performance and robust visual processing.

4.1 Blimp and ROV description

The small-size indoor blimp used is illustrated in Figure 4.1. It is composed of a balloon, a gondola and a remote controller. For the blimp to float in air, its envelope needs to be filled with a gas that is lighter then air, typically Helium. The gondola is a rigid structure attached to the bottom of the balloon. It contains the motor controllers, a radio receiver and the three thrusters for propulsion in the horizontal and vertical planes. Additionally, a mini camera with a video link was mounted on the gondola.

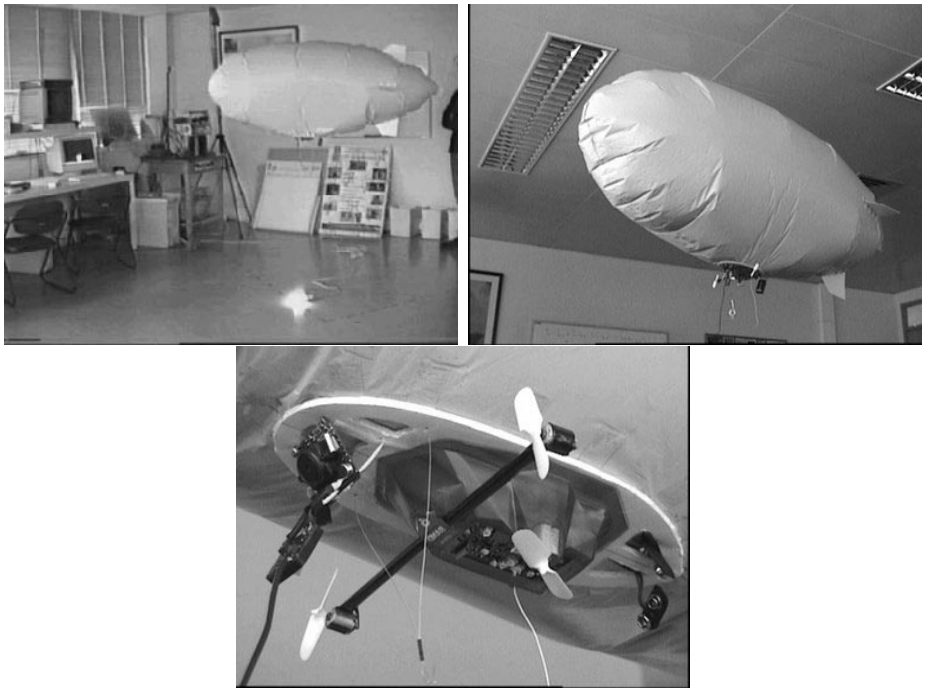


Figure 4.1: Radio controlled indoor blimp with on-board camera.

The *CCD* camera sends video signals to a remote computer via a video link in open air. Figure 4.2 shows a typical image of a scene as viewed by the blimp camera. The images received by the computer are processed and analyzed so as to derive proper control signals, sent to the blimp via a radio link.

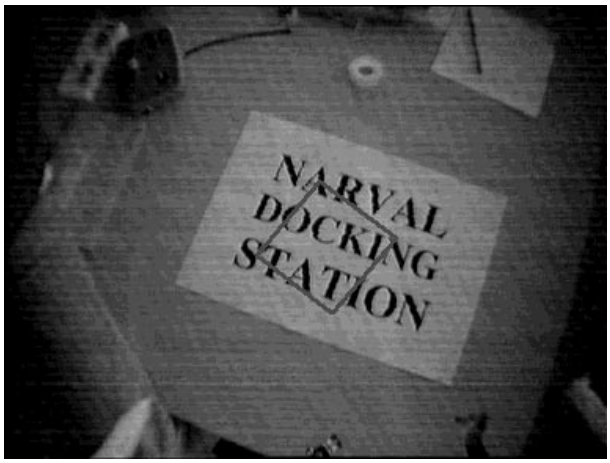


Figure 4.2: Image as seen from the blimp on-board camera while looking down to the ground-plane.

The remote operated underwater robot is illustrated in Figure 4.3. The underwater robot is equipped, among other sensors, with an on-board pan and tilt camera. The camera is mounted rigidly to the ROV but its pan and tilt angles can be controlled separately, resulting in two extra degrees of freedom for the camera. The ROV is wired to a remote processing unit by an umbilical. Video signals go up to the ground surface were they are processed in a remote computer. Figure 4.4 shows a typical image of an underwater scene as viewed from the robot camera. Control signals for the ROV thrusters are sent down via the umbilical to the ROV motors.

4.2 Blimp dynamic model

In order to derive a mathematical model, it is useful to first review some physical principles of airship operation [10, 17]. First of all, an important characteristic is the aerostatic lift, which, unlike the lift forces generated over a wing surface, is independent of flight speed. The aerostatic lift force comes from *Archimedes' Principle* and is equal to the mass of the volume of air displaced by the airship's envelope. The aerostatic lift is also known as the buoyant force. An upward lift is

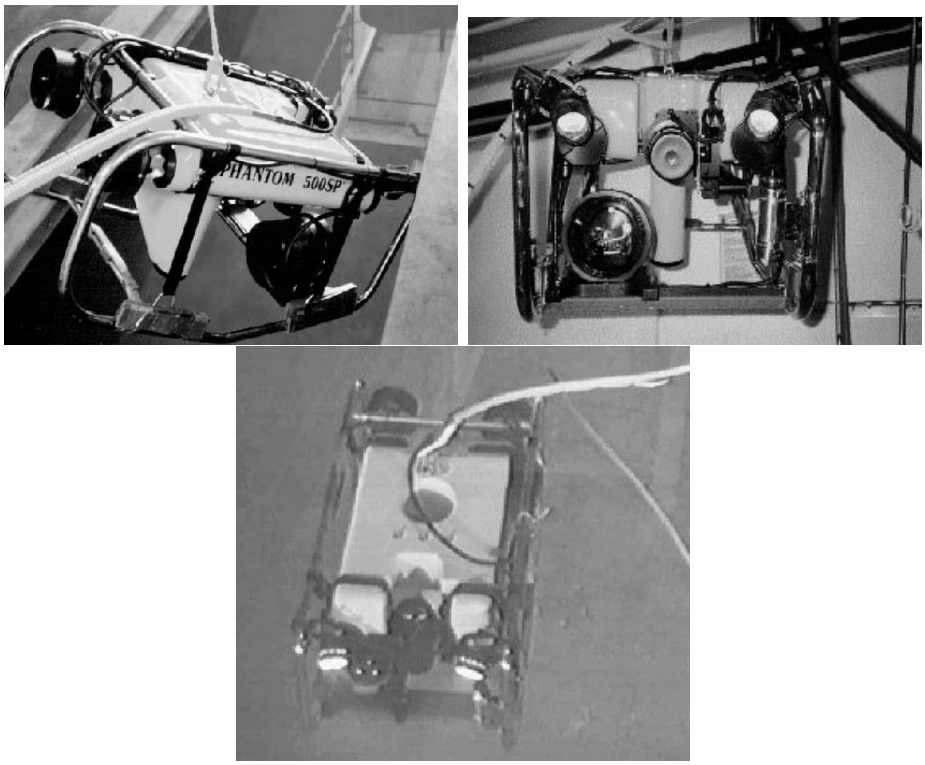


Figure 4.3: Remote operated underwater robot with on-board pan- and tilt-camera.

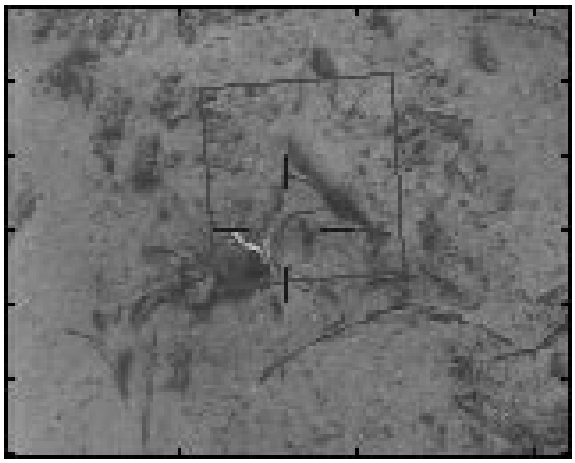


Figure 4.4: Typical image of an underwater scene. The ROV on-board camera is looking downwards to the sea bottom.

obtained when the airship's envelope contains a gas with a density lower then air. Helium is the most commonly used lifting gas.

Second, a buoyant body in motion in a fluid displaces each fluid particle in the direction of motion of the body. The fluid therefore gains kinetic energy and the body experiences a resistance to its motion. This effect can be taken into account by considering added mass and inertia terms. As the blimp displaces a relative large volume of air during flight, these properties become significant and the body behaves as if it had a mass and moments of inertia substantially higher than those indicated by conventional physical modelling.

Finally, because the airship center-of-mass is difficult to locate and time-variant during flight, motion has to be referenced to by a system of orthogonal body axes, $\mathcal{F}_M$, placed at the geometric center of the envelope volume, as shown in Figure 4.5. The centre of volume can be assumed to coincide with the center of buoyancy.

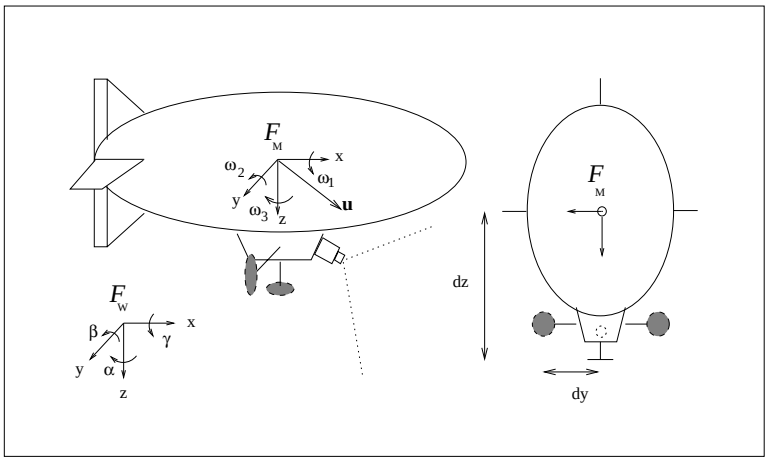


Figure 4.5: Definition of reference frames.

Considering the vehicle as a rigid body (not taking into account its elasticity) the dynamic model can be obtained by writing down the six degrees of freedom *Newton-Euler* equations of motion resolved into the body-fixed reference frame [8, 10]:

$$M\dot{\mathbf{v}} + \mathbf{c}(\mathbf{v}) + D(\mathbf{v})\mathbf{v} + \mathbf{g}(\alpha, \beta, \gamma) = \tau, \tag{4.1}$$

The various elements in equation (4.1) will be explained in more detail in the following.

The velocity vector. The vector $\mathbf{v} = [\mathbf{u}\ \mathbf{w}]^T$, is the 6×1 velocity vector containing the three components of linear velocity $\mathbf{u} = [u_1\ u_2\ u_3]^T$ and the three components of angular velocity $\mathbf{w} = [\omega_1\ \omega_2\ \omega_3]^T$ measured in the earth-fixed inertial reference frame $\mathcal{F}_W$ and expressed in the body-fixed reference frame $\mathcal{F}_M$.

The mass matrix. The 6×6 mass matrix $M \equiv M_{RB} + M_A$ contains all masses and inertias of the rigid body (M_{RB}) and the added mass and inertia terms (M_A). Considering the gondola as a point mass, aligned along the z-axis of the body-fixed reference frame, the planes $(x - z)$ and $(y - z)$ in the body-fixed reference frame form symmetry planes for the mass distribution. This implies various simplifications for the mass matrix. In the first place, all cross-coupling inertial terms become zero and the inertia matrix becomes diagonal. In the second place, one can neglect the contribution from off-diagonal elements in the added mass matrix M_A, thus suggesting the following model: $M_A = diag\{A_{11}, A_{22}, A_{33}, A_{44}, A_{55}, A_{66}\}$. The elements $A_{ii},\ i \in (1, 2, .., 6)$ can be estimated from the dimensions of the airship's hull, as described in [8].

Coriolis and centrifugal terms. The 6×1 vector $\mathbf{c}(\mathbf{v}) = \mathbf{c}_{RB}(\mathbf{v}) + \mathbf{c}_A(\mathbf{v})$ groups all quadratic terms, containing the centrifugal and Coriolis dynamic forces.

Damping matrix. The matrix $D(\mathbf{v}) \equiv D_S(\mathbf{v}) + D_H(\mathbf{v})$, is the 6×6 aerodynamic damping matrix. Linear and quadratic skin friction drag are modeled with $D_S(\mathbf{v})$

and arise due to laminar and turbulent boundary layers. Damping due to vortex shedding, which depends on the streamline of the airship's hull, can be modeled with $D_H(\mathbf{v})$. For small indoor blimps moving at low speed, laminar boundary layer conditions can be assumed, considering only linear skin friction coefficients. For this case, the following model is proposed: $D = diag\{X_{u_1}, Y_{u_2}, Z_{u_3}, K_{\omega_1}, M_{\omega_2}, N_{\omega_3}\}$. The elements of D can be estimated either from wind-tunnel testing or system identification tools.

Gravity and buoyancy vector. The vector $\mathbf{g}(\alpha, \beta, \gamma)$, is the 6×1 restoring forces vector containing the gravity and buoyancy forces. Since these forces are defined in the earth-fixed inertial frame, they need to be resolved into body-fixed axes by a rotation matrix. The standard fixed $x - y - z$ angles representation is used to parameterize the rotation matrix with the $roll(\gamma)$, $pitch(\beta)$ and $yaw(\alpha)$ angles.

Thruster vector. The vector $\tau = \tau_{\mathbf{A}} + \tau_{\mathbf{P}}$, is the 6×1 applied forces vector containing all forces and moments acting on the rigid body due to the aerodynamic control surfaces ($\tau_{\mathbf{A}}$) and the propulsive units ($\tau_{\mathbf{P}}$). For an indoor blimp moving at low speed under laminar boundary layer conditions, aerodynamic control surfaces will have no effect and thus are not included. In this case, the vector τ will be a function of the geometrical arrangement of the propulsive units around the body axes: $\tau = [T_{cmn}\ 0\ T_v\ 0\ (d_z T_{cmn})\ (d_y T_{diff})]^T$, where $T_{cmn} = (T_s + T_p)$ is the common mode component of thrust coming from the starboard and port side propellers, T_v is the thrust resulting from the vertical propeller, $T_{diff} = T_s - T_p$ is the difference between starboard and port side propeller thrust, d_y is the horizontal offset from the center of volume of the horizontal propellers and d_z is the vertical offset of the vertical propeller (see Figure 4.5).

4.3 Blimp kinematics

For control and navigation purposes, the velocity vector in equation (4.1) must be transformed to the earth-fixed inertial frame, $\mathcal{F}_W$, so as to obtain the vehicle position and orientation over time in the fixed world reference frame. This leads to the kinematic relations. The relative position and orientation of $\mathcal{F}_M$ with respect to $\mathcal{F}_W$ will be denoted by the vector $\eta = [x\ y\ z\ \alpha\ \beta\ \gamma]^T$, where a standard $roll(\gamma), pitch(\beta), yaw(\alpha)$ fixed angles representation is assumed for the orientation. The rotation matrix describing the orientation of $\mathcal{F}_M$ in $\mathcal{F}_W$ is then given by:

$$ {}^W_M R = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}, \tag{4.2} $$

where $c(.) = \cos(.), s(.) = \sin(.)$. The kinematics relation can then be written as:

$$ \dot{\eta} = \begin{bmatrix} J_1(\alpha,\beta,\gamma) & 0_{3\times3} \\ 0_{3\times3} & J_2(\alpha,\beta,\gamma) \end{bmatrix} \mathbf{v}, \tag{4.3} $$

where $J_1 = {}^W_M R$ is the rotation matrix and J_2 a Jacobian matrix relating the angular velocity vector to the time derivatives of the attitude parameters. The Jacobian J_2 can be easily calculated from the rotation matrix (see [3]). For the fixed angles representation it is given by:

$$ J_2(\alpha,\beta,\gamma) = \begin{bmatrix} 1 & s\gamma t\beta & c\gamma t\beta \\ 0 & c\gamma & -s\gamma \\ 0 & \frac{s\gamma}{c\beta} & \frac{c\gamma}{c\beta} \end{bmatrix}, \tag{4.4} $$

where $t(.) = \tan(.)$. Integration over time of equation (4.3) gives the vehicles position and orientation over time in $\mathcal{F}_W$.

In the design of control laws, it is sometimes useful to distinguish the vehicles controllable degrees of freedom from its velocity screw:

$$ \mathbf{T}_M = \begin{bmatrix} u_1 & u_3 & \omega_3 \end{bmatrix}^T \tag{4.5} $$

where $\mathbf{T}_M$ is the vehicle control input vector containing the three controllable degrees of freedom, resulting from the geometric arrangement of the thrusters. This vector specifies the non-holonomic constraints of the vehicle.

4.4 Camera kinematics

In this section we consider a pan and tilt camera to be rigidly attached to the vehicle. Vision will be used to control the robots so that measurements will be done in the camera reference frame. The desired trajectories of the camera reference frame play an important role in the design of control laws. It is therefore useful to derive the camera controllable degrees of freedom as a function of the vehicle control inputs and the two extra degrees of freedom introduced by the pan and tilt unit.

To do so, we first derive a model for a pan and tilt camera, relating the pan and tilt velocities to the resulting velocity of the camera frame. Then we consider the case of zero pan and tilt angles and velocities, so as to obtain the camera controllable degrees of freedom resulting from the vehicle control input vector. Finally, the overall camera kinematic relations are obtained from the composition of the two separate cases.

Pan and tilt camera. The schematics of a pan and tilt camera is depicted in Figure 4.6. The model includes the camera reference frame $\mathcal{F}_C$, in which the tilt angle θ_{tilt} is defined and also a base frame $\mathcal{F}_B$, in which the pan angle θ_{pan} is defined. The camera frame is rigidly attached to the base frame, in such a way that its only degree of freedom relative to the base frame, is the tilt angle. The only degree of freedom considered for the base frame is the pan angle.

The orientation of $\mathcal{F}_C$ in $\mathcal{F}_B$ depends only on the camera tilt angle and is therefore given by:

$$ {}^B_C R = \begin{bmatrix} \cos(\theta_{tilt}) & 0 & \sin(\theta_{tilt}) \\ 0 & 1 & 0 \\ -\sin(\theta_{tilt}) & 0 & \cos(\theta_{tilt}) \end{bmatrix} \tag{4.6} $$

The position of the origin of the camera frame expressed in the base frame is given by the constant vector ${}^B\mathbf{P}_C = [a\ b\ c]^T$. Here we consider $b = c = 0$. Defining $\mathbf{v}_C = [\mathbf{u}_C\ \mathbf{w}_C]^T$ as the 6×1 camera velocity screw, then the components of this

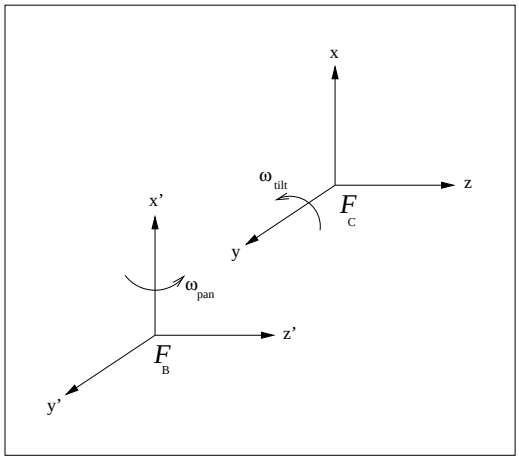


Figure 4.6: Schematics of a pan and tilt camera; $\mathcal{F}_B$ base frame in which the pan angle is defined; $\mathcal{F}_C$ camera reference frame in which the tilt angle is defined.

screw due to the pan and tilt velocities, ω_{pan} and ω_{tilt} can be resolved into the camera reference frame from superposition:

$$\mathbf{w}_C = [0\ \omega_{tilt}\ 0]^T + {}^C_B R\ \mathbf{w}_B \tag{4.7}$$

$$\mathbf{u}_C = {}^C_B R(\mathbf{u}_B + \mathbf{w}_B \times {}^B\mathbf{P}_C), \tag{4.8}$$

where $\mathbf{w}_B = [\omega_{pan}\ 0\ 0]^T$ and $\mathbf{u}_B = [0\ 0\ 0]^T$ define the base frame velocity screw. These equations can be written into a more compact form as:

$$\mathbf{v}_C = J_3(\theta_{tilt}) \begin{bmatrix} \omega_{pan} \\ \omega_{tilt} \end{bmatrix}, \tag{4.9}$$

where $J_3(\theta_{tilt})$ is a Jacobian relating the rates of the pan and tilt angles to the resulting camera velocity screw, as given by:

$$J_3(\theta_{tilt}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \cos(\theta_{tilt}) & 0 \\ 0 & 1 \\ \sin(\theta_{tilt}) & 0 \end{bmatrix} \tag{4.10}$$

Note that with ${}^B\mathbf{P}_C = [a\ 0\ 0]^T$, the Jacobian only depends on the tilt angle. In practice, a camera with a pan and tilt unit often comes with angle encoders so that

the pan and tilt angles can be sensed within the unit.

Camera motion due to vehicle motion. Now we consider the pan and tilt camera to be rigidly attached to the vehicle as illustrated in Figure 4.7. The camera reference frame is indicated by $\mathcal{F}_C$, having the camera optical axis pointing along its z-axis. The coordinates of the origin of the camera frame with respect to the vehicle reference frame are constant and given by ${}^M\mathbf{P}_C = [d_x\ d_y\ d_z]^T$. The orientation of $\mathcal{F}_C$ in $\mathcal{F}_M$ is given by the rotation matrix ${}^M_C R$, which is usually parameterized by the x-y-z fixed angle representation given by the respective γ, β, α angles: ${}^M_C R = {}^M_C R(\gamma, \beta, \alpha)$. When controlling the camera pan and tilt angles, this rotation matrix becomes time-variant, depending on the current value of the pan and tilt angles: ${}^M_C R(\gamma + \theta_{pan}, \beta + \theta_{tilt}, \alpha)$.

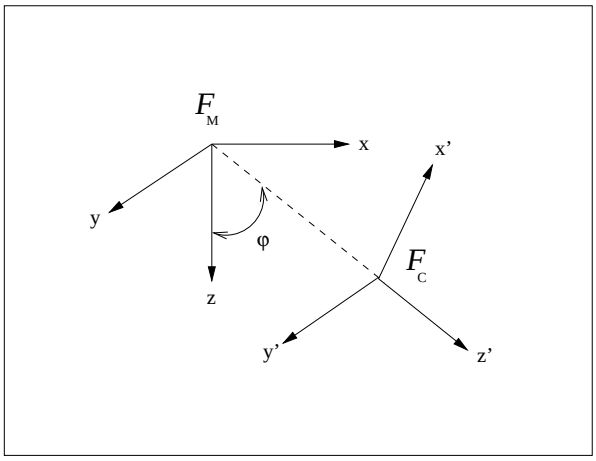


Figure 4.7: Position and orientation of the camera frame, $\mathcal{F}_C$, in the vehicle reference frame, $\mathcal{F}_M$.

Throughout the rest of this thesis, we consider the orientation of the pan and tilt camera in the vehicle frame to be $\gamma = \alpha = 0$ and $\beta = \varphi$, as illustrated in Figure 4.7. For the case $\varphi = 0$, the camera points down, looking perpendicular to the ground-plane. The rotation matrix thus becomes a function of some established value of φ

and the current pan (θ_{pan}) and tilt (θ_{tilt}) angles:

$$ {}^{M}_{C}R(\theta_{pan}, \varphi{+}\theta_{tilt}) = \begin{bmatrix} \cos(\theta_{tilt}+\varphi) & \sin(\theta_{tilt}+\varphi)sin(\theta_{pan}) & \sin(\theta_{tilt}+\varphi)\cos(\theta_{pan}) \\ 0 & \cos(\theta_{pan}) & -\sin(\theta_{pan}) \\ -\sin(\theta_{tilt}+\varphi) & \cos(\theta_{tilt}+\varphi)\sin(\theta_{pan}) & \cos(\theta_{tilt}+\varphi)\cos(\theta_{pan}) \end{bmatrix} \tag{4.11} $$

Taking $\mathbf{v}_C = [\mathbf{u}_C \ \mathbf{w}_C]^T$ as the 6×1 camera velocity screw, the relation between the camera and the vehicle velocity screws is given by:

$$ \mathbf{w}_C = {}^{C}_{M}R\mathbf{w} \tag{4.12} $$

$$ \mathbf{u}_C = {}^{C}_{M}R(\mathbf{u} + \mathbf{w} \times {}^{M}\mathbf{P}_C) \tag{4.13} $$

where $\mathbf{w}$ and $\mathbf{u}$ are the angular and linear components of the vehicle velocity vector.

For control purposes it is useful to relate the camera velocity screw to the vehicle control inputs. To do so, we substitute $\mathbf{w} = [0 \ 0 \ \omega_3]^T$ and $\mathbf{u} = [u_1 \ 0 \ u_3]^T$ into equations (4.12) and (4.13), where we have plugged the elements of the vehicle control input vector, $\mathbf{T}_M$ (as defined in equation (4.5)), into the vehicle velocity screw. Rewriting the resulting expressions into matrix form yields:

$$ \mathbf{v}_C = J_4(\varphi, \theta_{pan}, \theta_{tilt}, {}^{M}\mathbf{P}_C)\mathbf{T}_M, \tag{4.14} $$

where $J_4(\varphi, \theta_{pan}, \theta_{tilt}, {}^{M}\mathbf{P}_C)$ is the 6×3 control input Jacobian, written as a function of the camera orientation and position in the vehicle reference frame. For controlled pan and tilt angles, this Jacobian is time-variant.

In the case of the blimp, the on-board camera has no pan and tilt unit ($\theta_{tilt} = \theta_{pan} = 0$), so that the rotation matrix simplifies to:

$$ {}^{M}_{C}R = \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \tag{4.15} $$

The resulting control input Jacobian in this case becomes time-invariant and has

the following particular structure:

$$J_4(\varphi, {}^M\mathbf{P}_C) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & -d_y\cos(\varphi) \\ 0 & 0 & d_x \\ \sin(\varphi) & \cos(\varphi) & -d_y\sin(\varphi) \\ 0 & 0 & -\sin(\varphi) \\ 0 & 0 & 0 \\ 0 & 0 & \cos(\varphi) \end{bmatrix} \tag{4.16}$$

This time-invariant control input Jacobian plays an important role in the design of control laws, later on in the thesis.

Overall camera kinematics. After having obtained the camera velocity screw as a function of the pan and tilt velocities as well as a function of the vehicle control input vector, the overall camera kinematic relations can be obtained from the composition of equations (4.14) and (4.9), resulting in:

$$\mathbf{v}_C = J_4(\varphi, \theta_{pan}, \theta_{tilt}, {}^M\mathbf{P}_C)\mathbf{T}_M + J_3(\theta_{tilt}) \begin{bmatrix} \omega_{pan} \\ \omega_{tilt} \end{bmatrix} \tag{4.17}$$

This expression gives all degrees of freedom of the camera velocity screw, that are controllable from the vehicle control input vector and the pan and tilt velocities.

If no pan and tilt unit is available on-board, this relation simplifies to:

$$\mathbf{v}_C = J_4(\varphi, {}^M\mathbf{P}_C)\mathbf{T}_M, \tag{4.18}$$

where J_4 is given by equation (4.16) and is time-invariant.

4.5 Summary

In this chapter we introduced the blimp and the underwater robot used for experiments in this thesis. The dynamic and kinematic models of the blimp were useful for simulation purposes and to gather insight into the physical principals of operation.

Since the main goal in this thesis is to explore the use of vision in order to control the robots, measurements are done in the camera reference frame. The relationship

between the camera velocity screw and the vehicles control input vector plays an important role in the design of control laws for visual navigation. This relationship is a function of the camera extrinsic parameters which throughout the thesis are assumed to be known. In practice, they can be obtained from a camera calibration procedure.

The underwater robot is equipped with a pan and tilt unit, introducing two extra degrees of freedom for the camera. These were modeled in the base frame of the pan and tilt unit. Upon controlling the pan and tilt angles, the orientation of the camera frame in the vehicle frame becomes time-variant, depending on the current values of the pan and tilt angles. The pan and tilt degrees of freedom will be used in a later stage for image stabilization.

Chapter 5

Station Keeping and Docking

The problem addressed in this chapter is that of controlling a floating vehicle (either the blimp or the ROV) in order to achieve station keeping and docking. Station keeping consists in stabilizing the vehicle relative to some external reference frame or environmental region, thus rejecting external disturbances like currents. The docking problem is intimately related to station keeping, and consists of controlling the vehicle in order to attain a desired position and/or orientation relative to a chosen coordinate frame. Vision is used to extract information about the vehicle's pose so as to realize the station keeping and docking behaviors. These tasks are then formulated in a visual servoing framework.

Although a dynamic model for the blimp was derived in Chapter 4, the control laws that shall be proposed in this chapter are based on kinematic error functions only. However, deriving the dynamic model gave us a better insight into the system's behavior and the coupling effects between kinematic variables could be identified.

Section 5.1 starts with an outline of the various visual servoing strategies that are established in literature. A tutorial on this topic can be found in [27]. For navigation purposes, these strategies can be roughly classified into two architectures: (i) position based (or 3D-) visual servoing and (ii) image based servoing. In the case of 3D servoing, images are used to reconstruct the scene and estimate 3D positions/orientations from visual information. In the image based approach, features

are measured directly from the image plane and used to synthesize the control laws without an intermediate reconstruction phase.

In Section 5.2, the station keeping task is defined for the case of 3D visual servoing. In Section 5.3 this task is defined for the image based approach. Since docking and station keeping are intimately related (docking can be thought of as station keeping over a wider range), these tasks also apply for docking.

Finally, an image stabilization technique is proposed which controls the camera pan and tilt angles in such a way that it drives a visual landmark to the image center. This adds stability to the station keeping tasks since the visual landmark remains centered in the image. The problem of how to accommodate the image stabilization technique into the proposed visual servoing tasks will be discussed in the final section.

5.1 Visual servoing architectures

Visual control loops have been introduced in order to increase the flexibility and the accuracy of robot systems. Most references deal with visual servoing for positioning tasks in manipulator control ([27, 6, 20]) and the approaches can be mainly classified into 3D visual servoing and image based servoing. Recently, these approaches have been applied to the problem of mobile robot navigation ([19, 32, 18]). The main difficulty in these cases arise due to the non-holonomic constraints of the vehicle when not all degrees of freedom are controllable.

The approach adopted for visual servoing follows the theory referred to as the *task function approach* [18, 6], where the servoing task is formulated as regulating a particular error function to zero. The task is represented by a particular alignment between the camera frame and the landmark and is fulfilled when the error function is zero.

3D servoing. In the 3D approach, features are extracted from the image and used to estimate the pose $\mathbf{x}$ of the camera relative to the target. In this thesis, a planar landmark is chosen as a target.

A kinematic error function is defined between the current and desired camera pose. For station keeping, this error function is given by the difference between the actual and desired pose:

$$E(\mathbf{x}) = \mathbf{x} - \mathbf{x}^*, \tag{5.1}$$

where $\mathbf{x}^*$ is the desired pose. For simulation purposes, the real values for $\mathbf{x}$ can be obtained from integrating the kinematic relations of the camera given by equations (4.12) and (4.13), if not considering the pan and tilt degrees of freedom. An estimate of the current pose, $\hat{\mathbf{x}}$, is used for feedback and obtained from the self-localization method described in Section 5.2. Using feedback, a regulator is defined that exponentially reduces the errors to zero. Since the kinematic errors are defined in Cartesian space, it is relatively easy to obtain a controller design based upon geometric insight.

The main advantage of the 3D approach is that it directly controls the camera trajectory in Cartesian space. However, since the control design is based on error functions in Cartesian space, the image features used in the reconstruction phase may leave the image and lead to servoing failure. Another problem is that stability is difficult to analyze since errors in the pose estimation cannot be analytically computed as a function of calibration errors [20].

Image based servoing. With image based visual servoing, the error signals are defined directly in terms of image feature parameters. An image based visual servoing task is therefore represented by an image error function $e(\mathbf{s})$, where $\mathbf{s}$ is the image feature parameter vector. If the camera is observing a static landmark, then

the elements in $\mathbf{s}$ are only dependent on the relative pose of the camera frame with respect to the landmark. Defining the pose of the camera as an element $\mathbf{x} \in SE_3$, the image feature parameter vector becomes a function of this pose, $\mathbf{s}(\mathbf{x})$.

Although the error $e(\mathbf{s})$ is defined on the image parameters, the robot control input is typically defined in task space coordinates. Therefore it is necessary to relate changes in the image features to changes in the relative camera pose. This relationship is often referred to as the *image Jacobian* or the *interaction matrix*:

$$\dot{\mathbf{s}} = L(\mathbf{s}, Z)\mathbf{v}_C \tag{5.2}$$

In this equation, $\dot{\mathbf{s}}$ gives point velocities of image features, $\mathbf{v}_C$ is the camera velocity screw and L the image Jacobian, which depends on the current values of the image features and also on their relative depth, Z. In general, the image Jacobian is a non-square $n \times 6$ matrix, where n represents the number of image features.

If the chosen image features are the coordinates $\mathbf{s} = [x\ y]^T$ of a projection in the image plane of the 3D point, $[X\ Y\ Z]^T$ (expressed in the camera frame), then the image Jacobian is given by the motion field (see Appendix A):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix} \mathbf{v}_C \tag{5.3}$$

The image based visual servoing task can now be formulated as the regulation of the image features in $\mathbf{s}$ to some desired value $\mathbf{s}^*$:

$$e(\mathbf{s}) = C\big(\mathbf{s} - \mathbf{s}^*\big), \tag{5.4}$$

where the matrix C allows to take more measurements than the actual number of task constraints into account. From feedback it follows that an appropriate control law is given by:

$$\mathbf{v}_C = -\lambda e(\mathbf{s}), \tag{5.5}$$

where λ is some positive scalar defining the speed of convergence. In order to guarantee the convergence of the above control law, the error has to be decreasing

over time, $\dot{e}(\mathbf{s}) < 0$. The error rate is given by:

$$\dot{e}(\mathbf{s}) = \frac{\delta e}{\delta \mathbf{s}}.\frac{\delta \mathbf{s}}{\delta t} = \frac{\delta e}{\delta \mathbf{s}}.\frac{\delta \mathbf{s}}{\delta \mathbf{x}}.\frac{\delta \mathbf{x}}{\delta t}, \tag{5.6}$$

which can be rewritten, using equations (5.4) and (5.2) and substituting $\frac{\delta x}{\delta t} = \mathbf{v}_C$, as:

$$\dot{e}(\mathbf{s}) = CL(\mathbf{s}, Z)\mathbf{v}_C \tag{5.7}$$

Substituting the control law given in equation (5.5) into equation (5.7), the following differential equation is obtained for the error function:

$$\dot{e}(\mathbf{s}) = -\lambda CL(\mathbf{s}, Z)e(\mathbf{s}) \tag{5.8}$$

Exponential convergence of the error function is ensured under the condition:

$$CL(\mathbf{s}, Z) > 0 \tag{5.9}$$

A possible choice for C is the pseudo inverse $C = L(\mathbf{s}, Z)^+$ of the image Jacobian [6]. Since the pseudo inverse $L(\mathbf{s}, Z)^+$ depends on Z-component of each feature point, these values need to be estimated. Using this choice however may lead the system close to or even reach a singularity for the image Jacobian $L(\mathbf{s}, Z)$. An alternative is to consider a constant positive matrix $C = L(\mathbf{s}_d, Z_d)^+$ given by the pseudo-inverse of the image Jacobian computed at the desired image features, $\mathbf{s} = \mathbf{s}_d$ and $Z = Z_d$. In this case, the convergence of the task function is guaranteed in the neighborhood of the desired 3D camera position (station keeping). However, care should be taken if the initial camera position is too far away from the desired position (docking).

5.2 3D Station keeping

In Chapter 3, we introduced the problem of image registration and proposed a method to track the projection of a planar surface in the image-plane over time,

based on planar projective transformations. In this section we consider the problem of robot pose estimation from this image registration. These estimates can then be used in a visual feedback loop for the design of a 3D station keeping control law.

Camera trajectory reconstruction. The geometry of a camera observing a plane while traveling in 3D space, is depicted in Figure 5.1. The camera reference frame at the initial position and orientation is indicated by $\mathcal{F}_C$ and its pose at some successive time instant is given by $\mathcal{F}_{C^*}$. Under rigid motion of the camera reference frame in 3D space, the relative position and orientation of $\mathcal{F}_C$ in $\mathcal{F}_{C^*}$ is given by a rotation matrix R (note: $R = {}^{C^*}_{C}R$) and a translation vector $\mathbf{t}$.

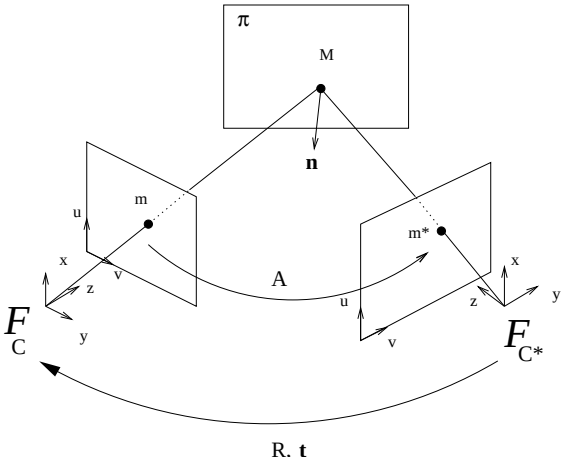


Figure 5.1: Geometry of the camera trajectory reconstruction.

The coordinates of an observed point M, belonging to the plane π, are defined in $\mathcal{F}_C$ and given by ${}^{C}\mathbf{P}_M = [M_x\ M_y\ M_z]^T$. The coordinates of the same point, but expressed in $\mathcal{F}_{C^*}$ are related to ${}^{C}\mathbf{P}_M$ by:

$$ {}^{C^*}\mathbf{P}_M = R\,{}^{C}\mathbf{P}_M + \mathbf{t} \tag{5.10}$$

The observed plane π is defined in $\mathcal{F}_C$ by its normal $\mathbf{n} = [a\ b\ c]^T$ and distance d

to the origin of the camera coordinate system at the initial time instant, given by:

$$\pi : \ aX + bY + cZ = d \tag{5.11}$$

If the point M belongs to the plane π, then:

$$\mathbf{n}^T \, {}^C\mathbf{P}_M = d, \tag{5.12}$$

which can be rewritten as:

$$\frac{\mathbf{n}^T \, {}^C\mathbf{P}_M}{d} = 1, \tag{5.13}$$

This allows to rewrite equation (5.10) as:

$${}^{C^*}\mathbf{P}_M = \big(R + \frac{\mathbf{t}\mathbf{n}^T}{d}\big){}^C\mathbf{P}_M, \tag{5.14}$$

relating the coordinates of point M when expressed in $\mathcal{F}_C$ and $\mathcal{F}_{C^*}$.

As seen from Section 2.4, the projections m and $m*$ of the point M into the normalized camera coordinate system of $\mathcal{F}_C$ and $\mathcal{F}_{C^*}$, are related by a homography:

$$m^* = Am, \tag{5.15}$$

which can be obtained from the inter-image homography in pixels, by knowing the camera intrinsic parameters. Moreover, the projective coordinates of m and m^* in the normalized camera coordinate system are given by ${}^C\mathbf{P}_M$ and ${}^{C^*}\mathbf{P}_M$, respectively. Comparing equations (5.14) and (5.15), it follows that the homography A is related to the geometric parameters of the camera motion by:

$$A = R + \frac{\mathbf{t}\mathbf{n}^T}{d} \tag{5.16}$$

Given an inter-image homography, it is thus possible to reconstruct the relative displacement of the camera coordinate frame. In [7], this decomposition is resolved, which is also known as *scaled Euclidean reconstruction*. It follows that upon determining the inter-image homography, it is possible to explicitly compute the normal

$\mathbf{n}$ to the plane and the relative rotation R of the camera retina. Since the homography is defined up to a scale factor, the distance to the plane, d, and the relative camera translation $\mathbf{t}$ can only be recovered up to a scale factor, allowing to explicitly compute their ratio $\mathbf{t}^* = \mathbf{t}/d$.

To reconstruct the real camera translation, *a priori* information is needed of for example the real value of d in $\mathcal{F}_C$. The real camera translation can then be determined according to:

$$\mathbf{t}_{real} = d_{real}\mathbf{t}^*, \tag{5.17}$$

where $\mathbf{t}^*$ is the scaled translation vector. Usually, the real distance to the plane can be obtained from a sensor other then the camera (e.g. an acoustic depth sensor), allowing to determine the scale factor.

Results on homography decomposition. The quality of the reconstructed camera trajectory depends mainly on the accuracy of the homography estimation. Since the homography is calculated from the corner coordinates of the tracked image region, small errors on these coordinates result in errors on the trajectory reconstruction.

To illustrate this problem, some tests were realized with simulated camera motions in 3D. We included translations along the x, y- and z-axis and rotation around the camera optical axis. These are the principal components of motions for a camera mounted on the vehicles, pointing down and observing the ground-plane perpendicularly. The same test was performed for different sizes of tracked image regions.

In Figures 5.2, the selected template sizes are shown for two experiments. The $L2$-norm between the difference of real corner coordinates and estimated corner coordinates, $\|\mathbf{q} - \hat{\mathbf{q}}\|$, are also plotted for both cases. This norm indicates the tracking error over time. It follows that for both cases, this error does not exceed 1.2 pixels during the experiment. This leads to a maximum "average" offset of

$1.2/8 = 0.15$ pixels on each corner coordinate.

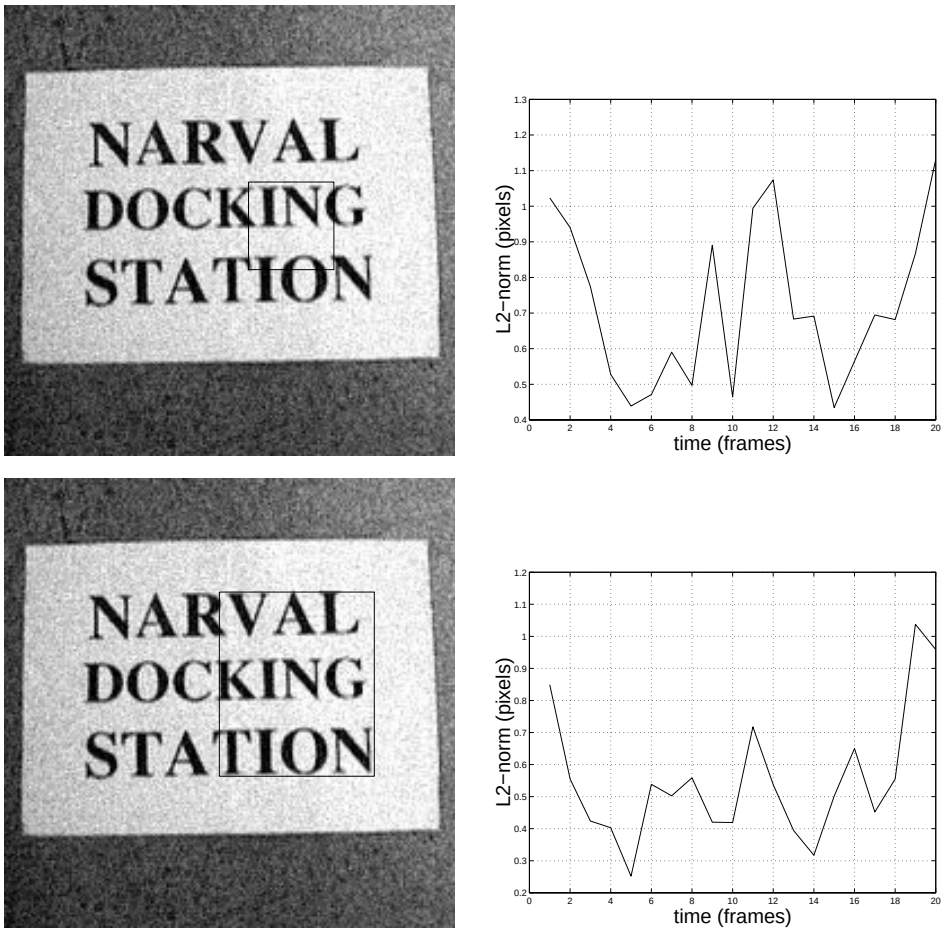


Figure 5.2: Initial selected template and quality of tracking in the image plane for a small window (top) and a large ROI (bottom). The $L2$-norm is obtained from the difference between the real and estimated coordinates of the tracked windows over time.

The results obtained from homography decomposition are indicated in Figure 5.3 for both cases. During reconstruction, the camera intrinsic parameters were assumed to be known. Both the real trajectory and the estimated trajectory are shown in each plot. It follows that the trajectory reconstruction for the bigger template is of better quality than for the case of the small template, whereas the tracking error in the image plane is of the same order.

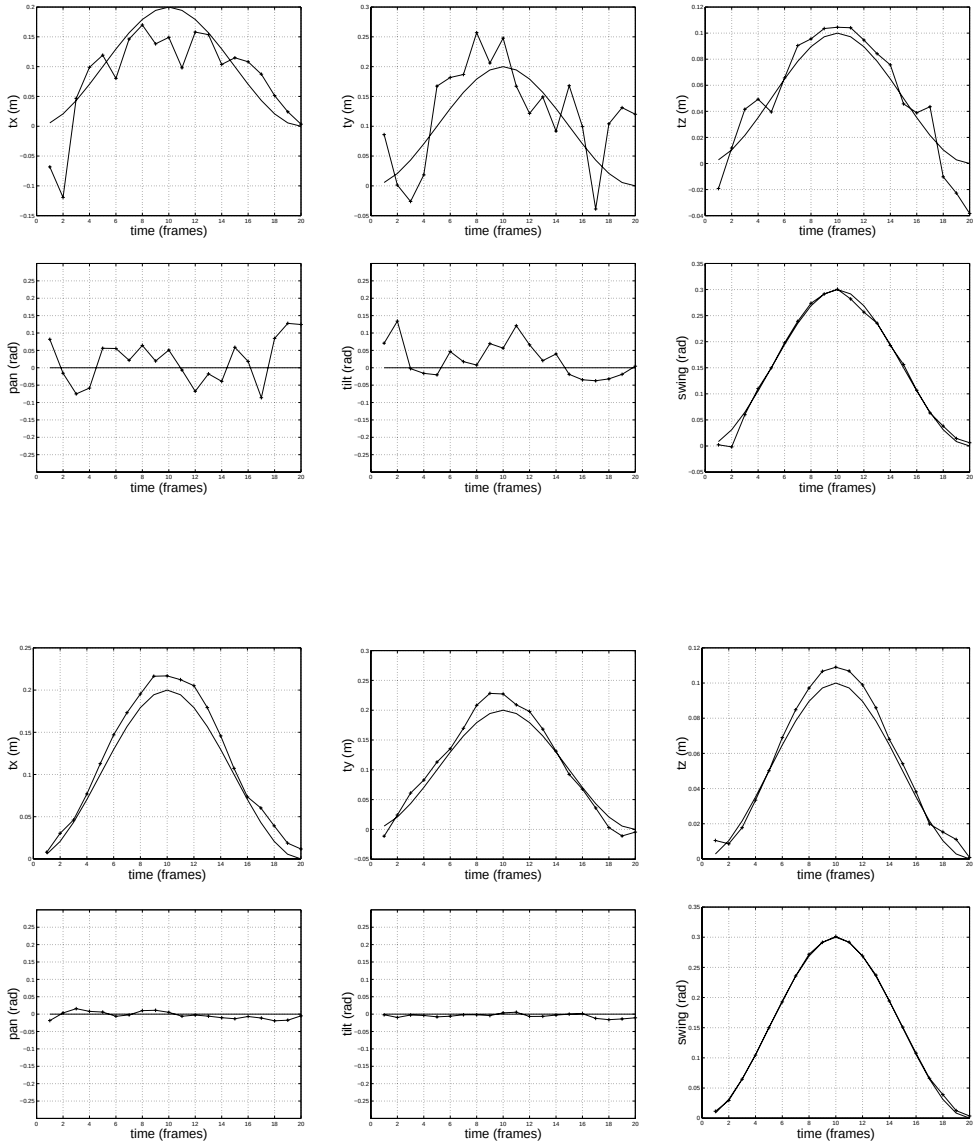


Figure 5.3: Quality of the trajectory reconstruction from homography decomposition of the smaller (top) and bigger (bottom) template. The continous plots are ground truth values and the + marked plots are estimated values.

The main errors in both cases arise in the reconstruction of relative displacement in the x- and y-direction. This is due to the fact that if no strong perspective distortion is present, translation in the image plane can be accounted for by either a camera translation or a rotation around the *pan* and *tilt* angles, leading to an ambiguity in the reconstruction. The rotation around the camera optical axis (swing angle) and the translation along the z-axis are reconstructed satisfactory for both cases. These camera motions correspond to unique observable deformations in the image plane.

Decoupled control design. The control problem addressed here is that of stabilizing the vehicle about a desired pose in the horizontal plane defined by the xy-axes of the earth-fixed reference frame, $\mathcal{F}_W$, while maintained at some fixed height. For a floating vehicle moving at low speed, it is reasonable to assume zero roll and pitch angles, $\gamma = \beta = 0$. Under this assumption, the model described by equations (4.1) and (4.3) can be considered decoupled into two non-interacting (or lightly interacting) subsystems, describing motion in the horizontal plane and in the vertical plane of $\mathcal{F}_W$.

Referring to the kinematic relations of the vehicle as given by equation (4.3), motion in the horizontal plane can be described by a simplified kinematic model:

$$\dot{x} = u_1 \cos(\alpha) \quad \dot{y} = u_1 \sin(\alpha) \quad \dot{\alpha} = \omega_3 \tag{5.18}$$

where (x, y) are Cartesian coordinates of the origin of the vehicle reference frame, $\mathcal{F}_M$, in the horizontal plane, α is the orientation (yaw-angle) of the vehicle in the horizontal plane and (u_1, ω_3) are the vehicle linear forward velocity and angular velocity around its z-axis, expressed in the vehicle reference frame.

The control inputs are the vehicle forward speed, u_1 and the vehicle angular speed, ω_3, resulting from the common-mode and differential-mode signals of the thrust of the left and right propellers. This motion model captures the non-holonomic

constraints of the vehicle motion in the horizontal plane.

Under the decoupling assumption, motion in the vertical plane is described by the simple kinematic relation: $\dot{z} = u_3$, where u_3 is the vehicle vertical linear speed.

Control in the horizontal plane. In [4], a non-linear control law is proposed to stabilize a non-holonomic robot about a desired posture (x_d, y_d, α_d) by using smooth time-varying feedback control. Using this control strategy, the desired behavior for the closed loop system is to stabilize the vehicle in the horizontal plane towards any final position and orientation, with time-varying feedback of the type:

$$\begin{bmatrix} u_1 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ 0 & -g(t) & -k_3 \end{bmatrix} \begin{bmatrix} & R^T_{2\times 2}(\alpha) & 0 \\ & & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_d \\ y - y_d \\ \alpha - \alpha_d \end{bmatrix}, \tag{5.19}$$

where u_1 is the vehicle forward speed, ω_3 is the vehicle angular speed in the horizontal plane , k_1 and k_3 are positive numbers, $R(\alpha)$ is the 2×2 rotation matrix describing the robot orientation in the horizontal plane and $g(t)$ is a bounded function at least once differentiable such that $\delta g(t)/\delta t$ does not tend to zero when t tends to infinity; for instance, $g(t) = \sin(t)$. The rotation matrix $R(\alpha)$ is included so as to transform the error vector from the fixed reference frame to the vehicle reference frame.

The station keeping task in this case is represented by the error function $E(\mathbf{x}) = [x\ y\ \alpha]^T - [x_d\ y_d\ \alpha_d]^T$, where $\mathbf{x}$ is the partial pose of the vehicle frame $\mathcal{F}_M$ instead of the camera frame (as in equation (5.1)). The error function at each time instant is given by the relative displacement of the vehicle frame with respect to the initial desired position and orientation. This displacement can be obtained from the camera trajectory reconstruction, as illustrated in Figure 5.4, where $\mathcal{F}_M$ and $\mathcal{F}_C$ are the initial position and orientation of the vehicle and camera frame and $\mathcal{F}_{M*}$ and $\mathcal{F}_{C*}$ indicate their current position and orientation in 3D space.

The relative position and orientation of $\mathcal{F}_{M^*}$ in $\mathcal{F}_M$ is described by the transfor-

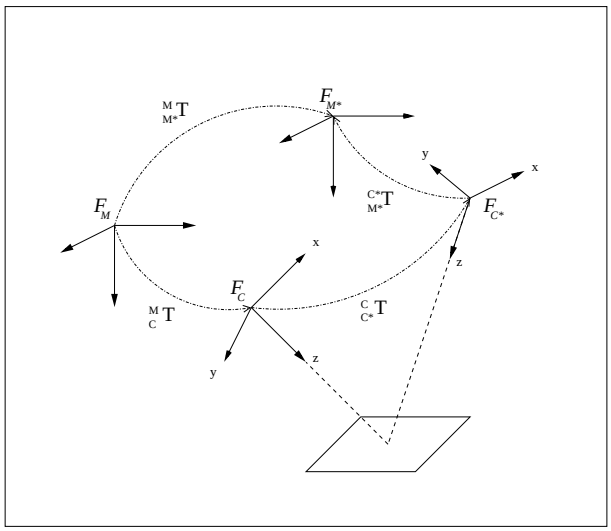


Figure 5.4: The relative displacement of the actual vehicle pose, ${}^{M}_{M^*}T$, can be obtained from the composition of ${}^{M}_{C}T, {}^{C}_{C^*}T$ and ${}^{C^*}_{M^*}T$.

mation ${}^{M}_{M^*}T$ defined as:

$$ {}^{M}_{M^*}T = \begin{bmatrix} {}^{M}_{M^*}R & {}^{M}\mathbf{P}_{M^*} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{5.20} $$

This transformation can be obtained from the composition of the following transformations:

$$ {}^{M}_{M^*}T = {}^{M}_{C}T \, {}^{C}_{C^*}T \, {}^{C^*}_{M^*}T, \tag{5.21} $$

where ${}^{C}_{C^*}T$ results from the camera trajectory reconstruction and ${}^{M}_{C}T$ and ${}^{C^*}_{M^*}T$ are given by the position and orientation of the camera frame in the vehicle frame at the initial and current time instant. Note that the camera trajectory reconstruction recovers the relative translation up to a scale factor, depending on the initial distance to the plane. Assuming this initial distance to be given by some measurement other than vision, the scale factor can be determined and the real displacement obtained.

Control in the vertical plane. Measuring the relative displacement along the z-axis of the vehicle, a PID-controller can be tuned for dynamically controlling the

altitude by regulating the relative error to zero:

$$u_3 = K_p e_z(t) + K_d \dot{e}_z(t) + K_i \int_0^t e(\tau) d\tau \tag{5.22}$$

where $e_z = z_r - z$ is the error in altitude. The closed loop system aims at stabilizing the vehicle at some fixed height, rejecting any disturbances coming from variations in the buoyancy force or dynamic coupling effects between the vehicle forward velocity, u_1 (generating small pitching moments) and altitude, z.

5.3 Image based station keeping

In this section, two different image based visual servoing task will be formulated. First a *point-to-point* positioning task will be proposed, which regulates the centroid of the tracked window to the image center. However, by observing only the centroid in the image plane, no information can be extracted about the orientation and vertical displacement of the camera frame in 3D space. Therefore a hybrid control strategy is proposed that uses 3D servoing for the vertical plane as described in the previous section, whereas the orientation is not regulated to a desired value at all.

Second, an image based control law will be derived that combines the information of the four corners of the tracked window so as to regulate each of them to a desired position in the image plane. In this case, the relative position of the four corners in the image plane reveal sufficient information so as to control both positional offset and orientation of the camera in the horizontal plane as well as the altitude of the camera in the vertical plane.

For both tasks, the design of the control laws is obtained by considering only those camera degrees of freedom resulting from the vehicle control input vector. The degrees of freedom defined by the camera pan and tilt angles will be used later on for image stabilization.

Point-to-point positioning. Consider the centroid of the tracked window, as illustrated in Figure 5.5, to be the only observed image feature.

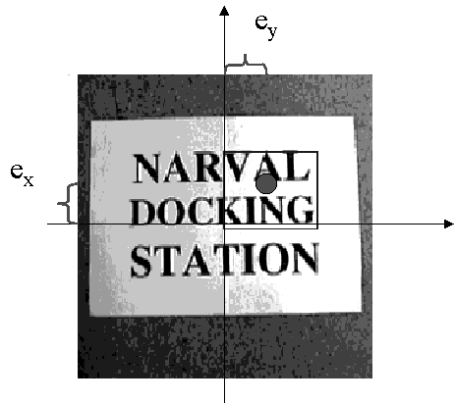


Figure 5.5: Point-to-point positioning by driving the centroid of the tracked window to the image center.

The velocity of the centroid in the image plane is related to the camera velocity screw in 3D space by:

$$\dot{\mathbf{s}} = L(\mathbf{s}, Z)\mathbf{v}_C, \tag{5.23}$$

where $\mathbf{s} = [x_c\ y_c]^T$ contains the centroid coordinates, $\mathbf{v}_C$ is the camera velocity screw and $L(\mathbf{s}, Z)$ is the image Jacobian as given in equation (5.2). The desired camera velocity screw that drives the centroid, $\mathbf{s}$, to any desired position, $\mathbf{s}^*$, is given by the following control law:

$$\mathbf{v}_C = -\lambda L(\mathbf{s}, Z)^+(s - s^*) \tag{5.24}$$

Since the vehicle has limited controllable degrees of freedom, the resolved camera velocity screw may require trajectories that are physically impossible to achieve for the vehicle. Therefore, the vehicle non-holonomic constraints need to be taken into account.

Assuming zero pan and tilt angles and velocities, the relation between the camera velocity screw and the control inputs vector is given by the control input Jacobian, $J_4(\varphi, {}^M\mathbf{P}_C)$, given in equation (4.18). Substituting $\mathbf{v}_C = J_4(\varphi, {}^M\mathbf{P}_C)\mathbf{T}_M$ into equa-

tion (5.23), a new image Jacobian is obtained that relates image point velocities to the vehicle control inputs:

$$\dot{\mathbf{s}} = L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C)\mathbf{T}_M, \tag{5.25}$$

where $L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C)$ is the new image Jacobian given by:

$$L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C) = L(\mathbf{s}, Z)J_4(\varphi, {}^M\mathbf{P}_C) \tag{5.26}$$

In this equation, L_M is written also as a function of the camera extrinsic parameters with respect to the vehicle reference frame.

The control law that drives the centroid to the desired position and respects the vehicle non-holonomic constraints is now given by:

$$\mathbf{T}_M = -\lambda L_M^+(s - s^*) \tag{5.27}$$

Since this solution is under-constrained, the correct pseudo-inverse is given by $L_M^+ = L_M^T(L_M L_M^T)^{-1}$. For station keeping, a possible pseudo-inverse can be obtained from $L_M(\mathbf{s}^*, Z, \varphi, {}^M\mathbf{P}_C)$, were the desired centroid position is at the image center, $\mathbf{s}^* = (0, 0)$. If the camera extrinsic parameters are known and given by $\varphi = 0$ (camera looking down perpendicularly to the ground plane) and $\mathbf{P_{co}} = [d_x\ d_y\ d_z]$, the resulting pseudo-inverse simplifies into the following structure:

$$L_M^+ = Z \begin{bmatrix} -1 & -\frac{d_y}{d_x} \\ 0 & 0 \\ 0 & -\frac{1+2d_y^2}{d_x} \end{bmatrix} \tag{5.28}$$

This results into the following simple control law:

$$\mathbf{T}_m = -\lambda Z \begin{bmatrix} -e_x - \frac{d_y}{d_x}e_y \\ 0 \\ -\frac{1+2d_y^2}{d_x}e_y \end{bmatrix}, \tag{5.29}$$

where e_x and e_y are the centroid offset from the image center along the x- and y-axis of the normalized camera frame. It is straight forward to verify that the matrix

$L_M^+ L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C) > 0$ is definite positive, which guarantees the convergence of the control law. Note however, that the pseudo-inverse was obtained around the desired image feature position so that the convergence prove only is valid for small deviations of the camera in 3D space around the position corresponding to the desired view.

It follows that the vehicle vertical speed is not controlled in driving the centroid to the image center. Intuitively, this makes sense since from observing only one feature point in the image plane no information can be extracted about camera motions along projection rays. Therefore, for point-to-point positioning using only the centroid information, a hybrid control strategy is required that uses 3D servoing for the vertical motion, as described in the previous section.

Equation (5.29) also shows that with $\varphi = 0$, the lateral offset e_y of the centroid can only be controlled if $d_x \neq 0$.

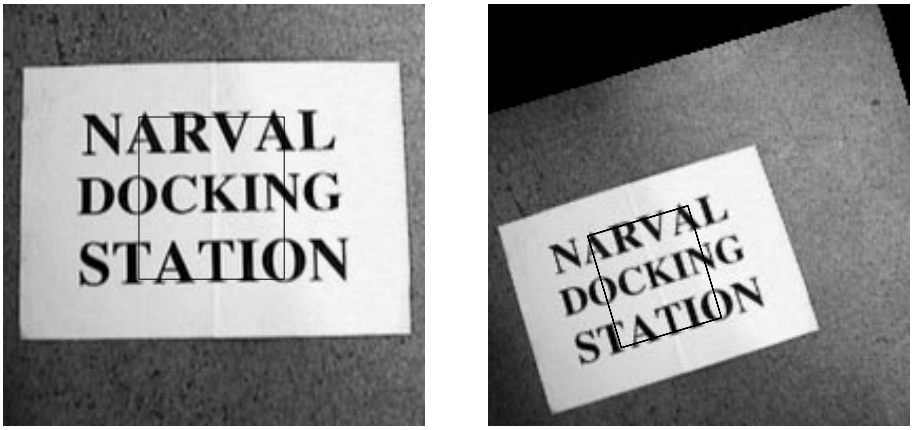
Since a proportional control law is not completely satisfactory in the presence of constant disturbances, a vectorial PID control is used instead, such that the desired control input is given by:

$$\mathbf{T}_M = -\lambda L_M^+ (K_p \mathbf{e} + K_d \dot{\mathbf{e}} + K_i \int \mathbf{e} dt), \tag{5.30}$$

where $K_p(2 \times 2), K_d(2 \times 2)$ and $K_i(2 \times 2)$ are positive diagonal matrices.

Visual servoing for position, orientation and altitude control. We now design an image based control law that combines the information of the four corners of the tracked window so as to regulate each of them to a desired position in the image plane. The problem is illustrated in Figure 5.6, where the current and desired view are given. The position of the four corners in the image plane conveys information about the vehicle orientation, altitude and translational offset in 3D space.

Defining the vector $\mathbf{s} = [x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4]^T$, containing the current coordi-



(a) (b)

Figure 5.6: (a) desired view, (b) current view.

nates of each corner, then their point velocities are related to the camera velocity screw by:

$$\dot{\mathbf{s}} = L(\mathbf{s}, Z)\mathbf{v}_C, \tag{5.31}$$

where $L(\mathbf{s}, Z)$ is the 8×6 image Jacobian obtained from stacking the 2×6 Jacobians corresponding to the motion field of each corner coordinate pair.

Relating the corner velocities in the image plane to the vehicle control inputs allows to take the vehicle non-holonomic constraints into account. Assuming zero pan and tilt angles and velocities, the new Jacobian is given by:

$$\dot{\mathbf{s}} = L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C)\mathbf{T}_M \tag{5.32}$$

where L_M is obtained from substituting $\mathbf{v}_C = J_4(\varphi, {}^M\mathbf{P}_C)\mathbf{T}_M$ into equation (5.31), resulting in:

$$L_M(\mathbf{s}, Z, \varphi, {}^M\mathbf{P}_C) = L(\mathbf{s}, Z)J_4(\varphi, {}^M\mathbf{P}_C) \tag{5.33}$$

The control law that regulates the corners to their desired position in the image plane is given by:

$$\mathbf{T}_M = -\lambda L_M^+(\mathbf{s} - \mathbf{s}^*) \tag{5.34}$$

Now the solution is over-determined, which means that there are more feature parameters than task degrees of freedom. The correct pseudo-inverse for this case is given by $L_M^+ = (L_M^T L_M)^{-1} L_M^T$. This pseudo inverse is a 8×3 matrix which makes analytic analysis much harder, even for simplified cases where a constant pseudo inverse is used around some equilibrium state.

5.4 Image stabilization

The use of kinematic models for visual servoing (for both 3D as image based servoing) is not always realistic for floating vehicles with relative slow dynamics. Therefore it is likely that during station keeping maneuvers, the target gets out of view due to limited bandwidth in acceleration. In an attempt to avoid these situations, image stabilization is used, aiming at centering the target in the image by controlling the camera pan and tilt angles. We consider a separate control scheme that independently controls these angles.

Image stabilization with pan and tilt control. The image stabilization task is formulated as the regulation of the centroid of the tracked region to the image center, using the camera pan and tilt control inputs. Defining $\mathbf{s} = [x_c \; y_c]^T$ as the centroid of the tracked region in the image plane, its time derivatives are related to the camera velocity screw according to equation (5.23). The relationship between the camera velocity screw, $\mathbf{v}_C$ and the pan and tilt velocities is given in equation (4.9). Substituting equation (4.9) into equation (5.23), then relates image point velocities to pan and tilt velocities:

$$\dot{\mathbf{s}} = L_{pan/tilt}(\mathbf{s}) \begin{bmatrix} \omega_{pan} \\ \omega_{tilt} \end{bmatrix}, \tag{5.35}$$

where $L_{pan/tilt}(\mathbf{s})$ is the new 2×2 image Jacobian given by:

$$L_{pan/tilt}(\mathbf{s}) = L(\mathbf{s}, Z) J_3(\theta_{tilt}) \tag{5.36}$$

In this equation, $J_3(\theta_{tilt})$ is the Jacobian relating pan and tilt velocities to the camera velocity screw, as in equation (4.9). Due to the structure of $J_3(\theta_{tilt})$, the new image Jacobian does not depend on the relative depth of image points. Considering the tracked centroid to be in the neighborhood of the image center, a control law that drives the centroid to the image center can be obtained as:

$$\begin{bmatrix} \omega_{pan} \\ \omega_{tilt} \end{bmatrix} = -\lambda L^{-1}_{pan/tilt}(\mathbf{s}^*)(\mathbf{s} - \mathbf{s}^*), \tag{5.37}$$

where $\mathbf{s}^* = [0\ 0]^T$ is at the image center. After calculating, the following expression for the desired pan and tilt velocities is obtained:

$$\begin{bmatrix} \omega_{pan} \\ \omega_{tilt} \end{bmatrix} = -\lambda \begin{bmatrix} 0 & \frac{1}{cos(\theta_{tilt})} \\ -1 & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix}, \tag{5.38}$$

where, e_x and e_y are given by the centroid offset from the image center. Verifying equation (5.9), it follows that local exponential convergence of the control law is guaranteed.

Image stabilization and 3D Station Keeping. We now consider the case of performing 3D station keeping with image stabilization. One of the drawbacks on 3D servoing was that since the camera trajectory is controlled in 3D space, it can not be guaranteed that the visual target stays within the camera field of view under closed-loop control. Since the image stabilization tasks aims at maintaining the visual target centered in the image, we expect to avoid these situations up to some extent.

Upon simultaneously controlling the vehicle and the camera pan and tilt angles, the reconstruction of the relative camera displacement reveals an ambiguity when trying to reconstruct the relative displacement of the vehicle reference frame from it. Changes in the camera orientation can be originated both in pan and tilt control actions as well as in changing orientation of the vehicle frame due to the 3D station keeping maneuvers.

As seen from equation (5.21), the relative displacement of the vehicle reference frame can be uniquely reconstructed from the estimated camera trajectory if the camera position and orientation in the vehicle frame is known at each time instant. Upon actively controlling the camera pan and tilt angles, this orientation changes over time, so that it is necessary to sense the pan and tilt angles.

Once the the relative displacement of the vehicle reference frame is obtained, it is straightforward to perform the 3D station keeping task, as given by the control laws in equations (5.19) and (5.22). Upon fulfilling the 3D station keeping task for the vehicle reference frame, the camera alignment with the target will converge to its initial alignment, with the visual target centered in the image.

Image stabilization and image based servoing. When simultaneously executing the image based station keeping task and the image stabilization task, some care need to be taken. Considering for example the case of point-to-point positioning, both tasks are represented by the same error function so that they can be fulfilled by controlling the camera pan and tilt angles only. The regulation to zero of this error function, defined on image features, can thus be fulfilled without stabilizing the vehicle in 3D space.

In order to realize station keeping, we need to decouple this task from the image stabilization task. This can be done by transforming the corresponding error function according to a mapping that warps the current observed image to an image that would be viewed from zero pan and tilt angles. From equation (5.16), it follows that the inter-image homography upon pure camera rotation is given by: $A = R(\theta_{pan}, \theta_{tilt})$, where A is obtained for the normalized camera model and $R(\theta_{pan}, \theta_{tilt})$ is the rotation matrix parameterized by the pan and tilt angles. Knowing these angles at each time instant, it is possible to construct this rotation matrix. This allows to obtain the following inverse mapping on normalized image coordi-

nates:

$$A^{-1} = R^T(\theta_{pan}, \theta_{tilt}), \tag{5.39}$$

which warps the current image back to a virtual view corresponding to zero pan and tilt angles. Writing the components of the station keeping error function (as used in the previous established image based control laws) into homogeneous representations, their transformed values can be obtained according to this inverse mapping. Note that due to the pure rotation, this mapping is independent of the depth of the image features.

Intuitively, under this mapping, the error function for image based station keeping is measured in an artificially generated view, corresponding to zero pan and tilt angles. This is illustrated in Figure 5.7.

(a) (b) (c)

Figure 5.7: (a) current view if no image stabilization is used, (b) current view with image stabilization; (c) virtual image obtained from re-warping the current view in (b) according to the inverse of the pan and tilt angles.

The figure shows the advantage of using image stabilization, since the real view obtained if no image stabilization was realized (zero pan and tilt angles) contains a partially occluded visual landmark, whereas the current view with image stabilization maintains the landmark centered in the image.

Chapter 6

Experimental results

In this chapter we present and discuss experimental results obtained with the proposed approaches, for controlling both the blimp and an underwater vehicle. A simulation model, including the dynamic behavior of a floating vehicle moving in 3D space, was developed and used to demonstrate results obtained with the 3D visual servoing architecture. Some of these results are also described in [31].

Real experiments and results obtained with image based visual servoing are presented for both the blimp and the ROV [30]. In the case of the blimp, the experiments were executed in a laboratory environment under controlled conditions whereas the test done with the ROV were realized at open sea in Villefranche, France.

6.1 3D visual servoing for the simulation model

We start with presenting results obtained with the 3D visual servoing architecture when applied to an implemented simulation model of a floating robot, including its kinematics and dynamics. Since no parameter identification is done for the real blimp, the parameters for the model were established based on the blimp geometric shape and upon experimentally tuning. This way, the dynamics of the simulation model roughly captures the time constants and coupling behaviour of the real sys-

tem. The blimp model is characterized by the following set of parameters.

Mass matrix ($M = M_{RB} + M_A$):
$M_{RB} = diag\{0.526, 0.526, 0.526, 0.04, 0.1, 0.1\}$ $M_A = diag\{0.1395, 0.3437, 0.3437, 0, 0.0148, 0.0148\}$
Damping matrix ($D = D_S + D_H$):
$D_S = diag\{0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$ $D_H = 0$
Center-of-mass (CG) in $\mathcal{F}_m$:
${}^M\mathbf{P}_{CG} = [0,\ 0,\ 0.35]^T$
Geometric arrangement of the thruster in $\mathcal{F}_M$:
$d_y = 0.1 \quad d_z = 0.4$

Recalling the decoupled control laws as proposed in Section 5.2, station keeping is realized by positioning and orienting the vehicle in the horizontal plane towards the desired pose, while maintaining the vehicle at constant height. Figure 6.1 shows the simulation results of a station keeping experiment with the vehicle. At the left side the trajectory in 3D is illustrated under closed-loop control. The initial lateral deviation in the horizontal plane ($x_0 = 0, y_0 = -3, \alpha_0 = 0$) is reduced and the control stabilizes at the origin ($x_d = 0, y_d = 0, \alpha_d = 0$). The non-linear controller, designed

for a kinematic model, performs well when applied to the non-linear dynamic model of the vehicle, generating smooth control signals. Note that at sample $t \approx 600$ the vehicle starts moving backward. The blimp altitude is dynamically controlled to remain fixed at $z = 0$. The non-zero linear velocity (u_1) generated by the non-linear controller introduces a small pitch angle, resulting in small variations of the blimp height. These variations are dynamically compensated by the altitude controller.

6.2 Image Based servoing with the Blimp

The first real experiments obtained with the blimp describe point-to-point positioning with image based servoing. Station Keeping is thus realized so as to maintain the vehicle at a fixed position, without taking into account its orientation in the horizontal plane. The altitude is controlled based on the tracked quadrangle area. The tracking frequency established was 12 Hz for images with a 128×192 pixel-size and using a 450Mhz processor.

Figure 6.2 shows the temporal evolution of the error signals and control inputs during a docking and station keeping experiment.

Figure 6.2(a) shows the image trajectory of the target point (centroid of the tracked window) under closed-loop control. The control strategy aims at driving this point to the image center (docking) and keep it as close as possible to this center (station keeping). Figure 6.2(b) shows the error between the areas of the reference window and the tracked window and indicates that the blimp is approximately maintained at a constant height. The control signals (the blimp's linear and angular velocities) for the docking and station keeping experiment are plotted in Figure 6.2(c,d).

The approach followed led to encouraging results in moving the error close to zero, and maintaining the target close to the image center. The main difficulties

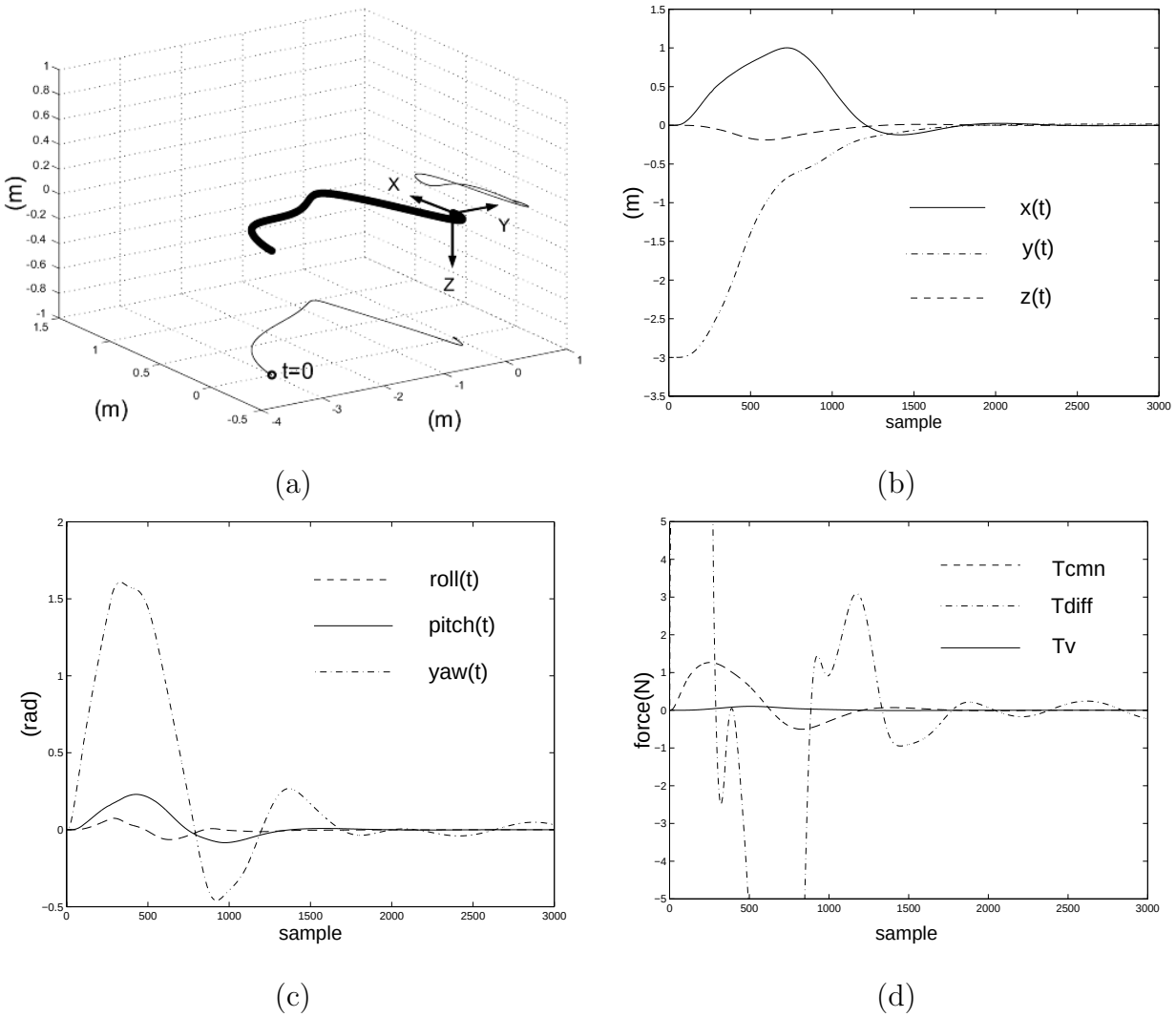


Figure 6.1: Simulation results of a station keeping test with the simulation model; (a) vehicle under closed-loop control while recovering an initial lateral deviation; projections of the trajectory onto the horizontal and vertical plane are shown together with the earth-fixed reference frame ; (b) position over time; (c) orientation over time; (d) generated control signals.

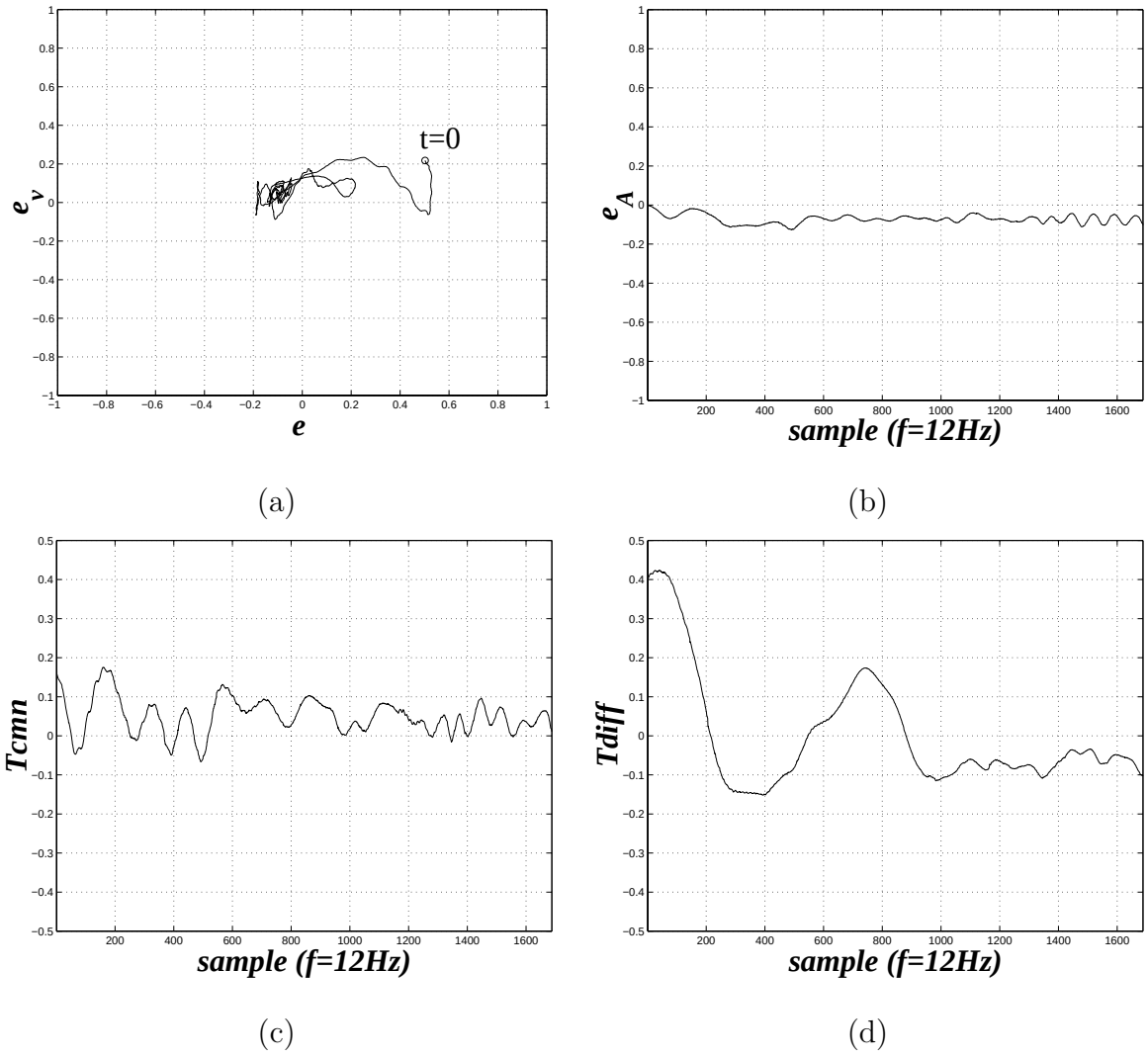


Figure 6.2: Docking and station keeping test with the blimp. (a) trajectory of the centroid of the tracked window (image errors in normalized pixel coordinates); (b) difference between the area of the image patch and the tracked window; (c) common mode forward velocity control; (d) differential angular velocity control.

arise when the target moves laterally. In this case, since the blimp does not have lateral degrees of freedom, the only solution is to compensate this error by rotating the blimp. Then, as the rotation is not performed around the camera optical axis, it induces a translation motion in the image plane, which will generate errors for the forward motion control. This fact explains that the forward motion control, T_{cmn} in Figure 6.2, shows an oscillatory behavior near to its reference value (zero).

6.3 Station keeping and docking with an underwater ROV

We have performed numerous tests with the proposed approaches to control an underwater vehicle. The experiments described in the following sections correspond to the various control strategies and show the advantage of using the camera degrees of freedom in addition to those of the blimp.

6.3.1 Point-to-point positioning

In Figure 6.3, the results of a station keeping test with the ROV at open sea are illustrated. In this case, point-to-point station keeping was performed, controlling the centroid of the tracked window to the image center by image based visual servoing. The altitude in this case is automatically controlled by the ROV auto-depth function (water pressure).

Figure 6.3(a), shows the window selected initially. It has a positional offset in the coordinates of the centroid. The crosses in the images indicate the image center. Figure 6.3(b) shows the tracked window after 100 frames, from which it can be observed that the centroid gets closer to the image center. At open sea, the main perturbations arise due to current streams of waves. From Figure 6.3(d) and (e) it

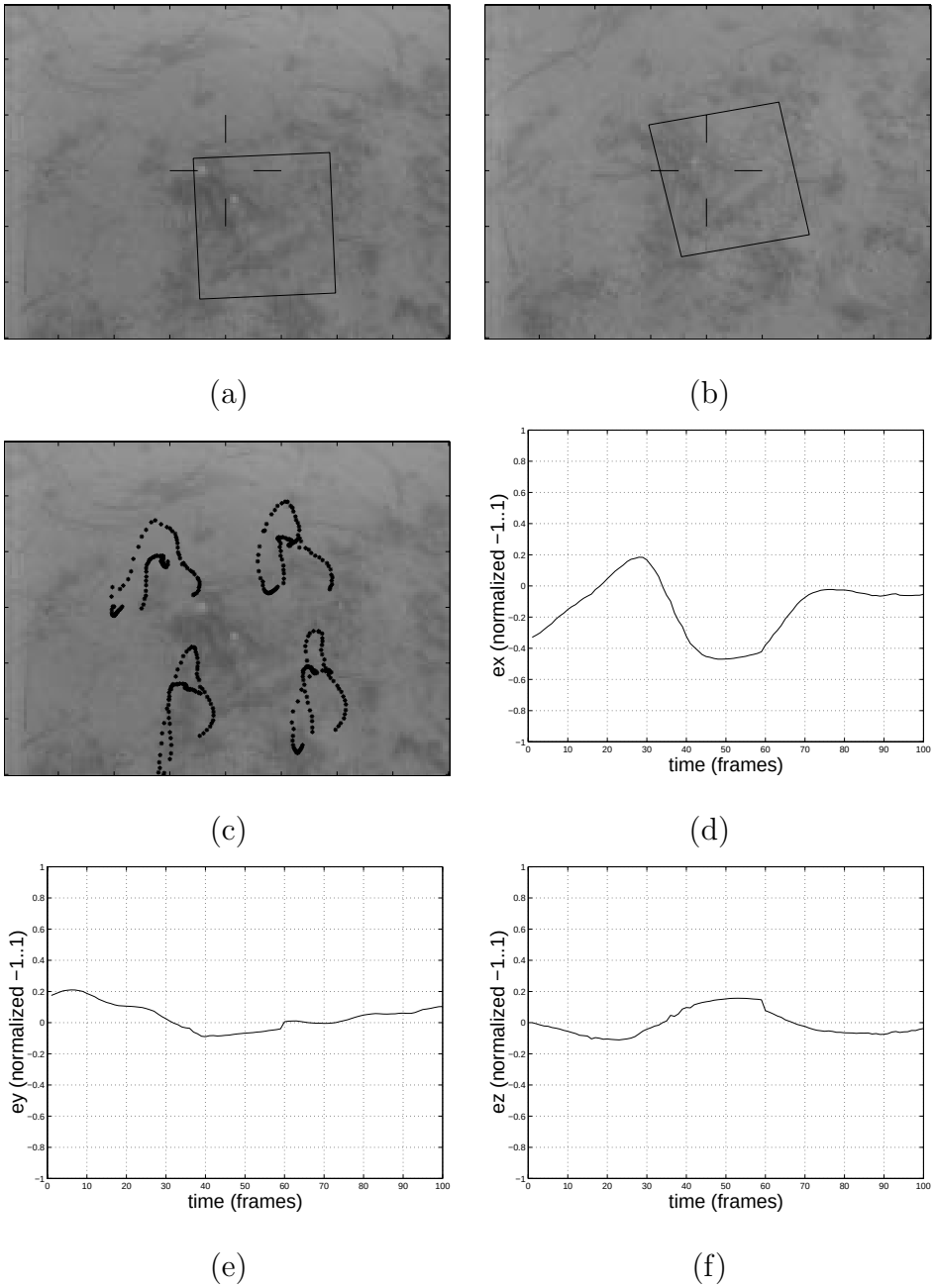


Figure 6.3: Point-to-point positioning with ROV at open sea; (a) initially selected landmark; (b) landmark position in image at the end of the experiment; (c) trajectory of the window corners during the trial; (d) positional offset along the image x-axis; (e) positional offset along the image y-axis; (f) error in window area.

follows that at frame 30, the positional offset of the centroid is almost zero. A few moments after, a strong current applied to the vehicle causes positional errors both in the horizontal and vertical plane. The evolution of the error functions shows these kind of periodical distortions, whose oscillatory shape is thus explained by underwater currents due to waves.

The trajectory of the corners of the tracked window is plotted in Figure 6.3(c). Also note the poor visibility and little texture available in images of underwater scenes. In spite of these difficult conditions, the tracking system performed robustly.

6.3.2 Image based servoing for position, orientation and altitude

In this experiment, the ROV position, orientation and altitude is controlled based upon visual information only. The *Station Keeping* test consist of a time span of 300 frames, corresponding to approximately 40 seconds. Figure 6.4 illustrates some results. It shows the position of the tracked window upon initialization, as well as the final position at the end of the task. Figure 6.4(c) shows that, during the station keeping maneuvers, the landmark partially becomes out of sight (at t=60 frames), indicating either slow dynamics or strong currents. Nevertheless, an estimate of the current window position is computed, based upon optical flow information, so that the controller is able to drive the landmark back to the center. The same situation is repeated at t=230.

In this experiment, the ROV altitude is also controlled based on image features only. In Figure 6.4(f), the temporal evolution of the window area is plotted, which indirectly represents the altitude. The ROV is maintained at a constant altitude during the maneuvers.

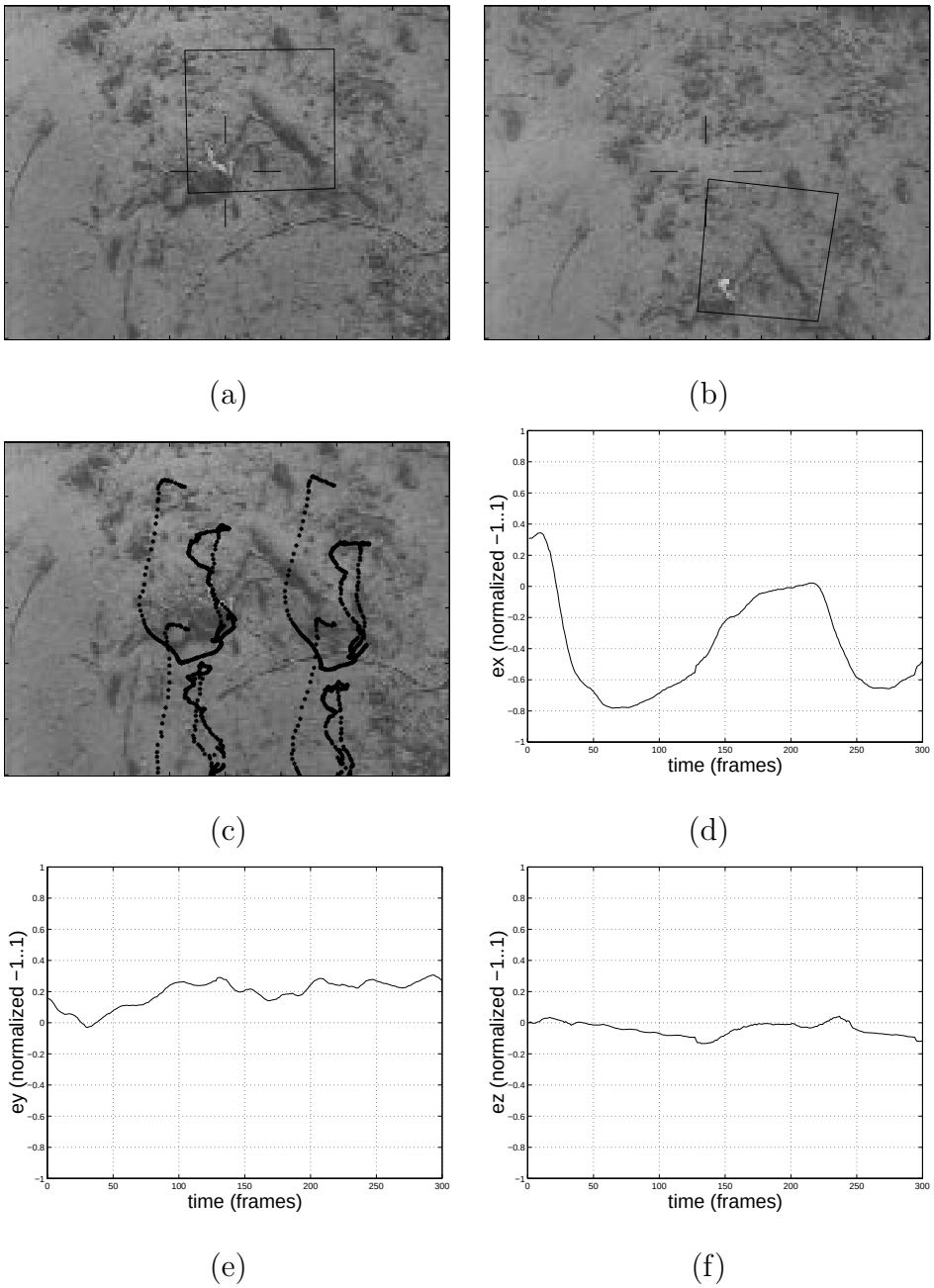


Figure 6.4: Image based control for position, orientation and altitude of the ROV; (a) initial selected landmark; (b) landmark at the end of the recording phase; (c) temporal trajectory of the window corner coordinates; (d) positional offset along the image x-axis; (e) positional offset along the image y-axis; (f) error in window area.

6.3.3 Image stabilization with the on-board camera

In this experiment, only the on-board camera pan and tilt angles are controlled, whereas the vehicle degrees of freedom are not used at all. This demonstrates two aspects. First it shows the capability of centering the landmark in the image center by visually servoing the pan and tilt motors and second, since the ROV is floating uncontrolled, the experiment shows the effect of external disturbances due to waves and underwater currents.

From Figure 6.5(a) it follows that the initial selected window is near to the image center. Since the ROV is not controlled, a drift of this window in the image plane is observed. The resulting positional offset is compensated with the camera pan and tilt angle, maintaining the centroid near to the image center. However, at time $t = 140$, the tilt angle reaches its limit resulting into a corresponding increase in offset along the image x-axis. Note from Figure 6.5(b) that at the last recorded frame, the landmark is partially out of sight, introducing errors in the tracking performance (since optical flow information is integrated).

6.3.4 Image based servoing with image stabilization

In this experiment, the ROV position, orientation and altitude is controlled together with the camera pan and tilt angle so as to realize some image stabilization.

In Figure 6.6 the temporal evolution of the error signals are plotted together with the initial and final view of the camera. Figure 6.6(c) shows that the the addition of image stabilization improves the ability to reject external disturbances, during the trial. The positional offset of the tracked window corners has a much smaller amplitude then in the previous tests. From Figure 6.6(g) it can be noted that the ROV is under closed loop control, since the pan angle at the end of the experiment tends to zero. This means that a lateral offset first is compensated by increasing the pan angle and afterwards regulated back to zero by controlling the vehicle.

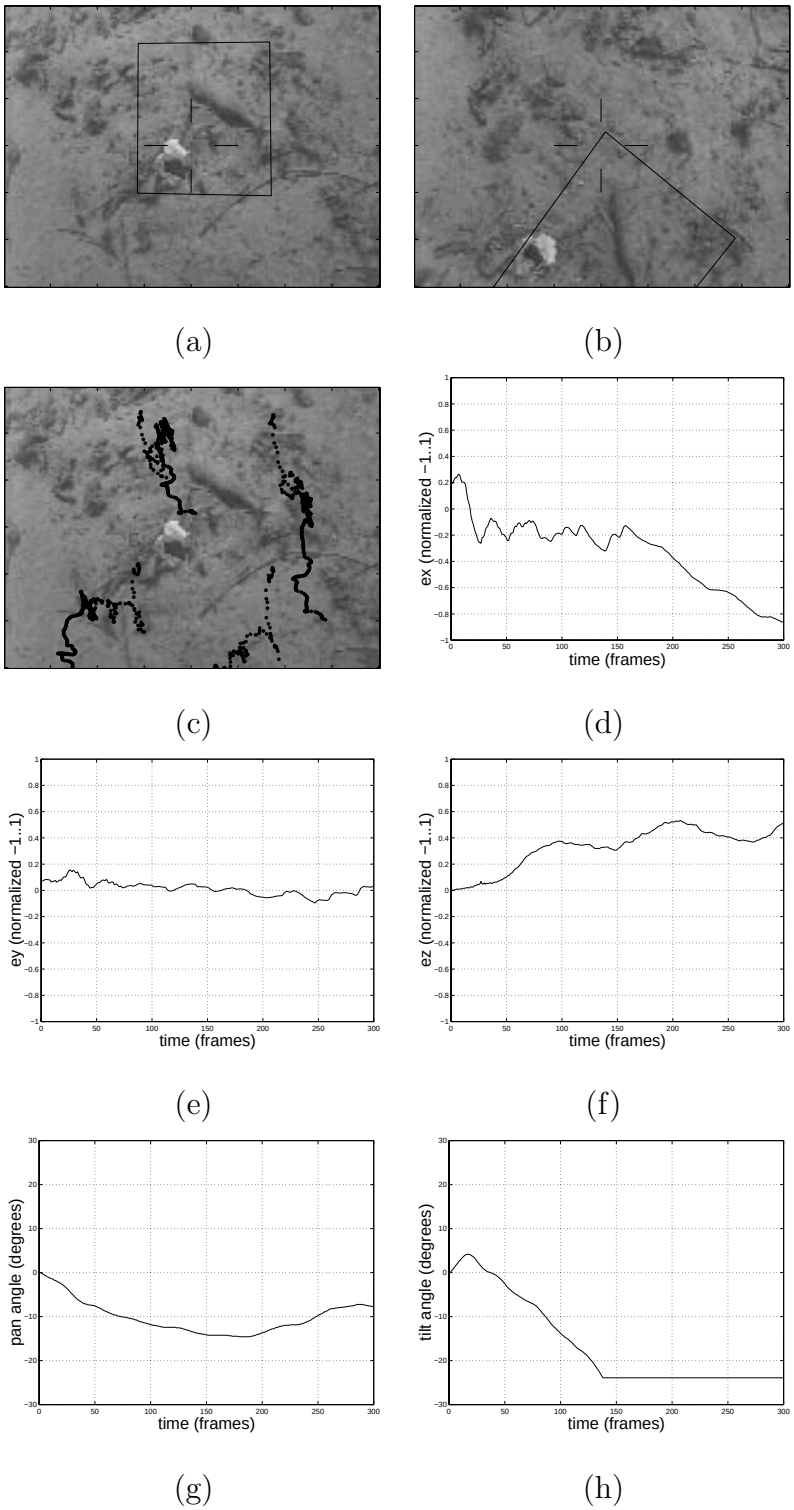


Figure 6.5: Image stabilization with pan and tilt unit for the ROV onboard camera at open sea; (a) initial selected landmark; (b) landmark at the end of the recording phase; (c) temporal trajectory of the window corner coordinates; (d) positional offset along the image x-axis; (e) positional offset along the image y-axis; (f) error in window area; (g) temporal evolution of the pan angle; (h) temporal evolution of the tilt angle.

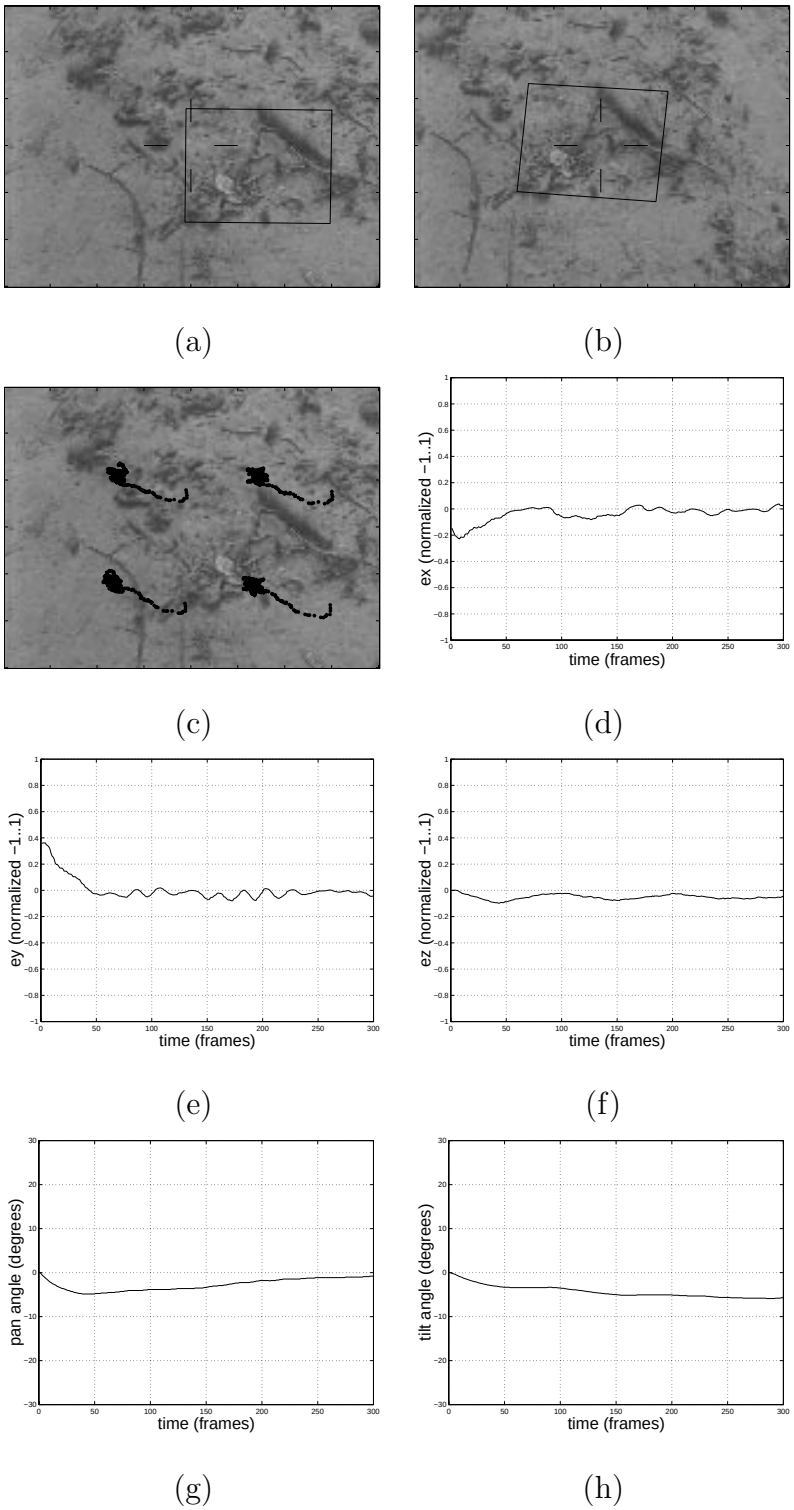


Figure 6.6: Image based control for position, orientation and altitude of ROV at open sea, with image stabilization; (a) initial selected landmark; (b) landmark at the end of the recording phase; (c) temporal trajectory of the window corner coordinates; (d) positional offset along the image x-axis; (e) positional offset along the image y-axis; (f) error in window area; (g) temporal evolution of the pan angle; (h) temporal evolution of the tilt angle.

Chapter 7

Conclusions

In this thesis we explored the use of vision to perform station keeping and docking with floating robots, such as blimps and underwater vehicles. From the visual information, the robots were able to self-localize themselves relative to a visual landmark in an unstructured scene. In other cases, the visual information is directly used for control purposes. The only assumption made is that the landmark is locally planar.

The deformations of the selected image regions were tracked by integrating optic flow information, to predict the approximate deformation parameters, with a correlation based optimization method to accurately register the current view with the reference image. This method uses a set of motion sampling vectors that sample the search space for expected image deformations. Planar projective motion models were considered that cover the whole range of image deformations that occur when a camera floats in 3D space. Using the motion vectors, we were able to calculate a set of difference templates that contribute to the solution of the image registration problem. An interesting feature of the tracker is that it allows the off-line calculation of these templates, increasing the tracker frequency significantly. This makes the method extremely useful for real-time tracking applications. The tracking frequency established was 12 Hz for images with a 128×192 pixel-size and using a 450Mhz processor.

Also a method was proposed for substituting motion vectors by substituting dif-

ference templates in the pre-calculated database. This methodology increased the range over which image deformations can be estimated, while preserving a low computational effort.

The advantages of integrating optic flow computation were twofold: (i) it enhanced the robustness of the tracking algorithm in the sense that larger image deformations could be tracked and (ii) it provides a means of keeping track of image motion in those situations where the visual landmark becomes out of sight.

A high tracking performance was obtained with sub-pixel accuracy. Even in the case of underwater images, were little texture and non-uniform lighting are common problems, the tracking algorithm performed well. This allowed to reconstruct the camera trajectory based on homography decomposition. With the reconstructed camera trajectory, the station keeping task can be formulated in 3D space. This however, lead to unsatisfactory results when tested with the real robots. The main reason was that translations in the image plane can be accounted for with either camera translations or small pan and tilt rotations. Also the non-holonomic constraints of the vehicles required a non-linear time varying feedback control law. This design is based on kinematic modeling of the vehicles and did not apply well to the real non-linear dynamic systems.

Station keeping based on measurements in the image plane performed well on both robots, although the control laws were also derived based on kinematic relationships between the desired camera motion and the vehicles control inputs. From the experiments we conclude that the image based visual servoing approach is most appropriate for implementing station keeping and docking behaviors in underwater robots. The main motivation behind this conclusion is that the image based approach does not depend on a reconstruction phase and that image features can be tracked accurately within the image plane.

The introduction of an image stabilization technique added more robustness to

the visual servoing strategies because image features remain approximately centered in the image plane. This overcomes to some extent the non-holonomic constraints of the vehicles. However, the implementation required a transformation on the error functions in the image based approach.

Future work points out into various directions. In the design of control laws, camera intrinsic and extrinsic parameters were assumed to be known. In practice this is not always realistic since camera calibration is not trivial. The effect of calibration errors therefore should be studied in the sense of stability/sensitivity of the various proposed control laws.

Upon deriving the various image based control laws for station keeping, the proposed solutions include an inverse of the corresponding image jacobian, calculated at the reference values. This only guarantees exponential convergence of the error functions in a small neighborhood of the desired camera position and orientation. A better understanding of the structure of the various image jacobians allows to identify singularities in the error function space. This can be helpful for designing more robust strategies for docking, were larger error values are common.

Also the vehicles non-linear and highly coupled dynamic behavior should be brought into the image plane upon designing the various control laws.

Finally, we point at the integration of the station keeping behavior with other navigation behaviors, such as image-mosaic based navigation [11]. Other examples are contour following and obstacle avoidance.

Appendix A

Motion field and Optic Flow

In this appendix, the concepts of *motion field* and *optic flow* are introduced. For literature review see [15].

Motion field. The motion field associates a velocity vector to each point in the image. Consider a point P, observed by a camera that is rotating and translating with linear and angular velocities $\mathbf{t} = [t_x, t_y, t_z]^\top$ and $\omega = [\omega_x, \omega_y, \omega_z]^\top$. The velocity of P relative to the camera, expressed in the camera reference frame is given by:

$$\frac{dP}{dt} = -\mathbf{t} - \omega \times P \tag{A.1}$$

The motion projected on the image plane can be obtained by considering the pinhole camera model with normalized coordinates [7]. In this model, a point $P = (X, Y, Z)$ in space is mapped to the point $p = (x, y)$ in the image plane with coordinates:

$$x = \frac{X}{Z}, \quad y = \frac{Y}{Z} \tag{A.2}$$

We can now determine $(\dot{x}, \dot{y})$, the derivative of the coordinates of p with respect to time, obtaining :

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2}, \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} \tag{A.3}$$

Using equation(A.3) together with equation (A.2) one obtains:

$$\dot{X} = \frac{\dot{x}Z^2 + xZ\dot{Z}}{Z}, \dot{Y} = \frac{\dot{y}Z^2 - yZ\dot{Z}}{Z} \tag{A.4}$$

Also, from equations (A.1) and (A.2), we have:

$$\dot{Z} = -t_z - (\omega_x y - \omega_y x)Z \tag{A.5}$$

Substituting the equations (A.4) and (A.5 into equation (A.3), we finally obtain:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -1 - x^2 & y \\ 1 + y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{A.6}$$

This equation relates the velocity of an image point p, projection of a 3D point, P, induced by the 3D motion of the camera, and it is known as the motion field.

Optic flow. Brightness patterns in the image move as the objects that give rise to them move relative to the camera. The optic flow is defined as the apparent motion of these brightness patterns. Ideally, the optic flow will correspond to the motion field, which means that the observed flow would be a measurement the motion field. However, this is not always true (see [15]).

Consider $E(x, y, t)$ to be the image irradiance at time t of the image point (x, y). Let $u(x, y)$ and $v(x, y)$ be the components of the optic flow along the x- and y-axis of the image coordinate system. The assumption of image brightness constancy is written as:

$$E(x, y, t) = E(x + u\delta t, y + v\delta t, t + \delta t) \tag{A.7}$$

Also assuming that the image irradiance varies slowly over time, a Taylor series expansion of the right hand side of the previous equation yields:

$$E(x, y, t) = E(x, y, t) + \frac{\partial E}{\partial x}\delta x + \frac{\partial E}{\partial y}\delta y + \frac{\partial E}{\partial t}\delta t + e \tag{A.8}$$

where e represents the higher order terms of the expansion. Canceling $E(x, y, t)$ on both sides, dividing trough δt and taking the limit $\delta t \to 0$, one obtains:

$$E_x u + E_y v + E_t = 0, \tag{A.9}$$

where we have used

$$(u, v) = (\frac{dx}{dt}, \frac{dy}{dt})$$

$$(E_x, E_y, E_t) = (\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial t})$$

This equation is known as the *optical flow constraint equation*, and can be re-written as the inner product between the image gradient (E_x, E_y) and the optical flow (u, v) as follows:

$$(E_x, E_y) \cdot (u, v) = -E_t \tag{A.10}$$

At each point, the optical flow constraint defines a single equation for the two unknown components of the flow. Hence, the optic flow cannot be completely determined from local measurements of E_x, E_y and E_t at pixel (x, y), which is known as the *aperture problem*. It is only possible to recover the component of the optic flow in the direction of the brightness pattern gradient (E_x, E_y), which is usually referred to as the *normal flow*. The normal flow, $(u, v)_\perp$, can be determined at each image point as:

$$(u, v)_\perp = -\frac{E_t}{\sqrt{E_x^2 + E_y^2}} (E_x, E_y) \tag{A.11}$$

Appendix B

Motion sampling sets

In this appendix, the translational, affine and planar projective motion sampling vectors, as used in the implementation of the tracking algorithm, are listed. For each set $\{\mathbf{\Delta q}_i : i \in (1 \ldots 48)\}_k, k \in 1, 2, 3$, the sampling vectors that form a basis are shown, sampling the search space for expected image deformations.

The adaptive parts of each set (used to substitute the last observed motion in the databases) are not listed since they are dynamic. Upon initialization, these additional motion sampling vectors are randomly chosen.

The positional offsets of the four corners of the ROI is used to define the various motion models, according to the generic structure of $\Delta\mathbf{q}_i = \{x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4\}$, where (x_1, y_1) are the top left corner coordinates, (x_2, y_2) are the top right corner coordinates, (x_3, y_4) are the bottom right corner coordinates and (x_4, y_4) are the bottom left corner coordinates of the ROI.

Translational sampling vectors:

number	sampling vector	number	sampling vector
1	{3, 3, 3, 3, 3, 3, 3, 3}	25	{1, 6, 1, 6, 1, 6, 1, 6}
2	{1, 3, 1, 3, 1, 3, 1, 3}	26	{0, 6, 0, 6, 0, 6, 0, 6}
3	{0, 3, 0, 3, 0, 3, 0, 3}	27	{-1, 6, -1, 6, -1, 6, -1, 6}
4	{-1, 3, -1, 3, -1, 3, -1, 3}	28	{6, 1, 6, 1, 6, 1, 6, 1}
5	{-3, 3, -3, 3, -3, 3, -3, 3}	29	{-6, 1, -6, 1, -6, 1, -6, 1}
6	{3, 1, 3, 1, 3, 1, 3, 1}	30	{6, 0, 6, 0, 6, 0, 6, 0}
7	{1, 1, 1, 1, 1, 1, 1, 1}	31	{-6, 0, -6, 0, -6, 0, -6, 0}
8	{0, 1, 0, 1, 0, 1, 0, 1}	32	{1, -6, 1, -6, 1, -6, 1, -6}
9	{-1, 1, -1, 1, -1, 1, -1, 1}	33	{0, -6, 0, -6, 0, -6, 0, -6}
10	{-3, 1, -3, 1, -3, 1, -3, 1}	34	{-1, -6, -1, -6, -1, -6, -1, -6}
11	{3, 0, 3, 0, 3, 0, 3, 0}	35	{6, -1, 6, -1, 6, -1, 6, -1}
12	{1, 0, 1, 0, 1, 0, 1, 0}	36	{-6, -1, -6, -1, -6, -1, -6, -1}
13	{-1, 0, -1, 0, -1, 0, -1, 0}	37	{6, 6, 6, 6, 6, 6, 6, 6}
14	{-3, 0, -3, 0, -3, 0, -3, 0}	38	{3, 6, 3, 6, 3, 6, 3, 6}
15	{3, -3, 3, -3, 3, -3, 3, -3}	39	{-3, 6, -3, 6, -3, 6, -3, 6}
16	{1, -3, 1, -3, 1, -3, 1, -3}	40	{-6, 6, -6, 6, -6, 6, -6, 6}
17	{0, -3, 0, -3, 0, -3, 0, -3}	41	{6, 3, 6, 3, 6, 3, 6, 3}
18	{-1, -3, -1, -3, -1, -3, -1, -3}	42	{-6, 3, -6, 3, -6, 3, -6, 3}
19	{-3, -3, -3, -3, -3, -3, -3, -3}	43	{6, -6, 6, -6, 6, -6, 6, -6}
20	{3, -1, 3, -1, 3, -1, 3, -1}	44	{3, -6, 3, -6, 3, -6, 3, -6}
21	{1, -1, 1, -1, 1, -1, 1, -1}	45	{-3, -6, -3, -6, -3, -6, -3, -6}
22	{0, -1, 0, -1, 0, -1, 0, -1}	46	{-6, -6, -6, -6, -6, -6, -6, -6}
23	{-1, -1, -1, -1, -1, -1, -1, -1}	47	{6, -3, 6, -3, 6, -3, 6, -3}
24	{-3, -1, -3, -1, -3, -1, -3, -1}	48	{-6, -3, -6, -3, -6, -3, -6, -3}

Affine sampling vectors:

number	sampling vector	number	sampling vector
1	{-3, 0, -3, 0, 3, 0, 3, 0}	25	{0, -3, 0, -3, 0, 0, 0, 0}
2	{-1, 0, -1, 0, 1, 0, 1, 0}	26	{0, -1, 0, -1, 0, 0, 0, 0}
3	{1, 0, 1, 0, -1, 0, -1, 0}	27	{0, 1, 0, 1, 0, 0, 0, 0}
4	{3, 0, 3, 0, -3, 0, -3, 0}	28	{0, 3, 0, 3, 0, 0, 0, 0}
5	{0, -3, 0, 3, 0, 3, 0, -3}	29	{0, 0, 0, 0, 0, -3, 0, -3}
6	{0, -1, 0, 1, 0, 1, 0, -1}	30	{0, 0, 0, 0, 0, -1, 0, -1}
7	{0, 1, 0, -1, 0, -1, 0, 1}	31	{0, 0, 0, 0, 0, 1, 0, 1}
8	{0, 3, 0, -3, 0, -3, 0, 3}	32	{0, 0, 0, 0, 0, 3, 0, 3}
9	{-3, 0, 3, 0, 3, 0, -3, 0}	33	{-6, 0, -6, 0, 6, 0, 6, 0}
10	{-1, 0, 1, 0, 1, 0, -1, 0}	34	{6, 0, 6, 0, -6, 0, -6, 0}
11	{1, 0, -1, 0, -1, 0, 1, 0}	35	{0, -6, 0, 6, 0, 6, 0, -6}
12	{3, 0, -3, 0, -3, 0, 3, 0}	36	{0, 6, 0, -6, 0, -6, 0, 6}
13	{0, -3, 0, -3, 0, 3, 0, 3}	37	{-6, 0, 6, 0, 6, 0, -6, 0}
14	{0, -1, 0, -1, 0, 1, 0, 1}	38	{6, 0, -6, 0, -6, 0, 6, 0}
15	{0, 1, 0, 1, 0, -1, 0, -1}	39	{0,-6, 0,-6, 0, 6, 0, 6}
16	{0, 3, 0, 3, 0, -3, 0, -3}	40	{0, 6, 0, 6, 0, -6, 0, -6}
17	{-3, 0, 0, 0, 0, 0, -3, 0}	41	{-6, 0, 0, 0, 0, 0, -6, 0}
18	{-1, 0, 0, 0, 0, 0, -1, 0}	42	{6, 0, 0, 0, 0, 0, 6, 0}
19	{1, 0, 0, 0, 0, 0, 1, 0}	43	{0, 0, -6, 0, -6, 0, 0, 0}
20	{3, 0, 0, 0, 0, 0, 3, 0}	44	{0, 0, 6, 0, 6, 0, 0, 0}
21	{0, 0, -3, 0, -3, 0, 0, 0}	45	{0, -6, 0, -6, 0, 0, 0, 0}
22	{0, 0, -1, 0, -1, 0, 0, 0}	46	{0, 6, 0, 6, 0, 0, 0, 0}
23	{0, 0, 1, 0, 1, 0, 0, 0}	47	{0, 0, 0, 0, 0, -6, 0, -6}
24	{0, 0, 3, 0, 3, 0, 0, 0}	48	{0, 0, 0, 0, 0, 6, 0, 6}

Planar projective sampling vectors:

number	sampling vector	number	sampling vector
1	{1, 0, 0, 0, 0, 0, 0, 0}	25	{-3, 0, 0, 0, 0, 0, 0, 0}
2	{0, 1, 0, 0, 0, 0, 0, 0}	26	{0, -3, 0, 0, 0, 0, 0, 0}
3	{0, 0, 1, 0, 0, 0, 0 ,0}	27	{0, 0, -3, 0, 0, 0, 0, 0}
4	{0, 0, 0, 1, 0, 0, 0, 0}	28	{0, 0, 0, -3, 0, 0, 0, 0}
5	{0, 0, 0, 0, 1, 0, 0, 0}	29	{0, 0, 0, 0, -3, 0, 0, 0}
6	{0, 0, 0, 0, 0, 1, 0, 0}	30	{0, 0, 0, 0, 0, -3, 0, 0}
7	{0, 0, 0, 0, 0, 0, 1, 0}	31	{0, 0, 0, 0, 0, 0, -3, 0}
8	{0, 0, 0, 0, 0, 0, 0, 1}	32	{0, 0, 0, 0, 0, 0, 0, -3}
9	{3, 0, 0, 0, 0, 0, 0, 0}	33	{-6, 0, 0, 0, 0, 0, 0, 0}
10	{0, 3, 0, 0, 0, 0, 0, 0}	34	{0, -6, 0, 0, 0, 0, 0, 0}
11	{0, 0, 3, 0, 0, 0, 0, 0}	35	{0, 0, -6, 0, 0, 0, 0, 0}
12	{0, 0, 0, 3, 0, 0, 0, 0}	36	{0, 0, 0, -6, 0, 0, 0, 0}
13	{0, 0, 0, 0, 3, 0, 0, 0}	37	{0, 0, 0, 0, -6, 0, 0, 0}
14	{0, 0, 0, 0, 0, 3, 0, 0}	38	{0, 0, 0, 0, 0, -6, 0, 0}
15	{0, 0, 0, 0, 0, 0, 3, 0}	39	{0, 0, 0, 0, 0, 0, -6, 0}
16	{0, 0, 0, 0, 0, 0, 0, 3}	40	{0, 0, 0, 0, 0, 0, 0, -6}
17	{-1, 0, 0, 0, 0, 0, 0, 0}	41	{6, 0, 0, 0, 0, 0, 0, 0}
18	{0, -1, 0, 0, 0, 0, 0, 0}	42	{0, 6, 0, 0, 0, 0, 0, 0}
19	{0, -1, 0, 0, 0, 0, 0, 0}	43	{0, 0, 6, 0, 0, 0, 0, 0}
20	{0, 0, 0, -1, 0, 0, 0, 0}	44	{0, 0, 0, 6, 0, 0, 0, 0}
21	{0, 0, 0, 0, -1, 0, 0, 0}	45	{0, 0, 0, 0, 6, 0, 0, 0}
22	{0, 0, 0, 0, 0, -1, 0, 0}	46	{0, 0, 0, 0, 0, 6, 0, 0}
23	{0, 0, 0, 0, 0, 0, -1, 0}	47	{0, 0, 0, 0, 0, 0, 6, 0}
24	{0, 0, 0, 0, 0, 0, 0, -1}	48	{0, 0, 0, 0, 0, 0, 0, 6}

Bibliography

[1] A. Bernardino, J. Santos-Victor, and G. Sandini. Tracking planar structures with log-polar images. In *Proc. of the 8th International Symposium on Intelligent Robotic Systems - SIRS2000*, University of Reading, England, July 2000.

[2] G. Conte and A. Serrani. Robust nonlinear motion control for auvs. *IEEE Robotics and Automation Magazine*, 1999.

[3] J. I. Craig. *Introduction to Robotics.* Addison Wesley, 1986.

[4] C.Canudas de Wit, B. Siciliano, and G. Bastin (Eds). *Theory of Robot Control.* Springer-Verlag, 1996.

[5] A. Elfes, S. Sigueira Bueno, M. Bergerman, and J.Jr.G Ramos. A semi-autonomous robotic airship for environmental monitoring missions. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, Leuven, Belgium, May 1998.

[6] B. Espiau, F. Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[7] O. Faugeras. *Three-Dimensional Computer Vision- A Geometric Viewpoint.* MIT Press, 1993.

[8] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. John-Wiley and Sons, 1995.

[9] M. Gleicher. Projective registration with difference decomposition. In *Proc. of the IEEE Conf. of Computer Vision and Pattern Recognition*, June 1997.

[10] S.B. Varella Gomes and J.Jr.G. Ramos. Airship dynamic modeling for autonomous operation. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, Leuven, Belgium, May 1998.

[11] Nuno Gracias and José Santos-Victor. Underwater video mosaics as visual navigation maps. *Computer Vision and Image Understanding*, 79(1):66–91, July 2000.

[12] G. Hager and P-Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *IEEE Conf. of Computer Vision and Pattern Recognition*, 1996.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge Universtity Press, 2000.

[14] P. Heckbert. Fundamentals of texture mapping and image warping. In *Masters Thesis*, U.C. Berkeley, 1989.

[15] B. Horn. *Robot Vision*. MIT Press, 1986.

[16] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[17] G. A. Khoury and J.D. Gillet. *Airship Technology*. Cambridge University Press, 1999.

[18] J. Kosecka. Visually guided navigation. *Robotics and Autonomous Systems*, 21(1):37–50, July 1997.

[19] J.F. Lots, D.M. Lane, and E. Trucco. Application of 2 1/2 d visual servoing to underwater vehicle station keeping. In *Proc. of the IEEE Oceans Conference*, Providence, USA, September 2000.

[20] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 - d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, April 1999.

[21] S. Mann and R. Picard. Virtual bellows: Constructing high quality stills from video. In *Proc. of the IEEE Conf. on Image Processing*, November 1994.

[22] R.L. Marks, H.H. Wang, M. J. Lee, and S. M. Rock. Automatic visual station keeping of an underwater robot. In *Proc. of the IEEE Oceans Conference*, Brest, France, September 1994.

[23] S. Negahdaripour and S. Lee. Motion recovery from image sequences using only first order optical flow information. *International Journal of Computer Vision*, (9(3)):163–184, 1992.

[24] J. S. Riedel. Shallow water stationkeeping of an autonomous underwater vehicle: The experimental results of a disturbance compensation controller. In *Proc. of the IEEE Oceans Conference*, Providence, USA, September 2000.

[25] P. Rives and J. Borrelly. Visual servoing techniques applied to an underwater vehicle. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, Albuquerque, New Mexico, April 1997.

[26] J. Santos-Victor and G. Sandini. Visual behaviors for docking. *Computer Vision and Image Understanding*, 67(3):223–238, September 1997.

[27] S.Hutchinson, G.D.Hager, and P.I.Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5), 1996.

[28] M. Subbarao and A. Waxman. Closed form solutions to image flow equations for planar surfaces in motion. *Computer Vision Graphics and Image Processing*, 36, 1986.

[29] R. Szeliski. Image mosaicing for tele-reality applications. In *Proc. of the IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, 1994.

[30] S. van der Zwaan, A. Bernardino, and J. Santos-Victor. Vision based station keeping and docking for an aerial blimp. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2000)*, Kagawa University, Takamatsu, Japan, October 31 - November 5 2000.

[31] S. van der Zwaan, M. Perrone, A. Bernardino, and J. Santos-Victor. Control of an aerial blimp based on visual input. In *Proc. of the 8th International Symposium on Intelligent Robotic Systems - SIRS2000*, University of Reading, England, July 2000.

[32] H. Zhang and J.P.Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. In *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, USA, May 1999.