Contributions to dense visual tracking and visual servoing using robust similarity criteria

Bertrand Delabarre

2014

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Bertrand Delabarre

préparée à l'unité de recherche 6074 - IRISA
Institut de Recherche en Informatique et Systèmes Aléatoire
Université de Rennes 1

Contributions to dense visual tracking and visual servoing using robust similarity criteria

Thèse soutenue à Rennes le 23/12/2014

devant le jury composé de :

Patrick BOUTHEMY
Directeur de recherche Inria / président de jury

Thierry CHATEAU
Professeur, Université Blaise Pascal / rapporteur

Christophe DOIGNON
Professeur, Université de Strasbourg / rapporteur

Hideaki UCHIYAMA
Assistant Professor, Kyushu University / examinateur

Éric MARCHAND
Professeur, Université Rennes 1 / directeur de thèse

Remerciements

A mon jury

Je tiens tout d'abord à remercier les membres de mon jury de thèse. Patrick Bouthemy, qui m'a fait l'honneur d'accepter de le présider, Thierry Chateau et Christophe Doignon qui ont accepté d'apporter leur expertise dans le domaine pour rapporter sur mon manuscrit ainsi que Hideaki Uchiyama qui a accepté de prolonger son séjour en France pour faire partie de ce jury. Je les remercie pour l'attention portée au manuscrit et les précieuses remarques présentes dans les rapports ainsi que pour le fait d'avoir accepté de se déplacer pour participer à un jury de thèse un 23 décembre. Je les remercie également pour leurs remarques et questions qui m'ont permis d'améliorer ce document ainsi que d'étendre ma curiosité vers de nouvelles possibilités apportées par mes travaux.
Je tiens également à remercier chaleureusement mon directeur de thèse Eric Marchand. Il m'a permis, durant ces trois années de doctorat, d'évoluer avec un parfait mélange d'autonomie et de conseils sur les directions à prendre. Merci pour tous ces conseils ainsi que toutes ces relectures et corrections, que ce soit lors de soumissions d'articles ou de ce manuscrit. En plus de tout cela, je tiens à le remercier pour les opportunités qu'il m'a donné, notamment lors de mes différents stages au sein de l'équipe.

A ma famille

Je tiens à remercier ma famille pour son aide et ses encouragements durant ces trois années. Mes parents tout d'abord, sans qui je n'aurais pu être là (littéralement étant donné qu'il m'ont ammené et ramené un nombre incalculable de fois avant que je me décide à passer mon permis). Ils m'ont également permis de vivre dans un environnement parfait durant toutes mes études, ce qui a largement contribué aux résultats obtenus. Je remercie aussi mon frère pour son soutien tout au long de mes études. Ces trois personnes ayant eu à réexpliquer ce qu'est une thèse à tout le reste de ma famille durant trois ans je veux aussi les féliciter. Enfin, j'ai une pensée pour mon oncle et ma tante qui ont suivi avec attention mes travaux durant ces trois années.

A mes collègues et amis

J'ai eu la chance de pouvoir effectuer ces travaux de recherche au sein de l'équipe Lagadic. Je tiens dès lors à remercier François Chaumette, directeur de cette equipe dans laquelle il a su instaurer une ambiance de travail à la fois très motivante et très chaleureuse. Parmis les nombreux collègues que j'ai cotoyé durant ces quatres années, je tiens tout d'abord à remercier ceux qui ont eu ce privilège doublé de malédiction qu'est le partage d'un bureau

avec moi. Antoine tout d'abord, de nos années de stage à la fin de son doctorat, dont la bonne humeur durant ces heures passées à débattre d'un nombre impressionant de sports différents m'a toujours impressionné. Rafik ensuite, qui a fait reigner une bonne ambiance dans le bureau en toutes circonstances. Enfin Aly et Manikandan qui ont réussi à survivre ma période de rédaction et soutenance, ce qui n'était pas forcément joué d'avance. Un grand merci aussi aux différents permanents de l'équipe. Alexandre et sa bonne humeur face à la défaite au ping pong, Marie et sa bonne humeur contagieuse, Fabien dont l'aide a été précieuse pour pouvoir réaliser tous ces programmes expérimentaux ainsi qu'à Paolo et Vincent que j'ai moins eu le temps de cotoyer. Les assistantes aussi, à commencer par Céline qui m'a aidé à de nombreuses reprises pour toutes ces démarches avec une efficacité hors-pair ainsi que Hélène, dont l'aide pour la soutenance m'a permis de me concentrer à cent pourcent sur la thèse. Les collègues se sont succédés avec toujours la même ambiance au sein de l'équipe, des anciens temps avec Laurent, Olivier, Caroline, Céline ou François Chapeau aux tout nouveaux comme Pedro, Lesley-Ann, Jason, Thomas ou Fabrizio. Merci aussi à ceux avec que j'ai croisé pendant plusieurs années comme Giovanni, Suman, Nicolas, Pierre, Le, Riccardo, Lucas ou Souriya. Enfin, un grand merci à mes "Smash Brothers" dont François et ses épeistes, Aurélien et son spacing qui nous a tous fait fondre une durite à un moment ou à un autre, Clément et sa manette gamecube officielle Panasonic, Vishnu et son Falco bottom tier sans oublier Quentin et Noël qui cherchent encore un main. Pour finir, je souhaite remercier les amis qui m'ont soutenu pendant ces trois ans. Les "Sonyistes" Romain et Benjamin d'un côté, les "Haloistes" Yann et Fabien de l'autre.

Contents

"Eyes can only see what the
mind is ready to understand"

Henri Bergson

Introduction

Motion is a very instinctive task for human beings. They can move around with accuracy and simplicity, almost without having to do anything consciously. This localization and navigation tasks are made possible by the perception we have of our surrounding environment, mostly thanks to our vision capabilities. In an age when more and more actions are made through automated systems, the understanding of that process, natural to human beings, is essential to provide any automated systems with autonomy. Autonomy is the key component that can allow systems to adapt to more complex tasks. For example, assembly robots have been used for years in fields such as car assembly, but they can never adapt to any situations, or react without the assistance of a human operator to guide them. They can only repeat what they have been taught to do. Autonomy is then what differentiates a simple automate from an intelligent system. Autonomous vision systems are more and more present in today's society and cover a wide range of applications from the entertainment area with video game cameras that can detect and track body parts to emerging driverless car that can travel hundreds of miles without human intervention passing by household vacuum cleaners that map the rooms they need to clean and design optimal paths.

But autonomy is not easy to achieve when confronted to localizing or moving an automated system. Human autonomy is obtained thanks to a huge amount of *a priori* data. That knowledge is gathered throughout life and used by the brain to interpret the information gathered by the different senses. Human brain is the key element of human autonomy, analyzing all data perceived and known beforehand in order to control the body while respecting its limitations. But since human brain activity is not yet understood completely, the whole process then needs to be modeled from perception to action if it is to be adapted to create intelligent systems. This is why the amount of research in the domain of computer vision is so vast. And the puzzle is far from being completed, judging by the number of new problems and solutions discovered everyday and explained in the huge computer vision literature.

Strategies can be designed in order to create autonomy in intelligent systems. Among those is the use of computer vision that offers a lot of new possibilities across a lot of domains from TV broadcasting to health, surveillance, robot control, spatial applications and many other application fields. Its modeling is also very tied to human behaviour, since vision is a very important part of localization and motion for human beings. Realizing computer vision applications is also made ever easier by the conjunction of the constant lowering costs of video equipment and augmentation of computing capabilities. Even though other approaches that often mimic human or animal senses are very interesting, such as for example the use of ultrasonic or acceleration data, the works presented in this document are only based on vision. The objective of this thesis is then to propose algorithms that use vision in order to provide more autonomy to automated systems. In this context, two areas of study are tackled: visual tracking, which is the process of estimating the displacement of an object in an image sequence, and visual servoing which aims at controlling a robot based on information gathered from one or several cameras.

In order to perform visual tracking and visual servoing tasks, several approaches have been proposed throughout the years. Most of them use features extracted from the images such as keypoints, lines, contours or more complex models and use the geometrical

information they provide in order to estimate the displacement present in the image sequence. Those approaches, referred to as geometrical approaches, do not take into account the whole amount of information provided by the camera and need to rely on algorithmic layers that extract and match those local features. The scope of this thesis is to follow a different road, one that has been growing over the past years which propose to use the whole image information in order to estimate displacement, designing what are referred to as dense or direct approaches. The use of a set of pixel luminance as visual feature then allows to take into account all the information provided by the images. Comparing those global features between reference and current images thanks to similarity or dissimilarity functions then allows to estimate the displacement. But contrary to the geometrical data of the scene, that luminance data is much more subject to perturbations. Taking for example the common case of light variations during the visual tracking or servoing task, the perceived luminance of the scene will change whereas its geometry will not. This is why a crucial step in the elaboration of dense visual tracking and servoing tasks is the choice of the (dis)similarity measure used to compare the global visual feature. Since that measure will determine whether or not the task is being successful, any shortcomings it might have in any given situation can result in failures of the process. This is why it is the key element of any dense visual tracking or servoing approach. Another important factor to consider is the motion model used to estimate the displacement present in the scene. Depending on the degrees of freedom allowed by the chosen model, some motions could be tracked successfully or not. On the other hand, taking into account complex models when dealing with much more basic motions can result in sub-efficient or over-parameterized optimization processes.

In order to avoid those pitfalls, this document details several motion models and (dis)similarity functions that have been used in the literature to perform visual tracking or servoing applications. After analyzing the pros and cons of several similarity functions, we propose to consider the sum of conditional variance and the mutual information in order to design new visual tracking and visual servoing approaches. The sum of conditional variance is chosen for its natural robustness to global variations of the scene and allows to create very simple and efficient solutions that add robustness to the sum of squared differences approaches commonly used. The mutual information is also considered in this document in order to create more complex but more robust applications in cases where large local perturbations occur. Taking advantage of the properties of the homography warp, a robust dense model-based tracking application is proposed and declined in two versions: one based on the sum of conditional variance and the other based on the mutual information. Extending the considered motion models to non-rigid solutions that take into account deformations of the tracked template, a non-rigid approach is also designed using a thin-plate spline warp function. We then take advantage of the duality that exists between visual tracking and visual servoing to propose new visual servoing approaches. Indeed, where visual tracking estimates the parameters of an image projection method, visual servoing follows the same road but actually moves a robot in order to get the desired projection of the image. The use of the sum of conditional variance is later extended to the design of a new dense visual servoing algorithm that is naturally robust to light variations. The mutual information is also considered through a normalized approach in order to define a new normalized mutual information-based visual servoing approach which is then adapted to omnidirectional cameras. Throughout this thesis, our common objective is to design systems that use the whole information they are provided with and to test their capabilities without having to add any algorithmic layers in order to enhance their robustness. Since a lot of strategies already exist that can be added to potentially any system (such as

robust estimation), we aim at showing how the choices that are made when designing the core principle of any visual tracking or servoing can influence its possibilities before adding any robustification step that might not be needed anymore.

Organization of this document

This document is organized as follows. In the first chapter, the basic notions of computer vision necessary to define visual tracking and visual servoing tasks are detailed. The rest of the manuscript is then composed of two principal components.

A first part is dedicated to visual tracking. Chapter two then recalls the current state of the art of visual tracking. Our contributions in the field are then detailed in chapters three and four. Chapter three details a dense robust model-based tracking algorithm and declines it into two versions, the first one based on the sum of conditional variance and the second one on the mutual information. Several experimental validations are then exposed that validate the proposed algorithm in numerous perturbed conditions. Chapter four focuses on non-rigid tracking, where our second contribution in the field of visual tracking, a non-rigid dense visual tracking approach based on the sum of conditional variance or the mutual information, is detailed and tested, again in numerous conditions.

The second part of this thesis is devoted to visual servoing. Once classical methods presented in the literature have been detailed in chapter five, a dense robust visual servoing process based on the sum of conditional variance is detailed and tested on a real six degrees of freedom robot in several different conditions in chapter six. Finally, chapter seven details how to use a normalized version of the mutual information in order to perform visual servoing. The technique is then extended to omnidirectional cameras and several experiments are detailed that validate the approach.

Contributions and publications

This work has led to several publications in both the computer vision and robotic communities. Following the order they appear in throughout this paper, here are the contributions with the corresponding publications:

- Our first contribution is an algorithm that considers the 3D model of an object in order to determine which face of the object is visible and uses that information to track the model. A dense model-based tracking algorithm that optimizes the sum of conditional variance in order to track objects with face appearance and disappearance is then designed [C1]. The method is then extended to the use of the mutual information in order to add robustness to local perturbations [C3]. Both approaches are validated through experimentations in several different perturbed conditions.

- We propose to extend non-rigid dense visual tracking to the use of the sum of conditional variance [C5]. The method is then extended to the mutual information. The resulting algorithms are validated in different perturbed conditions.

Since a strong duality exists between the visual tracking and visual servoing processes, the robust (dis)similarity functions used to propose new visual tracking algorithms are then used to design new visual servoing approaches.

- A new visual servoing approach using the sum of conditional variance is therefore introduced [C2]. Considering this new cost function for visual servoing allows to be

naturally robust to global scene variations, which is exposed through several experimental validations on a real robot with good accuracy results.

- We propose a new visual servoing approach and extend it to the use of an omnidirectional camera [C1]. Based on a normalized version of the mutual information, the application is demonstrated to be robust to global and local perturbations while keeping good accuracy results through several experimental validations on a real robot.

In parallel to these contributions, a UAV absolute localization framework was realized and published in [C4]. The proposed approach performs a visual tracking step to align the images gathered by a UAV with respect to a georeferenced mosaic. Taking advantage of the robustness offered by the mutual information, the process is able to align views taken in different seasons or featuring moving vehicles or pedestrians. As the position of the UAV is known with relation to a georeferenced mosaic, its position in the world frame is known for each new image, which allows to localize it at every new frame. The process has been validated on a real drone and GPS data was compared successfully to the results obtained thanks to a Google Maps mosaic. Since this application is not related in a major way to the rest of the contributions detailed previously, it has not been detailed in this document. Interested reader is refered to [C4].

International conference papers

[C1] B. Delabarre, G. Caron, E. Marchand. – Omnidirectional Visual Servoing using the Normalized Mutual Information. – in *10th IFAC Symposium on Robot Control, Syroco 2012*, Dubrovnik, Croatia, September 2012.

[C2] B. Delabarre, E. Marchand. – Visual Servoing using the Sum of Conditional Variance. – in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'12*, Vilamoura, Portugal, October 2012.

[C3] B. Delabarre, E. Marchand. – Camera Localization using Mutual Information-based Multiplane Tracking. – in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'13*, Tokyo, Japan, November 2013.

[C4] A. Yol, B. Delabarre, A. Dame, J-E. Dartois, E. Marchand. – Vision-based Absolute Localization for Unmanned Aerial Vehicles. – in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'14*, Chicago, USA, September 2014.

[C5] B. Delabarre, E. Marchand. – Dense non-rigid visual tracking with a robust similarity function. – in *IEEE Int. Conf. on Image Processing, ICIP'14*, Paris, France, October 0214.

National conference papers

[CN1] B. Delabarre, E. Marchand. – Suivi multiplans 3D utilisant la somme des variances conditionnelles. – in *Congrès francophone des jeunes chercheurs en vision par ordinateur, ORASIS'13*, Cluny, France, June 2013.

1

Notations and elementary background in computer vision

This chapter introduces different mathematical notations and geometrical tools that are required by computer vision algorithms. begin by explaining how Euclidean geometry allows to represent the world as seen by a camera. Starting from 3D point transformations the whole perspective projection pipeline is detailed in order to see how images are gathered by cameras. Methods that allow to project 3D points in the image plane are then defined for perspective cameras. Finally, different 2D transformations used to model the motion undergone by objects in the image are detailed.

1.1 Geometrical modeling of image formation

To grasp the information that is perceived by a camera, one needs to connect the gathered data to the real world. To that end, a modeling of the projection process is needed. This is done thanks to Euclidean geometry and the transformations that represent the position and displacement of objects in the scene and its projection on the image plane. In this section we therefore recall the basics of Euclidean geometry before describing how the world is perceived through a camera.

1.1.1 Modeling the world

In order to be able to describe the position of an object in the 3D world, the first element needed is a frame. Indeed, the concept of position can only be relevant in our situation with respect to a fixed point in space. This fixed point and the directions in which the coordinates of any point will be parameterized are respectively the origin and the orthogonal axes of a frame. In our works, three frames are commonly referred to: the world frame F_w, the object frame F_o and the camera frame F_c. F_w is an arbitrary frame representing the world, F_o is the object frame and F_c is the camera frame. 3D transformations allow to express points in one frame or the other. Let us for example take the case of an object present in the field of view of the camera. A point of the object ${}^o\mathbf{X}$ expressed in F_o is defined as a vector of its coordinates along the axes of the frame F_o, giving ${}^o\mathbf{X} = [{}^oX, {}^oY, {}^oZ]^\top$. This point can also be seen in F_c, giving ${}^c\mathbf{X}$. To express its coordinates, the transformation linking F_o and F_c is required.

This transformation can be modeled with a translation part ${}^c\mathbf{t}_o$ and a rotation part ${}^c\mathbf{R}_o$. The first term ${}^c\mathbf{t}_o \in \mathbb{R}^3$ represents the position of O_o in F_c and the last term ${}^c\mathbf{R}_o$ is a rotation matrix that expresses the orientation of axes of F_o in F_c. $SO(3)$ is the Special

Orthogonal group, also called 3D rotation group. It is defined as:

$$ {}^{c}\mathbf{R}_o \in SO(3) \quad \text{where} \quad SO(3) = \left\{ {}^{c}\mathbf{R}_o \in \mathbb{R}^{3\times 3} \mid {}^{c}\mathbf{R}_o{}^{\top} {}^{c}\mathbf{R}_o = \mathbf{I}_3, det({}^{c}\mathbf{R}_o) = 1 \right\}. \quad (1.1) $$

From those terms a point $\mathcal{X}$ expressed in F_o, $\mathbf{X}_o$, can be translated in F_c, giving $\mathbf{X}_c$ given by:

$$ {}^{c}\mathbf{X} = {}^{c}\mathbf{R}_o \, {}^{o}\mathbf{X} + {}^{c}\mathbf{t}_o. \quad (1.2) $$

1.1.2 Homogeneous coordinate system

Transposing this relation in a projective space allows to go from an affine equation as in equation (1.2) to a linear expression. To that end, a 3D point in the Euclidean space $\mathcal{X} \in \mathbb{E}^3$ represented by its coordinates $\mathbf{X} \in \mathbb{R}^3$ can be described by a point $\bar{\mathcal{X}}$ in a projective space $\mathbb{P}^3$ represented by its homogeneous coordinates $\bar{\mathbf{X}}^{\top} = [wX, wY, wZ, w]$, where w is a scalar. In that case $\mathbb{P}^3$ can be seen as an extension of $\mathbb{E}^3$ in which every point $w\mathbf{X} \in \mathbb{R}^3$ corresponds to a unique point $\bar{\mathcal{X}}$.
Then, normalizing and using the homogeneous coordinate system, equation (1.2) becomes:

$$ {}^{c}\bar{\mathbf{X}} = {}^{c}\mathbf{M}_o \, {}^{o}\bar{\mathbf{X}} \qquad \text{where} \qquad {}^{c}\mathbf{M}_o = \begin{bmatrix} {}^{c}\mathbf{R}_o & {}^{c}\mathbf{t}_o \\ \mathbf{0} & 1 \end{bmatrix}. \quad (1.3) $$

${}^{c}\mathbf{M}_o$ is then defined as the homogeneous transfer matrix expressing the 3D rigid transformation from F_o to F_c in $\mathbb{P}^3$. It is part of the Special Euclidean group $SE(3)$ defined by:

$$ {}^{c}\mathbf{M}_o \in SE(3) \quad \text{where} \quad SE(3) = \left\{ {}^{c}\mathbf{M}_o = \begin{bmatrix} {}^{c}\mathbf{R}_o & {}^{c}\mathbf{t}_o \\ \mathbf{0} & 1 \end{bmatrix} \mid {}^{c}\mathbf{R}_o \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}. \quad (1.4) $$

1.1.3 Homogeneous matrix parameterization

Other parameterizations of that transformation exist and can be composed of fewer parameters. The set of three parameters forming the translation part of the transformation are needed as they represent the displacement between F_o in F_c but not all rotation values are necessary. Indeed, as expressed in equation (1.1), the rotation matrix follows very specific rules. The induced constraints translate the fact that the displacement of an object must not affect its size or orientation. A rotation matrix must then be composed of three orthogonal unit vectors. In order to respect those constraints, several parameterization exist.

1.1.3.1 Euler angles

The simplest and most popular way to represent a 3D rotation is to use Euler angles. It consists in decomposing the rotation matrix as a combination of three simple one directional rotations along three orthogonal axes, typically r_x, r_y and r_z. Only three parameters are therefore required and the technique works very well as long as the convention on the order of the rotation axes used is known beforehand (a rotation on r_x then r_y and finally r_z will not give the same result as the same rotations but on r_y, r_x and then r_z). This simplicity, the fact that the representation is very intuitive and the facility to retrieve inverse parameters are the reasons it has been very widely used but the technique presents one major drawback: the gimbal-lock. If two rotation axes become aligned then a degree of freedom is lost.

1.1.3.2 Quaternions

To avoid the possibility of gimbal lock, quaternions have also been used to express 3D rotations. Quaternions are an extension of complex numbers, resulting in hypercomplex numbers constituted of four parameters representing a scalar and a vector. In a frame F with axes $\mathbf{i}, \mathbf{j}, \mathbf{k}$, a quaternion $\mathbf{q} = (a, b, c, d)$ would represent the point satisfying the equation $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$. A rotation of angle θ around an axis e would then be represented by the quaternion:

$$\mathbf{q} = \left[\cos\left(\frac{\theta}{2}\right) \quad \mathbf{e}^\top \sin\left(\frac{\theta}{2}\right)\right]^\top . \tag{1.5}$$

This representation allows a more complete representation of rotations to the cost of increased complexity since additional constraints apply such as having a norm equal to one. Because of this added complexity this technique is less used despite lifting the possibility of gimbal lock.

1.1.3.3 Axe and angle of the rotation

This representation models a rotation matrix as rotation of a certain angle around a 3D axis. It is very well suited to represent rotations as it is not subject to gimbal lock and can represent a rotation with only three parameters without any added constraints. This is why it will be used in our works. Considering a rotation of angle θ around a unitary axis $\mathbf{u} = [u_x, u_y, u_z]^\top$ can be described by a vector $\theta\mathbf{u} = [\theta u_x, \theta u_y, \theta u_z]^\top$ of norm $\| \theta\mathbf{u} \| = \theta$. That representation is referred to as the exponential canonical representation and can be mapped to a rotation matrix. There are two methods to get $\mathbf{R}$ from $\theta\mathbf{u}$: through the exponential map or thanks to Rodrigues' formula.

Exponential map Using the Lie algebra, the rotation matrix can be expressed from $\theta\mathbf{u}$ through the exponential map:

$$\begin{aligned} \mathbf{R} &= exp([\theta\mathbf{u}]_\times) \\ &= \sum_{n=0}^{\infty} \frac{[\theta\mathbf{u}]_\times^n}{n!} = \mathbf{I} + [\theta\mathbf{u}]_\times + \frac{1}{2!}[\theta\mathbf{u}]_\times^2 + \frac{1}{3!}[\theta\mathbf{u}]_\times^3 + ... \end{aligned} \tag{1.6}$$

where $[\theta\mathbf{u}]_\times$ is the skew-symmetric matrix associated to $\theta\mathbf{u}$ which is defined as a square matrix whose transpose is also its negative, giving $[\mathbf{v}]_\times^\top = -[\mathbf{v}]_\times$. For a vector $\mathbf{v} = [v_x \;\; v_y \;\; v_z]^\top$, its skew matrix is defined by:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} . \tag{1.7}$$

Rodrigues' formula Another way to find the rotation matrix from $\theta\mathbf{u}$ is to use Rodrigues' formula:

$$\mathbf{R} = \mathbf{I} + \frac{\sin\theta}{\theta}[\mathbf{u}]_\times + \frac{(1-\cos\theta)}{\theta^2}[\mathbf{u}]_\times . \tag{1.8}$$

From equation (1.8), the parameters of $\theta\mathbf{u}$ can also be found as:

$$\begin{aligned} \cos\theta &= \frac{\text{trace}(\mathbf{R}) - 1}{2} \\ \sin\theta\,[\mathbf{u}]_\times &= \frac{\mathbf{R} - \mathbf{R}^\top}{2} \end{aligned} \tag{1.9}$$

where trace($\mathbf{R}$) is the sum of the diagonal elements of the matrix $\mathbf{R}$. This is the model that is used in this document when performing 3D localization (see chapter 3).

1.1.4 Image formation

As shown in the previous sections, the 3D world and the different 3 transformations can be modeled thanks to several parameterization. But the case of study of this document does not occur in the 3D world. The available information come directly in the form of images provided by a digital camera. This is why it is necessary to define how the 3D world is perceived by a digital camera. As different cameras will also yield different results this section will first give details about perspective (pin-hole) cameras before extending the matter to omnidirectional cameras.

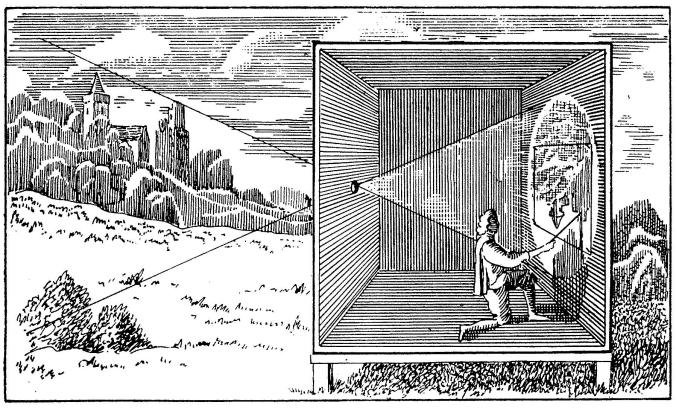


Figure 1.1: Representation of the *camera obscura*. The world is seen upside down as the light passes through a small opening in the dark chamber.

1.1.4.1 Pin-hole camera model

This section deals with the formation of digital images by cameras. Since it is crucial to understand displacement within the images, a focus will be made around how the 3D world is projected to result in digital images produced by digital cameras. Light emission and photometric information will not be discussed as the interest here is purely geometrical. The perspective projection model that is commonly used in computer vision [Faugeras, 1993, Hartley and Zisserman, 2001] will be the one considered in this document. Its role is to take into account the physical properties of a projection onto a plane in order to give a simple link between 3D world and 2D image data. Current cameras are similar to the first *camera obsucra* (fig. 1.1). Indeed, it was composed of a box in which a hole allowed to receive the light reflected or emitted by the environment, and a plane where the light was projected in the dark chamber. Nowadays, this is done by the camera but the projection center and the projection plane act as the hole and the plane of the *camera obscura*.To simplify understanding and representation however, we will consider the projection plane as being in front of the projection point. This also has this advantage of allowing the image of the world not to be upside down on the projection plane (also called image plane) $\mathcal{P}_\pi$. From that model, we can compute the 2D coordinates on the image plane $\mathbf{x} = [x \;\; y]^\top \in \mathbb{R}^2$ of a 3D point $\mathcal{X}$ expressed in the camera frame ${}^c\mathbf{X} = [{}^cX, \; {}^cY, \; {}^cZ]^\top$ as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{{}^cZ} \begin{bmatrix} {}^cX \\ {}^cY \end{bmatrix} \tag{1.10}$$

where f is the focal distance of the camera which represents the position of the projection plane along the ${}^c\mathbf{z}$ axis of F_c. Expressing the relation thanks to homogeneous coordinates

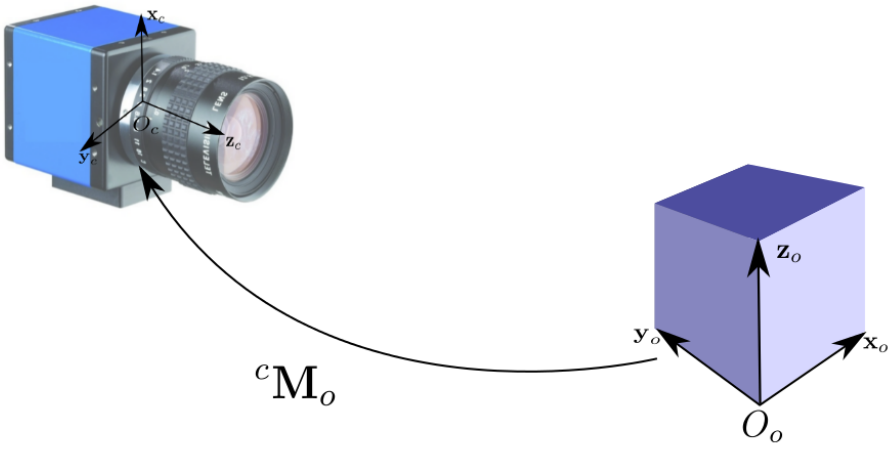


Figure 1.2: Representation of the world in the Euclidean space. The orthogonal basis associated to the object is called object frame and the one attached to the camera is called camera frame.

allows us to get a linear equation from $\mathcal{R}^3$ to $\mathcal{P}^2$:

$$ {}^c\bar{\mathbf{x}} = \mathbf{A}\,{}^c\bar{\mathbf{X}} \quad \text{where} \quad \mathbf{A} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{array}{l} {}^c\bar{\mathbf{x}} = [x,\ y,\ 1]^\top \\ {}^c\bar{\mathbf{X}} = [{}^cX,\ {}^cY,\ {}^cZ,\ 1]^\top . \end{array} \tag{1.11} $$

${}^c\bar{\mathbf{x}}$ is the projection of $\mathcal{X}$ on the image plane in homogeneous coordinates. These coordinates are expressed in the metric space. A supplementary step is therefore needed in order to easily manipulate digital images.

1.1.4.2 From world projection to digital images

A digital image is the result of the discretization of the image plane onto a regular grid. The elements of the grid are pixels (picture elements). This grid can be expressed through a $w \times h$ array where w is the width of the image and h its height. Each pixel is then an element of the matrix of coordinates (u, v) (see Figure 1.3).The values of each pixel represent the light perceived by the camera. This value can be a grey level value, therefore expressed through one value usually coded between 0 and 255, or can express color information. Considering a point $\mathbf{x} = [x,\ y]^\top$ in meters, its position on the grid represented by its pixel coordinates $\mathbf{p} = [u,\ v]^\top$ is parameterized by four degrees of freedom. They represent the dimensions of a pixel in meter l_x and l_y and the coordinates of the principal point $\mathbf{p}_0 = [u_0,\ v_0]^\top$. The principal point is the point where the $\mathbf{z}$ axis of F_c intersects the image plane, which means its metric normalized coordinates on the projection plane are ${}^c\mathbf{p}_0 = [0,\ 0]^\top$ and its 3D normalized coordinates are ${}^c\mathbf{P}_0 = [0,\ 0,\ f]^\top$. The relation between $\mathbf{p}$ and $\mathbf{x}$ is given by:

$$ \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} \frac{1}{l_x} & 0 \\ 0 & \frac{1}{l_y} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} . \tag{1.12} $$

Using homogeneous coordinates in order to get a linear expression then means performing

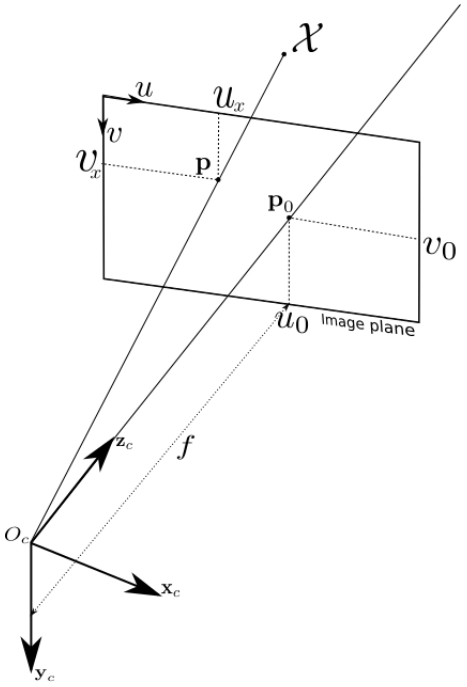


Figure 1.3: The 3D world is captured by the camera sensors on the image plane. the coordinates of every point in the image can be computed from its 3D coordinates and the intrinsic camera parameters.

a projective transformation $\mathbf{K}'$ from $\mathbb{P}^2$ to $\mathbb{P}^2$:

$$\bar{\mathbf{p}} = \mathbf{K}'\mathbf{x} \quad \text{where} \quad \mathbf{K}' = \begin{bmatrix} \frac{1}{l_x} & 0 & u_0 \\ 0 & \frac{1}{l_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.13}$$

Combining previous equations allows to express the transformation from the 3D point ${}^c\bar{\mathbf{X}}$ in F_c to the projected point in the image $\bar{\mathbf{p}}$:

$$\bar{\mathbf{p}} = \mathbf{K}'\mathbf{A}\,{}^c\bar{\mathbf{X}} = \mathbf{K}\mathbf{\Pi}\,{}^c\bar{\mathbf{X}} \quad \begin{matrix} \text{where} \quad \mathbf{K} = \begin{bmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\ \text{and} \quad \mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix} \tag{1.14}$$

where $p_x = \frac{f}{l_x}$ and $p_y = \frac{f}{l_y}$. $\mathbf{K}$ is called the intrinsic parameter matrix as it contains all the intrinsic parameters of the camera while $\mathbf{\Pi}$ is the projection matrix. To express $\bar{\mathbf{p}}$ from the 3D point in the object frame we need the whole transformation pipe which is:

$$\bar{\mathbf{p}} = \mathbf{K}\mathbf{\Pi}\,{}^c\mathbf{M}_o\,{}^o\bar{\mathbf{X}}. \tag{1.15}$$

In this case ${}^c\mathbf{M}_o$ is called the pose of the camera. Pose is only tied to the camera position in the world whereas intrinsic parameters are highly dependent on the camera and its lens. This is because they are closely tied to physical parameters that can vary between cameras built following the same fabrication process. Even though an estimation of intrinsic parameters is usually given by the constructor, it is crucial to compute them accurately, which

is done by calibration [Brown, 1971, Tsai, 1986]. A calibration pattern is used to match the 3D positions to image coordinates of the point giving a system of equations. Solving this system allows to compute the parameters of $\mathbf{K}$ and can also be used to estimate ${}^{c}\mathbf{M}_{o}$ at the same time. Let us note that this description is only valid considering a perfect optical system presenting no deformation of the world. As it is obviously not necessarily the case in real life applications several works have added distortions to the projection model. The added parameters controlling radial distortions in the projection can be estimated [Faugeras, 1993, Hartley and Zisserman, 2001] and added to the projection model. However in these works those deformations are considered negligible and therefore only the perspective projection model detailed before is used.

1.2 Linear warping functions: parameterizing the image motion

Being able to determine the displacement of an object in a sequence of images means being able to model the aforementioned displacement. In this section we present several transformation functions that allow to model different types of displacement in the images. These functions are called warping functions, noted $w(.)$, and are parameterized by several factors. They allow to map a point $\mathbf{p}_k = [x \;\; y]$ in an image $\mathbf{I}_k$ to a point $\mathbf{p}_{k+1} = w(\mathbf{p}_k, \mathbf{u})$ in an image $\mathbf{I}_{k+1}$ with $\mathbf{u}$ the parameters of the warp.

1.2.1 Translation model

The simplest movement model there is. It was the first warp function to be used in order to perform visual tracking [Lucas and Kanade, 1981] (see Chapter one for details on visual tracking). Considering only two parameters which represent a translation along the x and y axes of the image, it offers a poor freedom of displacement for the object in the camera frame. Indeed, it can only depict a planar translational motion parallel to the image plane. It is however very efficient in these conditions as only two parameters need to be estimated. The transformation function is then:

$$\mathbf{p}_{k+1} = w(\mathbf{p}_k, \mathbf{u}) = \mathbf{p}_k + \mathbf{t} \tag{1.16}$$

where $\mathbf{u} = \mathbf{t} = [t_x \;\; t_y]^\top$.

1.2.2 sRt transformation

The sRt warp, often called similitude transformation or "shape-preserving model", adds freedom to the translation warp [Goodall, 1991]. It considers, in addition to the translation, a scale factor (represented by a scalar s) and a rotation $\mathbf{R}$ (of angle ϕ around the $\mathbf{z}$ axis). It can then be expressed through four parameters:

$$\mathbf{p}_{k+1} = w(\mathbf{p}_k, \mathbf{u}) = (s)\mathbf{R}(\phi)\mathbf{p}_x + \mathbf{t} \tag{1.17}$$

with:

$$\mathbf{R} = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix}. \tag{1.18}$$

The parameter vector is then $\mathbf{u} = (s,\ \phi,\ t_x,\ t_y)$. It depicts the displacement of an object moving, rotating and changing size in the image plane, but can also be used to estimate the movement of the camera with relation to a fixed object. In this case the translated

displacement is a combination of a 3D translation and a rotation along the **z** axis of the image plane. The object then has to move in a plane parallel to the image plane in order for the model to be accurate. This is why it is widely used for example in the domain of UAV navigation where perspective effects are negligible [Yol et al., 2014].

1.2.3 Affine model

Not restraining the transformation matrix in the sRt model to translate a 2D rotation leads to the affine model. It is then a generalization of the sRt warp which adds freedom to modify the shape of the considered object. In doing so it is not a strict representation of a particular motion in the 3D space anymore but it constitutes a good compromise to cope with minor perspective effects while adding only two parameters:

$$\mathbf{p}_{k+1} = w(\mathbf{p}_k, \mathbf{u}) = \mathbf{A}\mathbf{p}_k + \mathbf{t} \qquad \text{where} \qquad \mathbf{A} = \begin{pmatrix} a_0 & a_1 \\ a_2 & a_3 \end{pmatrix}. \tag{1.19}$$

In that case, the parameter vector becomes $\mathbf{u} = (a_0,\ a_1,\ a_2,\ a_3,\ t_x,\ t_y)$.

1.2.4 Homography

The homography warp is widely used in the domain of computer vision [Faugeras and Lustman, 1988, Baker and Matthews, 2004, Benhimane and Malis, 2006b]. It allows to fully describe the 3D motion of a planar object through a 2D transformation. As it takes into account the perspective it is very useful. It is also very closely related to 3D information of the scene which can be really helpful when performing visual tracking. This is why it will be used in this document when performing visual tracking tasks dealing with rigid piece-wise planar objects. Considering a point $\mathcal{X}$ belonging to a plane $\mathcal{P}_1$. Every point ${}^1\mathbf{X}$ in a frame F_1 belonging to $\mathcal{P}_1$ then satisfies the following expression:

$${}^1\mathbf{X}\,\mathbf{n} = d \tag{1.20}$$

which can also be written as:

$$\frac{\mathbf{n}^\top {}^1\mathbf{X}}{d} = 1 \tag{1.21}$$

where $\mathbf{n}$ is a unit vector normal to $\mathcal{P}_1$ expressed in F_1 and d is the distance from $\mathcal{P}_1$ to the origin of F_1. As shown in equation (1.2), $\mathcal{X}$ can be expressed in another frame F_2 as:

$${}^2\mathbf{X} = {}^2\mathbf{R}_1\ {}^1\mathbf{X} + {}^2\mathbf{t}_1. \tag{1.22}$$

Factorizing this expression thanks to equation (1.21) leads to:

$${}^2\mathbf{X} = \mathbf{H}\ {}^1\mathbf{X} \qquad \text{where} \qquad \mathbf{H} = {}^2\mathbf{R}_1 + \frac{{}^2\mathbf{t}_1\,\mathbf{n}^\top}{d} \tag{1.23}$$

where $\mathbf{H}$ is a (3×3) matrix called the homography matrix. Thanks to the relation expressed in equation (1.14), that definition is true on both homogeneous 2D coordinates and 3D coordinates. To express that relation in homogeneous 2D coordinates, let us consider the projections of both ${}^1\bar{\mathbf{X}}$ and another point ${}^2\bar{\mathbf{X}}$, belonging to a plane $\mathcal{P}_2$:

$$\begin{aligned} \bar{\mathbf{p}}_1 &= \mathbf{K}\mathbf{\Pi}\ {}^1\bar{\mathbf{X}} \\ \bar{\mathbf{p}}_2 &= \mathbf{K}\mathbf{\Pi}\ {}^2\bar{\mathbf{X}} \end{aligned} \qquad \text{meaning} \qquad \begin{aligned} {}^1\bar{\mathbf{X}} &= \mathbf{\Pi}^\top\mathbf{K}^{-1}\ \bar{\mathbf{p}}_1 \\ {}^2\bar{\mathbf{X}} &= \mathbf{\Pi}^\top\mathbf{K}^{-1}\ \bar{\mathbf{p}}_2. \end{aligned} \tag{1.24}$$

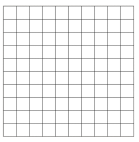
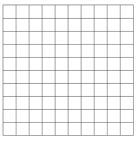
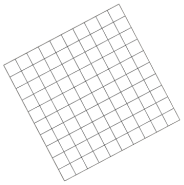
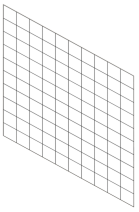
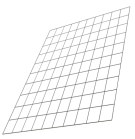
Reference			
Translation warp			$\mathbf{u} \in \mathbb{R}^2$
sRt warp			$\mathbf{u} \in SL(2) \times \mathbb{R}^2$
Affine warp			$\mathbf{u} \in \mathbb{R}^6$
Homography warp			$\mathbf{u} \in SL(3)$ (2D) or $\mathbf{u} \in SE(3)$ (3D)

Figure 1.4: Overview of commonly used rigid transformation functions. The more parameters, the more freedom is given.

Inputting equation (1.24) into equation (1.23) finally allows us to express a relation between the two image points:

$$\begin{aligned}
\bar{\mathbf{p}}_2 &= \mathbf{K\Pi}\,{}^2\bar{\mathbf{X}} \\
\bar{\mathbf{p}}_2 &= \mathbf{K\Pi}\,\mathbf{H}\,{}^1\bar{\mathbf{X}} \\
\bar{\mathbf{p}}_2 &= \mathbf{K\Pi}\,\mathbf{H}\,\mathbf{\Pi}^\top\mathbf{K}^{-1}\,\bar{\mathbf{p}}_1 \\
\bar{\mathbf{p}}_2 &= \mathbf{G}\,\bar{\mathbf{p}}_1 \qquad \text{where} \qquad \mathbf{G} = \mathbf{K\Pi}\,\mathbf{H}\,\mathbf{\Pi}^\top\mathbf{K}^{-1}.
\end{aligned} \tag{1.25}$$

From this equation it is the possible to get a 3D homography and even to get back to the rotation and translation parts of the transformation. However this is only the case when $\mathbf{G}$ belong to the special Lie group ($\mathbf{G} \in SL(3)$) which is not assured by equation (1.25). A possible solution for this is to proceed as proposed by the authors of [Benhimane and Malis, 2004] who showed that when the determinant of $\mathbf{G}$ is fixed to 1 and the matrix is computed thanks to some properties of the Lie Algebra, the resulting matrix belongs to $SL(3)$. As this transformation function will be used in this document let us describe how its parameters will be considered. Rewriting equation (1.25) leads to:

$$\bar{\mathbf{p}}_b = \begin{pmatrix} g_0 & g_1 & g_2 \\ g_3 & g_4 & g_5 \\ g_6 & g_7 & 1 \end{pmatrix} \bar{\mathbf{p}}_a. \tag{1.26}$$

Thanks to the homogeneous coordinate system, only 8 parameters are needed and can therefore be stocked in a vector $\mathbf{u} \in SL(3)$.

1.3 Conclusion

In this chapter, the necessary tools to model a 3D environment are defined. The projection pipeline from this 3D environment in the 2D image plane and onto digital images is then detailed. Classical image motion models that are commonly used in computer vision are then defined. Using the tools exposed in this chapter, the next part will recall the state of the art of visual tracking before two contributions are proposed.

PART 1

Visual Tracking

Visual tracking has been one of the main research areas of the computer vision domain for years, leading to a very rich literature. The objective of a visual tracking task is to estimate the motion of an object and to localize it in an image sequence. It is used within a very wide range of applications. For example, automatic surveillance has used visual trackers for a long time in order to follow pedestrians in image sequences. Robot manipulation is also tied very closely to this topic, as more and more robotic tasks are automatized through the use of cameras relying visual tracking algorithms. Augmented reality is another application domain which uses such algorithms in order to add a virtual layer to real-world data. Other application domains also include network broadcasting, professional sports or movie production that track targets or include augmented reality in their programs. Camera firmwares also use visual tracking in order to create panoramas or perform mosaicing. In order to be considered successful, a visual tracking algorithm needs to be able to estimate accurately the displacement featured in an image sequence but must also be robust and time-efficient. The robustness of a visual tracking task is therefore key to its success. This is due to the fact that image sequences where no perturbations appear are very rare in everyday-world applications and most of the time the featured scenes undergo perturbations ranging from small global light variations to more complicated things like specular spots, occlusions or change in modality.
In the following chapters, robust (dis)similarity functions are then considered in order to propose dense visual tracking processes that do not rely on any robustification steps to be able to cope with perturbations in image sequences. Taking advantage of several motion models, these functions are used to propose algorithms that track multiple planes of a 3D model in the same optimization and that are able to perform registration of non-rigid surfaces.

2

Visual tracking overview

This chapter presents a summary of common visual tracking approaches. The main strategies used to perform visual tracking tasks are defined and their advantages and drawbacks are discussed. In order to use image information as data for visual tracking, it is crucial to be able to model it. This has been done extensively over the years and in this chapter a distinction is made between two classes of modeled visual information, also called visual features. On one hand are geometrical features, which extract information from the image to perform visual tracking and on the other hand are dense methods, which use directly light intensity information.

2.1 Using geometrical features to estimate displacement

Geometrical features rely on the extraction of geometrical data from the images. Those features can represent different types of information (points, lines, ...) but they all have in common the fact that they need to be extracted for each new image of the sequence. In this section, several types of geometrical features are shown and strategies designed to use geometrical features are then discussed.

2.1.1 Local geometrical features

A large set of geometrical features have been used over the years to define objects to be localized using visual tracking. The first approaches used points of interest [Harris and Stephens, 1988], which can be extended to dots [Marchand, 1999]. Combining points led to using straight lines [Deriche and Faugeras, 1990], segments [Hager and Belhumeur, 1998, Marchand, 1999, Boukir et al., 1998], ellipses [Vincze, 2001] or contours [Berger, 1994, Blake and Isard, 1998]. The main drawback of these geometrical features is that they are heavily specific to image sequences. Not all sequences can for example be tracked using lines or ellipses. It is for this reason that those simple models are also used to create enhanced models, combining several types of visual information into a more complete descriptor. A good example of local descriptors combined to get more global descriptors are SIFT descriptors [Lowe, 2004] or another speeded up robust feature (SURF descriptors) [Bay et al., 2006]. Starting from a point feature, an extension is made by adding gradient information in the surrounding area and looking at that data on several levels of a pyramid. Geometrical descriptors have been combined in a lot of different ways over the years in order to get better tracking results in more complex situations where simple features are not always successful. Considering several lines constrained by a 3D model can for example lead to model-based tracking [Brooks et al., 1979, Lowe, 1991, Drummond and Cipolla, 2002, Comport et al., 2006, Petit et al., 2014]. Adding points on the visible faces of the

model then create a hybrid visual tracking task than increases robustness [Pressigout and Marchand, 2007]. The main advantage of geometrical descriptors is computational cost. Because few features are considered, the tracking process is simple and therefore efficient. It is also naturally quite robust, as the geometry of an object is not subject to variations due to changes in lighting conditions. But some drawbacks can also be accounted for. The presence of visible geometrical features is obviously necessary which can be problematic in some scenes if some features become occluded for example, since the features need to be extracted or re-localized in every image. Finally, considering only a small number of features is very efficient computationally speaking but it means that perturbations that impact only a few number of features can have a significant impact on the results of the visual tracking task.

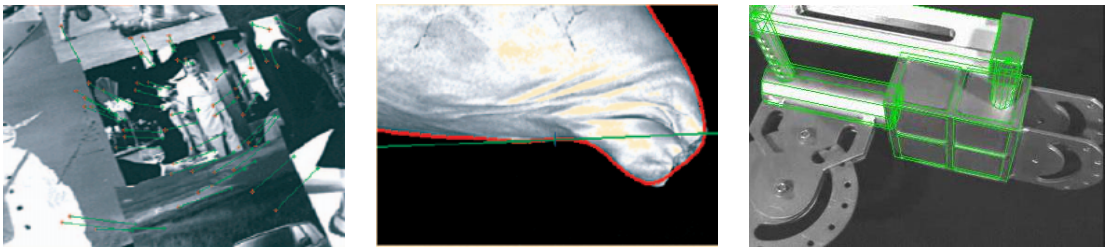


Figure 2.1: Example of points, contours and line segments used to track an object thanks to the ViSP library [Marchand et al., 2005].

2.1.2 From local features to motion estimation

Chapter 1 has shown how motion can be modeled several ways. Depending on the considered displacement model, the tracking can be done in the image plane, creating a 2D registration and leading to a motion estimation problem, or in the 3D world, leading to the estimation of the relative pose between the camera and the object. The registration problem can be seen as the optimization of the similarity between features currently seen and other information known beforehand. 2D registration using geometrical features often

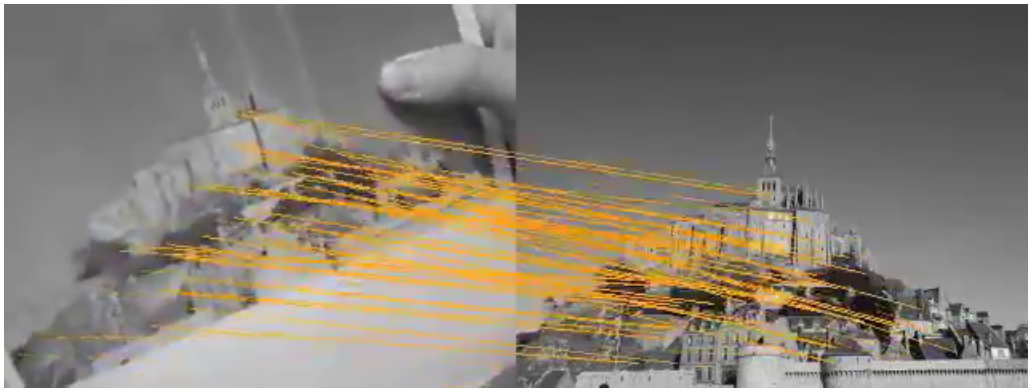


Figure 2.2: Example of point matching using the ViSP library [Marchand et al., 2005].

starts by an extraction and matching step of a set of geometrical features. Using keypoints for example, if the chosen model is only represented by a set of points then at each new frame the corresponding keypoints need to be extracted from the new current image and matched with points in the previous image. Several techniques exist to perform this first

phase in the tracking process, the most common technique being RANSAC [Fischler and Bolles, 1981]. Short for "RANdom SAmple Consensus", the RANSAC method starts from an arbitrary result and iteratively verifies it. Whether the result is good or not, a new iteration is launched until the correct estimation is reached. The method is very easy to use but can be very inefficient as it is non-deterministic. This is why information are often added and to use point features, descriptors like SIFT or SURF are used. Thanks to the added data, the points can be matched more efficiently. More complex features can also require other algorithmic layers before the tracking process can be started. For example when using lines, curves, ellipses or more complex models their position needs to be estimated in the image, which is often done by performing a visual tracking phase on lower level features. It is for example what is done to use lines thanks to the moving edges algorithm proposed by [Bouthemy, 1989] which was later used and extended to track more complex shapes by adding 3D constraints [Petit et al., 2014] which use points displacements in order to estimate lines or curves motion. Once the features have been located in the image plane the registration phase consists in minimizing the difference between the projection of the features in the current image $\mathbf{s}$ and their corresponding counterparts in the reference image $\mathbf{s}^*$. Using a warp function w in order to represent their motion parameterized by a vector $\mathbf{u}$, the goal is to find the optimal parameters $\widehat{\mathbf{u}}$ that best map the displaced features $w(\mathbf{s}, \widehat{\mathbf{u}})$ to $\mathbf{s}^*$:

$$\widehat{\mathbf{u}} = \arg\min_{\mathbf{u}} \sum_{k=0}^{N_s} \| \mathbf{s}_k^* - w(\mathbf{s}_k, \mathbf{u}) \| \tag{2.1}$$

where N_s is the number of features considered in the model. For example, let us consider $\mathbf{s}$ to represent point features $\mathbf{x}$ used in order to estimate the parameters of an affine warp [Shi and Tomasi, 1994]. Equation (2.1) then becomes:

$$(\widehat{\mathbf{A}}, \widehat{\mathbf{t}}) = \arg\min_{\mathbf{A},\mathbf{t}} \sum_{k=0}^{N_s} \| \mathbf{x}_k^* - (\mathbf{A}\mathbf{x}_k + \mathbf{t}) \| . \tag{2.2}$$

Since most feature models cannot be linearly linked with their motion, the use of non-linear optimization schemes are often needed in order to estimate optimal motion parameters. Different examples of non-linear optimizations are recalled on Frame 1. Let us also add that in most cases aberrant data is present either due to bad detection or scene perturbations. In those situations robustness layers can be added in order to filter out that data and not take it into account for the optimization. For example RANSAC [Fischler and Bolles, 1981], least median of square regression [Rousseeuw, 1984] or M-estimators [Huber, 1981] have been considered [Hager and Belhumeur, 1998, Lepetit and Fua, 2005, Fitzgibbon, 2003] (M-estimator techniques are recalled in Frame 2).

2.1.3 3D registration: Pose estimation

Another way to see the registration problem is to estimate the position of the object in the 3D camera frame F_c or the object 3D frame F_o instead of estimating the displacement in the image plane. In that case, *a priori* knowledge is often required such as for example the intrinsic camera parameters γ (often known in the form of the matrix $\mathbf{K}$) or the 3D model of the chosen object. Various approaches have used a known model before tracking it as [Drummond and Cipolla, 2002, Vacchetti et al., 2004, Comport et al., 2006]. Other strategies have also been considered such as learning the model from images [Lim and Binford, 1988] or while the tracking phase is being performed [Davison et al., 2007], a process often called SLAM, short for "Simultaneous Location And Tracking". Let us

consider here the most common approach, where the 3D model of the object and the camera parameters are known. In this case, the registration problem can be seen as the optimization of the similarity between the known and projected models M and $pr_\gamma(M)$ over the parameters of the relative position between the camera and the model represented by the pose vector $\mathbf{r}$:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} f(I^*, pr_\gamma(M, \mathbf{r})). \tag{2.3}$$

Comparing the model to geometrical features extracted in I^* is very interesting since the geometrical properties of the object are known and it is easy to project them in the image plane thanks to a homogeneous transformation between the object and camera frames. As seen in the previous chapter, a homogeneous transformation can be modeled by a vector of 6 parameters $\mathbf{r} \in SE(3)$. These pose parameters can be obtained through the minimization of the error between the observed and desired features. Considering a model made of points for example, from four points ${}^o\mathbf{X}_n$ (with homogeneous coordinates $\widehat{\mathbf{x}_n}$ in the image) representing the model,a current pose estimation can be defined as:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \sum_{n=0}^{N} \parallel \mathbf{x}_n^* - \mathbf{K}{}^c\mathbf{M}_o(\mathbf{r}){}^o\mathbf{X}_n \parallel . \tag{2.4}$$

From that formulation, the optimization can be performed.The problem being highly non-linear which means that optimization schemes are still needed, iteratively refining a current estimation of an update of the parameters $\mathbf{r}$:

$$\widehat{\Delta\mathbf{r}}_k = \arg\min_{\Delta\mathbf{r}^k} \sum_{n=0}^{N} \parallel \mathbf{x}_n^* - \mathbf{K}{}^c\mathbf{M}_o(\mathbf{r}_k + \Delta\mathbf{r}_k){}^o\mathbf{X}_n) \parallel . \tag{2.5}$$

Another way to look at this optimization problem is to use the duality between visual tracking and visual servoing and consider the tracking task as a virtual visual servoing (VVS) task. A virtual camera is then moved thanks to a velocity vector $\mathbf{v}$ computed by the VVS step. Since the velocity depicts the variation of the pose with respect to time, the lapse of time between two updates of the pose can be considered as a gain, leading to an update rule of the pose parameters [Marchand and Chaumette, 2002, Sundareswaran and Behringer, 1998]:

$${}^c\mathbf{M}_o^{k+1} = e^{[\mathbf{v}]}{}^c\mathbf{M}_o^k \tag{2.6}$$

where $e^{[\mathbf{v}]}$ is the exponential map of $\mathbf{v}$. This method is very successful as its simplicity over the previous technique makes its less complex to use.

Frame 1 - Non-linear optimization schemes

Considering an objective function f that is to be optimized (here we will consider f a dissimilarity function and therefore minimize it). If f is parameterized by a parameter vector $\mathbf{u}$, then the problem can be written as:

$$\widehat{\mathbf{u}} = \arg\min_{\mathbf{u}} f(\mathbf{u}). \tag{2.7}$$

as f is not linear in most cases involving visual features, it needs to be iteratively minimized thanks to a non-linear optimization process. This means that, at each new iteration k the warp parameters $\mathbf{u}$ are updated thanks to an increment $\Delta\mathbf{u}$ that has been estimated by the last minimization step. This is done until convergence. For example, considering a parameter update performed as an addition (as would be the case with a translation warp) and point features $\mathbf{x}$, equation (2.7) can be rewritten as:

$$\widehat{\Delta\mathbf{u}} = \arg\min_{\Delta\mathbf{u}} f(\mathbf{x}^*, w(\mathbf{x}, \mathbf{u} + \Delta\mathbf{u})). \tag{2.8}$$

Taking the classical case of using a difference function, this can be rewritten as:

$$\begin{aligned} \widehat{\Delta\mathbf{u}_k} &= \arg\min_{\Delta\mathbf{u}_k} \sum_{n=0}^{N_s} \parallel \mathbf{x}_n^* - w(\mathbf{x}_n, \mathbf{u}_{k+1}) \parallel \\ &= \arg\min_{\Delta\mathbf{u}_k} \sum_{n=0}^{N_s} \parallel \mathbf{x}_n^* - w(\mathbf{x}_n, \mathbf{u}_k + \Delta\mathbf{u}_k) \parallel . \end{aligned} \tag{2.9}$$

Several approaches exist in order to perform that optimization. In this frame we will detail three of the most encountered which are the ones that will be used in this document.

Steepest gradient method

This is the simplest approach. Approaching the curve of the function with a plane and going in the steepest possible direction allows to navigate towards a global minimum. In order to find that direction, the increment is computed thanks to the estimated plane parameters. Those parameters can be derived from the measure's definition:

$$\widehat{\Delta\mathbf{u}_k} = -\alpha \left. \frac{\partial f(\mathbf{u})}{\partial \Delta\mathbf{u}} \right|_{\mathbf{u}=\mathbf{u}_k} \tag{2.10}$$

where α is a scale factor that is applied in order to avoid large steps that would cause the optimization to diverge towards a local minimum.

Gauss-Newton method

To perform a more accurate local approximation, the Gauss-Newton method approaches the function with a parabola. This is done by considering the Taylor expansion of the displacement function:

$$w(\mathbf{x}_n, \mathbf{u}_k + \Delta\mathbf{u}_k) \approx w(\mathbf{x}_n, \mathbf{u}_k) + \frac{\partial w(\mathbf{u}_k)}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} \tag{2.11}$$

which leads to:

$$\widehat{\Delta\mathbf{u}_k} = -\mathbf{J}^{+} \parallel \mathbf{x}_n^* - w(\mathbf{x}_n, \mathbf{u}_k) \parallel \qquad \text{where} \qquad \mathbf{J} = \frac{\partial w(\mathbf{u})}{\partial \Delta\mathbf{u}}. \tag{2.12}$$

Let us note that the Gauss-Newton approach is a particular case of the Newton-Raphson method. In the latter the Hessian would not be approached whereas here it is considered to be equal to $\mathbf{J}^{+}\mathbf{J}$.

Levenberg-Marquardt method
In between the two strategies detailed before lies the Levenberg-Marquardt method. The update rule is computed as:

$$\widehat{\Delta\mathbf{u}_k} = -(\mathbf{J}_{\top}\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}_{\top} \parallel \mathbf{x}_n^* - w(\mathbf{x}_n, \mathbf{u}^k) \parallel . \tag{2.13}$$

The λ parameter in this case can be seen as a tuning parameter. The bigger λ is, the closer the chosen direction is to a steepest gradient and a smaller λ leads to a method close to a Gauss-Newton approach which allows for a finer-tuned optimization.

Frame 2 - M-estimators

Outlying data is present in most cases when performing visual servoing. This can be due to several factors, amongst which are erroneous data resulting of occlusions, matching errors or illumination variations. Those data can be problematic and this is why the optimization must adapt to limit the impact of the outliers. This is why a M-estimator ρ is often added to the similarity function f:

$$\widehat{\Delta\mathbf{u}_k} = \arg\min_{\Delta\mathbf{u}^k} \sum_{n=0}^{N_s} \rho(\mathbf{x}_n^* - w(\mathbf{x}_n, \mathbf{u}^k + \Delta\mathbf{u}^k)). \tag{2.14}$$

Several M-estimators have been proposed in the literature. Instead of using a simple quadratic function $\rho(x) = x^2$ when optimizing the SSD for example, a Lorentzien kernel can be used [Fitzgibbon, 2003]:

$$\rho(x) = log\left(1 + \frac{t^2}{\sigma}\right) \tag{2.15}$$

where σ is a tuning parameter of the estimator. This function is defined the same way for every value of x, but other functions can be conditionals as the Huber function [Hager and Belhumeur, 1998]:

$$\rho(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } \parallel x \parallel \leq \sigma \\ \sigma \parallel x \parallel -\frac{1}{2}\sigma^2 & \text{elsewhere} \end{cases} \tag{2.16}$$

Finally, another M-estimator commonly used is the Tuckey function [Lepetit and Fua, 2005]:

$$\rho(x) = \begin{cases} \frac{\sigma^2}{6}\left[1 - \left(1 - \left(\frac{x}{\sigma}\right)^2\right)^3\right] & \text{if } \parallel x \parallel \leq \sigma \\ \frac{\sigma^2}{6} & \text{elsewhere} \end{cases} \tag{2.17}$$

This section has shown how feature-based approaches use geometrical properties of the scene to perform 2D and 3D visual tracking. But these methods require a step of extraction and matching before performing the motion estimation. In order to avoid that algorithmic layer that can be erroneous or computationally expensive, other approaches propose to use the whole image data as visual feature, resulting in what are called dense approaches.

2.2 Dense visual tracking

Instead of using features extracted from the images that represent the geometrical properties of an object, its appearance can be considered to perform visual tracking. Modeling the way the object is seen, with colors, luminances or textures is then an alternative to the use of geometrical features. Several works have been proposed that use this idea. In order to perform visual tracking, histograms have been used [Comaniciu et al., 2000], but approaches closer to the texture of the object have also been successfully introduced [Lucas and Kanade, 1981, Irani and Anandan, 1998, Jurie and Dhome, 2001, Benhimane and Malis, 2004, Dame and Marchand, 2011]. These approaches are referred to as dense techniques and consider more features, using image templates, in order to get a more global representation of the scene. But considering more data leads to more computationally expensive algorithms. The appearance of an object may also be subject to lighting variations in the scene. Indeed, light can create shifts in the luminance of a texture or specularities can appear. This is why dense visual tracking algorithms have to find ways to achieve robustness. Contrary to geometrical approaches, dense methods cannot end up in situations where no feature can be found, due to the quantity of considered data. That redundancy of information also allows for a very accurate estimations of the displacement which are often robust to partial occlusions of the object. Moreover, the information is directly present in the images. It does not need to be extracted or pre-processed which eliminates potential issues such as erroneous detections or feature matching failures.

2.2.1 Using luminance data: Similarity measures

Using texture to perform visual tracking means being able to estimate a similarity or dissimilarity between image patterns. Indeed, a visual tracking task as defined in [Shi and Tomasi, 1994] aims at optimizing a cost function in order to find a displacement in the image. Hereafter are several similarity or dissimilarity functions that can be used to perform visual tracking.

2.2.1.1 Sum of Squared Differences (SSD)

The sum of squared differences (SSD) compares directly the light intensities of each pixel between a seen image I and a reference I^*:

$$SSD(I, I^*) = \sum_{k=1}^{N_{\mathbf{x}}} [I(\mathbf{x}_k) - I^*(\mathbf{x}_k)]^2 \tag{2.18}$$

where $N_{\mathbf{x}}$ is the number of pixels considered in I^*. Its simplicity and the fact that it is very computationally efficient have made it the most used registration function for dense visual tracking approaches [Shi and Tomasi, 1994, Hager and Toyama, 1998]. The main drawback of this approach is the fact that it is very poorly robust to scene variations. Light perturbations, occlusions or specularities are problematic when using the SSD as a registration function. To circumvent this fact, several layers have been added to the SSD in

order to add robustness, such as adding parameters modeling the structure of light [Silveira and Malis, 2007]. But those approaches introduce complexity and the simplicity of using the SSD instead of a more complex but more robust function then vanishes.

2.2.1.2 Sum of Conditional Variance (SCV)

Recently, it has been proposed a tracking algorithm based on the sum of conditional variance [Richa et al., 2011]. The SCV is a template-based dissimilarity function but rather than using the raw image I, as it is the case for the SSD, it is adapted, at each new frame, to match the illumination conditions of the template image I^*, creating an adapted patch $\hat{I}$ thanks to an expectation operator $\mathcal{E}$:

$$\hat{I}(\mathbf{x}) = \mathcal{E}(I^*(\mathbf{x}) \mid I(\mathbf{x})). \tag{2.19}$$

This operator computes, for each grey level in I, an adapted one which reflects the changes the current template needs to undergo to match the illumination conditions of I^*:

$$\hat{I}(j) = \sum_i i \, \frac{p_{II^*}(i,j)}{p_I(j)} \tag{2.20}$$

where p_I and p_{II^*} are respectively the probability density function and joint probability density function of I and I^*:

$$\begin{aligned} p_{II^*}(i,j) &= P(I^*(\mathbf{x}) = i, I(\mathbf{x}) = j) \\ &= \frac{1}{N_\mathbf{x}} \sum_{k=1}^{N_\mathbf{x}} \alpha(I^*(\mathbf{x}_k) - i)\alpha(I(\mathbf{x}_k) - j) \end{aligned} \tag{2.21}$$

where $\alpha(u) = 1$ if and only if $u = 0$. From this, the probability density function of I is given by:

$$p_I(i) = \sum_j p_{II^*}(i,j). \tag{2.22}$$

Finally, the dissimilarity function is given by:

$$SCV(I, I^*) = \sum_{k=1}^{N_\mathbf{x}} \left[I^*(\mathbf{x}_k) - \hat{I}(\mathbf{x}_k) \right]^2. \tag{2.23}$$

The SCV is then able to adapt to the modality of the scene, allowing to register when global light perturbation occur, but is still subject to failure when confronted to local variations such as occlusions or specularities. The simplicity stemming from the similarity with SSD and increased robustness with relation to the SSD make it a good compromise with relation to other measures when looking for robustness without much increased complexity.

2.2.1.3 Zero-mean Normalized Cross Correlation (ZNCC)

The normalized cross correlation (used for tracking purposes in [Irani et al., 1992]), later extended to the zero-mean normalized cross correlation (ZNCC) [Scandaroli et al., 2012] is, like the SCV, invariant to global illumination variations. Considering $\bar{I}$ as the average intensity value of I, the ZNCC of two images I and I^* is defined as:

$$ZNCC(I, I^*) = \sum_{k=1}^{N_\mathbf{x}} \frac{(I(\mathbf{x}_k) - \bar{I})(I^*(\mathbf{x}_k) - \bar{I^*})}{\sigma_I \, \sigma_{I^*}}. \tag{2.24}$$

In equation (2.24), σ_I and σ_{I^*} are the standard deviations of I and I^*, which are obtained thanks to the following expression:

$$\sigma_I = \sqrt{\sum_{k=1}^{N_{\mathbf{x}}^I} (I(\mathbf{x}_k) - \bar{I})^2}. \tag{2.25}$$

The ZNCC was not considered in the reminder of this document since it adds complexity compared to the SCV but the results of both methods are very comparable, with good robustness to global variations and a convergence area very close to the SSD one.

2.2.1.4 Mutual Information (MI)

The mutual information, as defined by Shannon [Shannon, 2001], represents the quantity of information shared by two signals. It is not a difference based on intensities like the SSD, SCV or ZNCC but a similarity criterion based on the entropies of the considered sources. It is defined by:

$$\mathrm{MI}(I, I^*) = \mathrm{H}(I) + \mathrm{H}(I^*) - \mathrm{H}(I, I^*). \tag{2.26}$$

The entropy $\mathrm{H}(I)$ is a measure of the randomness of a random variable. Given a discrete variable I with a dynamic d, its entropy is given by the following equation:

$$\mathrm{H}(I) = -\sum_{r=0}^{d} p_I(r) \log\left(p_I(r)\right) \tag{2.27}$$

where $p_I(r)$ represents the probability distribution function of I (the probability for a given pixel of I to have an intensity r). Following the same principle, the joint entropy $\mathrm{H}(I, I^*)$ of two sources I and I^* is defined by:

$$\mathrm{H}(I, I^*) = -\sum_{r,t=0}^{d} p_{II^*}(r,t) \log\left(p_{II^*}(r,t)\right) \tag{2.28}$$

where $p_{II^*}(r,t)$ is the joint probability distribution function of I and I^*.

2.2.2 2D Registration

Dense 2D registration differs from 2D registration using geometrical features in the fact that it does not need any low-level matching or tracking layers. Those appearance-based methods aim at optimizing the (dis)similarity between the textures of a reference template I^* composed of N_x pixels and a patch extracted from the current image and transformed thanks to a warp function w:

$$\widehat{\mathbf{u}} = \arg\min_{\mathbf{u}} \sum_{n=0}^{N_{\mathbf{x}}} f(I^*(\mathbf{x}_n), I(w(\mathbf{x}_n, \mathbf{u}))) \tag{2.29}$$

where $w(\mathbf{x}, \mathbf{u})$ represents the point in the image plane resulting of the warping of $\mathbf{x}$ parameterized by the vector $\mathbf{u}$ and f a dissimilarity function.

2.2.2.1 Optimizing the sum of squared differences: KLT

The Kanade Lucas Tomasi visual tracking approach, often referred to as KLT [Lucas et al., 1981, Shi and Tomasi, 1994, Baker and Matthews, 2004], is the most commonly used algorithm in order to estimate the displacement of a template between two images by minimizing the sum of squared differences between a current and reference template. It is based on the fact that if the brightness constancy constraint is respected, meaning if $I_0(\mathbf{x}) = I_k(w(\mathbf{x}, \mathbf{0}))$, then we can estimate the transformation $w(\mathbf{x}, \mathbf{u})$ that best maps a pixel $\mathbf{x}$ in the first image of a sequence I_0 to a pixel $\mathbf{x}_k$ in the k^{th} image I_k of the sequence:

$$I_k(\mathbf{x}_k) = I_0(\mathbf{x}) = I_k(w(\mathbf{x}, \mathbf{u}_k)). \tag{2.30}$$

Writing the problem as equation (2.29) leads to:

$$\widehat{\mathbf{u}}_k = \arg\min_{\mathbf{u}_k} \sum_{n=0}^{N_\mathbf{x}} \left[I_k(w(\mathbf{x}_n, \mathbf{u}_k)) - I^*(\mathbf{x}_n) \right]^2 \tag{2.31}$$

Again, this is a non-linear optimization task since the pixel values of $I^*(\mathbf{x})$ evolve with no linear relation to their coordinates $\mathbf{x}$. To perform this task, non-linear optimization schemes must be used as detailed in Frame 1. Starting from an estimation $\mathbf{u}_k = \widehat{\mathbf{u}_{k-1}}$, an iterative process is performed in order to find the increment $\Delta\mathbf{u}_k$ that will update $\mathbf{u}_k$ into $\mathbf{u}_{k+1}$ and allow the process to move on to the next frame:

$$\widehat{\Delta\mathbf{u}_k} = \arg\min_{\Delta\mathbf{u}_k} \sum_{n=0}^{N_\mathbf{x}} \left[I_k(w(\mathbf{p}_n, \mathbf{u}_k + \Delta\mathbf{u}_k) - I^*(\mathbf{p}_n) \right]^2. \tag{2.32}$$

Performing a first order Taylor expansion on $w(\mathbf{p}_n, \mathbf{u}_k + \Delta\mathbf{u}_k)$ gives the SSD function:

$$SSD(\Delta\mathbf{u}_k) = \sum_{n=0}^{N_\mathbf{x}} \left[I_k(w(\mathbf{p}_n, \mathbf{u}_k)) + \nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} - I^*(\mathbf{p}_n) \right]^2. \tag{2.33}$$

where ∇I_k is the gradient of the image at the pixel $w(\mathbf{p}_n, \mathbf{u}_k)$. As the goal is to minimize this value, its derivative needs to be equal to zero, which leads to:

$$\frac{\partial SSD}{\partial \Delta\mathbf{u}_k} = 2 \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[I_k(w(\mathbf{p}_n, \mathbf{u}_k)) + \nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} - I^*(\mathbf{p}_n) \right] = 0. \tag{2.34}$$

Defining H as a Gauss-Newton approximation of the Hessian matrix of the SSD:

$$H = \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[\nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \right] \tag{2.35}$$

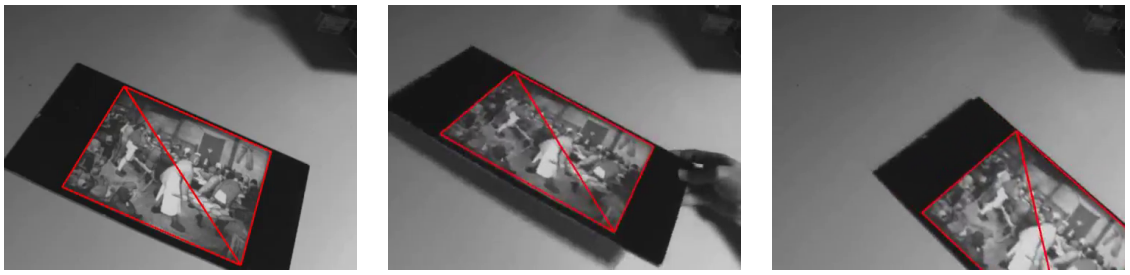


Figure 2.3: Example of visual tracking using the SSD as registration method [Baker and Matthews, 2004] using the ViSP library [Marchand et al., 2005].

Frame 3 - Classical warp update rules

As we explained before, non-linear optimization methods rely on a first guess of the good parameters that is iteratively updated until convergence. That warp update is the linchpin of the process and several strategies can be used.

Forward additional approach

The most simple and intuitive method is the forward additional approach [Lucas et al., 1981] which was presented with the KLT approach in section 2.2.2.1. The increment of warp parameter is then computed as:

$$\widehat{\Delta\mathbf{u}_k} = \arg\min_{\Delta\mathbf{u}_k} f\left(I_k(w(\mathbf{p}, \mathbf{u}_k + \Delta\mathbf{u}_k)), I^*(\mathbf{p})\right) \qquad \text{leading to} \qquad \mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \Delta\mathbf{u}_k. \tag{2.36}$$

The problem of this method is that it is not efficient to compute since the Jacobian matrix has to be updated at each iteration.

Forward compositional approach

Another way to perform the optimization is to use a forward compositional update rule:

$$\widehat{\Delta\mathbf{u}_k} = \arg\min_{\Delta\mathbf{u}_k} f\left(I_k(w(w(\mathbf{p}, \Delta\mathbf{u}_k)), \mathbf{u}_k), I^*(\mathbf{p})\right) \tag{2.37}$$

leading to an update rule which is expressed as:

$$w(\mathbf{p}, \mathbf{u}_{k+1}) \leftarrow w(w(\mathbf{p}, \Delta\mathbf{u}_k), \mathbf{u}_k). \tag{2.38}$$

This forward compositional method [Shum and Szeliski, 2000] is more intuitive and possesses a complexity that does not increase significantly over the additional approach, which makes it a common technique in visual tracking.

Inverse compositional approach

Inverse schemes [Baker and Matthews, 2001] allow an increased efficiency over forward schemes. This is due to the fact that instead of aiming at aligning a current image on a reference image, they align the template on a projection of the current image:

$$\widehat{\Delta\mathbf{u}_k} = \arg\min_{\Delta\mathbf{u}_k} f\left(I^*(w(w(\mathbf{p}, \Delta\mathbf{u}_k), \mathbf{u}_k)), I_k(\mathbf{p})\right) \tag{2.39}$$

leading to an update rule which is expressed as:

$$w(\mathbf{p}, \mathbf{u}_{k+1}) \leftarrow w(w^{-1}(\mathbf{p}, \Delta\mathbf{u}_k), \mathbf{u}_k). \tag{2.40}$$

This formulation allows an increase in efficiency, as computation of the Jacobian of the visual tracking can be done beforehand and not repeated at each new iteration.

the parameter update computation rule can be expressed as:

$$\Delta\mathbf{u}_k = H^{-1} \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I_k \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[I_k(w(\mathbf{p}_n, \mathbf{u}_k)) - I^*(\mathbf{p}_n) \right]. \tag{2.41}$$

The warp parameters $\mathbf{u}_k$ are then updated thanks to $\Delta\mathbf{u}_k$ and the process is iterated until convergence. The process is here considered to have converged when the norm of $\Delta\mathbf{u}_k$ is lower than a defined threshold. Let us notice that this optimization is formulated following

a forward additional warp update rule. Several other strategies can be used instead. Their details are exposed in Frame 3.

2.2.2.2 Optimizing the sum of conditional variance

Optimizing the SCV in order to perform visual tracking is close to the SSD approach. As proposed by the authors of [Richa et al., 2011], let us expose the SCV-based visual tracking algorithm. In order to explain the algorithm, details will be given using an efficient second-order minimization. Proposed by the authors of [Benhimane and Malis, 2007], this method allows a very efficient computation of an approximation of the Hessian matrix. This will allow to give an overview of this optimization technique. Let us add that a Gauss-Newton approach such as the one performed in the KLT is also possible, such as the second order approximation of the Hessian is also possible for a SSD approach. Let us start by defining the SCV-based visual tracking task. As stated in section 2.2.1.2, the measure can be defined as:

$$SCV(I, I^*) = \sum_{n=1}^{N_{\mathbf{x}}} \left[I(\mathbf{x}_n) - \widehat{I}^*(\mathbf{x}_n) \right]^2. \tag{2.42}$$

Here, the formulation is reversed, and it is the template that is adapted to the current image, resulting in:

$$SCV(I_k, I^*) = \sum_{n=1}^{N_{\mathbf{x}}} \left[I_k(w(\mathbf{x}_n, \mathbf{u}_k + \Delta\mathbf{u}_k)) - \widehat{I}^*(\mathbf{x_n}) \right]^2. \tag{2.43}$$

Performing a second order Taylor expansion of I about the warp parameters leads to:

$$\widehat{I}^*(\mathbf{p}_n) = I_k(w(\mathbf{p}_n, \mathbf{u}_k)) + \nabla I_k \frac{\partial w(\mathbf{p}_n, \mathbf{u}_k)}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} + \frac{1}{2} \frac{\partial^2 I_k(w(\mathbf{p}_n, \mathbf{u}_k))}{\partial \Delta\mathbf{u}^2} \Delta\mathbf{u}. \tag{2.44}$$

Performing another Taylor expansion, this time on the Jacobian matrix evaluated at $\mathbf{u} = \mathbf{0}$ gives:

$$\frac{\partial \widehat{I}^*(\mathbf{x})}{\partial \Delta\mathbf{u}} \approx \nabla I_k \frac{\partial w(\mathbf{p}_n, \mathbf{u}_k)}{\partial \Delta\mathbf{u}}. \tag{2.45}$$

Inputting equation (2.45) into equation (2.44) yields:

$$\widehat{I}^*(\mathbf{p}_n) = I_k(w(\mathbf{p}_n, \mathbf{u}_k)) + \frac{1}{2} \left[\nabla I_k \frac{\partial w(\mathbf{p}_n, \mathbf{u}_k)}{\partial \Delta\mathbf{u}} + \frac{\partial \widehat{I}^*(\mathbf{x})}{\partial \Delta\mathbf{u}} \right] \Delta\mathbf{u}. \tag{2.46}$$

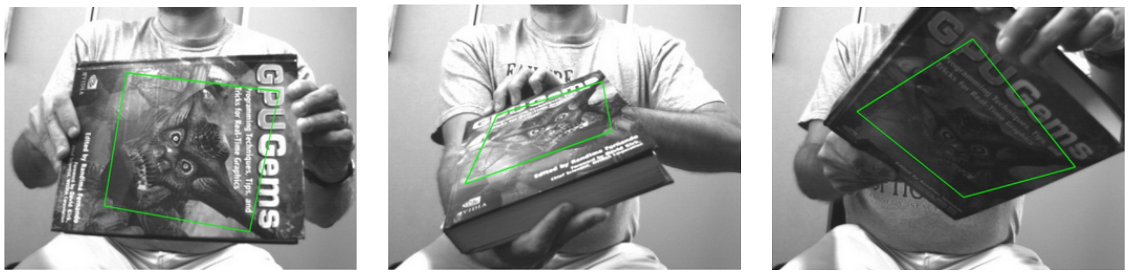


Figure 2.4: Example of visual tracking using the SCV as registration method. More details available in [Richa et al., 2011].

From this, the update parameters can be estimated thanks to:

$$\Delta\mathbf{u}_k = -2\left[\nabla I_k \frac{\partial w(\mathbf{p}_n, \mathbf{u}_k)}{\partial \Delta\mathbf{u}} + \frac{\partial \widehat{I^*}(\mathbf{x})}{\partial \Delta\mathbf{u}}\right]^+ \left(I_k(w(\mathbf{p}_n, \mathbf{u}_k)) - \widehat{I^*}(\mathbf{p}_n)\right). \tag{2.47}$$

As for the SSD-based technique, the iterative process is stopped when the increment parameters of the warp become small enough that the optimization can be considered as having converged.

2.2.2.3 Optimizing the mutual information

Optimizing the mutual information between a template and a current image is more complex than the two optimization processes described earlier. This is mainly due to the fact that is does not rely on minimizing to zero a dissimilarity anymore but instead it consists in optimizing to an unknown value a similarity function. It also relies on being able to derive the probability density functions with relation to the warp parameters. As defined in [Viola and Wells, 1997] and latter extended in [Dame and Marchand, 2012], the registration task using the mutual information can be defined:

$$\widehat{\Delta\mathbf{u}_k} = \arg\max_{\Delta\mathbf{u}_k} \sum_{i,j} p_{II^*}(i,j) \log\left(\frac{p_{II^*}(i,j)}{p_I(i)p_{I^*}(j)}\right) \tag{2.48}$$

where $p_I(i)$ and $p_{I^*}(j)$ are respectively the marginal probability density functions of I and I^* and $p_{II^*}(i,j)$ is the joint probability density function of I and I^*. The Jacobian of the mutual information $\mathbf{G}$:

$$\mathbf{G} = \left.\frac{\partial MI(I_k(w(\mathbf{x}, \mathbf{u}_k \circ \Delta\mathbf{u}_k)), I^*)}{\partial \Delta\mathbf{u}}\right|_{\Delta\mathbf{u}=\mathbf{0}} \tag{2.49}$$

can be obtained from the derivation of equation (2.48) [Dowson and Bowden, 2006]:

$$\mathbf{G} = \sum_{i,j} \frac{\partial p_{II^*}(i,j)}{\partial \Delta\mathbf{u}} \left(1 + \log\left(\frac{p_{II^*}(i,j)}{p_{I^*}(j)}\right)\right). \tag{2.50}$$

Let us point out that the formulation $w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u})$ is used instead of $w(w(\mathbf{x}, \Delta\mathbf{u}), \mathbf{u}))$ to represent the forward compositional warp update rule for clarity purposes. From equation (2.50), it is clear that the derivative of the joint probability density function with

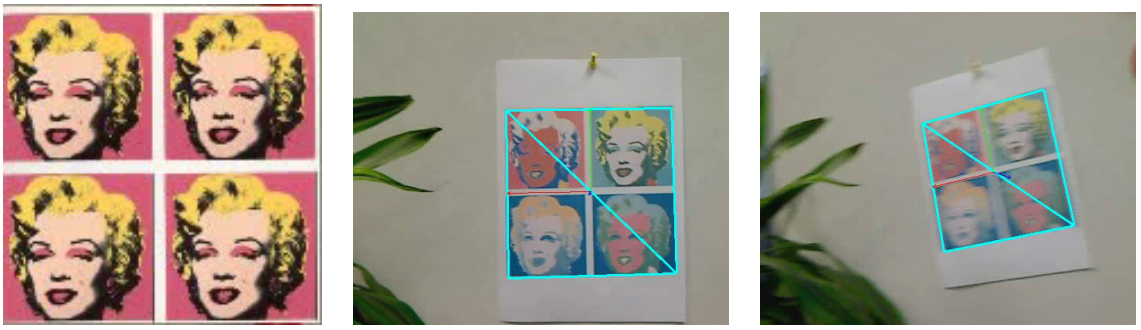


Figure 2.5: Visual tracking results using the MI approach. On the left is the reference image which is of a different modality with relation to the tracked template, which shows that the measure is naturally robust to variations of the modality of the images.

relation to the warp parameters is needed for the optimization. In order to be able to get those derivatives, histogram binning is performed when computing the probability density functions. The details of the process are detailed in Frames 4 and 5. Considering B-spline histogram binning, the joint probability becomes:

$$p_{\overline{II^*}}(i,j) = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \beta(i - \overline{I}(w(\mathbf{x}, \mathbf{u}_k \circ \Delta\mathbf{u}_k))) \, \beta(j - \overline{I^*}(\mathbf{x})) \tag{2.51}$$

where $\beta(i)$ represents a B-spline function (see Frame 4 for details). The value of $\beta(j - \overline{I^*}(\mathbf{x}))$ being the same no matter $\Delta\mathbf{u}$, the derivative needed in equation (2.50) can be expressed as:

$$\frac{\partial p_{\overline{II^*}}(i,j)}{\partial \Delta\mathbf{u}} = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \frac{\partial \beta(i - \overline{I}(w(\mathbf{x}, \mathbf{u}_k \circ \Delta\mathbf{u}_k)))}{\partial \Delta\mathbf{u}} \, \beta(j - \overline{I^*}(\mathbf{x})). \tag{2.52}$$

The use of B-spline functions means that this derivative is easy to compute as it is given by:

$$\begin{aligned}
\frac{\partial \beta(i - \overline{I}(w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u})))}{\partial \Delta\mathbf{u}} &= -\frac{\partial \beta}{\partial i} \, \frac{\partial \overline{I}(w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u}))}{\partial \Delta\mathbf{u}} \\
&= -\frac{\partial \beta}{\partial i} \, \nabla\overline{I} \, \frac{\partial w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u})}{\partial \Delta\mathbf{u}}\bigg|_{\Delta\mathbf{u}=0} \\
&= -\frac{\partial \beta}{\partial i} \, \nabla\overline{I} \, \frac{\partial w(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \, \frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \mathbf{x}}\bigg|_{\Delta\mathbf{u}=0} \\
&= -\frac{\partial \beta}{\partial i} \, \nabla\overline{I}(w(\mathbf{x}, \mathbf{u})) \, \frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \mathbf{x}}\bigg|_{\Delta\mathbf{u}=0}
\end{aligned} \tag{2.53}$$

The difference between $\nabla\overline{I}$ and $\nabla\overline{I}(w(\mathbf{x}, \mathbf{u}))$ is important here. The first is the gradient of the image $\overline{I}$ evaluated at the position $w(\mathbf{x}, \mathbf{u})$ whereas $\nabla\overline{I}(w(\mathbf{x}, \mathbf{u}))$ is the gradient of the warped image $\overline{I}(w(\mathbf{x}, \mathbf{u}))$. $\frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \mathbf{x}}$ is the derivative of a warped point with relation to the warp parameters used to transform its coordinates. As it needs only to be computed considering $\Delta\mathbf{u} = 0$ thanks to the compositional scheme, it is only dependent on the coordinates of the point in the reference image. Let us take for example the case of a homography warp presented in section 1.2.4. The transformation of a point $\mathbf{x} = [x \; y]^\top$ is given by:

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 + \Delta u_0 & \Delta u_1 & \Delta u_2 \\ \Delta u_3 & 1 + \Delta u_4 & \Delta u_5 \\ \Delta u_6 & \Delta u_7 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.54}$$

which yields:

$$x_w = \frac{(1 + \Delta u_0) \, x + \Delta u_1 \, y + 1}{\Delta u_6 \, x + \Delta u_7 \, y + 1} \qquad \text{and} \qquad y_w = \frac{\Delta u_3 \, x + (1 + \Delta u_4) \, y + 1}{\Delta u_6 \, x + \Delta u_7 \, y + 1} \tag{2.55}$$

The coordinates of the warped point can therefore be written as:

$$w(\mathbf{x}, \Delta\mathbf{u}) = \frac{1}{\Delta u_6 \, x + \Delta u_7 \, y + 1} \begin{bmatrix} (1 + \Delta u_0) \, x + \Delta u_1 \, y + \Delta u_2 \\ \Delta u_3 \, x + (1 + \Delta u_4) \, y + \Delta u_5 \end{bmatrix} \tag{2.56}$$

Setting $x' = ((1 + \Delta u_0) \, x + \Delta u_1 \, y + \Delta u_2)$, $y' = (\Delta u_3 \, x + (1 + \Delta u_4) \, y + \Delta u_5)$ and deriving for every parameters allows us to write:

$$\frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \mathbf{x}} = \frac{1}{\Delta u_6 \, x + \Delta u_7 \, y + 1} \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x \, x' & -y \, x' \\ 0 & 0 & 0 & x & y & 1 & -x \, y' & -y \, y' \end{bmatrix}. \tag{2.57}$$

Frame 4 - HISTOGRAM BINNING

Histogram binning is very important when manipulating image histograms in computer vision. The most natural way to compute empirical image probability density functions is to use a Kronecker function to test the value of each pixel over every possible value:

$$p_I(i) = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \delta(i - I(\mathbf{x})) \tag{2.58}$$

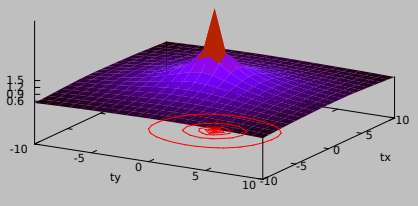
where $\delta(x) = 1$ when $x = 0$, 0 elsewhere. A problem arises from this formulation. A non-negligible number of bins of the histogram will likely be empty if we consider the whole dynamic of the image for the histogram. To overcome the possibility of empty bins in the histogram, a simple solution is to perform a dynamic scaling of the image. Let us consider the dynamic d_I of an image I, which will usually be 256 for a grey level image. In order to reduce the number of bins in the histogram, a scaling can be done:

$$\overline{I}(\mathbf{x}) = I(\mathbf{x}) \frac{d_{\overline{I}} - 1}{d_I - 1} \tag{2.59}$$

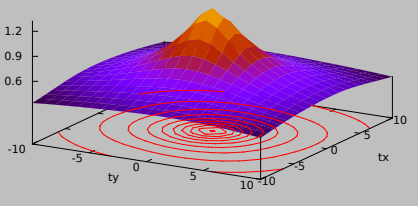
where the scaled image $\overline{I}$ has a dynamic of $d_{\overline{I}}$. Let us add that the formulation of equation (2.58) is no longer adequate to the computation of probability density functions on $\overline{I}$. This is due to the fact that the scaled image is now essentially composed of real values instead of integers as grey levels. Using a Kronecker function here would therefore result in the loss of an important amount of data. This is why an interpolation method is required. Several approaches have been proposed as using Gaussian functions [Viola and Wells, 1997] or approaching Gaussian functions with B-splines [Maes et al., 1997]. The simplicity of the latter method makes it the one we will use, resulting in a probability density function estimation given by:

$$p_I(i) = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \beta(i - \overline{I}(\mathbf{x})) \tag{2.60}$$

Further details on how to compute and differentiate B-spline functions are reminded in Frame 5. Histogram binning is very useful when computing the mutual information. It allows a smoothing of the cost function, depicted by the following mutual information representation:



MI computed using 256 bins



MI computed using 12 bins

For more details on the computation of the shape of the mutual information, please refer to section 2.3.

Frame 5 - B-SPLINE INTERPOLATION

B-splines are very useful for interpolation purposes. Defining B_n as the centered cardinal B-spline function of order n, B_n can be expressed through a recursive formulation. Let us first expose the first order B-spline function B_1:

$$B_1(x) = \begin{cases} 1 & \text{if } \| x \| < 0.5 \\ 0 & \text{elsewhere} \end{cases} \tag{2.61}$$

Every other order B-spline function is then defined as:

$$B_n(x) = (B_{n-1} * B_1)(x) \tag{2.62}$$

where $*$ represents a convolution. The results are, for the example of the lower orders B_2 and B_3:

$$B_2(x) = \begin{cases} 1+x & \text{if } x \in [-1, 0[\\ 1-x & \text{if } x \in [0, -1[\\ 0 & \text{elsewhere} \end{cases} \tag{2.63}$$

$$B_3(x) = \begin{cases} \frac{1}{2}(1.5+x)^2 & \text{if } x \in [-1.5, -0.5[\\ 1+x-(0.5+x)^2 & \text{if } x \in [-0.5, 0] \\ 1-x-(0.5-x)^2 & \text{if } x \in [0, 0.5] \\ \frac{1}{2}(1.5-x)^2 & \text{if } x \in [0.5, 1.5[\\ 0 & \text{elsewhere} \end{cases} \tag{2.64}$$

The derivatives of the B-splines can also be expressed with relation to lower order B-spline functions:

$$\frac{\partial B_n(x)}{\partial x} = B_{n-1}\left(x - \frac{1}{2}\right) - B_{n-1}\left(x + \frac{1}{2}\right). \tag{2.65}$$

When $\Delta\mathbf{u} = \mathbf{0}$, meaning the warp is an identity warp, $x' = x$, $y' = y$ and the denominator is equal to 1, leading to:

$$\left.\frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \mathbf{x}}\right|_{\Delta\mathbf{u}=0} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x^2 & -yx \\ 0 & 0 & 0 & x & y & 1 & -xy & -y^2 \end{bmatrix}. \tag{2.66}$$

To optimize the mutual information, Newton's method of optimization is used in [Dame and Marchand, 2012]. The process maximizes the MI by minimizing its gradient to zero. Using a Taylor expansion, the gradient of the MI can be expressed as:

$$\frac{\partial MI(\mathbf{u} \circ \Delta\mathbf{u})}{\partial \Delta\mathbf{u}} \simeq \left.\frac{\partial MI(\mathbf{u} \circ \Delta\mathbf{u})}{\partial \Delta\mathbf{u}}\right|_{\Delta\mathbf{u}=0} + \left.\frac{\partial^2 MI(\mathbf{u} \circ \Delta\mathbf{u})}{\partial \Delta\mathbf{u}^2}\right|_{\Delta\mathbf{u}=0} \Delta\mathbf{u}. \tag{2.67}$$

Denoting the Hessian matrix $\left.\frac{\partial^2 MI(\mathbf{u}\circ\Delta\mathbf{u})}{\partial \Delta\mathbf{u}^2}\right|_{\Delta\mathbf{u}=0}$ as $\mathbf{H}$, the parameter increment can therefore be expressed as:

$$\Delta\mathbf{u} = -\mathbf{H}^{-1}\mathbf{G}. \tag{2.68}$$

The Hessian matrix of the mutual information can, similarly its gradient, be expressed thanks to the partial derivatives of the probability density functions:

$$\left.\frac{\partial^2 MI(\mathbf{u} \circ \Delta\mathbf{u})}{\partial \Delta\mathbf{u}^2}\right|_{\Delta\mathbf{u}=0} = \sum_{i,j} \frac{\partial p_{II^*}}{\partial \Delta\mathbf{u}}^\top \frac{\partial p_{II^*}}{\partial \Delta\mathbf{u}} \left(\frac{1}{p_{II^*}} - \frac{1}{p_{I^*}}\right) + \frac{\partial^2 p_{II^*}}{\partial \Delta\mathbf{u}^2} \left(1 + \log\left(\frac{p_{II^*}}{p_{I^*}}\right)\right). \tag{2.69}$$

where $p_{II^*}(i,j)$ is noted p_{II^*} for clarity purposes. The B-spline functions being twice differentiable, the second order derivative of the joint probability density function is given by:

$$\frac{\partial^2 p_{II^*}}{\partial \Delta \mathbf{u}^2} = \frac{1}{N_\mathbf{x}} \sum_\mathbf{x} \frac{\partial^2 \beta(i - \overline{I}(w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u})))}{\partial \Delta \mathbf{u}^2} \beta(j - \overline{I^*}(\mathbf{x})). \tag{2.70}$$

As for the first order derivative, chain derivation gives the expression of the second order derivative:

$$\begin{aligned} \frac{\partial^2 \beta(i - \overline{I}(w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u})))}{\partial \Delta \mathbf{u}^2} &= \frac{\partial}{\partial \Delta \mathbf{u}} \left[-\frac{\partial \beta}{\partial i} \frac{\partial \overline{I}(w(\mathbf{x}, \mathbf{u} \circ \Delta\mathbf{u}))}{\partial \Delta \mathbf{u}} \right] \\ &= \frac{\partial^2 \beta}{\partial i^2} \frac{\partial \overline{I}}{\partial \Delta \mathbf{u}}^\top \frac{\partial \overline{I}}{\partial \Delta \mathbf{u}} - \frac{\partial \beta}{\partial i} \frac{\partial^2 \overline{I}}{\partial \Delta \mathbf{u}^2}. \end{aligned} \tag{2.71}$$

where:

$$\begin{aligned} \frac{\partial^2 \overline{I}}{\partial \Delta \mathbf{u}^2} &= \frac{\partial}{\partial \Delta \mathbf{u}} \left(\nabla \overline{I}(w(\mathbf{x}, \mathbf{u})) \frac{\partial \overline{I}(w(\mathbf{x}, \Delta\mathbf{u}))}{\partial \Delta \mathbf{u}} \right) \\ &= \frac{\partial w}{\partial \Delta \mathbf{u}}^\top \nabla^2 \overline{I}(w(\mathbf{x}, \mathbf{u})) \frac{\partial w}{\partial \Delta \mathbf{u}} + \begin{bmatrix} \nabla_x \overline{I}(w(\mathbf{x}, \mathbf{u})) & 0 \\ 0 & \nabla_y \overline{I}(w(\mathbf{x}, \mathbf{u})) \end{bmatrix} \begin{bmatrix} \frac{\partial^2 w_x}{\partial \Delta \mathbf{u}^2} \\ \\ \frac{\partial^2 w_y}{\partial \Delta \mathbf{u}^2} \end{bmatrix} \end{aligned} \tag{2.72}$$

with $\nabla^2 \overline{I}(w(\mathbf{x}, \mathbf{u}))$ the Hessian of the warped image $\overline{I}(w(\mathbf{x}, \mathbf{u}))$ with respect to the warp parameters $\Delta\mathbf{u}$ and $\frac{\partial^2 w}{\partial \Delta \mathbf{u}^2}$ the Hessian of the warp function with relation to the parameters $\Delta\mathbf{u}$.

2.3 Robustness of the similarity functions

To assess the robustness of the different similarity functions detailed previously, a comparison was made on the results they gave on a chosen picture. Starting from a set position, we first extract a patch of 400×400 pixels (see Fig. 2.6). We then proceed to move a window around that position thanks to a 2D translation vector $\mathbf{t} = (tx, ty)^\top$, starting at a relative position of (-10,-10) pixels from the start. At each position the four (dis)similarity functions are computed and the results are plotted to create a 3D shape in order to evaluate the optimization possibilities given by each function.

2.3.1 Nominal case

The first experiment was done in order to have a basis for comparison. In order to do that, we evaluated the four detailed functions (SSD, SCV, ZNCC, MI) in nominal conditions. The template and considered patch are therefore extracted from the same images in order to show the shape of the measures with relation to the translation parameters in nominal conditions. The results are detailed on figure 2.7. They show that the optimum for each measure is located at $\mathbf{t} = \mathbf{0}$ which was to be expected in nominal conditions. The four shapes of the cost functions present smooth surface with no local minima or maxima which confirms the fact that these functions are well fitted for optimization procedures and can therefore be used for visual tracking without any supplementary robustness layer in these conditions.

Figure 2.6: Setup of the comparative study. At each position correspond a new patch to evaluate the functions.

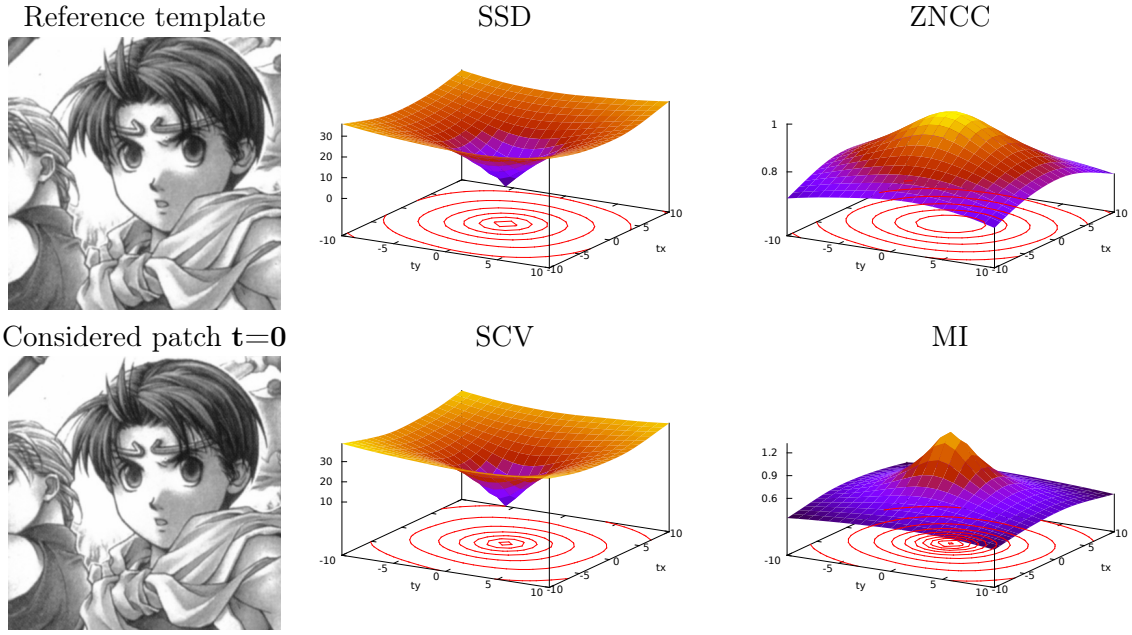


Figure 2.7: Evaluation of the four detailed registration functions in nominal conditions.

2.3.2 Noisy images

The next experiment was realized in order to assess the robustness of every measure with respect to image noise. Image noise is often present in real-life visual tracking applications, as cameras often struggle to present perfect image quality either due to physical (sensor quality) or software (image coding) reasons. This is why this time white Gaussian noise was added to the template image before extracting the patches. Figure 2.8 shows the

corresponding results for each function. We can see that all four functions are robust to white Gaussian noise. Even if they are indeed impacted as their optima are less pronounced, the four smooth shapes show global minima/maxima located on $\mathbf{t} = \mathbf{0}$ with no local minima/maxima.

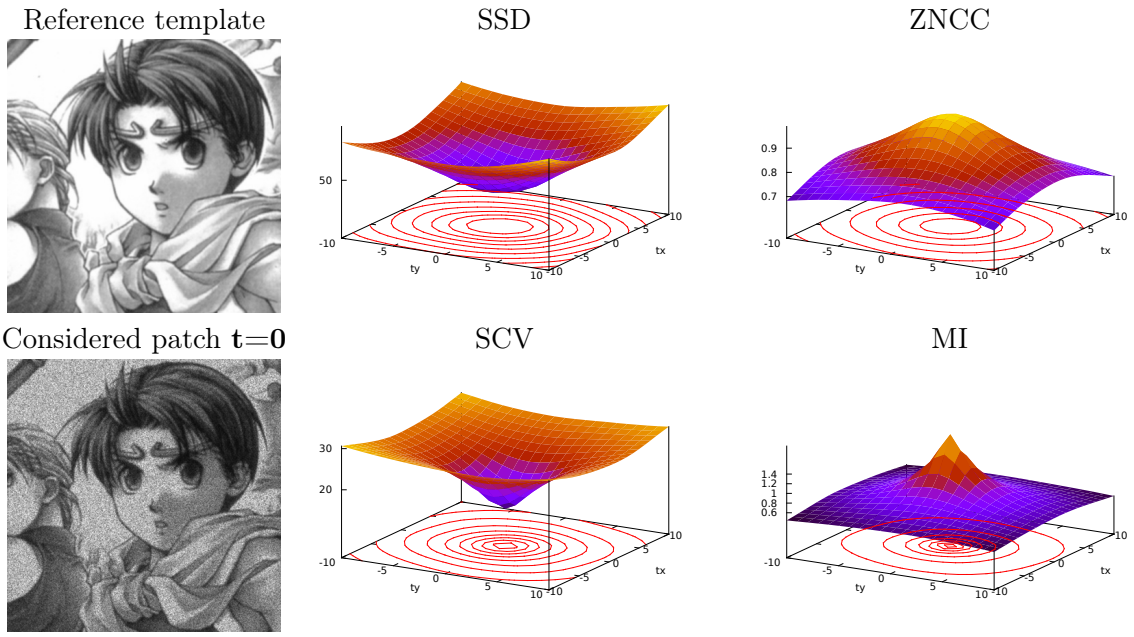


Figure 2.8: Evaluation of the four detailed registration functions when confronted to white gaussian noise.

2.3.3 Global light variations

Another perturbation that can occur in real-life visual tracking is light variation. Global variations need to be dealt with as they are a common phenomenon and can occur from one frame to another as the atmospheric conditions change. To simulate that perturbation, the template image was recovered of a grey filter in order to simulate a shift in luminosity. The experiment was launched again and the results can be seen on figure 2.9. The shape of the SSD becomes problematic in these conditions. It shows no minimum and the variations are very weak. Moreover the curve of the shape clearly goes toward a lighter zone positioned upper than the window position, which means that any optimization for visual tracking would diverge towards that part of the image. It is then obviously not fit for optimization in these conditions without added robustness layers. The other three functions however, as they represent measures invariant to global variations are still perfectly fit for optimization (no local optimum, global optimum located for $\mathbf{t} = \mathbf{0}$) even though their values are impacted by the perturbation.

2.3.4 Local light variations: specularities

Another problem often caused by the lighting of the scene is called specularities. They occur when a beam of light touches a non-lambertian surfaces. Those surfaces, which encompass most surfaces, display a white spot when the light touches them according to a certain angle. The spot is often called specularity as displayed on figure 2.10. This is what we simulated in this experiment in order to evaluate the natural robustness of the four

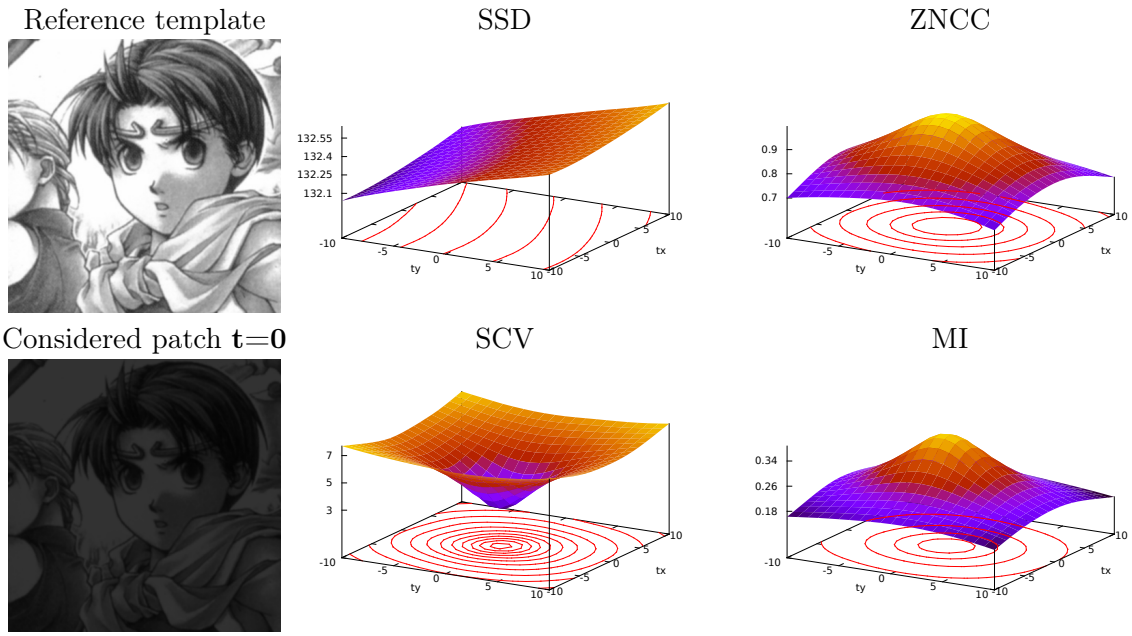


Figure 2.9: Evaluation of the four detailed registration functions when confronted to global light variations.

considered functions in these conditions. The results show that SSD is not able to cope with such perturbations. It also shows that both SCV and ZNCC are at their limit with a very little slope in those conditions which means that most specularities would prove to be problematic. Finally, MI is here again the most robust function since the specularity did not significantly alter the shape of the measure.

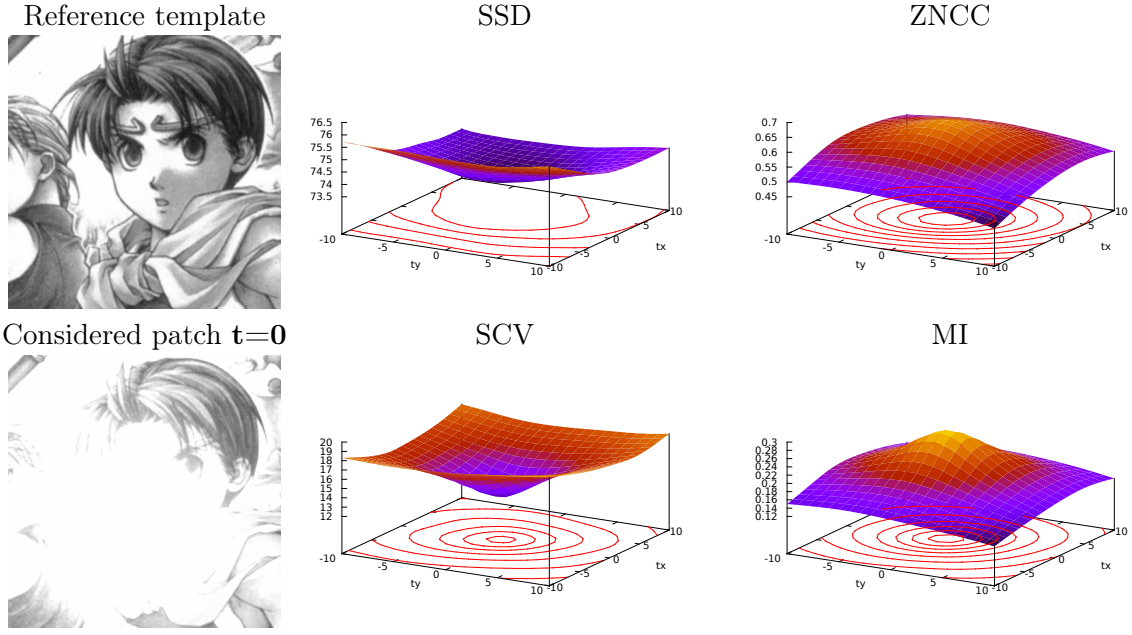


Figure 2.10: Evaluation of the four detailed registration functions when confronted to local light variations.

2.3.5 Occlusions

When tracking an object, occlusions can also happen. They are often due to other objects moving along a different trajectory than the followed template. According to their size, occlusions can be problematic for most (dis)similarity functions.To assess the behaviour of the functions with relative to occlusions images have been added over the current view in order to mask parts of the reference with different textures. Results shown on figure 2.11. We can see that with the exception of the mutual information, and even though a moderate part of the template is still visible, the three other similarity functions are thrown of by the perturbation. The SSD still has a global minimum due to the fact that the grey levels present in the occlusions are close to the background but a plateau is appearing and the scale of variation is minimal, which means that in that situation there are very few chances for the optimization to reach a good position. Both the SCV and ZNCC are lost and unusable as such in these conditions, as they try to globally adapt to a local perturbation, which cannot work well.

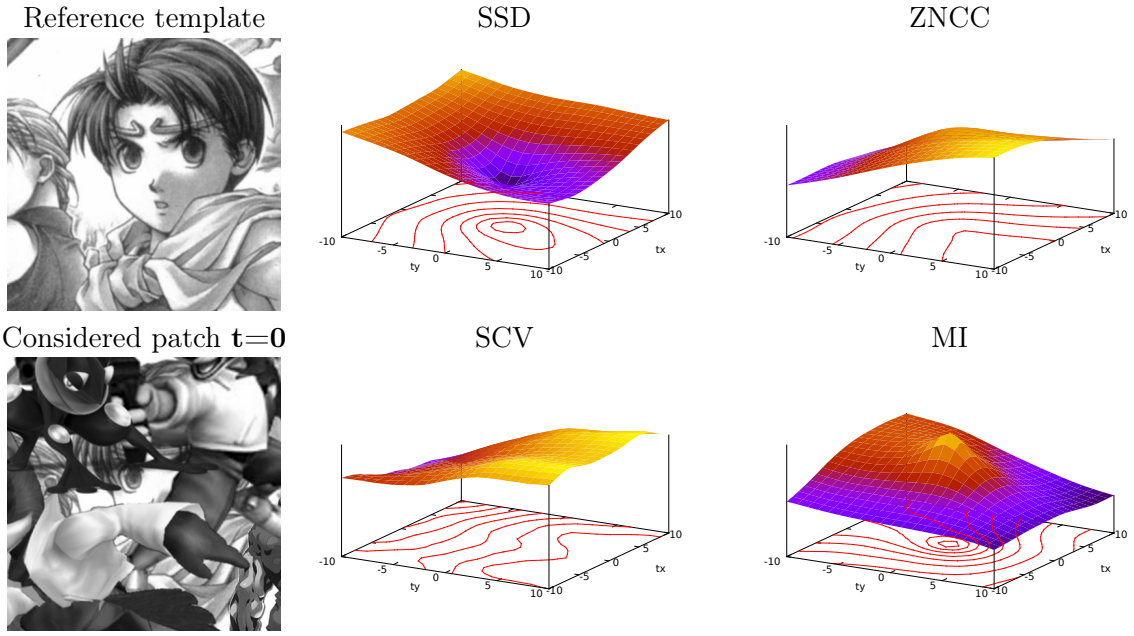


Figure 2.11: Evaluation of the four detailed registration functions when confronted to occlusions.

2.3.6 Different modalities

The final experiment was realized in order to assess the possibilities of the registration functions when confronted to images presenting different modalities. Being robust to multi modalities is very interesting as it means for example being able to continue tracking a target on infrared pictures or being able to compare for example maps and satellite images. This is why in the experiment we simulated a IR image of the template thanks to an image manipulation software dedicated to such manipulations, see figure 2.12 for more details, and evaluated our functions in those conditions. Results show clearly that both SCV, ZNCC and MI are able to cope with that perturbation as they are able to adapt to such situations contrary to the SSD which is completely thrown off by the variation of histogram.

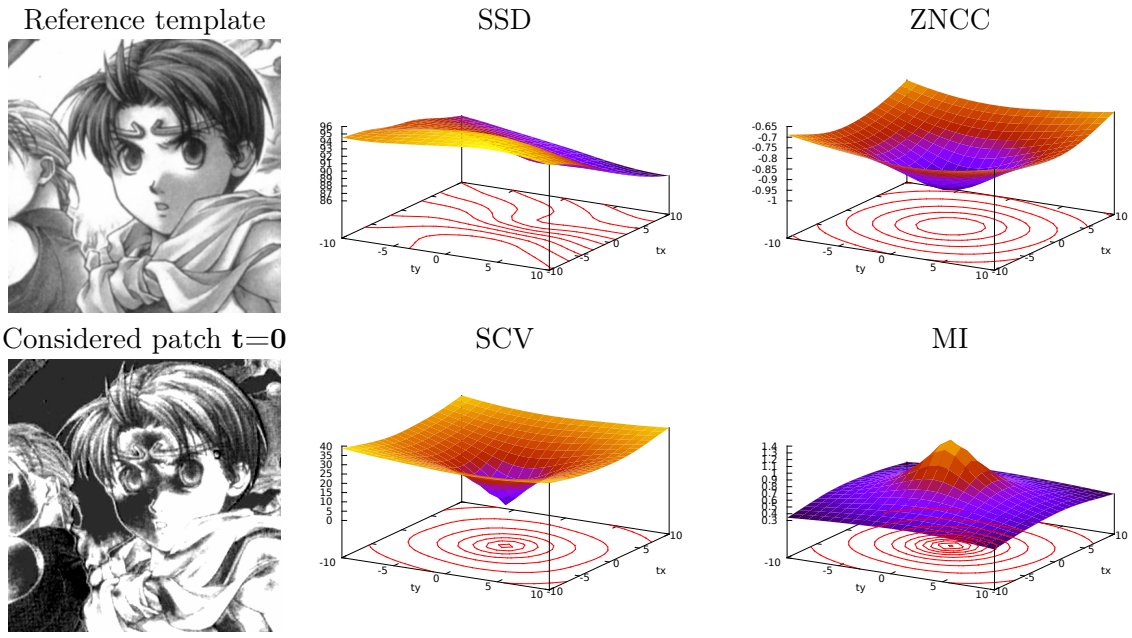


Figure 2.12: Evaluation of the four detailed registration functions on images with different modalities.

2.4 Conclusion

In this chapter, several methods proposed in the literature to perform image registration have been discussed. Starting from the classical techniques based on sparse geometrical features and extending to dense methods that use directly the whole image information, several visual tracking algorithms have been detailed. Then, (dis)similarity functions have been discussed, and four of them have been compared in different conditions. From the results it appears that the SCV is a good compromise between robustness and simplicity, whereas the MI appears to be very robust in numerous conditions, albeit more complex. Building on these results, the next chapters will detail new visual tracking approaches based on these (dis)similarity functions.

3

Model-based dense visual tracking

The objective of this chapter is to propose a new dense model-based approach to 3D visual tracking (or 3D localization) that is robust to global variations. To that end, an algorithm proposed in the literature that allows to track several planes linked by euclidean constraints is first detailed. Then, it is extended to optimize robust (dis)similarity functions and to take into account several faces of the object in the same optimization loop in order to estimate the parameters of a pose vector $\mathbf{r} \in SE(3)$. First, the SCV is used to propose an approach that is simple, since it is close to the SSD case, and robust to global variations of the scene. Then, the method is extended to optimize the mutual information. Though more complex and more computationally challenging, this approach is very robust as the MI is robust to numerous perturbations. We then explain how to take into account a 3D model by detecting which faces are visible by the camera. Only the visible face are then used in the optimization process. The two proposed algorithms are then validated and compared in several conditions before an analysis of their convergence domain is realized.

3.1 3D Registration: Pose estimation

In [Benhimane and Malis, 2006b], a method that includes euclidean constraints between several planes into the tracking task has been proposed. In order to do that, the optimization is done on 6 parameters representing a 3D transformation in SE(3) using the sum of squared differences (SSD) as the registration function. Considering a template I^* of size $N_{\mathbf{x}}$ pixels representing the projection of a plane in the 3D frame. The tracking process then consists in finding the parameters of the transformation $\mathbf{T}(\mathbf{r})^k \in \mathrm{SE}(3)$ translating the displacement of the considered scene at the iteration k. Considering an inverse compositional approach [Baker and Matthews, 2001], the goal of the optimization is to find the optimal increment of parameters $\boldsymbol{\Delta}\mathbf{r}$ which verifies $\forall \mathbf{x}_i \in I$:

$$\begin{aligned} I(w(\mathbf{x}_i, \mathbf{r}^{k-1})) &= I^*(w^{-1}(\mathbf{x}_i, \boldsymbol{\Delta}\mathbf{r})) \\ &= I^*(\mathbf{G}(\boldsymbol{\Delta}\mathbf{r})^{-1}\mathbf{x}). \end{aligned} \tag{3.1}$$

The 3D transformation $\mathbf{T}(\mathbf{r})$ being the same for every plane in the scene the process can therefore track several planes in a unique optimization loop, the difference between planes coming from the different matrices $\mathbf{G}$ induced by $\boldsymbol{\Delta}\mathbf{r}$. The homography warps not being the same for every plane, the warp associated to a plane l will be noted w_l for the reminder of this document. The optimal increment of parameters $\widehat{\boldsymbol{\Delta}\mathbf{r}}$ is obtained by minimizing the SSD between the current image warped with the displacement parameters computed at

the last frame $\mathbf{r}$ and the template warped with current parameters $\widehat{\Delta\mathbf{r}}$:

$$\widehat{\Delta\mathbf{r}} = \arg\min_{\Delta\mathbf{r}} \sum_{l} \sum_{i=1}^{N_{\mathbf{x}_l}} \left[I^*(w_l(\mathbf{x}_i, \Delta\mathbf{r})) - I(w_l(\mathbf{x}_i, \mathbf{r}^{k-1})) \right]^2. \tag{3.2}$$

The displacement is subsequently updated as follows:

$$\mathbf{T}(\mathbf{r})^k \leftarrow \mathbf{T}(\mathbf{r})^{k-1} \mathbf{T}(\widehat{\Delta\mathbf{r}})^{-1}. \tag{3.3}$$

To perform this minimization as was proposed in [Baker and Matthews, 2001] in SL(3) and [Benhimane and Malis, 2006b] in SE(3), let us start by expressing the first order Taylor expansion associated to the chosen registration function:

$$SSD(\Delta\mathbf{r}) = \sum_{l} \sum_{i=1}^{N_{\mathbf{x}_l}} \left[I^*(w_l(\mathbf{x}_i, \Delta\mathbf{r})) - I(w_l(\mathbf{x}_i, \mathbf{r}^{k-1})) \right]^2 \tag{3.4}$$

which is defined as:

$$SSD(\Delta\mathbf{r}) \simeq \sum_{l} \sum_{i=1}^{N_{\mathbf{x}_l}} \left[I^*(\mathbf{x}_i) - I(w_l(\mathbf{x}_i, \mathbf{r}^{k-1})) \right]^2 + \mathbf{J}(\Delta\mathbf{r})\Delta\mathbf{r} \tag{3.5}$$

where $\mathbf{J}(\Delta\mathbf{r})$ is the Jacobian matrix of $SSD(\Delta\mathbf{r})$.

Decomposing the Jacobian matrix thanks to the different transformations applied to each pixel gives:

$$\begin{aligned} \mathbf{J}(\Delta\mathbf{r}) &= \frac{\partial I^*}{\partial w_l} \frac{\partial w_l}{\partial \mathbf{K}} \frac{\partial \mathbf{K}}{\partial \mathbf{T}} \frac{\partial \mathbf{T}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \Delta\mathbf{r}} \\ &= \mathbf{J}_{I^*} \mathbf{J}_{w_l} \mathbf{J}_{\mathbf{K}} \mathbf{J}_{\mathbf{T}} \mathbf{J}_{\mathbf{x}}(\Delta\mathbf{r}) \end{aligned} \tag{3.6}$$

leading to [Benhimane and Malis, 2006b]:

$$\widehat{\Delta\mathbf{r}} = -\left(\mathbf{J}_{I^*} \mathbf{J}_{w_l} \mathbf{J}_{\mathbf{K}} \mathbf{J}_{\mathbf{T}} \mathbf{J}_{\mathbf{x}}(\mathbf{0})\right)^+ SSD(\mathbf{0}). \tag{3.7}$$

Let us note that an ESM approach can also be chosen to perform the optimization as it was chosen in [Benhimane and Malis, 2006b]. In this eventuality, the update of the displacement is given by:

$$\widehat{\Delta\mathbf{r}} = -\left(\left(\frac{\mathbf{J}_I + \mathbf{J}_{I^*}}{2} \right) \mathbf{J}_{w_l} \mathbf{J}_{\mathbf{K}} \mathbf{J}_{\mathbf{T}} \mathbf{J}_{\mathbf{x}}(\mathbf{0}) \right)^+ SSD(\mathbf{0}). \tag{3.8}$$

The problem when using the SSD as the registration function of a registration process as performed in [Benhimane and Malis, 2006a] is the fact that it is not adapted to the perturbations that are usually undergone by real life scenes such as illumination variations or occlusions. This is why we will present here two 3D dense registration methods. The first one is based on the Sum of conditional variance and the second on is based on the mutual information. The first approach, using the SCV, should therefore be robust to global light variations as possibly undergone by most real-life scenes but still quite easily performed while the MI version should add robustness at the expense of being more complex and possible more time consuming.

3.2 Optimization over several planes: visibility issue

3D models are usually made of sets of triangles or planes linked to one another by 3D transformations. Therefore, when dealing with dense registration of a 3D model, the algorithm needs to know at every frame which faces are visible by the camera. To illustrate the process, let us take for example a 3D model composed of two textured squares linked by a side segment at a 90° angle. In order to determine the visibility of a face, a simple test is done. It is necessary to compute the angle between the plane's normal vector (expressed in the camera plane) and a vector between the camera position and the center of the plane. Depending on this angle, the visibility can then be assessed. Let us note that visibility test does not take into account concave shapes, where occlusions are possible between frames. Figure 3.1 and 3.2 show the different possible cases. In the first case (Fig 3.2), only one

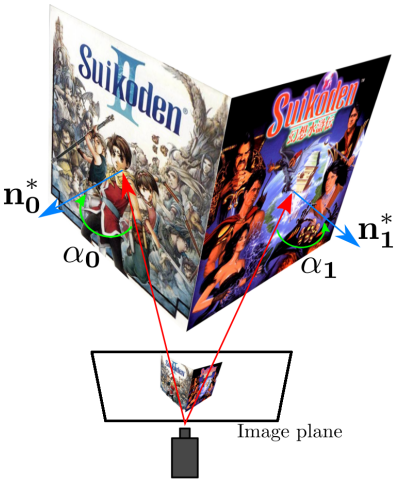


Figure 3.1: Example of two faces considered visible at the same time. The registration is then realized considering both of them.

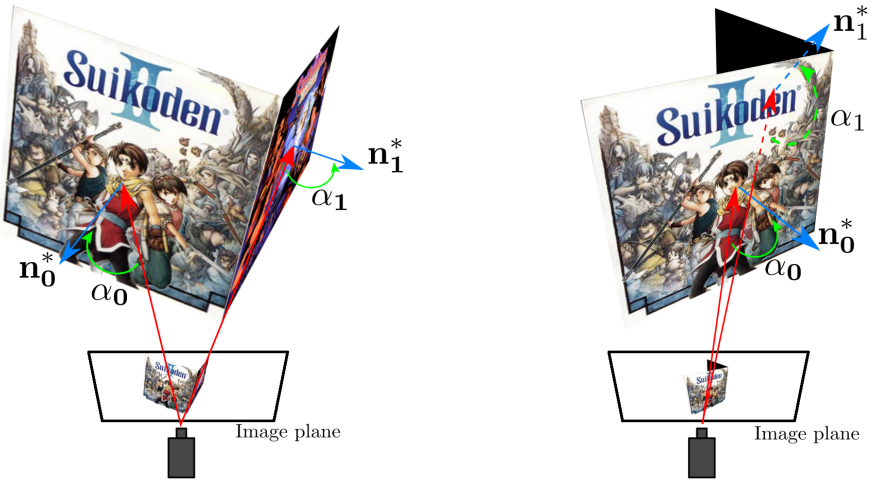


Figure 3.2: Two examples where only one plane is considered for the tracking process. On the left, one of the faces is not enough visible to be taken into account whereas on the right it is not seen at all by the camera.

plane is considered visible. On the right part of the figure, one face is totally occluded by the other and the angle between the camera line of view and the normal is then too important. On the left part of the figure, the face is seen in the image but our algorithm will not consider it visible as the computed angle is too narrow. This is done in order to insure a better registration quality. Indeed, an initialization of the reference template associated to the face needs to be determined when a new face appears in order to be included in the tracking process. But if the visibility of the texture is not sufficient, the reference taken into account for the registration will be of bad quality which will result in a bad registration process. This is why faces are considered visible only once the computed angle criterion has reached a minimal desired value. On the other hand, figure 3.1 shows a case where the two faces are seen and included in the registration process simultaneously. They are both considered visible since the value of the computed angle is in a determined window of visibility.

3.3 Using the SCV for dense model-based 3D registration

To perform dense 3D registration using the SCV, we chose to adapt the previous works proposed by the authors of [Benhimane and Malis, 2006a]. The basic principle therefore remains and the approach is reorganized and adapted to the use of the SCV as the cost function. The approach is still a minimization to zero of a cost function, the SCV, that needs to be performed over the parameters of the 3D pose of the camera $\mathbf{r}$. Considering a 3D model composed of several planes, a homography can be determined in the image plane for each of them. To do that, let us start from the definition of the 3D homography:

$$ {}^{b}\mathbf{X} = \mathbf{H}\,{}^{a}\mathbf{X} \qquad \text{where} \qquad \mathbf{H} = {}^{b}\mathbf{R}_a + \frac{{}^{b}\mathbf{t}_a\,\mathbf{n}^\top}{d} \tag{3.9} $$

A point expressed in the object frame $\mathbf{x}_l^o$ belonging to a plane l can therefore be expressed in the camera frame thanks to the homography given by:

$$ \mathbf{H}_l = {}^{c}\mathbf{R}_o + \frac{{}^{c}\mathbf{t}_o\,\mathbf{n}_l{}^\top}{d_l}. \tag{3.10} $$

Let us refer to this homography as $\mathbf{H}_l(\mathbf{r})$ where $\mathbf{r}$ is the pose vector translating the frame transformation allowing to know the homogeneous transformation ${}^{c}\mathbf{M}_o$ between F_o and F_c from which ${}^{c}\mathbf{R}_o$ and ${}^{c}\mathbf{t}_o$ can be extracted. Considering the camera parameters $\mathbf{K}$ known (we recover them through a well-known process of calibration [Tsai, 1986]), the homographies associated to each face of the model can be translated in the image plane as seen in chapter 1:

$$ \mathbf{G}_l(\mathbf{r}) = \mathbf{K}\,\mathbf{H}_l(\mathbf{r})\,\mathbf{K}^{-1}. \tag{3.11} $$

The pose vector $\mathbf{r}$ is the same for every plane l of the object, as it translates the position of the camera with relation to the object, but the homography $\mathbf{G}_l(\mathbf{r})$ is different for every plane, as it depends on its normal and distance to the origin. Each plane is therefore warped in the image plane according to a different warp w_l although all warps depend on the same pose vector $\mathbf{r}$:

$$ w_l(\bar{\mathbf{x}}, \mathbf{r}) = \mathbf{G}_l(\mathbf{r})\,\bar{\mathbf{x}}. \tag{3.12} $$

Now that the transformation function has been determined, let us detail the algorithm. Since the objective is to estimate the parameters of the relative pose between the camera

and the object $\mathbf{r}$, the task to optimize is given by:

$$\widehat{\Delta\mathbf{r}} = \arg\min_{\Delta\mathbf{r}} \sum_{l=1}^{N_l} SCV\left(w_l(I^*, \Delta\mathbf{r}), w_l(I, \mathbf{r})\right) \tag{3.13}$$

where $w(I, \mathbf{r})$ is a generalization made to translate the result of the warp of every point in the image I according to parameters $\mathbf{r}$: $\sum_{N_\mathbf{x}} I(w(\mathbf{x}, \mathbf{r}))$. Considering $\hat{I}(\mathbf{x}) = \mathcal{E}(I^*(\mathbf{x}) \mid I(\mathbf{x}))$, the cost function of the proposed approach expressed in equation (3.13) can be rewritten as:

$$\widehat{\Delta\mathbf{r}} = \arg\min_{\Delta\mathbf{r}} \sum_{l=1}^{N_l} \sum_{k=1}^{N_\mathbf{x}} \left[I^*(w_l(\mathbf{x}_k, \Delta\mathbf{r})) - \hat{I}(w_l(\mathbf{x}_k, \mathbf{r}))\right]^2. \tag{3.14}$$

To perform this optimization task, let us first express the Taylor expansion of the registration function in $\mathbf{r} = \mathbf{0}$. The registration function being the SCV:

$$SCV(\Delta\mathbf{r}) = \sum_{l=1}^{N_l} \sum_{k=1}^{N_\mathbf{x}} \left[I^*(w_l(\mathbf{x}_k, \Delta\mathbf{r})) - \hat{I}(w_l(\mathbf{x}_k, \mathbf{r}))\right]^2 \tag{3.15}$$

the Taylor expansion gives:

$$SCV(\Delta\mathbf{r}) \simeq \sum_{l=1}^{N_l} \sum_{k=1}^{N_\mathbf{x}} \left[I^*(w_l(\mathbf{x}_k, \mathbf{0})) - \hat{I}(w_l(\mathbf{x}_k, \mathbf{r}))\right]^2 + \mathbf{J}(\mathbf{0})\Delta\mathbf{r} + \frac{1}{2}\,\mathbf{H}(\mathbf{0}, \Delta\mathbf{r})\Delta\mathbf{r} \tag{3.16}$$

where $\mathbf{J}$ and $\mathbf{H}$ are respectively the Jacobian and Hessian matrix of the SCV given by:

$$\mathbf{J}(\Delta\mathbf{r}) = \frac{\partial SCV(\Delta\mathbf{r})}{\partial \Delta\mathbf{r}} \qquad \text{and} \qquad \mathbf{H}(\Delta\mathbf{r}, \Delta\mathbf{r}) = \frac{\partial^2 SCV(\Delta\mathbf{r})}{\partial \Delta\mathbf{r}^2}. \tag{3.17}$$

From the Taylor expansion of the Jacobian matrix evaluated around $\Delta\mathbf{r}$:

$$\mathbf{J}(\Delta\mathbf{r}) = \mathbf{J}(\mathbf{0}) + \frac{1}{2}\,\mathbf{H}(\mathbf{0}, \Delta\mathbf{r}) \tag{3.18}$$

we can express, combining equations (3.16) and (3.18) :

$$SCV(\Delta\mathbf{r}) \simeq \sum_{l=1}^{N_l} \sum_{k=1}^{N_\mathbf{x}} \left[I^*(w_l(\mathbf{x}_k, \mathbf{0})) - \hat{I}(w_l(\mathbf{x}_k, \mathbf{r}))\right]^2 + \frac{1}{2}\left[\mathbf{J}(\mathbf{0}) + \mathbf{J}(\Delta\mathbf{r})\right]\Delta\mathbf{r}. \tag{3.19}$$

From this, the warp increment computation can be defined as:

$$\widehat{\Delta\mathbf{r}} = \left[\frac{1}{2}\left(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\Delta\mathbf{r})\right)\right]^+ SCV(\mathbf{0}) \tag{3.20}$$

using an Efficient Second order Minimization (ESM) strategy [Benhimane and Malis, 2007]. The Jacobian matrix of the SCV can be separated in several matrices, leading to:

$$\begin{aligned} \mathbf{J}(\Delta\mathbf{r}) &= \mathbf{J}_{I^*}\mathbf{J}_{w_l}\mathbf{J}_\mathbf{K}\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_\mathbf{x}(\Delta\mathbf{r}) \\ \mathbf{J}(\mathbf{0}) &= \mathbf{J}_{I^*}\mathbf{J}_{w_l}\mathbf{J}_\mathbf{K}\mathbf{J}_\mathbf{T}\mathbf{J}_\mathbf{x}(\mathbf{0}) \end{aligned} \tag{3.21}$$

which yields:

$$\widehat{\Delta\mathbf{r}} = \left[\frac{1}{2}\left(\mathbf{J}_{I^*} + \mathbf{J}_{\hat{I}}\right)\mathbf{J}_{w_l}\mathbf{J}_\mathbf{K}\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_\mathbf{x}(\mathbf{0})\right)\right]^+ SCV(\mathbf{0}). \tag{3.22}$$

Let us detail further the expressions of these matrices:

- The Jacobian $\mathbf{J}_{I^*}$ represents the gradient of the image I with relation to the spatial coordinates x, y and z. It is the given by:

$$\mathbf{J}_{I^*} = \left[\nabla I_x \ \nabla I_y \ 0\right]^\top . \tag{3.23}$$

  $\mathbf{J}_{I^*}$ needs therefore only to be computed once whereas $\mathbf{J}_{\hat{I}}$ needs to be computed at each new frame.

- The Jacobian $\mathbf{J}_{w_l}$ is associated to the warp and is a (3×9) matrix given, for any point $\mathbf{x} = \left[x \ y \ 1\right]^\top$:

$$\mathbf{J}_{w_l} = \begin{bmatrix} \mathbf{x}^\top & \mathbf{0} & -x\mathbf{x}^\top \\ \mathbf{0} & \mathbf{x}^\top & -y\mathbf{x}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} . \tag{3.24}$$

  $\mathbf{J}_{\mathbf{K}}$ is a (9×9) matrix that only relies on the intrinsic parameters of the camera:

$$\mathbf{J}_{\mathbf{K}} = \begin{bmatrix} p_x\mathbf{K}^\top & \mathbf{0} & u_0\mathbf{K}^\top \\ \mathbf{0} & p_y\mathbf{K}^\top & v_0\mathbf{K}^\top \\ \mathbf{0} & \mathbf{0} & \mathbf{K}^\top \end{bmatrix} . \tag{3.25}$$

  Let us remark that both $\mathbf{J}_{w_l}$ and $\mathbf{J}_{\mathbf{K}}$ being constant throughout the registration process, they can be computed beforehand.

- For the last two matrices are related to the 3D transformation and the the 3D point. They are computed as a whole, giving a (9×6) matrix composed of:

$$\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}(\mathbf{0}) = \begin{bmatrix} \mathbf{n}^*(\tau_x\mathbf{n}^* + \mathbf{b}_x)^\top & \left[\tau_x\mathbf{n}^* + \mathbf{b}_x\right]_\times \\ \mathbf{n}^*(\tau_y\mathbf{n}^* + \mathbf{b}_y)^\top & \left[\tau_y\mathbf{n}^* + \mathbf{b}_y\right]_\times \\ \mathbf{n}^*(\tau_z\mathbf{n}^* + \mathbf{b}_z)^\top & \left[\tau_z\mathbf{n}^* + \mathbf{b}_z\right]_\times \end{bmatrix} . \tag{3.26}$$

  In this case, the vectors $\mathbf{b}_x, \mathbf{b}_y, \mathbf{b}_z$ represent the unitary vectors composing the reference frame, $\mathbf{n}^* = \left[n_x, \ n_y, \ n_z\right]^\top$ is a normal vector to the considered plane and $\tau = \left[\tau_x, \ \tau_y, \ \tau_z\right]$ is the vector resulting of the current transformation rearranged as:

$$\tau = \frac{-\mathbf{R}^\top\mathbf{t}}{1 + \mathbf{n}^{*\top}\mathbf{R}^\top\mathbf{t}}. \tag{3.27}$$

  Since the values in this matrix depends on the current transformation between the object and the camera, it needs to be computed at each iteration of the process. Let us note that as $\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}$ is already linked to the current transformation $\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}(\mathbf{0}) = \mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}(\Delta\mathbf{r})$.

SCV-BASED 3D REGISTRATION

Initialization

- Gather each face from the model.
- Compute $\mathbf{J_K}$ thanks to the camera parameters.
- Compute $\mathbf{J}_{w_l}$ for each plane.

Re-initialization on face (dis)appearance

- Compute the gradient of the reference image $\mathbf{J}_{I^*}$ for each visible face.

Tracking phase (iterate until $\Delta\mathbf{r}$ is small enough)

- Warp every points of I thanks to the pose parameters $\mathbf{r}$.
- Compute the probability density functions p_{II^*} and p_I.
- Compute the cost function $SCV(\Delta\mathbf{r})\Big|_{\Delta\mathbf{r}=0}$.
- Compute the Jacobians $\mathbf{J}_I$ and $\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}(\mathbf{0})$.
- Compute the warp increment $\Delta\mathbf{r}$.
- Get the inverse parameters $\Delta\mathbf{u}^{-1}$ and update $\mathbf{r}$.

3.4 Using the MI for dense model-based 3D registration

The use of the SCV is a good middle ground between simplicity and robustness. However, when dealing with important local perturbations, it has been shown that it was not robust enough on its own. This is why we developed an approach that allows to perform model-based 3D registration using the mutual information. As it was shown in section 2.2.1.4, the mutual information is a difference of entropy between two images:

$$\mathrm{MI}(\overline{I}, \overline{I^*}) = \mathrm{H}(\overline{I}) + \mathrm{H}(\overline{I^*}) - \mathrm{H}(\overline{I}, \overline{I^*}). \tag{3.28}$$

But since the optimization will not be done on one template but a set of templates, each one representing a face of the model. This leads to a modified cost function that is linked to the position of the camera with relation to the considered object:

$$\begin{aligned} \mathrm{MI}(\Delta\mathbf{r}) &= \mathrm{MI}(\bigcup_l w_l(\overline{I}, \mathbf{r}), \bigcup_l w_l(\overline{I^*}, \Delta\mathbf{r})) \\ &= \mathrm{H}(\bigcup_l w_l(\overline{I}, \mathbf{r})) + \mathrm{H}(\bigcup_l w_l(\overline{I^*}, \Delta\mathbf{r})) - \mathrm{H}(\bigcup_l w_l(\overline{I}, \mathbf{r}), \bigcup_l w_l(\overline{I^*}, \Delta\mathbf{r})) \end{aligned} \tag{3.29}$$

where $\mathrm{MI}(\bigcup_l w_l(\overline{I}, \mathbf{r}), \bigcup_l w_l(\overline{I^*}, \Delta\mathbf{r}))$ is the the mutual information computed on the union of all currently visible faces on the object projected in the image. Considering a set of visible

faces, the probability density functions need to be redefined as:

$$\begin{aligned} p_{\overline{II^*}}(i,j,\mathbf{r},\Delta\mathbf{r}) &= \frac{1}{N_{\mathbf{x}}^l}\sum_l\sum_{\mathbf{x}}\beta(i-\overline{I}(w_l(\mathbf{x},\mathbf{r})))\,\beta(j-\overline{I^*}(w_l(\mathbf{x},\Delta\mathbf{r})) \\ p_{\overline{I}}(i,j,\mathbf{r}) &= \frac{1}{N_{\mathbf{x}}^l}\sum_l\sum_{\mathbf{x}}\beta(i-\overline{I}(w_l(\mathbf{x},\mathbf{r}))) \\ p_{\overline{I^*}}(i,j,\Delta\mathbf{r}) &= \frac{1}{N_{\mathbf{x}}^l}\sum_l\sum_{\mathbf{x}}\beta(j-\overline{I^*}(w_l(\mathbf{x},\Delta\mathbf{r})) \end{aligned} \tag{3.30}$$

where $N_{\mathbf{x}}^l$ is the total number of points used for template registration considering all the visible faces of the model. From that expression of the probabilities, the value of the MI is determined by:

$$\mathrm{MI}(\Delta\mathbf{r}) = \sum_{r,t=0}^{N_d} p_{II^*}(r,t,\mathbf{r},\Delta\mathbf{r})\log\left(\frac{p_{II^*}(r,t,\mathbf{r},\Delta\mathbf{r})}{p_I(r,\mathbf{r})p_{I^*}(t,\Delta\mathbf{r})}\right) \tag{3.31}$$

where N_d is the chosen dynamic for the histogram binning of the image (see Frame 4 for more details). At each new iteration a new warp update is computed as:

$$\Delta\mathbf{u} = -\mathbf{H}^{-1}\mathbf{G} \tag{3.32}$$

where $\mathbf{G}$ and $\mathbf{H}$ are respectively the Gradient and Hessian matrices of the MI. The gradient represents the first order derivatives of the cost function with relation to the warp increment. It is therefore given by:

$$\mathbf{G} = \frac{\partial MI(\Delta\mathbf{r})}{\partial\Delta\mathbf{r}} \simeq \frac{\partial MI(\Delta\mathbf{r})}{\partial\Delta\mathbf{r}}\bigg|_{\Delta\mathbf{r}=0} + \frac{\partial^2 MI(\Delta\mathbf{r})}{\partial\Delta\mathbf{r}^2}\bigg|_{\Delta\mathbf{r}=0}\Delta\mathbf{r}. \tag{3.33}$$

As for the Hessian matrix, it represents the second order derivatives with relation to the warp update parameters:

$$\mathbf{H} = \frac{\partial^2 MI(\Delta\mathbf{r})}{\partial\Delta\mathbf{r}^2}\bigg|_{\Delta\mathbf{r}=0} = \sum_{i,j}\frac{\partial p_{\overline{II^*}}}{\partial\Delta\mathbf{r}}^\top\frac{\partial p_{\overline{II^*}}}{\partial\Delta\mathbf{r}}\left(\frac{1}{p_{\overline{II^*}}}-\frac{1}{p_{\overline{I^*}}}\right)+\frac{\partial^2 p_{\overline{II^*}}}{\partial\Delta\mathbf{r}^2}\left(1+\log\left(\frac{p_{\overline{II^*}}}{p_{\overline{I^*}}}\right)\right). \tag{3.34}$$

Given that the B-splines chosen to perform the histogram binning are twice differentiable, the second order derivatives of the probability density function can be expressed as:

$$\frac{\partial^2 p_{\overline{II^*}}}{\partial\Delta\mathbf{r}^2} = \frac{1}{N_{\mathbf{x}}^l}\sum_{N_l}\sum_{\mathbf{x}}\frac{\partial^2\beta(j-\overline{I^*}(w_l(\mathbf{x},\Delta\mathbf{r})))}{\partial\Delta\mathbf{r}^2}\,\beta(i-\overline{I}(w_l(\mathbf{x},\mathbf{r}))). \tag{3.35}$$

As for the first order derivatives of the B-splines, chain derivation gives the expression of their second order derivative:

$$\begin{aligned} \frac{\partial^2\beta(j-\overline{I^*}(w_l(\mathbf{x},\Delta\mathbf{r})))}{\partial\Delta\mathbf{r}^2} &= \frac{\partial}{\partial\Delta\mathbf{r}}\left[-\frac{\partial\beta}{\partial j}\frac{\partial\overline{I^*}(w_l(\mathbf{x},\Delta\mathbf{r}))}{\partial\Delta\mathbf{r}}\right] \\ &\simeq \frac{\partial^2\beta}{\partial j^2}\frac{\partial\overline{I^*}}{\partial\Delta\mathbf{r}}^\top\frac{\partial\overline{I^*}}{\partial\Delta\mathbf{r}} \end{aligned} \tag{3.36}$$

where:

$$\frac{\partial\overline{I^*}}{\partial\Delta\mathbf{r}} = \mathbf{J}_{\overline{I^*}}\mathbf{J}_{w_l}\mathbf{J}_{\mathbf{K}}\mathbf{J}_{\widehat{\mathbf{T}}}\mathbf{J}_{\mathbf{x}}(\Delta\mathbf{r}) \tag{3.37}$$

MI-BASED 3D REGISTRATION

Initialization

- Gather each face from the model.
- Compute $\mathbf{J_K}$ thanks to the camera parameters.
- Compute $\mathbf{J}_{w_l}$ for each plane.

Re-initialization on face (dis)appearence

- Compute the approached Hessian of the visible model $\mathbf{H}$ and its inverse $\mathbf{H}^{-1}$.

Tracking phase (iterate until $\Delta\mathbf{r}$ is small enough)

- Warp every points of $\overline{I}$ thanks to the pose parameters $\mathbf{r}$.
- Compute the probability density functions $p_{\overline{II^*}}$ and $p_{\overline{I^*}}$ thanks to histogram binning.
- Compute $\mathbf{G}$ the Gradient of the MI.
- Compute the warp increment $\Delta\mathbf{r}$.
- Get the inverse parameters $\Delta\mathbf{u}^{-1}$ and update $\mathbf{r}$.

3.5 Experimental results

In this section, a focus will be put on experiments evaluating the proposed model-based 3D registration techniques. The methodology is the same for both approaches. First, an original pose is computed from the first image of the considered sequence by matching four points in the image with their 3D correspondences. From that initial pose and the knowledge of the 3D model that is to be tracked in the sequence, the tracking process is initialized.

3.5.1 Comparison between simple plane and multiplane approaches

This experiment was realized in order to evaluate the advantages of taking several planes into account at the same time. The setup is the following. A sequence figuring an angle building was taken. Two model-based tracking approaches were then launched on the sequence. The first approach used two trackers, one for each considered plane. The two trackers then performed independently. The other method was to use the multiplane strategy proposed earlier. Results are shown on figure 3.3. The two monoplane trackers approach fails to track the building when a part disappears and then appears again. This is due to one tracker failing because of a registration on a small number of points. On the other hand the multiplane approach uses the relation between the planes to keep a space-time coherency and prevents the failure encountered with the other strategy. This experience then shows that the proposed approach performs well in nominal conditions and that the use of a model to link several faces in the same optimization allows for a good robustness to (dis)appearances of faces in the scene.

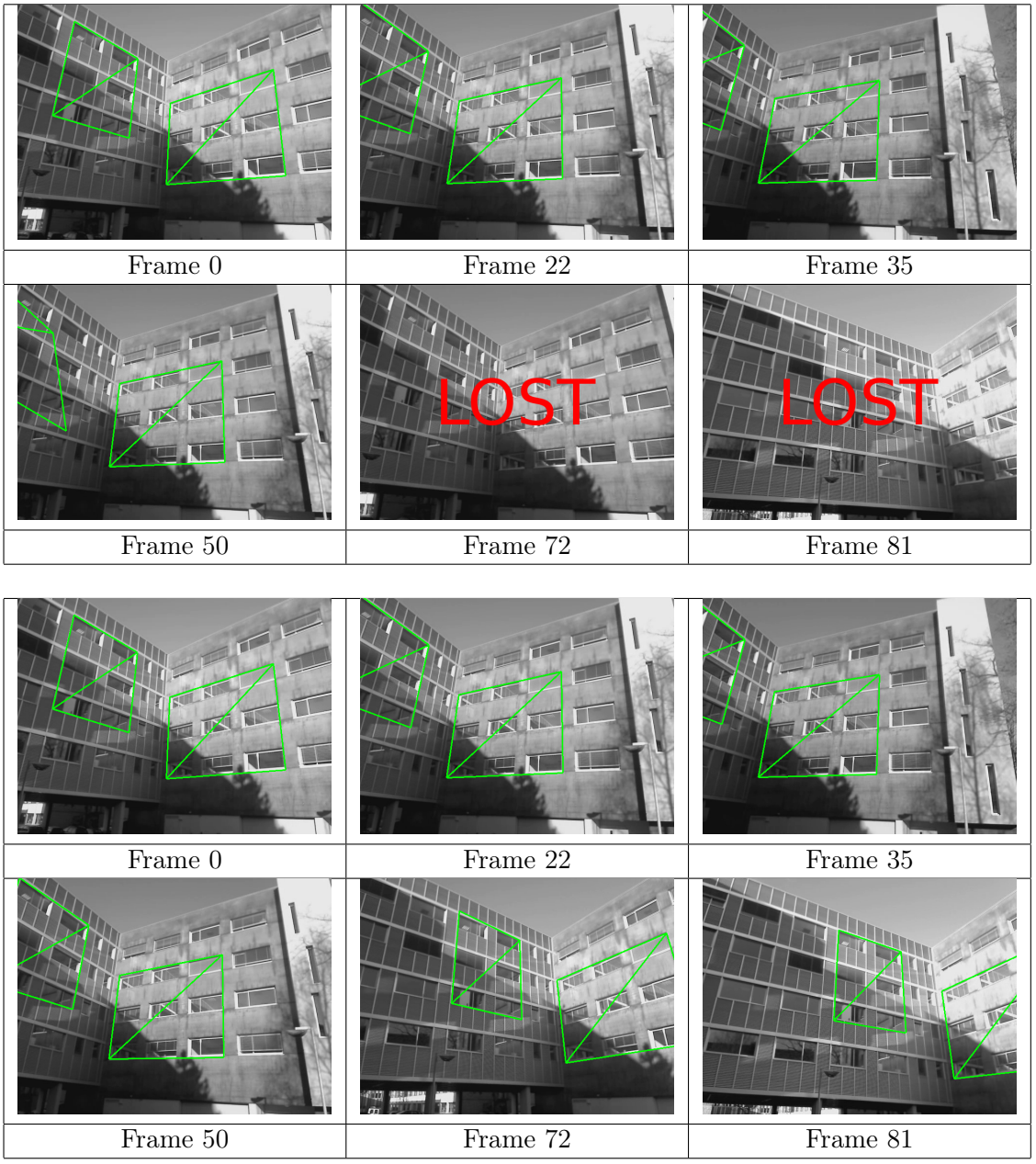


Figure 3.3: Difference between two monoplane trackers on the top and one multiplane tracker on the bottom.

3.5.2 Tracking in nominal conditions

The goal of this experiment is to validate both the SCV and MI approaches in a nominal cases. In order to do that, the performances of the trackers were evaluated on two sequences featuring a model seen in conditions where no perturbation arose. The two chosen sequences showed a parallelepiped to be followed. The first sequence then depicts a cube and fast camera displacements that allow to assess the robustness of the methods in these conditions. The second sequence features a box that is rotating on a plate, therefore creating appearances and disappearances of faces.

3.5.2.1 Cube sequence

The first sequences shows a moving cube. The cube is placed on a table and the camera is moved around its location. The sequence features appearances and disappearances of cube faces. It also contains blur on a consequent number of frames as the camera is moving quite fast at times. Results show that both algorithms give the same good results, with a precise localization throughout the sequence. Figure 3.4 shows pictures taken when faces appear or disappear in the sequence. They also show a similarity of localization results as the displayed object frame allows to confirm. Finally, the resulting camera trajectory in 3D for both algorithms can be found on figure 3.5. The trajectories are very close from one approach to the other and they are not oscillating, even though the motion of the camera is important. The figures show however that the SCV algorithm gives a slightly more shaky trajectory when the motion is brutal. This is due to the appearance of blur in the corresponding frames, which is not taken into account by any added robustness layer here.

3.5.2.2 Box sequence

The box sequence shows a teabox rotating on itself. The rotation angle is greater than 180° which allows to consider the robustness of the approach to drift effects that can be caused by reinitializations of the faces at inappropriate frames. Figure 3.6 shows frames from that sequence. It is clear from frame 120 on that perspective effects combined with reinitializations have cause a minor drift. However, the displayed object frame is also seen to be stable at a very good position which means that those drift effects did not perturb the localization process. Let us also note that results are only shown here for the MI-based approach as the results of the SCV version of the algorithm were the same, as it was the case for the previous experiment. Here again, the trajectory of the camera is shown on figure 3.7. The curves show clearly the rotation of the box and the several motions of the camera that is moved during the sequence, along with the fact that the pose computed by the algorithm is not shaky here either.

Once the viability of the proposed approaches has been demonstrated, several experiments have been realized in order to evaluate their robustness in perturbed conditions. Indeed, the main objective when designing these approaches has been set to create robust solutions. The following experiments then aim at demonstrating that the proposed approach are robust in different conditions without any added robustness layers.

3.5.3 Robustness towards illumination variations

Both the SCV and MI have been shown in section 2.3 to be robust to global perturbations. The most commonly found perturbation is light variation. This is why in this experiment a model is tracked along a sequence featuring several light changes. These perturbations were realized by switching on and off the lights of a room while the camera is moving and creating blur. The results of both localization strategies can be found on Figure 3.9. The first clear conclusions that can be drawn is that in the same conditions as the two proposed approaches, the SSD alone fails tracking the target as soon as the light changes in the images. Finally, another interesting observation can be made. When using only 8 bins to perform the MI optimization as it is the case here, there are frames where the MI gives a slightly less accurate result than the SCV method which uses 64 bins. On the other hand, once the blur is too important, the SCV fails to give good results. But since using much more bins for the MI (as can be done for the SCV) would result in a less well

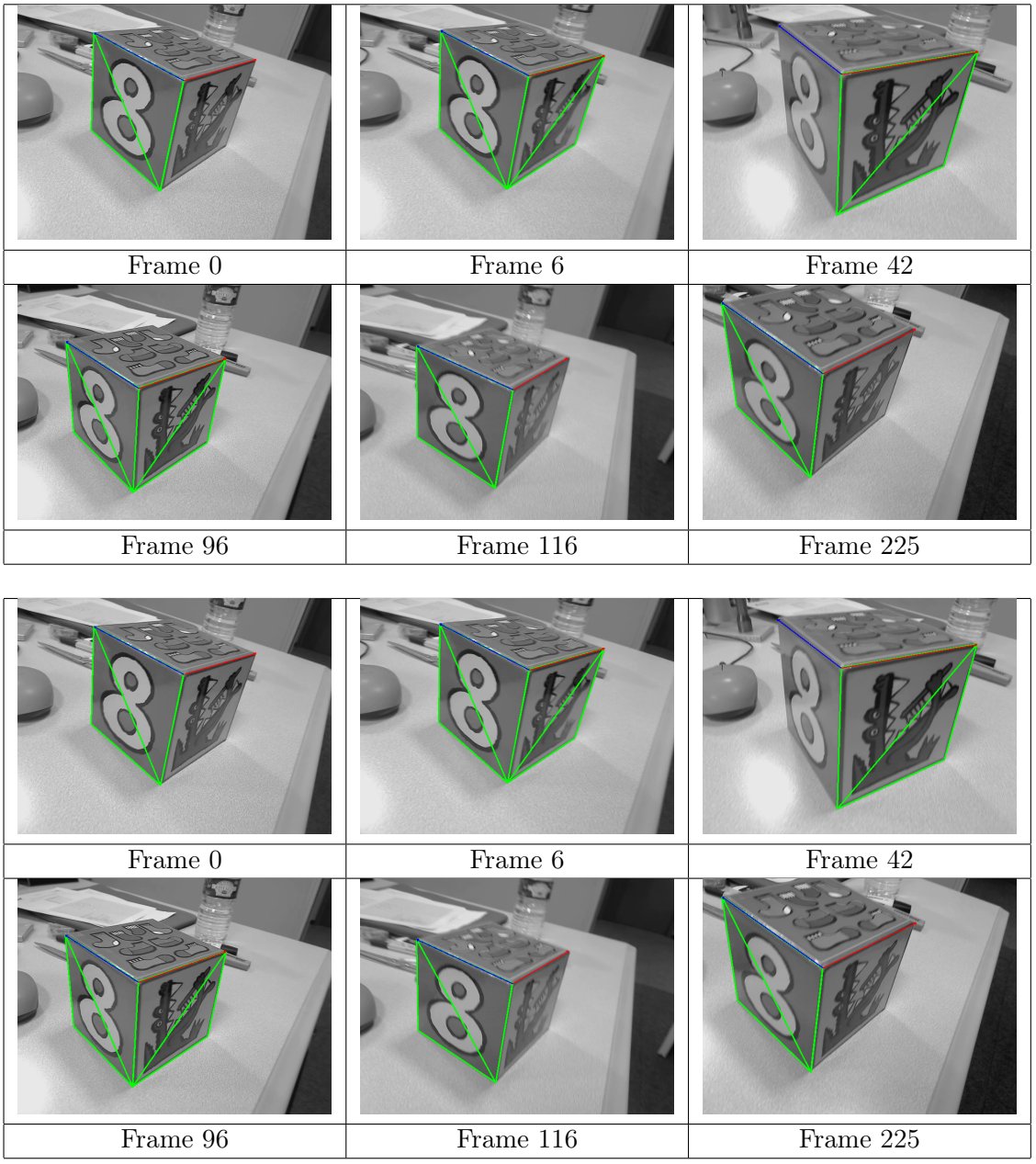


Figure 3.4: Results of both the SCV (on the top) and MI (on the bottom) model-based algorithms on the cube sequence.

conditioned cost function and a higher computational cost, a trade-off can be seen between the SCV and MI approaches. The MI is more robust, as it is able to cope with a wider range of perturbations but the SCV can give better results in conditions where only global variations appear. Figure 3.8 shows the tracking results of the MI more precisely. It shows how the MI is able to cope with situations where even the human eye can begin to have difficulty registering the current and reference templates.

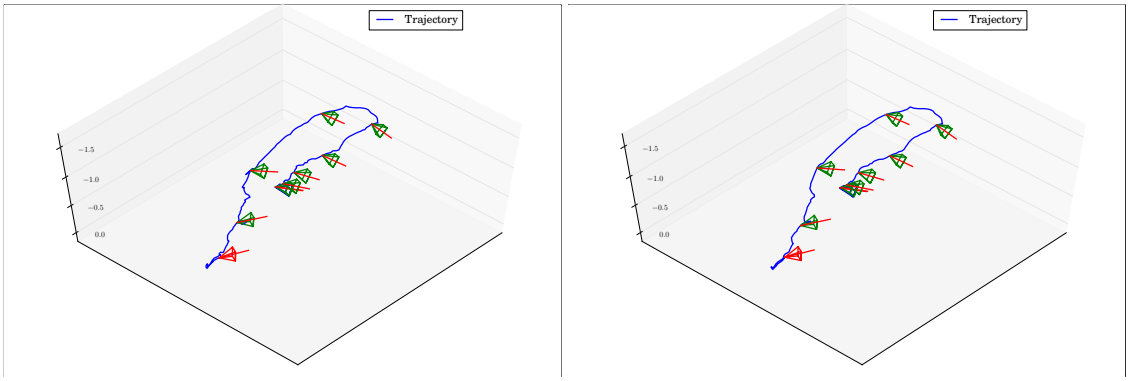


Figure 3.5: Camera trajectories for the SCV (on the left) and MI (on the right) model-based algorithms on the cube sequence.

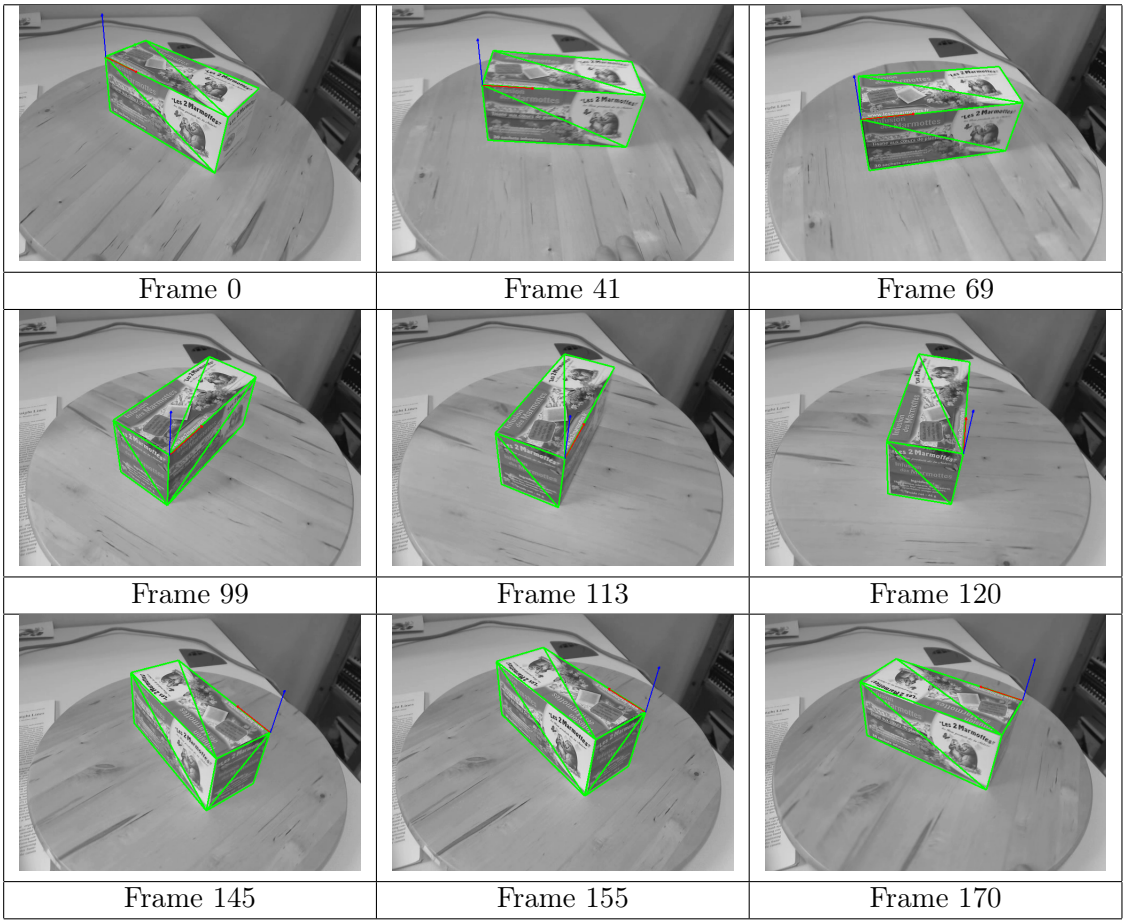


Figure 3.6: Results of the MI model-based algorithm on the box sequence.

3.5.4 Robustness towards local perturbations

Previous experiments have validated the proposed approaches in sequences featuring global variations such as illumination changes. In this section, the objective is to assess the robustness of the approaches in situation where local perturbations appear such as occluded parts of the object or specular spots. As shown in section 2.3, the MI is still able to give good results whereas the SCV fails as its expression cannot model that kind of perturbations.

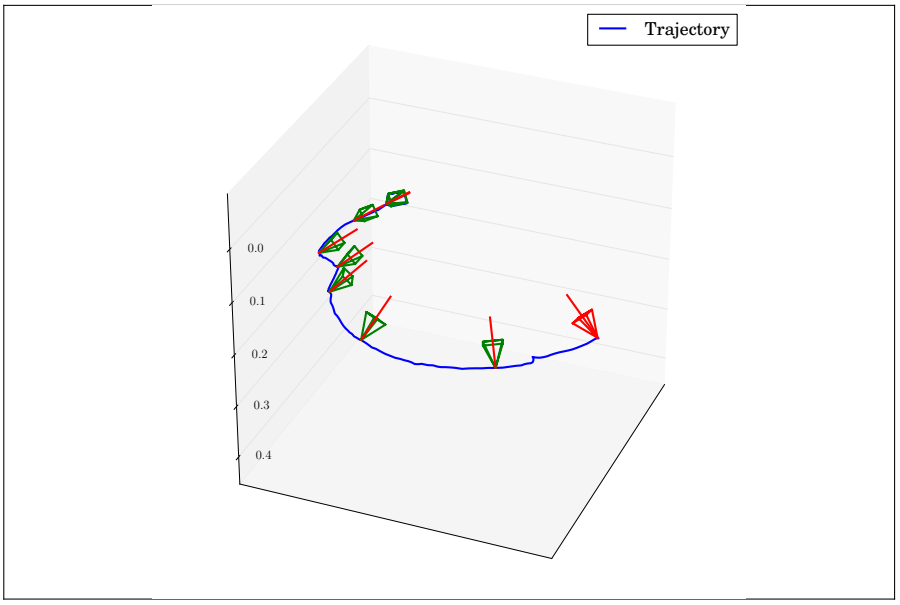


Figure 3.7: Camera trajectory for the MI model-based algorithm on the teabox sequence.

Figure 3.8: On the left of each pair is the tracking result of our MI algorithm and on the right what is actually seen in the image at the place of the left face. Let us add that the patch is very blurred on several occasions but the tracking is still successful.

3.5.4.1 Case of specularities

The experiment described in this section aims at evaluating the robustness of the approaches when confronted to specular spots on the templates. Specular spots are the

Figure 3.9: Results of the SSD (on the left), SCV (in the middle) and MI (on the right) model-based algorithms in light varying conditions. The SSD is lost at the first light variation, whereas light variations do no impact the SCV but it is still lost when too much blur appears in the image.

results of light hitting a non-lambertian surface at an angle. The nature of the material then creates a white spot on the image. This is a very challenging problem for visual tracking techniques as the visual data change and a part of the texture is often lost. In this experiment, 3D registration using SSD, SCV and MI were launched on a sequence featuring a large specular spot on a face of the model. Even though one of the faces is still in nominal conditions, both the SSD and SCV failed to localize the model as soon as the specular task comes into play. The MI-based version on the other hand can cope with that kind of perturbations and is able to complete the localization. Results of this tracking process are shown on Figure 3.10.

3.5.4.2 Case of occlusions

Another challenging local perturbation is the presence of occluded areas of the template. Depending on the area that undergoes an occlusion, several techniques such as M-estimators have been shown to be successful but the goal is here to assess the basic robustness of the approaches, before any additional layer is put into place. A video sequence is then shown here where a large area of a face is occluded. As is the case with specularities, neither the SSD nor SCV are able to cope with that situation. The MI on the other hand still gives good results. Although challenged at some points of the sequence where the accuracy of the localization is impacted when the occlusion appears or disappears, the process is still a success, and a good localization is obtained as shown on Figure 3.11

3.5.5 Convergence domain analysis

Final experiments were also realized to analyze the convergence domain of each approach in different conditions. Once the tracker has been initialized with the first frame of the sequence, the localization process is started from an image in the sequence and the pose parameters are set to the corresponding ground truth. The parameters are then perturbed with white Gaussian noise on the pose of chosen σ and the tracking is launched. After the tracking is over, the resulting pose parameters and the ground truth are compared and if the error is small enough, the tracking is considered successful. The process is repeated 500 times for each method in each situation. The results are shown on figure 3.13. When adding noise to the pose parameters, a σ_R of 0.01 rad is chosen for the rotations since their impact is very important on the pose and σ_t is chosen in a range from 0.002 to 0.05 m to see the impact of the starting position (see figure 4.11 for examples of starting positions). The curves show that mutual information is the only possible solution when confronted to important scene variations such as large occlusions or specularities which validates the results of the previous experiments which showed similar results. The convergence graphs also show that in nominal conditions the convergence domain of both SCV and MI, whilst not being as important as the SSD, is very wide which validates the fact that, as stated in section 2.3, both SCV and MI are viable solutions to replace the SSD as cost function for localization processes.

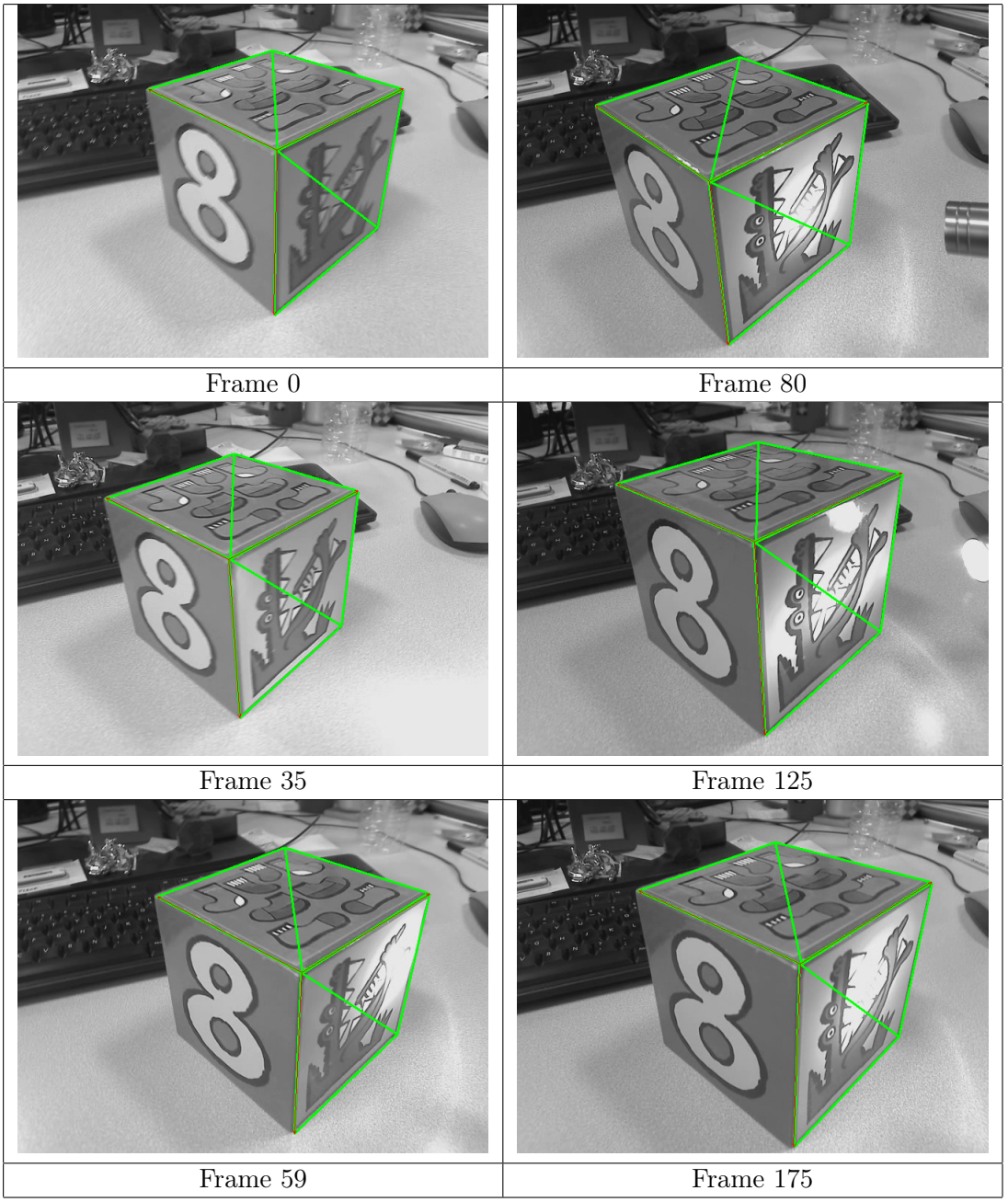


Figure 3.10: Results of the MI model-based algorithm when confronted to specularities. Let us add that the left face is willingly omitted from the optimization process in order to give more weight to the perturbation caused by the specular spot.

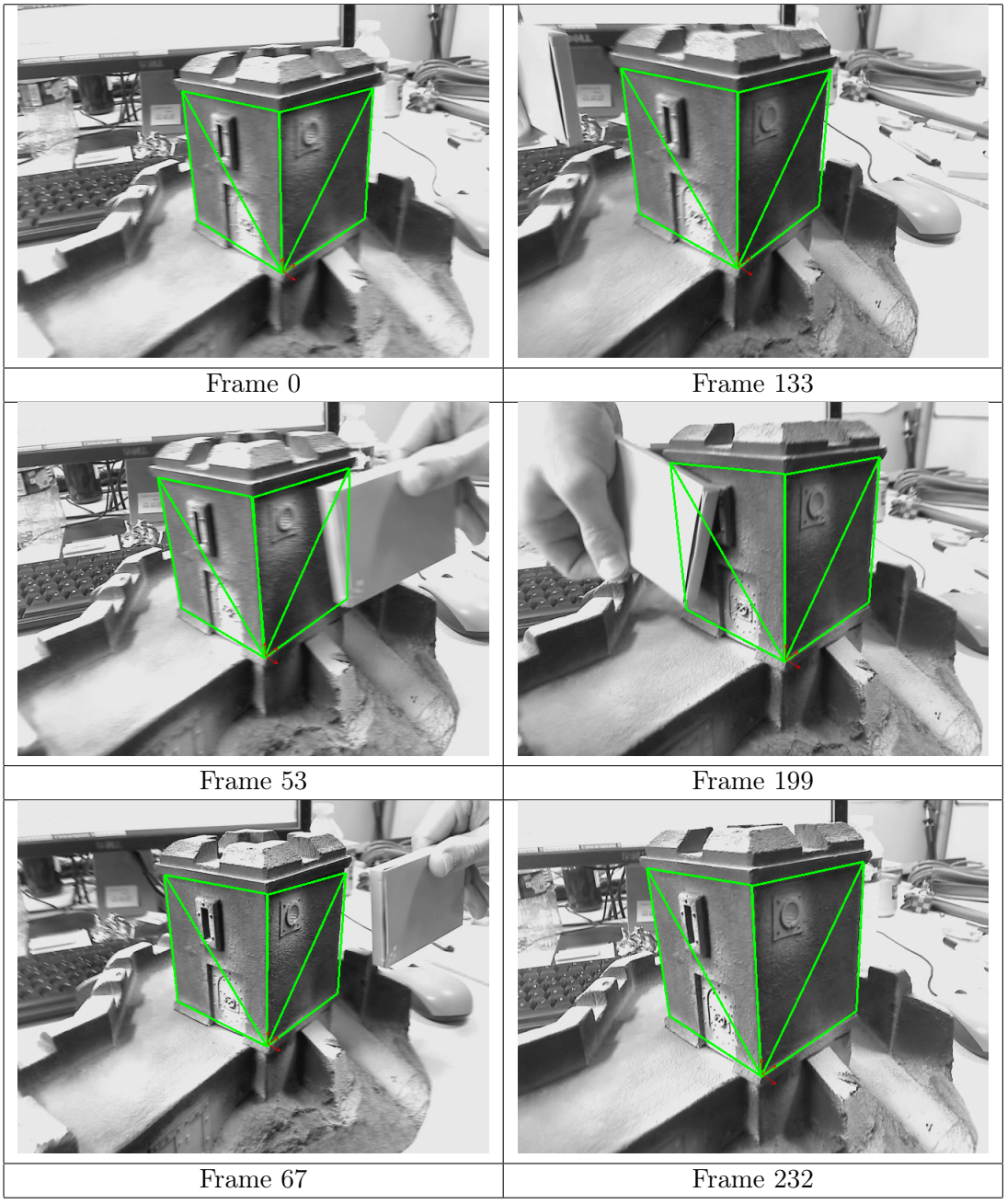


Figure 3.11: Results of the MI model-based algorithm when confronted to occlusions.

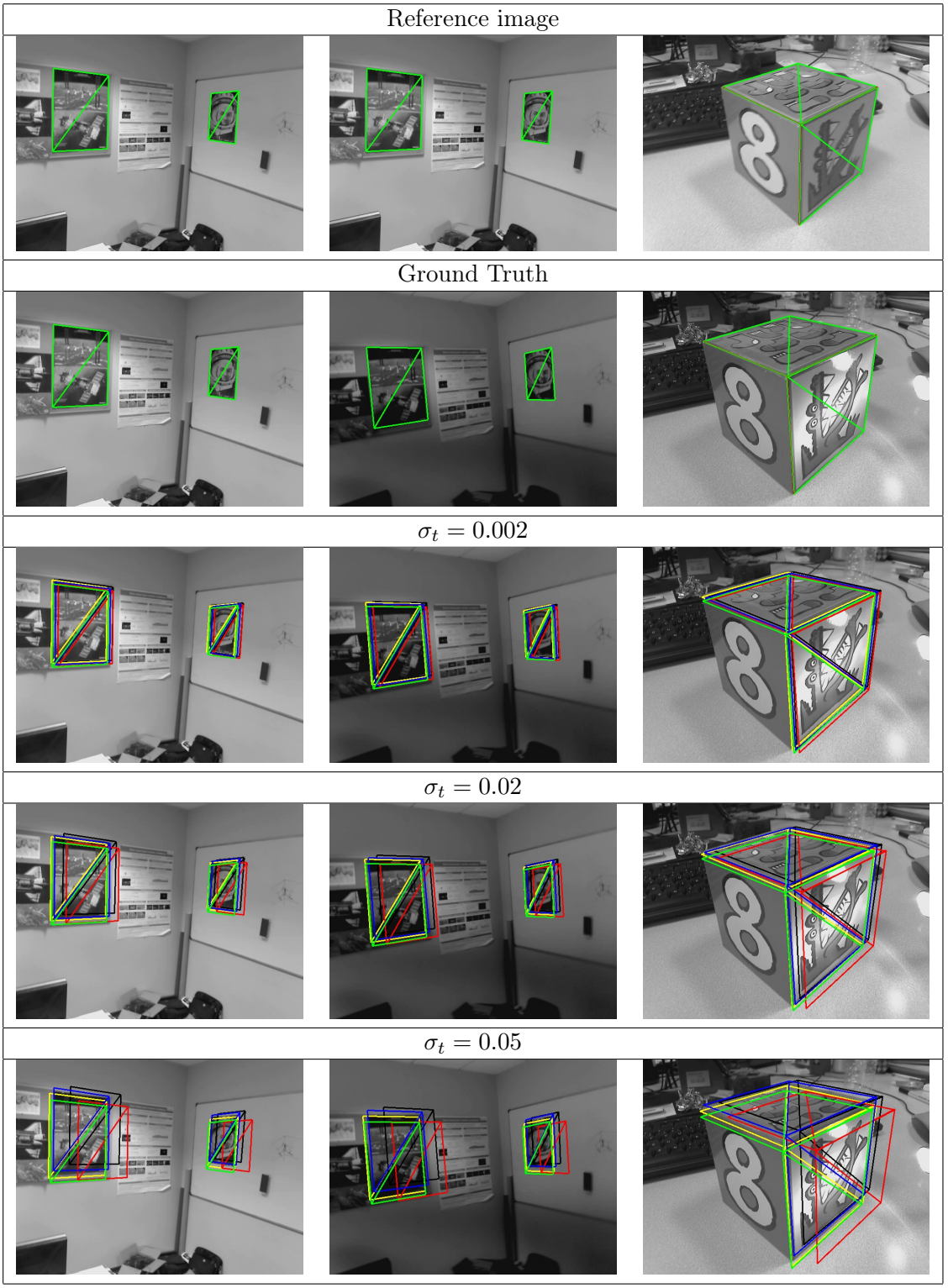


Figure 3.12: Examples of starting positions with different σ_t. For all experiments $\sigma_R = 0.01$.

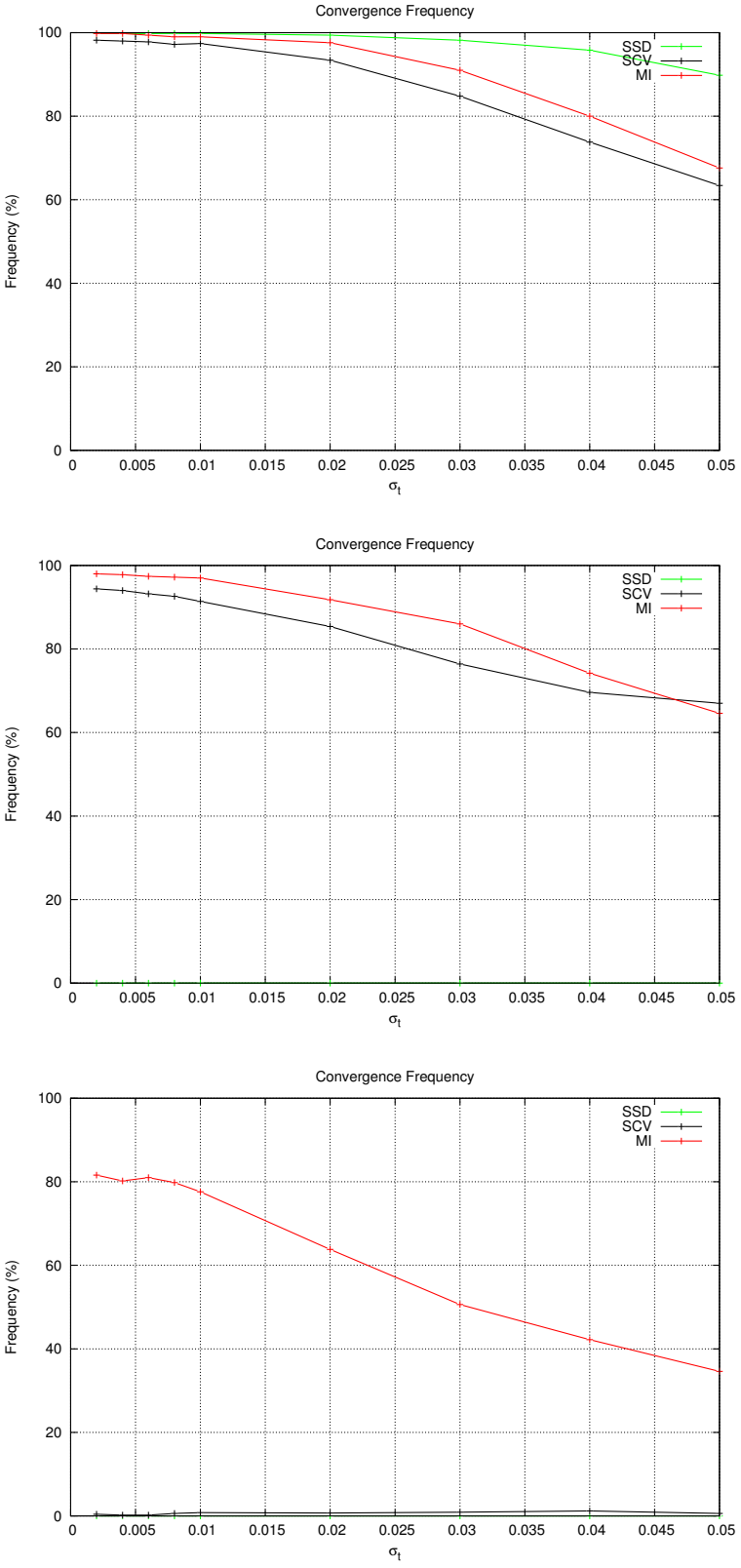


Figure 3.13: From top to bottom : convergence frequency in nominal conditions (column 1 of fig. 4.11), in light varying conditions (column 2 of fig. 4.11) and when confronted to specularities (column 3 of fig. 4.11).

3.6 Conclusion

In this chapter, two new algorithms have been proposed to perform dense model-based visual tracking. The first method is based on the SCV and is shown to be very close to the SSD in terms of implementation but more robust to global light variations. The other one is based on the mutual information and is validated through experimentations in very perturbed condition, for example when confronted to specular spots or occlusions. Finally, an estimation of the convergence domains of both approaches shows the algorithms to be efficient in a variety of situations. Building on these results, the next chapter aims at adding more degrees of freedom to the observable motion by considering a non-rigid warp function.

4

Dense non-rigid 2D image registration

The work presented in chapter 3 considered rigid surfaces. It relies on transformation warps such as the affine warp or homographies in order to model the displacement of planar surfaces. Other works have proposed approaches that rely on less constrained transformation functions. Depending on the considered non-rigid warping function, the algorithms vary. In this chapter, two approaches will be proposed that perform dense non-rigid robust 2D registration. First, an algorithm based on the SCV is described that allows to track non-rigid surfaces in sequences where global variations appear while staying computationally close to a SSD approach. The second proposition is to consider the MI as cost function. The resulting algorithm is therefore more robust to local perturbations, at the expense of complexity. Before detailing and validating the proposed approaches, non-rigid warp functions will be defined and an overview of the literature on the subject will be made in order to understand the context of the proposed works.

4.1 Non-rigid warping functions: adding freedom to the motion

The limit of the transformation functions defined before is that they only model the displacement of specific rigid bodies. If the model is not adapted to the object (*e.g.* non planar surfaces) or if deformations appear, they are not relevant anymore and cannot cope with the perturbations. In order to prevent these drawbacks, several works have proposed to use deformable motion warps [Zhu, 2007]. Several warping functions can be considered, in this section we will detail Free Form Deformations [Gay-Bellile et al., 2010] and a Radial Basis Function [Arad and Reisfeld, 1995, Ruprecht and Müller, 1993] called Thin-plate Spline warp [Bookstein, 1989].

4.1.1 Free form deformation (FFD)

Free form deformations [Sederberg and Parry, 1986] are often used to model 3D solid deformations for image registration [Rueckert et al., 1999]. FFDs allow to warp a chosen region of interest by deforming a regular grid of control points. The set of control points $\mathbf{s} = \left[s_0, \ldots, s_{N_p}\right]$ is deformed thanks to the parameters of the warp $\mathbf{u}$ that represent the displacement of the control points.

$$w(\mathbf{x}, \mathbf{u}) = \sum_{l=0}^{3} \sum_{m=0}^{3} \beta_l(v)\beta_m(w) \left(\mathbf{s}_{i+l,j+m} + \mathbf{u}_{i+l,j+m}\right) \tag{4.1}$$

where v and w are normalized coordinates and β_l and β_m are cubic B-spline interpolation coefficients. This transformation function allows a great freedom of deformation. It has,

however, no built-in smoothing factor to help keep a consistency of the deformation. This is why optimizations computing a FFD displacement are often associated with supplementary smooth terms [Gay-Bellile et al., 2010].

4.1.2 Thin-plate spline (TPS)

Thin-plate spline(TPS) warps belong to the radial basis functions warps. They minimize the bending energy of the parameterized surface based on a set of control points inducing space coherency constraints. The considered warping function is a combination of an affine warp and an added deformation term:

$$w(\mathbf{p},\mathbf{u}) = \begin{pmatrix} a_0 & a_1 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_2 \\ a_5 \end{pmatrix} + \mathbf{X}(\mathbf{p},\mathbf{c}) \tag{4.2}$$

with $\mathbf{X}(\mathbf{p},\mathbf{c})$ a deformation kernel that depend on both the current point $\mathbf{p}$ and a set of control points $\mathbf{c} = \left[c_0,\ \ldots,\ c_{N_p}\right]$. The deformation kernel used in the TPS warp is a thin-plate kernel:

$$\begin{aligned} \phi(x) &= \frac{x^{(4-p)}\ \log(x)}{\sigma} \text{ for } 4-p \in 2\mathbb{N} \\ \phi(x) &= \frac{x^{(4-p)}}{\sigma} \text{ elsewise} \end{aligned} \tag{4.3}$$

where σ and p are parameters that control the freedom given to the deformation, p controlling the smoothness of the surface and σ the spatial influence of the kernel. In the remainder of this work, we set $\sigma = 2$ and $p = 2$, leading to:

$$\phi(x) = \frac{1}{2}\ x^2\ \log(x). \tag{4.4}$$

This leads to the full equation of the warping function:

$$w(\mathbf{p},\mathbf{u}) = \underbrace{\begin{pmatrix} a_0 & a_1 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_2 \\ a_5 \end{pmatrix}}_{\text{Affine warp}} + \underbrace{\sum_{k=1}^{N_p} \begin{pmatrix} w_x^k \\ w_y^k \end{pmatrix} \phi(d^2(\mathbf{p},\mathbf{c}_k))}_{\text{TPS kernel}}. \tag{4.5}$$

where N_p is the number of considered control points $\mathbf{c}$, w_x^k is the weight of the k^{th} control point along the x axis and $d(\mathbf{x},\mathbf{y})$ is the euclidean distance between two points $\mathbf{x}$ and $\mathbf{y}$. From equation (4.5), we can deduce a warp parameter vector of dimension $2N_p + 6$:

$$\mathbf{u}^\top = (a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ \mathbf{w}_x^\top\ \mathbf{w}_y^\top). \tag{4.6}$$

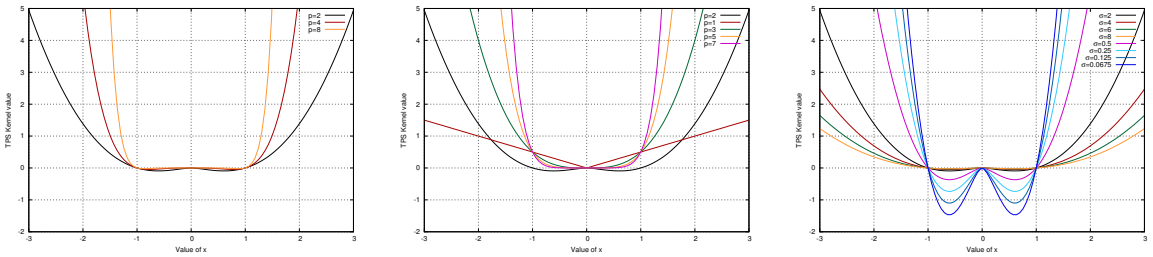
where $\mathbf{w}_x^\top$ and $\mathbf{w}_y^\top$ are vectors composed of the weighting criterion for each control point:

$$\mathbf{w}_x^\top = \begin{pmatrix} w_x^0 & \ldots & w_x^{N_p-1} \end{pmatrix} \tag{4.7}$$

$$\mathbf{w}_y^\top = \begin{pmatrix} w_y^0 & \ldots & w_y^{N_p-1} \end{pmatrix}. \tag{4.8}$$

4.2 An overview of dense non-rigid surface registration

Several works have been proposed that use non-rigid warp functions in order to perform non-rigid image registration. This section then aims at providing with an understanding of interesting approaches that have been proposed in this field.



a). Influence of p when $4 - p \in 2\mathbb{N}$

b). Influence of p when $4 - p \notin 2\mathbb{N}$

c). Influence of σ

Figure 4.1: Influence of the different parameters on the shape of the TPS kernel. We can see that the choice of σ=p=2 in black is interesting as the shape of the function is well defined but not too much in order to avoid local inconsistencies.

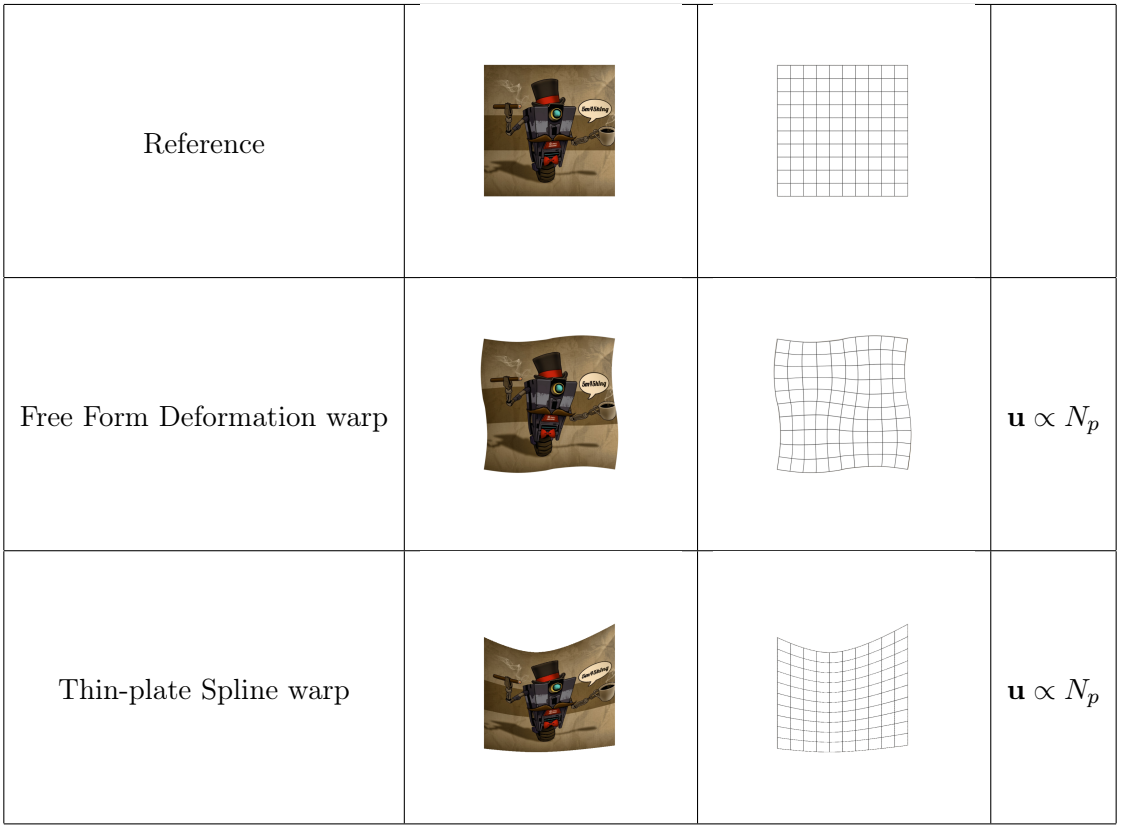


Figure 4.2: Overview of commonly used non-rigid transformation functions.

4.2.1 Using FFDs for non-rigid 2D registration

In [Gay-Bellile et al., 2010], a non-rigid direct 2D registration that relies on the use of FFD warps has been proposed. They propose to minimize the SSD between set of points from a reference template and another set from a current image warped thanks to a FFD warp. This warp can in this case be written as:

$$w(\mathbf{s}, \mathbf{u}) = \sum_{l=0}^{3} \sum_{m=0}^{3} \beta_l(v)\beta_m(w) \left(\mathbf{s}_{i+l,j+m} + \mathbf{u}_{i+l,j+m}\right) \tag{4.9}$$

where v and w are normalized coordinates and β_l and β_m are cubic B-spline interpolation coefficients. $\mathbf{s}$ is then a set of control points and $\mathbf{u}$ includes the displacement of the control points. From that warp, a minimization of the SSD can be performed:

$$SSD(\mathbf{u}) = I^*(\mathbf{x}) - I(w(\mathbf{x}, \mathbf{u})). \tag{4.10}$$

As the FFD do not have a built-in layer of smoothness, the optimization needs to add one. The optimization is then composed of the SSD part called data term $\mathcal{E}_d(\mathbf{u}) = SSD(\mathbf{u})$ and the smoothing term $\mathcal{E}_s(\mathbf{u})$:

$$\mathcal{E}(\mathbf{u}) = \mathcal{E}_d(\mathbf{u}) + \lambda_s \mathcal{E}_s(\mathbf{u}) \tag{4.11}$$

with λ_s and weight parameter. The chosen smooth term, called bending energy, is related to the TPS. It was shown to be well suited for registering smooth surfaces as shown in [Bartoli et al., 2004]. Considering $\mathcal{C} = \mathcal{C}_{xx}^\top \mathcal{C}_{xx} + 2\mathcal{C}_{xy}^\top \mathcal{C}_{xy} + \mathcal{C}_{yy}^\top \mathcal{C}_{yy}$ where $\mathcal{C}_{ab} = \frac{\partial^2 w}{\partial a \partial b}$, $\mathcal{E}_s(\mathbf{u})$ can be expressed as:

$$\mathcal{E}_s(\mathbf{u}) = \mathbf{u}^\top \mathrm{diag}(\mathcal{C}, \mathcal{C})\, \mathbf{u} = {\mathbf{u}_x}^\top \mathcal{C} \mathbf{u}_x + {\mathbf{u}_y}^\top \mathcal{C} \mathbf{u}_y. \tag{4.12}$$

The registration can then be done by performing either non-linear optimization scheme. Let us add that in [Gay-Bellile et al., 2010], another term is added in order to deal with occlusions and extend the algorithm to detect and handle self-occlusions due to deformations of the template.

4.2.2 Using TPS for non-rigid 2D registration

Another way to perform non-rigid registration is to use the TPS warp as transformation function of the process. In [Brunet et al., 2011], that strategy has been chosen, resulting in a direct non-rigid image registration technique. When performing such registration, the first problem is to be able to parameterize the transformation with relation to warp parameters. The second problem is that in order to be able to use an inverse compositional optimization scheme, the warp needs to be invertible. In this paragraph we will show how the works of [Brunet et al., 2011] have proposed to solve those problematics. Considering a point $\mathbf{x}$, the result of the transformation can be parameterized as:

$$w(\mathbf{x}, \mathbf{u}) = \mathbf{o}_\mathbf{x}^\top \begin{pmatrix} \mathbf{\Omega} \\ \mathbf{A}^\top \end{pmatrix} \tag{4.13}$$

where $\mathbf{A}$ is the matrix composed of the affine transformation parameters:

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \end{pmatrix} \tag{4.14}$$

$\mathbf{\Omega}$ is a $(N_p \times 2)$ matrix composed of the weights associated to the control points:

$$\mathbf{\Omega}^\top = \begin{pmatrix} w_x^1 & \dots & w_x^p \\ w_y^1 & \dots & w_y^p \end{pmatrix} \tag{4.15}$$

and $\mathbf{o}_\mathbf{x}^\top = \begin{pmatrix} \phi(d^2(\mathbf{x}, \mathbf{c}_0)) & \dots & \phi(d^2(\mathbf{x}, \mathbf{c}_k)) & x & y & 1 \end{pmatrix}$ (see figure 4.3). Regrouping the equation (4.13) for all control points allows the parameters of the warp $\mathbf{A}$ and $\mathbf{\Omega}$ to be computed thanks to the following linear system [Bookstein, 1989]:

$$\begin{pmatrix} \mathbf{K} + \lambda \mathbf{I} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0}_{3\times 3} \end{pmatrix} \begin{pmatrix} \mathbf{\Omega} \\ \mathbf{A}^\top \end{pmatrix} = \begin{pmatrix} \mathbf{P}' \\ \mathbf{0}_{3\times 2} \end{pmatrix} \tag{4.16}$$

where $\mathbf{K}$ is a $(N_p \times N_p)$ matrix given by $\mathbf{K}_{i,j} = \phi(d^2(\mathbf{c}_i, \mathbf{c}_j))$, $\mathbf{P} = \begin{pmatrix} c_j^x & c_j^y & 1 \end{pmatrix}$ and $\mathbf{P'} = \begin{pmatrix} c'^x_j & c'^y_j & 1 \end{pmatrix}$ where $\mathbf{c}'_k$ is the k^{th} control point once displaced. Let us add that $\lambda\mathbf{I}$ is a regularization term. The TPS warp tends to an affine transformation when λ increases. The (3×2) null matrix $\mathbf{0}_{3\times2}$ is added in order to enforce side conditions:

$$\mathbf{P}^\top \mathbf{\Omega} = \mathbf{0}. \tag{4.17}$$

Those conditions allow the considered equations to be twice differentiable. The goal is to regularize the function on the borders of the template. In order to compute the inverse parameters, the inverse equation of (4.16) must be computed. The matrix on the left side of the relation being invertible by blocs, inverting equation (4.16) leads to:

$$\begin{pmatrix} \mathbf{\Omega} \\ \mathbf{A}^\top \end{pmatrix} = \mathbf{E}_\lambda \mathbf{P'} \tag{4.18}$$

where $\mathbf{E}_\lambda$ is a $(N_p + 3 \times N_p)$ matrix resulting of the invert of the transfer matrix expressed in equation (4.16) which is given by:

$$\mathbf{E}_\lambda = \begin{pmatrix} \mathbf{K^{-1}} \left(\mathbf{Id} - \mathbf{P} \left(\mathbf{P}^\top \mathbf{K^{-1}} \mathbf{P} \right)^{-1} \mathbf{P}^\top \mathbf{K^{-1}} \right) \\ \left(\mathbf{P}^\top \mathbf{K^{-1}} \mathbf{P} \right)^{-1} \mathbf{P}^\top \mathbf{K^{-1}} \end{pmatrix}. \tag{4.19}$$

We can therefore revert the warp from the current control points set $\mathbf{P}^0$ to $\mathbf{P}$ as in [Gay-Bellile et al., 2007] giving a new set $\mathbf{P'}$ thanks to the following system:

$$\mathbf{P'} = (\mathbf{O}_p \mathbf{E}_\lambda)^{-1} \mathbf{P}^0. \tag{4.20}$$

In this case $\mathbf{O}_p$ is a $(N_p \times N_p + 3)$ transfer matrix which i^{th} line is composed of the vector $\mathbf{o}^\top_{\mathbf{c_i}}$ associated to the i^{th} control point. This relation is also true for warp composition (see figure 4.4). Considering two sets of warped control points $\mathbf{P}$ and $\mathbf{P'}$, the composition of the two warps resulting in a set of control points $\mathbf{P''}$ is given by:

$$\mathbf{P''} = \mathbf{O}^\top_{\mathbf{P}} \mathbf{E}_\lambda \mathbf{P'}. \tag{4.21}$$

Let us note that in [Brunet et al., 2011], it has been shown possible to perform the registration process using Jacobian matrices learned beforehand.

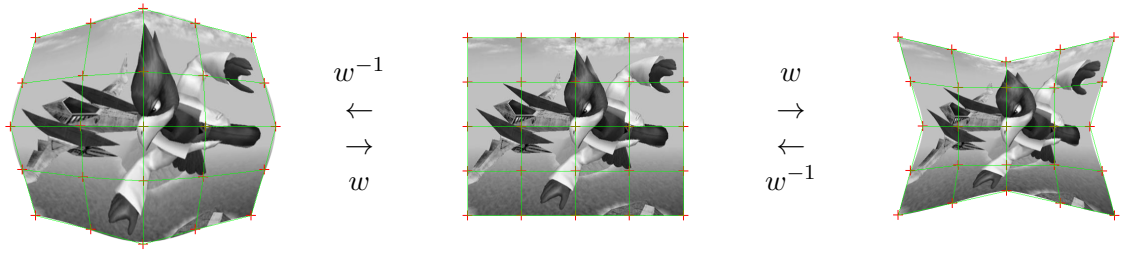


Figure 4.3: TPS warp inverting. Equation (4.20) allows to find the parameters of the inverse warp w^{-1}.

4.2.3 Dense non-rigid 2D image registration

The problem when using the SSD in order to perform dense non-rigid registration is the same as with 3D registration: robustness. If perturbations occur, they need to be taken into

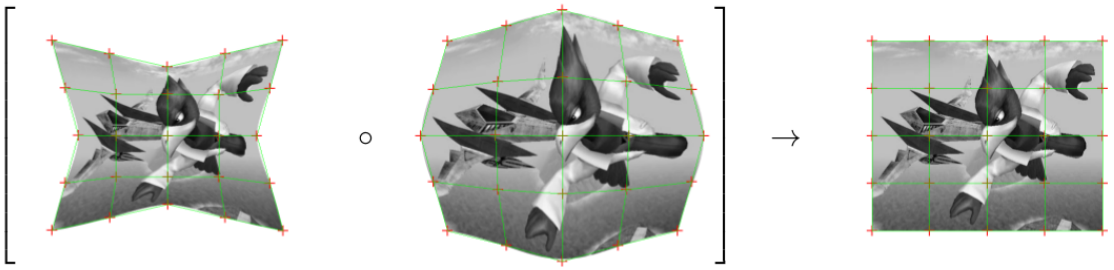


Figure 4.4: TPS warp composition. Composing the warp with its inverse allows to get back to the starting image.

account in a specific added algorithmic layer. In this chapter we show two new approaches to the problem that optimize the SCV and MI over the parameters of a thin-plate spline warp function. The SCV method is then naturally robust to global light variations and still easy to introduce at the same time. On the other hand, the MI version is more robust to local perturbations but this added robustness comes at the expense of simplicity.

4.3 Using the SCV for dense non-rigid 2D registration

In this section we will show a method to perform dense non-rigid 2D registration using the SCV. In order to elaborate our algorithm we chose to look at what had been done in [Brunet et al., 2011]. But contrary to what was done in those works, we do not want to use any learning of the Jacobian matrices. Indeed, in order to be able to judge the natural robustness ensuing the use of the SCV, we chose not to add any robustification step. In order to perform that minimization, let us first recall the expression of the cost function of the algorithm parameterized by the vector $\mathbf{u}$ containing the parameters of a TPS warp (see section 4.1.2):

$$SCV(\Delta\mathbf{u}) = \sum_{n=1}^{N_\mathbf{x}} \left[I^*(w(\mathbf{x}_n, \Delta\mathbf{u})) - \hat{I}(w(\mathbf{x}_n, \mathbf{u})) \right]^2. \tag{4.22}$$

where $\hat{I}$ is the current image after the grey level adaptation has been done with relation to the histogram of the reference image I^*:

$$\hat{I}(\mathbf{x}) = \mathcal{E}(I(\mathbf{x}) \mid I^*(\mathbf{x})) \tag{4.23}$$

that is given by the following relation, for each grey level i in I:

$$\hat{I}(i) = \sum_j j \, \frac{p_{II^*}(i,j)}{p_{I^*}(i)}. \tag{4.24}$$

The registration process minimizing to zero the SCV is then expressed through the following equation:

$$\widehat{\Delta\mathbf{u}} = \arg\min_{\Delta\mathbf{u}} \sum_{n=1}^{N_\mathbf{x}} \left[I^*(w(\mathbf{x}_n, \Delta\mathbf{u})) - \hat{I}(w(\mathbf{x}_n, \mathbf{u})) \right]^2 \tag{4.25}$$

where $\mathbf{u}$ is the vector containing the parameters of a TPS warp. Transforming a point $\mathbf{x} = [x\ y]$ into $w(\mathbf{x}, \mathbf{u})$ thanks to this warp is done thanks to the following relation:

$$w(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} a_0 & a_1 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_2 \\ a_5 \end{pmatrix} + \sum_{k=1}^{N_p} \begin{pmatrix} w_x^k \\ w_y^k \end{pmatrix} \phi(d^2(\mathbf{x}, \mathbf{c}_k)). \tag{4.26}$$

The vector $\mathbf{u}$ then contains both the affine parameters and the weights associated to each control point:

$$\mathbf{u}^\top = (a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ \mathbf{w}_x^\top\ \mathbf{w}_y^\top). \tag{4.27}$$

where $\mathbf{w}_x^\top$ and $\mathbf{w}_y^\top$ are vectors composed of the weighting criterion for each control point:

$$\mathbf{w}_x^\top = \begin{pmatrix} w_x^0 & \dots & w_x^{N_p-1} \end{pmatrix} \tag{4.28}$$

$$\mathbf{w}_y^\top = \begin{pmatrix} w_y^0 & \dots & w_y^{N_p-1} \end{pmatrix}. \tag{4.29}$$

A first order Taylor derivation of the cost function yields:

$$SCV(\Delta\mathbf{u}) = \sum_{n=0}^{N_\mathbf{x}} \left[I^*(w(\mathbf{x}_n, \Delta\mathbf{u})) + \nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} - \hat{I}(w(\mathbf{x}_n, \mathbf{u})) \right]^2. \tag{4.30}$$

where ∇I^* is the gradient of the image at the pixel $w(\mathbf{x}_n, \Delta\mathbf{u})$. In order to minimize this value, we compute its derivative vector and nullify it as it will be equal to zero upon reaching the desired minimum:

$$\frac{\partial SCV}{\partial \Delta\mathbf{u}} = 2 \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[I^*(w(\mathbf{x}_n, \Delta\mathbf{u})) + \nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \Delta\mathbf{u} - \hat{I}(w(\mathbf{x}_n, \mathbf{u})) \right] = 0. \tag{4.31}$$

For clarity purposes, let us define H as the Gauss-Newton approximation of the Hessian matrix of the SCV:

$$H = \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[\nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \right] \tag{4.32}$$

Finally, we can express the warp parameter increment computation thanks to this matrix and the cost function:

$$\Delta\mathbf{u} = H^{-1} \sum_{n=0}^{N_\mathbf{x}} \left[\nabla I^* \frac{\partial w}{\partial \Delta\mathbf{u}} \right]^\top \left[I^*(w(\mathbf{x}_n, \Delta\mathbf{u})) - \hat{I}(w(\mathbf{x}_n, \mathbf{u})) \right]. \tag{4.33}$$

To be able to perform this optimization, we then need to know the derivative the warp function with relation to its parameters $\frac{\partial w}{\partial \Delta\mathbf{u}}$. Let us then show how to compute that matrix. In order to derive the expression of the warped point with relation to its parameters, let us first develop equation (4.26). The warped point $w(\mathbf{x}, \Delta\mathbf{u}) = [x_w\ y_w]^\top$ can be expressed as:

$$\begin{aligned} x_w &= a_0\ x + a_1\ y + a_2 + \sum_{k=1}^{N_p} \mathbf{w}_x[k]\, \phi(d^2(\mathbf{x}, \mathbf{c}_k)) \\ y_w &= a_3\ x + a_5\ y + a_5 + \sum_{k=1}^{N_p} \mathbf{w}_y[k]\, \phi(d^2(\mathbf{x}, \mathbf{c}_k)). \end{aligned} \tag{4.34}$$

The derivative of the warped point $w(\mathbf{x}, \Delta\mathbf{u})$ with relation to its warp parameters is then given by:

$$\frac{\partial w(\mathbf{x}, \Delta\mathbf{u})}{\partial \Delta\mathbf{u}} = \begin{pmatrix} \mathbf{J_A} & \mathbf{J_\Omega} \end{pmatrix}. \tag{4.35}$$

where:

$$\mathbf{J_A} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \tag{4.36}$$

$$\mathbf{J}_{\mathbf{\Omega}} = \begin{pmatrix} \phi(d^2(\mathbf{x},\mathbf{c}_0)) & \dots & \phi(d^2(\mathbf{x},\mathbf{c}_k)) & 0 & \dots & 0 \\ 0 & \dots & 0 & \phi(d^2(\mathbf{x},\mathbf{c}_0)) & \dots & \phi(d^2(\mathbf{x},\mathbf{c}_k)) \end{pmatrix}. \tag{4.37}$$

SCV-BASED NON-RIGID 2D REGISTRATION

Initialization

- Initialize a grid a control points on the reference image.
- Compute the derivative of I^* with relation to the warp parameters $\nabla I^* \frac{\partial w}{\partial \Delta \mathbf{u}}$.

Tracking phase (iterate until $\Delta\mathbf{u}$ is small enough)

- Warp every points of I thanks to the pose parameters $\mathbf{r}$.
- Compute the probability density functions p_{II^*} and p_I.
- Compute the cost function $SCV(\Delta\mathbf{u})\Big|_{\Delta\mathbf{r}=u}$.
- Compute the warp increment $\Delta\mathbf{u}$.
- Get the inverse parameters $\Delta\mathbf{u}^{-1}$ and update $\mathbf{u}$.

4.4 Using the MI for dense non-rigid 2D registration

Finally, let us show how we propose to use the MI as the cost function for a dense non-rigid 2D registration technique. The main idea behind the approach is close to the one used to define our 3D registration technique exposed in section 3.4. For this algorithm we do not want to use any learning either, as with the previous SCV-based algorithm. Let us then define the cost function that will be used in the proposed approach.

$$\widehat{\Delta\mathbf{u}} = \arg\max_{\Delta\mathbf{u}} MI(\Delta\mathbf{u}) = \arg\max_{\Delta\mathbf{u}} \sum_{i,j} p_{\overline{II^*}}(i,j) \log\left(\frac{p_{\overline{II^*}}(i,j)}{p_{\overline{I}}(i)p_{\overline{I^*}}(j)}\right) \tag{4.38}$$

where $p_{\overline{I}}(i)$ and $p_{\overline{I^*}}(j)$ are respectively the marginal probability density functions of $\overline{I}$ and $\overline{I^*}$ and $p_{\overline{II^*}}(i,j)$ is the joint probability density function of $\overline{I}$ and $\overline{I^*}$. Let us recall that $\overline{I}$ and $\overline{I^*}$ are the result of histogram binning of the current and reference images I and I^*. In order to maximize this value to an unknown value, we will minimize its gradient to zero. The gradient of the mutual information $\mathbf{G}$ is the derivative of $MI(\Delta\mathbf{u})$ with respect to the warp parameters $\Delta\mathbf{u}$. It is given by the following expression:

$$\mathbf{G} = \frac{\partial MI(\overline{I}(w(\mathbf{x},\mathbf{u})), \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u})))}{\partial \Delta\mathbf{u}}\Bigg|_{\Delta\mathbf{u}=\mathbf{0}}. \tag{4.39}$$

The detailed expression of $\mathbf{G}$ can be obtained from the derivation of equation (4.38) [Dowson and Bowden, 2006]:

$$\mathbf{G} = \sum_{i,j} \frac{\partial p_{\overline{II^*}}(i,j)}{\partial \Delta\mathbf{u}} \left(1 + \log\left(\frac{p_{\overline{II^*}}(i,j)}{p_{\overline{I}}(i)}\right)\right). \tag{4.40}$$

The values of the probability density functions can be obtained by taking into account the B-spline binning:

$$\begin{aligned} p_{\overline{II^*}}(i,j) &= \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \beta(i - \overline{I}(w(\mathbf{x},\mathbf{u}))) \, \beta(j - \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u}))) \\ p_{\overline{I}}(i) &= \sum_{j=0}^{d} p_{\overline{II^*}}(i,j). \end{aligned} \tag{4.41}$$

The value of $\beta(i - \overline{I}(w(\mathbf{x},\mathbf{u})))$ being the same no matter $\Delta\mathbf{u}$, the derivative needed in equation (4.40) can be expressed as:

$$\frac{\partial p_{\overline{II^*}}(i,j)}{\partial \Delta\mathbf{u}} = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \frac{\partial \beta(j - \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u})))}{\partial \Delta\mathbf{u}} \, \beta(i - \overline{I}(w(\mathbf{x},\mathbf{u}))). \tag{4.42}$$

The use of a B-spline function means that this derivative is easy to compute as it is given by:

$$\begin{aligned} \frac{\partial \beta(j - \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u})))}{\partial \Delta\mathbf{u}} &= -\frac{\partial \beta}{\partial j} \, \frac{\partial \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u}))}{\partial \Delta\mathbf{u}} \\ &= -\frac{\partial \beta}{\partial j} \, \nabla \overline{I^*} \, \frac{\partial w(\mathbf{x},\Delta\mathbf{u})}{\partial \Delta\mathbf{u}} \end{aligned} \tag{4.43}$$

The derivative of the warped point with relation to the warp parameters is presented given in equation (4.35). Now that the computation of $\mathbf{G}$ has been detailed, let us define the warp increment computation:

$$\Delta\mathbf{u} = -\mathbf{H}^{-1}\mathbf{G}. \tag{4.44}$$

The Hessian matrix of the mutual information $\mathbf{H}$ can, similarly to its gradient, be expressed thanks to the partial derivatives of the probability density functions:

$$\frac{\partial^2 MI(\Delta\mathbf{u})}{\partial \Delta\mathbf{u}^2}\bigg|_{\Delta\mathbf{u}=0} = \sum_{i,j} \frac{\partial p_{\overline{II^*}}}{\partial \Delta\mathbf{u}}^{\top} \frac{\partial p_{\overline{II^*}}}{\partial \Delta\mathbf{u}} \left(\frac{1}{p_{\overline{II^*}}} - \frac{1}{p_{\overline{I}}} \right) + \frac{\partial^2 p_{\overline{II^*}}}{\partial \Delta\mathbf{u}^2} \left(1 + \log\left(\frac{p_{\overline{II^*}}}{p_{\overline{I}}} \right) \right). \tag{4.45}$$

where $p_{\overline{II^*}}(i,j)$ is noted $p_{\overline{II^*}}$ for clarity purposes. The B-spline functions being twice differentiable, the second order derivative of the joint probability density function is given by:

$$\frac{\partial^2 p_{\overline{II^*}}}{\partial \Delta\mathbf{u}^2} = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \frac{\partial^2 \beta(j - \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u})))}{\partial \Delta\mathbf{u}^2} \, \beta(i - \overline{I}(w(\mathbf{x},\mathbf{u}))). \tag{4.46}$$

As for the first order derivative, chain derivation gives the expression of the second order derivative:

$$\begin{aligned} \frac{\partial^2 \beta(j - \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u})))}{\partial \Delta\mathbf{u}^2} &= \frac{\partial}{\partial \Delta\mathbf{u}} \left[-\frac{\partial \beta}{\partial j} \frac{\partial \overline{I^*}(w(\mathbf{x},\Delta\mathbf{u}))}{\partial \Delta\mathbf{u}} \right] \\ &\simeq \frac{\partial^2 \beta}{\partial j^2} \left[\nabla \overline{I^*} \, \frac{\partial w(\mathbf{x},\Delta\mathbf{u})}{\partial \Delta\mathbf{u}} \right]^{\top} \left[\nabla \overline{I^*} \, \frac{\partial w(\mathbf{x},\Delta\mathbf{u})}{\partial \Delta\mathbf{u}} \right]. \end{aligned} \tag{4.47}$$

MI-BASED NON-RIGID 2D REGISTRATION

Initialization

- Initialize a grid a control points on the reference image.
- Compute the derivative of I^* with relation to the warp parameters $\nabla I^* \frac{\partial w}{\partial \Delta \mathbf{u}}$.
- Compute the approached Hessian of the visible model $\mathbf{H}$ and its inverse $\mathbf{H}^{-1}$.

Tracking phase (iterate until $\Delta \mathbf{u}$ is small enough)

- Warp every points of $\overline{I}$ thanks to the pose parameters $\mathbf{r}$.
- Compute the probability density functions $p_{\overline{II^*}}$ and $p_{\overline{I^*}}$ thanks to histogram binning.
- Compute $\mathbf{G}$ the Jacobian of the MI.
- Compute the warp increment $\Delta \mathbf{u}$.
- Get the inverse parameters $\Delta \mathbf{u}^{-1}$ and update $\mathbf{u}$.

4.5 Non-rigid 2D registration experimental validations

In order to evaluate the non-rigid 2D registration approaches proposed in this document, several experiments have been realized. The algorithm is initialized by defining the four corners of the tracked template on the first image of the sequence. A regular grid of control points is then initialized on the template. This grid is warped at each new frame and overlaid on the template in order to visualize the results of each experiment. After the results of the proposed algorithms have been detailed in several situations, the convergence domains of both methods are evaluated through the same principle as proposed in chapter 3.

4.5.1 Tracking in nominal conditions

The goal of this experiment is to validate the proposed approaches in nominal conditions. The two proposed algorithms are then launched on sequences on which a t-shirt is moved and deformed. From a starting position extracted on the first image of the sequence, the tracking is launched and results are recorded. Results show that even when the deformations are important the proposed algorithms can cope with them and keep a good tracking quality (see figure 4.5 for examples). Several other experiments have been carried out and their results are shown on figure 4.6. The first one is realized in order to assess the robustness to "out of the plane" motion. To do that, a piece of paper is folded, creating a deformation that alters the perspective of the object. Even though the precision of the results is lowered at several frames along the sequence due to the bending energy constraints, the tracking is still successful in those conditions. Another experiment is then performed in order to assess the robustness of the approaches to template compression. Even though no additional layer is used here to prevent the template to fold on itself (or to detect it as was proposed in [Gay-Bellile et al., 2010]), the registration is still able to cope with that situation by considering the lost zone as a small occlusion.

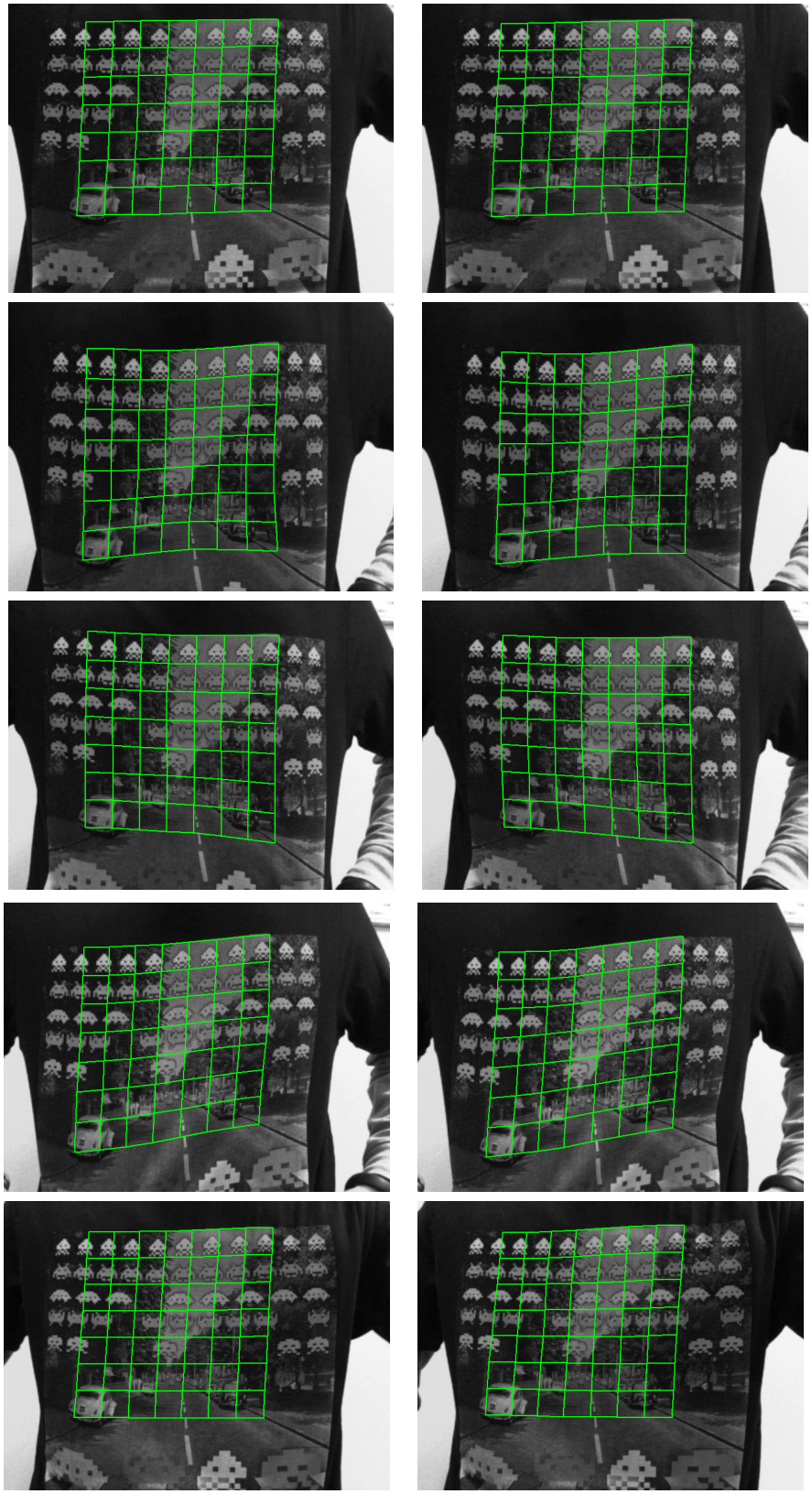


Figure 4.5: Results of the tracking algorithms in nominal conditions. Images show that the SCV (on the left) and MI (on the right) approaches give similar results in nominal conditions.

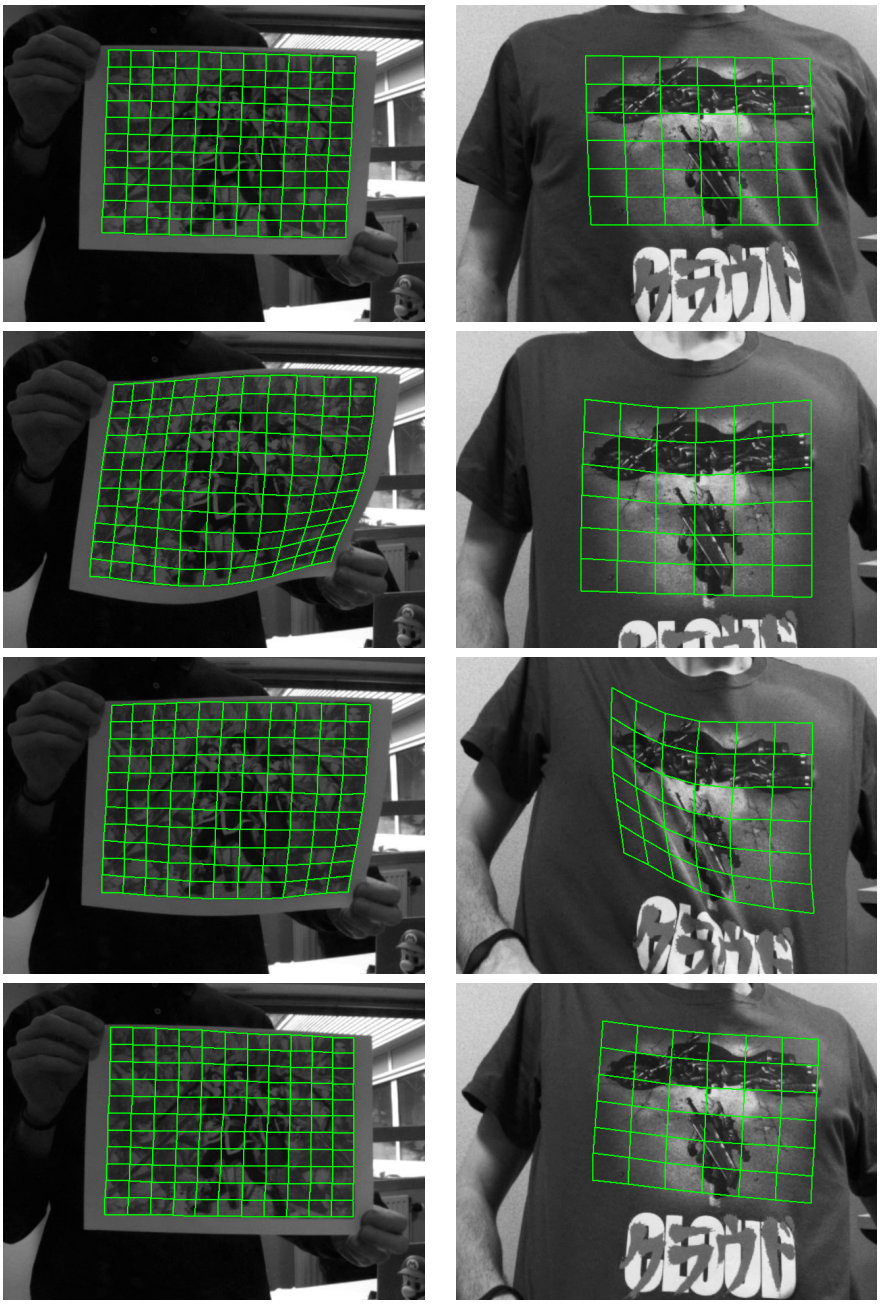


Figure 4.6: Results of the tracking algorithms when confronted to perspective effects (on the left) and compression/extension (on the right).

4.5.2 Tracking in perturbed conditions

The next experiments aim at validating our approaches in perturbed conditions. Several types of perturbations were considered. First, a global variation was taken into account in the form of a light variation. Then, the results of the MI version are shown on non planar objects and when confronted to local light variations.

4.5.3 Tracking in light changing conditions

In order to test the approaches when confronted to global light variations, the algorithms were launched on a video sequence similar to the one used for nominal conditions testing. It features a t-shirt being deformed but this time, the light of the room is switched off and on in order to affect the global illumination setting. Results displayed on figure 4.7 show that even in these conditions, the two approaches are able to register the template even though the light changes drastically. Let us note that a SSD approach was tested on this sequence and failed when the light is turned off for the first time.

4.5.4 Tracking when confronted to occlusions

In order to test the registration techniques proposed in this document when confronted to occlusions, the algorithms were launched on a t-shirt sequence in which an important occlusion appeared. The use of mutual information allows the approach to be robust in those conditions as the results show on figure 4.8. The SCV, as expected, is not able to cope with this and fails to register correctly when the occlusion becomes really noticeable. Let us note that the same experiment was realized using the SSD as our registration criterion which also led to a failure of the registration when the occlusion appeared in the template.

4.5.5 Tracking non planar objects

The next experiments were realized in order to evaluate our approaches on non-planar scenes. The first sequence represents a deflating balloon, a similar experiment as what was proposed in [Malis, 2007]. This creates a change in scale and a deformation as the volume of the balloon decreases. But thanks to the TPS warp the tracking is still successful since the deformation is taken into account in the algorithm (see figure 4.9). Let us note that results are shown only for the MI approach as the SCV fails to register the template properly in these conditions. Then, an experiment was realized involving important perspective effects. Although the displacement creates an important self occlusion the use of mutual information in coordination with the non-rigid warp allows the tracking to be successful both when the object is moved and when it is put back into its original position as shown on figure 4.10. As was the case for the last experiment SCV and SSD-based approaches failed to register the template properly, diverging when the self occlusion appears.

4.5.6 Empirical convergence domain analysis

Finally, an experiment was realized in order to analyze the convergence domain of the proposed approaches in different conditions. After the trackers have been initialized on the first frame of a sequence, the algorithms are launched and the resulting warp parameters estimated are recorded on chosen frames. A gaussian noise is then applied to those recorded parameters and the tracking phase is performed. It is considered as successful if in a given number of iterations (here 50) the mean distance by pixel between the warped control

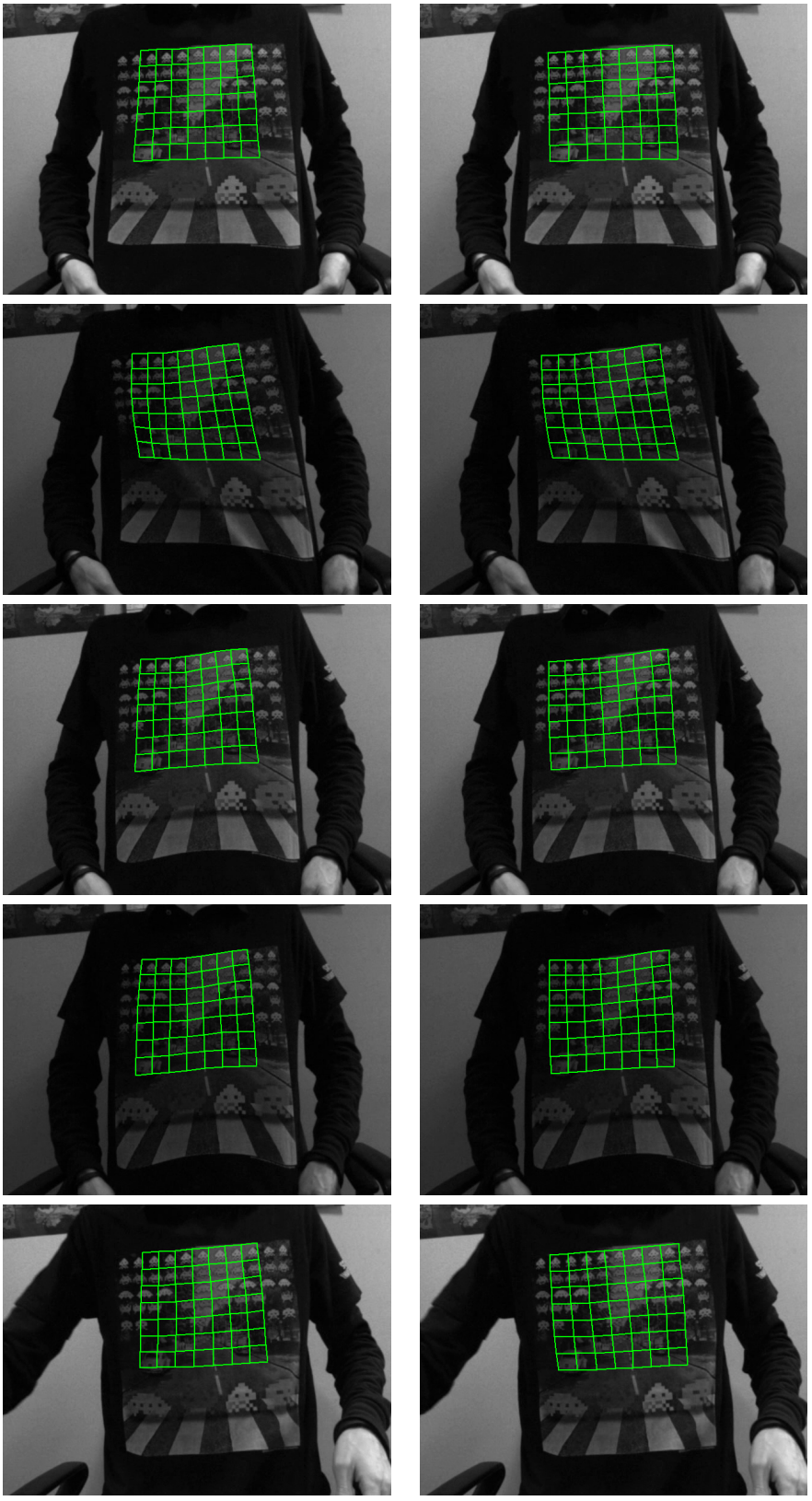


Figure 4.7: Results of the SCV (on the left) and MI (on the right) non-rigid tracking algorithms when confronted to light variations.

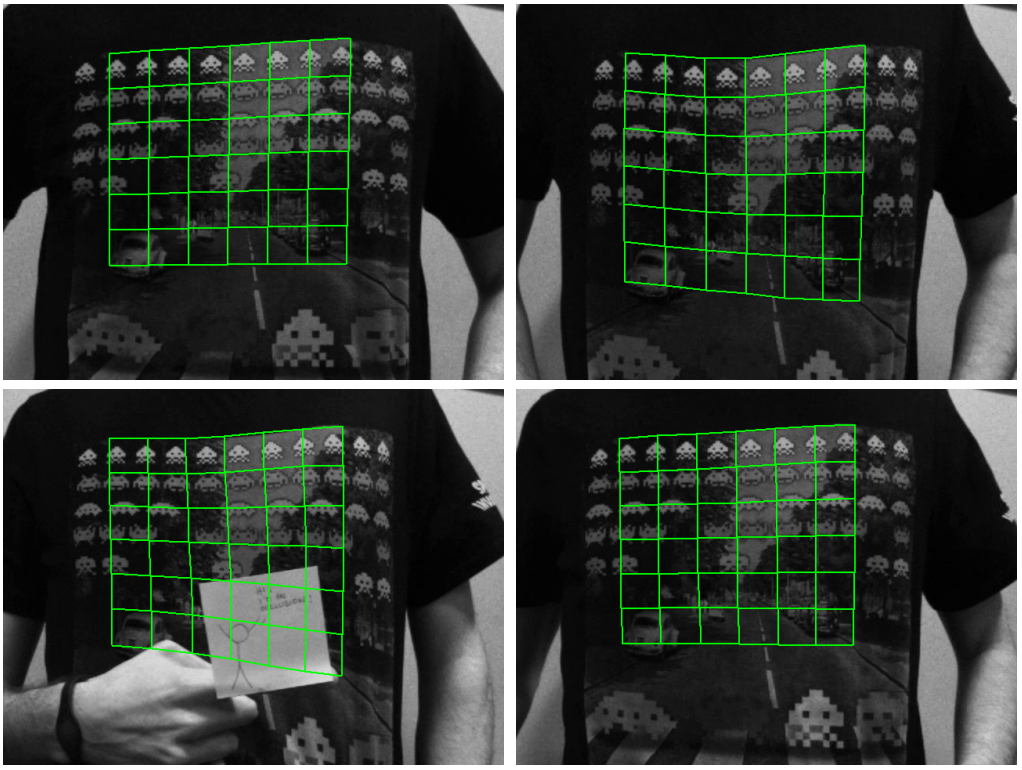


Figure 4.8: Results of the tracking algorithm on an occluded object. Results are shown for the MI approach which is the only successful one in this situation.

points and the recorded points is inferior to a defined value as was done in [Dame and Marchand, 2010, Dame and Marchand, 2012, Lieberknecht et al., 2009]. In this case that value is set to 1 pixel. This is done 500 times per frame using different noise kernels. Results are shown on figures 4.12 and 4.13. When applying the gaussian noise, a gaussian kernel σ_r of 0.01 is used for the rotation parameters of the affine transformation and several kernels σ_t ranging from 2 to 16 are used for the translation parameters giving a difference in starting positions up to 48 pixels on each axis. Figure 4.11 shows examples of considered starting positions. Results show several interesting observations. First, the SCV curves show that the method is not impacted by deformations or global light variations. This is exposed by the fact that the corresponding curves are very similar in these conditions. It comes to validate the fact that the SCV is not affected at all by global histogram variations. It is, however, greatly impacted by local variations as the occlusion curve shows. On the other hand, the MI approach is impacted by changes in the histogram, whether local or global, but shows a wider convergence domain in the conditions in which the tests are performed. This can be observed on the graphics, as the MI curves show a higher rate of perfect convergence whatever the noise setting is. Let us also note that using more control points have shown to produce more accurate results and expand the convergence domain, but since using more points increases the computation time of the algorithm, a middle ground between efficiency and accuracy can be found by tuning the registration parameters.

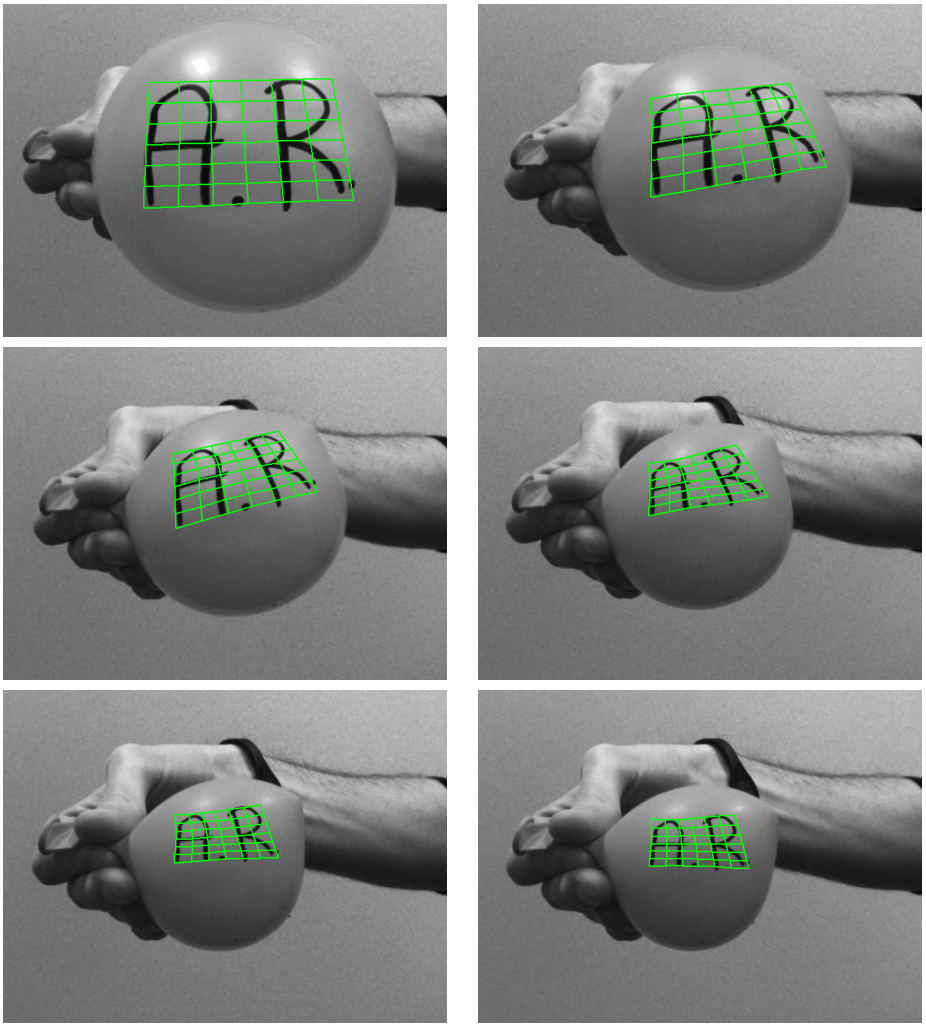


Figure 4.9: Results of the tracking algorithm on a non planar object. There again the MI registration approach is successful thanks to the non-rigid warp function.

4.6 Conclusion

In this chapter, two new approaches have been proposed that optimize the parameters of a non rigid warp functions in order to perform 2D registration robust to template deformations. Taking advantage of the possibilities given by the thin-plate spline warp function, a first approach is presented that optimizes the SCV in order to get a simple process that is robust to global perturbations of the scene. This is validated through several experimentations which show the method to be robust to these conditions. Then, the approach has been extended to the use of the MI in order to add more robustness. Again, several experiments validate the method on sequences featuring scene perturbations. An empirical estimation of the convergence domain of both approaches is finally provided.

Taking advantage of the strong duality between visual tracking and visual servoing, the next part will aim at defining dense visual servoing techniques robust to scene variations thanks to the robust (dis)similarity functions used in this part.

Figure 4.10: Results of the tracking algorithm on a three dimensional object. There again the tracking is successful even when a part of the object is occluding another one.

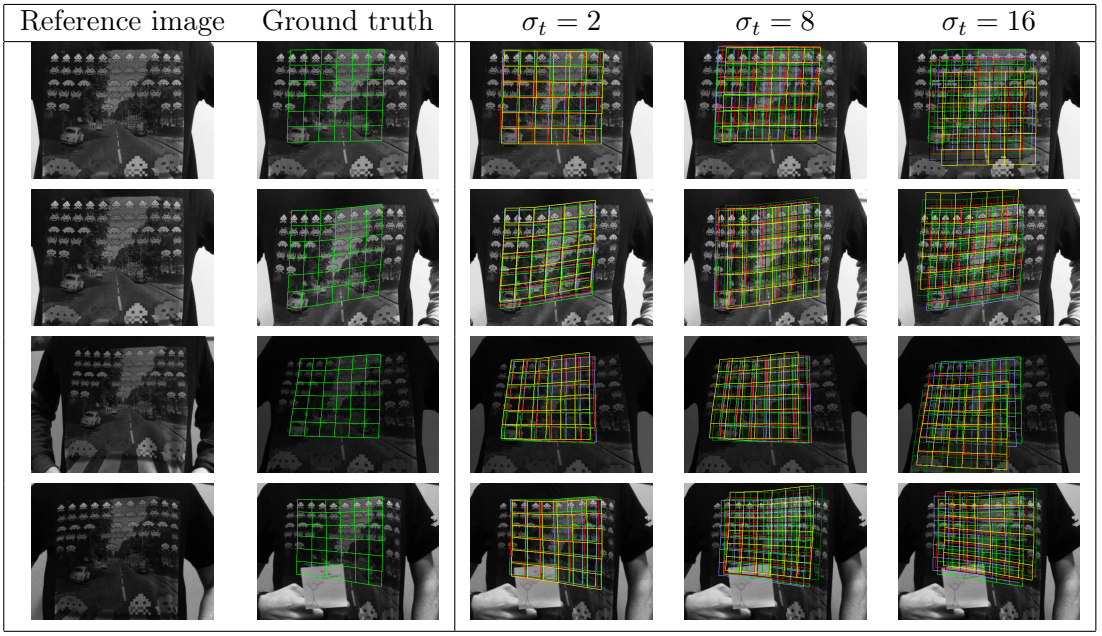


Figure 4.11: Examples of starting positions for convergence domain analysis purposes with different σ_t. For all experiments $\sigma_r = 0.01$.

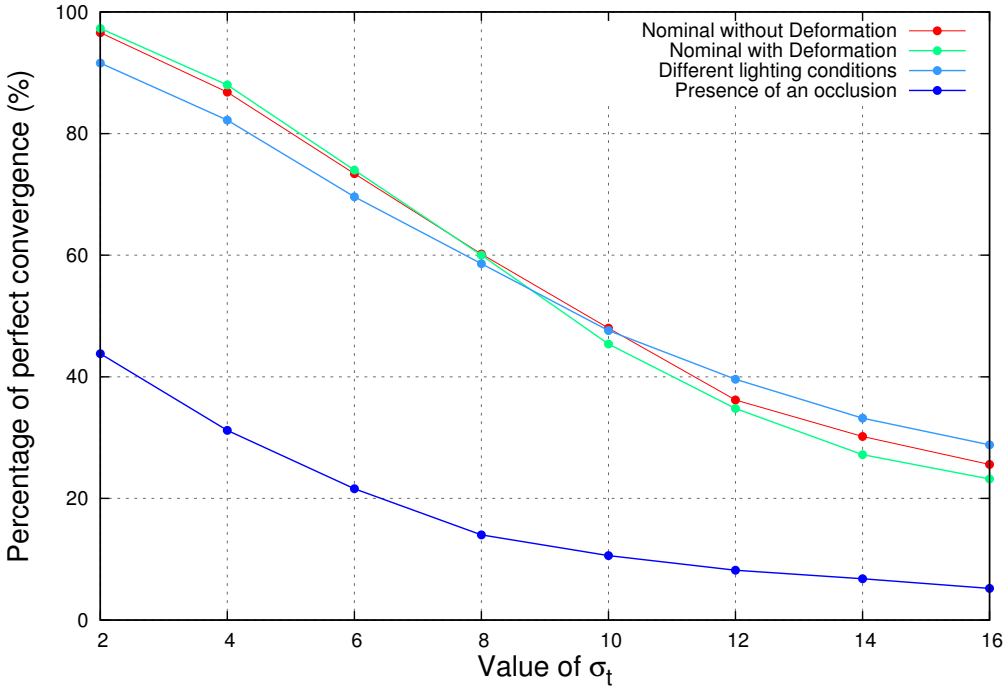


Figure 4.12: Convergence frequency of the SCV approach in nominal conditions without deformation (row 1 of fig. 4.11), with deformations (row 2 of fig. 4.11), when confronted to light variations (row 3 of fig. 4.11) and when confronted to occlusions (row 4 of fig. 4.11).

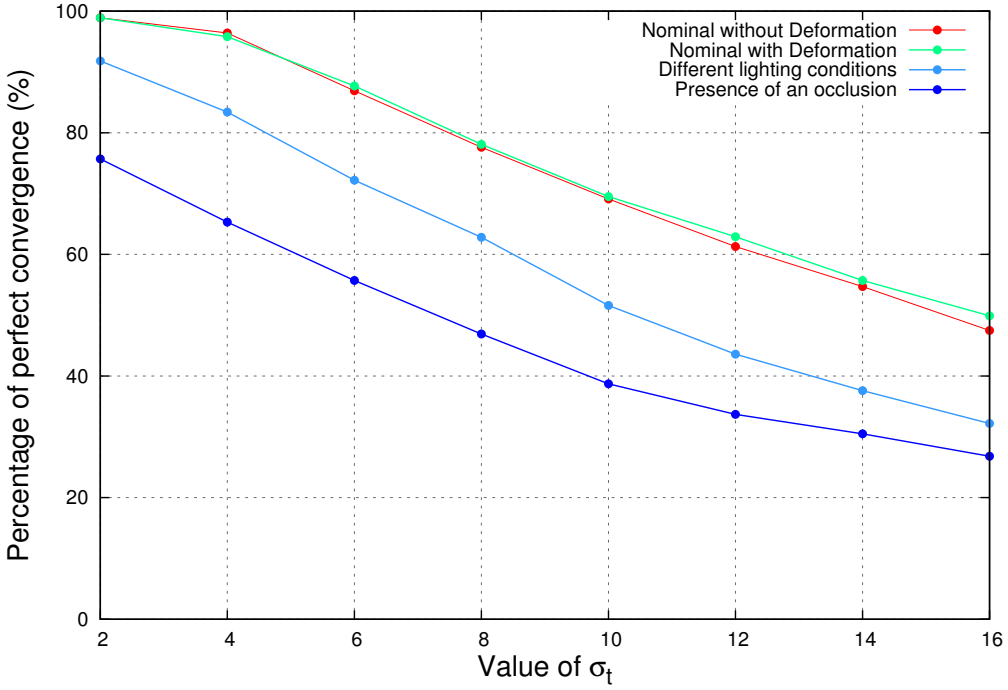


Figure 4.13: Convergence frequency of the MI approach in nominal conditions without deformation (row 1 of fig. 4.11), with deformations (row 2 of fig. 4.11), when confronted to light variations (row 3 of fig. 4.11) and when confronted to occlusions (row 4 of fig. 4.11).

PART 2

Visual Servoing

Visual servoing is the process allowing the control of a dynamic system, say a robot, thanks to visual data gathered by one or several cameras. The possibilities created by the use of vision sensors for robot control exceed what can be obtained with other types of sensors such as ultrasonic, acceleration or force sensors. Cameras can either be located on the end effector of the robot, thus creating an aye-in-hand system as will be the case of study in this document, or can only observe the machine, creating an eye-to-hand system. The possibilities created by visual servoing are numerous, ranging from positioning tasks to more complex navigation processes.

Previous chapters have shown how the choice of the cost function for visual tracking algorithms allows to achieve robustness of the process with no need for added robustness steps. Since visual tracking and visual servoing are based on the same objective (aligning a current view to a reference), a strong duality exists between both domains. This is why in this part several approaches to visual servoing are designed based on the same robust cost functions as considered for the previously proposed tracking algorithms.

5

Visual servoing overview

This chapter consists in a quick survey of common visual servoing approaches. The main strategies used to perform visual servoing tasks are defined and their advantages and drawbacks are then discussed.

Visual servoing was introduced in the late 80's early 90's in order to control robots thanks to cameras [Weiss, 1984, Feddema and Mitchell, 1989, Rives et al., 1989, Corke and Paul, 1990, Hashimoto, 1993]. Considering visual servoing, the goal is to control the robot thanks to visual data gathered by a camera. In order to be successful, the servoing task must therefore link the motion of the robot with the visual data. Figure 5.1 represents the classical visual servoing loop:

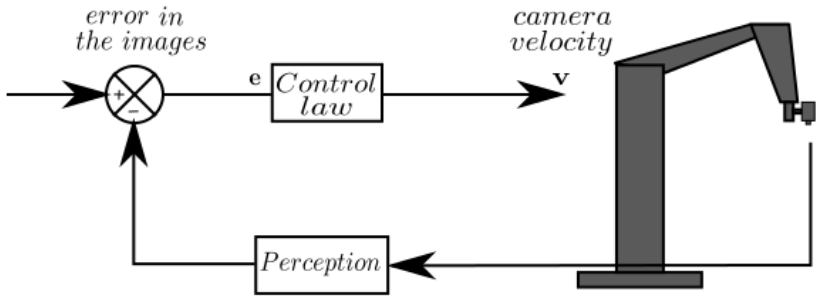


Figure 5.1: Classical visual servoing loop. The perception allows to evaluate a task $\mathbf{e}$ in order to compute a camera velocity $\mathbf{v}$. The robot is controlled according to a comparison between the desired and perceives visual data.

At each new iteration, the pose $\mathbf{r}$ of the robot is updated so as to regulate an error $\mathbf{e}(\mathbf{r})$ computed from visual data. The visual servoing task compares the current visual features $\mathbf{s}(\mathbf{r})$ extracted at the current pose $\mathbf{r}$ and the same features extracted at the desired position $\mathbf{s}^*$ in order to compute the velocity vector $\mathbf{v}$ that will be applied to the robot. The process is iterated until the error $\mathbf{e}(\mathbf{r})$ between $\mathbf{s}(\mathbf{r})$ and $\mathbf{s}^*$ is null, at what point the robot has reached the optimal pose $\hat{\mathbf{r}}$ which is equal to $\mathbf{r}^*$ if the task is successful. The servoing task can then be expressed as:

$$\hat{\mathbf{r}} = \arg\min_{\mathbf{r}} \|\mathbf{e}(\mathbf{r})\| \qquad \text{where} \qquad \mathbf{e} = \mathbf{s}(\mathbf{r}) - \mathbf{s}^* \tag{5.1}$$

with $\hat{\mathbf{r}}$ the optimal pose reached after the optimization which is successful if $\hat{\mathbf{r}}$ is equal to $\mathbf{r}^*$.

As for the tracking processes presented in the previous chapters, visual servoing approaches can be divided into two types of methods. On one hand the approaches that

define $\mathbf{e}(\mathbf{r})$ as a dissimilarity between two sets of geometrical features, therefore needing an algorithmic layer to extract and match primitives before performing the control, and on the other hand strategies that chose to optimize a (dis)similarity function between templates of pixel intensities in the images.

5.1 Positioning task using geometrical visual servoing

Most approaches in the literature use 2D or 3D geometrical features computed from extracted and already matched data from the images seen by the camera.Those features are then used to optimize a cost function, moving the robot to the desired position. To that end, the visual features seen at the current position are stored in a vector $\mathbf{s}(\mathbf{r})$ that will be compared with the features seen at the desired position $\mathbf{s}^*$. By specifying that the error $\mathbf{e}(\mathbf{r})$ decreases exponentially, therefore imposing $\dot{\mathbf{e}}(\mathbf{r}) = -\lambda\mathbf{e}(\mathbf{r})$, the motion of the robot can be controlled. This induces that when the robot is far from the goal the norm of its velocity will be greater than when it is almost to the desired position. The desired features are fixed and therefore constant all along the process. The derivative of the error function $\mathbf{e}(\mathbf{r})$, referred to as $\dot{\mathbf{e}}(\mathbf{r})$, is then equal to the variation of the features seen by the camera $\dot{\mathbf{s}}$ with relation to the camera position. The motion of the visual features in the image is also directly related to the camera velocity such as:

$$\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v} \tag{5.2}$$

where $\mathbf{L_s}$ is the interaction matrix associated to the visual servoing task [Chaumette and Hutchinson, 2006] that links the variation of the observed visual features in the image plane to the velocity vector $\mathbf{v}$ applied to the camera. This yields:

$$\mathbf{L_s}\mathbf{v} = -\lambda(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*). \tag{5.3}$$

And finally, the velocity used to regulate $\mathbf{e}(\mathbf{r})$ to zero is given by:

$$\mathbf{v} = -\lambda\mathbf{L_s^+}\mathbf{e}(\mathbf{r}) \tag{5.4}$$

with $\mathbf{L_s^+}$ the pseudo-inverse matrix of $\mathbf{L_s}$.As this formulation is only correct when $\mathbf{L_s}$ is a matrix of full-rank, the visual servoing process requires at least as much visual features

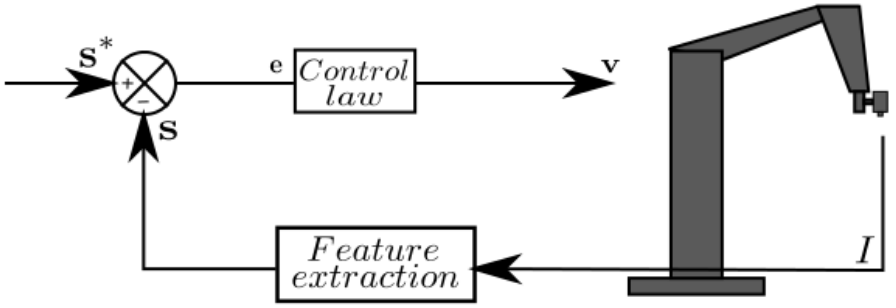


Figure 5.2: Classical visual servoing loop using 2D points. The points $\mathbf{s}$ are extracted from the current image I and compared to their desired counterpart $\mathbf{s}^*$ in order to compute the velocity vector $\mathbf{v}$.

at degrees of freedom it is to control on the robot. A wide variety of geometrical visual features have been used over the years. Basic 2D points (see section 5.1.1 for more details) have been used first [Rives et al., 1989]. The process was later extended to more complex features such as lines, spheres or cylinders [Chaumette and Rives, 1990, Espiau et al., 1992a].

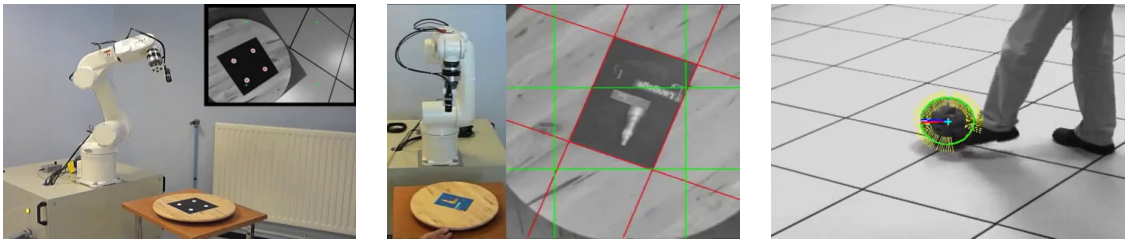


Figure 5.3: Visual servoing examples using, from left to right, points, lines and ellipses. See full videos: https://www.youtube.com/user/VispTeam/videos

5.1.1 2D geometrical visual servoing: using point features

In this section, the basis of geometrical visual servoing will be exposed. The example of a simple visual servoing task using only four points extracted from the image seen by the camera attached to the robot can be found in [Chaumette and Hutchinson, 2006]. The objective of the task is then to minimize the distance in the image between the position of each of the four points extracted from the current image $\mathbf{x}_i = (x,\ y), i = 1 \dots 4$ and their desired position $\mathbf{x}_i^*$. The current and desired features $\mathbf{s}(\mathbf{r})$ and $\mathbf{s}^*$ are then defined as:

$$\begin{aligned} \mathbf{s}(\mathbf{r}) &= (\mathbf{x}_0(\mathbf{r}),\ \mathbf{x}_1(\mathbf{r}),\ \mathbf{x}_2(\mathbf{r}),\ \mathbf{x}_3(\mathbf{r})) \\ \mathbf{s}^* &= (\mathbf{x}_0^*,\ \mathbf{x}_1^*,\ \mathbf{x}_2^*,\ \mathbf{x}_3^*)\,. \end{aligned} \tag{5.5}$$

Since the objective is to minimize the distance between $\mathbf{s}$ and $\mathbf{s}^*$, the error $\mathbf{e}$ to be regulated is given by:

$$\mathbf{e}(\mathbf{r}) = \mathbf{s}(\mathbf{r}) - \mathbf{s}^*. \tag{5.6}$$

In order to compute the control law expressed in equation (5.4), the interaction matrix $\mathbf{L_s}$ related to $\mathbf{s}(\mathbf{r})$ needs to be defined. The interaction matrix links the variation of the visual feature to the camera velocity $\mathbf{v}$. Since $\mathbf{s}^*$ is constant through the process, the interaction matrix $\mathbf{L_s}$ is given by:

$$\mathbf{L_s} = \begin{bmatrix} \mathbf{L_{x_0}} \\ \mathbf{L_{x_1}} \\ \mathbf{L_{x_2}} \\ \mathbf{L_{x_3}} \end{bmatrix} \tag{5.7}$$

where $\mathbf{L_x}$ is the interaction matrix associated to a 2D point $\mathbf{x}$. This matrix then translates the motion of the point $\mathbf{x}$ in the current image when the camera moves with the robot. In order to get its expression, let us recall how a point in metric space $\mathbf{x} = [x,\ y]$ is expressed thanks to its 3D coordinates ${}^c\mathbf{X} = [X,\ Y,\ Z]$: $\mathbf{x} = [X/Z,\ Y/Z]$. The evolution of the x and y coordinates of $\mathbf{x}$, referred to respectively as $\dot{x}$ and $\dot{y}$ can then be expressed as:

$$\begin{aligned} \dot{x} &= \frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2} = \frac{\dot{X} - x\dot{Z}}{Z} \\ \dot{y} &= \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} = \frac{\dot{Y} - y\dot{Z}}{Z}. \end{aligned} \tag{5.8}$$

In order to compute $\dot{\mathbf{x}}$, the derivatives of the 3D point ${}^c\mathbf{X}$ are needed. Assuming that the object does not move during the servoing task, the evolution $\dot{\mathbf{X}}$ of ${}^c\mathbf{X}$ depends only on the camera velocity $\mathbf{v}$. It is defined as $\mathbf{v} = (\boldsymbol{\nu},\ \boldsymbol{\omega})$, where $\boldsymbol{\nu}$ is the translational velocity and $\boldsymbol{\omega}$ is the angular velocity of the camera. The expression of $\dot{\mathbf{X}}$ is then given by the following relation:

$$\dot{\mathbf{X}} = -\boldsymbol{\nu} - \boldsymbol{\omega} \times \mathbf{X} \Longleftrightarrow \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -\nu_x - \omega_y Z - \omega_z Y \\ -\nu_y - \omega_z X - \omega_x Z \\ -\nu_z - \omega_x Y - \omega_y X \end{bmatrix} \tag{5.9}$$

where $\times$ denotes a cross product. Expressing equation (5.8) with the details given by equation (5.9) then allows to express the evolution of the point in the image with relation to the camera velocity:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{-\nu_x}{Z} + \frac{x\nu_z}{Z} + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ \frac{-\nu_y}{Z} + \frac{y\nu_z}{Z} + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{bmatrix}. \tag{5.10}$$

This equation can finally be written to satisfy the relation:

$$\dot{\mathbf{x}} = \mathbf{L}_\mathbf{x}\mathbf{v} \tag{5.11}$$

where:

$$\mathbf{L}_\mathbf{x} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & -(1+x^2) & -xy & -x \end{bmatrix}. \tag{5.12}$$

With $\mathbf{L}_\mathbf{x}$ now defined, the servoing task can be computed as:

$$\mathbf{v} = -\lambda \mathbf{L}_\mathbf{s}^{+} \left(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*\right). \tag{5.13}$$

Geometrical features-driven visual servoing needs an efficient and robust tracking layer that will detect, extract and match visual features over frames. That algorithmic layer can be problematic as it can be time consuming. It can also be faulty which can cause the servoing task to fail. Moreover, extracting data means not taking into account a large part of the available visual data. Other problems can arise as for example the visibility of the features that need to be insured and can then require to complexify the control law by switching between several optimization schemes in order to keep them into the image as was proposed in [Kermorgant and Chaumette, 2011].

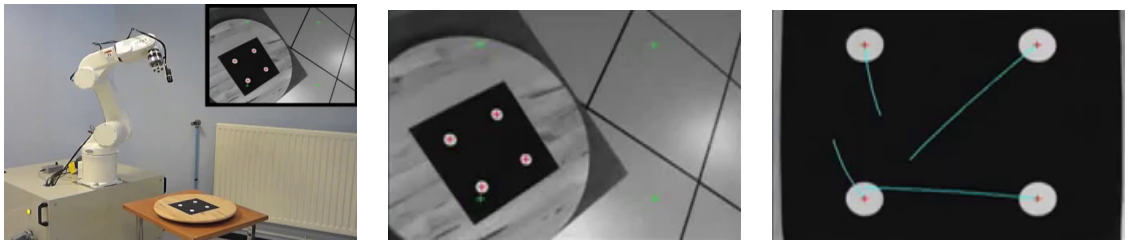


Figure 5.4: Classical visual servoing loop using four points. From left to right: Setup of the experiment - initial current image - final current image. See full video: https://www.youtube.com/watch?v=RJRma6xZ9G4

5.2 Using template tracking to perform visual servoing

Another way to perform visual servoing is to use the results given by a template tracking algorithm in order to localize the robot. In that case, the goal becomes to align the current

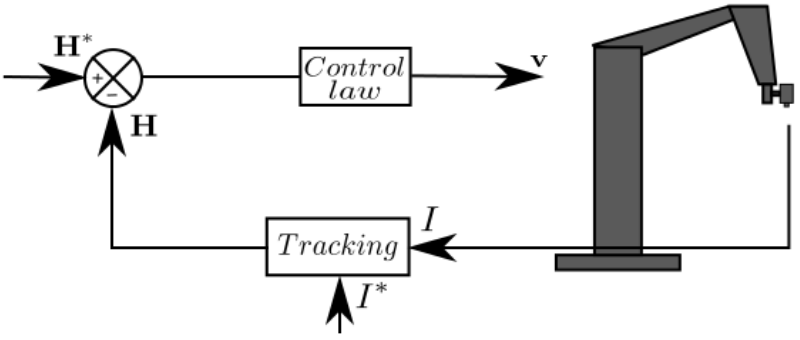


Figure 5.5: Classical visual servoing loop using a tracking process. The current warp parameters are computed on the current image I and compared to their desired value in order to compute the velocity vector $\mathbf{v}$.

image with the desired one by controlling the robot in order to obtain a desired registration result. Several approaches have been proposed that perform visual servoing following that strategy. For example, homographies have been used in [Vargas and Malis, 2005] and [Benhimane and Malis, 2007] to design a visual servoing framework, although they both chose to use the homography parameters are different. In [Vargas and Malis, 2005], it has been proposed to decompose the homography parameters in order to evaluate the 3D transformation between the current and desired position. In [Benhimane and Malis, 2007] on the other hand, the authors propose to use a tracking algorithm ([Benhimane and Malis, 2004]) to estimate the current homography which is then used directly to compute the control law of the task. Other methods have also been proposed [Crétual and Chaumette, 1998] that use camera motion parameters estimated using [Odobez and Bouthemy, 1995] in order to minimize the error between the motion of the camera and a desired motion.

5.3 Positioning task by direct visual servoing

In order to avoid any extraction or tracking process and to use the data given by the entire image, another visual servoing strategy has been proposed that uses the whole image as a feature. Using the pixel intensities as visual features, several approaches have been proposed, using different (dis)similarity functions in order to perform visual servoing. Let us note that there is a clear link between the methods presented in the first part of this document and their visual servoing counterparts.

5.3.1 Photometric visual servoing

A new approach has been proposed by the works of [Collewet et al., 2008, Collewet and Marchand, 2011] that uses directly the pixel luminance as visual feature which aims at minimizing to zero the sum of squared differences between the current image and the image recorded at the desired position. The image itself is then the visual feature considered by the control law, which yields:

$$\mathbf{s}(\mathbf{r}) = \mathbf{I}(\mathbf{r}) = [I(\mathbf{x}_{00}, \mathbf{r}) \dots I(\mathbf{x}_{mn}, \mathbf{r})] \tag{5.14}$$

where $m \times n$ is the size of the image I. This leads to a visual servoing task which is performed by minimizing to zero the SSD between the current and desired images:

$$\mathbf{e}(\mathbf{r}) = (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*)^\top (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*) \tag{5.15}$$

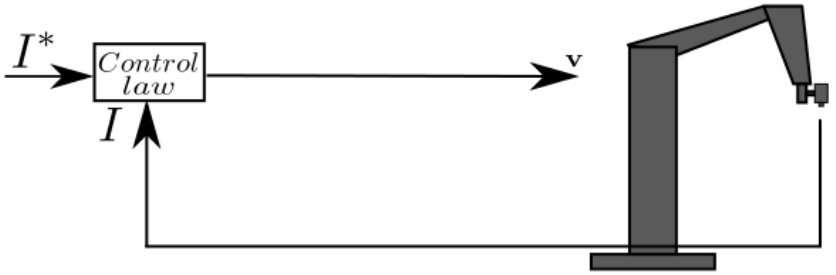


Figure 5.6: Classical dense visual servoing loop. The current image is directly compared to the image recorded at the desired position in order to compute the velocity vector $\mathbf{v}$.

leading to a classical visual servoing task velocity update given by:

$$\mathbf{v} = -\lambda \mathbf{L}_{\mathbf{I}}^{+} \left(\mathbf{I}(\mathbf{r}) - \mathbf{I}^{*} \right) \tag{5.16}$$

with $\mathbf{L_I}$ the interaction matrix of the dense feature $\mathbf{I}(\mathbf{r})$. The approach being based on the SSD, this means that it relies, as the KLT (see section 2.2.2.1), on the constant luminosity hypothesis. The luminance resulting from the projection of the 3D scene onto the image is therefore considered invariant during the whole positioning task for any point $\mathbf{x}$ in the image:

$$I(\mathbf{x} + d\mathbf{x}, t + dt) = I(\mathbf{x}, t) \tag{5.17}$$

where $d\mathbf{x}$ represents the motion featured in the image after the lapse of time dt. A Taylor expansion of order one allows to express the derivative of $I(\mathbf{r})$ with relation to time, referred to as $\dot{I}$:

$$\dot{I} = -\mathbf{L}_I \mathbf{v} \qquad \text{where} \qquad \mathbf{L}_I = \nabla I^{\top} \mathbf{L}_{\mathbf{x}}. \tag{5.18}$$

$\mathbf{L}_I(\mathbf{x})$ corresponds to the variation in luminance of a point $\mathbf{x}$ with relation to the motion of the camera. It is given by the following relation:

$$\mathbf{L}_I = \nabla I^{\top} \mathbf{L}_{\mathbf{x}} = (\nabla I_x \mathbf{L}_x + \nabla I_y \mathbf{L}_y) \tag{5.19}$$

where ∇I_x and ∇I_y are the two components of the gradient of I evaluated around the point $\mathbf{x}$ and $\mathbf{L}_x$ and $\mathbf{L}_y$ are the interaction matrices associated to the x and y coordinates of $\mathbf{x}$. As these equations are only true for each pixel on its own, considering the whole feature $\mathbf{I}(\mathbf{r})$ means aggregating the relations of all pixels into the same equation. The interaction matrix of the task is then given by:

$$\mathbf{L_I} = \begin{pmatrix} \mathbf{L}_{I(\mathbf{x_{00}})} \\ \vdots \\ \mathbf{L}_{I(\mathbf{x_{mn}})} \end{pmatrix}. \tag{5.20}$$

This approach validates the use of the luminance of the whole image luminance values in order to perform visual servoing. It was shown to possess some robustness towards small occlusions or specular spots caused by non-lambertian surfaces [Collewet et al., 2008]. But the main idea behind the method remains the conservation of luminance hypothesis leading to the use of the SSD as cost function (equation (5.17)). This limits the approach to scenes that do not undergo significant perturbations. Changes in light conditions or occluded

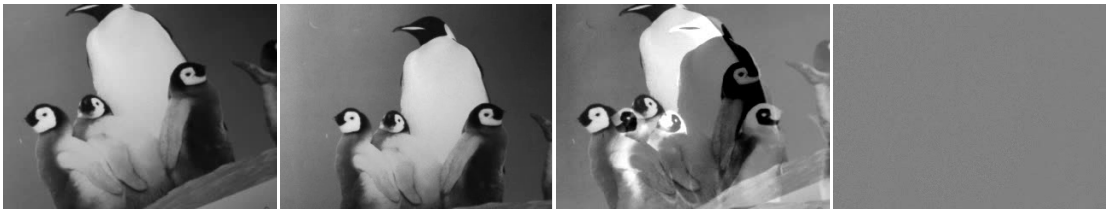


Figure 5.7: From left to right: Current image at starting and finishing positions, associated error images at starting and finishing positions.

parts of the reference are not taken into account when modeling the positioning task and performing visual servoing in outdoor scenes for example becomes very complicated with this approach. This is why the approach as been extended in [**?**] to take into account a more complete lighting representation. By better modeling the way light is reflected onto the camera by the object, the considered approach is then more robust, to the expense of complexity of the method.

5.3.2 Using the mutual information as a similarity criterion

Several works have built on the foundations layed in [Collewet et al., 2008] in order to extend the scope of dense visual servoing. In order not to have to assume the conservation of luminance expressed in equation (5.17), the works of [Dame and Marchand, 2009] have defined a visual servoing task that uses a more robust cost function. They propose to use the whole information present in the image, in the entropical sense [Shannon, 1948]. The chosen cost function is the mutual information that translates the quantity of information shared by two signals, in that case image signals. As was previously shown in section 2.3, this measure of similarity between two images is very robust to both local and global perturbations of the scene. The resulting task is then able to cope with global light variations, occlusions, specular spots or even different image modalities. This is very interesting as it opens up dense visual servoing to scenes where moving objects are present and could occlude the reference. It is also possible to imagine tasks where a reference can is taken at a time of the day and the servoing is launched at another time or tasks where a map reference and a current satellite image are used. Let us then expose how that approach can be performed. As it was shown in chapter 2, the mutual information is a difference of entropy between two images:

$$\mathrm{MI}(\overline{I}, \overline{I^*}) = \mathrm{H}(\overline{I}) + \mathrm{H}(\overline{I^*}) - \mathrm{H}(\overline{I}, \overline{I^*}). \tag{5.21}$$

For visual servoing purposes, the MI cost function is very similar to the 3D registration problem. The value of the MI with relation to the camera position is given by:

$$\mathrm{MI}(\mathbf{r}) = \sum_{r,t=0}^{N_d} p_{\overline{II^*}}(r, t, \mathbf{r}) \log \left(\frac{p_{\overline{II^*}}(r, t, \mathbf{r})}{p_{\overline{I}}(r, \mathbf{r}) p_{\overline{I^*}}(t)} \right) \tag{5.22}$$

This cost function is not directly usable for visual servoing by nullifying it as is the case for the SSD. This is due to the fact that when the robot is at the desired position, the MI is at its maximum between $\overline{I}(\mathbf{r})$ and $\overline{I^*}$. In order to create a visual servoing task, the MI must first be derived. That way the cost function will have its minimal value equal to zero when the positioning task is complete. The true measure that the servoing task will have to nullify is then the gradient of the MI. In order to express that control law, let us first

recall how a visual servoing task is defined. Using a proportional control law to model the problem, a velocity vector given to each degree of freedom of the robot is computed thanks to the following equation:

$$\mathbf{v} = -\lambda \hat{\mathbf{L}}_{\mathbf{e}}^{+} \mathbf{e} \tag{5.23}$$

where $\hat{\mathbf{L}}_{\mathbf{e}}^{+}$ is the pseudo-inverse matrix of the interaction matrix associated to the task $\mathbf{e}$. Here, the task is the minimization of the interaction matrix of $MI(\mathbf{r})$ noted $\mathbf{L}_{MI}^{\top}$ which is to be nullified at convergence, yielding:

$$\mathbf{v} = -\lambda \mathbf{H}_{MI}^{-1} \mathbf{L}_{MI}^{\top}. \tag{5.24}$$

where $\mathbf{H}_{MI}$ is the interaction matrix associated to $\mathbf{L}_{MI}$, also called Hessian matrix of the MI. In [Dame and Marchand, 2011], the method has been shown to be very robust to numerous situations, validating the use of the MI for visual servoing purposes. Several cases have been studied. The method was then shown to be able to handle positioning tasks in nominal conditions but also in situations involving global or local perturbations such as global or local light variations and occlusions. The proposed algorithm was also tested on scenes presenting different modalities. They have also presented the possibility to extend the method to simple navigation tasks using a visual path as reference and shown an example task where a trajectory is learned on a topological map (provided by the french "Institut Géographique National") and the navigation is realized by the robot using a satellite view of the same area.

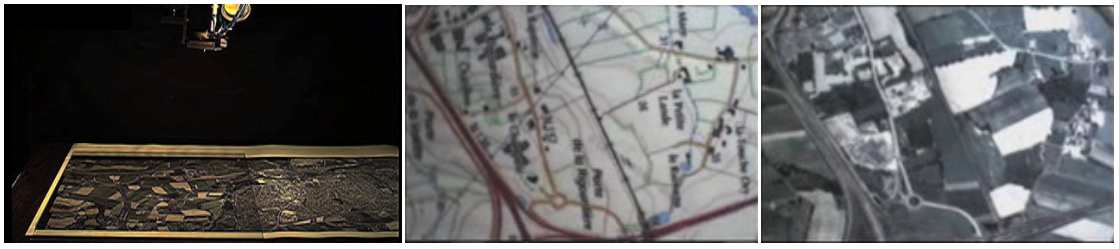


Figure 5.8: Visual navigation using the MI. From left to right: Servoing robot setup, current desired image along the visual path and current image perceived by the camera mounted on the robot.

5.4 Conclusion

In this chapter, the basis of visual servoing for positioning tasks are recalled. Ranging from geometrical features-based techniques to recent dense techniques, several approaches have been detailed. The objective of a visual tracking task is to estimate the parameters of a 2D transformation in order to align current and desired image features. Visual servoing also aims at aligning two sets of features by physically moving a camera. This results in a strong duality between both processes [Marchand and Chaumette, 2002]. The main difference is that the projection is modified through the motion of a robot. This is why in the next chapters the robust similarity functions that have been shown successful when performing visual tracking tasks in chapters 3 and 4 are adapted and extended in order to propose new visual servoing processes.

6

Visual servoing using the SCV

Our first contribution in the domain of visual servoing is the use of the SCV to perform direct visual servoing. As shown in chapters 3 and 4, this dissimilarity measure allows a natural robustness to global illumination variations all the while keeping a low computational cost. Adapting an image to the appearance of the second one, the measure has been shown to be successful when confronted to strong light variations. That robustness to global variations is very interesting in the scope of dense visual servoing, as it allows to perform servoing tasks outdoor for example where the ambient light is subject to fast changes while keeping a simple and computationally efficient algorithm. This is why in this chapter we propose to adapt the classical direct visual servoing approach presented in section 5.3 to this new criterion. The measure is then redefined to be used for servoing purposes and a simple control law is designed that allows to perform a positioning task robust to light variations. After being defined, the proposed approach is validated in nominal and perturbed conditions through several experimentations on a real six degree of freedom robot and the results are compared to the SSD-based photometric visual servoing.

6.1 Redesigning the SCV for use in visual servoing

6.1.1 Redefining the dissimilarity measure

As explained in chapter 1, the SCV is a dissimilarity measure, with a lot of similarity with the SSD. The main difference between those two measures is the fact that the SCV uses an estimator in order to predict what one of the two images should look like if it was perceived in the same conditions as the other, thanks to histogram manipulations. In order to be used within a visual servoing task, the SCV must be redefined with relation to the current camera pose. Two choices arise when defining the SCV measure that will be used. The first one is to use the strategy shown in [Richa et al., 2011] for tracking purposes and compare the current image to an adaptation of the reference image. The ensuing dissimilarity function is then:

$$SCV(\mathbf{r}) = \sum_{\mathbf{x}} (I(\mathbf{x}, \mathbf{r}) - \hat{I}(\mathbf{x})). \tag{6.1}$$

In order to compute the expected image $\hat{I}$, the marginal and joint probability density function of $I(\mathbf{r})$ and I^* are needed. Probability density functions are computed from the empirical analysis of I and I^*. As they both have the same dynamic d, which usually is 256 for usual grey level images, for each couple of grey levels i in $I(\mathbf{r})$ and j in I^* it is

possible to compute:

$$p_{II^*}(i,j) = P(I(\mathbf{x},\mathbf{r}) = i, I^*(\mathbf{x}) = j) \quad (6.2)$$

$$= \frac{1}{N}\sum_{\mathbf{x}} \psi(I(\mathbf{x},\mathbf{r}) - i)\psi(I^*(\mathbf{x}) - j) \quad (6.3)$$

where $\psi(a)$ equals 1 if a=0, 0 otherwise and p_{II^*} represents the probability that a pixel $\mathbf{x}$ takes the value i in $I(\mathbf{r})$ and j in I^*. From this joint probability distribution, the probability distributions of I^* can easily be found. For example the probability distribution for the apparition of a grey level j in I^* is given by:

$$p_{I^*}(j) = \sum_i p_{II^*}(i,j). \quad (6.4)$$

From these relations, the computation of the expected grey levels in $\hat{I}$ becomes, for each grey level j in I^*:

$$\hat{I}(j) = \sum_i i \ \frac{p_{II^*}(i,j)}{p_{I^*}(j)}. \quad (6.5)$$

The drawback of that first possible strategy lies in the computation of the interaction matrix associated to the servoing task $SCV(\mathbf{r})$. Indeed, the classical interaction matrix computation as expressed in equation (5.19) is made efficient by the fact that the desired image I^* is constant throughout the servoing process. But with this formulation of the SCV, it is not true anymore. The reference image is adapted at each frame, which means that the interaction matrix of the task should in theory integrate that factor in its computation. In [Delabarre and Marchand, 2012], we proposed to approach the interaction matrix by taking into account only the reference image in the interaction matrix expression. In this document another strategy will be used. Inspired by the contributions proposed in chapters 3 and 4, a new SCV servoing task can be proposed:

$$SCV(\mathbf{r}) = \sum_{\mathbf{x}} (\hat{I}(\mathbf{x},\mathbf{r}) - I^*(\mathbf{x})). \quad (6.6)$$

In that case, the expected image $\hat{I}$ is computed from the joint probability distribution between $I(\mathbf{r})$ and I^* and now represents what the current image should look like if it was perceived in the same conditions as the reference image. The histogram adaptation is then realized on every grey level of the current image I:

$$\hat{I}(i) = \sum_j j \ \frac{p_{II^*}(i,j)}{p_I(i)}. \quad (6.7)$$

That formulation then allows to keep a desired image I^* constant through the process, while the new expected image $\hat{I}(\mathbf{r})$ is now completely linked to the position of the camera.

6.1.2 SCV-based control law

The advantage of this new formulation is that the interaction matrix does not need to be approached in this case. As the adaptation is made on the current image, the reference is constant during the servoing process and its derivative is then actually null. The SCV error vector is therefore given by:

$$\mathbf{e}(\mathbf{r}) = (\widehat{\mathbf{I}}(\mathbf{r}) - \mathbf{I}^*)^\top (\widehat{\mathbf{I}}(\mathbf{r}) - \mathbf{I}^*). \quad (6.8)$$

A control law can then be defined to perform the SCV visual servoing task:

$$\frac{\partial \widehat{\mathbf{I}}(\mathbf{r})}{\partial t} = \mathbf{L}_{\hat{I}}\ \mathbf{v} \qquad \text{where} \qquad \mathbf{L}_{\hat{I}} = -\nabla \hat{I}^{\top} \mathbf{L}_{\mathbf{x}}. \tag{6.9}$$

$\mathbf{L}_{\hat{I}}(\mathbf{x})$ is the interaction matrix of the adapted image, which is given by the following relation:

$$\mathbf{L}_{\hat{I}} = -(\nabla \hat{I}_x \mathbf{L}_x + \nabla \hat{I}_y \mathbf{L}_y) \tag{6.10}$$

where $\nabla \hat{I}_x$ and $\nabla \hat{I}_y$ are the two components of the gradient of $\hat{I}$ evaluated around the point $\mathbf{x}$ and $\mathbf{L}_x$ and $\mathbf{L}_y$ are the interaction matrices associated to the x and y coordinates of $\mathbf{x}$. Here again, the equations are only true for each pixel on its own. Considering the whole image $\hat{I}(\mathbf{r})$ of size $m \times n$ then means regrouping the equation of every pixel in the image:

$$\mathbf{L}_{\hat{\mathbf{I}}} = \begin{pmatrix} \mathbf{L}_{\hat{I}(\mathbf{x}_{00})} \\ \vdots \\ \mathbf{L}_{\hat{I}(\mathbf{x}_{mn})} \end{pmatrix}. \tag{6.11}$$

The robot is finally controlled thanks to the velocity vector $\mathbf{v}$ computed as:

$$\mathbf{v} = -\lambda \mathbf{L}_{\hat{\mathbf{I}}} \left(\widehat{\mathbf{I}}(\mathbf{r}) - \mathbf{I}^{*} \right). \tag{6.12}$$

This equation can also be written considering a Levenberg-Marquardt optimization:

$$\mathbf{v} = -\lambda (\mathbf{H}_{\hat{\mathbf{I}}} + \mu\ \mathrm{diag}(\mathbf{H}_{\hat{\mathbf{I}}}))^{-1} \mathbf{L}_{\hat{\mathbf{I}}} (\hat{\mathbf{I}} - \mathbf{I}^{*}) \tag{6.13}$$

where $\mathbf{H}_{\hat{\mathbf{I}}} = \mathbf{L}_{\hat{\mathbf{I}}}^{\top} \mathbf{L}_{\hat{\mathbf{I}}}$ is an approximation of the Hessian of the SCV and μ is a tuning factor.

6.1.3 Histogram Binning

In order to smooth the similarity function to make the SCV more fit for optimization, histogram binning is used when performing direct visual servoing with the SCV. More details on how histogram binning is realized are available of Frame 4. The histograms are then computed on a scaled image which new dynamic is Nc, ensuring that they are composed of only Nc bins. The main issue when using approached probabilities is that if the two images are not similar enough, noise appears in the predicted image when using an important number of bins (see figure 6.1 for more details). On the other hand, if the chosen number of bins is too low, the resulting image loses a lot of high frequency details which results in a loss of accuracy for the positioning task. This is why we propose to adapt dynamically the number of bins during the servoing task. At first, the binning is done using 64 bins, which is a good trade-off as it allows to smooth the SCV shape while retaining sufficient level of detail in the image. Once the value of SCV is considered to have decreased significantly, indicating that the positioning task is close from completion, the number of bins increases to 256 to be more accurate upon convergence.

6.2 Experimental validations

In order to validate the proposed SCV-based visual servoing approach, several positioning tasks have been realized on a six degrees of freedom robot on which a camera is mounted in order to have a Eye-in-Hand servoing system. For all experiments, the six degrees of freedom of the robot are controlled by the servoing process. The control law used

Figure 6.1: Influence of the number of bins on the predicted image. On the left, with 8 bins the image is flattened and has lost a lot of high frequencies. 256 bins introduces more noise when the difference between I and I^* increases than 64. The frame on the borders represent the area in the image which is not included in the computation.

is the Levenberg-Marquardt approach proposed in equation (6.13). The image gathered from the camera is a 320 $\times$ 240 image which allows the task to consider the whole image (exception made of the outer borders of the image to avoid out of bounds evaluations) while keeping a good computational efficiency. The different experiments realized showed that the computation time is lower than the 60ms necessary to acquire a new frame, thus resulting in a real-time application running on a single thread of a core i7 2.8 GHz processor. The servoing task results will be analyzed by looking at several factors. The most important results are the translational and rotational positioning errors. As the goal is to get the robot to a precise position in space, those errors will allow us to assess the accuracy of the task. Other factors will include the evolution of the considered cost function along the task and the 3D trajectory of the robot. In order to understand the conditions of each experiment, several current pictures $I(\mathbf{r})$ perceived by the camera during the process will be shown along with corresponding expected images $\hat{I}(\mathbf{r})$. Finally, an interpretation of the image error $I(\mathbf{r}) - I^*$ will also be provided in order to see the effects of both camera motion and scene perturbations.

6.2.1 Visual servoing in nominal conditions

6.2.1.1 Positioning task

The objective of this first experiment is to validate the proposed SCV-based visual servoing approach in nominal conditions, featuring no scene perturbations. At the beginning of the experiment, the desired image I^* is recorded from the desired position $\mathbf{r}^*$ and the robot is moved to a starting position. The difference in position between desired and current along

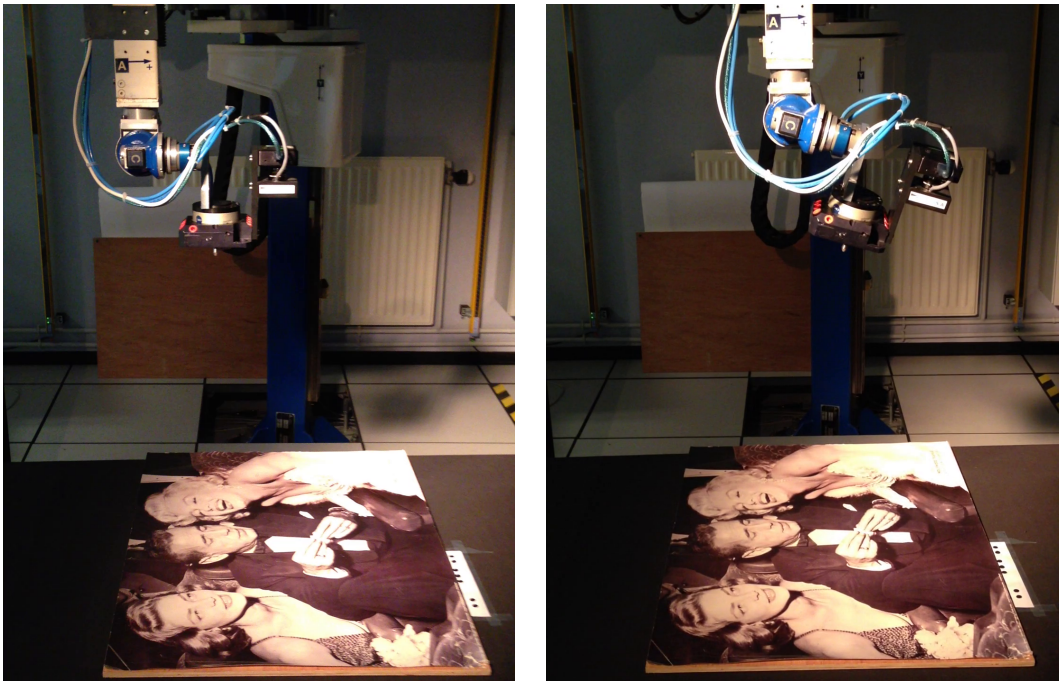


Figure 6.2: Six degree of freedom robot used for the experiments. We can see on the left the desired position and on the right the starting position for our experiments.

Figure 6.3: From left to right: image seen at desired and start position and corresponding image error. Pixel values on the image error are shifted by an offset of 128 grey levels in order to increase readability.

the x, y and z axes of the world frame are, respectively, $tx = 0.134\,m$, $ty = -0.174\,m$ and $tz = 0.073\,m$. The corresponding rotational errors are $rx = -0.28\,rad$, $ry = -0.18\,rad$ and $rz = -0.02\,rad$. Information relative to this experiment is shown on figure 6.4. Several observations can be made concerning this first experiment. First, the task is successful, leading the robot to the good position with a negligible residual distance to the goal of $\mathbf{\Delta r}^* = \{0.05mm, 0.01mm, 0.01mm, 1.2e^{-4}rad, 4.4e^{-7}rad, 1.9e^{-3}rad\}$. The second observation is that the evolution of the camera location is very smooth, not presenting any violent changes in direction. The SCV also vanishes without any oscillations, validating the use of the SCV for visual servoing in nominal conditions.

6.2.1.2 Comparison with photometric visual servoing

As the proposed SCV-based approach aims at enhancing the possibilities of the SSD-based method without adding the complexity of the MI-based servoing, a comparison was made

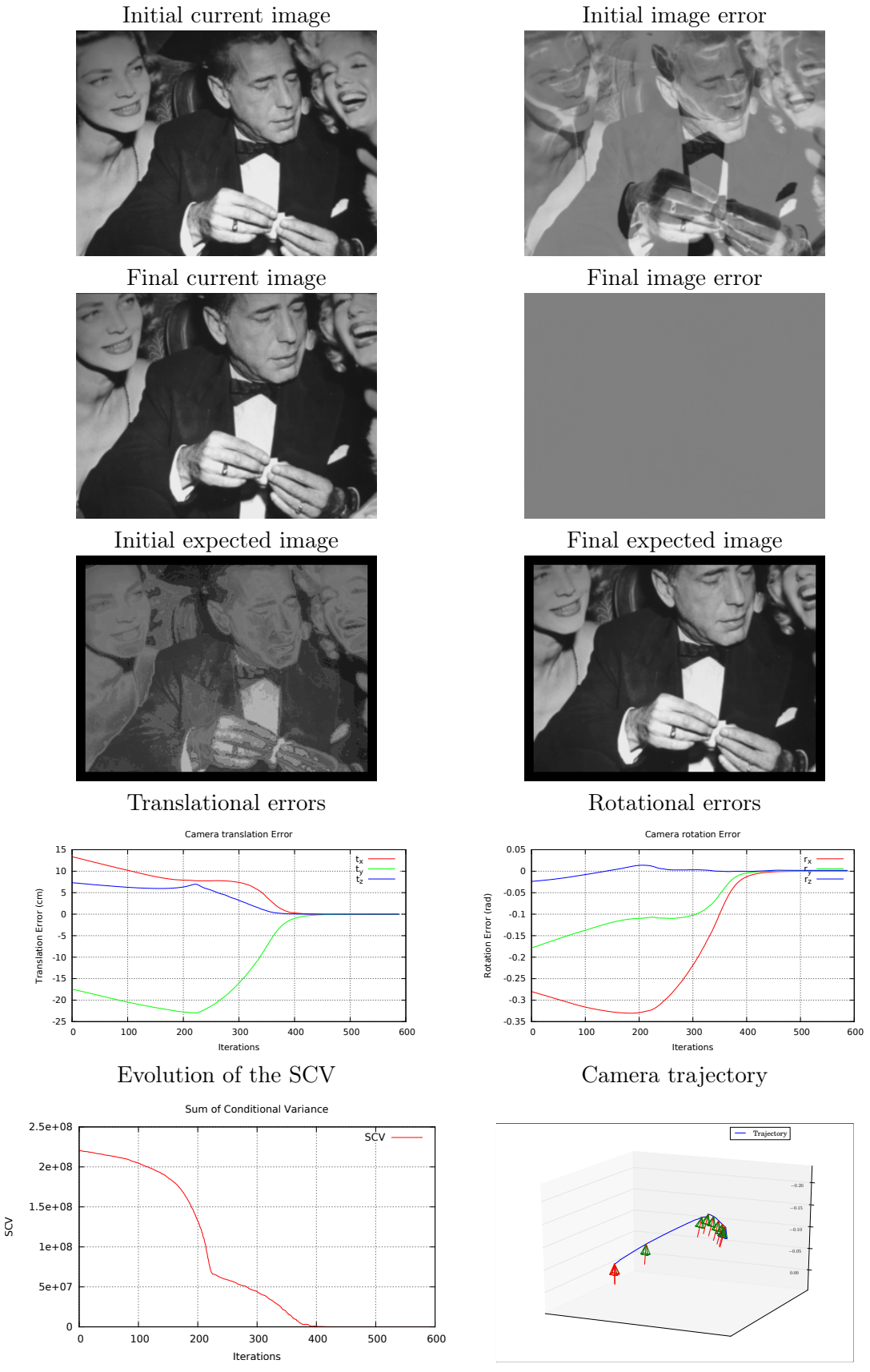


Figure 6.4: Results of SCV-based visual servoing in nominal conditions.

between our algorithm and the SSD visual servoing approach. The same task was then realized with that second algorithm. Results are shown on figure 6.5. It is clear from the translation and rotation error graphs that the convergence of the two methods give similar results in these conditions, as the evolution in both cases follow the same path which is confirmed by the trajectories displayed on the same figure. Let us add that the graphs shows a change of behaviour (around iteration 210). That can be explained by the fact that the observed texture presents big uniform grey level areas. The error then decreases very fast as it aims to align those before the final alignment over the higher frequencies, which is slower as the error created by them is lower.

6.2.2 Visual servoing confronted to light variations

6.2.2.1 Positioning task

The objective of this second experiment is to validate the proposed SCV-based visual servoing approach in perturbed conditions. This is why during the servoing process the light of the room is switched on and off several times during the experiment. The experiment is then initialized and launched in nominal conditions and global perturbations of the scene happen during the positioning task. Information relative to this experiment is shown on figures 6.6 and 6.7. This experiment is very interesting due to the fact that it clearly shows the possibilities of light adaptation of the expected image computation. Even though the light is turned on and off several times, as shown on the SCV evolution graph, the cost function adapts very quickly and is able to perform the servoing with a very good accuracy. The final residual positioning error is $\Delta\mathbf{r}^* = \{0.06mm, 0.02mm, 0.04mm, 2.8e^{-4}rad, 2.7e^{-4}rad, 9.3e^{-5}rad\}$. Let us note that the at the end of the task, the lights are in the same conditions as the start conditions. It can be noted that due to the low lighting, grain appear in the image causing a final position stabilizes at a mean distance of $2mm$ to $4mm$ from reach when the lights are totally out which is still a very good results given the perturbed conditions. It is also interesting to notice that the trajectory of the camera in perturbed conditions is very close to its trajectory in nominal conditions, validating the fact that the proposed algorithm is not impacted by these conditions.

6.2.2.2 Comparison with photometric visual servoing

In order to assess the differences between our algorithm and a classical SSD-based direct visual servoing approach, the same experiment was realized using that method. The visual servoing process was launched again and lights were turned off as before for the SCV. Results are exposed on figure 6.8. They show that in these conditions the SSD was not able to perform the task and drifted up to the point that it reached joint limits of the robot and had to be aborted. This comes to validate the proposed algorithm as it is still successful in conditions where the classical SSD-based approach failed.

6.2.3 Visual servoing confronted to occlusions

6.2.3.1 Positioning task

The objective of this final experiment is to validate the proposed SCV-based visual servoing when confronted to another type of perturbations. When moving a robot with relation to an object, one of the major problems that can be encountered is the loss of sight of a part of the object which can result from a loss of accuracy to a complete divergence of the

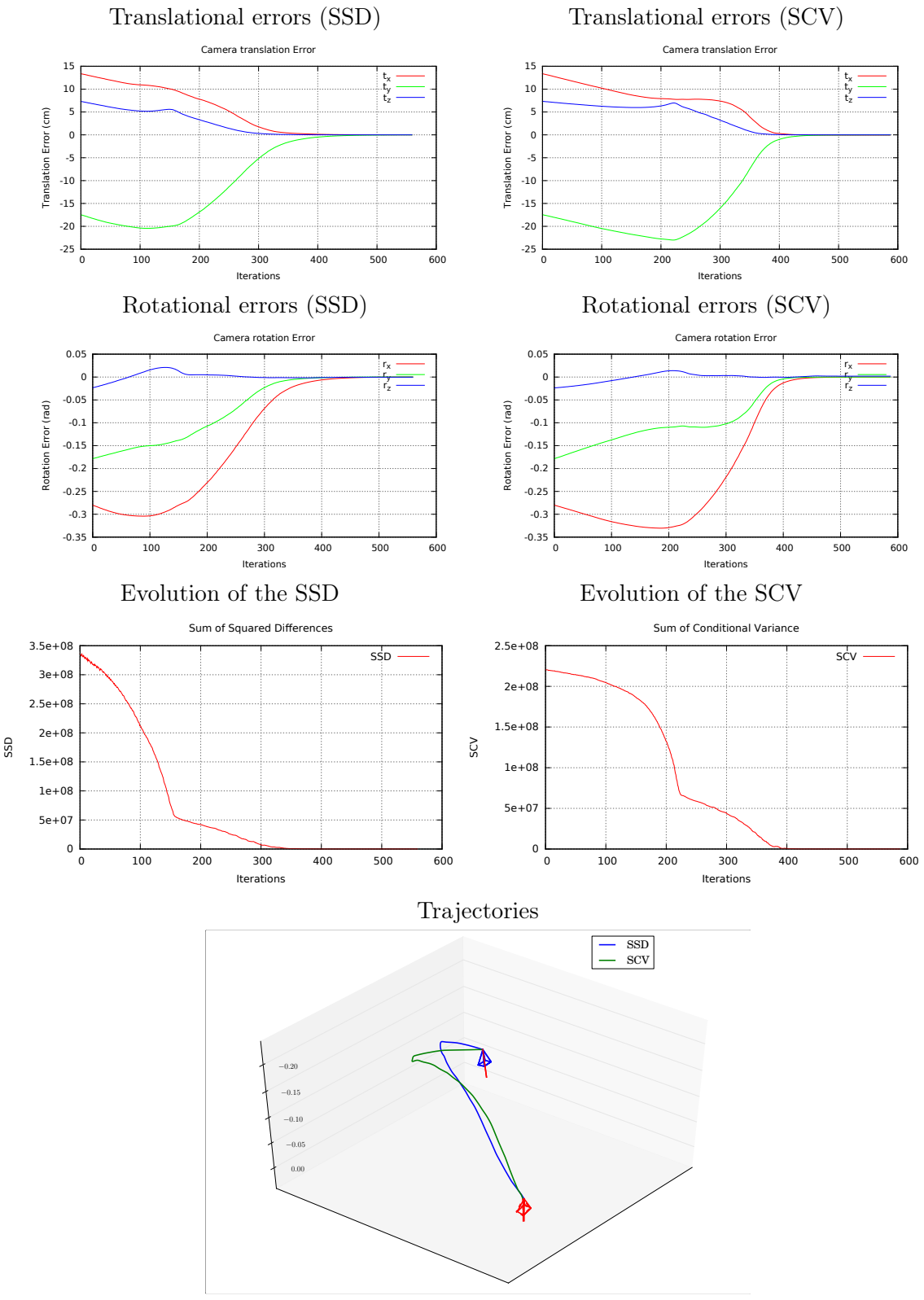


Figure 6.5: Comparison of SCV and SSD-based visual servoing methods in nominal conditions.

optimization process. This is why in this experiment an occlusion was added to the scene by adding and moving an object on the texture that is used to control the robot. We then added this type of perturbations to light variations, as section 6.2.2 demonstrated that light did not affect the SCV-based visual servoing. Information relative to this experiment is shown on figures 6.9 and 6.10. The results of this experiment allow to see the possibilities of the SCV visual servoing when confronted to local variations. Just as the SSD, the SCV is not designed to be robust to that type of perturbations, but still manages to give very good results in these conditions. The final residual positioning error is $\Delta\mathbf{r}^* = \{0.02mm, 0.08mm, 0.005mm, 1.6e^{-5}rad, 4.8e^{-4}rad, 2.2e^{-4}rad\}$. Let us note that at the end of the task, the perturbation is gone. Before the removal of the occlusion, the camera stabilizes with an accuracy of $3mm$ which is still a good result, since the occlusion really impacts the image due to its color and shadow. As soon as it is removed however, the task converges with a very good accuracy. Here again, the trajectory of the camera in these conditions is very close to the trajectory of the camera in nominal conditions, validating the fact that the proposed algorithm is robust to these conditions.

6.2.3.2 Comparison with photometric visual servoing

Again, a comparison was done in order to evaluate the differences between the proposed SCV approach and the classical SSD-based visual servoing. A similar experiment was therefore launched using the SSD-based algorithm and an object was moved upon the texture of the scene. This time however no light variations were added as the previous experiments showed the method was not able to cope with that situation. Results are exposed on figure 6.11. They show that in these conditions the SSD was able to perform the task but two observations can be made that show the proposed SCV-based approach to be more adapted to these situations. First, the residual error shown on the graph is greater with the SSD than with the SCV before the occlusion is removed, which means that our approach is more accurate. The other important factor is the trajectory of the camera, which is much more impacted for the SSD. All these observations then allow to validate the proposed SCV-based visual servoing algorithm as more efficient than its SSD-based counterpart in similar conditions.

6.3 Conclusion

In this chapter, a new way to perform dense photometric visual servoing has been proposed. Taking advantage of the adaptation properties of the SCV, the approach is robust to global light variations of the scene while being as simple to introduce and optimize as the classical SSD technique. Adapting the current view instead of the reference image also allowed the method to be computationally efficient. Several validations were realized on a real six degree of freedom robot that demonstrated the accuracy and robustness of the proposed approach.

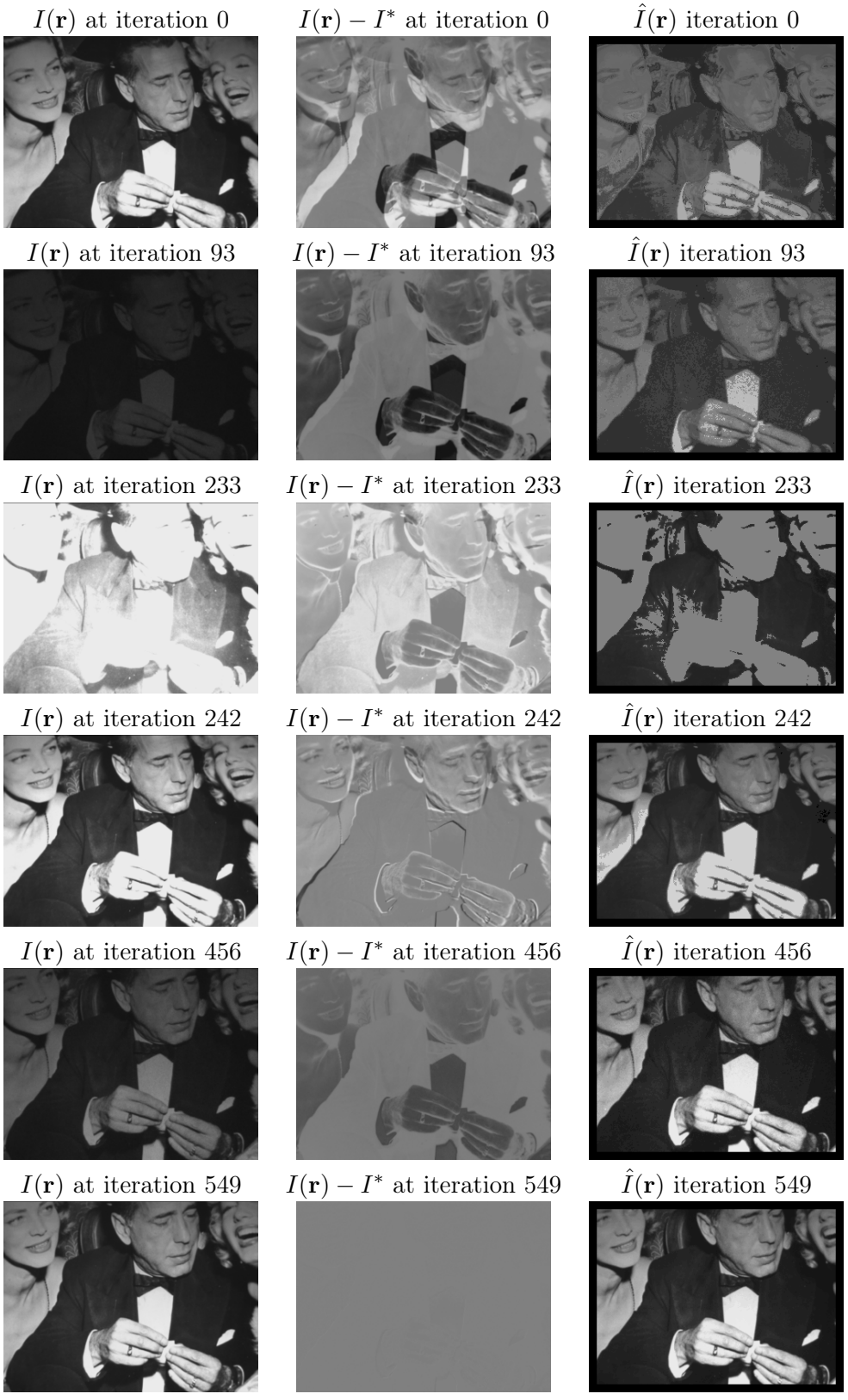


Figure 6.6: Results of SCV-based visual servoing in light varying conditions.

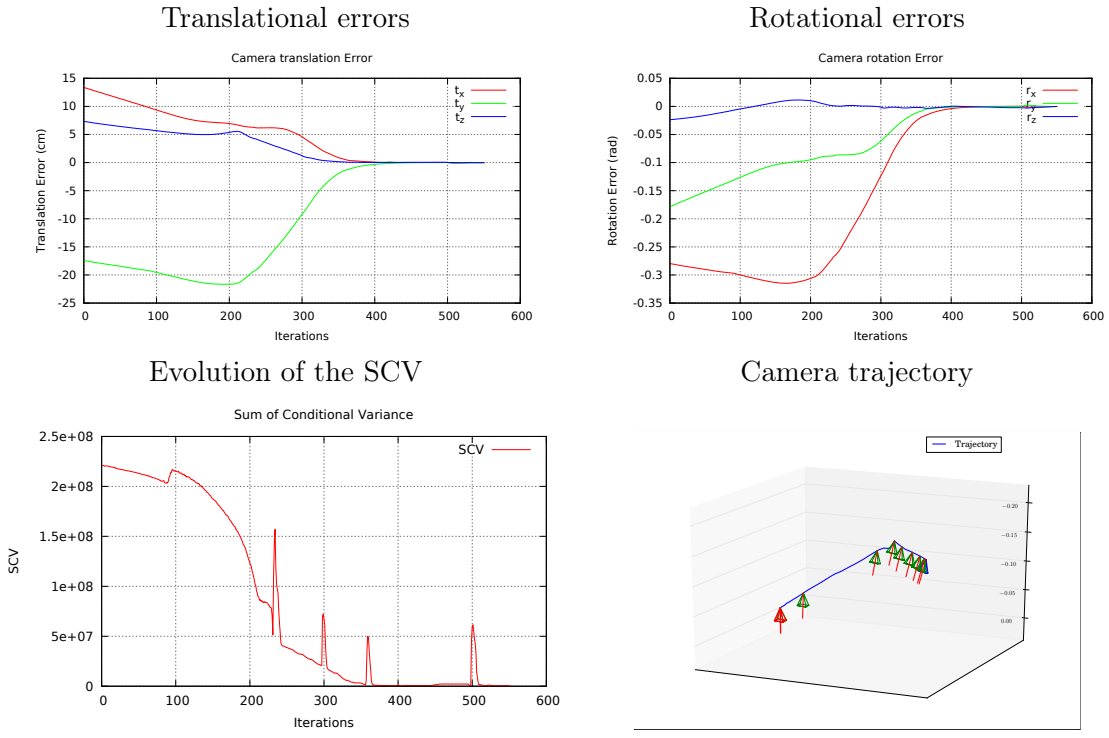


Figure 6.7: Results of SCV-based visual servoing in light varying conditions.

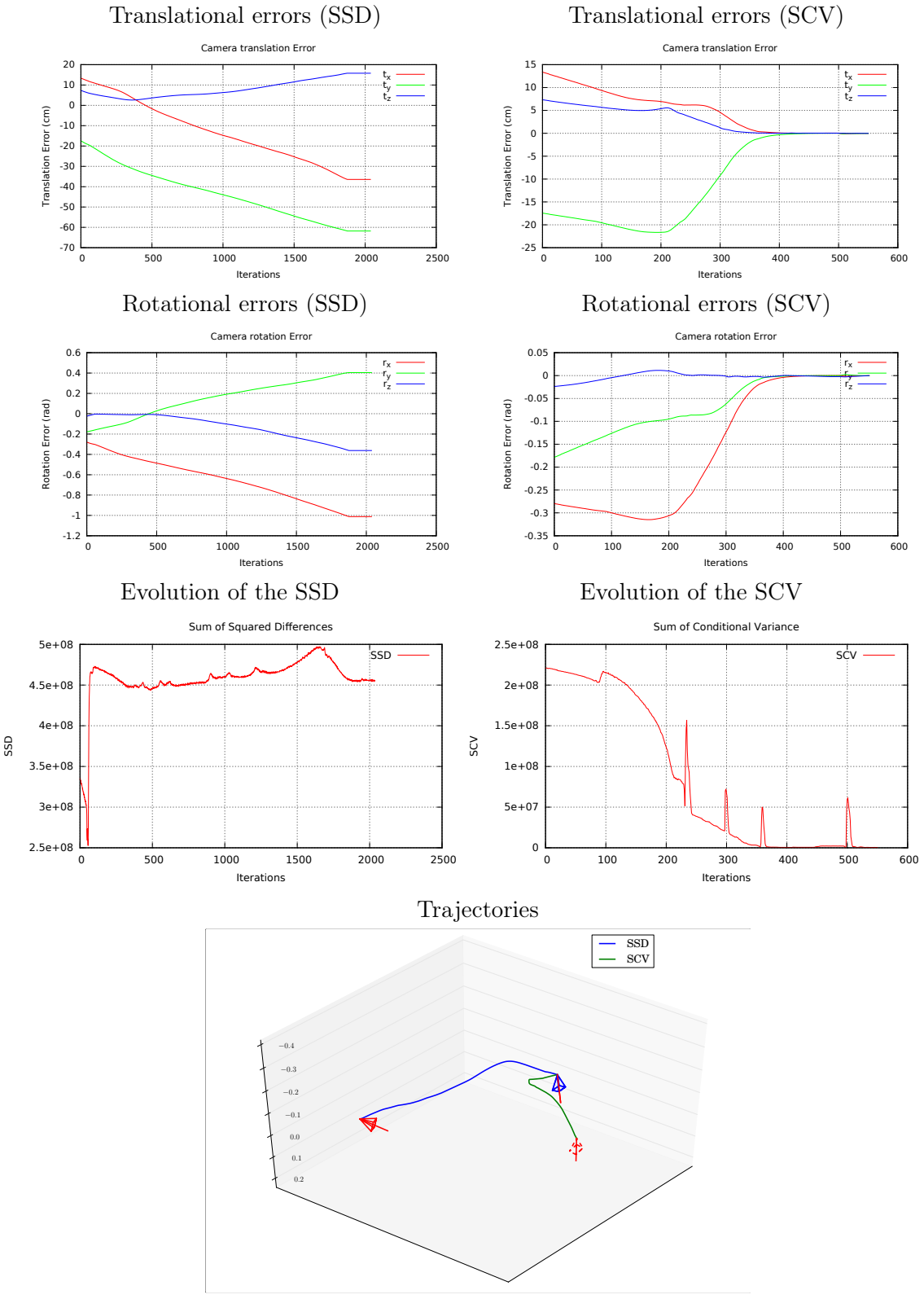


Figure 6.8: Comparison of SCV and SSD-based visual servoing methods in light varying conditions.

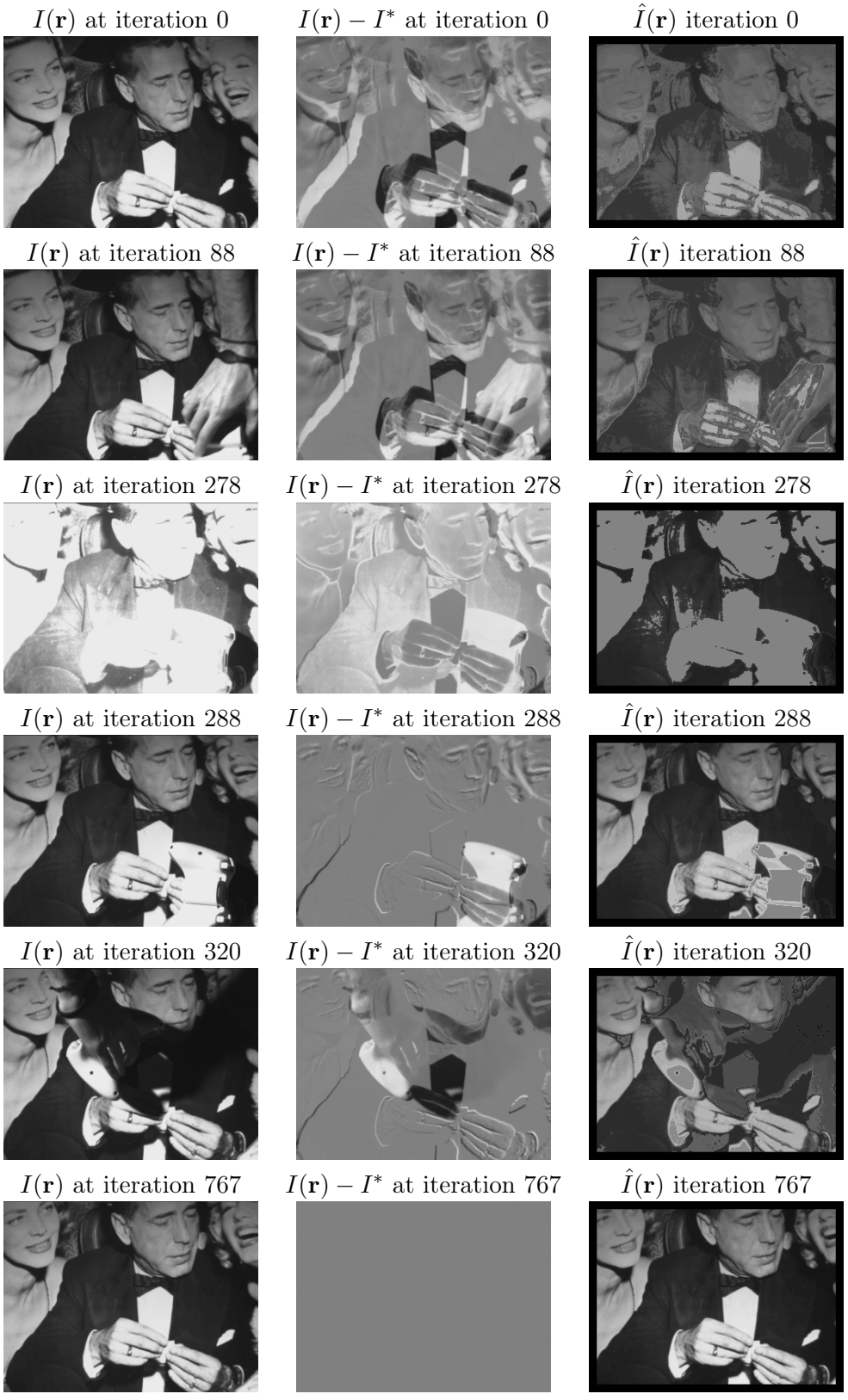


Figure 6.9: Results of SCV-based visual servoing when confronted to occlusions.

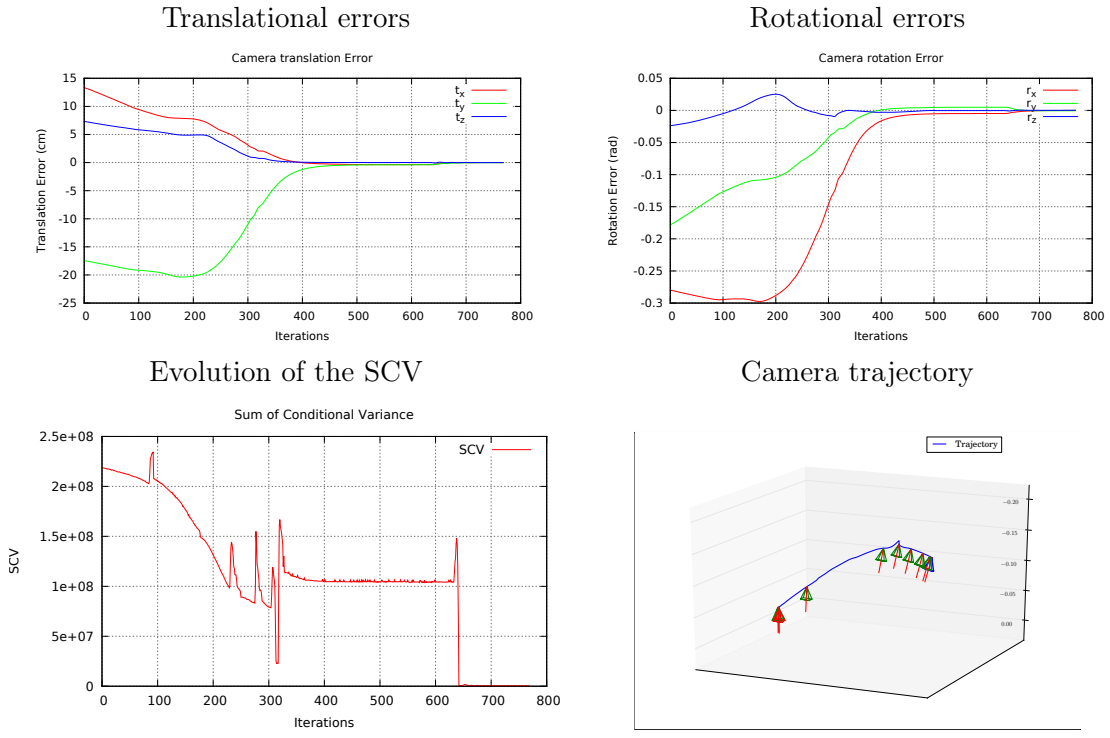


Figure 6.10: Results of SCV-based visual servoing when confronted to occlusions.

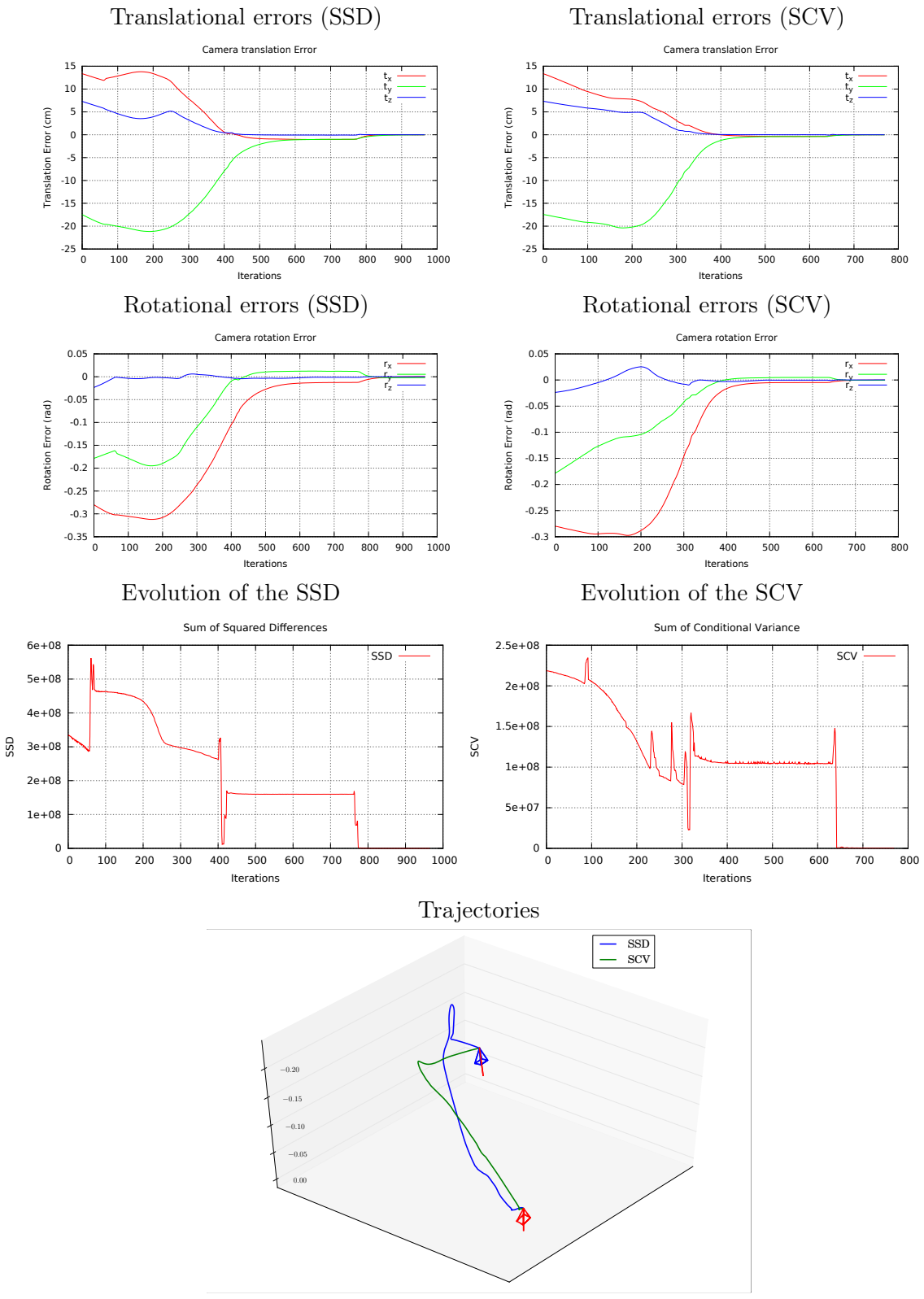


Figure 6.11: Comparison of SCV and SSD-based visual servoing methods in the presence of occlusions.

7

Visual servoing using a normalized mutual information

Our second contribution in the domain of direct visual servoing is a new way of using the entropy of the images in order to perform visual servoing. The MI is a similarity function, but its evaluation is flawed because of the fact that is has no concrete bound. Indeed, since it based on a difference of entropies its maximum value is directly linked to the entropy value of the considered images. Since that measure is different from one case to another, no comparison is possible between the results of several tasks. Moreover, it is not possible to know whether two images are visually close or not since the upper bound of the measure is different for each case and very rarely known. This is why in this chapter we first propose a visual servoing task based on a new formulation of the mutual information proposed in [Studholme and Hawkes, 1999]. This new similarity measure is a normalized version of the mutual information (NMI). After being detailed and validated through experimental tests, the proposed approach is compared with the state of the art MI approach of [Dame and Marchand, 2011]. Finally, the NMI is also adapted to fish-eye cameras through the use of the central projection model in order to create a robust omnidirectional dense visual servoing task.

7.1 Normalized mutual information-based visual servoing

7.1.1 Normalized mutual information

In order to achieve an even greater robustness without any added robustness step, we propose to use a formulation of the mutual information that is different from the one proposed by Shannon [Shannon, 1948] and introduced for visual servoing by [Dame and Marchand, 2011]. The considered similarity measure, called normalized mutual information (NMI), was proposed in [Studholme and Hawkes, 1999]. Instead of a difference of entropies, it is defined by a ratio of entropies, as shown by the expression of the measure:

$$NMI(I, I^*) = \frac{H(I) + H(I^*)}{H(I, I^*)}. \tag{7.1}$$

This formulation of the mutual information has also been shown in [Studholme and Hawkes, 1999] to be more robust to overlapping situations than the MI, in addition of having a maximum value equal to two for any pair of images. That upper bound is important, which is why details on its value are given here. By definition:

$$\begin{aligned} H(I, I^*) &> 0 \\ &\geq \max\left(H(I),\ H(I^*)\right) \end{aligned} \tag{7.2}$$

yielding:

$$\frac{1}{H(I,I^*)} \leq \frac{1}{\max\left(H(I),\ H(I^*)\right)}. \tag{7.3}$$

When $I = I^*$, they have the very same probability density function. Integrating the NMI formulation in the equation above by multiplying by $H(I) + H(I^*)$ then leads to:

$$\frac{H(I) + H(I^*)}{H(I,I^*)} \leq \frac{H(I) + H(I^*)}{\max\left(H(I),\ H(I^*)\right)}. \tag{7.4}$$

The probability density functions of I and I^* being the same, their entropy is also the same leading to: $\max\left(H(I),\ H(I^*)\right) = H(I) = H(I^*)$. Dividing both sides of the equation by that value of entropy then allows to express the upper bound of the NMI:

$$\frac{H(I) + H(I^*)}{H(I,I^*)} \leq 2. \tag{7.5}$$

7.1.2 NMI-based control law

This new formulation of the mutual information leads to a new definition of the Gradient and Hessian matrices $\mathbf{G}_{NMI}$ and $\mathbf{H}_{NMI}$ of the NMI. Starting from the definition given in equation (7.1), the numerator and denominator of the NMI, that will be referred to as u and v for readability purposes can be written as:

$$\begin{aligned}
u(\mathbf{r}) &= H(I(\mathbf{r})) + H(I^*) && (7.6)\\
&= -\sum_i p_I(i)\log\left(p_I(i)\right) - \sum_j p_{I^*}(j)\log\left(p_{I^*}(j)\right) && (7.7)\\
&= \sum_{i,j} -p_{II^*}(i,j)\log\left(p_I(i)\right) - p_{II^*}(i,j)\log\left(p_{I^*}(j)\right) && (7.8)\\
&= -\sum_{i,j} p_{II^*}(i,j)\log\left(p_I(i)p_{I^*}(j)\right) && (7.9)\\
v(\mathbf{r}) &= H(I(\mathbf{r}), I^*) && (7.10)\\
&= \sum_{i,j} -p_{II^*}(i,j)\log\left(p_{II^*}(i,j)\right) && (7.11)
\end{aligned}$$

In order to get the expression of $\mathbf{G}_{NMI}$ and $\mathbf{H}_{NMI}$, the first and second order derivatives of u and v are needed. The reference image I^* being constant throughout the servoing process, its probability density function is constant, leading to:

$$\begin{aligned}
\frac{\partial u(\mathbf{r})}{\partial \mathbf{r}} &= -\sum_{i,j} \frac{\partial p_{II^*}(i,j)}{\partial \mathbf{r}} \log(p_I(i)p_{I^*}(j)) && (7.12)\\
\frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} &= -\sum_{i,j} \frac{\partial p_{II^*}(i,j)}{\partial \mathbf{r}} (1 + \log(p_{II^*}(i,j))) && (7.13)\\
\frac{\partial^2 u(\mathbf{r})}{\partial \mathbf{r}^2} &= -\sum_{i,j} \frac{\partial^2 p_{II^*}(i,j)}{\partial \mathbf{r}^2} \log(p_I(i)p_{I^*}(j)) + \frac{1}{p_{II^*}(i,j)} \frac{\partial p_{II^*}(i,j)}{\partial \mathbf{r}}^2 && (7.14)\\
\frac{\partial^2 v(\mathbf{r})}{\partial \mathbf{r}^2} &= -\sum_{i,j} \frac{\partial^2 p_{II^*}(i,j)}{\partial \mathbf{r}^2} (1 + \log(p_{II^*}(i,j))) + \frac{1}{p_{II^*}(i,j)} \frac{\partial p_{II^*}(i,j)}{\partial \mathbf{r}}^2. && (7.15)
\end{aligned}$$

$\mathbf{G}_{NMI}$ is then given by the following equation:

$$\mathbf{G}_{MI}(\mathbf{r}) = \frac{\frac{\partial u(\mathbf{r})}{\partial \mathbf{r}}.v(\mathbf{r}) - \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}}.u(\mathbf{r})}{v(\mathbf{r})^2}. \tag{7.16}$$

Finally, in order to simplify the readability of the expression of $\mathbf{H}_{NMI}$, the nominator and denominator of $\mathbf{G}_{NMI}$ are respectfully referred to as $\alpha(\mathbf{r})$ and $\kappa(\mathbf{r})$:

$$\alpha(\mathbf{r}) = \frac{\partial u(\mathbf{r})}{\partial \mathbf{r}} v(\mathbf{r}) - u(\mathbf{r}) \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} \tag{7.17}$$

$$\frac{\partial \alpha(\mathbf{r})}{\partial \mathbf{r}} = \frac{\partial^2 u(\mathbf{r})}{\partial \mathbf{r}^2} . v(\mathbf{r}) - u(\mathbf{r}) \frac{\partial^2 v(\mathbf{r})}{\partial \mathbf{r}^2} \tag{7.18}$$

$$\kappa(\mathbf{r}) = v(\mathbf{r})^2 \tag{7.19}$$

$$\frac{\partial \kappa(\mathbf{r})}{\partial \mathbf{r}} = 2 \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} v(\mathbf{r}) \tag{7.20}$$

leading to the computation of $\mathbf{H}_{NMI}$:

$$\mathbf{H}_{MI}(\mathbf{r}) = \frac{\frac{\partial \alpha(\mathbf{r})}{\partial \mathbf{r}} . \kappa(\mathbf{r}) - \alpha(\mathbf{r}) . \frac{\partial \kappa(\mathbf{r})}{\partial \mathbf{r}}}{\kappa(\mathbf{r})^2} . \tag{7.21}$$

Once that those two matrices have been defined, the servoing task follows the same principle as what was done by [Dame and Marchand, 2011]. At each new frame, a velocity vector is computed as:

$$\mathbf{v} = -\lambda \mathbf{H}_{NMI}^{-1} \mathbf{G}_{NMI}^{\top} . \tag{7.22}$$

7.2 Experimental validations

In order to validate the proposed NMI-based visual servoing approach, several positioning tasks have been realized using the visual servoing setup exposed in section 6.2. The NMI-based control law designed in the previous section is then used to control the six degrees of freedom of the robot and results are recorded. Here again, the whole image is considered to build the control law and the process runs at video-rate on the same single thread of a core i7 2.8 GHz processor, since the task is computed faster than the 60ms necessary to acquire a new frame. The visual servoing task results will be analyzed by looking at the same factors as in the previous chapter in order to keep a coherency between the results. The most important factors considered will then be the translational and rotational positioning errors. Since the goal is to get the robot to a precise position in space, those errors will allow us to assess the accuracy of the proposed method. Other factors include the evolution of the considered cost function along the task and the 3D trajectory of the robot. In order to understand the conditions of each experiment, several current pictures $I(\mathbf{r})$ perceived by the camera during the process will be shown and an interpretation of the image error $I(\mathbf{r}) - I^*$ will also be provided in order to see the effects of both camera motion and scene perturbations (although this error is never considered in the control law).

7.2.1 Visual servoing in nominal conditions

The objective of this first experiment is to validate the proposed NMI-based visual servoing approach in nominal conditions, featuring no scene perturbations. At the beginning of the experiment, the desired image I^* is recorded from the desired position $\mathbf{r}^*$ and the robot is moved to a starting position. The difference in position between desired and current along the x, y and z axes of the world frame are, respectively, $tx = 0.134\ m$, $ty = -0.174\ m$ and $tz = 0.073\ m$. The corresponding rotational errors are $rx = -0.28\ rad$, $ry = -0.18\ rad$ and $rz = -0.02\ rad$. Information relative to this experiment is shown on figure 7.1. The first observation that can be made related to this first experiment is that the positioning task

is successful, leading the robot to the good position with a negligible residual distance to the goal of $\Delta\mathbf{r}^* = \{0.09mm, 0.01mm, 0.01mm, 2.8e^{-4}rad, 1.2e^{-6}rad, 2.1e^{-3}rad\}$. Another interesting point is the fact that the trajectory of the camera is good, not deviating of shaking along the task. It is important to note that an adaptive gain is used here in order to keep a good velocity upon convergence, which explains the shape of the NMI evolution. Finally, the value of the NMI upon convergence can be seen not to be equal to its maximum value of 2. Although this could be problematic in order to have a good understanding of the similarity between the two images, it is important to notice the facts that this is caused by the use of B-spline binning and that the criteria is still very useful even in these conditions, since it is possible to define similarity thresholds which is not possible with the classical formulation of the MI. All those results then come to validate the use of the NMI for visual servoing in nominal conditions.

7.2.1.1 Comparison with classical MI-based visual servoing

As the proposed NMI-based approach aims at enhancing the possibilities given by the classical MI-based servoing algorithm, a comparison was made between our proposed algorithm and the MI-based visual servoing approach proposed by [Dame and Marchand, 2011]. The same task was then realized with that second algorithm. Results are shown on figure 7.2. It is clear from the translation and rotation error graphs that the convergence accuracy of the two methods give similar results in these conditions, as the evolution in both cases follow the same path which is confirmed by the trajectories displayed on the same figure. The proposed NMI-based visual servoing approach then allows to obtain the same results as the MI-based method, albeit the fact that the proposed algorithm can provide with an understanding of the similarity between the current and desired images since even if the theoretical maximum value is altered by the histogram binning, it is still possible to evaluate a similarity based on the NMI value.

7.2.2 Visual servoing in perturbed conditions: light variations

The objective of this second experiment is to validate the proposed NMI-based visual servoing approach when confronted to perturbations. This is why during the servoing process the light of the room is switched on and off. The visual servoing process is then initialized and launched in nominal conditions and global perturbations of the scene happen during the positioning task. Information relative to this experiment is shown on figures 7.3 and 7.4. The results of this experiment clearly show the robustness of the proposed approach in globally perturbed conditions. Here, the stabilization of the algorithm with no lights in the room shows a residual positioning error of $1mm$, which is very interesting at is shows that the algorithm is still able to remain accurate while perturbed. The experiment also allow to show the robustness of the method with relation to noise as noticeable grain appear in the image because of the lack of light. Turning the lights back on finally allow to reach a final error of $\Delta\mathbf{r}^* = \{0.03mm, 0.02mm, 0.01mm, 8.4e^{-4}rad, 4.1e^{-5}rad, 1.3e^{-3}rad\}$. That accuracy and the fact that the trajectory is not impacted by the conditions then come to validate the use of the proposed approach in globally perturbed conditions.

7.2.2.1 Comparison with classical MI-based visual servoing

In order to assess the differences between our algorithm and a classical MI-based direct visual servoing approach, the same experiment was realized using that method. The servoing was launched again and the lights were turned off as before for the SCV. Results

are exposed on figure 6.8. They show that in these conditions the SSD was not able to perform the task and drifted up to the point that it reached joint limits of the robot and had to be aborted. This comes to validate the proposed algorithm as when put in the same perturbed conditions, it succeeded where the classical SSD-based approach failed.

7.2.3 Visual servoing in perturbed conditions: occlusions

This final experiment aims to validate the proposed NMI-based visual servoing when confronted to occlusions of the scene. As it was exposed before, this type of perturbations lead to a loss of relevant information in the current image and a good visual tracking process needs to be robust in these conditions. This is why in this experiment an occlusion was added to he scene by adding and moving an object on the texture that is used to control the robot. Results of the proposed visual servoing approach when confronted to occlusions are shown on figures 7.6 and 7.7. This experiment is very interesting as it clearly shows the robustness of the proposed servoing approach when confronted to scene occlusions. Even though the occlusion is still present at the end of the process, the task converges with a very good accuracy of $\Delta\mathbf{r}^* = \{0.01mm, 0.04mm, 0.01mm, 1.3e^{-3}rad, 2.5e^{-5}rad, 1.1e^{-3}rad\}$. The proposed NMI-based visual servoing task is then not impacted whatsoever by this perturbation which is a very interesting result. Here again, the trajectory of the servoing task in these conditions is very close to the trajectory of the servoing in nominal conditions even though some noise is added to the trajectory, validating the fact that the proposed algorithm is very robust to occluded parts of the scene.

7.2.3.1 Comparison with classical MI-based visual servoing

A final comparison was then done in order to evaluate the proposed NMI approach when compared to the state of the art MI-based visual servoing. The similar experiment was therefore launched using the MI-based algorithm and the same object was introduced and moved upon the texture of the scene. Results are exposed on figure 7.8. Let us remark that the occlusion was removed at the end of the process, therefore creating the spike in the MI value. Just as the proposed NMI approach, the MI is able to cope with the situation and converge, but its trajectory is much more impacted by the conditions of the experiment, which tend to show that our approach is better suited to perform visual servoing in these conditions than the classical MI-based algorithm.

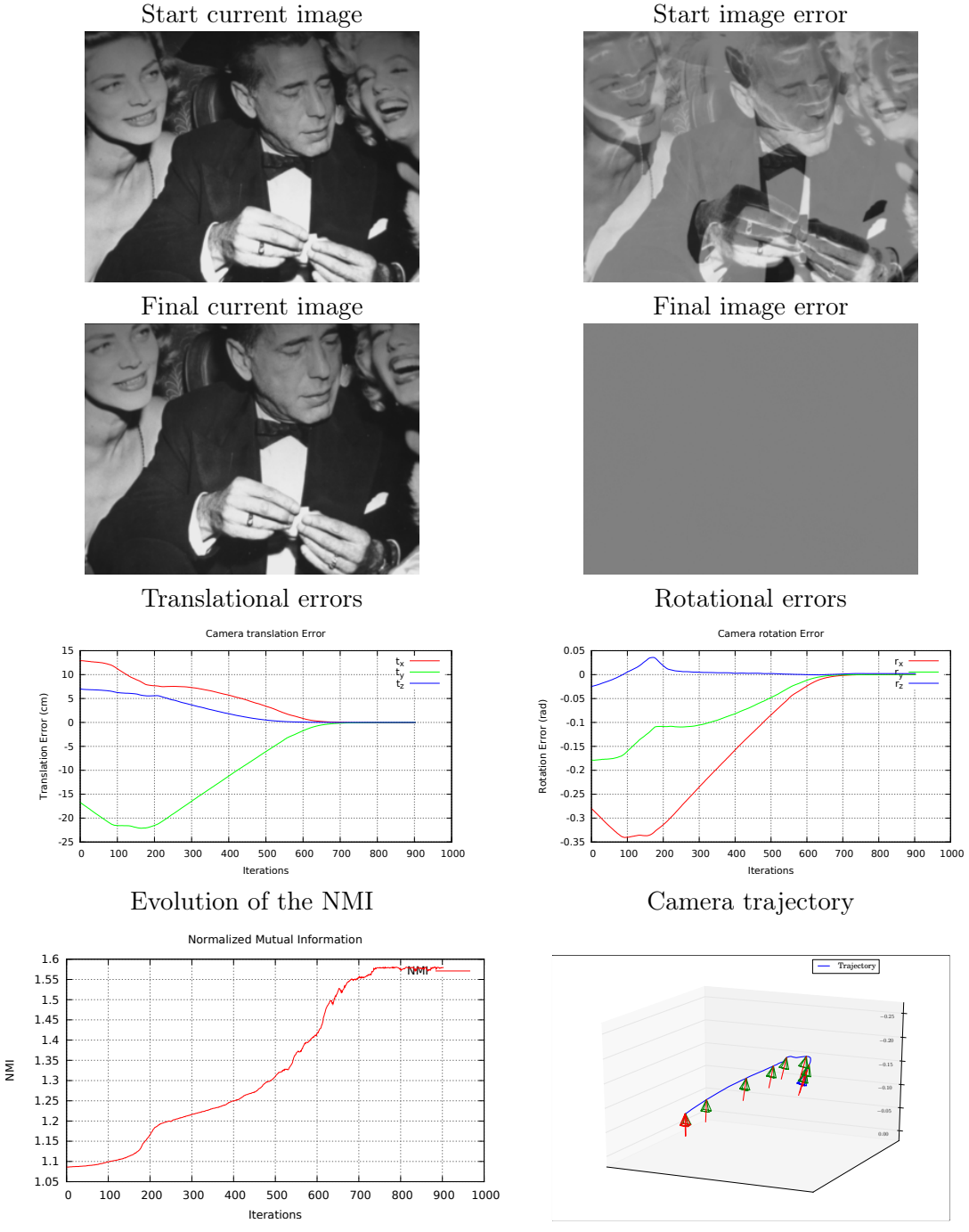


Figure 7.1: Results of NMI-based visual servoing in nominal conditions.

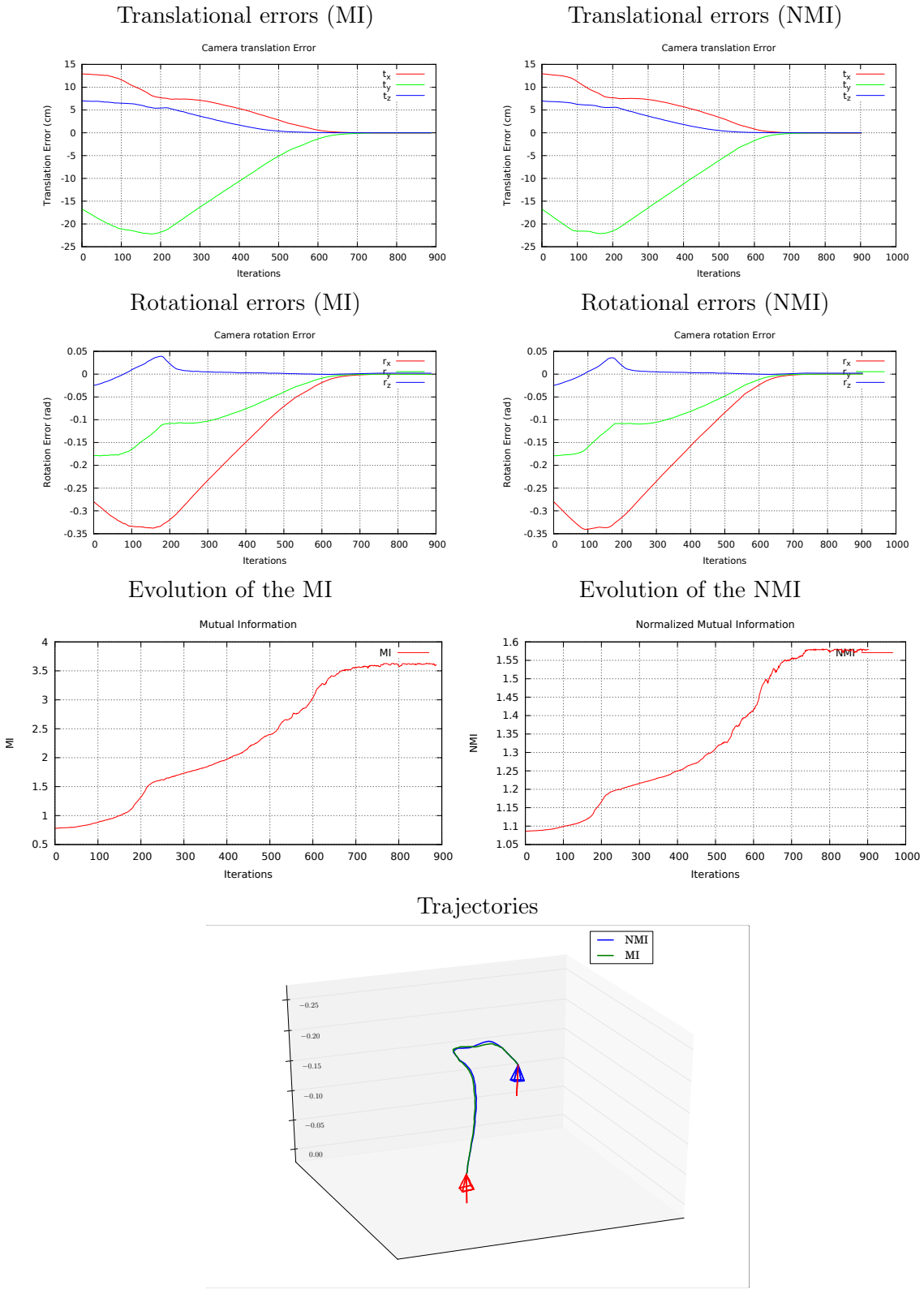


Figure 7.2: Comparison of NMI and MI-based visual servoing methods in nominal conditions.

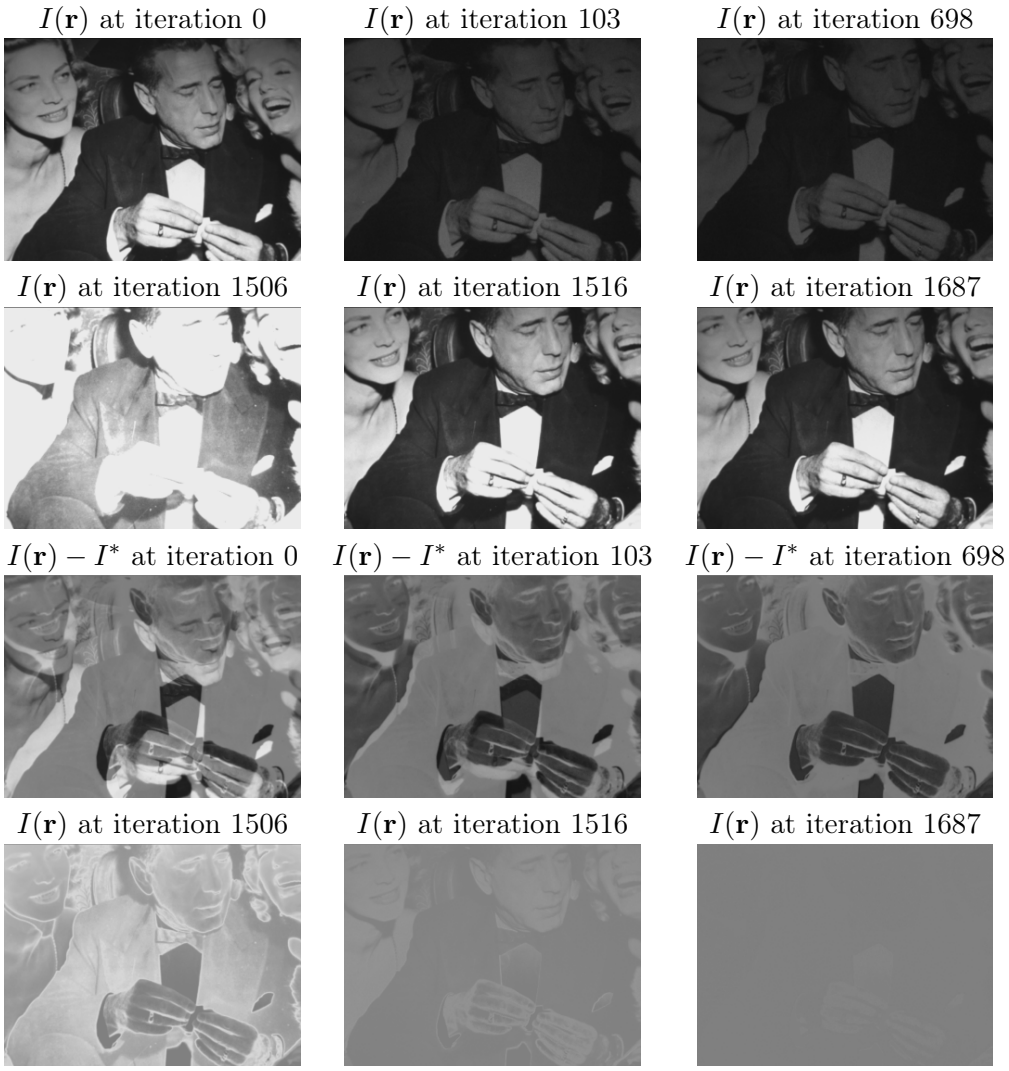


Figure 7.3: Results of NMI-based visual servoing in light varying conditions.

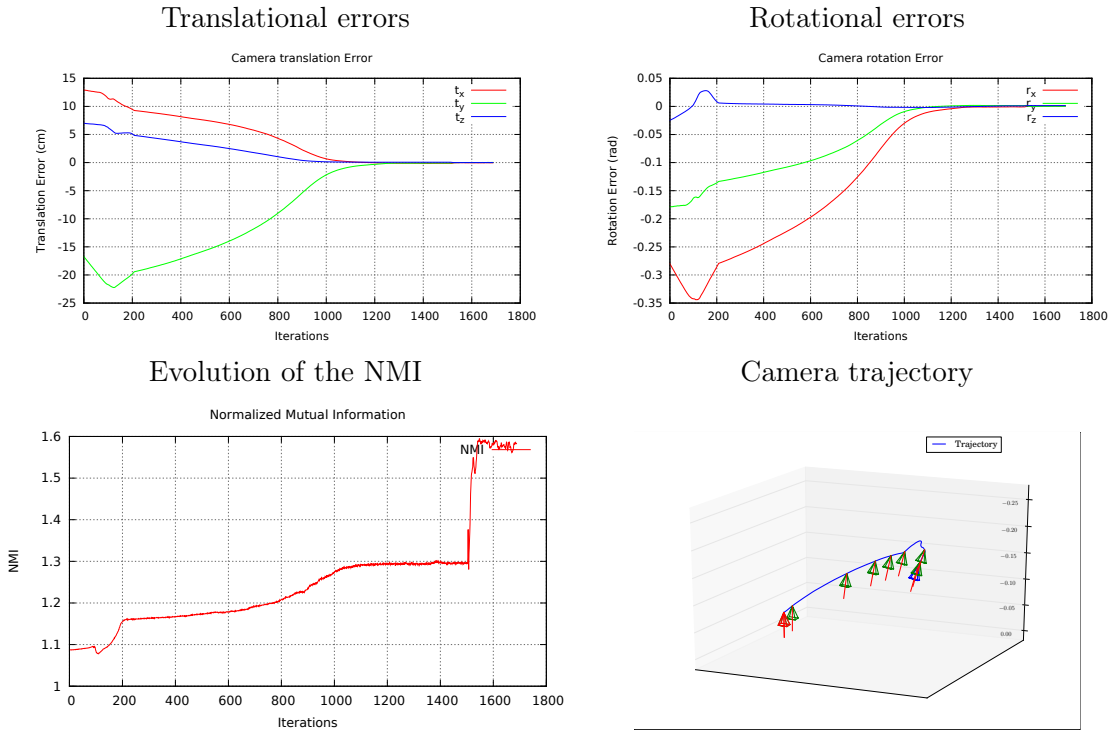


Figure 7.4: Results of NMI-based visual servoing in light varying conditions.

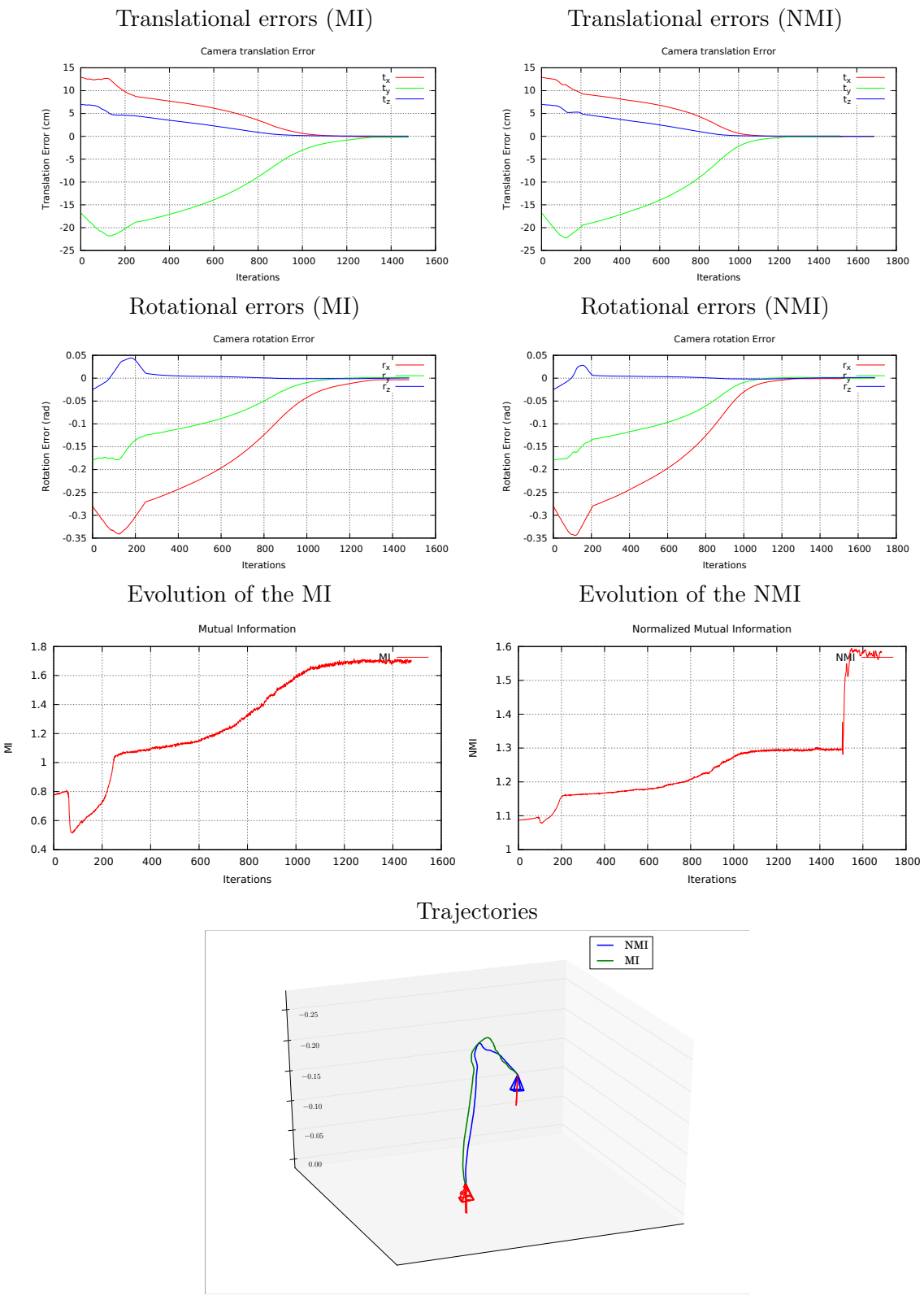


Figure 7.5: Comparison of NMI and MI-based visual servoing methods in light varying conditions.

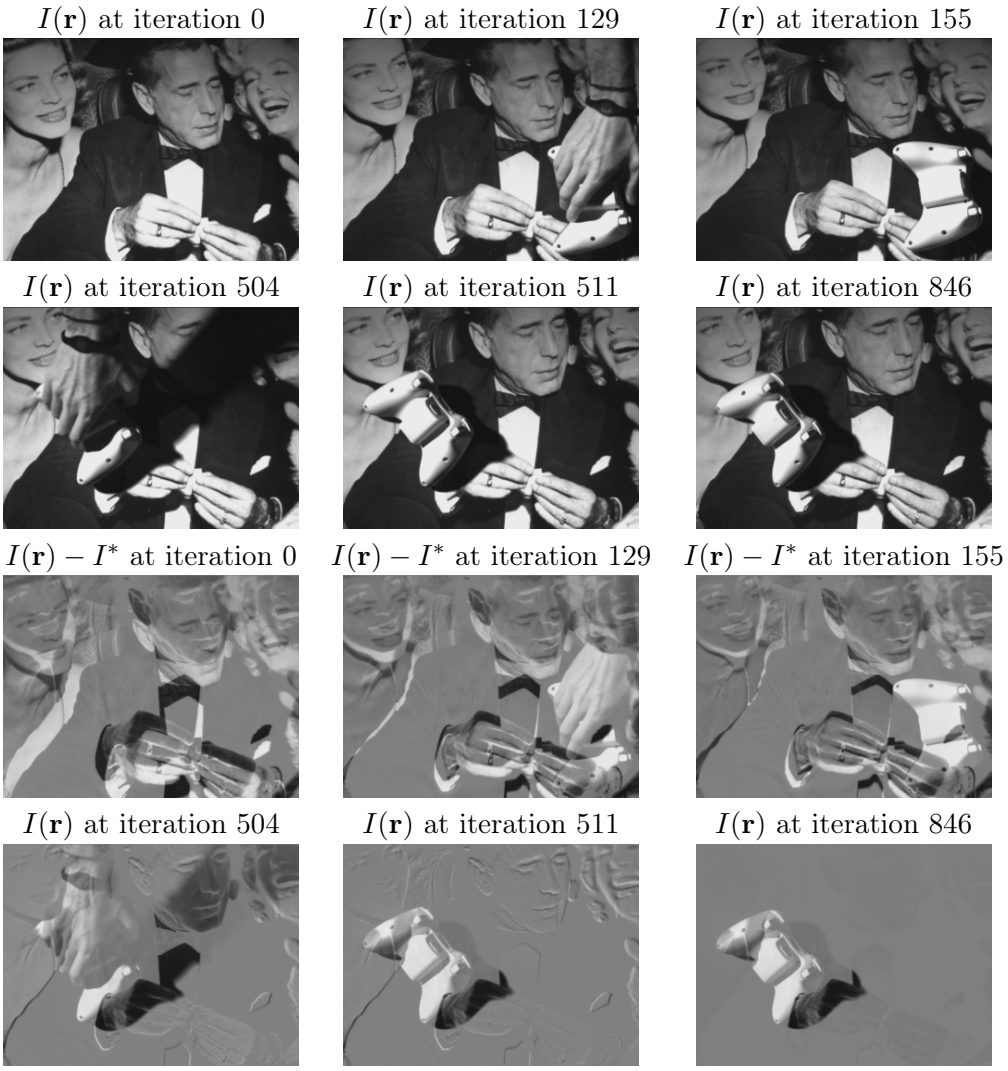


Figure 7.6: Results of NMI-based visual servoing when confronted to occlusions.

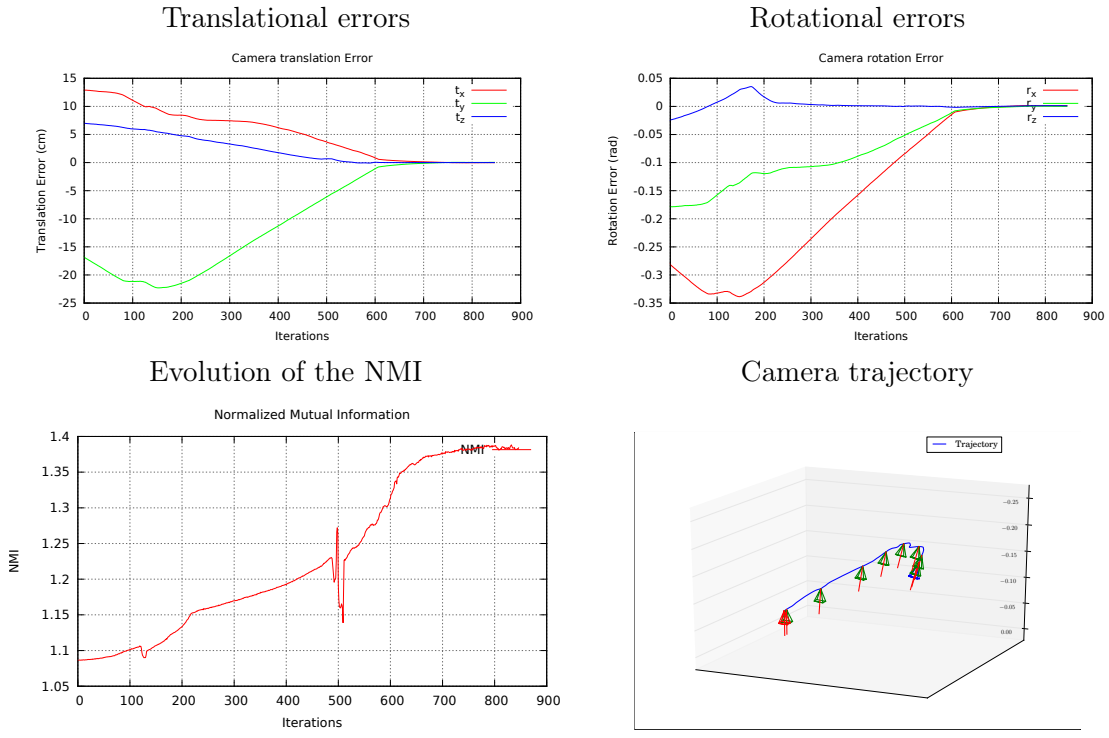


Figure 7.7: Results of NMI-based visual servoing when confronted to occlusions.

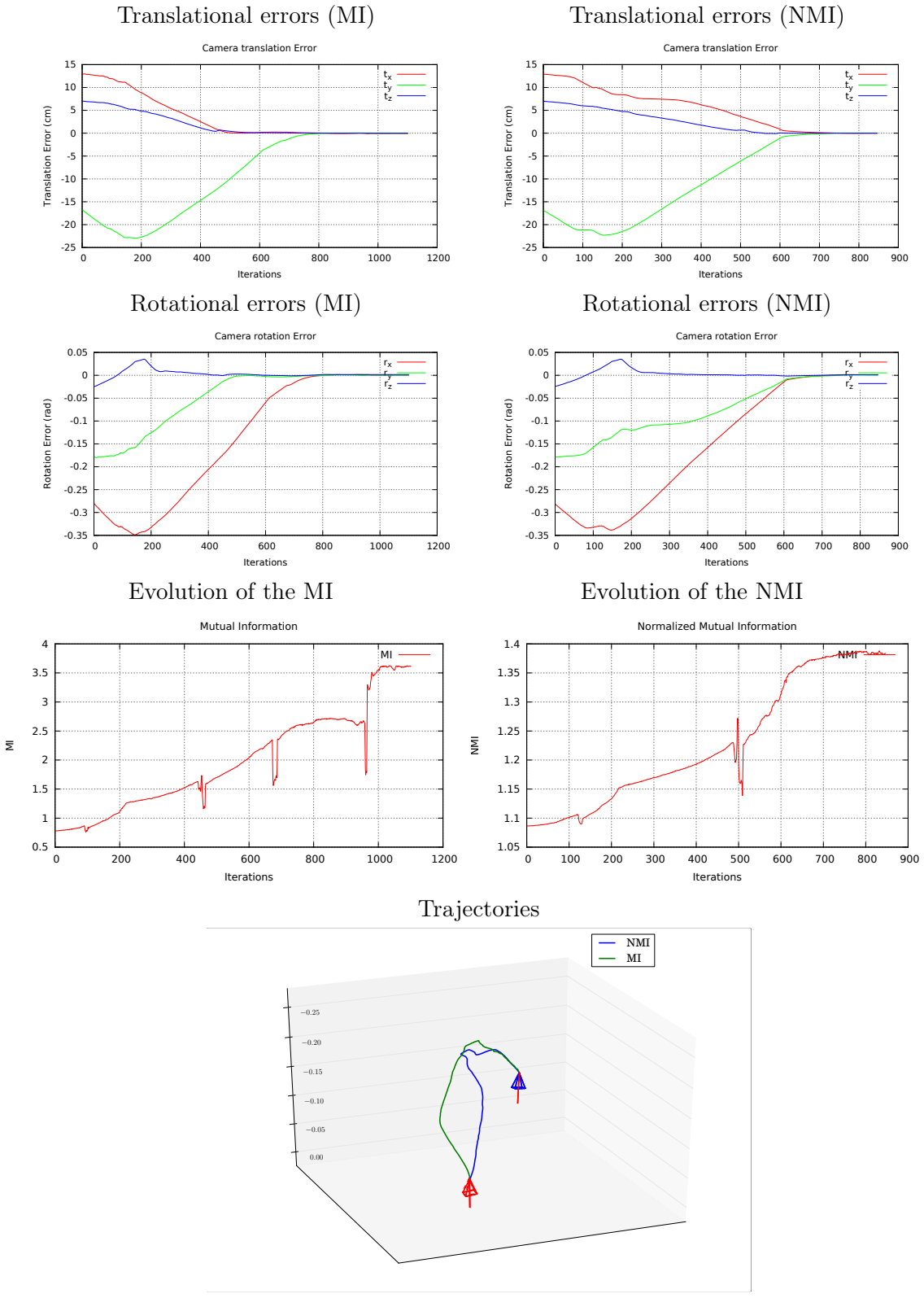


Figure 7.8: Comparison of NMI and MI-based visual servoing methods when confronted to occlusions.

7.3 Extending the NMI visual servoing to omnidirectional cameras

The previous visual servoing approaches have been designed to be used on perspective cameras. The problem of such cameras is that they have access to a limited amount of visual information, since their field of view is limited. This is why we propose to extend the NMI approach, which has shown to give the best results in our experiments, to omnidirectional cameras. The proposed method is then generic, as it does not depend on any particular projection model. Having access to more information of the scene, the resulting visual servoing processes are then very robust and accurate. The problem when using such sensors is the fact that the projection pipeline which allows to get an image point from a 3D point is very different with relation to the perspective case. This is why in the reminder of this chapter, the central projection model commonly used to handle omnidirectional cameras is recalled, before it is used to adapt the previously proposed approach to fish-eye cameras.

7.3.1 Using a central projection model

Using a central projection model is adapted to most omnidirectional cameras. The most commonly used model is the unified model proposed in [Barreto, 2001] for catadioptric cameras. It has also been shown to be adaptable to fish-eye lenses in [Ying and Hu, 2004]. Following the illustration shown in figure 7.9, a 3D point ${}^c\mathbf{X}$ is first projected on a normalized sphere, giving ${}^c\mathbf{X}_n$ which is then projected onto the projection plane. A meter-to-pixel conversion such as the one detailed in equation (1.15) is finally used to compute the point coordinates in the image. Following this model, for a 3D point ${}^c\mathbf{X} = [X,\ Y,\ Z]^\top$, its projection in the image plane is given by:

$$\bar{\mathbf{x}} = pr_\gamma({}^c\mathbf{X}) \quad \text{with} \quad \begin{cases} x = \frac{X}{Z+\xi\rho} \\ y = \frac{Y}{Z+\xi\rho} \end{cases} \tag{7.23}$$

where $\rho = \sqrt{X^2+Y^2+Z^2}$ and ξ is an intrinsic parameter of the camera which can be estimated through a calibration process. The coordinates of X_n, the point on the unit sphere can also be computed such that:

$$\mathbf{X}_n = pr_\mathcal{S}({}^c\mathbf{X}) \quad \text{with} \quad \begin{cases} X_\mathcal{S} = \frac{X}{\rho} \\ Y_\mathcal{S} = \frac{Y}{\rho} \\ Z_\mathcal{S} = \frac{Z}{\rho} \end{cases} \tag{7.24}$$

${}^c\mathbf{X}_n$ can also be estimated by an inverse projection of the point on the image plane:

$${}^c\mathbf{X}_n = pr_\gamma^{-1}(\bar{\mathbf{x}}) = \begin{pmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1}x \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1}y \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} - \xi \end{pmatrix} \tag{7.25}$$

which is of interest when handling image data from omnidirectional cameras. Since the resolution is not the same everywhere in the image, working directly on the normalized sphere can be very useful to get more accurate results.

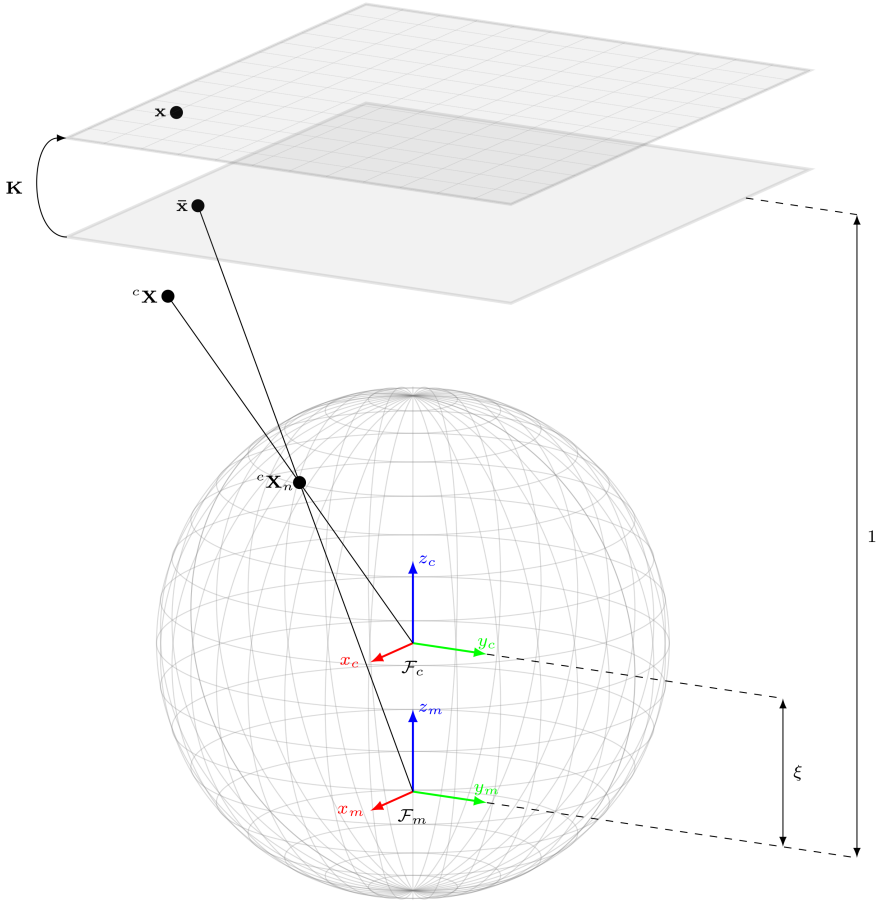


Figure 7.9: Illustration of the unified projection model (proposed with the authorization of [Markovic et al., 2014]). Full pipeline of projection from a 3D point ${}^{c}\mathbf{X}$ to a point $\mathbf{x}$ in the image.

7.3.2 Considering fish-eye images for visual servoing

When designing the proposed visual servoing task, two main issues had to be taken into account. First, the images gathered by the camera do not only represent the scene. An important part of the image is composed of black pixels where no information appears and of a circular zone containing the image data (see figure 7.11 for visualization of the setup. A first step then needs to be taken in order to only work on that part of the image and not take into account the black pixels which will not move while performing the visual servoing task. The second and most important problem is the fact that an object will not have the same projection in the image depending on its position. Its size and shape will differ according to its position in the scene as it will not be located on the same part of the equivalence sphere, a better resolution being given to points above the poles of the sphere. This creates two possibilities when computing the control law. Image gradients are needed in order to compute the derivatives of the probability density functions, but theneighbourhood of each point can either be considered directly on the image plane or can be sampled around the point on the equivalence sphere and then projected back onto the image. The difference between the two strategies can be seen on figure 7.11. [Caron et al., 2010, Demonceaux et al., 2011] have demonstrated that working on the equivalence sphere is more adapted to the use of omnidirectional sensors.

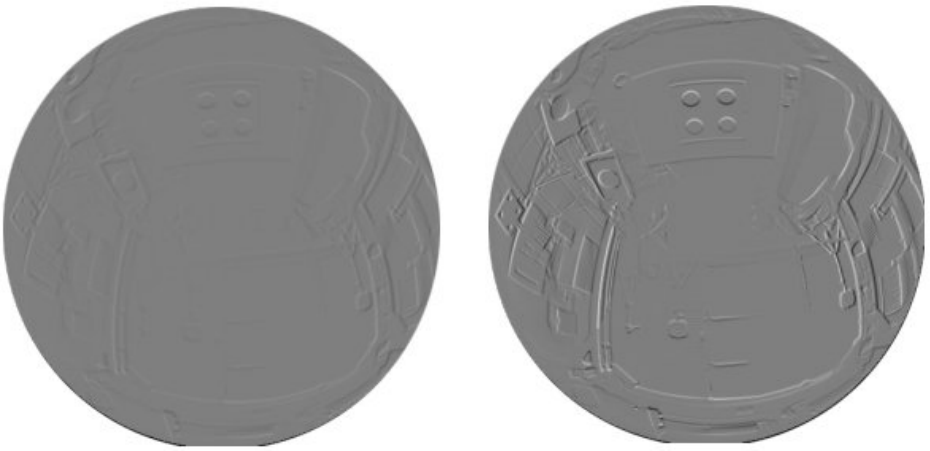


Figure 7.10: From left to right: Gradient image computed on the image plane and gradient image computed from the equivalence sphere. [Caron et al., 2010]

7.3.3 Redefinition of the NMI-based control law

Just as the computation of the gradient of the image, the computation of the interaction matrix of a point $\mathbf{L_x}$, involved in the computation of the derivatives of the probability density functions of the images, can be done considering either the 3D point on the image plane or the point on the equivalence sphere. Depending whether the considered point belongs to the image plane (resulting in an image plane visual servoing task) or to the equivalence sphere (resulting in a Cartesian spherical visual servoing task), two control laws can be defined. The differences between the two approaches are exposed in this section. Let us remark that only the part of the algorithm that differ from the perspective case is exposed hereafter.

7.3.3.1 Image Plane Visual Servoing (IPVS)

Considering image plane visual servoing (*i.e* considering the point $\mathbf{x}$ on the image plane), gradients computation are the same as with perspective cameras, as the neighbourhood of every point is considered directly in the image. The interaction matrix however changes. It was shown in [Espiau et al., 1992b] that the interaction matrix can be expressed as the product of two Jacobians:

$$\mathbf{L_x} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{r}} \tag{7.26}$$

where $\frac{\partial \mathbf{x}}{\partial \mathbf{X}}$ represents the motion of a point in the image with relation to the corresponding point in 3D and $\frac{\partial \mathbf{X}}{\partial \mathbf{r}}$ the motion of the 3D point with relation to the camera pose. The advantage of this formulation is that the second term of the product does not depend on the type of projection and therefore only the first part needs to be redefined. The interaction matrix can then be computed following the same steps as in the case of perspective images shown in section 5.1.1, replacing the perspective projection by the unified model given in equation (7.23):

$$\mathbf{L_x} = \begin{pmatrix} -\frac{1+x^2(1-\xi(\gamma+\xi))+y^2}{\rho(\gamma+\xi)} & \frac{\xi xy}{\rho} & \frac{\gamma x}{\rho} & xy & -\frac{(1+x^2)\gamma-\xi y^2}{\gamma+\xi} & y \\ \frac{\xi xy}{\rho} & -\frac{1+y^2(1-\xi(\gamma+\xi))+x^2}{\rho(\gamma+\xi)} & \frac{\gamma y}{\rho} & \frac{(1+y^2)\gamma-\xi x^2}{\gamma+\xi} & -xy & -x \end{pmatrix} \tag{7.27}$$

with $\gamma = \sqrt{1 + (1 - \xi^2)(x^2 + y^2)}$. Details on how to get this expression of the interaction matrix can be found in [Barreto et al., 2003].

7.3.4 Cartesian Spherical Visual Servoing (CSVS)

Another way to perform visual servoing when considering a unified projection model is to compute image gradients and interaction matrices considering the equivalence sphere rather than the image plane. In order to compute image gradients starting from the equivalence sphere, a 3D neighbourhood is sampled around the point ${}^c\mathbf{X}_n$ and every point is then projected onto the image plane in order to compute the gradient. Let us notice that, as the resulting positions in the image are real values, bilinear interpolation is used in order to evaluate luminance values. Following this strategy, the interaction matrix of a point can be considered about ${}^c\mathbf{X}_n$. In this case, it is defined as shown in [Hamel, 2002]:

$$\mathbf{L}_{\mathbf{c}\mathbf{X}_\mathbf{n}} = \frac{\partial {}^c\mathbf{X}_n}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{r}} = \begin{pmatrix} \frac{1}{\rho}({}^c\mathbf{X}_n \, {}^c\mathbf{X}_n^\top - \mathbf{I}_3) & [{}^c\mathbf{X}_n]_\times \end{pmatrix} \tag{7.28}$$

where $[{}^c\mathbf{X}_n]_\times$ is the skew matrix of the vector ${}^c\mathbf{X}_n$.

7.4 Experimental results

Several experiments are performed here to validate the proposed method using the NMI visual servoing approach detailed before. The experimental setup is very similar to what was done in the case of perspective images. This time however, the camera mounted on the robot is a fish-eye lens 7.11, allowing a 180° field of view. Following the same experimentation process, the servoing task was realized in different conditions and the results were saved for analysis. For the first experiment, in nominal conditions, both the IPVS and CSVS algorithms are considered in order to validate the proposed approach in both cases, but for the latter experiments, only the CSVS method is used as it was shown to be more adapted to omnidirectional cameras [Caron et al., 2010]. For readability purposes, graphics of NMI were created depicting the evolution of (NMI-1) instead of NMI, giving a measure between 0 and 1.

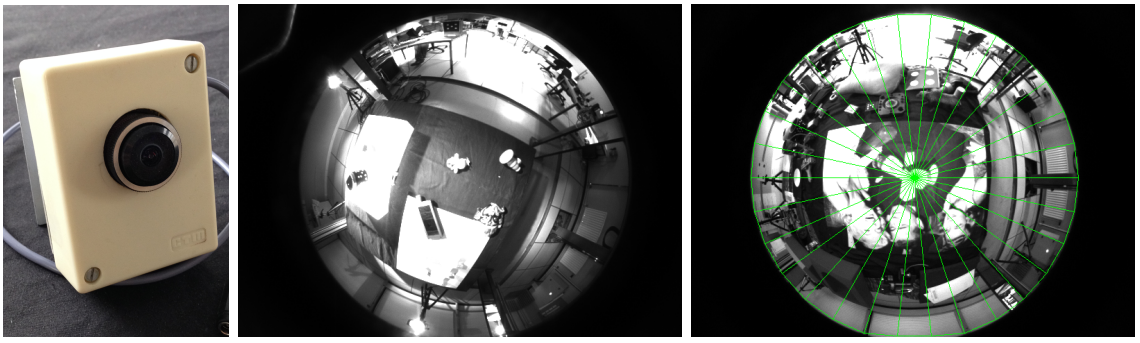


Figure 7.11: From left to right: Fish-eye camera used for our experiments, image gathered by the camera and zone initialization by Delaunay triangulation.

7.4.1 Visual servoing in nominal conditions

The aim of this experiment is both to validate the proposed approach in nominal conditions and to compare the two strategies exposed in section 7.3.3. Thus, both tasks were launched

from the same sets of positions and the results of the servoing tasks were monitored (see figures 7.13 to 7.12). Results show that both positioning tasks are successful, with a good accuracy since the residual position error upon convergence is below $0.1mm$ for both tasks. Analyzing the behaviour of both methods shows that the IPVS algorithm presents some oscillations before stabilizing, which does not appear with the CSVS technique. This comes to validate the fact that working on the equivalence sphere is a better solution, as it takes more into account the properties of the unified projection model used here for visual servoing and results in a lesser amount of approximations needed to compute the gradients and interaction matrices.

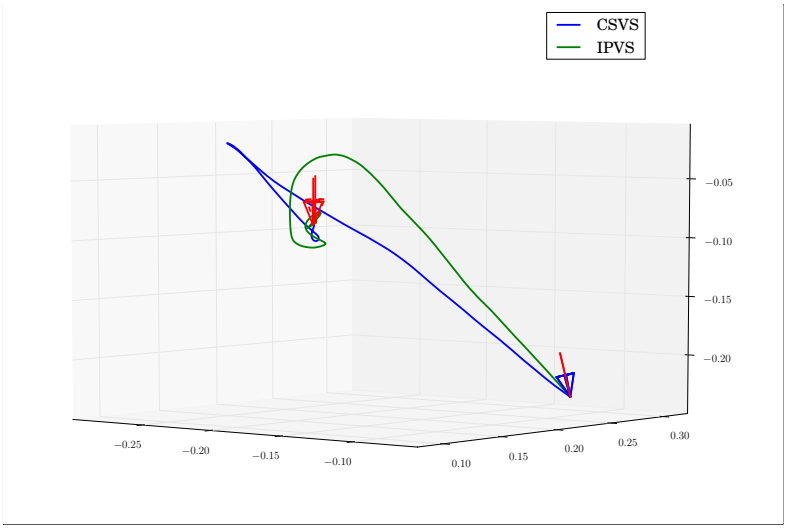


Figure 7.12: Trajectory of the camera in the 3D world. The IPVS method shows oscillations upon convergence.

7.4.2 Experiment 2: light variations

The objective of this second experiment is to assess the robustness of the method towards illumination changes. This was done using the CSVS approach alone, since the results in our nominal experiments show a better behaviour compared to the IPVS method, validating the same claim made in [Caron et al., 2010] in our situation. Light variations were achieved during the task by using spot lights present in the field of view of the camera in order to change the illumination of the room. Even though the final positioning accuracy is impacted by the perturbations, with a final residual error of $2mm$, the task is still successful and the robot stabilizes at the desired position even from distant starting points. The trajectory is moderately impacted, as the overall course of the camera stays the same even though small variations appear. Figure 7.14 also shows the evolution of the cost function and positioning errors which are very similar to the nominal case, which shows that the proposed visual servoing process is robust to illumination variations.

7.4.3 Experiment 3: impact of occlusions

Finally, an experiment was also designed in order to evaluate the robustness of the method with respect to occlusions. In order to do that, several objects are moved and people enter the camera field of view during the servoing task. Results are shown on figure 7.15. The proposed visual servoing algorithm was not affected by those perturbations of the scene

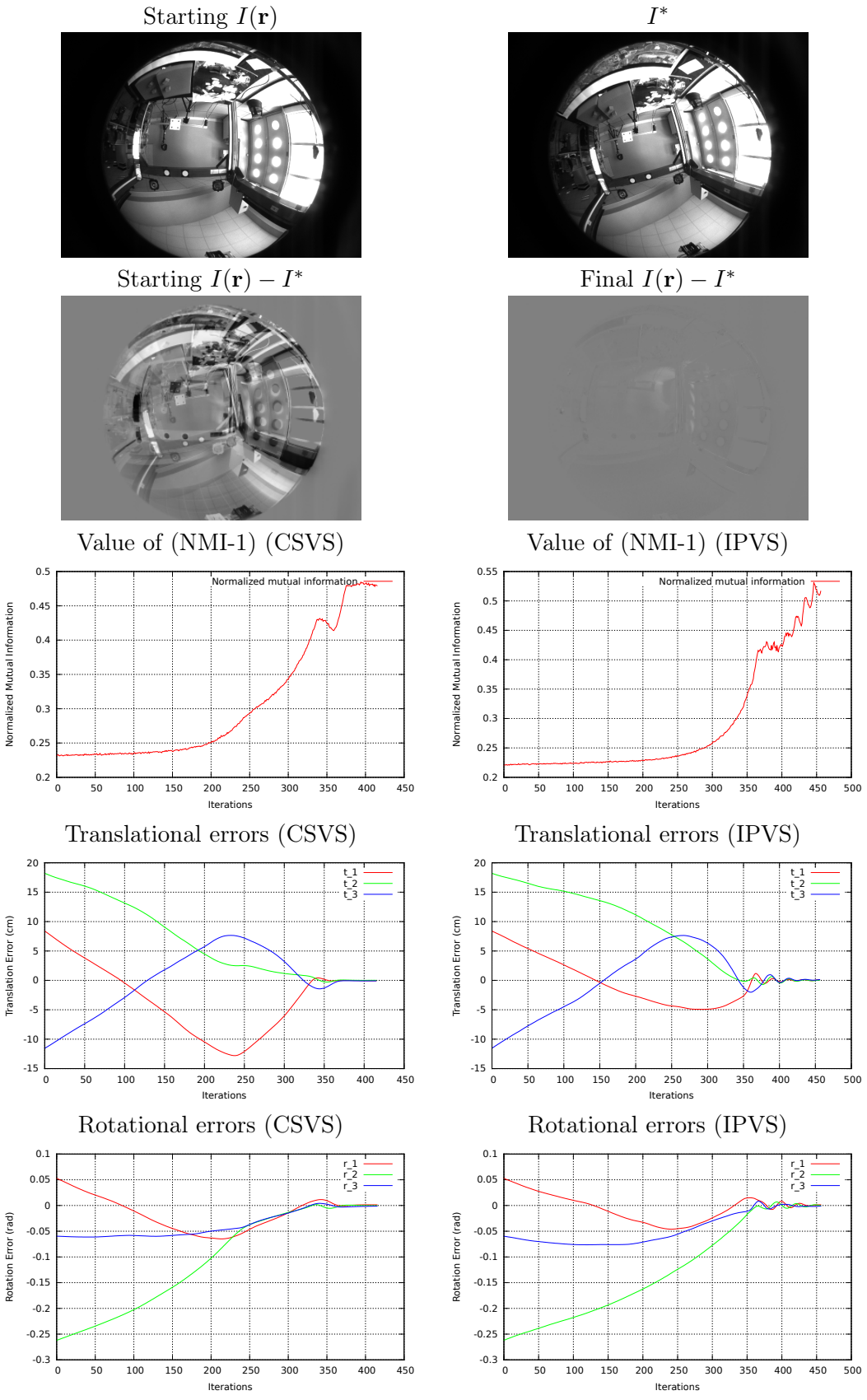


Figure 7.13: Results of the IPVS and CSVS algorithms in nominal conditions.

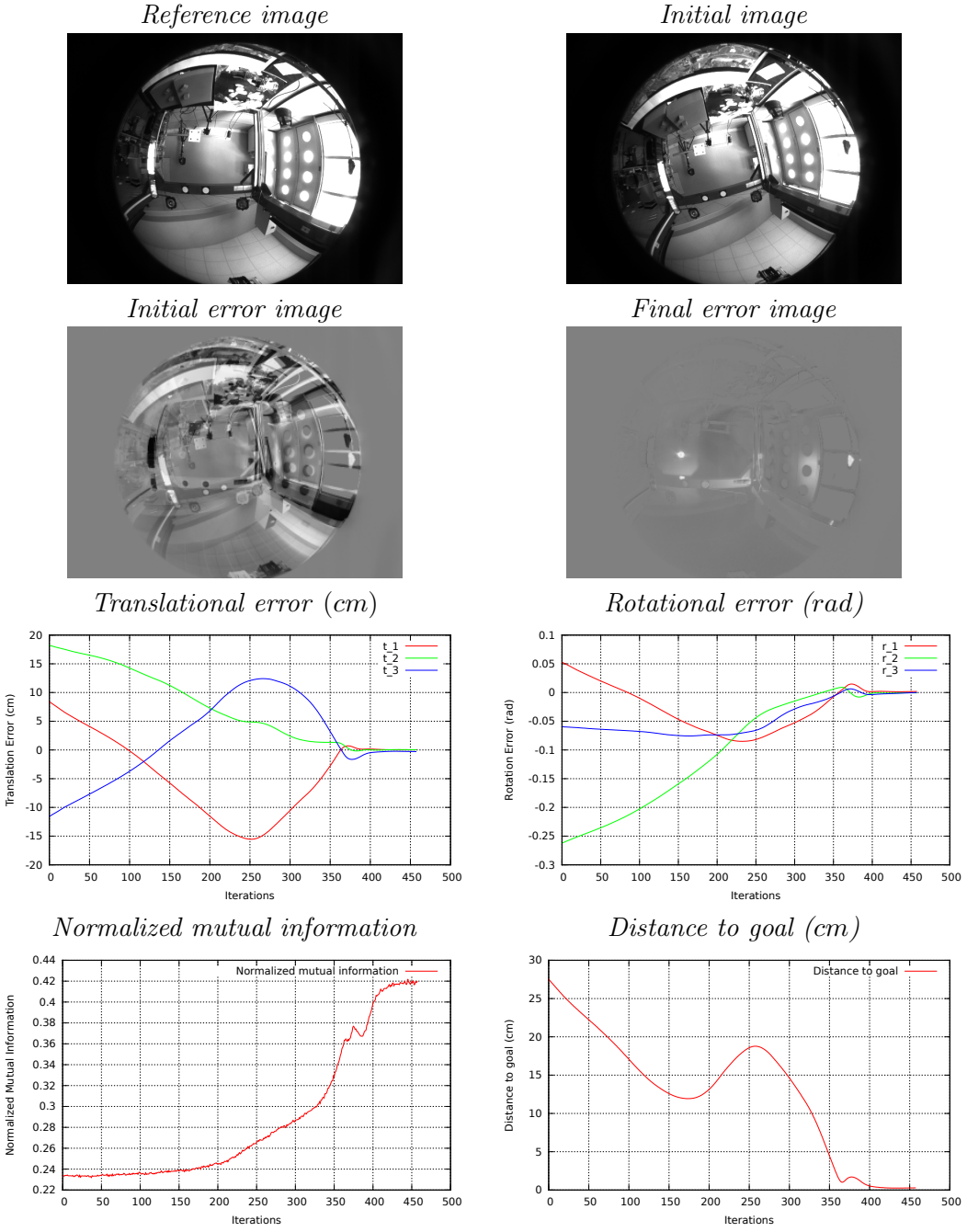


Figure 7.14: Omnidirectional visual servoing using normalized mutual information: impact of illumination changes.

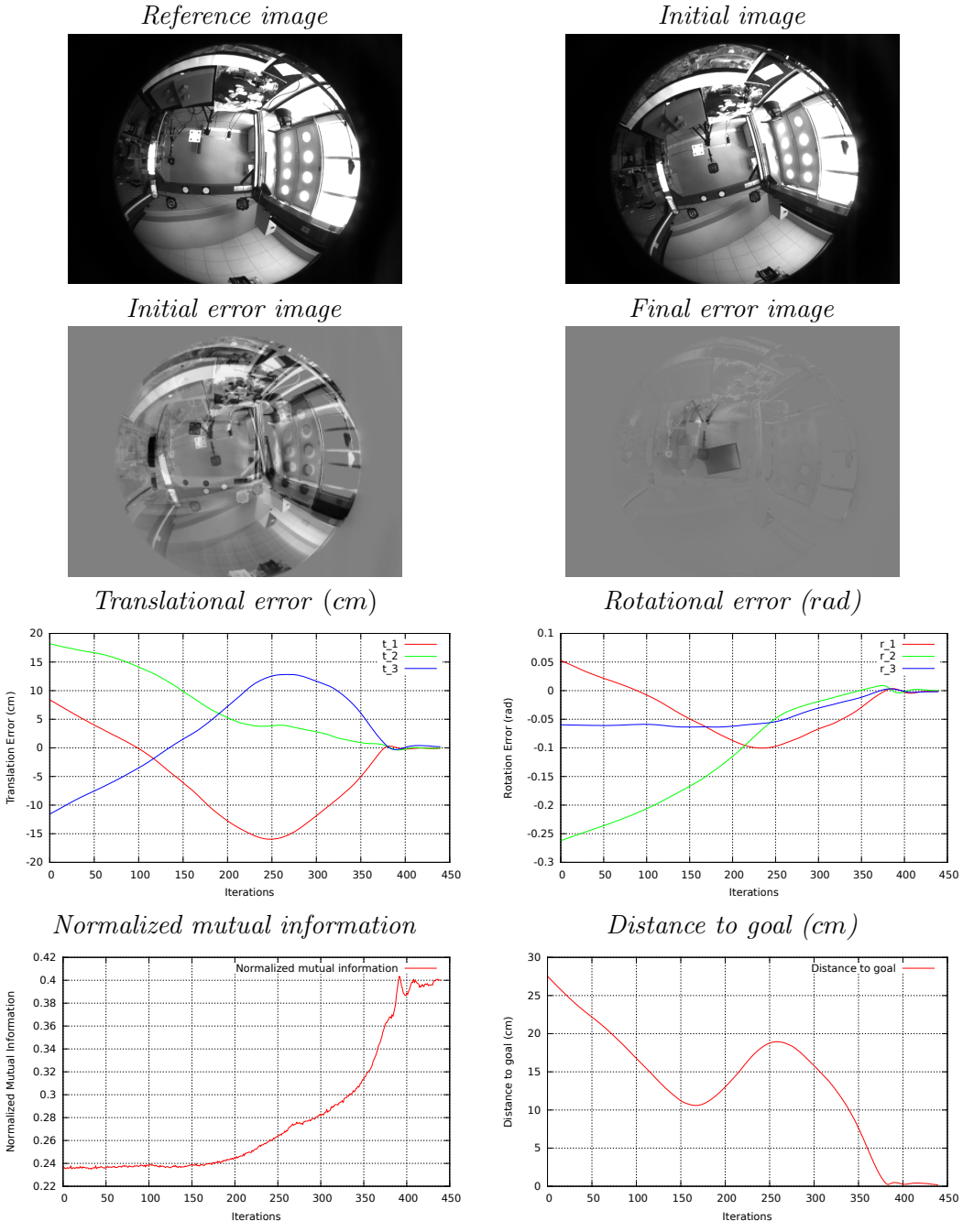


Figure 7.15: Omnidirectional visual servoing using normalized mutual information: impact of occlusions.

as both the error and NMI graphs show no significant oscillations or important variations. The servoing task then succeeds with a very good accuracy, the final residual positioning error being below $2mm$. Here again the camera stabilizes with an very low final positioning error and the trajectory is very similar to the nominal case, which shows that the proposed visual servoing process is robust to occlusions in the scene.

7.5 Conclusion

In this chapter, a new approach to dense visual servoing has been proposed. Using a normalized formulation of the mutual information, the technique has been shown to be robust to a wide variety of scene perturbations, while allowing an assessment of the similarity between the goal and current images, which is something that is impossible to do with the classical formulation of the MI, as its upper bound is not a constant value. The approach, after being validated on a real six degree of freedom robot, has been extended to omnidirectional cameras. Two ways of performing servoing on omnidirectional cameras have then been detailed, working either in the image plane or on the equivalence sphere. The two techniques are finally validated in several perturbed conditions on a real six degree of freedom robot on which a fish-eye lens is mounted.

Conclusion

Advances in technology made possible thanks to research in the domain of computer vision are now more and more present in our society. Visual tracking and visual servoing are often key elements of those technologies. They have allowed autonomous applications in the fields of robot manipulation of complex object, vehicle navigation, augmented reality or special effects for example. Most of the approaches used are based on the extraction of geometrical features in the image sequences in order to estimate motion parameters. While these approaches have shown, since the early 90's, great results in many situations, they show two non negligible negative points that can affect the efficiency of the task: they need to rely on the extraction of features in the images and they only use a small amount of the available information. In order to circumvent those drawbacks, recent approaches have proposed to directly consider the luminance values given by a set of pixels covering the whole image. The process then becomes the alignment of two image templates using similarity or dissimilarity functions. But since most approaches use a simple luminance difference in order to perform the alignment, other layers are often needed in order for the task to be robust. Indeed, considering a difference between two sets of luminance is only possible if the luminance of a point in the sequence is considered constant throughout the whole process, which is not the case in most outdoor scenarios for example. This is why in this document we proposed solutions that are based on image histograms and entropies, as even if the representation of the information present in the scene might change the information in itself does not. Our main focus has then been on designing dense visual tracking and visual servoing tasks whose robustness is a direct result of the choice of the (dis)similarity function used and do not require any additional steps to cope with scene variations.

Several research works on this subject have then been proposed in this document. In the domain of model-based tracking, two algorithms have been proposed to localize a camera with respect to a moving object whose faces appear and disappear during an image sequence. The first uses the sum of conditional variance in a simple tracking approach that is robust to global variations of the scene. The second one extends the same principle to the use of the mutual information in order to get a more robust, albeit more complex, tracking process. The two robust cost functions have then been used in the context of non-rigid registration, resulting in two robust non-rigid tracking processes. The four proposed tracking approaches have been validated through several experiments, including tests on sequences undergoing perturbations and convergence domain analysis.

Since there is a strong duality between the visual tracking and visual servoing problems, the (dis)similarity functions considered in the proposed tracking approaches have lead to new visual servoing processes. A first approach is designed considering the sum of conditional variance instead of the sum of squared differences to perform a simple robust visual servoing task. Although the common use of the sum of conditional variance is to adapt the reference template to the current camera view, we show that redefining the measure in order to adapt the current view to the reference image allows to define a more computationally efficient servoing task. Then, another formulation of the mutual information is introduced. Contrary to the classical mutual information, the proposed formulation has a constant upper bound, which allows to assess the similarity of any couple of images. That similarity measure is then designed to elaborate a robust visual

servoing process. Finally, this approach is extended to omnidirectional cameras. Here again, experimental validations are shown performing visual servoing tasks on a six degree of freedom robot in conditions where perturbations heavily impact the perception of the scene by the camera.

Perspectives

Several works could still be realized in order to extend our research. First, the proposed visual servoing tasks could be used in order to realize more complex robotic tasks such as a navigation application. Using the concept of visual path, where the trajectory of the navigation is represented by images taken along the path, a succession of visual servoing tasks using the sum of conditional variance or the normalized mutual information could be realized. Since those two (dis)similarity measures possess fixed optimum values, a switching criteria could easily be derived from their evolution to navigate along a path. Several paths could also be recorded, with intersection points. A graph-like representation of all the paths could then allow a robot to travel to an array of locations autonomously. Finally, studies on the 3D trajectory of the robot while performing the servoing task would be very interesting in order to assess if different ways of optimizing the cost functions could result in straighter trajectories in the 3D space.

Other visual tracking applications could also be designed by extending the proposed visual tracking algorithms. For the model-based approach, the use of the resulting pose of the camera should allow a servoing loop to optimize the difference between desired and current poses of the camera. This visual tracking algorithm could also be enhanced. An automatic initialization by detecting the model autonomously should be considered. This would allow to keep a bank of known objects that would then be used to automatically initialize tracking tasks on several objects.

The non-rigid visual tracking algorithms proposed could also be used for visual tracking purposes. To that end, we could imagine a visual servoing task based on the affine parameters of the thin-plate spline warping function. That way, a robot could be positioned with respect to a deforming object by only taking into account its motion in space and not be impacted by its deformations. But since affine parameters should not allow to control six degrees of freedom of the robot, two options could be considered: the former would be to design a visual servoing task controlling only a part of the degrees of freedom of the robot. The latter would be to extend the thin-plate spline warp. Redefining the motion model so as not to be based on an affine warp basis anymore but instead on a homography or a quadratic motion model for example. Our visual tracking works could also be extended to other robust similarity functions that have proven to be useful in such scenarios like the zero-mean normalized cross correlation or other divergences commonly used in theory of information applications.

Finally, it would be very interesting to improve the implementations of our applications in order to get even more efficient algorithms that could be used on-board dedicated robotic hardware. Considering tools such as Cuda or OpenCL to take advantage of the possibilities of GPU parallelization should result in very fast tasks that could even be used directly to control UAVs.

PART 3
Appendices

Appendix A: General tools

In this document, vectors are referred to using normal-sized bold letters. Matrices are noted with bold capital letters. The inverse of a matrix of size $r \times c$ $\mathbf{A}$ and its transpose matrix are then respectively $\mathbf{A}^{-1}$ and $\mathbf{A}^{\top}$. The identity matrix is referred to as $\mathbf{I}$. The Moore-Penrose pseudo-inverse [Moore, 1920, Penrose, 1955] of a matrix $\mathbf{A}$ is noted $\mathbf{A}^{+}$. The pseudo-inverse matrix is a generalization of the inverse matrix and is widely used to solve problems where matrices are not invertible. It is defined as:

$$\begin{aligned} \mathbf{A}^{+} &= \left(\mathbf{A}^{\top}\mathbf{A}\right)^{-1}\mathbf{A}^{\top} \qquad \text{when} \qquad c \leq r \\ \mathbf{A}^{+} &= \mathbf{A}^{\top}\left(\mathbf{A}\mathbf{A}^{\top}\right)^{-1} \qquad \text{when} \qquad c > r \end{aligned} \tag{7.29}$$

Other methods can be used to compute the pseudo inverse matrix, such as the Singular Value Decomposition (which will be referred to as SVD). The SVD is a factorization of a matrix. It is very useful for several applications such as obtaining the solution of an exact linear system or finding a least-squares solution in the case of an over-constrained system. If $\mathbf{A}$ is a $r \times c$ matrix composed of real components, there exists a factorization such that:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top} \tag{7.30}$$

where $\mathbf{U}$ and $\mathbf{V}^{\top}$ are respectively $r \times r$ and $c \times c$ unitary matrices and $\mathbf{\Sigma}$ is a $r \times c$ diagonal matrix. When using the SVD to compute a pseudo-inverse, one must compute:

$$\mathbf{A}^{+} = \mathbf{V}\mathbf{\Sigma}^{+}\mathbf{U}^{\top} \tag{7.31}$$

where $\mathbf{\Sigma}^{+}$ is the pseudo-inverse of $\mathbf{\Sigma}$. It is formed by replacing every non-zero diagonal entry of $\mathbf{\Sigma}$ by its reciprocal and transposing the resulting matrix.

References

[Arad and Reisfeld, 1995] Arad, N. and Reisfeld, D. (1995). Image warping using few anchor points and radial functions. In *Computer Graphics Forum*, volume 14, pages 35–46. Wiley Online Library.

[Baker and Matthews, 2001] Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'01*, pages 1090 – 1097.

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *Int. Journal of Computer Vision*, 56(3):221–255.

[Barreto, 2001] Barreto, J.P. Araujo, H. (2001). Issues on the geometry of central catadioptric images. In *Int. Conf. on Computer Vision and Pattern Recognition*, Hawai, USA.

[Barreto et al., 2003] Barreto, J. P., Martin, F., and Horaud, R. (2003). Visual servoing/tracking using central catadioptric images. In *Experimental Robotics VIII*, pages 245–254. Springer.

[Bartoli et al., 2004] Bartoli, A., Zisserman, A., et al. (2004). Direct estimation of non-rigid registrations. In *British machine vision conference*, pages 899–908.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417.

[Benhimane and Malis, 2004] Benhimane, S. and Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJ Int. Conf. on Intelligent Robots Systems*, volume 943-948, page 1, Sendai, Japan.

[Benhimane and Malis, 2006a] Benhimane, S. and Malis, E. (2006a). Homography-based 2d visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'06*, Orlando, Fl.

[Benhimane and Malis, 2006b] Benhimane, S. and Malis, E. (2006b). Integration of Euclidean constraints in template-based visual tracking of piecewise-planar scenes. In *IEEE/RSJ International Conference on Intelligent Robots Systems*.

[Benhimane and Malis, 2007] Benhimane, S. and Malis, E. (2007). Homography-based 2d visual tracking and servoing. *Int. Journal of Robotics Research*, 26(7):661–676.

[Berger, 1994] Berger, M.-O. (1994). How to track efficiently piecewise curved contours with a view to reconstructing 3D objects. In *Int. Conf on Pattern Recognition, ICPR'94*, pages 32–36, Jerusalem.

[Blake and Isard, 1998] Blake, A. and Isard, M. (1998). *Active Contours*. Springer Verlag.

[Bookstein, 1989] Bookstein, F. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585.

[Boukir et al., 1998] Boukir, S., Bouthemy, P., Chaumette, F., and Juvin, D. (1998). A local method for contour matching and its parallel implementation. *Machine Vision and Application*, 10(5/6):321–330.

[Bouthemy, 1989] Bouthemy, P. (1989). A maximum likelihood framework for determining moving edges. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):499–511.

[Brooks et al., 1979] Brooks, R. A., Creiner, R., and Binford, T. O. (1979). The acronym model-based vision system. In *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1*, pages 105–113. Morgan Kaufmann Publishers Inc.

[Brown, 1971] Brown, D. (1971). Close-range camera calibration. *Photogrammetric Engineering*, 4(2):127–140.

[Brunet et al., 2011] Brunet, F., Gay-Bellile, V., Bartoli, A., Navab, N., and Malgouyres, R. (2011). Feature-driven direct non-rigid image registration. *International journal of computer vision*, 93(1):33–52.

[Caron et al., 2010] Caron, G., Marchand, E., and Mouaddib, E. (2010). Omnidirectional photometric visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'10*, Taipei, Taiwan.

[Chaumette and Hutchinson, 2006] Chaumette, F. and Hutchinson, S. (2006). Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90.

[Chaumette and Rives, 1990] Chaumette, F. and Rives, P. (1990). Vision-based-control for robotic tasks. In *IEEE Int. Workshop on Intelligent Motion Control*, pages 395–400, Istanbul, Turquie.

[Collewet and Marchand, 2011] Collewet, C. and Marchand, E. (2011). Photometric visual servoing. *IEEE Trans. on Robotics*, 27(4):828–834.

[Collewet et al., 2008] Collewet, C., Marchand, E., and Chaumette, F. (2008). Visual servoing set free from image processing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'08*, pages 81–86, Pasadena, CA.

[Comaniciu et al., 2000] Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE.

[Comport et al., 2006] Comport, A., Marchand, E., Pressigout, M., and Chaumette, F. (2006). Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Trans. on Visualization and Computer Graphics*, 12(4):615–628.

[Corke and Paul, 1990] Corke, P. I. and Paul, R. P. (1990). Video-rate visual servoing for robots. In *Experimental Robotics I*, pages 429–451. Springer.

[Crétual and Chaumette, 1998] Crétual, A. and Chaumette, F. (1998). Stabilisation dynamique d'une caméra pan and tilt par asservissement visuel sur des images sous-marines. Rapport final de convention Inria/Ifremer.

[Dame and Marchand, 2009] Dame, A. and Marchand, E. (2009). Entropy-based visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'09*, pages 707–713, Kobe, Japan.

[Dame and Marchand, 2010] Dame, A. and Marchand, E. (2010). Accurate real-time tracking using mutual information. In *IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'10*, Seoul, Korea.

[Dame and Marchand, 2011] Dame, A. and Marchand, E. (2011). Mutual information-based visual servoing. *IEEE Trans. on Robotics*, 27(5:958-969).

[Dame and Marchand, 2012] Dame, A. and Marchand, E. (2012). Second order optimization of mutual information for real-time image registration. *IEEE Trans. on Image Processing*, 21(9):4190–4203.

[Davison et al., 2007] Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

[Delabarre and Marchand, 2012] Delabarre, B. and Marchand, E. (2012). Visual servoing using the sum of conditional variance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'12*, pages 1689–1694, Vilamoura, Portugal.

[Demonceaux et al., 2011] Demonceaux, C., Vasseur, P., and Fougerolle, Y. (2011). Central catadioptric image processing with geodesic metric. *Image and Vision Computing*, 29(12):840–849.

[Deriche and Faugeras, 1990] Deriche, R. and Faugeras, O. (1990). Tracking line segments. In *European Conference on Computer Vision, ECCV 90*, pages 259–268.

[Dowson and Bowden, 2006] Dowson, N. and Bowden, R. (2006). A unifying framework for mutual information methods for use in non-linear optimisation. In *European Conference on Computer Vision, ECCV'06*, volume 1, pages 365–378.

[Drummond and Cipolla, 2002] Drummond, T. and Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):932–946.

[Espiau et al., 1992a] Espiau, B., Chaumette, F., and Rives, P. (1992a). A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326.

[Espiau et al., 1992b] Espiau, B., Chaumette, F., and Rives, P. (1992b). A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326.

[Faugeras, 1993] Faugeras, O. (1993). *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, Massachusetts.

[Faugeras and Lustman, 1988] Faugeras, O. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508.

[Feddema and Mitchell, 1989] Feddema, J. and Mitchell, O. (1989). Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. on Robotics and Automation*, 5(5):691–700.

[Fischler and Bolles, 1981] Fischler, N. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communication of the ACM*, 24(6):381–395.

[Fitzgibbon, 2003] Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153.

[Gay-Bellile et al., 2007] Gay-Bellile, V., Bartoli, A., and Sayd, P. (2007). Feature-driven direct non-rigid image registration. In *BMVC*, pages 1–10.

[Gay-Bellile et al., 2010] Gay-Bellile, V., Bartoli, A., and Sayd, P. (2010). Direct estimation of nonrigid registrations with image-based self-occlusion reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):87–104.

[Goodall, 1991] Goodall, C. (1991). Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 285–339.

[Hager and Belhumeur, 1998] Hager, G. and Belhumeur, P. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.

[Hager and Toyama, 1998] Hager, G. and Toyama, K. (1998). The XVision system: A general-purpose substrate for portable real-time vision applications. *Computer Vision and Image Understanding*, 69(1):23–37. Also Research Report Yale University.

[Hamel, 2002] Hamel, T. Mahony, R. (2002). Visual servoing of an under-actuated dynamic rigid-body system: An image-based approach. *IEEE Int. Trans. on Robotics and Automation*, Vol. 18, No. 2.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey Conference*, pages 147–151, Manchester.

[Hartley and Zisserman, 2001] Hartley, R. and Zisserman, A. (2001). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[Hashimoto, 1993] Hashimoto, K., editor (1993). *Visual Servoing : Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapor.

[Huber, 1981] Huber, P.-J. (1981). *Robust Statistics*. Wiler, New York.

[Irani and Anandan, 1998] Irani, M. and Anandan, P. (1998). Robust multi-sensor image alignment. In *IEEE Int. Conf. on Computer Vision, ICCV'98*, pages 959–966, Bombay, India.

[Irani et al., 1992] Irani, M., Rousso, B., and Peleg, S. (1992). Detecting and tracking multiple moving objects using temporal integration. In *ECCV'92*, pages 282–287.

[Jurie and Dhome, 2001] Jurie, F. and Dhome, M. (2001). Real time 3D template matching. In *Int. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 791–796, Hawai.

[Kermorgant and Chaumette, 2011] Kermorgant, O. and Chaumette, F. (2011). Combining ibvs and pbvs to ensure the visibility constraint. In *Intelligent Robots and Systems IROS'11*, pages 2849–2854. IEEE.

[Lepetit and Fua, 2005] Lepetit, V. and Fua, P. (2005). Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89.

[Lieberknecht et al., 2009] Lieberknecht, S., Benhimane, S., Meier, P., and Navab, N. (2009). A dataset and evaluation methodology for template-based tracking algorithms. In *IEEE Int. Symp on Mixed and Augmented Reality, ISMAR'09*, pages 145–151.

[Lim and Binford, 1988] Lim, H. and Binford, T. (1988). Curved surface reconstruction using stereo correspondence. In *Science Applications International Corp, Proceedings: Image Understanding Workshop,*, volume 2.

[Lowe, 1991] Lowe, D. (1991). Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(5):441–450.

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110.

[Lucas and Kanade, 1981] Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence, IJCAI'81*, pages 674–679.

[Lucas et al., 1981] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679.

[Maes et al., 1997] Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. (1997). Multimodality image registration by maximization of mutual information. *IEEE trans. on Medical Imaging*, 16(2):187–198.

[Malis, 2007] Malis, E. (2007). An efficient unified approach to direct visual tracking of rigid and deformable surfaces. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2729–2734. IEEE.

[Marchand, 1999] Marchand, E. (1999). Visp: A software environment for eye-in-hand visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'99*, volume 4, pages 3224–3229, Detroit, Michigan.

[Marchand and Chaumette, 2002] Marchand, E. and Chaumette, F. (2002). Virtual visual servoing: a framework for real-time augmented reality. In Drettakis, G. and Seidel, H.-P., editors, *EUROGRAPHICS'02 Conf. Proceeding*, volume 21(3) of *Computer Graphics Forum*, pages 289–298, Saarebrücken, Germany.

[Marchand et al., 2005] Marchand, E., Spindler, F., and Chaumette, F. (2005). ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52. Special Issue on "Software Packages for Vision-Based Control of Motion", P. Oh, D. Burschka (Eds.).

[Markovic et al., 2014] Markovic, I., Chaumette, F., Petrovic, I., et al. (2014). Moving object detection, tracking and following using an omnidirectional camera on a mobile robot. In *IEEE Int. Conf. on Robotics and Automation, ICRA'14*.

[Moore, 1920] Moore, E. (1920). Abstract. *Bulletin of the American Mathematical Society*, 26(394-395):38.

[Odobez and Bouthemy, 1995] Odobez, J.-M. and Bouthemy, P. (1995). Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4):348–365.

[Penrose, 1955] Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413.

[Petit et al., 2014] Petit, A., Marchand, E., and Kanani, A. (2014). Combining complementary edge, point and color cues in model-based tracking for highly dynamic scenes. In *IEEE Int. Conf. on Robotics and Automation, ICRA'14*, pages 4115–4120, Hong Kong, China.

[Pressigout and Marchand, 2007] Pressigout, M. and Marchand, E. (2007). Real-time hybrid tracking using edge and texture information. *Int. Journal of Robotics Research, IJRR*, 26(7):689–713.

[Richa et al., 2011] Richa, R., Sznitman, R., Taylor, R., and Hager, G. (2011). Visual tracking using the sum of conditional variance. In *IEEE Conference on Intelligent Robots and Systems, IROS'11*, pages 2953–2958, San Francisco.

[Rives et al., 1989] Rives, P., Chaumette, F., and Espiau, B. (1989). Visual servoing based on a task function approach. In *1st Int. Symp. on Experimental Robotics, ISER'89*, pages 412–428, Montreal, Canada.

[Rousseeuw, 1984] Rousseeuw, P. (1984). Least median of squares regression. *Journal American Statistic Association*, 79:871–880.

[Rueckert et al., 1999] Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: application to breast mr images. *IEEE Transactions on Medical Imaging*, 18(8):712–721.

[Ruprecht and Müller, 1993] Ruprecht, D. and Müller, H. (1993). *Free form deformation with scattered data interpolation methods*. Springer.

[Scandaroli et al., 2012] Scandaroli, G., Meilland, M., and Richa, R. (2012). Improving NCC-based direct visual tracking. In *European conference on Computer Vision*, ECCV'12, pages 442–455.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM.

[Shannon, 1948] Shannon, C. (1948). A mathematical theory of communication. *Bell system technical journal*, 27:379–423, 623–656.

[Shannon, 2001] Shannon, C. (2001). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'94*, pages 593–600, Seattle, Washington.

[Shum and Szeliski, 2000] Shum, H.-Y. and Szeliski, R. (2000). Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130.

[Silveira and Malis, 2007] Silveira, G. and Malis, E. (2007). Real-time visual tracking under arbitrary illumination changes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'07*, Minneapolis, USA.

[Studholme and Hawkes, 1999] Studholme, C. Hill, D. and Hawkes, D. (1999). An overlap invariant entropy mesure of 3d medical image alignment. *Pattern Recognition*, 32(99):71–86.

[Sundareswaran and Behringer, 1998] Sundareswaran, V. and Behringer, R. (1998). Visual servoing-based augmented reality. In *IEEE Int. Workshop on Augmented Reality*, San Francisco.

[Tsai, 1986] Tsai, R. (1986). An efficient and accurate camera calibration technique for 3D machine vision. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'86*, pages 364–374, Miami, Floride.

[Vacchetti et al., 2004] Vacchetti, L., Lepetit, V., and Fua, P. (2004). Combining edge and texture information for real-time accurate 3d camera tracking. In *ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'2004*, volume 2, pages 48–57, Arlington, Va.

[Vargas and Malis, 2005] Vargas, M. and Malis, E. (2005). Visual servoing based on an analytical homography decomposition. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 5379–5384. IEEE.

[Vincze, 2001] Vincze, M. (2001). Robust tracking of ellipses at frame rate. *Pattern Recognition*, 34(2):487 – 498.

[Viola and Wells, 1997] Viola, P. and Wells, W. (1997). Alignment by maximization of mutual information. *Int. Journal of Computer Vision*, 24(2):137–154.

[Weiss, 1984] Weiss, L. (1984). Dynamic visual servo control of robots. an adaptive image based approach. Technical Report CMU-RI-TR-84-16, Carnegie-Mellon University.

[Ying and Hu, 2004] Ying and Hu (2004). Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model ? In *European Conference on Computer Vision*, volume 1, Prague, Czech.

[Yol et al., 2014] Yol, A., Delabarre, B., Dame, A., Dartois, J.-E., and Marchand, E. (2014). Vision-based absolute localization for unmanned aerial vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'14*, pages 3429–3434, Chicago, IL.

[Zhu, 2007] Zhu, Y.-M. (2007). Mutual information-based registration of temporal and stereo retinal images using constrained optimization. *Comput. Methods Prog. Biomed.*, 86(3):210–215.

Abstract

In this document, we address the visual tracking and visual servoing problems. They are crucial thematics in the domain of computer and robot vision. Most of these techniques use geometrical primitives extracted from the images in order to estimate a motion from an image sequences. But using geometrical features means having to extract and match them at each new image before performing the tracking or servoing process. In order to get rid of this algorithmic step, recent approaches have proposed to use directly the information provided by the whole image instead of extracting geometrical primitives. Most of these algorithms, referred to as direct techniques, are based on the luminance values of every pixel in the image. But this strategy limits their use, since the criteria is very sensitive to scene perturbations such as luminosity shifts or occlusions.

To overcome this problem, we propose in this document to use robust similarity measures, the sum of conditional variance and the mutual information, in order to perform robust direct visual tracking and visual servoing processes. Several algorithms are then proposed that are based on these criteria in order to be robust to scene perturbations. These different methods are tested and analyzed in several setups where perturbations occur which allows to demonstrate their efficiency.

Keywords: Visual tracking, visual servoing, mutual information, sum of conditional variance, computer vision.

Résumé

Dans cette thèse, nous traitons les problèmes de suivi visuel et d'asservissement visuel, qui sont des thèmes essentiels dans le domaine de la vision par ordinateur. La plupart des techniques de suivi et d'asservissement visuel présentes dans la littérature se basent sur des primitives géométriques extraites dans les images pour estimer le mouvement présent dans la séquence. Un problème inhérent à ce type de méthode est le fait de devoir extraire et mettre en correspondance des primitives à chaque nouvelle image avant de pouvoir estimer un déplacement. Afin d'éviter cette couche algorithmique et de considérer plus d'information visuelle, de récentes approches ont proposé d'utiliser directement la totalité des informations fournies par l'image. Ces algorithmes, alors qualifiés de directs, se basent pour la plupart sur l'observation des intensités lumineuses de chaque pixel de l'image. Mais ceci a pour effet de limiter le domaine d'utilisation de ces approches, car ce critère de comparaison est très sensibles aux perturbations de la scène (telles que les variations de luminosité ou les occultations).

Pour régler ces problèmes nous proposons de nous baser sur des travaux récents qui ont montré que des mesures de similarité comme la somme des variances conditionnelles ou l'information mutuelle permettaient d'accroître la robustesse des approches directes dans des conditions perturbées. Nous proposons alors plusieurs algorithmes de suivi et d'asservissement visuels directs qui utilisent ces fonctions de similarité afin d'estimer le mouvement présents dans des séquences d'images et de contrôler un robot grâce aux informations fournies par une caméra. Ces différentes méthodes sont alors validées et analysées dans différentes conditions qui viennent démontrer leur efficacité.

Mots-clefs: Suivi visuel, asservissement visuel, information mutuelle, somme des variances conditionnelles, vision par ordinateur.