Visual Servoing Using Trifocal Tensor

Marwan Osman

Supervised by: François Chaumette
INRIA Rennes Bretagne Atlantique

A Thesis Submitted for the Degree of
MSc Computer Vision (MsCV)

· 2015 ·

Abstract

Visual servoing is an approach for controlling the motion of a robotic system from visual measurements. Many works have been realized in the past in this area, and is mainly divided into two main categories: Image-based, and Pose-based Visual Servoing. The trifocal tensor is well known in computer vision for tracing geometric information from three images of the same scene. The purpose of this thesis is to design a visual servoing method of a 6-DOF manipulator or robot based on the three-view projective geometry properties. Few studies where conducted on this work but they didn't provide a generic analytical solution for 6-DOF robots. This method differs than the two main visual servoing approaches as the control loop is closed over projective measures, which are the trifocal tensor elements. These projective measures are found directly from images across three views, without explicitly recovering the camera pose or directly closing the loop in the image space. The trifocal tensor geometric model is more robust than the two view geometry models as it involves the information given by a third view, and the set of correspondences obtained is more robust to outliers.

Nothing of me is original. I am the combined effort of everybody I've ever known. . . .

Chuck Palahniuk

Acknowledgments

Many thanks to all my professors and colleagues in the VIBOT/MsCV family for giving me the opportunity of joining this family. To my supervisor Prof. Chaumette for his patience and guidance through all the past months working on this thesis. To my best friends for their support and keeping me virtually at home. Finally, special thanks to my family and parents for their unconditional and endless support to help me achieve who I am today.

To my father, Ahmed, for teaching me to think critically,
and to my mother, Eman, for teaching me to love unconditionally.

Contents

List of Figures

Chapter 1

Introduction

Since the early days of the robotics field, the idea of using vision for robot guidance has been a major research area. Due to major advances in the computer vision domain, and advances in manufacturing low cost vision-based sensors, the usage of vision-based sensors for robot guidance and manipulation has been considerably studied in the literature, and often preferred over other types of sensors due to their high cost and complexity. These vision sensors allow the robot to navigate autonomously in partially or totally unknown environments. An image of the current scene of the environment can be taken, processed, analyzed, and then decide actions for the robot in real time. This technique of using the visual sensor to control the motion of a robot is formally known as **Visual Servoing**.

1.1 Problem Definition

Designing a robust and accurate visual servoing method is still a growing field of research. Because they rely on techniques from the computer vision and image processing fields, visual servoing methods inherit also their problems such as dealing with occlusions, illumination and colors differences, recovering geometrical features, detecting and matching feature points.

Some works have suggested processing the acquired image through a geometric model. Whether to recover fully or partially the 3D pose of the scene and use that information in the visual servoing feedback loop. An early work based on the epipolar geometry, where the image information relies on the epipoles. This work and similar relied on the two-views geometry, stereo vision systems, for estimating the fundamental matrix and acquire the needed geometric information

about the scene or the environment.

More recent works have proposed using three-views geometry methods and estimate the trifocal tensor, which encapsulates the intrinsic geometry between the three views. It is analogous for the fundamental matrix in stereo vision systems. However these works didn't propose a generic analytical visual servoing method using the trifocal tensor. Either they presented an analytical 3-DOF special case of using the trifocal tensor with non-holonomic planar robots, or they presented a 6-DOF robot but with numerical estimation of the visual servoing interaction matrix.

1.2 Objectives

The purpose of this thesis is to design a visual servoing method for a 6-DOF system based on the trifocal tensor geometric properties. The main objective required is ensure that embedding the trifocal tensor into the visual servoing loop would still allow the servoing system to converge to the desired goal state. This can be achieved by:

- Researching the state of the art literature on using the trifocal tensor in visual servoing based systems.
- Developing an analytical form to embed the trifocal tensor into the visual servoing loop.
- Implementing the method and validate the results in simulations.
- Experimenting the method on different robot configurations.

1.3 Document Organization

Chapter 2 begins by introducing the basic foundations and concepts that support the rest of this work: Trifocal Geometry, and Visual Servoing. As well as presenting previous works achieved combining these two topics together. Next, Chapter 3 presents the analytical development of the new method and the proposed algorithm to be implemented. The obtained results are discussed in details in Chapter 4. Finally, Chapter 5 presents an overall conclusion for the work in hand, as well as possible improvements and future work.

Chapter 2

Background

In this chapter, we review the basics and concepts for visual servoing and the three-view geometry. We present the Trifocal Tensor definition, how to recover 3D pose from the tensor, and how to compute the trifocal tensor from points correspondences across different views images. Also we present the basic framework for visual servoing control systems, and the main approaches used in the field. Based on these basics we can develop the needed method combining the mathematics from both sides. We also review the previous works that tackled using the three-view geometry into the visual servoing control loop system.

2.1 Trifocal Tensor

2.1.1 Tensor Notation

Tensors are geometric objects used to represent linear relations between vectors, scalars, and other tensors [1]. A tensor can be represented as a multi-dimensional array of numerical values. The order of a tensor is the dimensionality of the array needed to represent it. Scalars are single numbers and are thus 0th-order tensors. Vectors are 1-dimensional array, 1st-order tensors arranged in a column or row. Matrices are 2-dimensional arrays, 2nd-order tensors arranged as a 2D array of numbers. Similarly, a tensor with three indices may be thought as a 3D array of numbers.

Tensors provide a natural and concise mathematical framework for formulating and solving problems in areas of physics. Tensors express the relationship between vectors, hence they are

independent of a particular choice of coordinate system.

The notation for a tensor is similar to that of a matrix, except that a tensor may have an arbitrary number of indices *e.g.:*$A_{ijk\cdots}$. In addition, a tensor with rank $r+s$ may be of mixed type (r, s), consisting of r `contravariant (upper)` indices and s `covariant (lower)` indices. In tensor notation, a vector v would be written v_i, where $i = 1, \cdots, m$, and a matrix is a tensor of type $(1, 1)$ would be written as A^j_i.

Tensor notation can provide a very concise way of writing vector and more general identities. For example, the dot product $u.v$ can be simply written as

$$u.v = u_i v^i$$

where repeated indices are summed over. This is called `Einstein Summation` [2]. It is a notational convention for simplifying expressions including summations of vectors, matrices and general tensors. The convention can be best illustrated through the following equation

$$c^i_k = a^i_j b^j_k = \sum_j a_{ij} b_{jk}$$

Similarly, the cross product can be concisely written as

$$(u \times v)_i = \epsilon_{ijk} u^j v^k,$$

where ϵ_{ijk} is the permutation tensor defined for $r, s, t = 1, \cdots, 3$ as follows:

$$\epsilon_{rst} = \begin{cases} 0 & \text{unless } r, s \text{ and } t \text{ are distinct} \\ +1 & \text{if } rst \text{ is an even permutation of } 123 \\ -1 & \text{if } rst \text{ is an odd permutation of } 123 \end{cases}$$

2.1.2 Three-View Geometry

The Trifocal Tensor is a $3 \times 3 \times 3$ array of numbers that incorporates all projective geometric relationships among three views [3]. It relates the coordinates of corresponding points or lines in three views, being independent of the scene structure and depending only on the relative motion among the three views and their intrinsic calibration parameters. Hence, the trifocal tensor can be considered as the generalization of the fundamental matrix in three views. It can also be seen as a collection of three rank-two 3×3 matrices T_1, T_2, T_3.

The geometric basis for the trifocal tensor can be deduced from the incidence relationship of three corresponding lines. We start by supposing a line 3-space is imaged in three views as in Figure.2.1.

The planes back-projected from the lines in each view must all meet in a single line in space, the 3D line that projects to the mated line in the three images. Since in general three arbitrary planes in space do not meet in a single line, this geometric incidence condition provides a genuine constraint on sets of corresponding lines.

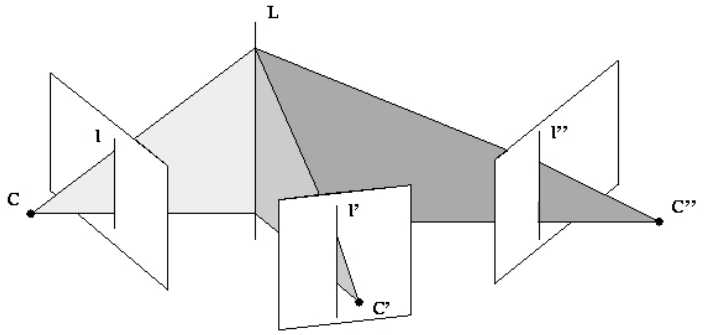


Figure 2.1: Trifocal geometry of three views

Let $l_i \leftrightarrow l'_i \leftrightarrow l''_i$ be the set of corresponding lines of L. Let the camera matrices for the three views be $P = \begin{bmatrix} I \mid 0 \end{bmatrix}$, $P' = \begin{bmatrix} A \mid a_4 \end{bmatrix}$, $P'' = \begin{bmatrix} B \mid b_4 \end{bmatrix}$, where A and B are 3×3 matrices, and the vectors a_i and b_i are the $\mathtt{i}$-th columns of the respective camera matrices for $i = 1, \cdots, 4$. A and B are the homographies from the first to the second and third cameras respectively. Due to our choice for the first camera matrix, a_4 and b_4 are consequently the epipoles of the first camera in views two and three respectively.

$$a_4 = e' = P'C \qquad\qquad b_4 = e'' = P''C$$

Each image line back-projects to a plane as shown in Figure.2.1. These planes are

$$\pi = P^T l = \begin{pmatrix} l \\ 0 \end{pmatrix} \qquad \pi' = P'^T l' = \begin{pmatrix} A^T l' \\ a_4^T l' \end{pmatrix} \qquad \pi'' = P''^T l'' = \begin{pmatrix} B^T l'' \\ b_4^T l'' \end{pmatrix}$$

The intersection constraint of the three planes in the common line in 3-space can be expressed algebraically by the requirement that the 4×3 matrix $M = \begin{bmatrix} \pi & \pi' & \pi'' \end{bmatrix}$ has rank 2. Points on the line of intersection may be represented as $X = \alpha X_1 + \beta X_2$, with X_1 and X_2 linearly

independent. Such points lie on all three planes and so $\pi^T X = \pi'^T X = \pi''^T X = 0$. It follows that $M^T X = 0$. Consequently M has a 2-dimensional null-space since $M^T X_1 = 0$ and $M^T X_2 = 0$.

Since the rank of M is 2, there is a linear dependence between its columns m_i, such that.

$$M = \begin{bmatrix} m_1, m_2, m_3 \end{bmatrix} = \begin{bmatrix} l & A^T l' & B^T l'' \\ 0 & a_4^T l' & b_4^T l'' \end{bmatrix}$$
$$m_1 = \alpha m_2 + \beta m_3$$

From the 2nd row of M, $\alpha = k(b_4^T l'')$ and $\beta = -k(a_4^T l')$ for some scalar k. Applying this back to the 1st row get

$$l = (b_4^T l'')A^T l' - (a_4^T l')B^T l''$$
$$l = (l''^T b_4)A^T l' - (l'^T a_4)B^T l''$$

The `i`-th coordinate l_i of l may be written as

$$l_i = l''^T (b_4 a_i^T) l' - l'^T (a_4 b_i^T) l''$$
$$l_i = l'^T (a_i b_4^T) l'' - l'^T (a_4 b_i^T) l''$$

This relationship can be expressed with the permutation tensor such as

$$\mathcal{T}_i = a_i b_4^T - a_4 b_i^T \tag{2.1}$$
$$l_i = l'^T \mathcal{T}_i l'' \tag{2.2}$$

$\mathcal{T}$ is then the trifocal tensor relating the 3 views together. It has 27 elements. There are 26 independent rations apart from the common overall scaling factor of the tensor. However, the tensor has only 18 independent degrees of freedom. Each of 3 camera matrices has 11 degrees of freedom which makes 33 in total. However, 15 degrees of freedom must be subtracted to account for the projective world frame, thus leaving 18 degrees of freedom.

A point x on the line l must satisfy $x^T l = \sum_i x^i l_i = 0$. From (2.2), this may be written as $l'^T (\sum_i x^i \mathcal{T}_i) l'' = 0$ which means that there exists a 3D point X mapping to x in the first image, and to points on the lines l' and l'' in the second and third images. It's the incidence relationship that holds point-line-line correspondence.

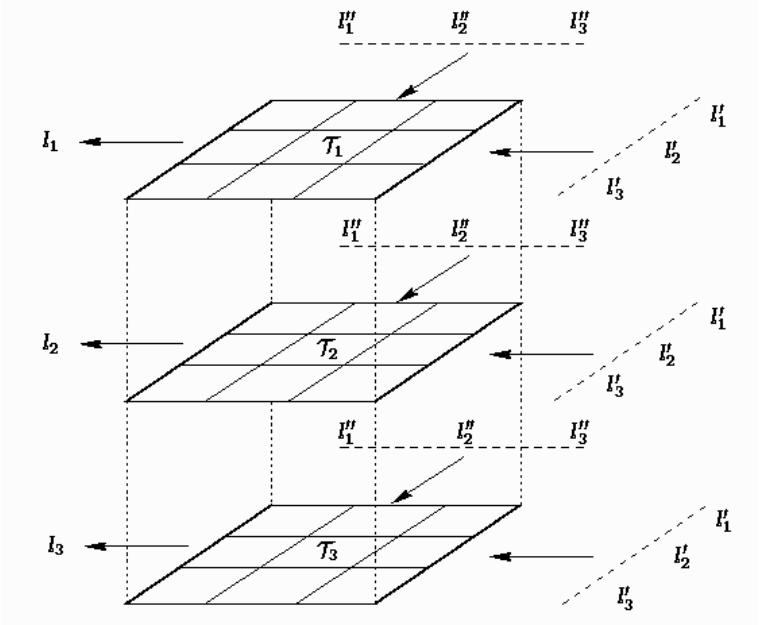


Figure 2.2: A 3D representation of the trifocal tensor $l_i = l'_j l''_k \mathcal{T}_i^{jk}$

To develop the point-point-point correspondence relationship, we can explore the homography map between the points on first and third images given by $x'' = Hx$ and $l = H^T l''$. Thus we get

$$h_i = \mathcal{T}_i^T l'$$

Similarly, the homography from the first to the second views

$$h_i = \mathcal{T}_i l''$$

Using these results back to our point-line-line correspondence relation,

$$x'' = (\sum_i x^i \mathcal{T}_i^T) l' x''^T [x'']_\times = l'^T (\sum_i x^i \mathcal{T}_i)[x'']_\times = 0^T$$

The line l' passes through x' may be written as $l' = [x']_\times y'$ for some point y' on l'. Then

$$y'^T [x']_{times} (\sum_i x^i \mathcal{T}_i)[x'']_\times = 0^T$$

This relation holds true for all lines l' through x' and so is independent of y', hence this relation can be written as

$$[x']_{times}(\sum_i x^i \mathcal{T}_i)[x'']_\times = 0_{3\times 3}$$

which expresses the point-point-point coincidence relationship required. Expressing this relation in proper tensor notation is then given by

$$x^i(x'^j \epsilon_{jpr})(x''^k \epsilon_{kqs})\mathcal{T}_i^{pq} = 0_{rs} \tag{2.3}$$

2.1.3 Recovering Projection Matrices

Since the trifocal tensor embeds the geometry of the three cameras in our scene, this implies that the camera matrices may be computed from the trifocal tensor up to a projective ambiguity [3].

First, the epipoles e', e'' are retrieved. Let u_i and v_i be the left and right null-vectors respectively of $\mathcal{T}_i$, *i.e.:* $u_i^T \mathcal{T}_i = \mathbf{0}^T, \mathcal{T}_i v_i = 0$. Then the epipoles are obtained as the null vectors to the following 3×3 matrices:

$$e'^T[u_1, u_2, u_3] = 0 \text{ and } e''^T[v_1, v_2, v_3] = 0 \tag{2.4}$$

Next, to retrieve the camera matrices P', P''. The epipoles are normalized to unit norm, then:

$$P' = [[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]e''|e'] \text{ and } P'' = [(e''e''^T - I)[\mathcal{T}_1^T, \mathcal{T}_2^T, \mathcal{T}_3^T]e'|e''] \tag{2.5}$$

2.1.4 Estimating Trifocal Tensor

The trifocal tensor can be computed from feature correspondences across the three views. Computation of tensor using correspondences of points or lines has been studied extensively in literature. The main linear algorithm presented by `Hartley` [3] is the straightforward method to compute the tensor. Other methods make use of `RANSAC` (RANdom SAmple Consensus) to enable the tensor estimation to be robust to outliers that usually cause the linear algebraic method to fail [4].

Each triplet of corresponding image points gives 4 equations linearly independent. Therefore, a minimum set of 7 correspondences of points are needed for the trifocal tensor computation to uniquely determine the 27 entries of the tensor matrix. Given n point correspondences, let

A be a matrix of size $4n \times 27$ and t a vector containing all entires of the tensor, then we have

$$At = 0 \tag{2.6}$$

By applying the least square method or SVD, the tensor is obtained from the solution to this linear system. However, `Hartley` has shown that this kind of algorithm does not do well if all the points are of the form $(x_1, x_2, 1)$ in homogeneous coordinates with x_1 and x_2 very much larger than 1. Therefore, it is necessary to normalize the points of each image separately before computing the tensor. The points are translated in each image so that the centroid of all measured points is at the origin of the image coordinates, and then scaled so that the average distance of a point from the origin is $\sqrt{2}$ units. This way the average point will be something like $(1, 1, 1)$ in homogeneous coordinates, and each of the homogeneous coordinate will be approximately of equal weight. This transformation improves the condition of the matrix of equations and leads to a better solution. Consequently, a de-normalization step is needed in order to work with the original image coordinates.

This method still suffers from two issues

1. The tensor is parameterized by all its entries and this does not take into account the tensor geometrical constraints. Therefore, there is always a risk of obtaining an invalid tensor.

2. Using all available point correspondences causes the tensor to be significantly affected by the presence of strong outliers.

Hence, an algebraic minimization is needed there after, to use the linear solution as an initial estimate and re-parameterize it by the 24 entries of the projection matrices P' and P''. The desired tensor is found by minimizing the residual error. The overall procedure is briefly explained here.

1. Normalize the set of point triplets by performing transformations H, H' and H''. Here $\hat{x} = Hx$, $\hat{x}' = H'x'$, $\hat{x}'' = H''x''$.

2. Compute the tensor linearly by solving a set of equations of the form $At = 0$, where A expresses 2.3, and t is the vector of entries of the tensor.

3. De-normalize the tensor by $T_i = H'^{-1} \sum_j (H^T(i, j)T_j)H''^{-T}$

4. Find the two epipoles e' and e'' from the tensor using 2.4.

5. According to Equation 2.1, construct the 28×12 matrix E such that $t = Ea$, where a is the vector representing entries of a_i and b_i.

6. Compute the tensor by minimizing the algebraic error AEa subject to $Ea = 1$.

The above algorithm leads to a geometrically valid tensor because its entries satisfies the equation 2.1. However, the main weakness of the algebraic minimization method is that all point correspondences are involved in the tensor computation and still suffers from outliers in practice.

2.2 Visual Servoing

2.2.1 Basic Framework of Visual Servoing

Visual Servoing refers to the family of closed loop control techniques to control the degrees of freedom of an actuated system with visual feedback [5]. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or on a mobile robot, or the camera can be fixed in the workspace so that it can observe the robot motion from a stationary configuration. Visual Servoing relies on techniques from image processing, computer vision, and control theory.

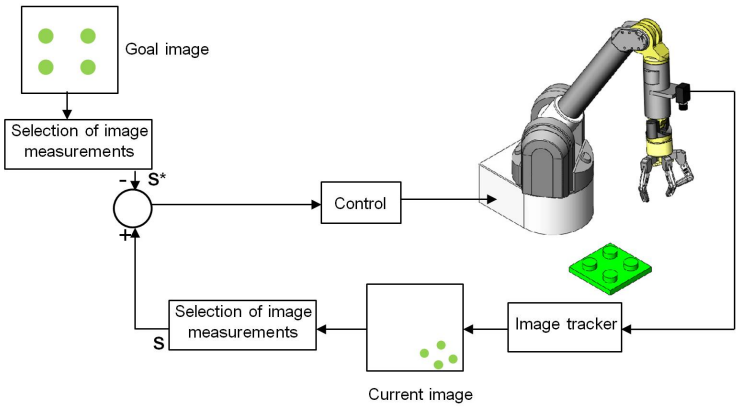


Figure 2.3: Block diagram of the basic Visual Servoing loop control

The goal of many robotic applications is to place the robot at a desired configuration to manipulate an object in the environment. The objective of closed loop control techniques is to minimize an error $e(t)$ defined as

$$e(t) = s(t) - s^* \tag{2.7}$$

where s is a vector of visual features, and the vector s^* has the desired values of these features at the desired configuration.

Computer vision algorithms are used for the tracking of visual features on the object. There are three main classes of visual servoing: Image-based Visual Servoing (IBVS), Position-based Visual Servoing (PBVS), and Hybrid Visual Servoing (HVS) [5] [6]. In IBVS, the features are computed from the 2D image data, while in PBVS, the features are computed from the 3D estimated pose from image measurements. HVS use a combination of 2D and 3D visual features.

Then, to design the control scheme, the relationship between the time variation of the features s and the camera velocity has to be found. Noting the spatial velocity of the camera $u_c = (v_c, \omega_c)$, where v_c is the instantaneous linear velocity of the origin of the camera frame, and ω_c is the instantaneous angular velocity of the camera frame. The relationship between $\dot{s}$ and u_c is given by

$$\dot{s} = L_s u_c \tag{2.8}$$

where L_s is called the **interaction matrix** related to s. This matrix relates the rate of changes of the image feature velocities $\dot{s}$ to the rate of change of the pose parameters which is the camera velocity. From 2.7 and 2.8, the relationship between the camera velocity and the derivative of the error is

$$\dot{e} = L_s u_c \tag{2.9}$$

Considering u_c as the input to the robot controller, to ensure an exponential decoupled decrease of the error, we would like to have $\dot{e} = -\lambda e$, using 2.9 we obtain the control law equation

$$u_c = -\lambda L_s^+ e \tag{2.10}$$

where L_s^+ is the pseudo-inverse of L_s. This is the basic visual servoing closed loop system framework. Different visual servoing approaches differ only in the selection of the visual features s and deriving their corresponding interaction matrices L_s.

The choice of the visual features is important and generally affect the performance of the visual servoing system.

2.2.2 Image-Based Visual Servoing (IBVS)

In the IBVS approach, the visual features are defined directly in the 2D image space. This is done without the explicit calculation of the relative camera frames at current and desired

configurations.

Considering a 3D point $[XYZ]^T$ expressed in the camera frame and its projection onto the image plane $[xy]^T$. If we take the image coordinates in the image plane of the 3D point as visual features, then the corresponding interaction matrix L_s has the following form

$$L_s = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.11}$$

The value Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of Z. Similarly, the camera intrinsic parameters are involved in the computation of x and y. Thus L_s^+ cannot be directly used and an estimation or an approximation $\widehat{L}_s^+$ must be used in practice.

The IBVS approach is more robust to modeling and measurement noise than PBVS. However, the IBVS approach only guarantees local asymptotic stability with a local control law theoretically. It also suffers from degenerate cases and fails to reach the goal state.

2.2.3 Position-Based Visual Servoing (PBVS)

In PBVS, the visual features are expressed in terms of the 3D rigid-body transformation from the initial to the desired states, with respect to some reference coordinate frame. The camera acts as a 3D sensor.

First we define three coordinate frame: the current camera frame F_c, the desired camera frame F_{c^*} and a reference frame F_o attached to the object. The standard notation is to use a leading superscript to denote the frame with respect to which a set of coordinates is defined.

s can be defined to be $(t, \theta u)$, where t is a translation vector, and θu is the angle parameterization for the rotation. If t is defined relative to the object frame F_o, then

$$s = ({}^c t_o, \theta u), s^* = ({}^{c^*} t_o, 0), e = ({}^c t_o - {}^{c^*} t_o, \theta u) \tag{2.12}$$

And the corresponding interaction matrix is given by

$$Ls = \begin{bmatrix} -\mathbf{I_3} & [{}^c t_o]_\times \\ \mathbf{0} & L_{\theta u} \end{bmatrix} \tag{2.13}$$

where I_3 is the 3×3 identity matrix and $L_{\theta u}$ is given by

$$L_{\theta u} = I_3 - \frac{\theta}{2}[u]_\times + \left(1 - \frac{sinc\theta}{sinc^2\frac{\theta}{2}}\right)[u]_\times^2 \tag{2.14}$$

If the pose parameters are perfectly estimated, the choice of e causes the rotational motion to follow a geodesic with an exponential decreasing speed and causes the translational parameters involved in s to decrease with the same speed. The trajectory in the image of the origin of the object frame follows a pure straight line. However, the camera trajectory does not follow a straight line.

Another PBVS scheme can be designed by choosing $s = ({}^{c^*}t_c, \theta u)$, which lead to having $s^* = \mathbf{0}, \mathbf{e} = \mathbf{s}$, and the following interaction matrix

$$L_s = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & L_{\theta u} \end{bmatrix} \tag{2.15}$$

This definition introduces a decoupling between translational and rotational motions, which allow a simpler control scheme. The resulting camera trajectory is then a pure straight line.

PBVS approach ensure global asymptotic stability if the pose estimation or tracking is perfect. The pose estimation algorithm must be realized in real-time to calculate the error at every control iteration, and hence more computationally expensive than IBVS approach. However, PBVS approach is more sensitive to image measurement noise and outliers. It also requires full knowledge of the geometric object model, the camera intrinsic calibration, and the robot kinematic calibration.

2.3 Previous Works

There exist some approaches which cannot be considered part of the main two classes of Visual Servoing approaches IBVS and PBVS. These approaches differ than the two main visual servoing classes by closing the control loop over projective measures, which are not computed directly from the image space or the 3D pose.

Epipolar geometry can be used to estimate depth, which appears in the interaction matrix of point features [7]. Chesi *et al.* controlled a holonomic mobile robot from a partially calibrated camera using the symmetry of epipolar geometry without point correspondences [8]. Benhimane and Malis developed a homography-based approach without reconstructing any 3D parameters [9]. Lopez-Nicolas *et al.* designed a homography-based controller which considers the non-holonomic constraints [10]. These methods use the two-view geometry between the observed and desired views and ignore their relation with the initial view. Epipolar geometry is not well-conditioned if the features are coplanar or the baseline is short, while homography-based approaches require dominant planes.

Recent works introduced using the three-views geometry in the visual servoing loop. The trifocal tensor encapsulates the intrinsic geometry between the three views, and It is analogous for the fundamental matrix in stereo vision systems. The application of trifocal tensor in visual servoing has been neglected until recently. Becerra and Sagues used a simplified trifocal tensor as measurement and estimate and track the pose of a non-holonomic mobile robot with Extended Kalman Filter (EKF) [11]. Lopez-Nicolas *et al.* used the constrained camera motion on a mobile robot and linearized the input-output space for control [12]. This approach provided an analytical interaction matrix, which relates the variations of 9 elements of the trifocal tensor to the motion velocities. Shademan used the trifocal tensor for 6-DOF visual servoing [13]. Unlike Lopez-Nicolas, Shademan didn't provide an analytical derivation for the interaction matrix, it was estimated numerically and used in the control law. Shademan argued that using the trifocal tensor to control 6-DOF visual servoing loop was not introduced prior to his work due to the difficulty in linearizing the input-output space in the case of the generalized 6-DOF camera motions.

The work of this thesis is based on the previous approaches introduced by Lopez-Nicolas [12] and Shademan [13]. The main contribution of this thesis is to design a generalized 6-DOF visual servoing approach based on the trifocal tensor elements as visual features. The trifocal tensor geometric model is more robust than the two view geometry models as it involves the information given by a third view, and the set of correspondences obtained is more robust to outliers.

Chapter 3

Methodology

In this chapter, we present the development of a visual servoing method based on using the trifocal tensor elements as visual features in our visual servoing control loop. We begin by developing the needed control law using the tensor, finding the respective interaction matrix, and then exploring the normalization approach for the obtained tensor and interaction matrix. Finally we represent the overall procedure for the proposed algorithm to be implemented.

3.1 Control Law

To formulate the control law of the trifocal tensor based visual servoing (3.1), we need to establish the model of the system and find the relation between the input of the control and the trifocal tensor coefficients. First, we need to find the interaction matrix relating the control input and the tensor derivatives. The error will be the difference between the current tensor and the desired tensor values.

$$\begin{aligned} \dot{\mathcal{T}}_{(jkl)} &= L_{\mathcal{T}_{(jkl)}} \begin{pmatrix} v_c \\ w_c \end{pmatrix} \\ u_c &= -\lambda L^{+}_{\mathcal{T}_{(jkl)}} e \end{aligned} \tag{3.1}$$

3.1.1 Coefficients of the Trifocal Tensor

To compute the relation between the coefficients of the trifocal tensor and the Projective matrices, first we write the Tensor relation (2.1) with the proper visual servoing notation as follows

$$\mathcal{T}_{(jkl)} = a_{(kj)}b_{(4l)} - a_{(4k)}b_{(lj)} \tag{3.2}$$

For the sake of maintaining the Visual Servoing convention, we omit the use of i as an index for the tensor relation. The letter i further on will be indicating the initial view in the control system. The Camera positions are then written as C_{c^*}, C_c, C_i for the desired, current, and initial camera positions respectively. The same may also be done for the projection matrices, and instead of using the matrices A, B to express the homography between the first and second or third views, we will use the standard notation from Visual Servoing with the corresponding Rotation matrices ${}^cR_{c^*}, {}^iR_{c^*}$ respectively. For the translations a_4, b_4, we will use ${}^ct_{c^*}, {}^it_{c^*}$ respectively. Leading superscripts will denote the frame with respect to which a set of coordinates are defined. Thus, the rotation matrix ${}^cR_{c^*}$ gives the orientation of the desired camera frame relative to the current camera frame.

$$C \Rightarrow C_{c^*} \text{ Desired camera position}$$
$$C' \Rightarrow C_c \text{ Current camera position}$$
$$C'' \Rightarrow C_i \text{ Initial camera position}$$

$$P_d = P = [I|0]$$
$$P_c = P' = [A|a_4] :^c \mathbf{M}_{c^*} = [{}^c\mathbf{R}_{c^*}|{}^c\mathbf{t}_{c^*}]$$
$$P_i = P'' = [B|b_4] :^i \mathbf{M}_{c^*} = [{}^i\mathbf{R}_{c^*}|{}^i\mathbf{t}_{c^*}]$$

The Tensor relation (3.2) can then be expressed as follows

$$\mathcal{T}_{(jkl)} = {}^cR_{c^*(kj)}\ {}^it_{c^*(l)} - {}^ct_{c^*(k)}\ {}^iR_{c^*(lj)} \tag{3.3}$$

Expanding this relation further to compute the tensor coefficients is straight forward, expanding for values of j, k, l we get the following tensor coefficients:

$$
\begin{aligned}
\mathcal{T}_{(111)} &= {}^{c}R_{c^*(11)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(11)} \\
\mathcal{T}_{(112)} &= {}^{c}R_{c^*(11)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(21)} \\
\mathcal{T}_{(113)} &= {}^{c}R_{c^*(11)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(31)} \\
\mathcal{T}_{(121)} &= {}^{c}R_{c^*(21)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(11)} \\
\mathcal{T}_{(122)} &= {}^{c}R_{c^*(21)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(21)} \\
\mathcal{T}_{(123)} &= {}^{c}R_{c^*(21)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(31)} \\
\mathcal{T}_{(131)} &= {}^{c}R_{c^*(31)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(11)} \\
\mathcal{T}_{(132)} &= {}^{c}R_{c^*(31)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(21)} \\
\mathcal{T}_{(133)} &= {}^{c}R_{c^*(31)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(31)} \\
\mathcal{T}_{(211)} &= {}^{c}R_{c^*(12)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(12)} \\
\mathcal{T}_{(212)} &= {}^{c}R_{c^*(12)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(22)} \\
\mathcal{T}_{(213)} &= {}^{c}R_{c^*(12)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(32)} \\
\mathcal{T}_{(221)} &= {}^{c}R_{c^*(22)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(12)} \\
\mathcal{T}_{(222)} &= {}^{c}R_{c^*(22)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(22)} \\
\mathcal{T}_{(223)} &= {}^{c}R_{c^*(22)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(32)} \\
\mathcal{T}_{(231)} &= {}^{c}R_{c^*(32)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(12)} \\
\mathcal{T}_{(232)} &= {}^{c}R_{c^*(32)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(22)} \\
\mathcal{T}_{(233)} &= {}^{c}R_{c^*(32)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(32)} \\
\mathcal{T}_{(311)} &= {}^{c}R_{c^*(13)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(13)} \\
\mathcal{T}_{(312)} &= {}^{c}R_{c^*(13)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(23)} \\
\mathcal{T}_{(313)} &= {}^{c}R_{c^*(13)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(1)}\ {}^{i}R_{c^*(33)} \\
\mathcal{T}_{(321)} &= {}^{c}R_{c^*(23)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(13)} \\
\mathcal{T}_{(322)} &= {}^{c}R_{c^*(23)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(23)} \\
\mathcal{T}_{(323)} &= {}^{c}R_{c^*(23)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(2)}\ {}^{i}R_{c^*(33)} \\
\mathcal{T}_{(331)} &= {}^{c}R_{c^*(33)}\ {}^{i}t_{c^*(1)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(13)} \\
\mathcal{T}_{(332)} &= {}^{c}R_{c^*(33)}\ {}^{i}t_{c^*(2)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(23)} \\
\mathcal{T}_{(333)} &= {}^{c}R_{c^*(33)}\ {}^{i}t_{c^*(3)} - {}^{c}t_{c^*(3)}\ {}^{i}R_{c^*(33)}
\end{aligned}
\tag{3.4}
$$

Our objective is to move our robot from the initial camera position to the desired target camera position. Therefore, we have two special cases for the tensor coefficients values.

First when C_c at the initial position C_i

$$\begin{gathered}
{}^{c}R_{c^*} = {}^{i}R_{c^*},\ {}^{c}t_{c^*} = {}^{i}t_{c^*} \\
\mathcal{T}^{i}_{(jkl)} = {}^{c}R_{c^*(kj)}\ {}^{c}t_{c^*(l)} - {}^{c}t_{c^*(k)}\ {}^{c}R_{c^*(lj)} \\
\text{when } k = l, \mathcal{T}^{i}_{(jkl)} = 0 \\
\mathcal{T}^{i}_{(112)} = -\mathcal{T}^{i}_{(121)}, \mathcal{T}^{i}_{(113)} = -\mathcal{T}^{i}_{(131)}, \mathcal{T}^{i}_{(123)} = -\mathcal{T}^{i}_{(132)} \\
\mathcal{T}^{i}_{(212)} = -\mathcal{T}^{i}_{(221)}, \mathcal{T}^{i}_{(213)} = -\mathcal{T}^{i}_{(231)}, \mathcal{T}^{i}_{(223)} = -\mathcal{T}^{i}_{(232)} \\
\mathcal{T}^{i}_{(312)} = -\mathcal{T}^{i}_{(321)}, \mathcal{T}^{i}_{(313)} = -\mathcal{T}^{i}_{(331)}, \mathcal{T}^{i}_{(323)} = -\mathcal{T}^{i}_{(332)}
\end{gathered} \tag{3.5}$$

Second when C_c at the desired position C_{c^*}

$$\begin{gathered}
{}^{c}R_{c^*} = I,\ {}^{c}t_{c^*} = \mathbf{0} \\
\mathcal{T}^{*}_{(jkl)} = I_{(kj)}\ {}^{i}t_{c^*(l)} \\
\text{when } j \neq k, \mathcal{T}^{*}_{(jkl)} = 0 \\
\mathcal{T}^{*}_{(111)} = \mathcal{T}^{*}_{(221)} = \mathcal{T}^{*}_{(331)} = {}^{i}t_{c^*(1)} \\
\mathcal{T}^{*}_{(112)} = \mathcal{T}^{*}_{(222)} = \mathcal{T}^{*}_{(332)} = {}^{i}t_{c^*(2)} \\
\mathcal{T}^{*}_{(113)} = \mathcal{T}^{*}_{(223)} = \mathcal{T}^{*}_{(333)} = {}^{i}t_{c^*(3)}
\end{gathered} \tag{3.6}$$

3.1.2 Tensor Derivation

First, The derivatives of all the trifocal tensor elements with respect to time are generally as following:

$$\dot{\mathcal{T}}_{(jkl)} = {}^{c}\dot{R}_{c^*(kj)}\ {}^{i}t_{c^*(l)} + {}^{c}R_{c^*(kj)}\ {}^{i}\dot{t}_{c^*(l)} - {}^{c}\dot{t}_{c^*(k)}\ {}^{i}R_{c^*(lj)} - {}^{c}t_{c^*(k)}\ {}^{i}\dot{R}_{c^*(lj)} \tag{3.7}$$

Since our initial camera C_i is fixed, the elements of the derivatives of its rotation matrix and its transpose vector are equal to zero, *i.e.:* ${}^{i}\dot{t}_{c^*(l)} = {}^{i}\dot{R}_{c^*(jl)} = 0$. Our trifocal tensor elements derivative is then simplified to:

$$\dot{\mathcal{T}}_{(jkl)} = {}^{c}\dot{R}_{c^*(kj)}\ {}^{i}t_{c^*(l)} - {}^{c}\dot{t}_{c^*(k)}\ {}^{i}R_{c^*(lj)} \tag{3.8}$$

The spatial velocity of the camera is $u_c = (v_c, \omega_c)^T$, where v_c and ω_c are the translational

and rotational velocities of the camera. From the geometry of the scene, we can deduce the following relationships:

$$
\begin{gathered}
[\omega_c]_\times = {}^{c^*}R_c{}^T\,{}^{c^*}\dot{R}_c = -{}^{c^*}\dot{R}_c{}^T\,{}^{c^*}R_c \\
{}^{c^*}\dot{R}_c{}^T = -[\omega_c]_\times\,{}^{c^*}R_c{}^T \\
{}^{c^*}\dot{R}_c{}^T = -[\omega_c]_\times\,{}^{c}R_{c^*} \\
{}^{c}\dot{R}_{c^*} = -[\omega_c]_\times\,{}^{c}R_{c^*} \\
\text{and} \\
{}^{c^*}\dot{t}_c = {}^{c^*}R_c v_c,\ {}^{c}t_{c^*} = -{}^{c}R_{c^*}\,{}^{c^*}t_c \\
{}^{c}\dot{t}_{c^*} = -{}^{c}\dot{R}_{c^*}\,{}^{c^*}t_c - {}^{c}R_{c^*}\,{}^{c^*}\dot{t}_c \\
{}^{c}\dot{t}_{c^*} = [\omega_c]_\times\,{}^{c}R_{c^*}\,{}^{c^*}t_c - {}^{c}R_{c^*}\,{}^{c^*}R_c v_c \\
{}^{c}\dot{t}_{c^*} = [\omega_c]_\times\,{}^{c}t_{c^*} - v_c \\
{}^{c}\dot{t}_{c^*} = [{}^{c}t_{c^*}]_\times\omega_c - v_c \\
{}^{c}\dot{t}_{c^*(k)} = [-I|[{}^{c}t_{c^*}]_\times]u
\end{gathered}
$$

Substituting back into the tensor derivation (3.8), we get:

$$
\begin{aligned}
\dot{\mathcal{T}}_{(jkl)} &= -([\omega_c]_\times\,{}^{c}R_{c^*})_{(kj)}\ {}^{i}t_{c^*(l)} - ([{}^{c}t_{c^*}]_\times\omega_c - v_c)_{(k)}\ {}^{i}R_{c^*(lj)} \\
\dot{\mathcal{T}}_{(jkl)} &= -(\sum_m [\omega_c]_{\times(km)}\,{}^{c}R_{c^*(mj)})\ {}^{i}t_{c^*(l)} - (([{}^{c}t_{c^*}]_\times\omega_c)_{(k)} - v_{c(k)})\ {}^{i}R_{c^*(lj)}
\end{aligned}
\tag{3.9}
$$

where

$$
{}^{i}t_{c^*} = -{}^{i}R_{c^*}\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix},\ {}^{c}t_{c^*} = -{}^{c}R_{c^*}\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix},
$$

$$
{}^{i}R_{c^*} = \begin{bmatrix} R_{i(11)} & R_{i(12)} & R_{i(13)} \\ R_{i(21)} & R_{i(22)} & R_{i(23)} \\ R_{i(31)} & R_{i(32)} & R_{i(33)} \end{bmatrix},\ {}^{c}R_{c^*} = \begin{bmatrix} R_{c(11)} & R_{c(12)} & R_{c(13)} \\ R_{c(21)} & R_{c(22)} & R_{c(23)} \\ R_{c(31)} & R_{c(32)} & R_{c(33)} \end{bmatrix},
$$

$$
[\omega_c]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix},\ [{}^{c}t_{c^*}]_\times\omega_c = \begin{bmatrix} -{}^{c}t_{c^*(3)}\omega_y + {}^{c}t_{c^*(2)}\omega_z \\ {}^{c}t_{c^*(3)}\omega_x - {}^{c}t_{c^*(1)}\omega_z \\ -{}^{c}t_{c^*(2)}\omega_x + {}^{c}t_{c^*(1)}\omega_y \end{bmatrix}
$$

From (3.4) and (3.9), the derivative of the first tensor coefficient can be computed as follows:

$$\begin{aligned}\dot{\mathcal{T}}_{(111)} &= -(-R_{c(21)}\omega_z + R_{c(31)}\omega_y)\ {}^{i}t_{c^*(1)} - (-{}^{c}t_{c^*(3)}\omega_y + {}^{c}t_{c^*(2)}\omega_z - v_x)\ {}^{i}R_{c^*(11)} \\ \dot{\mathcal{T}}_{(111)} &= R_{i(11)}v_x - (R_{c(31)}\,{}^{i}t_{c^*(1)} - R_{i(11)}\,{}^{c}t_{c^*(3)})\omega_y + (R_{c(21)}\,{}^{i}t_{c^*(1)} - R_{i(11)}\,{}^{c}t_{c^*(2)})\omega_z \\ \dot{\mathcal{T}}_{(111)} &= R_{i(11)}v_x - \mathcal{T}_{(131)}\omega_y + \mathcal{T}_{(121)}\omega_z\end{aligned} \tag{3.10}$$

By expanding the derivatives for the rest of the coefficients, we can deduce this compact form for the tensor coefficients derivatives:

$$\dot{\mathcal{T}}_{(jkl)} = R_{i(lj)}v_{c(k)} - \sum_{m} [\omega_c]_{\times(km)}\mathcal{T}_{(jml)} \tag{3.11}$$

3.1.3 Interaction Matrix

To control the six degrees of freedom, at least six tensor coefficients are necessary. However, to avoid singularities, more than six tensor coefficients are considered. The interaction matrix taking into account all the tensor coefficients can be deduced from (3.11), and is given in (3.12). It is important to notice that pose of the initial camera location is necessary for the computation of the interaction matrix as well as the normalization factor.

It is also to be noticed there exists a decoupling between the translational velocities in the resulting interaction matrix. This decoupling ensures to obtain smooth camera trajectories for the motion in the 3D space.

$$
L_{\mathcal{T}_{(jkl)}} = \begin{bmatrix}
{}^{i}R_{c^*(11)} & 0 & 0 & 0 & -\mathcal{T}_{(131)} & \mathcal{T}_{(121)} \\
{}^{i}R_{c^*(21)} & 0 & 0 & 0 & -\mathcal{T}_{(132)} & \mathcal{T}_{(122)} \\
{}^{i}R_{c^*(31)} & 0 & 0 & 0 & -\mathcal{T}_{(133)} & \mathcal{T}_{(123)} \\
0 & {}^{i}R_{c^*(11)} & 0 & \mathcal{T}_{(131)} & 0 & -\mathcal{T}_{(111)} \\
0 & {}^{i}R_{c^*(21)} & 0 & \mathcal{T}_{(132)} & 0 & -\mathcal{T}_{(112)} \\
0 & {}^{i}R_{c^*(31)} & 0 & \mathcal{T}_{(133)} & 0 & -\mathcal{T}_{(113)} \\
0 & 0 & {}^{i}R_{c^*(11)} & -\mathcal{T}_{(121)} & \mathcal{T}_{(111)} & 0 \\
0 & 0 & {}^{i}R_{c^*(21)} & -\mathcal{T}_{(122)} & \mathcal{T}_{(112)} & 0 \\
0 & 0 & {}^{i}R_{c^*(31)} & -\mathcal{T}_{(123)} & \mathcal{T}_{(113)} & 0 \\
{}^{i}R_{c^*(12)} & 0 & 0 & 0 & -\mathcal{T}_{(231)} & \mathcal{T}_{(221)} \\
{}^{i}R_{c^*(22)} & 0 & 0 & 0 & -\mathcal{T}_{(232)} & \mathcal{T}_{(222)} \\
{}^{i}R_{c^*(32)} & 0 & 0 & 0 & -\mathcal{T}_{(233)} & \mathcal{T}_{(223)} \\
0 & {}^{i}R_{c^*(12)} & 0 & \mathcal{T}_{(231)} & 0 & -\mathcal{T}_{(211)} \\
0 & {}^{i}R_{c^*(22)} & 0 & \mathcal{T}_{(232)} & 0 & -\mathcal{T}_{(212)} \\
0 & {}^{i}R_{c^*(32)} & 0 & \mathcal{T}_{(233)} & 0 & -\mathcal{T}_{(213)} \\
0 & 0 & {}^{i}R_{c^*(12)} & -\mathcal{T}_{(221)} & \mathcal{T}_{(211)} & 0 \\
0 & 0 & {}^{i}R_{c^*(22)} & -\mathcal{T}_{(222)} & \mathcal{T}_{(212)} & 0 \\
0 & 0 & {}^{i}R_{c^*(32)} & -\mathcal{T}_{(223)} & \mathcal{T}_{(213)} & 0 \\
{}^{i}R_{c^*(13)} & 0 & 0 & 0 & -\mathcal{T}_{(331)} & \mathcal{T}_{(321)} \\
{}^{i}R_{c^*(23)} & 0 & 0 & 0 & -\mathcal{T}_{(332)} & \mathcal{T}_{(322)} \\
{}^{i}R_{c^*(33)} & 0 & 0 & 0 & -\mathcal{T}_{(333)} & \mathcal{T}_{(323)} \\
0 & {}^{i}R_{c^*(13)} & 0 & \mathcal{T}_{(331)} & 0 & -\mathcal{T}_{(311)} \\
0 & {}^{i}R_{c^*(23)} & 0 & \mathcal{T}_{(332)} & 0 & -\mathcal{T}_{(312)} \\
0 & {}^{i}R_{c^*(33)} & 0 & \mathcal{T}_{(333)} & 0 & -\mathcal{T}_{(313)} \\
0 & 0 & {}^{i}R_{c^*(13)} & -\mathcal{T}_{(321)} & \mathcal{T}_{(311)} & 0 \\
0 & 0 & {}^{i}R_{c^*(23)} & -\mathcal{T}_{(322)} & \mathcal{T}_{(312)} & 0 \\
0 & 0 & {}^{i}R_{c^*(33)} & -\mathcal{T}_{(323)} & \mathcal{T}_{(313)} & 0
\end{bmatrix} \tag{3.12}
$$

When the camera reach the desired position, the resulting interaction matrix is given in(3.13). We can observe this interaction matrix only depends on the initial camera pose. This implies that the interaction matrix will never have all elements equal to zero (except for the diagonal elements of the rotation matrix ${}^{i}R_{c^*}$ which are equal to 1), except when the initial camera pose is exactly the same as the desired camera pose, which is not a valid configuration in our visual servoing framework.

$$
L_{\mathcal{T}^*_{(jkl)}} = \begin{bmatrix}
{}^{i}R_{c^*(11)} & 0 & 0 & 0 & 0 & 0 \\
{}^{i}R_{c^*(21)} & 0 & 0 & 0 & 0 & 0 \\
{}^{i}R_{c^*(31)} & 0 & 0 & 0 & 0 & 0 \\
0 & {}^{i}R_{c^*(11)} & 0 & 0 & 0 & -{}^{i}t_{c^*(1)} \\
0 & {}^{i}R_{c^*(21)} & 0 & 0 & 0 & -{}^{i}t_{c^*(2)} \\
0 & {}^{i}R_{c^*(31)} & 0 & 0 & 0 & -{}^{i}t_{c^*(3)} \\
0 & 0 & {}^{i}R_{c^*(11)} & 0 & {}^{i}t_{c^*(1)} & 0 \\
0 & 0 & {}^{i}R_{c^*(21)} & 0 & {}^{i}t_{c^*(2)} & 0 \\
0 & 0 & {}^{i}R_{c^*(31)} & 0 & {}^{i}t_{c^*(3)} & 0 \\
{}^{i}R_{c^*(12)} & 0 & 0 & 0 & 0 & {}^{i}t_{c^*(1)} \\
{}^{i}R_{c^*(22)} & 0 & 0 & 0 & 0 & {}^{i}t_{c^*(2)} \\
{}^{i}R_{c^*(32)} & 0 & 0 & 0 & 0 & {}^{i}t_{c^*(3)} \\
0 & {}^{i}R_{c^*(12)} & 0 & 0 & 0 & 0 \\
0 & {}^{i}R_{c^*(22)} & 0 & 0 & 0 & 0 \\
0 & {}^{i}R_{c^*(32)} & 0 & 0 & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(12)} & -{}^{i}t_{c^*(1)} & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(22)} & -{}^{i}t_{c^*(2)} & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(32)} & -{}^{i}t_{c^*(3)} & 0 & 0 \\
{}^{i}R_{c^*(13)} & 0 & 0 & 0 & -{}^{i}t_{c^*(1)} & 0 \\
{}^{i}R_{c^*(23)} & 0 & 0 & 0 & -{}^{i}t_{c^*(2)} & 0 \\
{}^{i}R_{c^*(33)} & 0 & 0 & 0 & -{}^{i}t_{c^*(3)} & 0 \\
0 & {}^{i}R_{c^*(13)} & 0 & {}^{i}t_{c^*(1)} & 0 & 0 \\
0 & {}^{i}R_{c^*(23)} & 0 & {}^{i}t_{c^*(2)} & 0 & 0 \\
0 & {}^{i}R_{c^*(33)} & 0 & {}^{i}t_{c^*(3)} & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(13)} & 0 & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(23)} & 0 & 0 & 0 \\
0 & 0 & {}^{i}R_{c^*(33)} & 0 & 0 & 0
\end{bmatrix} \tag{3.13}
$$

3.2 Tensor Normalization

In this approach, no metric information, depth estimation or trifocal tensor decomposition were used. Hence, we need to define a common scale of the trifocal tensor for the control law. The trifocal tensor is normalized to get a fixed scale with $\mathcal{T}_N$. When the camera reaches the desired goal position, we can observe from (3.6) that the tensor coefficients are only related to the translation of the camera at the initial position which is a constant value vector. Also from (3.12), the first column elements $\mathcal{T}_{(111)}$, $\mathcal{T}_{(112)}$, $\mathcal{T}_{(113)}$, $\mathcal{T}_{(211)}$, $\mathcal{T}_{(212)}$, $\mathcal{T}_{(213)}$, $\mathcal{T}_{(311)}$, $\mathcal{T}_{(312)}$, $\mathcal{T}_{(313)}$ are respectively equal to ${}^{i}t_{c^*(1)}$, ${}^{i}t_{c^*(2)}$, ${}^{i}t_{c^*(3)}$, 0,0,0,0,0,0 at the desired goal position.

Choosing the elements $\mathcal{T}_{(111)}$, $\mathcal{T}_{(112)}$, $\mathcal{T}_{(113)}$ as normalization factors would seem a good choice for our method, as we would be interested in coefficients converging to non-zero values. However, it was found that for some scene configurations, these three coefficients could have values equal to zero at the initial pose. Which means at normalization step, this would cause dividing by a zero value. To avoid this problem, we will use all the nine coefficients of the first column to be our first normalization factor. We can then normalize the elements T_{jkl} for $k = 1$ as follows:

$$\begin{aligned} T_{jkl} &= \frac{\mathcal{T}_{jkl}}{\mathcal{T}_{1N}} \\ \mathcal{T}_{1N} &= \sqrt{{\mathcal{T}_{111}}^2 + {\mathcal{T}_{112}}^2 + {\mathcal{T}_{113}}^2 + {\mathcal{T}_{211}}^2 + {\mathcal{T}_{212}}^2 + {\mathcal{T}_{213}}^2 + {\mathcal{T}_{311}}^2 + {\mathcal{T}_{312}}^2 + {\mathcal{T}_{313}}^2} \end{aligned} \tag{3.14}$$

Unfortunately, this normalization factor is dependent on the tensor coefficients which varies with time. So we need to find also the derivative of the normalization factor.

$$\begin{aligned} \dot{\mathcal{T}}_{1N} &= \frac{1}{\mathcal{T}_N}(\mathcal{T}_{111}\dot{\mathcal{T}}_{111} + \mathcal{T}_{112}\dot{\mathcal{T}}_{112} + \mathcal{T}_{113}\dot{\mathcal{T}}_{113} + \mathcal{T}_{111}\dot{\mathcal{T}}_{111} \\ &\quad + \mathcal{T}_{212}\dot{\mathcal{T}}_{212} + \mathcal{T}_{213}\dot{\mathcal{T}}_{213} + \mathcal{T}_{311}\dot{\mathcal{T}}_{311} + \mathcal{T}_{312}\dot{\mathcal{T}}_{312} + \mathcal{T}_{313}\dot{\mathcal{T}}_{313}) \\ \dot{\mathcal{T}}_{1N} &= \frac{1}{\mathcal{T}_N}(\mathcal{T}_{111}(R_{i(11)}v_x - \mathcal{T}_{131}\omega_y + \mathcal{T}_{121}\omega_z) + \mathcal{T}_{112}(R_{i(21)}v_x - \mathcal{T}_{132}\omega_y + \mathcal{T}_{122}\omega_z) \\ &\quad + \mathcal{T}_{113}(R_{i(31)}v_x - \mathcal{T}_{133}\omega_y + \mathcal{T}_{123}\omega_z) + \mathcal{T}_{211}(R_{i(12)}v_x - \mathcal{T}_{231}\omega_y + \mathcal{T}_{221}\omega_z) \\ &\quad + \mathcal{T}_{212}(R_{i(22)}v_x - \mathcal{T}_{232}\omega_y + \mathcal{T}_{222}\omega_z) + \mathcal{T}_{213}(R_{i(32)}v_x - \mathcal{T}_{233}\omega_y + \mathcal{T}_{223}\omega_z) \\ &\quad + \mathcal{T}_{311}(R_{i(13)}v_x - \mathcal{T}_{331}\omega_y + \mathcal{T}_{321}\omega_z) + \mathcal{T}_{312}(R_{i(23)}v_x - \mathcal{T}_{332}\omega_y + \mathcal{T}_{322}\omega_z) \\ &\quad + \mathcal{T}_{313}(R_{i(31)}v_x - \mathcal{T}_{333}\omega_y + \mathcal{T}_{323}\omega_z)) \\ \dot{\mathcal{T}}_{1N} &= \begin{bmatrix} \sum_n \sum_m T_{n1m}R_{i(mn)} & 0 & 0 & 0 & \sum_n \sum_m -T_{n1m}\mathcal{T}_{n3m} & \sum_n \sum_m T_{n1m}\mathcal{T}_{n2m} \end{bmatrix} u_c \\ \dot{\mathcal{T}}_{1N} &= L_{1N}u_c \end{aligned} \tag{3.15}$$

The derivative of the normalized tensor coefficients T_{jkl} can then be found

$$\dot{T}_{jkl} = \frac{\dot{\mathcal{T}}_{jkl}}{\mathcal{T}_N} - \frac{\mathcal{T}_{jkl}\dot{\mathcal{T}}_N}{{\mathcal{T}_N}^2}$$
$$\dot{T}_{jkl} = \frac{L_{\mathcal{T}}u_c}{\mathcal{T}_N} - \frac{\mathcal{T}_{jkl}L_N u_c}{{\mathcal{T}_N}^2}$$
$$\dot{T}_{jkl} = \frac{L_{\mathcal{T}}u_c}{\mathcal{T}_N} - \frac{T_{jkl}L_N u_c}{\mathcal{T}_N}$$
$$\dot{T}_{jkl} = \frac{1}{\mathcal{T}_N}(L_{\mathcal{T}} - T_{jkl}L_N)u_c$$
$$L_T = \frac{1}{\mathcal{T}_N}(L_{\mathcal{T}} - T_{jkl}L_N)$$
$$L_{1T} = \frac{L_{1\mathcal{T}}}{\mathcal{T}_{1N}} - T_{j1l}\begin{bmatrix}\sum_n\sum_m T_{n1m}\frac{R_{i(mn)}}{\mathcal{T}_{1N}} & 0 & 0 & 0 & \sum_n\sum_m -T_{n1m}T_{n3m} & \sum_n\sum_m T_{n1m}T_{n2m}\end{bmatrix}$$
$$\dot{T}_{(j1l)} = \frac{R_{i(lj)}}{\mathcal{T}_{1N}}v_{c(1)} - T_{(j1l)}(\sum_n\sum_m T_{n1m}\frac{R_{i(mn)}}{\mathcal{T}_{1N}})v_{c(1)}$$
$$+T_{(j1l)}(\sum_n\sum_m T_{n1m}T_{n3m})\omega_y - T_{(j1l)}(\sum_n\sum_m T_{n1m}T_{n2m})\omega_z + \sum_m [\omega_c]_{\times(1m)}T_{(jml)} \tag{3.16}$$

Same approach can be applied again for the rest of the elements for $k = 2, 3$:

$$T_{j2l} = \frac{\mathcal{T}_{j2l}}{\mathcal{T}_{2N}}$$
$$\mathcal{T}_{2N} = \sqrt{{\mathcal{T}_{121}}^2 + {\mathcal{T}_{122}}^2 + {\mathcal{T}_{123}}^2 + {\mathcal{T}_{221}}^2 + {\mathcal{T}_{222}}^2 + {\mathcal{T}_{223}}^2 + {\mathcal{T}_{321}}^2 + {\mathcal{T}_{322}}^2 + {\mathcal{T}_{323}}^2}$$
$$\dot{\mathcal{T}}_{2N} = \begin{bmatrix}0 & \sum_n\sum_m T_{n2m}R_{i(mn)} & 0 & \sum_n\sum_m T_{n2m}\mathcal{T}_{n3m} & 0 & \sum_n\sum_m -T_{n2m}\mathcal{T}_{n1m}\end{bmatrix} u_c$$

$$T_{j3l} = \frac{\mathcal{T}_{j3l}}{\mathcal{T}_{3N}}$$
$$\mathcal{T}_{3N} = \sqrt{{\mathcal{T}_{131}}^2 + {\mathcal{T}_{132}}^2 + {\mathcal{T}_{133}}^2 + {\mathcal{T}_{231}}^2 + {\mathcal{T}_{232}}^2 + {\mathcal{T}_{233}}^2 + {\mathcal{T}_{331}}^2 + {\mathcal{T}_{332}}^2 + {\mathcal{T}_{333}}^2}$$
$$\dot{\mathcal{T}}_{3N} = \begin{bmatrix}0 & 0 & \sum_n\sum_m T_{n3m}R_{i(mn)} & \sum_n\sum_m -T_{n3m}\mathcal{T}_{n2m} & \sum_n\sum_m T_{n3m}\mathcal{T}_{n1m} & 0\end{bmatrix} u_c$$
(3.17)

Using (3.16) and (3.17), a general formula for $\dot{T}_{(jkl)}$ can then be found as following

$$\begin{aligned}\dot{T}_{(jkl)} = \frac{R_{i(lj)}}{\mathcal{T}_{kN}} v_{c(k)} - T_{(jkl)}(\sum_n \sum_m T_{nkm} \frac{R_{i(mn)}}{\mathcal{T}_{kN}}) v_{c(k)} \\ + T_{(jkl)}(\sum_n \sum_m T_{nkm} T_{nhm}) \omega_{(g)} - T_{(jkl)}(\sum_n \sum_m T_{nkm} T_{ngm}) \omega_{(h)} \\ + \sum_m [\omega_c]_{\times(km)} T_{(jml)} \\ \text{where } g = k\%3 + 1, h = (k+1)\%3 + 1\end{aligned} \tag{3.18}$$

From (3.18), the final normalized interaction matrix is given in (3.19). The normalized interaction matrix when reaching the desired pose is given in (3.20).

As we notice, this particular choice for the normalization factors has led to keep the decoupling property between the translational velocities, as each degree of freedom related tensor coefficients are normalized separately from other degrees of freedom. Other choices of different combinations of tensor coefficients as normalization factors, would introduce extra non-zero elements to the normalized interaction matrix and prevent keeping the decoupling property.

$$
\begin{bmatrix}
\frac{R_{i(11)}}{\mathcal{T}_{1N}} - T_{(111)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(131)} + T_{(111)}(\text{sum-wy1}) & T_{(121)} - T_{(111)}(\text{sum-wz1}) \\
\frac{R_{i(21)}}{\mathcal{T}_{1N}} - T_{(112)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(132)} + T_{(112)}(\text{sum-wy1}) & T_{(122)} - T_{(112)}(\text{sum-wz1}) \\
\frac{R_{i(31)}}{\mathcal{T}_{1N}} - T_{(113)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(133)} + T_{(113)}(\text{sum-wy1}) & T_{(123)} - T_{(113)}(\text{sum-wz1}) \\
0 & \frac{R_{i(11)}}{\mathcal{T}_{2N}} - T_{(121)}(\text{sum-vy}) & 0 & T_{(131)} - T_{(121)}(\text{sum-wx2}) & 0 & -T_{(111)} + T_{(121)}(\text{sum-wz2}) \\
0 & \frac{R_{i(21)}}{\mathcal{T}_{2N}} - T_{(122)}(\text{sum-vy}) & 0 & T_{(132)} - T_{(122)}(\text{sum-wx2}) & 0 & -T_{(112)} + T_{(122)}(\text{sum-wz2}) \\
0 & \frac{R_{i(31)}}{\mathcal{T}_{2N}} - T_{(123)}(\text{sum-vy}) & 0 & T_{(133)} - T_{(123)}(\text{sum-wx2}) & 0 & -T_{(113)} + T_{(123)}(\text{sum-wz2}) \\
0 & 0 & \frac{R_{i(11)}}{\mathcal{T}_{3N}} - T_{(131)}(\text{sum-vz}) & -T_{(121)} + T_{(131)}(\text{sum-wx3}) & T_{(111)} - T_{(131)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(21)}}{\mathcal{T}_{3N}} - T_{(132)}(\text{sum-vz}) & -T_{(122)} + T_{(132)}(\text{sum-wx3}) & T_{(112)} - T_{(132)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(31)}}{\mathcal{T}_{3N}} - T_{(133)}(\text{sum-vz}) & -T_{(123)} + T_{(133)}(\text{sum-wx3}) & T_{(113)} - T_{(133)}(\text{sum-wy3}) & 0 \\
\frac{R_{i(12)}}{\mathcal{T}_{1N}} - T_{(211)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(231)} + T_{(211)}(\text{sum-wy1}) & T_{(221)} - T_{(211)}(\text{sum-wz1}) \\
\frac{R_{i(22)}}{\mathcal{T}_{1N}} - T_{(212)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(232)} + T_{(212)}(\text{sum-wy1}) & T_{(222)} - T_{(212)}(\text{sum-wz1}) \\
\frac{R_{i(32)}}{\mathcal{T}_{1N}} - T_{(213)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(233)} + T_{(213)}(\text{sum-wy1}) & T_{(223)} - T_{(213)}(\text{sum-wz1}) \\
0 & \frac{R_{i(12)}}{\mathcal{T}_{2N}} - T_{(221)}(\text{sum-vy}) & 0 & T_{(231)} - T_{(221)}(\text{sum-wx2}) & 0 & -T_{(211)} + T_{(221)}(\text{sum-wz2}) \\
0 & \frac{R_{i(22)}}{\mathcal{T}_{2N}} - T_{(222)}(\text{sum-vy}) & 0 & T_{(232)} - T_{(222)}(\text{sum-wx2}) & 0 & -T_{(212)} + T_{(222)}(\text{sum-wz2}) \\
0 & \frac{R_{i(32)}}{\mathcal{T}_{2N}} - T_{(223)}(\text{sum-vy}) & 0 & T_{(233)} - T_{(223)}(\text{sum-wx2}) & 0 & -T_{(213)} + T_{(223)}(\text{sum-wz2}) \\
0 & 0 & \frac{R_{i(12)}}{\mathcal{T}_{3N}} - T_{(231)}(\text{sum-vz}) & -T_{(221)} + T_{(231)}(\text{sum-wx3}) & T_{(211)} - T_{(231)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(22)}}{\mathcal{T}_{3N}} - T_{(232)}(\text{sum-vz}) & -T_{(222)} + T_{(232)}(\text{sum-wx3}) & T_{(212)} - T_{(232)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(32)}}{\mathcal{T}_{3N}} - T_{(233)}(\text{sum-vz}) & -T_{(223)} + T_{(233)}(\text{sum-wx3}) & T_{(213)} - T_{(233)}(\text{sum-wy3}) & 0 \\
\frac{R_{i(13)}}{\mathcal{T}_{1N}} - T_{(311)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(331)} + T_{(311)}(\text{sum-wy1}) & T_{(321)} - T_{(311)}(\text{sum-wz1}) \\
\frac{R_{i(23)}}{\mathcal{T}_{1N}} - T_{(312)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(332)} + T_{(312)}(\text{sum-wy1}) & T_{(322)} - T_{(312)}(\text{sum-wz1}) \\
\frac{R_{i(33)}}{\mathcal{T}_{1N}} - T_{(313)}(\text{sum-vx}) & 0 & 0 & 0 & -T_{(333)} + T_{(313)}(\text{sum-wy1}) & T_{(323)} - T_{(313)}(\text{sum-wz1}) \\
0 & \frac{R_{i(13)}}{\mathcal{T}_{2N}} - T_{(321)}(\text{sum-vy}) & 0 & T_{(331)} - T_{(321)}(\text{sum-wx2}) & 0 & -T_{(311)} + T_{(321)}(\text{sum-wz2}) \\
0 & \frac{R_{i(23)}}{\mathcal{T}_{2N}} - T_{(322)}(\text{sum-vy}) & 0 & T_{(332)} - T_{(322)}(\text{sum-wx2}) & 0 & -T_{(312)} + T_{(322)}(\text{sum-wz2}) \\
0 & \frac{R_{i(33)}}{\mathcal{T}_{2N}} - T_{(323)}(\text{sum-vy}) & 0 & T_{(333)} - T_{(323)}(\text{sum-wx2}) & 0 & -T_{(313)} + T_{(323)}(\text{sum-wz2}) \\
0 & 0 & \frac{R_{i(13)}}{\mathcal{T}_{3N}} - T_{(331)}(\text{sum-vz}) & -T_{(321)} + T_{(331)}(\text{sum-wx3}) & T_{(311)} - T_{(331)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(23)}}{\mathcal{T}_{3N}} - T_{(332)}(\text{sum-vz}) & -T_{(322)} + T_{(332)}(\text{sum-wx3}) & T_{(312)} - T_{(332)}(\text{sum-wy3}) & 0 \\
0 & 0 & \frac{R_{i(33)}}{\mathcal{T}_{3N}} - T_{(333)}(\text{sum-vz}) & -T_{(323)} + T_{(333)}(\text{sum-wx3}) & T_{(313)} - T_{(333)}(\text{sum-wy3}) & 0
\end{bmatrix}
$$

$$
\text{sum-vx} = \sum_n \sum_m T_{n1m} \frac{R_{i(mn)}}{\mathcal{T}_{1N}}, \text{sum-wy1} = \sum_n \sum_m T_{n1m} T_{n3m}, \text{sum-wz1} = \sum_n \sum_m T_{n1m} T_{n2m}
$$

$$
\text{sum-vy} = \sum_n \sum_m T_{n2m} \frac{R_{i(mn)}}{\mathcal{T}_{2N}}, \text{sum-wx2} = \sum_n \sum_m T_{n2m} T_{n3m}, \text{sum-wz2} = \sum_n \sum_m T_{n2m} T_{n1m}
$$

$$
\text{sum-vz} = \sum_n \sum_m T_{n3m} \frac{R_{i(mn)}}{\mathcal{T}_{3N}}, \text{sum-wx3} = \sum_n \sum_m T_{n3m} T_{n2m}, \text{sum-wy3} = \sum_n \sum_m T_{n3m} T_{n1m} \tag{3.19}
$$

$$
\begin{bmatrix}
\frac{R_{i(11)}}{\mathcal{T}_N} - T_{(111)}\text{(sum-vx)} & 0 & 0 & 0 & T_{(111)}\text{(sum-wy1)} & -T_{(111)}\text{(sum-wz1)} \\
\frac{R_{i(21)}}{\mathcal{T}_N} - T_{(112)}\text{(sum-vx)} & 0 & 0 & 0 & T_{(112)}\text{(sum-wy1)} & -T_{(112)}\text{(sum-wz1)} \\
\frac{R_{i(31)}}{\mathcal{T}_N} - T_{(113)}\text{(sum-vx)} & 0 & 0 & 0 & T_{(113)}\text{(sum-wy1)} & -T_{(113)}\text{(sum-wz1)} \\
0 & \frac{R_{i(11)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(111)} \\
0 & \frac{R_{i(21)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(112)} \\
0 & \frac{R_{i(31)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(113)} \\
0 & 0 & \frac{R_{i(11)}}{\mathcal{T}_N} & 0 & T_{(111)} & 0 \\
0 & 0 & \frac{R_{i(21)}}{\mathcal{T}_N} & 0 & T_{(112)} & 0 \\
0 & 0 & \frac{R_{i(31)}}{\mathcal{T}_N} & 0 & T_{(113)} & 0 \\
\frac{R_{i(12)}}{\mathcal{T}_N} & 0 & 0 & 0 & 0 & T_{(221)} \\
\frac{R_{i(22)}}{\mathcal{T}_N} & 0 & 0 & 0 & 0 & T_{(222)} \\
\frac{R_{i(32)}}{\mathcal{T}_N} & 0 & 0 & 0 & 0 & T_{(223)} \\
0 & \frac{R_{i(12)}}{\mathcal{T}_N} - T_{(221)}\text{(sum-vy)} & 0 & -T_{(221)}\text{(sum-wx2)} & 0 & T_{(221)}\text{(sum-wz2)} \\
0 & \frac{R_{i(22)}}{\mathcal{T}_N} - T_{(222)}\text{(sum-vy)} & 0 & -T_{(222)}\text{(sum-wx2)} & 0 & T_{(222)}\text{(sum-wz2)} \\
0 & \frac{R_{i(32)}}{\mathcal{T}_N} - T_{(223)}\text{(sum-vy)} & 0 & -T_{(223)}\text{(sum-wx2)} & 0 & T_{(223)}\text{(sum-wz2)} \\
0 & 0 & \frac{R_{i(12)}}{\mathcal{T}_N} & -T_{(221)} & 0 & 0 \\
0 & 0 & \frac{R_{i(22)}}{\mathcal{T}_N} & -T_{(222)} & 0 & 0 \\
0 & 0 & \frac{R_{i(32)}}{\mathcal{T}_N} & -T_{(223)} & 0 & 0 \\
\frac{R_{i(13)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(331)} & 0 \\
\frac{R_{i(23)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(332)} & 0 \\
\frac{R_{i(33)}}{\mathcal{T}_N} & 0 & 0 & 0 & -T_{(333)} & 0 \\
0 & \frac{R_{i(13)}}{\mathcal{T}_N} & 0 & T_{(331)} & 0 & 0 \\
0 & \frac{R_{i(23)}}{\mathcal{T}_N} & 0 & T_{(332)} & 0 & 0 \\
0 & \frac{R_{i(33)}}{\mathcal{T}_N} & 0 & T_{(333)} & 0 & 0 \\
0 & 0 & \frac{R_{i(13)}}{\mathcal{T}_N} - T_{(331)}\text{(sum-vz)} & T_{(331)}\text{(sum-wx3)} & -T_{(331)}\text{(sum-wy3)} & 0 \\
0 & 0 & \frac{R_{i(23)}}{\mathcal{T}_N} - T_{(332)}\text{(sum-vz)} & T_{(332)}\text{(sum-wx3)} & -T_{(332)}\text{(sum-wy3)} & 0 \\
0 & 0 & \frac{R_{i(33)}}{\mathcal{T}_N} - T_{(333)}\text{(sum-vz)} & T_{(333)}\text{(sum-wx3)} & -T_{(333)}\text{(sum-wy3)} & 0
\end{bmatrix}
\tag{3.20}
$$

$$
\text{sum-vx} = \sum_n \sum_m T_{n1m} \frac{R_{i(mn)}}{\mathcal{T}_{1N}}, \text{sum-wy1} = \sum_n \sum_m T_{n1m} T_{n3m}, \text{sum-wz1} = \sum_n \sum_m T_{n1m} T_{n2m}
$$

$$
\text{sum-vy} = \sum_n \sum_m T_{n2m} \frac{R_{i(mn)}}{\mathcal{T}_{2N}}, \text{sum-wx2} = \sum_n \sum_m T_{n2m} T_{n3m}, \text{sum-wz2} = \sum_n \sum_m T_{n2m} T_{n1m}
$$

$$
\text{sum-vz} = \sum_n \sum_m T_{n3m} \frac{R_{i(mn)}}{\mathcal{T}_{3N}}, \text{sum-wx3} = \sum_n \sum_m T_{n3m} T_{n2m}, \text{sum-wy3} = \sum_n \sum_m T_{n3m} T_{n1m}
$$

3.3 Proposed Method

We have defined all the elements of the control law (3.1). The interaction matrix $L_{T_{(jkl)}}$ is computed as indicated in (3.19). The error e is computed using the difference between the current tensor coefficients and the desired tensor coefficients, $e = T_{(jkl)} - T^*_{(jkl)}$.

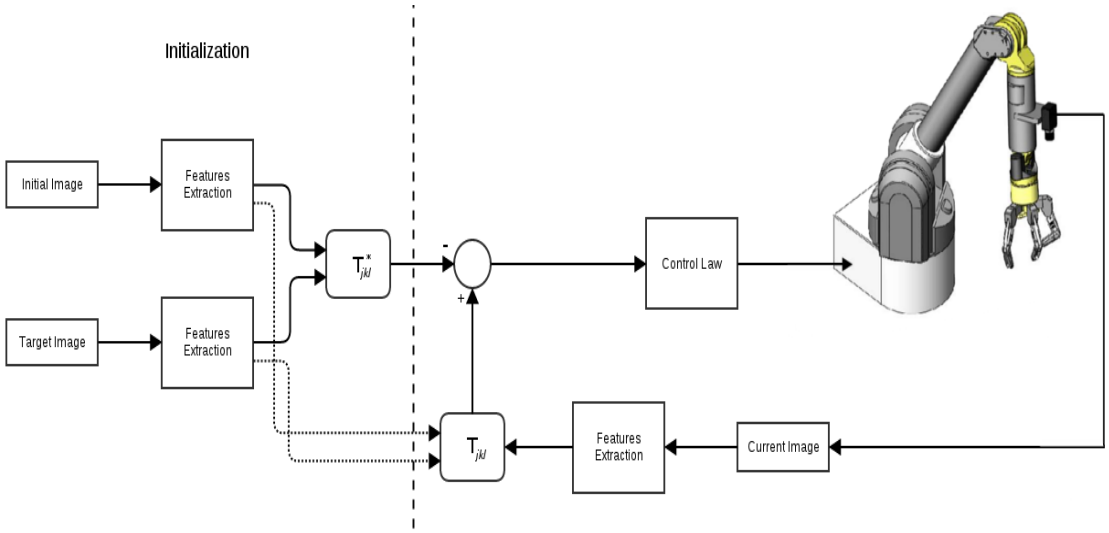


Figure 3.1: Block diagram of the proposed Trifocal Tensor Visual Servoing loop control

In Figure 3.1, we present the block diagram explaining the proposed trifocal tensor visual servoing control loop. At initialization, the desired tensor $T^*_{(jkl)}$ is computed from feature correspondences across three images obtained from the three camera poses: initial pose, desired pose, and current pose being equal to the desired pose. In this method, we assume the camera intrinsic parameters are known. It means before computing the tensor from image correspondences, matching points have to be transformed to calibrated coordinates.

The current tensor $T_{(jkl)}$ is computed inside the visual servoing loop at each iteration. Then the interaction matrix is computed using the current tensor. For simplicity, we make the assumption that the initial pose is known and the interaction matrix can be computed directly. The new error value is computed along with the pseudo-inverse of the interaction matrix and fed back to the control law to compute the required velocities to drive the camera to the desired pose. The system converges and the loop is terminated when the camera reaches the desired pose. This is evaluated when the sum squared of the error reaches a value less than a defined threshold, 1×10^{-6} for example.

Chapter 4

Experimental Results

In this chapter, we begin by describing the experimental setup for the proposed method we present. Then the results of applying the method on several scene configurations are presented. For each configuration we present a comparison between the ideal theoretical results obtained by computing the tensor from given known camera poses, and the practical results obtained from matching image correspondences and estimating the tensor numerically.

4.1 Experimental Setup

The proposed method was implemented in C++, and using the `ViSP` library. `ViSP` is an open-source visual servoing framework library developed by `INRIA` and written in C++ [14]. It is a modular cross platform library that allows prototyping and developing applications using visual tracking and visual servoing techniques. The implementation is divided into two main classes: Visual Servoing Manager, and Trifocal Tensor.

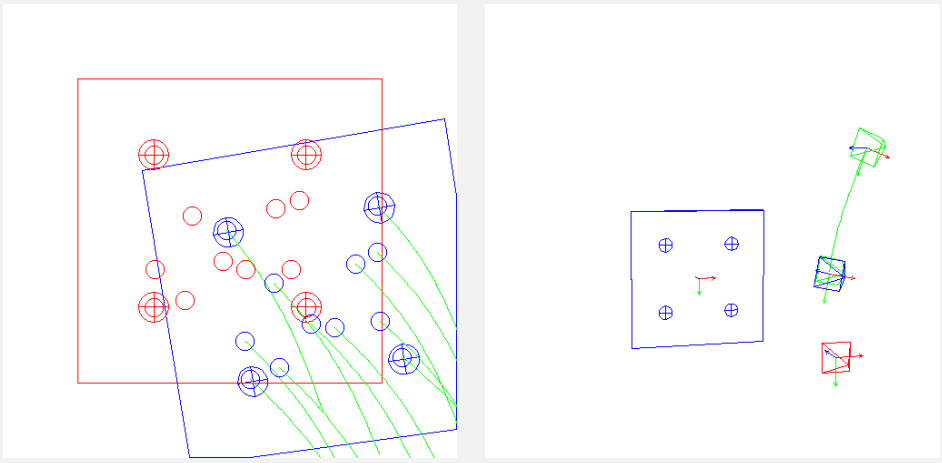
The Trifocal Tensor class was implemented similar to the available data types classes in `ViSP`, like Matrices and Vectors. This class is responsible for:

- Storing the numerical values of the tensor.
- Computing the tensor using pose projection matrices.
- Computing the tensor using image correspondences across three images.
- Recovering epipoles and projection matrices.

- Normalizing the tensor.
- Computing the interaction matrix corresponding to the tensor instance.

The Visual Servoing Manager class is where the simulator, the scene, and the visual servoing task are defined. The class is responsible for:

- Defining the scene configuration, the initial and desired poses.
- Storing different instances of the current and desired trifocal tensors.
- Computing the control law and velocities of the simulation camera.
- Plotting graphs and camera simulations showing the variation of tensor error values, computed velocities, and 3D camera end trajectories.



(a) (b)

Figure 4.1: Experimental Setup (a) Image view with Current/Desired camera projected images plotted in blue/red, with the points trajectory in green. (b) Scene view with initial/current/desired cameras plotted in green/blue/red, with the camera trajectory in green.

The scene setup is defined with two poses for the camera: initial and desired, along with a plate object to be tracked in the scene. A trifocal tensor requires at least 7 non coplanar points for estimating a tensor. Throughout our experiments, we used two sets of 8 and 12 points. There are no much differences between the results of the two sets because we are not considering points mismatch and outliers throughout the experiments. The following results are for the set of 12 points. Throughout the results we show two different views: Image view, and Scene view. Figure 4.1a shows the image view with projected current and desired camera images plotted in blue and red respectively, along with the points trajectories in green. Figure 4.1b shows the scene view with the position of the initial, current, and desired camera frames plotted in green, blue and red respectively. The resulting camera trajectory is plotted in green.

4.2 Experiment I: Pure Translation along x-axis, y-axis

For first experiment, we consider a pure translational motion along one axis. It's the basic test that can be done to ensure the convergence of the control loop. We consider a small translation of $0.1m$ along x-axis, then y-axis. Figure 4.2 shows the results of computing the tensor using the pose. Figure 4.2c shows the evolution of the camera velocities, and Figure 4.2d shows the evolution of the trifocal tensor coefficients error. Figure 4.3 shows the results of estimating the tensor through points correspondences. A translation along y-axis produces similar results.

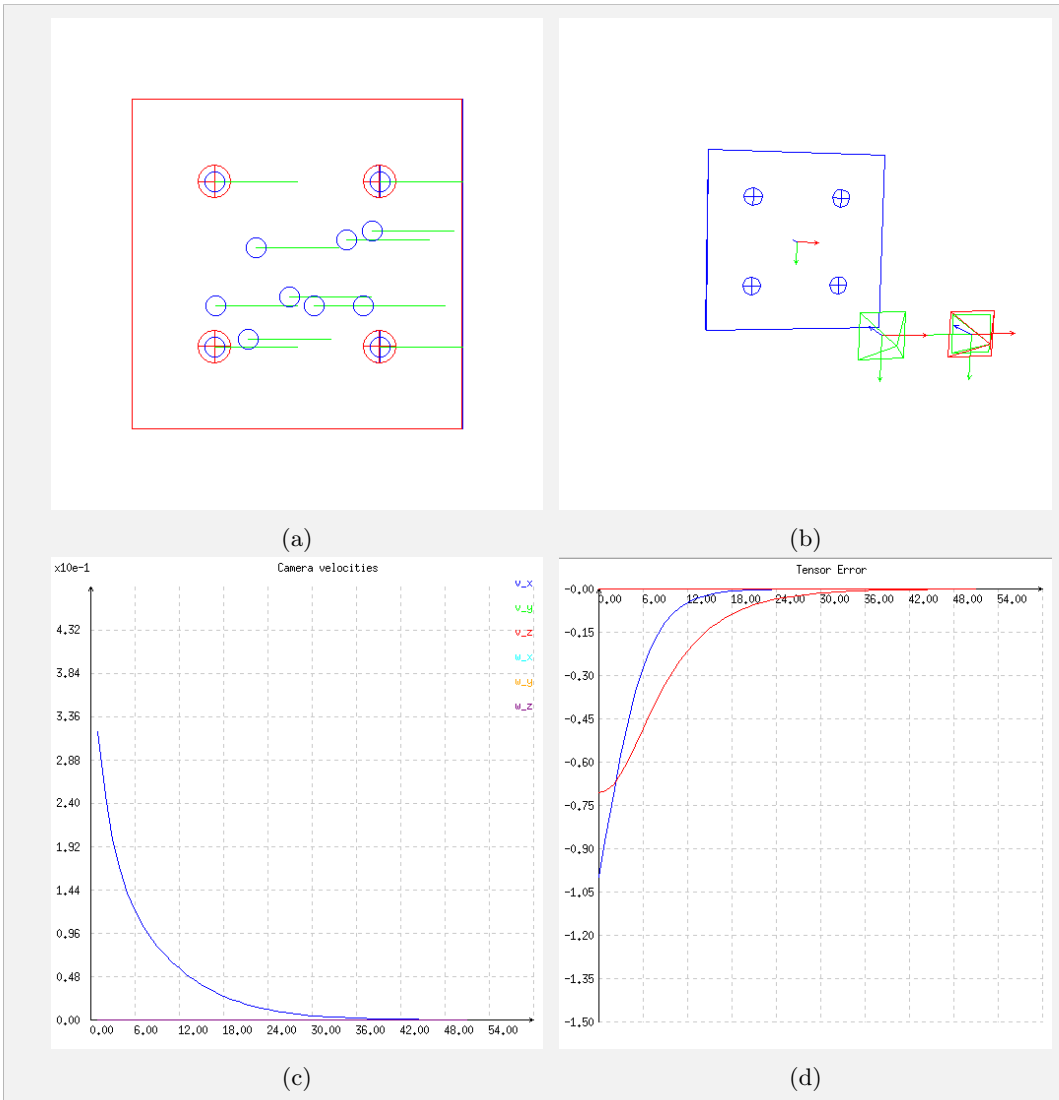


Figure 4.2: Translation along x-axis using tensor from poses. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

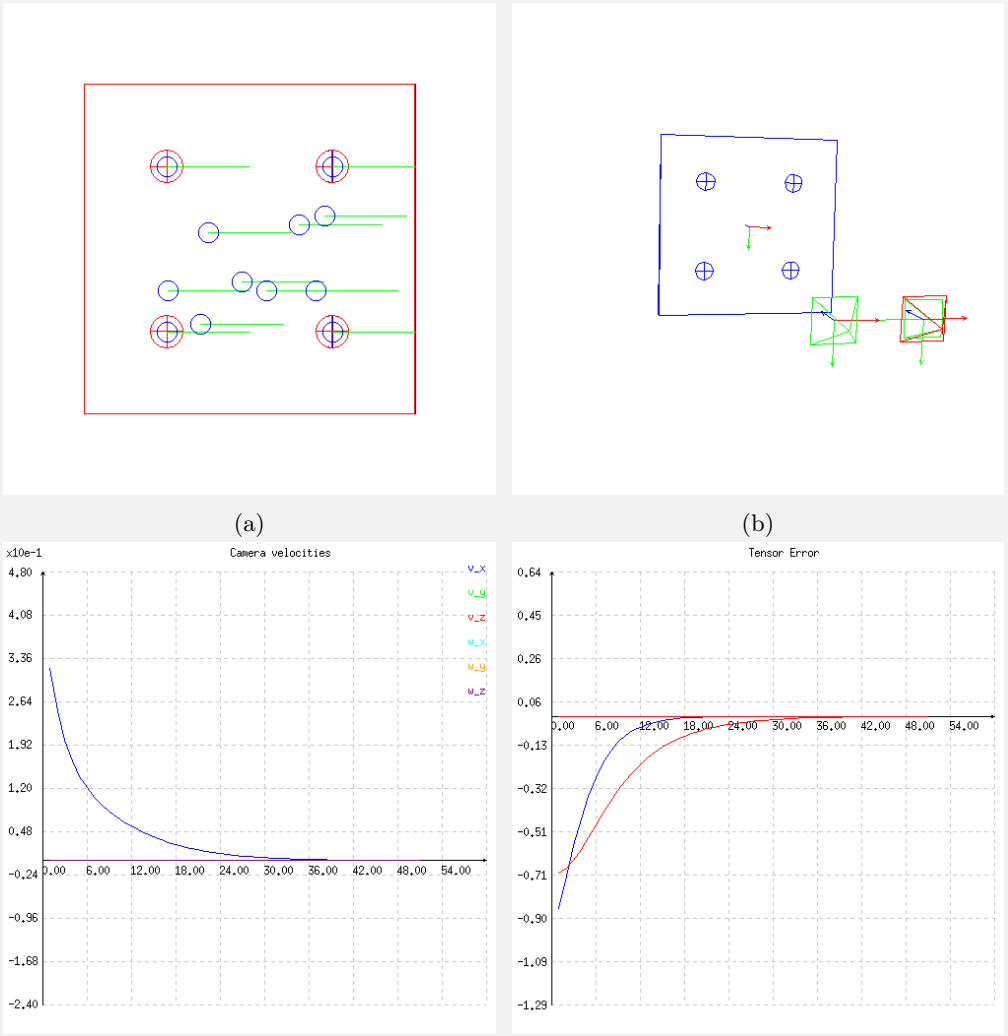
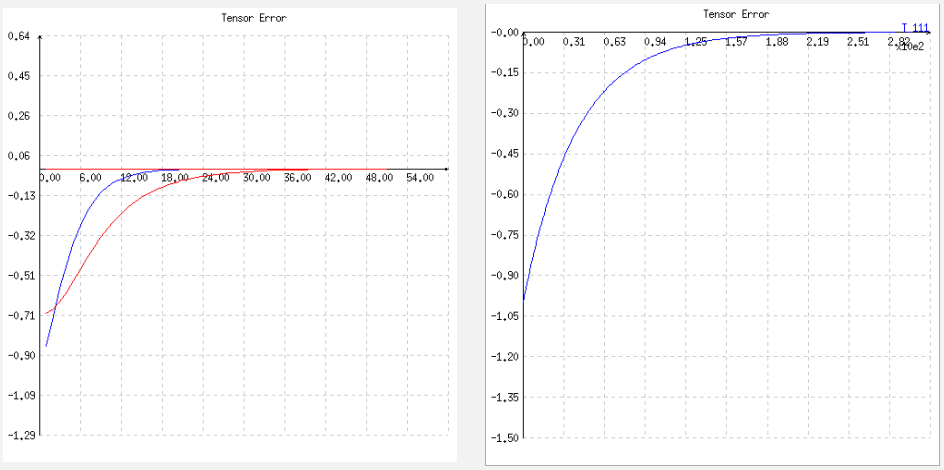


Figure 4.3: Translation along x-axis using estimated tensor. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

The system behaviour in both cases is identical. We observe the camera trajectory is linear, which is satisfactory. However, the error is not exactly exponentially decreasing. Several tests were conducted to analyse the source of this behaviour, and it was found that using a subset of tensor coefficients for computing the error and the interaction matrix leads to a perfect exponentially decreasing error. A subset was chosen experimentally to handle this specific configuration, but for different subsets are then required for different configurations. The problem of subset selection is discussed in details in 5.2. Figure 4.4 shows a comparison between the error plots using all the tensor coefficients and using a subset of them.



(a) (b)

Figure 4.4: Comparison between tensor coefficients error when (a) using all tensor coefficients (b) using a subset of tensor coefficients.

4.3 Experiment II: Pure Translation along z-axis

Motion along the z-axis is usually more challenging than xyz motion in the IBVS approach. This is due to poor motion resolvability when the camera moves towards the feature points [15]. On the contrary, using the trifocal tensor features this motion is not different than other types of motions. Figure 4.5 and Figure 4.6 show the results for the pose and estimated tensors respectively. Like a simple translation along x-axis or y-axis, a translation along z-axis also suffers the problem of selecting a subset of tensor coefficients.

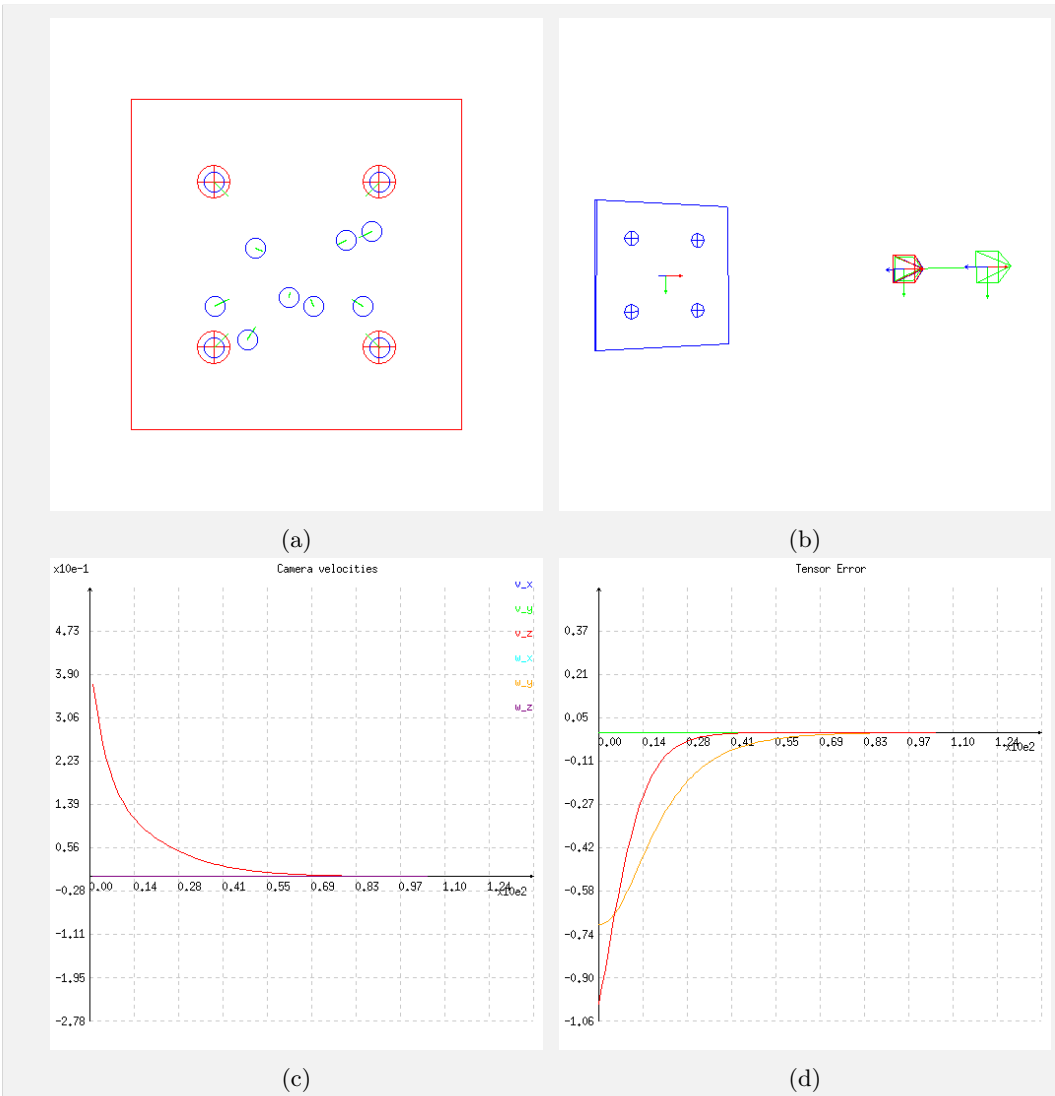


Figure 4.5: Translation along z-axis using tensor from poses. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

4.4 Experiment III: Translation along xyz-axes

Now we consider a motion along the 3 x,y, and z axes. From Figure 4.7d and 4.8d, we observe the tensor error is not behaving as a desired exponentially decreasing function. However, the camera trajectory is still linear, which is expected and satisfactory.

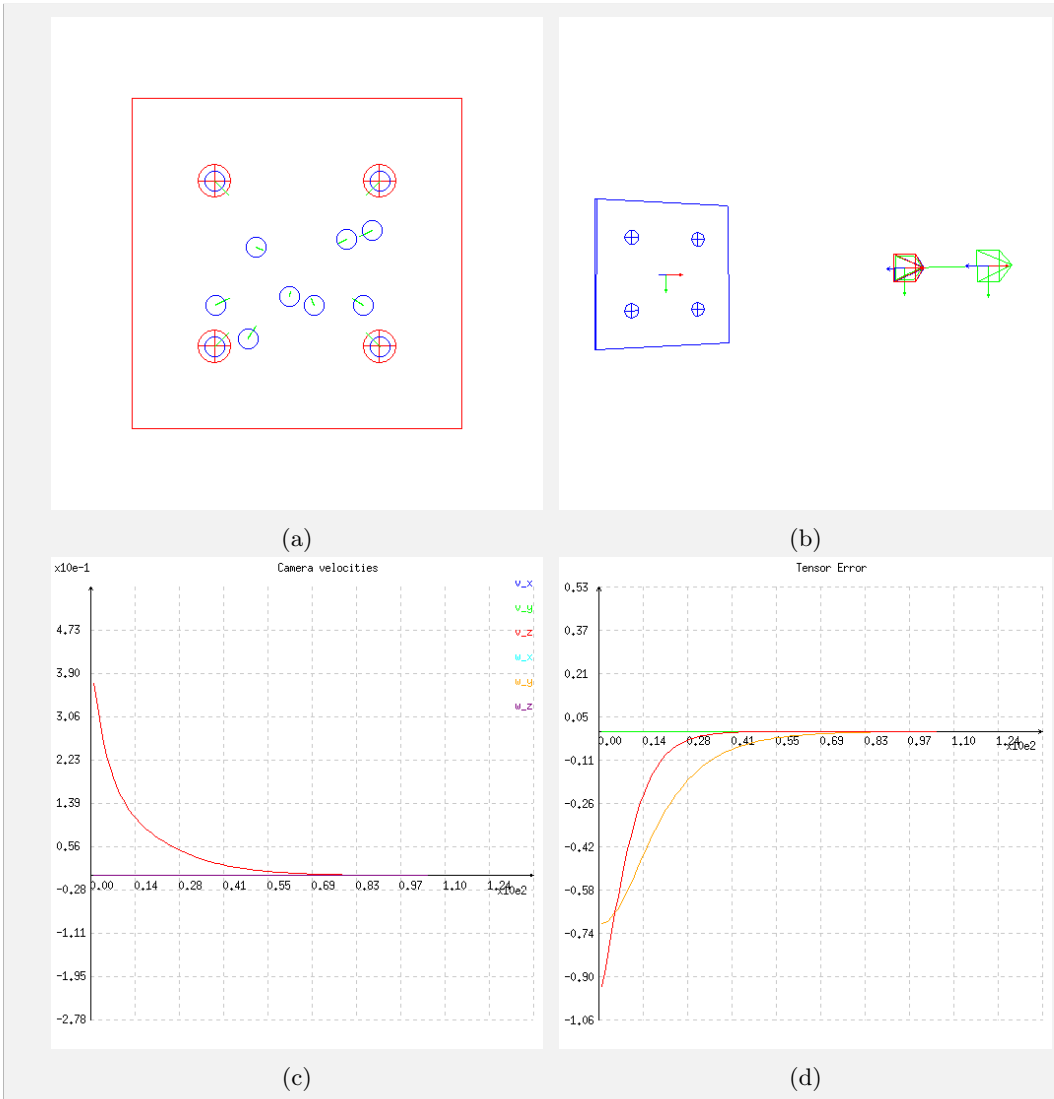


Figure 4.6: Translation along z-axis using estimated tensor. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

4.5 Experiment IV: Large rotation around z-axis

One of the most challenging IBVS configurations is the 180°rotation around the z-axis [5] [16]. This is due to the nature of the image-based control law which makes the camera retreat from the object instead of rotating around the z-axis. It is important to evaluate the visual servoing

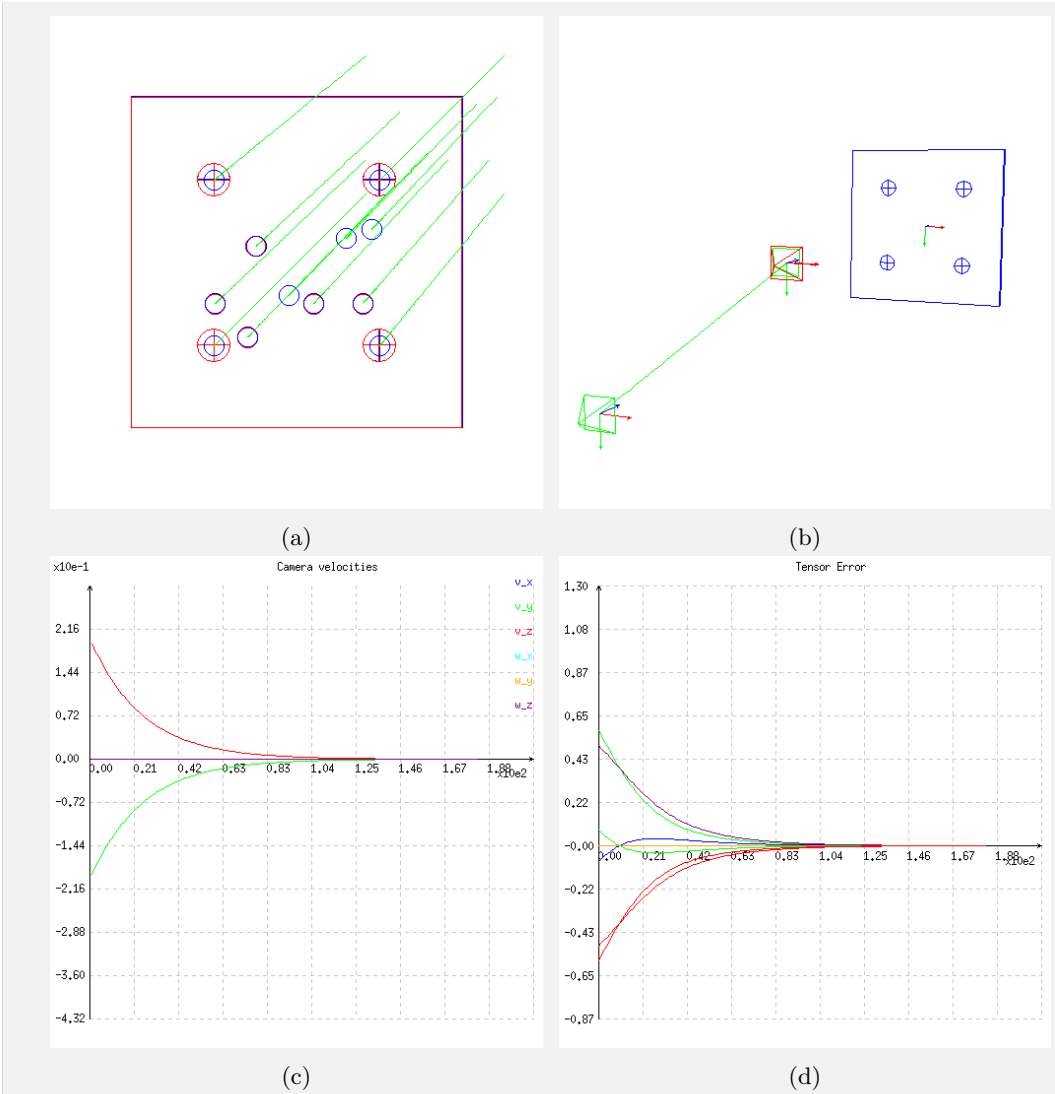


Figure 4.7: Translation along xyz-axes using tensor from poses. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

method for large z-axis rotations, close to 180°.

Here we consider a translation of $1m$ and a 170°rotation. As we can observe in Figure 4.9 and Figure 4.10, the retreat problem didn't occur. The image trajectories follow a spiral motion, which is exactly as expected due to the linear motion for the translation, and the rotational motion for the rotation.

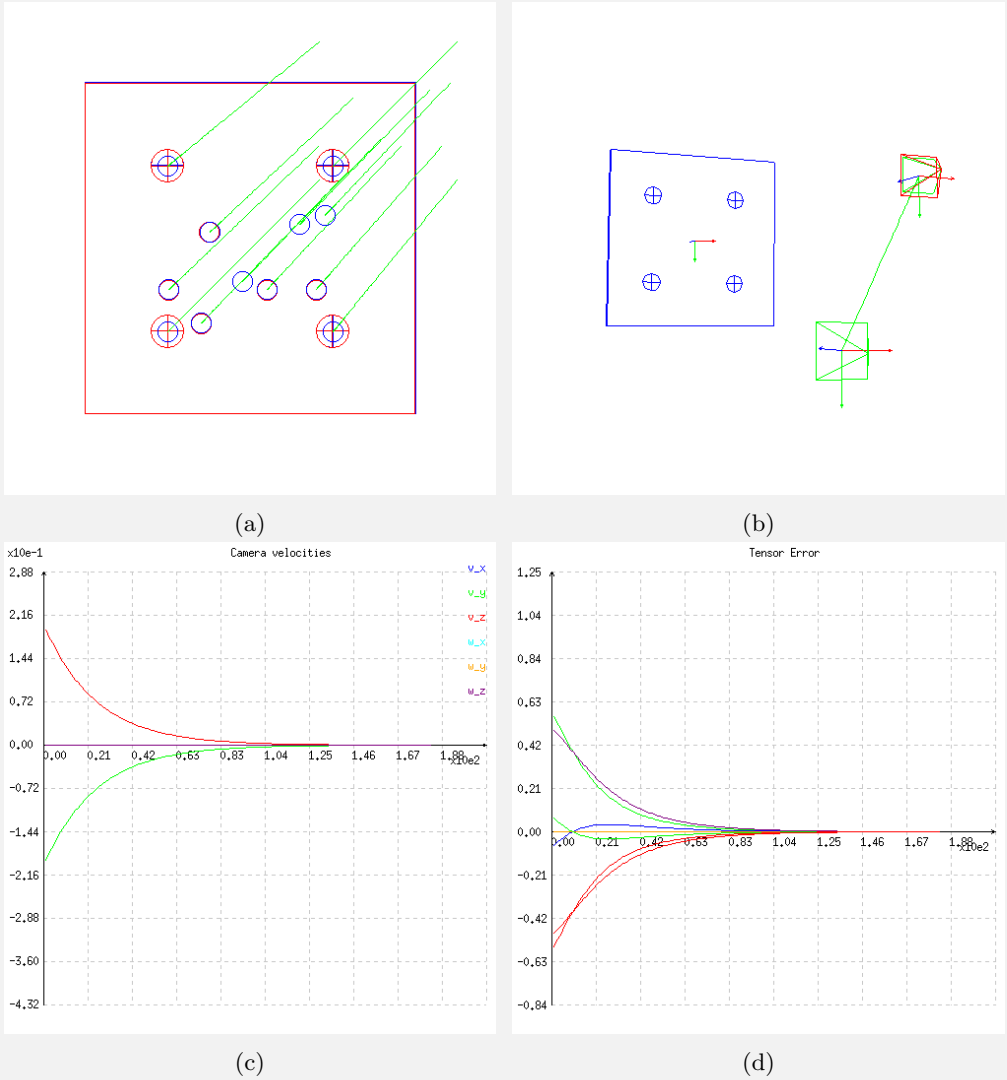


Figure 4.8: Translation along xyz-axes using estimated tensor. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

4.6 Experiment V: Generic Motion

In this experiment, we choose a generic camera motion: translations of $0.25m, -0.3m, 0.2m$, and rotations of $10°, -20°, 5°$ along x,y, and z axes respectively. The results in Figures 4.11 and 4.12 show smooth camera and image trajectories. However, comparing the two Figures 4.11c and

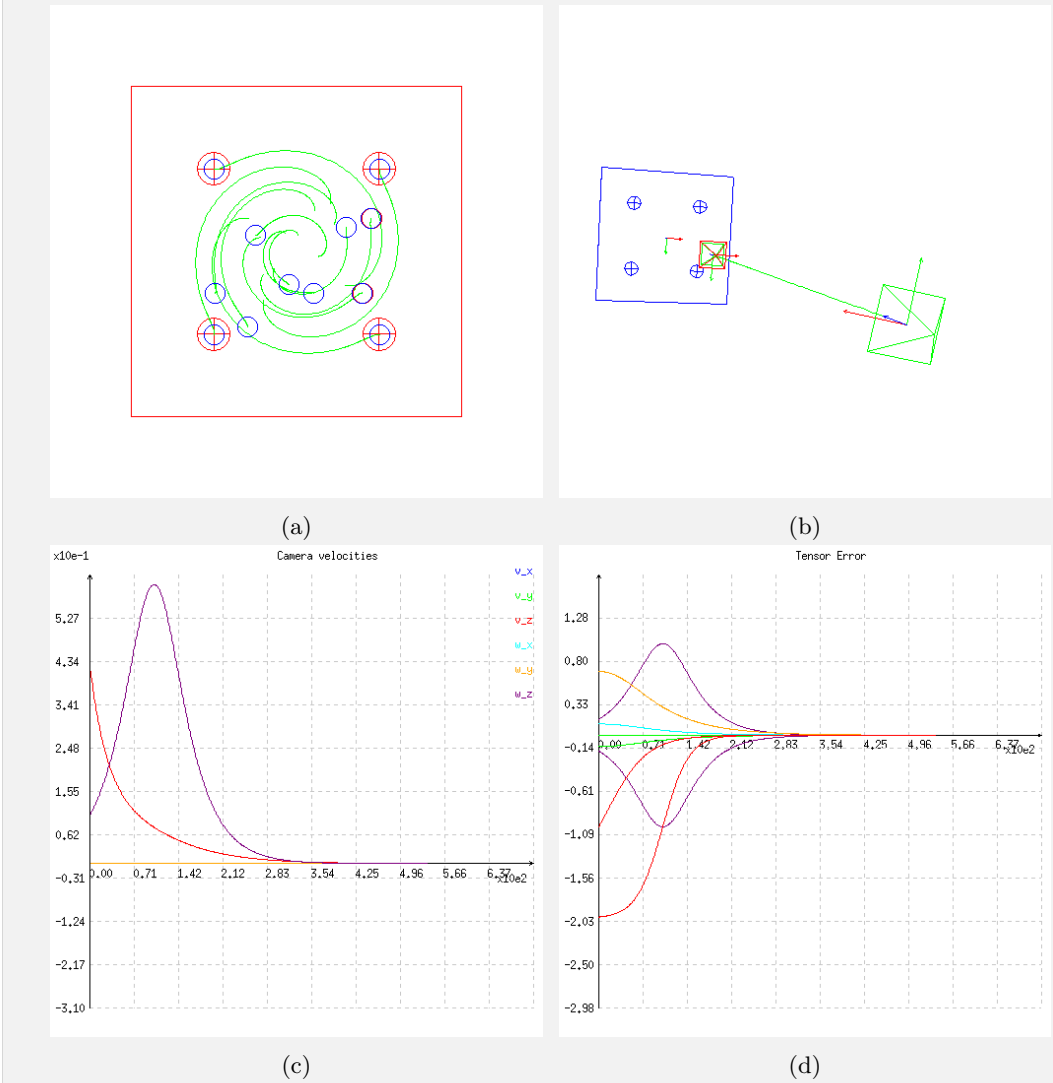


Figure 4.9: Translation and large rotation along z-axis using tensor from poses. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

4.12c, we notice small oscillations in the camera velocities initially. Lopez-Nicolas [12] mentioned similar oscillations due to noise during image points extraction and matching. Although in our case we omit points mismatch and outliers, but this behaviour is most likely due to small numerical error in the trifocal tensor estimation, as the pose tensor doesn't produce similar oscillations.

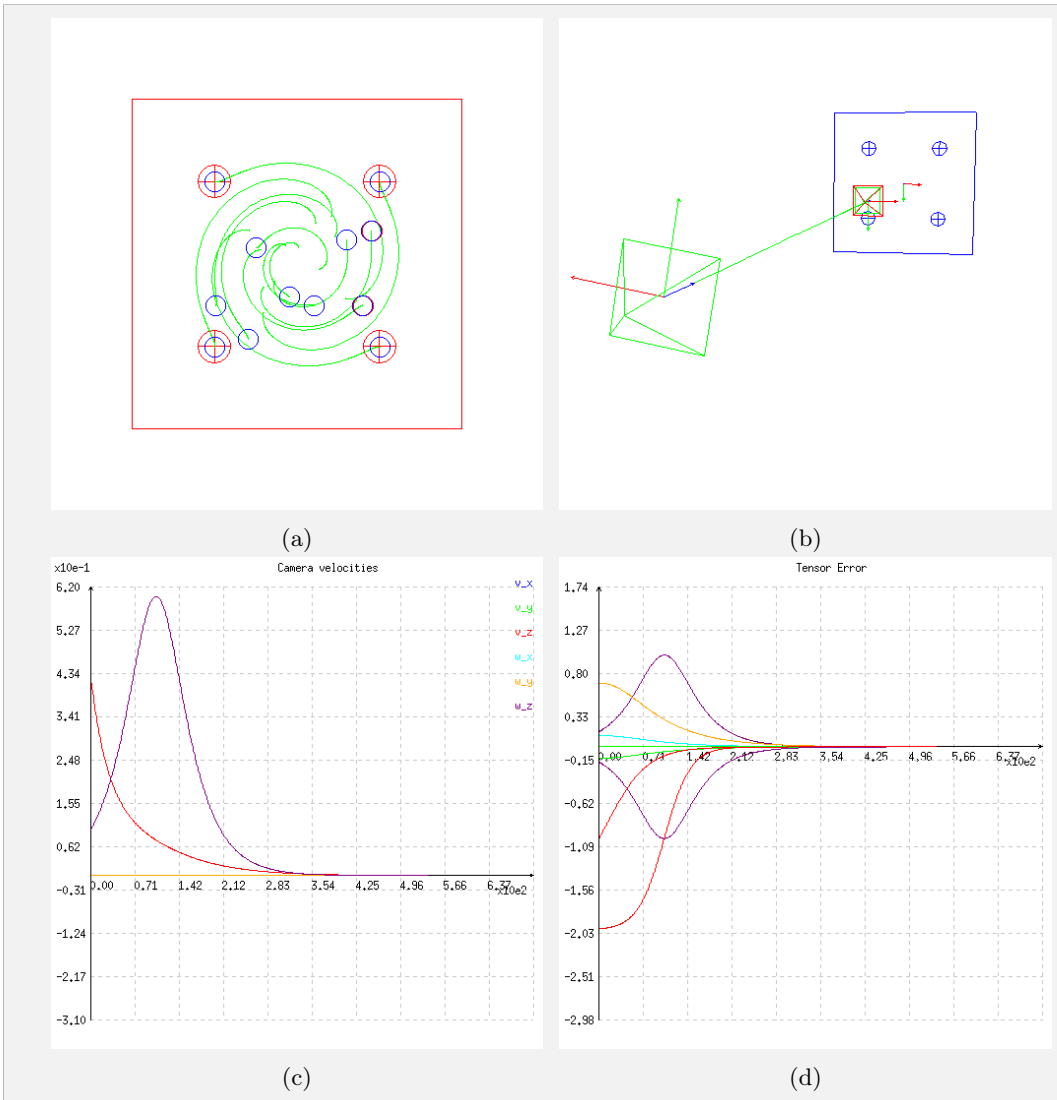
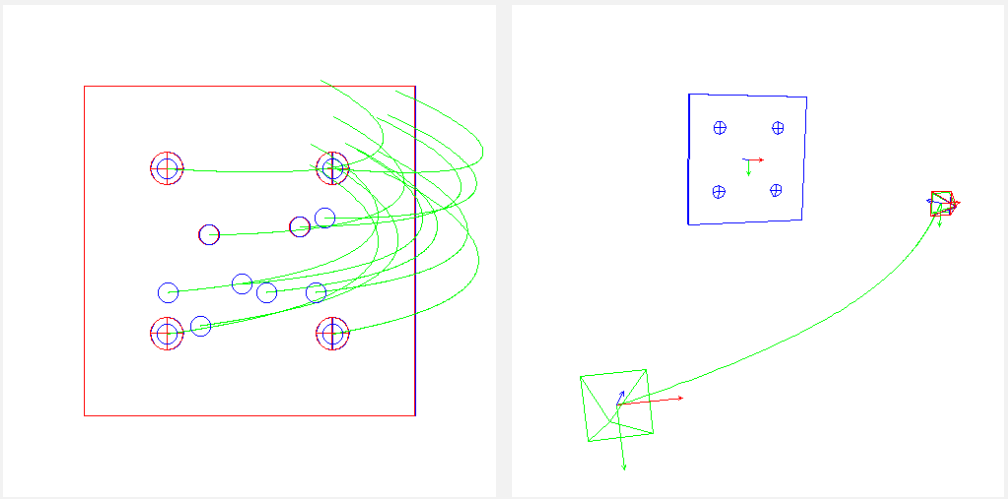
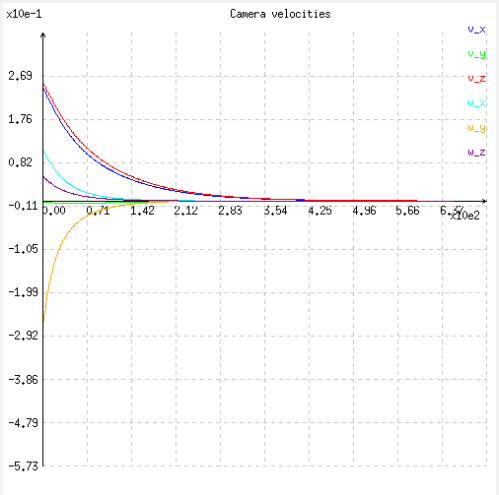


Figure 4.10: Translation and large rotation along z-axis using estimated tensor. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

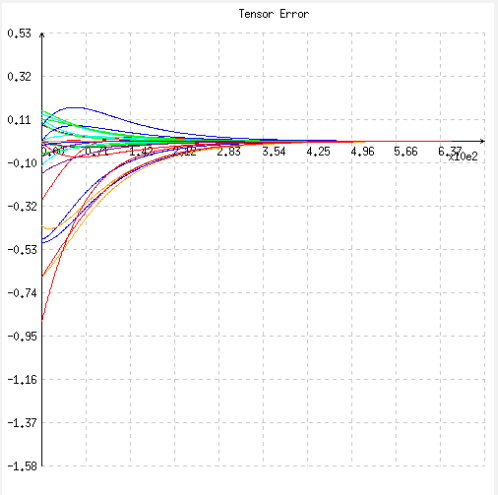


(a)

(b)



(c)



(d)

Figure 4.11: Generic motion using tensor from poses. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

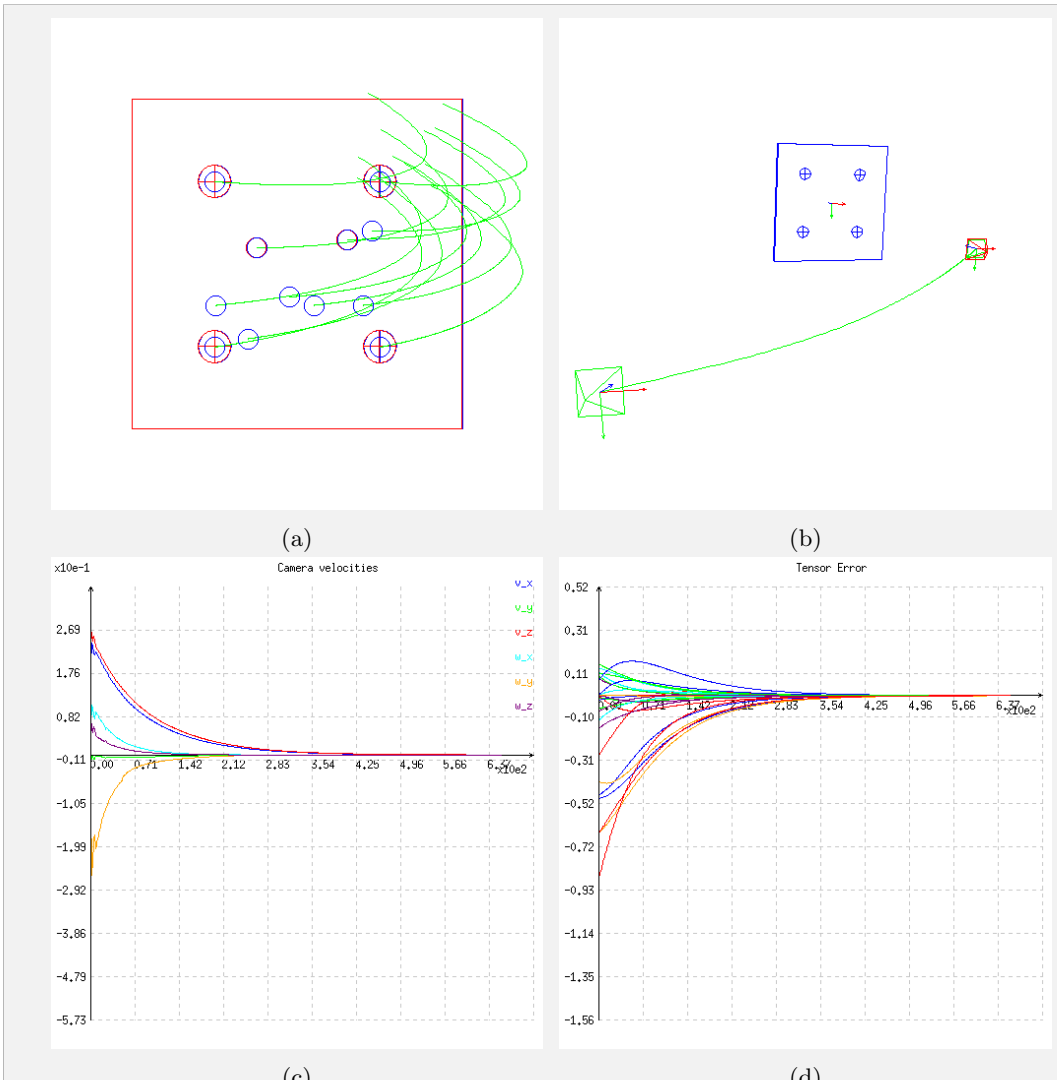


Figure 4.12: Generic motion using estimated tensor. (a) Image view. (b) Scene view. (c) Camera velocities. (d) Tensor coefficients error.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Visual Servoing is a growing active field of research. Recent advances in robotics and computer vision domains has led to the emergence of this vision-based closed loop control methods to control a robot manipulator or a mobile robot through vision acquired by a camera. Many works have studied closing the control loop over visual features computed either from the image space or the estimated 3D pose of the scene. Few works studied using stereo-vision geometry, and up until recently only a couple of works studied using the trifocal tensor into the visual servoing control loop.

An approach to incorporate the trifocal tensor estimated from the three-views geometry into the visual servoing control loop task has been presented in this work. This approach presents a generalized 6-DOF visual servoing task, with the control loop being closed over projective measures, namely the trifocal tensor coefficients. To the best of our knowledge, this approach is the first to propose a fully analytical design for a visual servoing task based on the trifocal tensor. Previous works either provided an analytical approach for a 3-DOF non-holonomic mobile robot, or provided a full 6-DOF approach but with an interaction matrix estimated numerically rather than analytically.

The experiments showed that this approach is working practically with very satisfactory results. In addition, this approach does not suffer from some of the problems existing in IBVS methods as like the retreat problem.

5.2 Future Work

5.2.1 Using A Subset of the Trifocal Tensor Coefficients

As presented in 4, it was found that using all the trifocal tensor coefficients inside the visual servoing loop does not produce the best results in terms of perfectly exponentially decreasing tensor error values. Thus, an automatic optimality selection criteria is needed to dynamically choose the most relevant tensor coefficients. For example, the singular value decomposition of the interaction matrix can reveal which degrees of freedom are most apparent. By selecting features and designing controllers that maximize these measures, the performance of the visual servoing system can be improved. These concepts has been discussed in details in [6], [17] and [18].

5.2.2 Stability Analysis

Simulation and practical experiments are not enough for judging the proposed method. We need to prove the existence of only one equilibrium state at the desired pose location. The analysis of the control law can be studied using Lyapunov stability analysis method [19].

5.2.3 Estimating The Initial Pose

We observe in (3.19) that the interaction matrix depends on the knowledge of the initial camera pose. In our experiments, we made an assumption that the initial pose is known from the setup of the simulation. Practically, this information might not be available, and the initial pose needs to be estimated. The initial pose can be retrieved by computing the equivalent projection matrix for the estimated trifocal tensor as shown in 2.1.3. However, one potential issue is that the estimation will be up to a scale value and not the real value. This scale problem needs to be addressed properly during the tensor normalization step to avoid the scale effects on computing the correct values for the interaction matrix.

5.2.4 Uncalibrated Camera

The current method assumes the cameras have already been calibrated before-hand. Practically, this may not be the case, so we need to extend the method to cover partially or totally uncalibrated cameras. The introduction of the camera intrinsic parameters matrix K into the formulas, will affect the obtained tensor computation, and hence, the corresponding interaction matrix.

Bibliography

[1] T. Rowland and E. W. Weisstein, "Tensor," http://mathworld.wolfram.com/Tensor.html, last visited on 03/02/2015.

[2] C. Stover and E. W. Weisstein, "Einstein summation," http://mathworld.wolfram.com/EinsteinSummation.html, last visited on 03/02/2015.

[3] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[4] P. H. Torr and A. Zisserman, "Robust parameterization and computation of the trifocal tensor," *Image and Vision Computing*, vol. 15, no. 8, pp. 591–605, 1997.

[5] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82–90, 2006.

[6] ——, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[7] E. Malis and F. Chaumette, "2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement," *International Journal of Computer Vision*, vol. 37, no. 1, pp. 79–97, 2000.

[8] G. Chesi, D. Prattichizzo, and A. Vicino, "A visual servoing algorithm based on epipolar geometry," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, 2001, pp. 737–742 vol.1.

[9] S. Benhimane and E. Malis, "Homography-based 2d visual servoing," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 2397–2402.

[10] G. Lopez-Nicolas, C. Sagues, J. Guerrero, D. Kragic, and P. Jensfelt, "Nonholonomic epipolar visual servoing," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 2378–2384.

[11] H. Becerra and C. Sagues, "Pose-estimation-based visual servoing for differential-drive robots using the 1d trifocal tensor," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* IEEE, 2009, pp. 5942–5947.

[12] G. López-Nicolás, J. J. Guerrero, and C. Sagüés, "Visual control through the trifocal tensor for nonholonomic robots," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 216–226, 2010.

[13] A. Shademan and M. Jagersand, "Three-view uncalibrated visual servoing," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.* IEEE, 2010, pp. 6234–6239.

[14] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *Robotics Automation Magazine, IEEE*, vol. 12, no. 4, pp. 40–52, Dec 2005.

[15] B. J. Nelson and P. K. Khosla, "Vision resolvability for visually servoed manipulation," *Journal of Robotic Systems*, vol. 13, no. 2, pp. 75–93, 1996.

[16] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control.* Springer, 1998, pp. 66–78.

[17] B. Nelson and P. Khosla, "Force and vision resolvability for assimilating disparate sensory feedback," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 714–731, Oct 1996.

[18] R. Sharma and S. Hutchinson, "Motion perceptibility and its application to active vision-based servo control," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 4, pp. 607–617, Aug 1997.

[19] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control.* Wiley New York, 2006, vol. 3.