TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Echtzeitsysteme und Robotik

6D Visual Servoing for Industrial Manipulators applied to Human-Robot Interaction Scenarios

Caixia Cai

Vollständiger Abdruck der von der Fakultät der Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Hans Michael Gerndt
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. habil. Alois Chr. Knoll
	2. Prof. Seth Hutchinson, University of Illinois, USA

Die Dissertation wurde am 20.09.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 16.03.2017 angenommen.

Abstract

This thesis presents a novel vision-based control system for industrial manipulators. The system is composed of a 6 DOF manipulator coupled with a stereo vision system using the eye-to-hand configuration. The robot is controlled using an uncalibrated visual servoing (VS) approach that addresses the challenges of choosing proper image features and designing a VS controller to enhance the tracking performance. The primary contribution of this thesis is the development of a new 6D visual servoing system and its integration into real-world scenarios. The proposed system consists of 2 parts: a visual servoing system based on a novel stereo camera model employing virtual orthogonal cameras to create a new *virtual visual space*, and a prioritized multi-constraints control framework where multiple constraints from the robot's structure and the external environment need to be simultaneously controlled while accomplishing the robot's main task.

In this work, a 6D pixel pose vector is extracted in the new virtual visual space. It represents the image positions as 6 linearly independent and orthogonal features, which are used as inputs for visual servoing instead of the classical visual features. The proposed new feature vector has good decoupling and linearizing properties, leading to a full-rank image Jacobian which allows avoiding classical problems, such as image space singularities and local minima. Moreover, simulation results with an eye-to-hand robotic system confirm the improvement in controller stability and motion performance with respect to classical visual servoing approaches. Further, by integrating the VS system with environment and robot model constraints in real world applications, we demonstrate that this work can be easily and safely integrated into a variety of robotic systems involving human-robot interaction.

Zusammenfassung

Diese Arbeit präsentiert ein neuartiges und robustes System zur bildgestützten Steuerung von industriellen Manipulatoren. Das System besteht aus einem Manipulator mit 6 Freiheitsgraden und einem Stereo-Kamera-System in einer Auge-zu-Hand-Konfiguration. Der Roboter wird durch einen unkalibrierten Visual Servoing Ansatz gesteuert, der sich mit der Herausforderung befasst, geeignete Bildmerkmale auszuwählen und die Performanz der Auswertung zu verbessern. Der primäre Beitrag dieser Arbeit ist die Entwicklung eines neuen 6-dimensionalen Visual Servoing Systems und seiner Integration in anwendungsnahen Szenarien. Das vorgestellte System besteht aus 2 Teilen: einem Visual Servoing System, welches auf einem neuartigen Stereo-Kamera-Modell basiert, das virtuell-orthogonale Kameras verwendet um einen virtuellen Bildraum zu beschreiben, und einer priorisierenden Robotersteuerung, die während der Ausführung der Hauptaufgabe mehrere Nebenbedingungen berücksichtigt, welche aus dem Aufbau des Roboters und seiner Umgebung abgeleitet werden.

Anstelle von klassischen Bildmerkmalen wird in dieser Arbeit ein 6D-Lagevektor im neuen virtuellen Bildraum extrahiert und als Merkmal für den Visual Servoing Ansatz genutzt. Jede Komponente des Lagevektors ist orthogonal und unabhängig und weist gute Entkoppelungs- und Linearisierungseigenschaften auf. Dies führt zu einer Bild-Jakobischen mit vollem Rang, die klassische Probleme, wie z.B. Singularitäten im Bildraum oder lokale Minima, umgeht. Zudem zeigen Simulationsergebnisse mit einem Auge-zu-Hand-Robotersystem, dass die Stabilität der Steuerung und die Bewegungsperformanz im Vergleich zu klassischen Visual Servoing Ansätzen verbessert wird. Durch die Integration des Visual Servoing Systems, unter Berücksichtigung von Umgebungs- und Robotereinschränkungen, wird gezeigt, dass die Ergebnisse dieser Arbeit leicht in eine Vielzahl bestehender Robotersysteme und Anwendungen übertragen werden können.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Alois Knoll for the opportunity to conduct my research in his esteemed group as well as for his encouragement and support. I would like to thank all my colleagues at the Robotics and Embedded Systems group and fortiss for their intellectual support. A special thanks to Prof. Seth Hutchinson to be my second supervisor to review my thesis.

Furthermore, I would also like to special thank Amy Bücherl, Gisela Hibsch, Gertrud Eberl and Ute Lomp for their support and co-operation in all administrative tasks. In addition I would also like to thank them for creating a healthy and friendly work environment.

Special thanks to my mentors, Dr. Suraj Nair and Dr. Emmanuel Dean with whom I have been working on the research topics. A special thanks to Dr. Suraj Nair, Nikhil Somani and Alexander Perzylo for proof reading the thesis. Finally, I would like to thank my friends and family, especially my parents, for the constant support and encouragement during the work on this thesis.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

In the quest for creating robust, precise and flexible robotic systems, significant efforts have been dedicated in ongoing robotic research towards the optimal use of sensors. Sensor-based approaches have proven especially useful in dealing with unstructured environments. Sensors extract information about the robot's interaction with the environment. This information can be incorporated in the robot's control loop to modify/adapt its behavior accordingly.

Several sensors have been developed over the past years to fit the requirements of diverse tasks, which have improved the performance of robots to a large extent. The most common sensors in robotics are force sensors, range finders, tactile sensors and vision sensors. Among these, vision sensors (e.g. camera) are the most popular, since visual sensors mimic the human sense of vision, and allow for contact-less measurement of the environment. This idea is motivated by the observation that humans and animals use primarily their visual perception to be able to interact with the environment.

Using a vision-based control system, a robot can perceive and react in complex and unpredictable surroundings. Many different approaches, adaptations and improvements for vision-based control have been developed over decades of research. The accuracy of the conventional 'look-then-move' approaches depend directly on the accuracy of the visual sensor and the robot manipulator. An alternative for increasing the accuracy of these sub-systems is to use a *visual-feedback* control loop which increases overall the accuracy of the system. Machine vision can provide closed-loop position control for a robot end-effector − this is referred to as *visual servoing* (VS).

Visual servoing is a technique for robot control, where visual feedback is used to close the robot control loop in order to increase the system accuracy. It is the fusion of results from many elemental areas including computer vision, kinematics and dynamics, control theory and

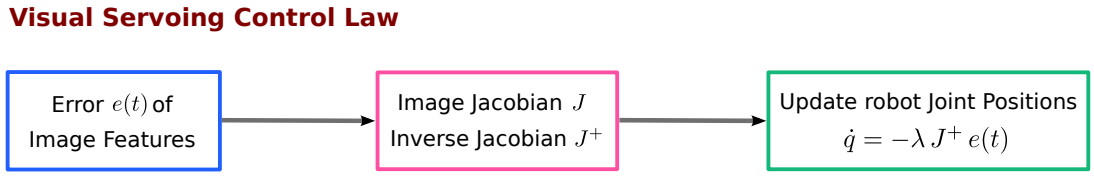


Figure 1.1: Vision-based control law.

optimization techniques. It can be used in many different domains, such as industrial, healthcare, service, space, etc. Visual servoing is an effective tool for handling the unstructured environments that are common for field and service robots. It also provides the potential to relax the mechanical accuracy and stiffness requirements for robot mechanisms and hence reduce their cost. The deficiencies of the mechanism would be compensated for by a vision sensor and feedback so as to achieve the desired accuracy.

A robotic task in visual servoing is specified in terms of image features extracted from a target object and their use for controlling the robot/camera motion through the scene (Fig. 1.1). Therefore, there is a large collection of research concerning the robustness with respect to selected features, structure of the employed control scheme, existence of errors and uncertainty in robot or camera calibration, and errors and uncertainty in input signals and in the object model. A vision sensor provides a large spectrum of potential visual features. It is possible to design visual servoing using both 2D features, such as the coordinates of points on the image, and 3D features, e.g. Cartesian pose. Based on the combinations of the choice of feature selection and control scheme, different behaviors can be obtained by the robot system. This wide range of possibilities is the reason behind the major difficulty in visual servoing, that is, to optimally build, select and design the visual data needed and the control scheme used for obtaining a suitable behavior of the system.

Using vision in robot control makes it possible to solve different problems based on the sensory visual data without any contact with the environment. However, some issues should be considered when vision sensors are used within robot system. These issues include designing maximal decoupled visual features, keeping features in the camera field of view at all times, and occlusion avoidance. The link between the visual features and the robot degrees of freedom, i.e., the interaction matrix (from vision to action), should ensure avoidance of singularities and local minima. The robot controller should ensure reliable convergence, global stability and robustness towards different types of noise.

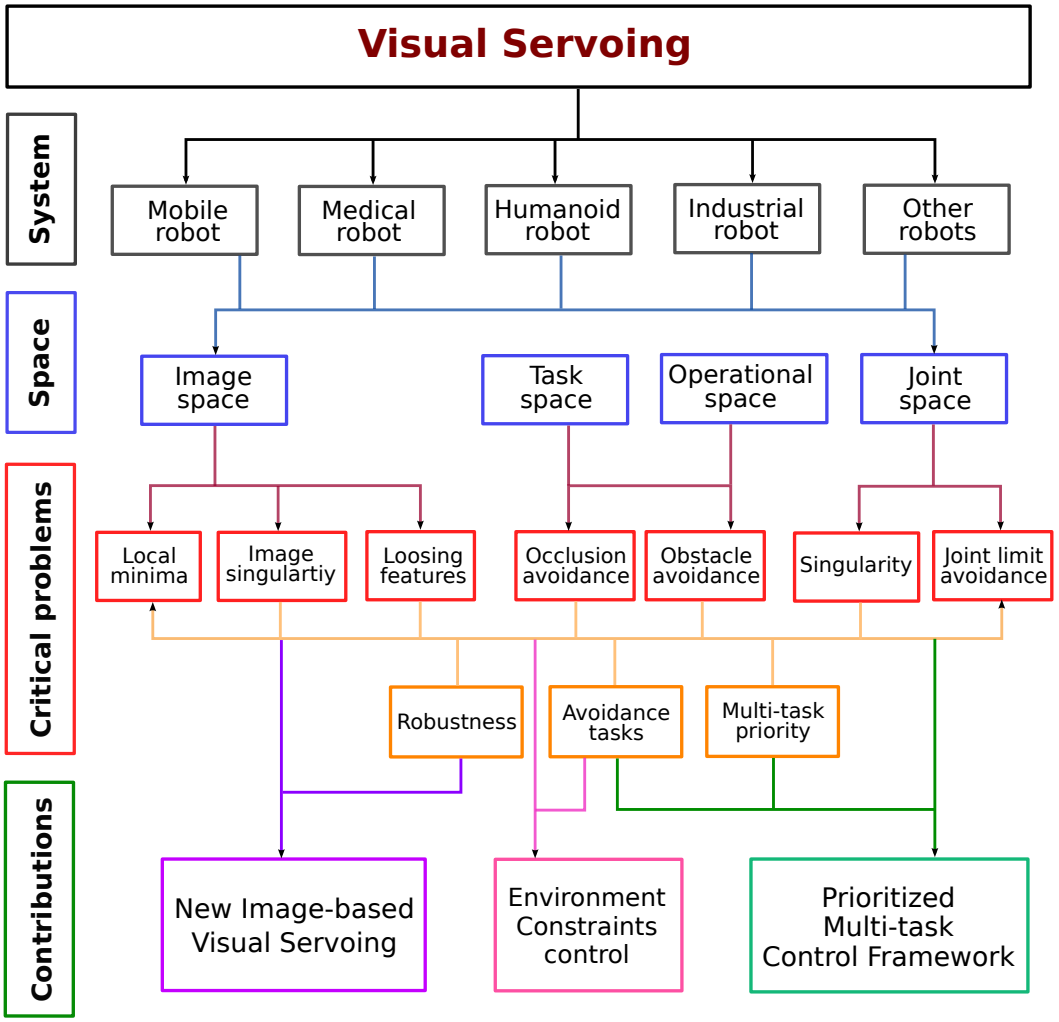


Figure 1.2: Thesis scope: problem domains and contributions.

1.1 Motivations

Visual servoing is a technology relevant for nearly all robotic systems that perform positioning and interaction tasks, and can be used for a wide variety of applications and on different robotic platforms, including industrial manipulators, mobile robots, medical robots, and humanoid robots (see Fig. 1.2). It involves working with different spaces in order to acquire the input signal, and transform a motion from one space to another in order to perform the required tasks. Visual servoing is a classical research problem in robotics, and encounters different sorts of problems: local minima, image singularities and loss of features from the camera field of view,

etc. (Fig. 1.2). The most effective methods for tackling these issues involve selecting the feature sets or modeling better suited control schemes. This motivates the research in extracting new ***image features*** and modeling new visual servoing ***control schemes***.

In order to use such visual servoing approaches in realistic applications, especially those involving human-robot interaction, the system must incorporate additional constraints/requirements arising from the robot or its environment. These requirements can be modeled using constraint-based approaches, also known as ***constrained problems***. The most frequently used method to define a reactive control task is to apply artificial attractive or repulsive potential fields and to use their gradients as control inputs τ. There are also approaches that model such tasks/restrictions in terms of their geometric properties. The control inputs from each task/constraint can also be combined using null-space projections. These types of problems form the motivation for the presented work.

1.2 Contributions

Image measurements are either used directly in the control loop or used for relative pose estimation of a workpiece with respect to a camera. The number of degrees of freedom (DOF) to be controlled by the employed control scheme determines the minimum number of independent features required. Therefore, it is desirable to choose features which are maximally decoupled, and at the same time facilitate a linear relation to the controlled degrees of freedom. Our contributions are algorithms and models for selecting such visual features for visual servoing system in oder to control robot.

The first and main contribution of this thesis is that we design six independent and orthogonal visual features, such that the corresponding interaction matrix (image Jacobian) has a maximal decoupled structure, without any image singularities. These new features improve the robustness and the stability of the system. Also, when redundant image point coordinates are used, the resulting coupled features might potentially lead to unstable or erratic robot motions. Hence, we propose the use of six independent image features to control 6D pose of the robot end-effector, in order to avoid such problems.

The second major contribution of this work is a control framework for industrial robots, that is necessary for integration of our visual servoing approaches in real-world applications involving human-robot interaction. In this framework, multiple tasks (expressed as constraints, e.g. desired operational positions and/or contact forces) with different priorities, along with limitations arising from the environment (e.g. obstacle avoidance) and robot model (e.g. joint

limits), can be simultaneously controlled. The tasks/constraints are modeled using geometric constraints or artificial potential fields. The combination of different constraints can be performed via direct composition of torques or using null-space projections.

1.3 Thesis Outline

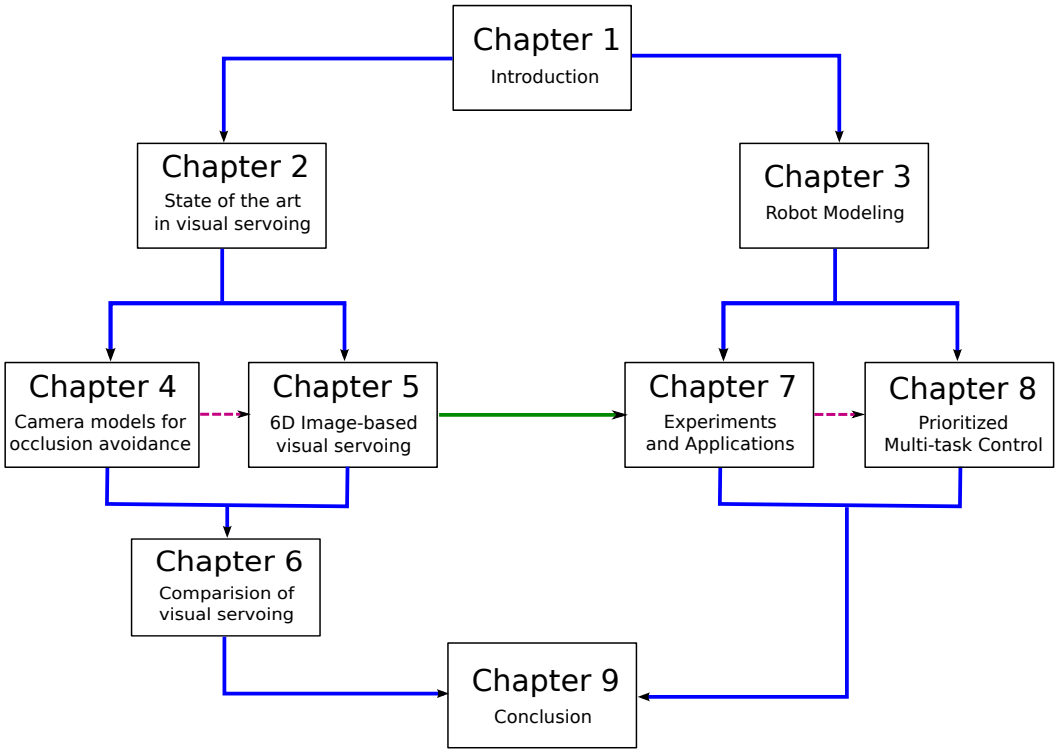


Figure 1.3: Thesis structure.

The thesis is written in two parts after the introductory chapter (see Fig. 1.3). The first part focuses on the design of new image features for visual servoing, (described in chapters 2 through 6). It presents a method to determine the analytical form of the interaction matrix related to these new image features, which is used to design a new 6D visual servoing. The merits of the proposed method vis-à-vis other classical schemes are validated in simulation. The second part of this thesis (chapters 7 and 8) focuses on the control framework to model the environment and task constraints that are necessary to safely integrate the proposed visual servoing into a human-robot interaction scenario. Finally, a summary of this work, the conclusions and directions for future work are presented in chapter 9. The details of the subsequent chapters are listed below.

- **Chapter 2: State of the art of Visual Servoing.** In this chapter, state of the art of visual servoing in terms of the used visual features, control schemes and problems that could appear is presented.

- **Chapter 3: Robot Modeling.** The kinematics and dynamics model of the industrial manipulator StäubliTX90 are presented in this chapter, which will be used in this thesis.

- **Chapter 4: Camera Models for Occlusion Avoidance.** In this chapter, two different solutions are provided for the visual occlusion problem in PBVS (e.g. out of camera field of view, visual occlusion by environments). Two different camera models are presented in these solutions to obtain the 3D Cartesian position for the objects. The properties of both camera models are validated on a real industrial robot. The proposed camera models are integrated in a human-robot interaction scenario with dynamics environment constraints using virtual impedance control [1].

- **Chapter 5: 6D Visual Servoing (6DVS).** This chapter focuses on novel image features for VS, where a 6D pixel pose vector is extracted from the visual information provided by two uncalibrated stereo cameras. This 6D pose vector represents the image positions as six orthogonal and independent signals, which are used as inputs for visual servoing instead of the classical visual features. A 6×6 image Jacobian is generated by using these new visual features, which allows to avoid local minima and image singularities, which are common problems in IBVS. Then a new visual servoing scheme, 6DVS, with better performance and properties is proposed. This chapter is the main contribution of the thesis, and its content is also presented in our publications [2, 3, 4].

- **Chapter 6: Comparison.** In this chapter, we compare the proposed 6D visual servoing (6DVS) with classical visual servoing methods (IBVS, PBVS and HYVS). Several standard tests are used to evaluate and compare the approaches in terms of steady state errors, transient systems performance, robustness to uncertainties and decoupling of controlled signals. The results of the evaluation prove the novel properties and better performance of the proposed 6DVS with respect to conventional VS approaches. The evaluation results are published in the paper [5].

- **Chapter 7: Experiments and real Applications.** In this chapter, experiments are performed to evaluate the proposed scheme on a standard industrial robot in a realistic human-robot interaction (HRI) scenario. In real experiments, a framework integrating

the proposed 6DVS with environment constraints [3, 4] is introduced for addressing safety issues of interaction tasks. Several constraints such as avoidance of robot singularities and collisions are integrated in this framework.

- **Chapter 8: Constrained Problems.** In this chapter, a prioritized, multi-task control framework is presented, that can realize force-based tasks in systems for industrial robots. In this framework, multiple tasks and constraints can be simultaneously controlled, and different priorities for tasks are accomplished through null-space projections [6]. Finally the proposed framework is evaluated for several typical industrial robotics applications like grasping, welding, erasing and peg-in-hole.

Chapter 2

State of the art of Visual Servoing

2.1 Visual Servoing

Visual servoing (VS) or vision-guided servoing refers to the use of vision in the low-level feedback loop of a robotic controller. Fast image processing is employed to provide reactive control behavior [7, 8, 9, 10, 11]. The task of visual servoing for robotic manipulators is to control the pose of the robot's end-effector relative to either a world coordinate frame or an object being manipulated, using real-time visual features extracted from the image [12, 13]. The error function (or task function) in visual servoing is defined as an error of the current visual features and the desired features [12, 13]. The aim of all vision-based control schemes is to regulate this error $e(t)$ and drive it to zero. Single or multiple cameras can be used to obtain visual information from a target in order to control a robot [14, 7, 13, 9, 15].

According to the features (s) used as feedback in minimizing the positioning error, VS is classified into Position-based Visual Servoing, Image-based Visual Servoing [16, 7, 9, 10, 11] and Hybrid Visual Servoing (e.g. 2-1/2D VS [17]). In Position-based servoing (PBVS or 3D VS), features are extracted from the image and used in conjunction with a geometric model of the target and the known camera model to estimate the relative 3D Cartesian pose of the target with respect to the camera. Feedback is computed by reducing errors in the estimated pose space [18, 19]. In image-based visual servoing (IBVS or 2D VS), the error is directly computed in terms of features expressed in image space [7, 20, 12]. Two aspects have the most significant impact on the behavior of any visual servoing scheme: the selection of visual features used as input of the control law, and the designed form of the control scheme. On one hand, the same feature set gives different behavior when employed in different control schemes and on the other

hand, the same control law gives different behavior when used with different feature sets [21]. The behavior obtained with combinations of these choices is often not as desired: selecting a specific feature set or a specific control scheme may lead to some stability and convergence problems. In the remainder of this chapter, a review in visual servoing is presented by focusing on such modeling issues.

2.2 Camera-robot Configurations

In visual servoing, there are two main configurations for combining the camera(s) and robot. The vision sensor can either be mounted on the robot (*eye-in-hand configuration*) or observing it (*eye-to-hand configuration*) [9, 11, 15]. For the same robot motion, the motion produced in the image will be opposite from one configuration to the other, see Fig. 2.1.

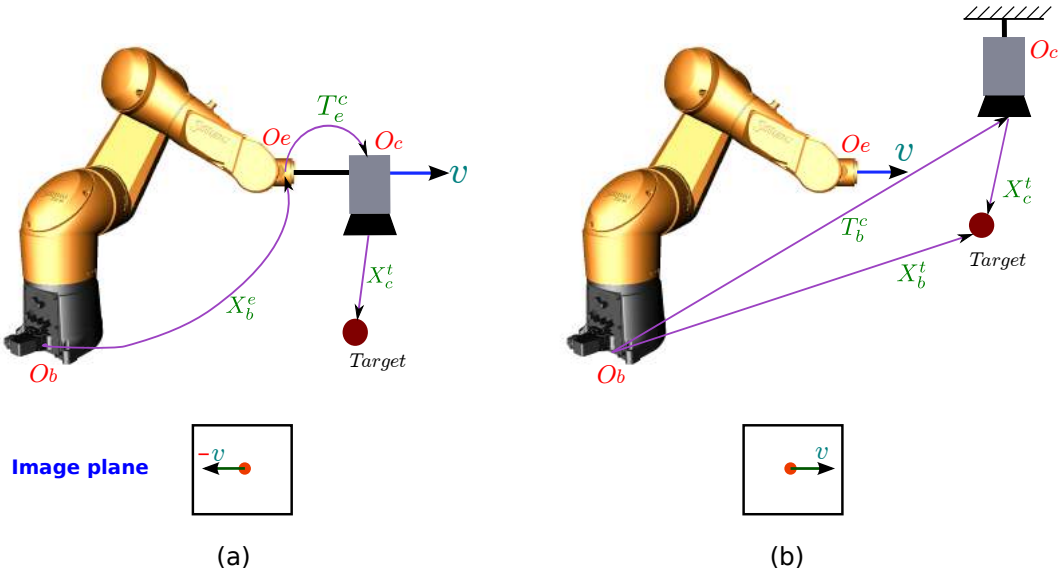


Figure 2.1: Top: (a) Eye-in-hand system, (b) Eye-to-hand system. Bottom: Opposite image motion produced by the same robot motion.

The first *eye-in-hand* configuration (see Fig. 2.1 (a)), has the camera is mounted on the robot's end-effector. Hence, there exists a known, often constant transformation T_e^c between the camera frame O_c and the end-effector frame O_e [22]. The pose of the target relative to the camera frame O_c is represented by X_c^t and the pose of the end-effector with respect to robot base frame O_b is X_b^e.

In the second *eye-to-hand* configuration, one or several cameras are placed in the workspace

to monitor a target, end-effector or both (see Fig. 2.1 (b)). In this configuration, the transformation matrix T_b^c between the camera frame O_c and the robot base frame O_b is constant and is computed once. The transformation between camera frame and robot end-effector frame need to be computed at each iteration using transformations between different frames. In this case, the camera image of the target is independent of the robot motion and the poses of the target with respect to camera frame O_c and robot base frame O_b are represented as X_c^t and X_b^t, respectively.

For either choice of configuration, camera calibration must be performed in order to have the set of camera intrinsic parameters. For the eye-in-had case, this amounts to determining T_e^c. For the fixed camera case, calibration is used to determine T_b^c. The former configuration is commonly used in visual servoing since it allows keeping target(s) in the field of view (e.g., when a grasping task requires ensured monitoring of the object and a grasping tool attached to the end effector [23]). Hybrid configurations can be constructed where eye-to-hand and eye-in-hand configurations are simultaneously used [24]. Within the scope of the work presented in this thesis, we focus on the **eye-to-hand** camera-robot configuration.

2.3 Selection of Visual Features

Visual servoing explicitly relies on the choice of the visual features s, that is the key point of this approach. Visual features observed by the vision sensors define the inputs to the control scheme in visual servoing. Vision sensors can be conventional 2D cameras (as often used in visual servoing), 2D ultrasound cameras [25] or omni-directional cameras that find motivation in robotics applications to avoid visibility problems due to the restricted field of view of conventional 2D camera [26, 27]. A vision sensor provides numerous of potential visual features. However, if no planning strategy is developed, the use of some visual features as input of the control scheme may lead to stability problems if the displacement that the robot has to achieve is very large [28]. Therefore, selection of good visual features is a crucial aspect of visual servoing as it is necessary for achieving optimal speed, accuracy, and reliability of image measurements. Consequently, it affects performance and robustness of visual servoing [29, 30, 28]. For this reason, we need to design ideal visual features for visual servoing, which should satisfactory the following criteria: local and-as far as possible-global stability of the system, robustness to calibration and modeling errors, non-singularity, local mimima avoidance, satisfactory trajectory of the system and of the features in the image, and, finally, a maximal decoupled and linear link (the ultimate goal) between the visual features and the degrees of freedom (DOFs) taken into account.

2.3.1 Geometric Features

Imaging measurements from vision sensor are either used directly in the control loop or used for relative pose estimation of a workpiece with respect to a camera. The most common visual features are the geometric features (e.g. points, segments, straight lines or spheres). Geometric features are defined to describe the geometrical contents of an image (2D visual features) or to relate a frame attached to a robot system with a frame attached to an object in a scene (3D visual features). Both 2D and 3D visual features can be used simultaneously in a hybrid form.

2D Visual Features

2D visual features are normally extracted in 2D image as points, lines, ellipses, region of interest or contours [29, 12, 31, 30, 32]. These features are defined from image measurements. In case of image points, Cartesian coordinates are generally used but it may be also possible to use their polar and cylindrical coordinates [33].

Besides, image moments and moment invariants can also be used in visual servoing [34, 35, 36, 37, 38, 39, 40]. Using image moments, the improvements with respect to classical visual servoing seem to be significant, since it allows a generic representation that is not only able to handle simple geometrical primitives, but also complex objects with unknown shapes. In [35] it discusses the use of image moments to formulate the visual Jacobian. This formulation allows for decoupling of the DOF based on type of moments chosen. It is shown in [37] that moment invariants can be used to design a decoupled 2D visual servoing scheme.

3D Visual Features

3D Visual features can also be selected in 3D Cartesian space such as pose or coordinates of 3D points [41, 19, 42]. Usually, object model and image measurements are used to compute or to estimate the relative pose between object and camera frames in the Cartesian space. In [43], the 3D coordinates of the points of the object are used as the feature vector. A priori knowledge about the camera calibration parameters is required. In PBVS, orientation in pose vector can be represented by the total format, roll-pitch-yaw or axis-angle formats [44] or quaternions [45].

Hybrid Visual Features

Several combinations between visual feature types can also be considered: e.g., a mixture of both kinds of 2D and 3D features is presented in [17, 46, 43, 47] and Polar and Cartesian parameterizations of image points coordinates as presented in [48].

Redundant Features

Utilizing redundant features can improve the performance of visual servoing control and increase the positioning accuracy by improving the corresponding minimum singular value of the extended image Jacobian [49]. However, processing a large feature set can sometimes be computationally expensive (or even infeasible). Therefore, the focus must be on the selection of an information-intensive and reliable set that possesses a minimum number of features.

2.3.2 Photometric Features

Contrary to most of related works in this domain where geometric visual features are usually used, photometric features computed from pixel intensities have been to used in visual servoing recently. The utilization of the photometric features does not rely on complex image processing such as feature extraction, matching, and tracking. Moreover, it is not very sensitive to partial occlusions and to coarse approximations of the depths required to compute the interaction matrix. This approach is realized by considering the whole image as a feature set from which the control input signal is defined [50, 51]. The input to the controller can belong to the eigenspace or kernel of an image pixel or can also be defined as the set of all image pixels itself.

In [50], image intensity is not used directly but an eigenspace decomposition is performed first to reduce the dimensionality of image data. The control is then performed in the eigenspace and not directly with the image intensity. Moreover, this way to proceed requires the off-line computation of this eigenspace and the projection of the image on this subspace for each new frame. An interesting approach, which also consider the pixels intensity, has been recently proposed in [52]. This approach is based on the use of kernel methods that lead to a high decoupled control law. It is also possible to use the luminance of all the pixels in the image as visual feature set [53, 54, 55].

2.3.3 Velocity Field Features

The velocity field in the image is chosen as visual features in [56], and the relation between the variations of velocity field features and the camera velocity is modeled. This approach is used for positioning a camera parallel to a plane and following a trajectory. The camera motions are controlled so that the current velocity field in the image becomes equal to motion field in the desired configuration. In [57], the application of the velocity field control is applied to visual servoing of a robot manipulator under fixed camera configuration. The desired velocity field v_d is defined in the image space, which defines a tangent vector that represents the desired image

feature velocity $\dot{\mathrm{x}}$ at every point of the image space. The velocity field error e is defined as the difference between the desired velocity field $\mathrm{v}_d(\mathrm{x})$ and the image feature velocity $\dot{\mathrm{x}}$. The velocity field is also used in [58] for controlling wheeled nonholonomic mobile robots by a fixed camera.

2.4 Error Functions in Visual Servoing

Classically, to achieve a visual servoing task, a set of visual features has to be selected from the image allowing to control the desired degrees of freedom. A control law has also to be designed so that these visual features s reach a desired value s_d, leading to a correct realization of the task. The control principle is thus to regulate the error vector $s - s_d$ to zero. To build this control law, the knowledge of the interaction matrix L_s is usually required.

2.4.1 Cartesian Space Control

As described in the visual servoing control tutorial [10], all visual servoing tasks can be expressed as the regulation to zero of an error $e(t)$ which is defined by

$$e(t) = s(\mathbf{m}(\mathbf{r}(t)), a) - s_d. \tag{2.1}$$

The vector $\mathbf{m}(\mathbf{r}(t))$ is a set of image measurements (e.g. the image coordinates of interest points, or the area, the center of gravity and other geometric characteristics of an object). These image measurements depend on the pose $\mathbf{r}(t)$ between the camera and the environment. They are used to compute a vector $s(\mathbf{m}(\mathbf{r}(t)), a)$ of visual features, in which a is a set of parameters that represent potential additional knowledge about the system (e.g. coarse camera intrinsic parameters or 3D model of objects). The vector s_d contains the desired value of the features, which can be either constant in the case of a fixed goal, or varying if the task consists in following a specified trajectory.

The design of the control scheme can be quite simple, the most straightforward approach is to design a velocity controller. For eye-in-hand systems, this matrix links the time variation of s to the camera instantaneous velocity v is given by:

$$\dot{s} = \mathbf{L}_s \, \mathbf{v}_o^c + \frac{\partial s}{\partial t} \tag{2.2}$$

where $\frac{\partial s}{\partial t}$ is the variation of s due to the object own motion, $L_s \in \mathbb{R}^{k \times n}$ is the visual features interaction matrix that represents the differential relationship between the features s and the camera frame [9, 59], $\mathrm{v}_o^c = \mathrm{v}_c - \mathrm{v}_o$ is the relative instantaneous velocity between the camera

frame O_c and the object frame O_o, v_c is the instantaneous camera velocity and v_o is the instantaneous object velocity. When the object is motionless, $\frac{\partial s}{\partial t} = 0$ and $\mathrm{v}_o^c = \mathrm{v}_c$, then we get:

$$\dot{s} = \mathbf{L}_s \, \mathbf{v}_c. \tag{2.3}$$

2.4.2 Joint Space Control

According to the robot velocity kinematics, the relation between $\dot{s}$ and the velocity of the joint variables $\dot{q}$ can be obtained as follows [59, 60]:

$$\dot{s} = \mathbf{L}_s \, \mathbf{V}_c^e \cdot J_q \, \dot{q} + \frac{\partial s}{\partial t} \tag{2.4}$$

where J_q is the Jacobian of the robot expressed in the end effector frame O_e and V_c^e is the transformation matrix to map the velocities expressed in robot end-effector frame O_e to the camera frame O_c, defined by [59]:

$$\mathbf{V}_c^e = \begin{bmatrix} R_c^e & sk(t_c^e)R_c^e \\ 0 & R_c^e \end{bmatrix} \tag{2.5}$$

in which $sk(*)$ is the *skew-symmetric matrix* and $T_c^e = [R_c^e, t_c^e]$ is the transformation between the end-effector frame and the camera frame. The matrix $\mathbf{V}_c^e$ remains constant for eye-in-hand configurations, while it has to be estimated at each iteration for eye-to-hand configurations.

If the object is not moving, $\frac{\partial s}{\partial t} = 0$, from (2.17) we get:

$$\dot{s} = J_s \, \dot{q} \tag{2.6}$$

where $J_s = \mathbf{L}_s \, \mathbf{V}_c^e \, J_q$ is the Jacobian of the visual feature, known as *Visual Jacobian*.

2.4.3 Interaction Matrix

A lot of works have concerned the modeling of the visual features and the determination of the analytical form of the interaction matrix $\mathbf{L}_s$. To give just an example, in the case of an image point with normalized Cartesian coordinates $\mathbf{x} = (x, y)$ and whose 3D corresponding point has depth Z, the perspective projection model [61, 62] is shown in Fig. 2.2.

In this model, the center of projection is considered at the origin of the camera frame O_c and the image plane is at $Z = f$, where f is the camera focal length. By considering a 3D point with coordinates $\mathbf{X} = [X, Y, Z]^T$ in the camera frame and using a perspective projection model [63], the point $\mathbf{X}$ is projected on a 2D point $\mathbf{x}$ of coordinates (x, y) on the image plane such that:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} = \begin{bmatrix} (u - c_x)/f\alpha \\ (v - c_y)/f \end{bmatrix} \tag{2.7}$$

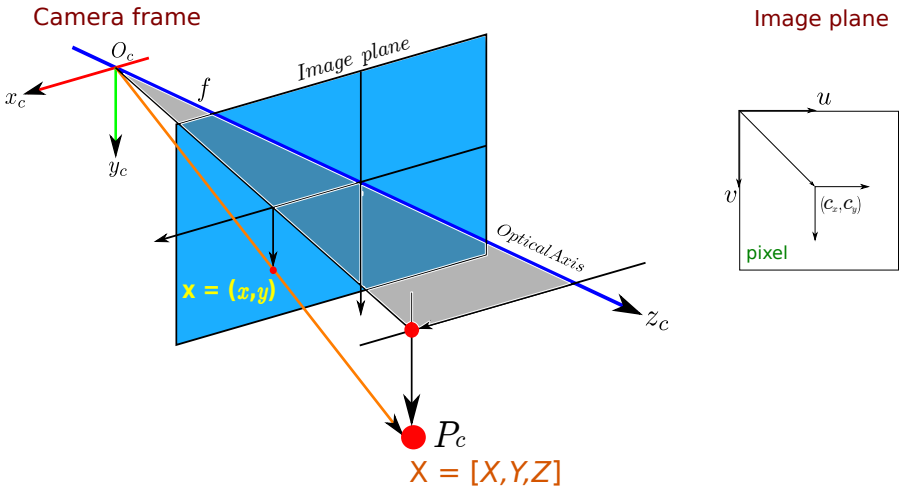


Figure 2.2: Pin hole camera model.

where $s = [u, v]$ is the image point coordinates in pixel unit, $c = [c_x, c_y]$ is the coordinates of the principle point, f is the focal length of the camera lens and α is the ratio of pixel dimension.

Interaction Matrix of Image Feature Points (2D)

As for the interaction matrix L_s of an image feature point it can be obtained as following. By taking the time derivative of (2.7) we get:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2}\\ \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2}\end{bmatrix} = \begin{bmatrix}\frac{1}{Z} & 0 & -\frac{X}{Z^2}\\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2}\end{bmatrix}\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{Z}\end{bmatrix} \tag{2.8}$$

In the eye-in-hand configuration, if the spatial velocity of the camera is given by $\mathbf{v_c} = (\upsilon, \omega)$ where $\upsilon = [\upsilon_x, \upsilon_y, \upsilon_z]^T$ and $\omega = [\omega_x, \omega_y, \omega_z]^T$ are the instantaneous linear and angular velocities of the origin of the camera frame both expressed in O_c, then the velocity of the 3D point $\mathbf{X}$ related to the camera velocity is defined using the fundamental kinematic equation $\dot{\mathbf{X}} = -\upsilon - \omega \times \mathbf{X}$ such that:

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{Z}\end{bmatrix} = \begin{bmatrix}-\upsilon_x - \omega_y Z + \omega_z Y\\ -\upsilon_y - \omega_z X + \omega_x Z\\ -\upsilon_z - \omega_x Y + \omega_y X\end{bmatrix} \tag{2.9}$$

$$= \left[\begin{array}{ccc|ccc}-1 & 0 & 0 & 0 & -Z & Y\\ 0 & -1 & 0 & Z & 0 & -X\\ 0 & 0 & -1 & -Y & X & 0\end{array}\right]\begin{bmatrix}\upsilon\\ \omega\end{bmatrix} \tag{2.10}$$

$$= \left[-I_3 \mid sk(\mathbf{X})\right]\begin{bmatrix}\upsilon\\ \omega\end{bmatrix} \tag{2.11}$$

where $sk(*)$ is the skew-symmetric matrix.

By injecting the values of $\dot{X}$, $\dot{Y}$ and $\dot{Z}$ from (2.10) in (2.8) and grouping for v, and ω we get the classical result [7]:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y\\ 0 & \frac{-1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{2.12}$$

$$\dot{\mathbf{x}} = \mathbf{L_x}\,\mathbf{v_c} \tag{2.13}$$

where $\mathbf{L_x}$ is the interaction matrix related to $\mathbf{x}$. If there is a set of k feature points $\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_k)$, the interaction matrix $\mathbf{L_x}$ of the set $\mathbf{x}$ is obtained by stacking L_{x_i} for all $\mathrm{x}_i \in \mathbf{x}$ to get:

$$\mathbf{L_x} = \begin{bmatrix}L_{\mathrm{x}_1}\\ \vdots\\ L_{\mathrm{x}_k}\end{bmatrix} \tag{2.14}$$

If the system has *eye-to-hand* configuration, then

$$\dot{\mathbf{X}} = v + \omega \times \mathbf{X} = \begin{bmatrix}I_3 & -sk(\mathbf{X})\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{2.15}$$

Hence, the interaction matrix L_{x} in equation (2.12) is

$$\mathbf{L_x} = \begin{bmatrix}\frac{1}{Z} & 0 & -\frac{x}{Z} & -xy & (1+x^2) & -y\\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -(1+y^2) & xy & x\end{bmatrix} \tag{2.16}$$

If it uses the pixel image point $s = [u, v]$ as the features, according to (2.7), equations (2.12), (2.13) can be rewritten as

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}\frac{f}{Z} & 0 & -\frac{u}{Z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v\\ 0 & \frac{f}{Z} & -\frac{v}{Z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{2.17}$$

$$\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v_c}. \tag{2.18}$$

2.5 Visual Servoing Control Schemes

Visual servoing control schemes mainly differ in the way that visual features s are designed. The control law [12, 19, 9, 28, 64, 65, 10, 11] affects the behavior of the system. In the control design phase, a number of properties should be considered such as local and global stability, robust behavior with respect to measurement and modeling errors, local or global exponential decrease, order of convergence, absence of local minima and singularities, obtaining suitable robot trajectory, and finally the degree of decoupling between the visual information and the controlled degrees of freedom.

2.5.1 Classical Approaches

The two most classical approaches are named image-based visual servoing (IBVS), in which s consists of a set of 2D parameters that are directly expressed in the image [7, 12], and position-based visual servoing (PBVS), in which s consists of a set of 3D parameters related to the pose between the camera and the target [7, 19].

2.5.1.1 Position-based Visual Servoing

In Position-based visual servoing (PBVS), features are extracted from the image and used to estimate the pose of the target with respect to camera. Feedback is computed by reducing errors in estimated pose space. In that case, the 3D parameters have to be estimated from the image measurements either through a pose estimation process using the knowledge of the 3D target model, or through a partial pose estimation process using the properties of the epipolar geometry between the current and the desired images, or finally through a triangulation process if a stereo vision system is considered [9, 19, 18, 66, 67, 42]. PBVS is known to have global asymptotic stability, i.e., the ability of a controller to stabilize the pose of the camera from any initial condition if 3D estimation is perfect. From the analytical proof, it is evident that this is true if the pose is perfect and impossible otherwise. When accurate 3D estimation is employed, decoupling rotation and translation is obtained. Errors in calibration propagate to errors in the 3D world, so accurate 3D estimation is essential to ensure robustness of PBVS [68].

No mechanism in PBVS ensures keeping the features visible within the camera field of view (FOV) when the translation is defined in the desired fixed end effector frame [19]. In comparison, if the translation is expressed in the camera frame, the trajectory in the image plane is improved under large camera rotation motion and features can be kept in the image plane for small rotations [42]. Several control schemes have been proposed to overcome the latter problem (for example 2.5 D visual servoing presented in [69], or nonlinear approach using a new 3D translation features in the control loop [70, 71]). In PBVS, the task function to be regulated is usually defined as the error between current and desired poses. The pose can also be selected as the pose of the camera or the end effector with respect to the object or any other reference frame in the world space.

Pose Estimation

Pose estimation is the key issue in position-based visual servoing, since small amount of image noise could lead to large errors in the pose estimation. The pose can be obtained from an

image of the object, the 3D CAD model of the object, or an estimation of the camera intrinsic parameters. When the pose is correctly estimated, this class of visual servoing is known to provide adequate system motion in the Cartesian space either in the moving camera frame or in a fixed frame [42]. The pose can be estimated using image points [72, 73, 74, 75, 76], using point and line correspondence [77], using point to region correspondence [78], using curves [79], or using other different geometrical features as in virtual visual servoing [80]. For obtaining more accurate pose estimation, different filters are often used to estimate its translational and rotational parameters, as shown in [81, 44, 82, 83, 84] and recently in [85].

Partial Pose Estimation

Camera translation and camera rotation can be estimated through the Essential matrix [86, 87, 88]. However, when the target is planar or when the motion performed by the camera between the desired and the current pose is a pure rotation, essential matrix cannot be estimated and it is more appealing to estimate the pose using a homography matrix [89].

2.5.1.2 Image-based Visual Servoing

In IBVS, error signal is defined on the basis of image features directly, without any supplementary estimation step. Different from PBVS, IBVS does not need to estimate the pose at each iteration which helps to provide a robust positioning control against calibration and modeling errors.

IBVS is characterized by several advantages. Firstly, direct control of the feature motion in the image plane allows the implementation of strategies aimed at keeping the target always in the field of cameras view, with approximately straight lines trajectories for image feature point [28, 90]. Another advantage of IBVS is that the positioning accuracy is insensitive to camera and target modeling errors [91, 9, 92, 93]. It is essentially a model-free method, without explicit requirement of the target model in practical applications and convergence is generally robust w.r.t. disturbances and uncertainties in the camera/robot model [94]. IBVS systems are usually preferred to position-based systems since they are usually less sensitive to image noise. However, some knowledge of the transformation between the sensor and the robot frame [22] is still required.

IBVS is however subject to some shortcomings. Firstly, it is hard to predict the trajectory of the end effector and the robot may reach its joint limits. Secondly, the end-effector translational and rotational motions are not directly controlled, and the usual coupling existing between

these motions makes it difficult to plan a pure rotational or a pure translational motion [28]. Also, since the system is usually highly coupled, the analytical domain of system stability is impossible to determine in the presence of camera modeling errors. Furthermore, usual IBVS is only locally asymptotically stable and may fail in the presence of large desired displacement [28, 95], which necessitates a path planning step to split a large displacement into smaller local movements [93]. Finally, potential failure occurs when IBVS is subject to image singularities or local minima [28].

Depth Estimation

In IBVS, providing some information about the depth of the object in the camera frame is usually necessary for the computations required to obtain the interaction matrix. Since the stability region for the error in depth estimation is not very large [96], it is necessary to accurately estimate the depth. For static objects, this estimation of the depth value can be obtained from the measurement of the current values of the feature points x and y and their image motion $\dot{x}$ and $\dot{y}$ [97, 98]. The depth parameters of planar and volumetric parametric primitives like points, lines, cylinders, spheres, etc. can be also obtained [99]. Another depth estimation method for static points without the explicit need for image motion estimation can be found in [100]. In [101], a laser pointer is used and the depth estimation can be achieved through a triangulation method.

Image Jacobian

The image interaction matrix (image Jacobian matrix), which is related to the image features, can be computed using direct depth information (depth-dependent Jacobian) [102, 103], or by approximation via on-line estimation of depth of the features (depth-estimation Jacobian) [10, 104, 105, 106], or using depth-independent image Jacobian matrix [107, 108, 109]. Additionally, many papers directly estimate on-line the complete image Jacobian in different ways [110, 111, 112, 113]. However, all these methods use redundant image point coordinates to define (as a general rule) a non-square image Jacobian, which leads to well-known problems such as the image singularities. In [35, 37, 38], using the image moments as features in visual servoing renders the corresponding image Jacobian matrix with a maximally decoupled structure, where the inherent problem-singularity of the interaction matrix is solved and the performance of IBVS is improved. However, this proposed approach is limited to planar symmetric objects.

2.5.2 Enhanced Visual Servoing Approaches

In order to overcome drawbacks of visual servoing control schemes, different modeling approaches have been considered such as sliding approaches [114], partitioning approaches [115, 90, 116, 117, 118], trajectory planning approaches [93, 119, 120], origin-shift in cylindrical coordinates [33], varying-feature-set approaches [121, 122, 123], and hybrid and switching approaches which will be discussed in a dedicated subsection 2.5.2.1.

2.5.2.1 Hybrid and Switching Strategies in Visual Servoing

To combine the advantages of both image-based (2D) and position-based (3D) visual servoing, numerous approaches have been proposed to model control schemes based on the utilization of hybrid and switching strategies. In addition to providing accurate control signal, research on hybrid visual servoing has focused on addressing issues like feature visibility, local minima avoidance, faster convergence, short camera path, continuous control signal, etc. Many of the hybrid methods address the above mentioned issues by integrating the 2D and 3D information in the feature space [116, 17, 43] or in the action space [124, 125, 126]. Switching approaches between PBVS and IBVS are used to utilize the strength of each one when the other is in a weaker configuration [125, 127, 128]. Hybrid and switching approaches are also used together by integrating PBVS and IBVS in a switching hybrid scheme [129, 124, 130, 131]. Finally, planning step-switching using laser spot is presented in [101].

2-1/2D Visual Servoing

The 2-1/2D visual servoing approach developed in [17, 89, 132, 133, 134, 116, 135] combines visual features expressed in 2D image and 3D Cartesian spaces. This hybrid controller tries to take advantage of both approaches by decoupling the rotation motion from its translational part considering one visible image point during the servoing process. This decoupling makes it possible to design two task functions: e_υ for the translation and e_ω for the rotation which improves the system behavior in the 3D space. A very interesting aspect in 2-1/2 D visual servoing is that, thanks to projective reconstruction, the knowledge of the 3D structure of the considered targets is no more necessary. However, the use of projection reconstruction implies that the corresponding control laws are more sensitive to image noise than classical image-based visual servoing. In this method, the task function is expressed in both Cartesian space and in the image space. The homography matrix that relates image points in the current and desired views is decomposed to extract the required rotation motion.

Switches Approaches

An algorithm which switches between image-based and position-based vision control algorithms is presented by Gans and Hutchinson [125, 127, 128]. In this hybrid switching method, the IBVS and PBVS run independently. Another principle of switching is presented by Chesi et al. [126]. It switches between elementary camera motions, mainly rotation and translation extracted by decomposing the homography matrix between the current and desired views. In these two switching methods, the control signal suffers from discontinuity when features approach the image border. They need large amount of time for convergence. Hafez and Jawahar [124] present a smooth linear combination of different visual servoing algorithms. The combining weights are computed using an error function of the weakness of the concern algorithm. Another hybrid method based on potential fields [93] is proposed for path planning in the image space. This method introduces the visibility and robot joint limits constraints into the design of the desired trajectories.

2.5.3 Dynamic Control Schemes

In the context of control schemes, kinematic-based controls cannot yield high performance and guarantee stability because the nonlinear forces in robot dynamics are neglected. This problem, known as *Dynamic Visual Servoing*, was studied in [136] for the eye-in-hand setup and for the fixed-camera configuration in [137]. Their methods work well when the camera intrinsic and extrinsic parameters are known. In order to avoid tedious and costly camera calibration, various authors [138, 139, 140, 141, 142] proposed to employ an adaptive algorithm to estimate the unknown camera parameters on-line. However, these methods are applicable to planar manipulators only. The problem of 3D uncalibrated visual servoing with robot dynamics has been tackled with new adaptive controllers [108, 143].

2.6 Problems in Visual Servoing

Selecting a suitable set of visual features and designing good control schemes should be taken into account for avoiding system failures. Most frequently encountered problems in visual servoing such as local minima, singularity and visibility problems are directly influenced by this choice, and can be enhanced by a better selection.

2.6.1 Local Minima

By definition, a local minimum is reached since the camera velocity is zero while the final cameras position is far away from its desired one. Local minima are defined such that $\mathbf{v} = 0$ and $s \neq s_d$ (or $\dot{s} \neq 0$). This is equivalent to:

$$s - s_d \neq \phi \in \text{Ker}(\mathbf{L}_s^+) \tag{2.19}$$

This results in convergence to a final pose that is different from the desired one.

If s is composed by three image points and $\mathbf{L}_s$ is full rank 6, then $\text{Ker}(\mathbf{L}_s^+) = 0$, which implies that there is none local minima. However, it is well known that the same image of three points can be seen from four different camera poses [144]. In other words, there exist four camera poses (that is four global minima) such that $s = s_d$. A unique pose can theoretically be obtained by using at least four points. However, in that case, $\dim(\mathbf{L}_s) = 8 \times 6$, implying that $\dim \text{Ker}(\mathbf{L}_s^+) = 2$. Using four points, the control law tries to enforces 8 constraints on the image trajectory while the system has only six degrees of freedom. In that case, due to the existence of unrealizable motions in the image that may be computed by the control law, a local minimum may be reached [28]. At the local minimum position, the error $s - s_d$ in the image do not completely vanish (residual error is approximately one pixel on each u and v coordinate). Introducing noise in the image measurement leads to the same results, which can also be obtained in real experiments.

Several control strategies have been used to avoid local minima in visual servoing. For example, in [129] and [128], a hybrid motion control strategy that explicitly considers the local minima problem is presented, while in [93] a path planning strategy is developed.

2.6.2 Singularity

It is well known that the image Jacobian (interaction matrix) may become singular during the visual servoing, if image points are chosen as visual features s. Let us consider that four points are used and the camera motion from its initial to desired poses is a pure rotation of 180^o around its optical axis. This 3D motion leads to an image motion corresponding to a symmetry around the principal point, and the interaction matrix is singular [28]. For the considered motion, the choice of image points coordinates is really inadequate. The singularity can be avoided when the same initial position is used, and a perfect camera trajectory (a pure rotation motion around the optical axis of the camera) can be achieved if straight lines are used in s instead of points. Indeed, the cylindrical parameters (ρ, θ) describe the position in the image of a straight

line [12].

Additional singularity configurations in IBVS have been described in the literature, such as: singularity of three points, singularity of a circle and singularity of the norm. The singularities in IBVS can be avoided in several ways for different situations. Usually, these singular configurations are avoided trivially by selecting features such that their interaction matrix is always of full rank. A secondary objective can be designed to try to avoid the singular configurations by the redundancy framework [145, 146]. Moreover, a classical approach in robotics is to use the damped-least-squares inverse [147, 148, 149, 150] instead of the Moore-Penrose pseudo inverse to reduce the effect of the singularity in terms of robustness. Finally, a regularization technique has recently been introduced in [151]. It also allows reducing the effect of the singularity, but with the price of decreasing the convergence speed, which is inefficient if the task consists of tracking a moving target.

2.6.3 Feature Visibility

Using classical 2D and 3D visual servoing and assuming a bad calibration and a large initial camera displacement, the target may leave the camera field of view [17, 152]. In general, there is no guarantee on the positioning accuracy of the system unless control points on both the end-effector and the target can be observed. Hence, it is desirable to have servoing controls able to always keep features in the camera field of view to obtain reliable feedback signal during the servoing process. To minimize the probability that the object leaves the FOV, several methods such as a repulsive potential field [93, 103, 152], a path planning strategy [93, 153], switching strategies [128], the changes of visibility in image features during the control task [122], using structure light [101], as well as intrinsic-free visual servoing approach[65] can be adopted.

2.7 Performance and Robustness

A dynamic performance of PBVS is presented in [42]. It considers different orientation formats in controlling the robot end effector in the desired and in the current end-effector frames. In [68], the effect of vision system properties to the performance of the control system are investigated by analyzing the propagation of image errors through pose estimation and visual servoing control law.

Quantitative performance metrics for specific tasks can be formulated using some metric measures as shown in [118]. These measures include the number of iterations required to converge, error at termination, maximum feature excursion, maximum camera excursion and

maximum camera rotation. The evaluation and the comparison between different visual servoing approaches can thus be performed for a given task.

Recently in [96], the robustness of standard image-based visual servoing control and efficient second order approximation method (ESM) was studied theoretically with respect to errors on the 3D parameters introduced in the interaction matrix when any central catadioptric camera is used as a sensor and when points coordinates are used as input of the control scheme. It has been noticed that the stability region is similar for all catadioptric cameras and it is not expected to increase by simply changing the type of central camera used.

In [154], a comprehensive comparison of IBVS and PBVS is presented by comparing system stability and dynamic performance in the Cartesian and image spaces on a common framework using both predefined and taught references. The robustness and sensitivity analyses are investigated with respect to the camera, target, and robot modeling errors.

2.8 Conclusion

In this chapter, we presented a review of the state of the art concerning two main components in visual servoing: features selection and design of the control scheme. The review of features selection can be summarized as illustrated in Fig. 2.3. For a chosen combination of feature selection and control scheme, different behaviors can be obtained by the robot system. Therefore, there is a large collection of research concerning the robustness with respect to selected features, structure of the employed control scheme, existence of errors and uncertainty in robot or camera calibration, and errors and uncertainty in input signals and in object model.

In this thesis, our work focuses on contributing the selected features. 6D orthogonal and independent signals are chosen as visual features for visual servoing system. Two solutions for visual occlusion problem are introduced to change the feature visibility in image features during the control task. For the proposed visual servoing, quantitative validations of the approach and comparisons to classical methods in terms of steady-state errors, transient systems performance in the Cartesian and image spaces, and robustness to uncertainties have been presented. Furthermore, other fundamental characteristics of the different methods, such as sensory space singularity and local minima are also compared.

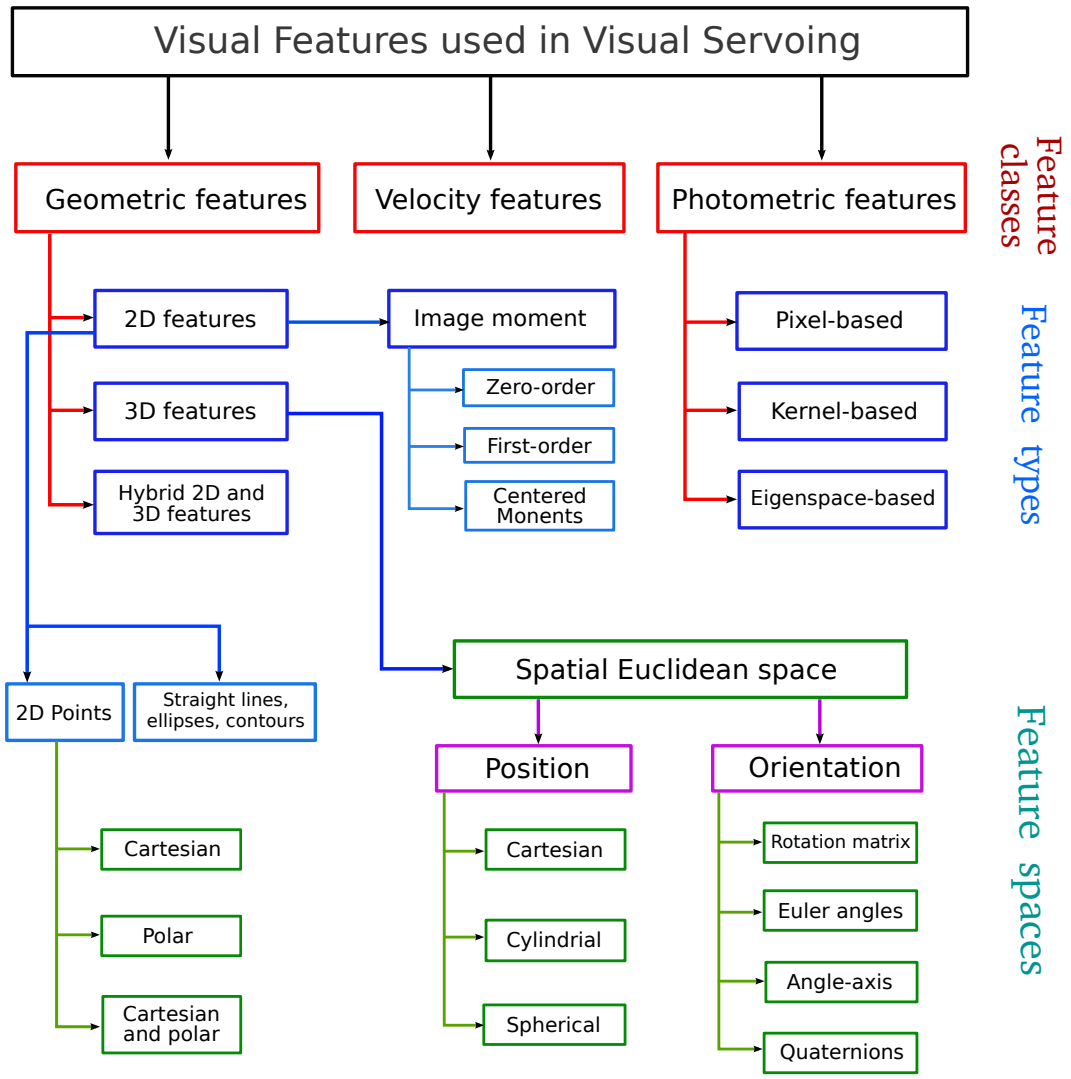


Figure 2.3: Visual features used in visual servoing.

Chapter 3

Robot Modeling

The main objective of this thesis is to implement the visual servoing for an industrial robot manipulator involved in human-robot interaction scenarios. Therefore, getting prior knowledge of the behavior of a manipulator is necessary. In this chapter, the kinematic and dynamic models of the StäubliTX90 industrial robot are described. Furthermore, some classical controllers which will be used in the thesis, are presented.

3.1 Forward Kinematics

The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. Stated more formally, the forward kinematics problem is to determine the position and orientation of the end-effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints.

To perform the kinematic analysis, we rigidly attach a coordinate frame to each link. In particular, we attach $o_i x_i y_i z_i$ to link i. By this convention, joint i connects link $i-1$ to link i. When joint i is actuated, link i moves, and then $o_i x_i y_i z_i$ is changed. The first coordinate frame $o_0 x_0 y_0 z_0$, which is attached to the robot base, is referred to as the inertial (base) frame. The final coordinate system $o_n x_n y_n z_n$ is commonly referred to as the end-effector or tool frame.

A robot manipulator with n joints will have $n+1$ links, since each joint connects two links:

- Link/Frame: **0**, 1, 2, 3, ..., n.
- Joint: 1, 2, 3, ..., n.

Now suppose $H_{i-1}^{i}, i = 1, 2, ..., n$ is the homogeneous transformation matrix that expresses the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. Taking the assumption that each joint has a single degree-of-freedom, H_{i-1}^{i} is a function of only a single joint variable, namely q_i.

The position and orientation of the end-effector with respect to the base frame is denoted by two elements, the vector $t_0^n \in \mathbb{R}^{3\times1}$, which gives the coordinates of the origin of the end-effector frame with respect to the base frame, and the rotation matrix $R_0^n \in SO(3)$. The definition of the homogeneous transformation matrix H_0^n is given by

$$H_0^n = \begin{bmatrix} R_0^n & t_0^n \\ 0_{1\times3} & 1 \end{bmatrix}. \tag{3.1}$$

Then the position and orientation of the end-effector in the base frame are given by transformation matrix

$$H_0^n = H_0^1(q_1)H_1^2(q_2)...H_{n-1}^n(q_n). \tag{3.2}$$

A commonly used convention for selecting frames of reference in robotic applications is the **Denavit-Hartenberg Representation** or ***D-H convention***. The details about this convention are described in the book by Spong [155]. Each *relative* homogeneous transformation H_{i-1}^{i} can be obtained using the Denavit-Hartenberg representation.

3.1.1 3DOF Model

Here we first represent the 3DOF model where we only care about the position of the end-effector. In this case the last three joints of the StäubliTX90[1] are controlled in a static position and the robot end-effector pose are decided by the first 3 DOF. The three-link elbow (RRR) manipulator configuration is shown in Fig. 3.1. The D-H parameters for this robot are shown in the Table 3.1. The end-effector position $X_{\text{ef}} \in \mathbb{R}^{3\times1}$ can be represented as:

$$X_{ef} = f(q), \qquad q = [q_1, q_2, q_3]^T. \tag{3.3}$$

When no ***Kinematic Decoupling***[2] has been used for the description of the kinematic chain of a robot arm, the ***Forward Kinematics*** is computed directly from the translation vector t_0^3, which is obtained from H_0^3 as follows

$$X_{ef} = t_0^3(q) \qquad \text{with} \qquad H_0^3 = \begin{bmatrix} R_0^3 & t_0^3(q) \\ 0 & 1 \end{bmatrix}. \tag{3.4}$$

[1] StaubliTX90: http://www.staubli.com/en/robotics/6-axis-scara-industrial-robot/medium-payload-6-axis-robot/6-axis-industrial-robot-tx90

[2] This topic will be covered in next section. For now, lets say that *Kinematic Decoupling* is a special coordinate frame arrangement where the orientation and position of the end-effector are controlled by a set of different joint variables, therefore they are decoupled from each other.

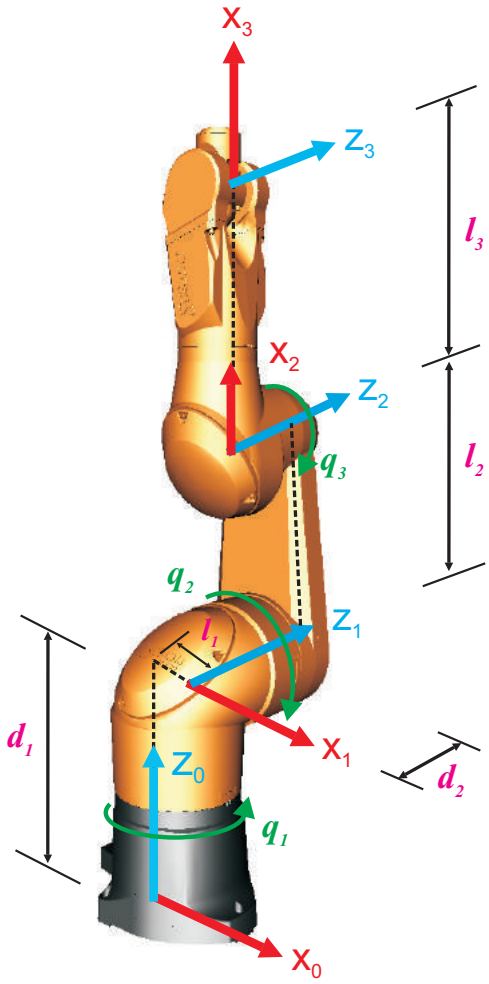


Figure 3.1: The StäubliTX90 configuration (3 DOF) .

Table 3.1: D-H Parameters of StäubliTX90 Industrial Robot (3 DOF).

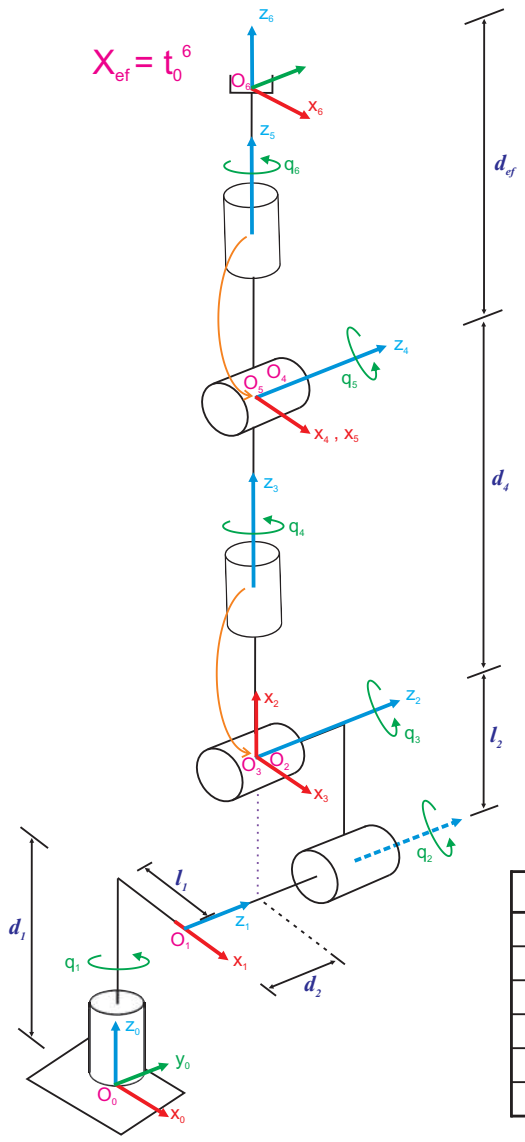
Link i	θ_i (rad)	d_i (m)	a_i (m)	α_i (rad)
1	$q_1 + 0$	$d_1 = 0.478$	$l_1 = 0.050$	$-\pi/2$
2	$q_2 + \pi/2$	$d_2 = 0.050$	$l_2 = 0.425$	0
3	$q_3 + 0$	0	$l_3 = 0.595$	0

The corresponding transformation H_{i-1}^{i} and H_0^3 can be obtained using the D-H parameters. Finally, we can obtain end-effector position using the definition of *Forward Kinematics* as:

$$X_{ef} = t_0^3(q) = \begin{bmatrix} l_1\cos(q_1) - d_2\sin(q_1) + l_2\cos(q_1)\sin(q_2) + l_3\sin(q_2+q_3)\cos(q_1) \\ l_1\sin(q_1) + d_2\cos(q_1) + l_2\sin(q_1)\sin(q_2) + l_3\sin(q_2+q_3)\sin(q_1) \\ d_1 + l_2\cos(q_2) + l_3\cos(q_2+q_3) \end{bmatrix}. \quad (3.5)$$

3.1.2 6DOF Model

Consider now the whole StäubliTX90 manipulator shown in Fig. 3.2. This manipulator is an example of an elbow (RRR) manipulator with a spherical wrist (RRR). This manipulator has an offset in the shoulder joint.



D-H parameters for StaeubliTX90

i	θ	d	a	α
1	q_1	d_1 (0.478)	l_1 (0.050)	-π/2
2	q_2 - π/2	d_2 (0.050)	l_2 (0.425)	0
3	q_3 +π/2	0	0	π/2
4	q_4	d_4 (0.425)	0	-π/2
5	q_5	0	0	π/2
6	q_6	d_{ef} (0.100+)	0	0

Figure 3.2: The StäubliTX90 configuration (6 DOF) and D-H parameters.

Using the D-H table in Fig. 3.2, the whole forward kinematics for StäubliTX90 can be

obtained by:

$$H_0^6 = H_0^1 * H_1^2 * H_2^3 * H_3^4 * H_4^5 * H_5^6 \tag{3.6}$$

$$= \begin{bmatrix} R_0^6 & t_0^6 \\ 0 & 1 \end{bmatrix}. \tag{3.7}$$

The position of the end-effector with respect to the robot base frame is

$$X_{ef} = t_0^6. \tag{3.8}$$

Kinematic Decoupling:

If we choose the coordinate frames in such a way that $o_4 = o_5 = o_6$ (Kinematic Decoupling: Fig. 3.3), then we have the position of the end-effector as

$$X_{ef} = t_0^{6'} + d_{ef} R_0^6 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.9}$$

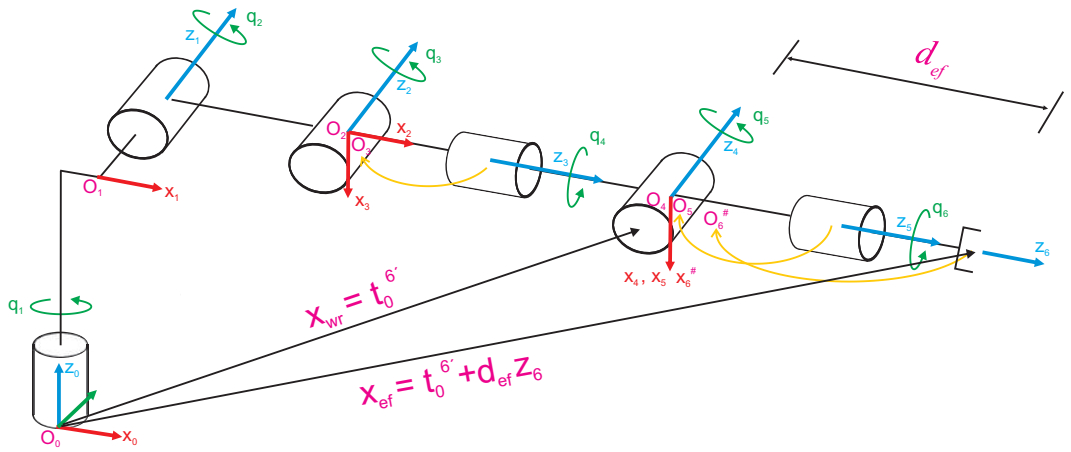


Figure 3.3: The StäubliTX90 configuration (6 DOF) for kinematic decoupling.

3.2 Inverse Kinematics

The forward kinematics problem is to be contrasted with the **inverse kinematics** problem, which is concerned with determining values for the joint variables that achieve a desired position and orientation for the end-effector of the robot.

Although the general problem of inverse kinematics is quite difficult, it turns out that for manipulators having **six** joints, with the last three joints intersecting at a point (such as the

StäubliTX90 above), it is possible to decouple the inverse kinematics problem into two simpler problems, known respectively as **inverse position kinematics** and **inverse orientation kinematics**, namely first finding the position of the intersection of the wrist axes, hereafter called the **wrist center** ($o_c = o_3$), and then finding the orientation of the wrist.

This class of manipulators can be modeled with ***Kinematic Decoupling***, which means that the position of the wrist center is a function of only the first three joint variables, and the motion axes z_3, z_4, z_5 of final three joints intersect at the wrist center o_c. Hence the motion of the final three links about these axes will not change the position of o_c, see Fig. 3.3.

The computation of the inverse kinematics can be summarized by the following algorithm.
Step 1: Find q_1, q_2, q_3 using Inverse Position, with the position of the wrist center X_{wr} given by

$$X_{wr} = t_0^{6'} = X_{ef} - d_{ef} \cdot z_6. \tag{3.10}$$

Step 2: Using the joint variables (q_1, q_2, q_3) determined in Step 1, evaluate R_0^3.
Step 3: The final three joint angles (q_4, q_5, q_6) can then be found as a set of
Euler angles (Z-Y-Z) corresponding to the rotation matrix R_3^6

$$R_3^6 = (R_0^3)^{-1} R_0^6 = (R_0^3)^T R_0^6. \tag{3.11}$$

3.2.1 Inverse Position: A Geometric Approach

For the common kinematics arrangements that we consider, we can use a geometric approach to find the variables q_1, q_2, q_3 corresponding to X_{wr} given by (3.10). Fig. 3.4 shows the elbow manipulator with shoulder offset, with the component of X_{wr} denoted by x_c, y_c, z_c.

From this figure, we see geometrically that

$$q_1 = \alpha - \theta_1; \tag{3.12}$$

$$q_2 = \frac{\pi}{2} - \theta_2 - \theta_3; \tag{3.13}$$

$$q_3 = \text{atan2}\,(CD,\, D). \tag{3.14}$$

in which atan2(y, x) is the arctangent function with two arguments, whose range is $(-\pi, \pi]$. atan2(y, x) is defined for all $(y, x) \neq (0, 0)$ and the sign of the angle is decided by sign of y. (The angle is positive when $y > 0$ and negative when $y < 0$.)

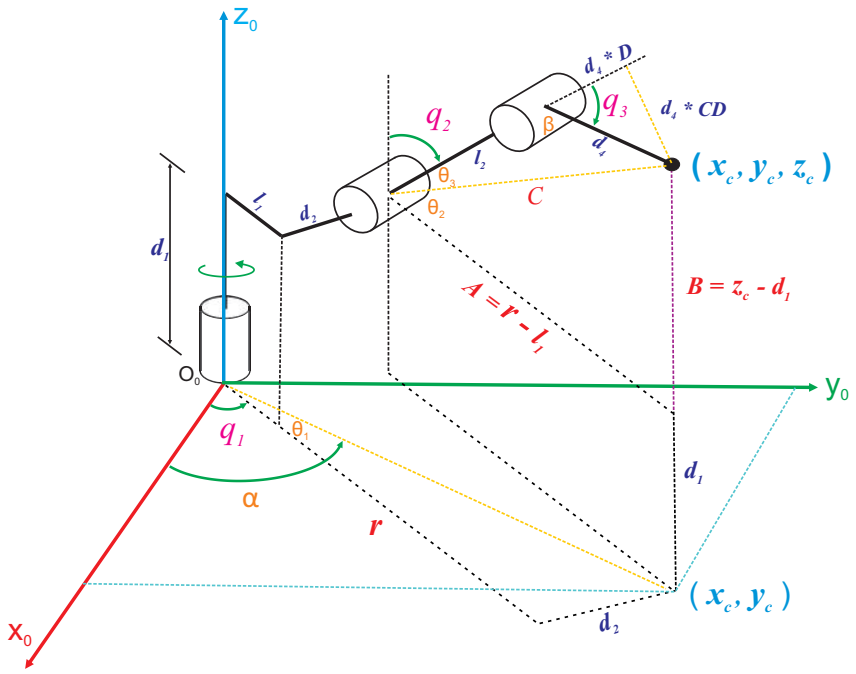


Figure 3.4: Elbow Manipulator with shoulder offset: the first 3 joints of StäubliTX90.

To see the joints, note that

$$r = \sqrt{x_c^2 + y_c^2 - d_2^2} \tag{3.15}$$

$$A = r - l_1 \tag{3.16}$$

$$B = z_c - d_1 \tag{3.17}$$

$$C = \sqrt{A^2 + B^2} \tag{3.18}$$

$$\alpha = \operatorname{atan2}(y_c,\, x_c) \tag{3.19}$$

$$\beta = \pi - q_3 \tag{3.20}$$

$$\theta_1 = \operatorname{atan2}(d_2,\, r) \tag{3.21}$$

$$\theta_2 = \operatorname{atan2}(B,\, A) \tag{3.22}$$

$$\theta_3 = \operatorname{atan2}(d_4 \cdot CD, \quad l_2 + d_4 \cdot D) \tag{3.23}$$

We can apply the **law of cosines** to obtain

$$D = \cos q_3 = -\cos\beta = \frac{C^2 - l_2^2 - d_4^2}{2 \cdot l_2 \cdot d_4} \tag{3.24}$$

$$CD = \sin q_3 = \text{elbowConfig} \cdot \sqrt{1 - D^2} \tag{3.25}$$

where the **elbowConfig** $= \pm 1$ are the two solutions for q_3 corresponding to the **elbow-up** configuration and **elbow-down** configuration, respectively.

3.2.2 Inverse Orientation

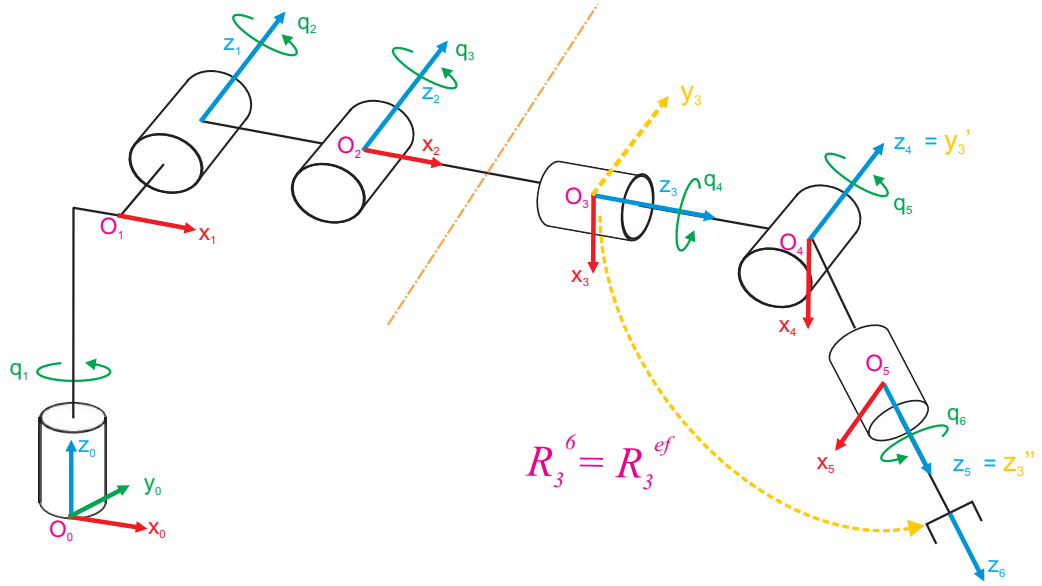


Figure 3.5: Spherical wrist: the final 3 joints of StäubliTX90.

In the previous subsection we used a geometric approach to solve the inverse position problem. This gives the values of the first three joint variables corresponding to a given position of the wrist origin. The inverse orientation problem is one of finding the values of the final three joint variables (q_4, q_5, q_6) corresponding to a given orientation (R_3^6) with respect to the frame $o_3x_3y_3z_3$. For a spherical wrist, this can be interpreted as the problem of finding a set of Euler angles corresponding to a given rotation matrix R. In Fig. 3.5, frame o_3'x_3'y_3'z_3' represents the new coordinate frames after rotating z_3 by ϕ, frame o_3"x_3"y_3"z_3" represents the new coordinate frame after the rotating y_3' by θ, and frame $o_6x_6y_6z_6$ represents the final frame, after the rotating z_3" by ψ. Therefore, we solve for three Euler angles, ϕ, θ, ψ, using equations for Z-Y-Z representation and then use the mapping

$$q_4 = \phi, \qquad q_5 = \theta, \qquad q_6 = \psi. \tag{3.26}$$

Z-Y-Z Euler Angles:

Given the rotation matrix as [1]

$$R_3^6 = R_{z,\phi} R_{y,\theta} R_{z,\psi} \tag{3.27}$$

$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{3.28}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.29}$$

we can get the Euler angles with

If: not both r_{13} and r_{23} are zero, then $r_{33} \neq \pm 1$, and we have

$$\phi = \text{atan2}\left(\text{wristConfig} \cdot r_{23}, \quad \text{wristConfig} \cdot r_{13}\right) \tag{3.30}$$

$$\theta = \text{atan2}\left(\text{wristConfig} \cdot \sqrt{1 - r_{33}^2}, \quad r_{33}\right) \tag{3.31}$$

$$\psi = \text{atan2}\left(\text{wristConfig} \cdot r_{32}, \quad -\text{wristConfig} \cdot r_{31}\right) \tag{3.32}$$

where the **wristConfig** $= \pm 1$ are the two solutions for q_5 corresponding to the wrist-up position and wrist-down position, respectively.

Else: $r_{13} = r_{23} = 0$, then the fact that R is orthogonal, implies that $r_{33} = \pm 1$:

if: $r_{33} = 1$, then $\theta = 0$. In this case, the sum $(\phi + \psi)$ can be determined as

$$\phi + \psi = \text{atan2}\left(r_{21}, \, r_{11}\right) \tag{3.33}$$

else: $r_{33} = -1$, then $\theta = \pi$. In this case it becomes

$$\phi - \psi = \text{atan2}\left(-r_{12}, \, -r_{11}\right) \tag{3.34}$$

Since only the sum and the subtraction can be determined in this case there are infinitely many solutions. We may take $\phi = \mathit{oldValue}$ by convention, and define ψ.

3.2.3 Summary of Inverse Kinematics of StäubliTX90

To summarize the preceding developments, we write down one solution to the inverse kinematics of the 6DOF elbow manipulator shown in Fig. 3.3 which has shoulder offset and a spherical wrist.

Given

$$X_{ef} = \begin{bmatrix} x_{ef} \\ y_{ef} \\ z_{ef} \end{bmatrix}, \qquad R_0^6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.35}$$

[1] Here we use the shorthand notation $c_\theta = \cos\theta, s_\theta = \sin\theta$ for trigonometric functions.

then with

$$x_c = x_{ef} - d_{ef}\, r_{13} \tag{3.36}$$

$$y_c = y_{ef} - d_{ef}\, r_{23} \tag{3.37}$$

$$z_c = z_{ef} - d_{ef}\, r_{33} \tag{3.38}$$

a set of *D-H* joint variables is given by

$$q_1 = \text{atan2}\,(y_c, x_c) - atan2(d_2, r) \tag{3.39}$$

$$q_2 = \frac{\pi}{2} - \text{atan2}\,(B, A) - \text{atan2}\,(d_4 \cdot CD,\, l_2 + d_4 \cdot D) \tag{3.40}$$

$$q_3 = \text{atan2}\,(\pm\sqrt{1 - D^2},\, D) \tag{3.41}$$

$$q_4 = \text{atan2}\,(\pm r_{23},\, \pm r_{13}) \tag{3.42}$$

$$q_5 = \text{atan2}\,(\pm\sqrt{1 - {r_{33}}^2},\, r_{33}) \tag{3.43}$$

$$q_6 = \text{atan2}\,(\pm r_{32},\, \mp r_{31}) \tag{3.44}$$

The other possible solutions are already explained above.

3.3 Velocity Kinematics

In order to control the robot manipulator, it is required to obtain not only forward and inverse kinematics but also differential kinematics, which relates the linear and angular velocities of the end-effector **(or any other point on the manipulator)** to the joint velocities of the robot. In particular, we will derive the linear and angular velocities ($v_0^n \in \mathbb{R}^{3\times1}$ and $\omega_0^n \in \mathbb{R}^{3\times1}$, respectively) of the end-effector frame with respect to the robot base frame as a function of the joint velocities $\dot{q}_i$.

Mathematically, the forward kinematic equations define a function between the space of Cartesian position and orientation and the space of joint positions. The velocity relationships are then determined by the **Jacobian** of this function. The Jacobian is a matrix-valued function and can be thought of as the vector version of the ordinary derivative of a scalar function. This Jacobian is one of the most important quantities in the analysis and control of robot motion.

For a n-link manipulator, the differential kinematics is defined as (3.45) and it is obtained taking the time derivative of the end-effector pose

$$\dot{\mathbf{X}}_{ef} = \begin{bmatrix} v_0^n \\ \omega_0^n \end{bmatrix} = J_0^n\, q_i \tag{3.45}$$

where J_0^n is given by

$$J_0^n = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}. \tag{3.46}$$

The matrix J_0^n is called the **Manipulator Jacobian** or **Jacobian** for short. Note that J_0^n is a $6 \times n$ matrix where n is the number of links.

3.3.1 Differential Kinematic for StäubliTX90

Given that the StäubliTX90 is a 6DOF robot manipulator, and all the joints are revolute with o_3, o_4, o_5 representing the origins of the coordinate frames associated to the spherical wrist, see Fig. 3.2. Then, following the above definitions, the Geometric Jacobian for this robot is

$$J_0^6 = \begin{bmatrix} z_0 \times t_0^6 & z_1 \times (t_0^6 - t_0^1) & z_2 \times (t_0^6 - t_0^2) & z_3 \times (t_0^6 - t_0^3) & z_4 \times (t_0^6 - t_0^4) & z_5 \times (t_0^6 - t_0^5) \\ z_0 & z_1 & z_2 & z_3 & z_4 & z_5 \end{bmatrix}. \tag{3.47}$$

Kinematic Decoupling: If we choose the coordinate frames in such a way that $o_4 = o_5 = o_6$ (Kinematic Decoupling: Fig. 3.3), then the Geometric Jacobian is given by

$$J_0^{6'} = \begin{bmatrix} z_0 \times t_0^{6'} & z_1 \times (t_0^{6'} - t_0^1) & z_2 \times (t_0^{6'} - t_0^2) & 0 & 0 & 0 \\ z_0 & z_1 & z_2 & z_3 & z_4 & z_5 \end{bmatrix}. \tag{3.48}$$

Remarks:

1. **In this case, the position of the end-effector is given by $X_{ef} = t_0^{6'} + d_6 z_6$ and its linear velocity is $J_v' \dot{q} + d_6 \dot{z}_6$.**
2. **Note that this form of the Jacobian does not necessarily give the correct relation between the velocity of the end-effector and the joint velocities. It is intended only to simplify the determination of singularities.**

3.4 Singularities

Since the Jacobian is a function of the configuration q, those configurations for which the rank of J decreases are of special significance. Such configurations are called **singularities**.

3.4.1 Decoupling of Singularities

In our robot, $n = 6$, the manipulator consists of a $3 - DOF$ arm with a $3 - DOF$ spherical wrist. In this case the Jacobian is a 6×6 matrix and a configuration q is singular if and only if

$$\det J(q) = 0. \tag{3.49}$$

Now, we can partition the Jacobian $J(q)$ into 3×3 blocks as

$$J = \begin{bmatrix} J_P \mid J_O \end{bmatrix} \quad = \quad \left[\begin{array}{c|c} J_{11} & J_{12} \\ J_{21} & J_{22} \end{array}\right]. \tag{3.50}$$

Case 1: Without Kinematic Decoupling

In this case we need to directly compute (Fig. 3.2)

$$\det(J(q)) = \det(J_{11})\,\det(J_{22}) - \det(J_{12})\,\det(J_{21}) \;=\; 0 \tag{3.51}$$

which is very difficult for computing the solutions.

Case 2: Kinematic Decoupling

Since the wrist axes intersect at a common point o_c, if we choose the coordinate frames in such a way that $o_4 = o_5 = o_6$ (Kinematic Decoupling: Fig. 3.3), then $J_{12} = 0_{3\times 1}$. In this case the Jacobian matrix (3.50) has the block triangular form

$$J \quad = \quad \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \tag{3.52}$$

with determinant

$$\det(J) = \det J_{11} \cdot \det J_{22}. \tag{3.53}$$

Therefore the set of singular configurations of the manipulator is the union of the set of **arm configurations** satisfying $\det(J_{11}) = 0$ and the set of **wrist configurations** satisfying $\det(J_{22}) = 0$.

3.4.2 Wrist Singularities

In the case 2: Kinematic Decoupling, we get

$$J_{22} = \begin{bmatrix} z_3 & z_4 & z_5 \end{bmatrix}. \tag{3.54}$$

From (3.54) we can now see that a spherical wrist is in a singular configuration whenever the vectors z_3 , z_4 and z_5 are linearly dependent. Referring to Fig. 3.6 [155] we see that this happens when the joint axes z_3 and z_5 are collinear.

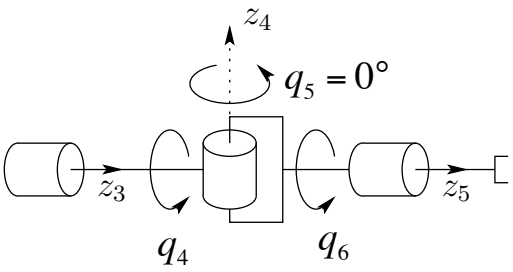


Figure 3.6: Spherical wrist singularity.

3.4.3 Arm Singularities

In order to investigate arm singularities we only need to compute J_{11} according to (3.52),

$$\det J_{11} = 0. \tag{3.55}$$

$q_3 = 0$ or π is one of the solutions for equation (3.55), which are shown in Fig. 3.7 [155].

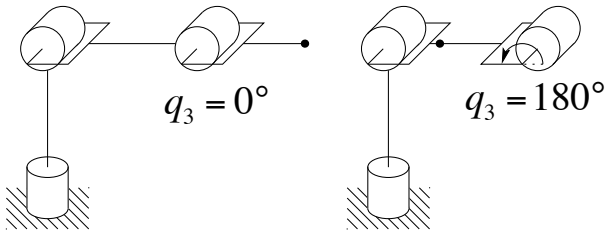


Figure 3.7: Elbow singularities of the elbow manipulator.

3.5 Dynamics

The dynamics of a serial n-link rigid, non-redundant, fully actuated robot manipulator can be written as follows

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} = \tau \tag{3.56}$$

where $q \in \mathbb{R}^{n\times 1}$ is the vector of joint positions, $\tau \in \mathbb{R}^{n\times 1}$ stands for the applied joint torques, $M(q) \in \mathbb{R}^{n\times n}$ is the symmetric positive definite inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the vector of centripetal and Coriolis effects, $G(q) \in \mathbb{R}^{n\times 1}$ is the vector of gravitational torques, and finally $B \in \mathbb{R}^{n\times n}$ is a diagonal matrix for the viscous frictions.

Using the **Iterative Euler-Lagrange method** to compute the dynamic equations of motion, the matrices are computed as:

1. Compute the Inertia matrix $M(q) \in \mathbb{R}^{n\times n}$:

$$M(q) \quad = \quad \sum_{i=1}^{n} m_i J_{\upsilon_i}^{cm_i}(q)^T J_{\upsilon_i}^{cm_i}(q) + J_{\omega_i}^{cm_i}(q)^T R_0^{cm_i}(q) I_i R_0^{cm_i}(q)^T J_{\omega_i}^{cm_i}(q).$$

2. Compute the matrix of Coriolis and Centripetal effects $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$:

$$c_{kj} \quad = \quad \frac{1}{2}\sum_{i=1}^{n}\left\{\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k}\right\}\dot{q}_i.$$

3. Compute the vector of gravitational torques $G(q) \in \mathbb{R}^{n\times 1}$:

$$G_k = \frac{\partial P}{\partial q_k}, \qquad P = \sum_{i=1}^{n} m_i g^T t_0^{cmi}.$$

4. Substituting M, C, G into the motion equation gives the dynamical equations (3.56).

3.5.1 Dynamics Properties

The dynamic model of the robot, represented by equation (3.56), presents the following properties [155].

1. Positive Definite Matrix

$M(q)$ is a **bounded symmetric positive** definite matrix that is, in general, configuration dependent: $M_+(q) = M_+(q)^T$.

2. Skew Symmetric Matrix

The matrix $N(q, \dot{q}) = \dot{M}(q) - 2C(q, \dot{q})$ is **skew symmetric**, that is, the components n_{ij} of N satisfy $n_{ij} = -n_{ji}$. In other words,

- $N = -N^T$;
- $X^T N X = 0$, where $X \in \mathbb{R}^{3\times 1}$ is any vector.

3. Linearity in the Parameters

The robot model described in (3.56) can be written in terms of a known state robot **Regressor** $Y = Y(q, \dot{q}, \ddot{q}) \in \mathbb{R}^{n\times m}$ and an unknown robot **Parameter** vector $\Theta \in \mathbb{R}^{m\times 1}$ by using nominal references $\dot{q}_r$ and $\ddot{q}_r$ as follows:

$$M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + G(q) + B\dot{q}_r = Y_r\Theta. \tag{3.57}$$

Subtracting the linear parameterization equation (3.57) to (3.56), produces the open-loop error dynamics

$$M(q)\dot{S}_q + C(q, \dot{q})S_q = \tau - Y_r\Theta \tag{3.58}$$

with the **joint error surface** S_q defined as $S_q = \dot{q} - \dot{q}_r$, where $\dot{q}_r$ represents the nominal reference of joint velocities. This nominal reference can be used to design a control.

These properties are important in terms of designing the control, as shown in the next section. This design is based on *Lyapunov Theory*.

3.6 Control

In this section we introduce some classical controllers briefly, that will be used in our system later. Here, we assume $B = 0$ for simplicity in equation (3.56) and use this equation subsequently.

3.6.1 PD Control with Gravity Compensation

Let a **constant** equilibrium posture be assigned for the system as the vector of desired joint variables q_d. It is desired to find the structure of the controller which ensures global asymptotic stability of the above posture. The determination of the control input which stabilizes the system around the equilibrium posture is based on the Lyapunov direct method.

An independent **joint** PD-control scheme can be written in vector form as

$$\tau = K_p \Delta q - K_d \dot{q} + G(q) \tag{3.59}$$

where $\Delta q = q_d - q$ is the error between the desired joint displacements q_d and the actual joint displacements q, and K_p, K_d are diagonal matrices of (positive) proportional and derivative gains, respectively. We first show that the PD control law (3.59) achieves asymptotic tracking of the desired joint positions.

To show that the above control law achieves zero steady state error, consider the Lyapunov function candidate

$$V = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}\Delta q^T K_p \Delta q. \tag{3.60}$$

Differentiating (3.60) with respect to time and substituting the PD control law (3.59) for τ, and recalling that q_d is constant, yields

$$\dot{V} = \dot{q}^T M(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{M}(q)\dot{q} + \Delta \dot{q}^T K_p \Delta q \tag{3.61}$$

$$= -\dot{q}^T K_d \dot{q} \leq 0, \tag{3.62}$$

where in the last equality we have used the fact that $\dot{M} - 2C$ is skew symmetric.

The above analysis shows that V is decreasing as long as $\dot{q} \neq 0$ for all system trajectories. It can be shown that the system reaches an *equilibrium posture*. To find such posture, notice that $\dot{V} \equiv 0$, only if $\dot{q} \equiv 0$. At the equilibrium ($\dot{q} \equiv 0$, $\ddot{q} \equiv 0$), it is

$$\Delta q = q_d - q = 0. \tag{3.63}$$

The above derivation rigorously shows that any manipulator equilibrium posture is globally asymptotically stable under a controller with a PD linear action and a nonlinear gravity compensating action.

Note: This PD control is only for regulation since $\dot{q}_d = 0$.

3.6.2 Inverse Dynamics Control

Consider now the problem of tracking a joint space trajectory. The nonlinear control law for *inverse dynamics control* is represented as

$$\tau = M(q)a_q + C(q, \dot{q})\dot{q} + G(q). \tag{3.64}$$

Then, since the inertia matrix M is invertible, the combined system (3.56) and (3.64) reduces to

$$\ddot{q} = a_q. \tag{3.65}$$

Given a desired trajectory and choosing the reference input as

$$r(t) = \ddot{q}_d + K_d\,\dot{q}_d + K_p\,q_d \tag{3.66}$$

the term a_q represents a new input to the system which is yet to be chosen. Since a_q can now be designed to control a linear second order system r, the obvious choice is to set

$$a_q = \ddot{q}_d + K_d\,(\dot{q}_d - \dot{q}) + K_p\,(q_d - q) \tag{3.67}$$

where K_p and K_d are diagonal matrices with diagonal elements consisting of position and velocity gains, respectively. Then the tracking error $e(t) = q - q_d$ satisfies

$$\ddot{e}(t) + K_d\,\dot{e}(t) + K_p\,e(t) = 0. \tag{3.68}$$

For this control law, two feedback loops are represented: an inner loop based on the manipulator dynamic model, and an outer loop operating on the tracking error.

Task Space Inverse Dynamics:

The tracking in task space can be achieved by modifying our choice of outer loop control $\ddot{q}$ in (3.65) while leaving the inner loop control unchanged. Let $X \in \mathbb{R}^6$ represent the end-effector pose using any minimal representation of $SO(3)$. Since X is a function of the joint variables q we have

$$\dot{X} = J(q)\dot{q} \tag{3.69}$$

$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{3.70}$$

where J is the analytical Jacobian of the robot. Given the double integrator system, (3.65), in joint space we see that if a_q is chosen as

$$a_q = J^{-1}\{a_X - \dot{J}\dot{q}\} \tag{3.71}$$

the result is a double integrator system in task space coordinates

$$\ddot{X} = a_X. \tag{3.72}$$

Given a task space trajectory $X_d(t)$, satisfying the same smoothness and boundedness assumptions as the joint space trajectory $q_d(t)$, we may choose a_X as

$$a_X = \ddot{X}_d + K_d(\dot{X}_d - \dot{X}) + K_p(X_d - X) \tag{3.73}$$

so that the Cartesian space tracking error, $e_X = X - X_d$, satisfies

$$\ddot{e}_X + K_d\dot{e}_X + K_p e_X = 0. \tag{3.74}$$

Therefore, a modification of the outer loop control achieves a linear and decoupled system directly in the task space coordinates, without the need to compute a joint space trajectory and without the need to modify the nonlinear inner loop control.

3.6.3 Regressor-based Control

As shown in the (3.57), using the nominal reference and joint error space S_q, the system then becomes as

$$M(q)\dot{S}_q + C(q,\dot{q})S_q = \tau - Y_r\Theta \tag{3.75}$$

where $S_q = \dot{q} - \dot{q}_r$.

The Lyapunov function

$$V = \frac{1}{2}{S_q}^T M(q) S_q \tag{3.76}$$

is used to show uniform ultimate boundedness of the tracking error. Calculating $\dot{V}$ yields

$$\dot{V} = {S_q}^T M\dot{S}_q + \frac{1}{2}{S_q}^T\dot{M}S_q \tag{3.77}$$

$$= {S_q}^T(\tau - Y_r\Theta). \tag{3.78}$$

By inspecting (3.78), we see that if we choose the control τ according to the equation as

$$\tau = -K_d S_q + Y_r\Theta \tag{3.79}$$

then

$$\dot{V} = -{S_q}^T K_d S_q \leq 0. \tag{3.80}$$

At the equilibrium point, $\dot{V} \equiv 0$. Then (3.80) implies that $S_q \equiv 0$ and hence velocity $\dot{q} = \dot{q}_r$. This means that we can modify the velocity reference $\dot{q}_r$ to control the robot behavior.

Case 1: Break

If we want a break, we can set the $\dot{q}_r = 0$.

Case 2: PD Control

If we want a PD controller for the robot system, we can define

$$(\dot{q} - \dot{q}_d) + K_p(q - q_d) = 0, \tag{3.81}$$

which yields velocity reference $\dot{q}_r$ as

$$\dot{q}_r = \dot{q}_d - K_p(q - q_d) \tag{3.82}$$

$$\ddot{q}_r = \ddot{q}_d - K_p(\dot{q} - \dot{q}_d). \tag{3.83}$$

3.6.4 Passivity Based Adaptive Control

The feedback linearization approach relies on exact cancellation of nonlinearities in the robot equations of motion. Its practical implementation requires consideration of various sources of uncertainties such as modeling errors, unknown loads, and computation errors. In this case, we can rewrite the (3.75) as

$$M(q)\dot{\hat{S}}_q + C(q, \dot{q})\hat{S}_q = \tau - \hat{Y}_r\Theta, \tag{3.84}$$

where joint error surface is

$$\hat{S}_q = \dot{q} - \hat{\dot{q}}_r = S_q - \Delta\dot{q}_r, \tag{3.85}$$

with $\Delta\dot{q}_r = \hat{\dot{q}}_r - \dot{q}_r$ as the estimation errors.

In the adaptive approach, the vector $\hat{\Theta}$ in (3.84) is taken to be a time-varying estimate of the true parameter vector Θ. Combing the control law (3.79) with (3.84) yields

$$\tau = -K_d\hat{S}_q + \hat{Y}_r\hat{\Theta}. \tag{3.86}$$

The parameter estimate $\hat{\Theta}$ may be computed using standard methods such as gradient or least squares. For example, using the gradient update law

$$\dot{\hat{\Theta}} = -\Gamma{\hat{Y}_r}^T\hat{S}_q \tag{3.87}$$

together with the Lyapunov function

$$V = \frac{1}{2}\left[\hat{S_q}^T M(q)\,\hat{S}_q + \Delta\Theta^T\Gamma^{-1}\Delta\Theta\right] \tag{3.88}$$

results in global convergence of the tracking errors to zero and boundedness of the parameter estimates since

$$\dot{V} = \hat{S_q}^T M\dot{\hat{S}}_q + \frac{1}{2}\hat{S_q}^T\dot{M}\hat{S}_q + \Delta\Theta^T\Gamma^{-1}\dot{\hat{\Theta}} \tag{3.89}$$

$$= -\hat{S_q}^T K_d\hat{S}_q + \hat{S_q}^T\hat{Y}_r\Delta\Theta - \Delta\Theta^T\hat{Y_r}^T\hat{S}_q \tag{3.90}$$

$$= -K_d\|\hat{S}_q\| \leq 0 \tag{3.91}$$

where $\Delta\Theta = \hat{\Theta} - \Theta$, $K_d = {K_d}^T \in \mathbb{R}_+^{n\times n}$ and $\Gamma \in \mathbb{R}_+^{m\times m}$ are constant matrices.

3.6.5 Force Control

Active interaction control strategies can be grouped into two categories: those performing direct force control and those performing indirect force control. The main difference between the two categories is that the former offer the possibility of controlling the contact force and moment to a desired value, thanks to the closure of a force feedback loop; the latter instead achieves force control via motion control, without explicit closure of a force feedback loop.

3.6.5.1 Direct Force Control

As compared to indirect force control, direct force control requires an explicit model of the interaction task. In fact, the user has to specify the desired motion and the desired contact force and moment in a consistent way with respect to the constraints imposed by the environment. A widely adopted strategy belonging to this category is *hybrid force/motion control*, which aims at controlling the motion along the unconstrained task directions and force (and moment) along the constrained task directions.

When the manipulator is in contact with the environment, the dynamic equations of rigid robot must be modified to include the reaction torque $J^T F_c$ corresponding to the end-effector force F_c. Thus the equations of motion of the manipulator in joint space (3.56) are modified as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau - J^T F_c. \tag{3.92}$$

3.6.5.2 Impedance Control

In the second category are *impedance control* (or *admittance control*), where the deviation of the end-effector motion from the desired motion due to the interaction with the environment

is related to the contact force through a mechanical impedance/admittance with adjustable parameters. A robot manipulator under impedance (or admittance) control is described by an equivalent mass-spring-damper system with adjustable parameters.

The idea behind Impedance Control is to regulate the mechanical impedance, through force feedback F_e. Let X_d be a reference trajectory defined in task space coordinates and let M, K_d, K_p be 6×6 matrices specifying desired inertia, damping and stiffness, respectively. Let $e = X_d - X$ be the tracking error in task space. Then the closed loop system is

$$F_e = M\ddot{e} + K_d\dot{e} + K_p e, \tag{3.93}$$

which results in desired impedance properties of the end-effector. Note that for $F_e = 0$ tracking of the reference trajectory, X_d, is achieved.

3.7 Conclusion

The details of robot model for 6DOF StäubliTX90 industrial manipulator are described in this chapter, including its kinematics and dynamics. Some classical controllers which will be later used in this thesis, are also reviewed. From the next chapter, we will start to introduce the vision based robot control, Visual Servoing (VS). We will first present two camera models for the position-based VS and show some solutions for the visual occlusion problem in PBVS in next chapter.

Chapter 4

Camera Models for Occlusion Avoidance

In this chapter, two different solutions are provided for the visual occlusion problem in PBVS (e.g. out of camera field of view, visual occlusion by environments). Two different camera models are presented to obtain the 3D Cartesian position of the objects, using the pixel features extracted from the image plane. The properties of both camera models are validated on a real industrial robot. The proposed camera models are integrated into a 3D PBVS in a human-robot interaction (HRI) scenario.

4.1 Object Pose Estimation

In PBVS, features are extracted from the image and used to estimate the 3D Cartesian pose of the target with respect to camera. Using these values, an error between the current and the desired pose of the robot is defined in the task space. The translation and the rotation in Cartesian space are explicitly reconstructed using pose estimation algorithms. The first step is to extract features from the image. The color-based object tracking algorithm, described in Appendix A, detects the objects and provides the pixel positions of the centroids of the detected objects in the image plane. In the following section, we present the relationship between these pixel positions in image space and the 3D positions in Cartesian space.

Pinhole Camera Model

The pinhole camera model in Fig. 4.1 can be represented as [156]

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c/z_c \\ y_c/z_c \\ 1 \end{bmatrix} \tag{4.1}$$

where $X_c = [x_c, y_c, z_c]^T$ is the 3D Cartesian position of point P with respect to the camera frame O_c, while $[u, v]^T$ is the projected position in pixels, $C = [c_x, c_y]^T$ is the position of the principal point in image plane (image center). f_x, f_y is the focal length of the lens used.

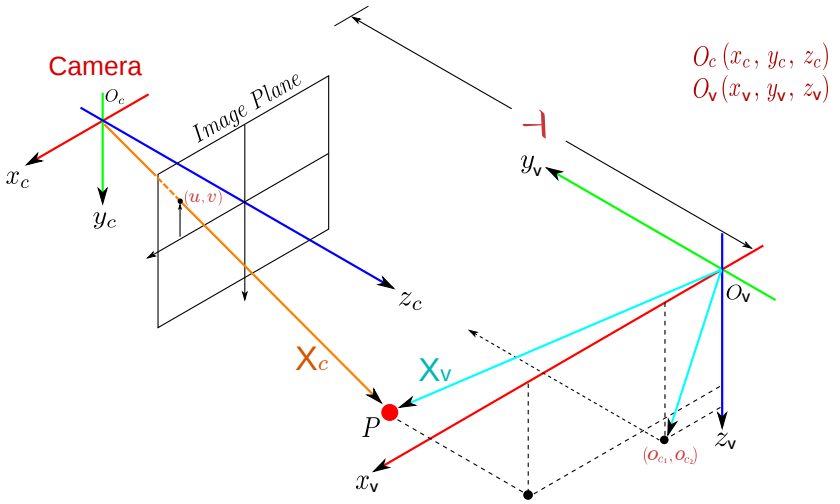


Figure 4.1: A general projective camera model.

Fig. 4.1 shows the point P represent in the world frame O_v with $X_\mathrm{v} = [x_\mathrm{v}, y_\mathrm{v}, z_\mathrm{v}]^T$. Suppose the axes x_c, x_v and y_c, z_v are parallel, respectively, with a distance λ. Then the relationship between frame O_v and O_c is

$$(R_\mathrm{v}^c)^{-1} = R_c^\mathrm{v} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \qquad t_\mathrm{v}^c = \begin{bmatrix} o_{c_1} \\ \lambda \\ o_{c_2} \end{bmatrix} \tag{4.2}$$

where $[o_{c_1}, o_{c_2}]^T$ is the position of the optical center of the camera with respect to the coordinate frame O_v.

Therefore, using these transformations we can get X_c as

$$X_c = R_c^\mathrm{v}(X_\mathrm{v} - t_\mathrm{v}^c) = \begin{bmatrix} x_\mathrm{v} - o_{c_1} \\ z_\mathrm{v} - o_{c_2} \\ \lambda - y_\mathrm{v} \end{bmatrix}. \tag{4.3}$$

Now, if the camera coordinate frame O_c is rotated an angle θ around z_c axis, keeping the planes $(x_c - y_c)$ and $(x_\mathrm{v} - z_\mathrm{v})$ parallel. The equation (4.3) can be rewritten as

$$X_c = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_\mathrm{v} - o_{c_1} \\ z_\mathrm{v} - o_{c_2} \\ \lambda - y_\mathrm{v} \end{bmatrix} = \begin{bmatrix} R(\theta) \cdot \begin{bmatrix} x_\mathrm{v} - o_{c_1} \\ z_\mathrm{v} - o_{c_2} \end{bmatrix} \\ \lambda - y_\mathrm{v} \end{bmatrix} \tag{4.4}$$

where the rotation matrix $R(\theta)$ is defined as

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{4.5}$$

Substituting (4.4) into (4.1) we obtain

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{-y_{\mathrm{v}} + \lambda} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{\mathrm{v}} - o_{c_1} \\ z_{\mathrm{v}} - o_{c_2} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{4.6}$$

The equation (4.6) represents the relationship between pixel position $[u, v]^T$ in image plane and 3D Cartesian position $[x_{\mathrm{v}}, y_{\mathrm{v}}, z_{\mathrm{v}}]^T$ of the point P. All of the parameters of the cameras (intrinsic and extrinsic parameters) are obtained by standard camera calibration methods [156].

The following sections describe two different camera models that can be used to obtain the 3D Cartesian positions.

4.2 Orthogonal Cameras System

This section describes the use of orthogonal configuration of cameras to obtain the 3D Cartesian position of objects. First, we introduce the *Composite Camera Model*, which consists of two cameras in orthogonal configuration. Furthermore, the camera model is extended to incorporate four orthogonal cameras, which provides a solution for visual occlusion for PBVS.

4.2.1 Composite Camera Model

Generally, one camera is not enough to get the 3D position of a target relative to a specific coordinate frame. Therefore, in order to get a model that allows to obtain the 3D position of a target through visual information, a second camera is placed in an orthogonal position with respective to the first camera. In Fig. 4.2, coordinate frames O_{c_1} and O_{c_2} are cameras frames and O_{v} is reference frame. Note that the planes $x_{c_1} - y_{c_1}$ and $x_{\mathrm{v}} - z_{\mathrm{v}}$ are parallel, and $x_{c_2} - y_{c_2}$ and $y_{\mathrm{v}} - z_{\mathrm{v}}$ are parallel.

Following the same procedure described in the previous section, the pinhole model for cameras is defined by the equation (4.6). Therefore, the model for camera 1 is given by

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \frac{1}{-y_{\mathrm{v}} + \lambda_1} \alpha_1 R(\theta_1) \begin{bmatrix} x_{\mathrm{v}} - o_{11} \\ z_{\mathrm{v}} - o_{12} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{4.7}$$

where θ_1 is the rotation angle of camera 1 along its optical axis, $O_1 = [o_{11}, o_{12}]^T$ is the position of the optical center with respect to the coordinate frame O_{v}, λ_1 is the distance from the camera frame O_{c_1} to the reference frame O_{v} along its optical axis. α_1 is defined as

$$\alpha_1 = \begin{bmatrix} f_{x_1} & 0 \\ 0 & f_{y_1} \end{bmatrix}. \tag{4.8}$$

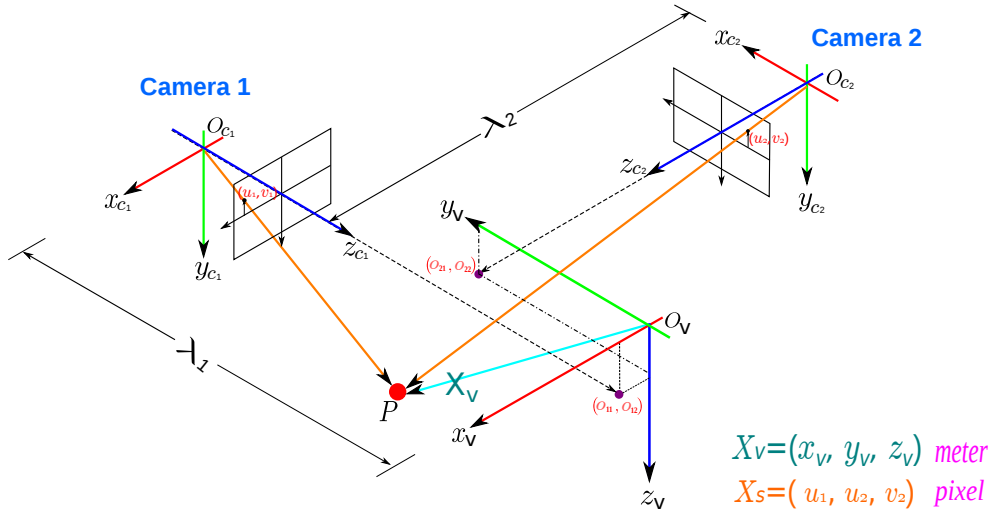


Figure 4.2: Two projective cameras which are orthogonal.

Similarly, camera 2 is defined as:

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \frac{1}{x_{\mathrm{v}} + \lambda_2} \alpha_2 R(\theta_2) \begin{bmatrix} y_{\mathrm{v}} - o_{21} \\ z_{\mathrm{v}} - o_{22} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{4.9}$$

Using properties of the rotation matrix $R(\theta_1)$ and the fact that $\alpha_1 = \begin{bmatrix} f_{x_1} & 0 \\ 0 & f_{y_1} \end{bmatrix}$ is a diagonal matrix, from (4.7), u_1 can be written in terms of v_1 as

$$u_1 = (\frac{f_{x_1}}{\cos\theta_1}) \cdot \frac{x_{\mathrm{v}} - o_{11}}{-y_{\mathrm{v}} + \lambda_1} - (\frac{f_{x_1} \sin\theta_1}{f_{y_1} \cos\theta_1}) \cdot v_1 + (\frac{f_{x_1} \sin\theta_1}{f_{y_1} \cos\theta_1}) \cdot c_y + c_x \tag{4.10}$$

$$= \gamma_1 \cdot \frac{x_{\mathrm{v}} - o_{11}}{-y_{\mathrm{v}} + \lambda_1} - \gamma_2 \cdot v_1 + \gamma_3 \tag{4.11}$$

where the constant parameters $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}$ are explicitly defined as

$$\gamma_1 = \frac{f_{x_1}}{\cos\theta_1} \tag{4.12}$$

$$\gamma_2 = \frac{f_{x_1}}{f_{y_1}} \tan\theta_1 \tag{4.13}$$

$$\gamma_3 = c_x + c_y \gamma_2. \tag{4.14}$$

Since the axes of u_2, v_2 and u_1 are orthogonal to each other. Therefore, based on (4.9) and (4.11), we can define a 3D pixel vector $X_s = [x_s, y_s, z_s]^T$ in frame O_s as

$$X_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ v_2 \end{bmatrix} = \overbrace{\begin{bmatrix} \gamma_1 & 0_{1\times 2} \\ 0_{2\times 1} & \alpha_2 R(\theta_2) \end{bmatrix}}^{R_\alpha} \begin{bmatrix} \frac{x_{\mathrm{v}} - o_{11}}{-y_{\mathrm{v}} + \lambda_1} \\ \frac{y_{\mathrm{v}} - o_{21}}{x_{\mathrm{v}} + \lambda_2} \\ \frac{z_{\mathrm{v}} - o_{22}}{x_{\mathrm{v}} + \lambda_2} \end{bmatrix} + \rho \tag{4.15}$$

where $\rho = [\gamma_3 - \gamma_2 v_1, c_x, c_y]^T$. The pixel position X_s constructs the *virtual 3D Pixel Space*.

Remarks:

1. **X_s is a 3D image features which are measured in pixels. It is not a Cartesian position.**
2. **We always define the reference frame O_v as a specific orientation, such as $\theta_1 = \theta_2 = 0$. In this case, then $\gamma_1 = f_{x_1}, \gamma_2 = 0$, which implies $\rho = [c_x, c_x, c_y]^T$ is constant and $R_\alpha \in \mathbb{R}^{3\times3}$ is a diagonal matrix.**

Therefore, equation (4.15) can be simplified as

$$X_s = \begin{bmatrix} f_{x_1} & 0 & 0 \\ 0 & f_{x_2} & 0 \\ 0 & 0 & f_{y_2} \end{bmatrix} \begin{bmatrix} \frac{x_\mathrm{v}-o_{11}}{-y_\mathrm{v}+\lambda_1} \\ \frac{y_\mathrm{v}-o_{21}}{x_\mathrm{v}+\lambda_2} \\ \frac{z_\mathrm{v}-o_{22}}{x_\mathrm{v}+\lambda_2} \end{bmatrix} + \begin{bmatrix} c_x \\ c_x \\ c_y \end{bmatrix}. \tag{4.16}$$

Now, taking the derivative of (4.16), the differential relationship between X_s and X_v is given by

$$\dot{X}_s = R_\alpha J_o \dot{X}_\mathrm{v} \tag{4.17}$$

where the Jacobian matrix $J_o \in \mathbb{R}^{3\times3}$ is defined as

$$J_o = \begin{bmatrix} \frac{1}{-y_\mathrm{v}+\lambda_1} & \frac{x_\mathrm{v}-o_{11}}{(-y_\mathrm{v}+\lambda_1)^2} & 0 \\ -\frac{y_\mathrm{v}-o_{21}}{(x_\mathrm{v}+\lambda_2)^2} & \frac{1}{x_\mathrm{v}+\lambda_2} & 0 \\ -\frac{z_\mathrm{v}-o_{22}}{(x_\mathrm{v}+\lambda_2)^2} & 0 & \frac{1}{x_\mathrm{v}+\lambda_2} \end{bmatrix}. \tag{4.18}$$

For our analysis to hold, the composite image Jacobian matrix J_o has to be invertible.

4.2.1.1 3D Cartesian Position

Equation (4.16) shows the relationship between the 3D pixel position $X_s = [u_1, u_2, v_2]^T$ and the 3D Cartesian position X_v. Now given projected image points in each camera as $[u_1, v_1]^T$ and $[u_2, v_2]^T$, we need to obtain the Cartesian position X_v of point P in frame O_v.

For simplicity, we defined the off-setted Cartesian vector $\overline{X_\mathrm{v}} \in \mathbb{R}^3$ as

$$\overline{X_\mathrm{v}} = \begin{bmatrix} \overline{x_\mathrm{v}} \\ \overline{y_\mathrm{v}} \\ \overline{z_\mathrm{v}} \end{bmatrix} = \begin{bmatrix} \frac{x_\mathrm{v}-o_{11}}{-y_\mathrm{v}+\lambda_1} \\ \frac{y_\mathrm{v}-o_{21}}{x_\mathrm{v}+\lambda_2} \\ \frac{z_\mathrm{v}-o_{22}}{x_\mathrm{v}+\lambda_2} \end{bmatrix} \tag{4.19}$$

According (4.15) and (4.19), we can get

$$\overline{X_\mathrm{v}} = (R_\alpha)^{-1}(X_s - \rho). \tag{4.20}$$

Then, given pixel positions $[u_1, v_1]^T$ and $[u_2, v_2]^T$, which implies $X_s = [u_1, u_2, v_2]^T$, we can compute the off-setted Cartesian vector $\overline{X_\mathrm{v}}$ through (4.20).

Extending (4.19), we have

$$\begin{cases} x_{\mathrm{v}} + y_{\mathrm{v}} \cdot \overline{x_{\mathrm{v}}} = \overline{x_{\mathrm{v}}} \cdot \lambda_1 + o_{11} \\ y_{\mathrm{v}} - x_{\mathrm{v}} \cdot \overline{y_{\mathrm{v}}} = \overline{y_{\mathrm{v}}} \cdot \lambda_2 + o_{21} \\ z_{\mathrm{v}} - x_{\mathrm{v}} \cdot \overline{z_{\mathrm{v}}} = \overline{z_{\mathrm{v}}} \cdot \lambda_2 + o_{22} \end{cases} \tag{4.21}$$

These equations can be written in matrix form

$$\begin{bmatrix} 1 & \overline{x_{\mathrm{v}}} & 0 \\ -\overline{y_{\mathrm{v}}} & 1 & 0 \\ -\overline{z_{\mathrm{v}}} & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\mathrm{v}} \\ y_{\mathrm{v}} \\ z_{\mathrm{v}} \end{bmatrix} = \begin{bmatrix} \overline{x_{\mathrm{v}}} \cdot \lambda_1 + o_{11} \\ \overline{y_{\mathrm{v}}} \cdot \lambda_2 + o_{21} \\ \overline{z_{\mathrm{v}}} \cdot \lambda_2 + o_{22} \end{bmatrix}. \tag{4.22}$$

Finally, from (4.22), 3D Cartesian position X_{v} can be obtained using

$$X_{\mathrm{v}} = \begin{bmatrix} x_{\mathrm{v}} \\ y_{\mathrm{v}} \\ z_{\mathrm{v}} \end{bmatrix} = \begin{bmatrix} 1 & \overline{x_{\mathrm{v}}} & 0 \\ -\overline{y_{\mathrm{v}}} & 1 & 0 \\ -\overline{z_{\mathrm{v}}} & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \overline{x_{\mathrm{v}}} \cdot \lambda_1 + o_{11} \\ \overline{y_{\mathrm{v}}} \cdot \lambda_2 + o_{21} \\ \overline{z_{\mathrm{v}}} \cdot \lambda_2 + o_{22} \end{bmatrix}. \tag{4.23}$$

where $[\overline{x_{\mathrm{v}}}, \overline{y_{\mathrm{v}}}, \overline{z_{\mathrm{v}}}]^T$ is obtained in (4.20).

4.2.1.2 Summarize

For two orthogonal cameras, given image projected points $[u_1, v_1]^T$ and $[u_2, v_2]^T$, which construct a 3D pixel position $X_s = [u_1, u_2, v_2]^T$, we can compute the 3D Cartesian position X_{v} with respect to the reference frame O_{v}. The process is summarized in Fig. 4.3 .

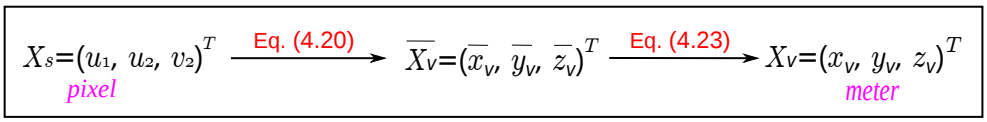


Figure 4.3: 3D position computation in orthogonal configuration.

4.2.2 Rotation Matrix between O_w and O_{v}

Fig. 4.4 and Fig. 4.5 show two cameras placed in orthogonal configuration with respect to reference frame O_{v}. Coordinate frame O_w is the world frame which is placed in the same position as O_{v} with different rotation. Then, in order to apply the models (4.15)-(4.23) proposed above and get 3D Cartesian position X_w with respect to world frame O_w, it is necessary to find out the rotation matrix between coordinates frames O_{v} and O_w.

The cameras are calibrated using the method described in [157], which gives the transformations between cameras and world frame O_w as

$$R^w_{c_1} = \begin{bmatrix} R^w_{c_1}(1,1) & R^w_{c_1}(1,2) & R^w_{c_1}(1,3) \\ R^w_{c_1}(2,1) & R^w_{c_1}(2,2) & R^w_{c_1}(2,3) \\ R^w_{c_1}(3,1) & R^w_{c_1}(3,2) & R^w_{c_1}(3,3) \end{bmatrix} \tag{4.24}$$

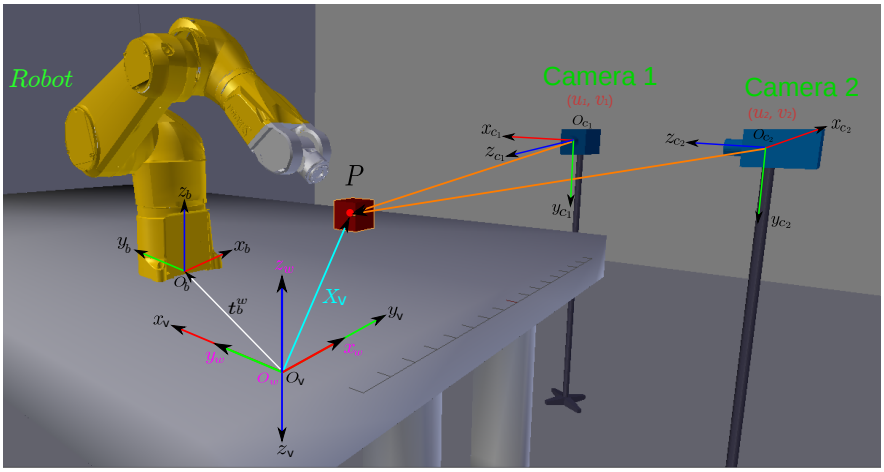


Figure 4.4: Geometric model of the orthogonal camera system.

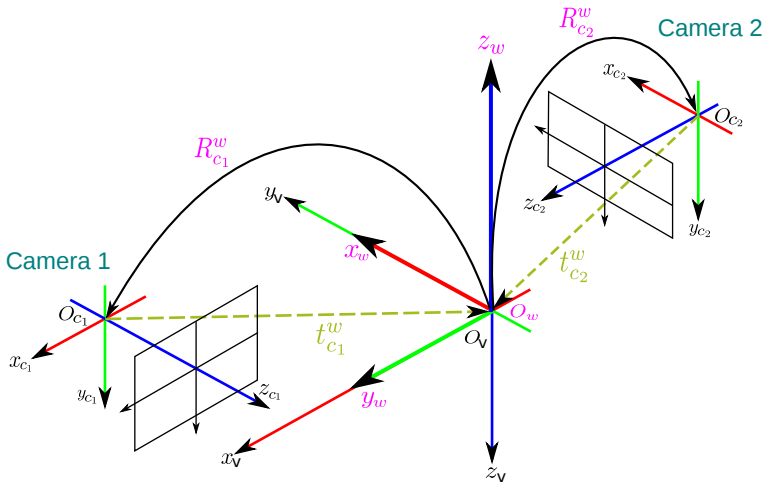


Figure 4.5: Two orthogonal camera configuration.

$$R_{c_2}^{w} = \begin{bmatrix} R_{c_2}^{w}(1,1) & R_{c_2}^{w}(1,2) & R_{c_2}^{w}(1,3) \\ R_{c_2}^{w}(2,1) & R_{c_2}^{w}(2,2) & R_{c_2}^{w}(2,3) \\ R_{c_2}^{w}(3,1) & R_{c_2}^{w}(3,2) & R_{c_2}^{w}(3,3) \end{bmatrix} \tag{4.25}$$

$$t_{c_1}^{w} = \begin{bmatrix} t_{c_1}^{w}(1,1) \\ t_{c_1}^{w}(2,1) \\ t_{c_1}^{w}(3,1) \end{bmatrix} \qquad t_{c_2}^{w} = \begin{bmatrix} t_{c_2}^{w}(1,1) \\ t_{c_2}^{w}(2,1) \\ t_{c_2}^{w}(3,1) \end{bmatrix}. \tag{4.26}$$

In previous subsection, we mentioned that the reference frame O_{v} is in a special case: $\theta_1 = \theta_2 = 0$. It means that coordinate frame O_{c_1} and O_{c_2} have no rotation around their z axes with respect to O_{v}. As shown in Fig. 4.5, x_{v} is the same direction as x_{c_1}, y_{v} is the same as

x_{c_2} and z_{v} is the same as y_{c_2}. Therefore, the rotation matrix (orientation) of the coordinate frame $O_w x_w y_w z_w$ with respect to $O_{\mathrm{v}} x_{\mathrm{v}} y_{\mathrm{v}} z_{\mathrm{v}}$ is the same as frame $O_w x_w y_w z_w$ with respect to $O_{\mathrm{v}} x_{c_1} x_{c_2} y_{c_2}$.

Using the definition and properties of a rotation matrix, the rotation between O_{v} and O_w is defined as

$$R^w_{\mathrm{v}} = \begin{bmatrix} x_w \cdot x_{c_1} & y_w \cdot x_{c_1} & z_w \cdot x_{c_1} \\ x_w \cdot x_{c_2} & y_w \cdot x_{c_2} & z_w \cdot x_{c_2} \\ x_w \cdot y_{c_2} & y_w \cdot y_{c_2} & z_w \cdot y_{c_2} \end{bmatrix} \tag{4.27}$$

$$= \begin{bmatrix} R^w_{c_1}(1,1) & R^w_{c_1}(1,2) & R^w_{c_1}(1,3) \\ R^w_{c_2}(1,1) & R^w_{c_2}(1,2) & R^w_{c_2}(1,3) \\ R^w_{c_2}(2,1) & R^w_{c_2}(2,2) & R^w_{c_2}(2,3) \end{bmatrix}. \tag{4.28}$$

Equation (4.28) represents the orientation of O_w with respect to the O_{v} coordinate frame. Then we can obtain

$$R^{\mathrm{v}}_w = (R^w_{\mathrm{v}})^T = \begin{bmatrix} R^w_{c_1}(1,1) & R^w_{c_2}(1,1) & R^w_{c_2}(2,1) \\ R^w_{c_1}(1,2) & R^w_{c_2}(1,2) & R^w_{c_2}(2,2) \\ R^w_{c_1}(1,3) & R^w_{c_2}(1,3) & R^w_{c_2}(2,3) \end{bmatrix}. \tag{4.29}$$

Finally, the 3D Cartesian position X_w with respect to world frame O_w is computed by

$$X_w = R^{\mathrm{v}}_w \cdot X_{\mathrm{v}}. \tag{4.30}$$

4.2.3 Avoid Visual Occlusion using four Orthogonal Cameras

As mentioned earlier, visual occlusion is one of the main issues when a visual servoing approach is employed to control a robot involved in a human-robot interaction task. In this section, we propose a new approach to avoid visual occlusions that emerge when an obstacle is placed between the camera and its target. It consists of placing four calibrated cameras in an orthogonal configuration around the workspace of the robot, as shown in Fig. 4.6.

As seen in the previous section, only two orthogonal cameras are needed to get the 3D Cartesian position of the target in real time. The only restriction is that the position of the cameras used to get the 3D position have to be orthogonal. The purpose of placing four cameras is to increase the possibility of keeping the targets in the field of view of cameras. Every time we can choose 2 orthogonal cameras which are not occluded by obstacles to compute the 3D Cartesian position.

In order to get the 3D position of a target relative to the coordinate frame O_w through visual information provided by four cameras placed orthogonally around a reference coordinate frame O_{v}, the following steps have to be applied:

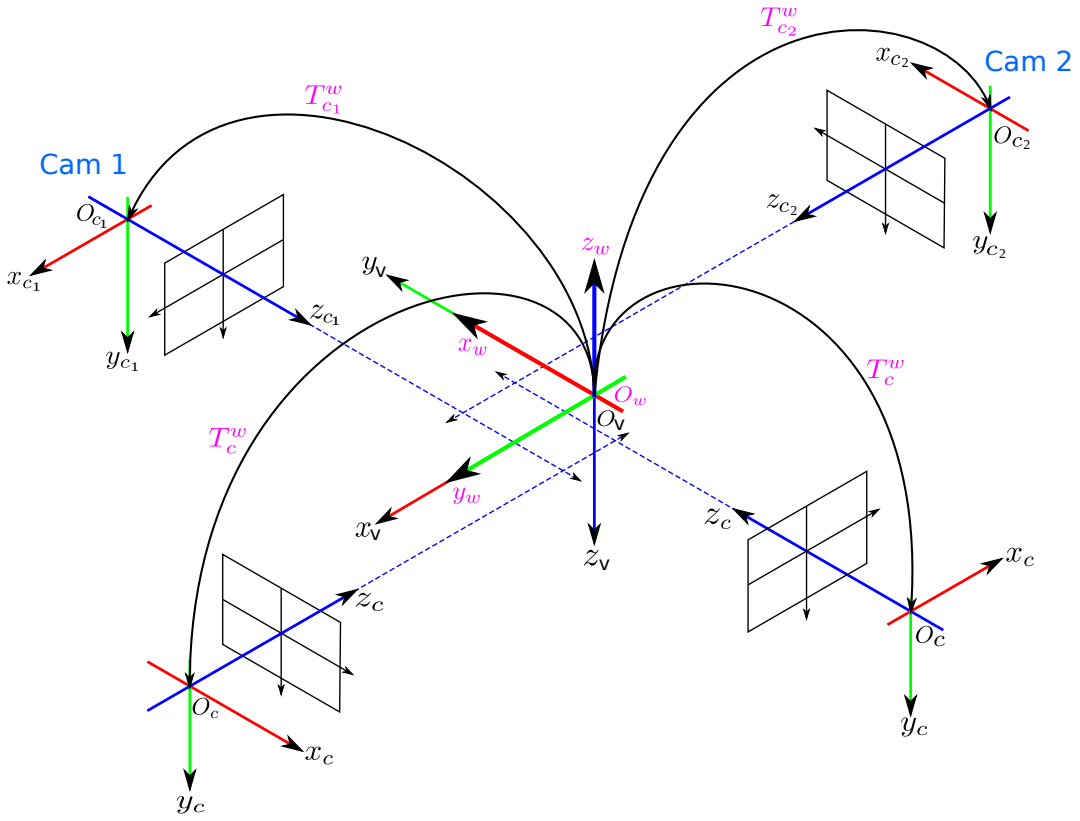


Figure 4.6: Four orthogonal camera configuration.

1. Place four cameras in orthogonal configuration, as shown in the Fig. 4.6.
2. Calibrate the four cameras orthogonal using the method described in [157].
3. Choose two orthogonal cameras from four cameras. The selected two cameras have to keep tracking the targets.
4. Decide camera one and two. The condition is that the camera one is placed on the positive side of the camera two. The choice of these two cameras decides the reference coordinate frame O_{v}.
5. Apply equation (4.23) to get the 3D position X_{v} relative to the coordinate frame O_{v}.
6. Transform the 3D position from coordinate frame O_{v} to world frame O_w using a rotation matrix R_w^{v} (4.29), to get X_w, shown as (4.30).

Remarks:

1. **The algorithm selects two orthogonal cameras from four cameras automatically and keeps changing the selection in real time.**

2. **It is important to mention that the configuration of reference frame O_{v} is always changing since it depends the selected two orthogonal cameras while the world coordinate frame O_w is fixed. Hence, the rotation (R^{v}_w) between frame O_{v} and O_w keeps changing. However, equation (4.29) provides a unifying solution for the orientation, which means that the model to get the 3D Cartesian position is never changed. The only changed values are $R^w_{c_1}$ and $R^w_{c_2}$ due to selected orthogonal cameras.**

3. **The model of the cameras never changes, making the algorithm fast.**

4.3 Uncalibrated Stereo Vision System

In previous section, we reviewed a methodology designed to obtain the 3D position of a target through visual information employing at least two cameras placed orthogonally. Similarly, in this section we reviewed another way to get the 3D position of a target using two cameras in a stereo configuration. The geometric camera model of stereo system is shown in Fig. 4.7.

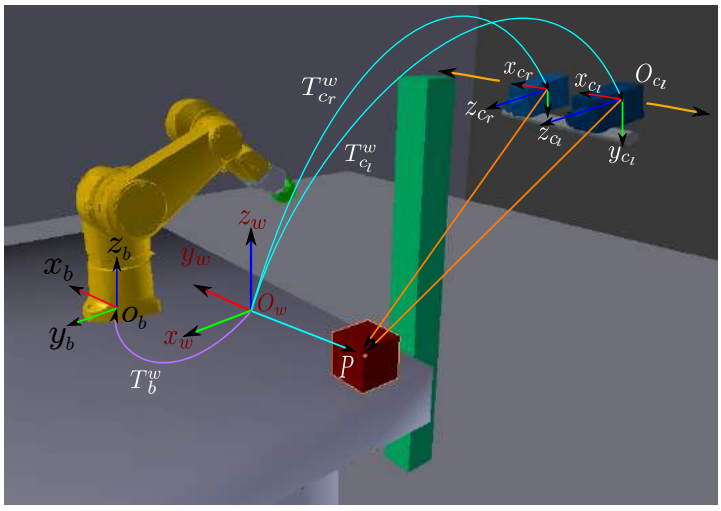


Figure 4.7: Geometric model of the stereo camera system.

4.3.1 3D Recovery from Stereo System

In the two-camera setup, each camera gives a different 3D back projection line. These two back projection lines usually coincide at exactly one point (Fig. 4.8 (a)). Given a stereo setup, it can find the 3D position of a point P by observing its position in two different cameras[156].

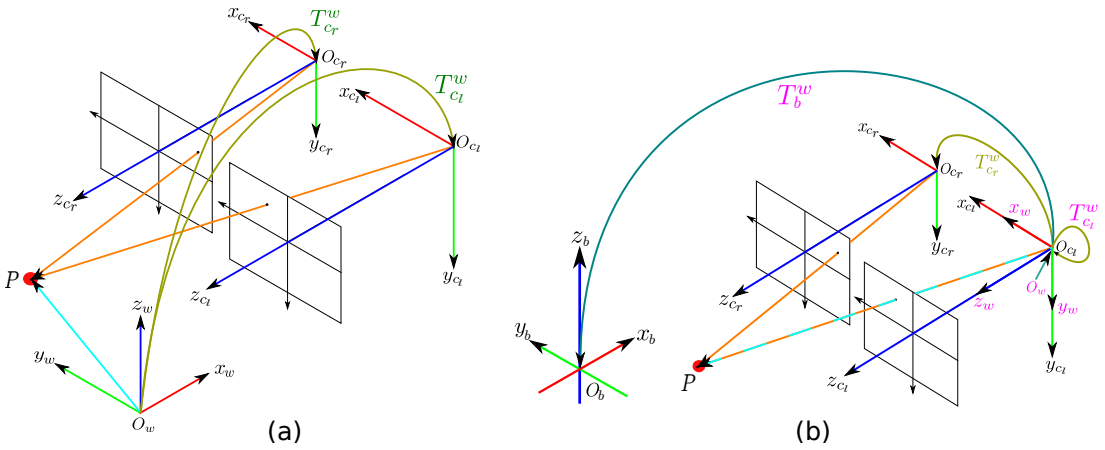


Figure 4.8: The stereo camera model: (a) shows a 3D point projected in two cameras, (b) the world coordinate frame is on the left camera frame point.

For two cameras with two projection matrices P_1 and P_2 we have

$$s_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad s_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = P_2 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4.31}$$

Given the observed image points in each camera (u_1, v_1) and (u_2, v_2), we want to solve the 3D position in the world frame $X_w = [x, y, z]^T$.

Multiplying out in terms of the P_i gives:

$$\begin{aligned} xu_i p^i_{31} + yu_i p^i_{32} + zu_i p^i_{33} + u_i p^i_{34} &= xp^i_{11} + yp^i_{12} + zp^i_{13} + p^i_{14} \\ xv_i p^i_{31} + yv_i p^i_{32} + zv_i p^i_{33} + v_i p^i_{34} &= xp^i_{21} + yp^i_{22} + zp^i_{23} + p^i_{24} \end{aligned} \tag{4.32}$$

where i is 1 or 2, and p^i_{11} denotes element (1,1) in P_i.

Since we want to solve this for $X_w = [x, y, z]^T$, these terms are collected on the left and other terms on the right, and the equation is then rewritten as:

$$AX_w = B \tag{4.33}$$

$$A = \begin{bmatrix} u_1 p^1_{31} - p^1_{11} & u_1 p^1_{32} - p^1_{12} & u_1 p^1_{33} - p^1_{13} \\ v_1 p^1_{31} - p^1_{21} & v_1 p^1_{32} - p^1_{22} & v_1 p^1_{33} - p^1_{23} \\ u_2 p^2_{31} - p^2_{11} & u_2 p^2_{32} - p^2_{12} & u_2 p^2_{33} - p^2_{13} \\ v_2 p^2_{31} - p^2_{21} & v_2 p^2_{32} - p^2_{22} & v_2 p^2_{33} - p^2_{23} \end{bmatrix} \tag{4.34}$$

and

$$B = \begin{bmatrix} p^1_{14} - u_1 p^1_{34} \\ p^1_{24} - v_1 p^1_{34} \\ p^2_{14} - u_2 p^2_{34} \\ p^2_{14} - v_2 p^2_{34} \end{bmatrix}. \tag{4.35}$$

We need to solve this for X_w. Since A is not square, the pseudo inverse is used:

$$X_w = (A^T A)^{-1} A^T B. \tag{4.36}$$

4.3.2 Stereo Camera Model

If we put the world coordinate frame O_w at the left camera origin point O_{c_l} (see Fig. 4.8 (b)), then given the relation between right camera and left camera (defined by the orientation matrix $R_r^l \in SO(3)$ and the translation vector $t_r^l \in \mathbb{R}^{3\times 1}$), we can define the projection matrices P_{c_l} and $P_{c_r} \in \mathbb{R}^{3\times 4}$ for each camera as

$$P_{c_l} = K_l \begin{bmatrix} I_{3\times 3} & 0_{3\times 1} \end{bmatrix} \qquad P_{c_r} = K_r \begin{bmatrix} R_r^l & t_r^l \end{bmatrix} \tag{4.37}$$

where K_l and K_r are the intrinsic camera matrices of the left and right cameras respectively[1].

Then, defining the observed image points in each camera as $p_l = [u_l, v_l]^T$, $p_r = [u_r, v_r]^T$, we can use triangulation [156] to compute the *relative* position $X_w = X_{c_l} = [x_c, y_c, z_c]^T$ with respect to the left camera O_{c_l}, which is shown in (4.36).

In order to control the robot, we need to compute the 3D position with respect to the robot base coordinate frame O_b. By knowing the transformation between O_w (the same as O_{c_l}) and O_b, $T_b^w = T_b^{c_l} = [R_b^{c_l} \quad t_b^{c_l}]$, we can get the robot end-effector position

$$X_{ef} = X_b = R_b^{c_l} X_{c_l} + t_b^{c_l}. \tag{4.38}$$

4.3.3 On-line Estimation of Rotation for the Stereo System

We define this system as uncalibrated because we not only assume that the calibration of the stereo vision system (left camera O_{c_l}) with respect to the robot base frame (O_b) is unknown, but we also consider the possibility of manually moving the camera to a better position. When the target object is occluded (visual occlusions), the stereo camera system can be moved to another position to maintain the target in the field of camera view. After moving the stereo cameras, an on-line estimation of the transformation $T_b^{c_l}$ (or T_b^w) is needed. This estimation uses the real-time information generated by the robot without stopping robot during the task execution.

Finding the rotation and translation between two sets of corresponding 3D point data, such that they are aligned, is a common problem. An illustration of the problem is shown below for the simplest case of 3 corresponding points (the minimum number of points required to solve).

As shown in Fig. 4.9, let the given 3D datasets A and B be represented in two coordinate frames. We want to find the best rotation R and translation t that will align the points in dataset A to dataset B.

$$P_B^i = RP_A^i + t, \tag{4.39}$$

[1]These parameters can be computed off-line.

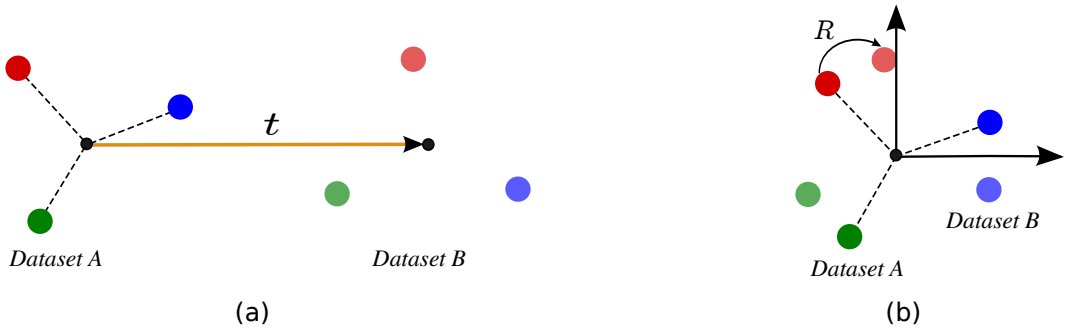


Figure 4.9: Two datasets of corresponding 3D points.

where $i = 1, 2, ..., n$, $(n \geq 3)$, $P = [x, y, z]^T$, P_A and P_B are points in dataset A and B respectively.

Finding the optimal rigid transformation matrix can be broken down into steps:

1. **Find the centroids of both dataset**

   The centroids $C_{A,B}$ are the average points and can be calculated as follows:

   $$C_A = \frac{1}{n}\sum_{i=1}^{n} P_A^i \qquad C_B = \frac{1}{n}\sum_{i=1}^{n} P_B^i. \tag{4.40}$$

2. **Estimation of Base Orientation** R

   There are several ways of finding optimal rotations between datasets. The easiest way is using *Singular Value Decomposition (SVD)*, as discussed in [158]. Define a 3×3 matrix M as

   $$M = \sum_{i=1}^{n} \left(P_A^i - C_A\right)\left(P_B^i - C_B\right)^T, \tag{4.41}$$

   then a least-squares fit of the rotation matrix can be written as

   $$R = U\,diag\left(1, 1, det\left(UV^T\right)\right)V^T \tag{4.42}$$

   where $(U, S, V) = svd(M)$.

3. **Compute the translation** t

   After the rotation R is obtained, t can be solved as:

   $$t = C_B - RC_A. \tag{4.43}$$

   In our case, P_B is defined as a set of end-effector 3D positions X_b with respect to robot base frame O_b, while P_A denotes a set of the corresponding position vectors X_{c_l} in stereo camera frame O_{c_l}. Hence, the estimated orientation $R = R_b^{c_l}$.

4.4 Discussion

In this chapter, two camera models were introduced to avoid visual occlusions for PBVS. The first camera model consists of four orthogonal cameras with precise calibration. The system automatic selects two composite cameras that keep tracking the targets to obtain the 3D Cartesian position for the robot controller. The second camera model is an uncalibrated stereo camera system, which can be moved manually to maintain the targets in the field of camera view when visual occlusion occurs. Both these approaches have been tested on a real industrial robot in a Human-robot-interaction scenario, see Appendix B.

Both methods have their advantages and disadvantages. The orthogonal camera model can provide 3D pixel positions on images. Each component of the 3D pixel positions is orthogonal and independent. If the 3D pixel positions are chosen as visual features for visual servoing, then a linear mapping between those pixel positions and Cartesian positions is obtained, which can be used to design a decoupled controller. However, for the orthogonal configuration, a complicated camera placement and accurate calibration is required. It is physically difficult to achieve this in real world.

In contrast, the uncalibrated stereo camera system does not require the exact calibration and can be moved. However, it can not generate independent image features. The choice of features directly influences on the performance of the control system and on the ability to analyze the closed-loop dynamics. If the classical features extracted from the stereo cameras are used as inputs for visual servoing, then some common problems such as image singularity and local minima may occur, which lead to stability and convergence issues.

The ideal case is to find a camera configuration which can combine the advantages of both approaches. In next chapter, we will propose a new camera model which combines these two camera configurations by mapping the stereo configuration into a virtual orthogonal camera arrangement. The measurements obtained by the virtual camera model can generate 3D orthogonal pixel positions, which are chosen as the visual features for visual servoing.

Chapter 5

6D Image-Based Visual Servoing

In this chapter, we present an approach (6DVS) to control a 6 DOF manipulator using an uncalibrated visual servoing (VS) approach that addresses the challenges of choosing proper image features for target objects and designing a VS controller to enhance the tracking performance. The system is composed of the manipulator coupled with a stereo vision system in the eye-to-hand configuration, see Fig. 5.1. We construct a new *virtual visual space*, using a novel stereo camera model employing virtual orthogonal cameras. A 6D pixel pose vector (W_s) is extracted from this virtual space. It represents the image positions as 6 linearly independent and orthogonal signals, which are used as inputs for visual servoing instead of the classical visual features. This leads to generate a full-rank 6×6 image Jacobian matrix which allows avoiding classical problems, such as image space singularities and local minima. Furthermore, several tasks are evaluated in simulation to show the performance and properties of the proposed method.

Figure 5.1: Real world experimental setup.

5.1 Problem Statement

5.1.1 Classical IBVS with a Stereo vision System

Based on the existing literature in this field, we can conclude that two main aspects have the impact on the behavior of a visual servoing scheme: the selection of *visual features* used as inputs of the control law and the form of the *control scheme*.

As described in [9], in classical IBVS, the image point position $s = [u, v]^T$ is chosen as the visual feature and the control scheme is

$$\dot{s} = L_s \dot{X}_c \tag{5.1}$$

in which the interaction matrix L_s is defined as (2.17).

$$L_s = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{bmatrix} \tag{5.2}$$

where f is the focal length expressed in pixel units.

If a stereo vision system is used and a 3D point is visible in both left and right images, it is possible to use as visual features

$$s = [p_l, p_r]^T = [u_l, v_l, u_r, v_r]^T \tag{5.3}$$

to represent the point by stacking in s the x and y coordinates of the observed point in the left and right images [10]. Therefore, the corresponding equation given in (5.1) is expressed in both left and right camera frame. More precisely, we have:

$$\dot{s} = L_s \dot{X}_c = \begin{bmatrix} L_{s_l} \\ L_{s_r} \end{bmatrix} \dot{X}_c. \tag{5.4}$$

To control a 6 DOF robot, at least three points are necessary. If we have a vector $W_b = [X_b, \theta_b]^T = [x_b, y_b, z_b, \alpha_b, \beta_b, \gamma_b]^T \in \mathbb{R}^{6\times 1}$, which is the pose of the end-effector in the robot base frame (in this case we choose Euler angles to representation the orientation of the end-effector), and a feature vector $s = [u_{1_l}, v_{1_l}, u_{1_r}, v_{1_r}, ..., u_{p_l}, v_{p_l}, u_{p_r}, v_{p_r}]^T \in \mathbb{R}^{4p\times 1}$, which contains p image points. The mapping between $\dot{s}$ and $\dot{W}_b$ is given by

$$\underbrace{\dot{s}}_{4p\times 1} = \underbrace{J_x}_{4p\times 6} \underbrace{\dot{W}_b}_{6\times 1} \tag{5.5}$$

where ($J_x = [L_{s_1}, ..., L_{s_p}]^T \cdot V_c^b$) $\in \mathbb{R}^{4p\times 6}$ is known as the image Jacobian, L_{s_i} is given by (5.4) and (5.2), and $V_c^b \in \mathbb{R}^{6\times 6}$ is the mapping to transform velocities expressed in the robot base frame to the camera frame, defined as (2.5).

5.1.2 The problem of Classical IBVS

If we consider ΔW_b as the input to a robot controller, then we need to compute the inverse mapping of $\dot{s}$ as

$$\Delta W_b = J_x^+ \Delta s, \tag{5.6}$$

where $\Delta *$ is an error function defined in the space $*$, $J_x^+ \in \mathbb{R}^{6\times 4p}$ is chosen as the Moore-Penrose pseudo inverse of J_x, which leads to the two characteristic problems of the IBVS method: the feature (image) space singularities and local minima. For most IBVS approaches we have $4p > 6$. In this case, the image Jacobian is singular when $rank(J_x) < 6$, while the image local minima is defined as the set of image locations $\Omega_s = \left\{ s | \dot{W}_b = 0, \text{but } \Delta s \neq 0, \Delta W_b \neq 0, \forall s \in \mathbb{R}^{4p\times 1} \right\}$ when using redundant image features. Examples of the problems generated by the local minima conditions are illustrated in [10] and [159].

5.1.3 Algorithm Design

In image-based control approach, the ideal case is to choose visual features with good decoupling and linearizing properties [35, 160] where the interaction matrix has neither local minima nor singularities. In our work, we propose a new camera model which can map the the classical image features s to a new visual representation defined as $W_s = [X_s, \theta_s]^T = [x_s, y_s, z_s, \alpha_s, \beta_s, \gamma_s]^T \in \mathbb{R}^{6\times 1}$. In this case, W_s is a 6D pixel pose vector defined in a 3D Image space (we call this space the *virtual visual space*). This visual pose is measured in pixels and it is composed of 3D visual position and 3D visual orientation.

The algorithm design and concept of the proposed 6DVS is shown in Fig. 5.2. The new visual features W_s has six orthogonal elements and they are linearly independent. If we consider W_s as the inputs to visual servoing control, then a new mapping as

$$\dot{W}_s = J_{\text{img}} \dot{W}_b \tag{5.7}$$

$$\begin{bmatrix} \dot{X}_s \\ \dot{\theta}_s \end{bmatrix}_{6\times 1} = \underbrace{J_{\text{img}}}_{6\times 6} \begin{bmatrix} \dot{X}_b \\ \dot{\theta}_b \end{bmatrix}_{6\times 1} = J_{\text{img}} \cdot J_a(q) \cdot q \tag{5.8}$$

is obtained, where $J_a(q)$ is the Jacobian matrix of the robot manipulator and q is the robot joint position.

The advantage of this new mapping is that, a full-rank *image Jacobian* matrix ($J_{\text{img}} \in \mathbb{R}^{6\times 6}$) can be obtained, and if we choose camera parameters properly, then the image space singularities and local minima can be avoided.

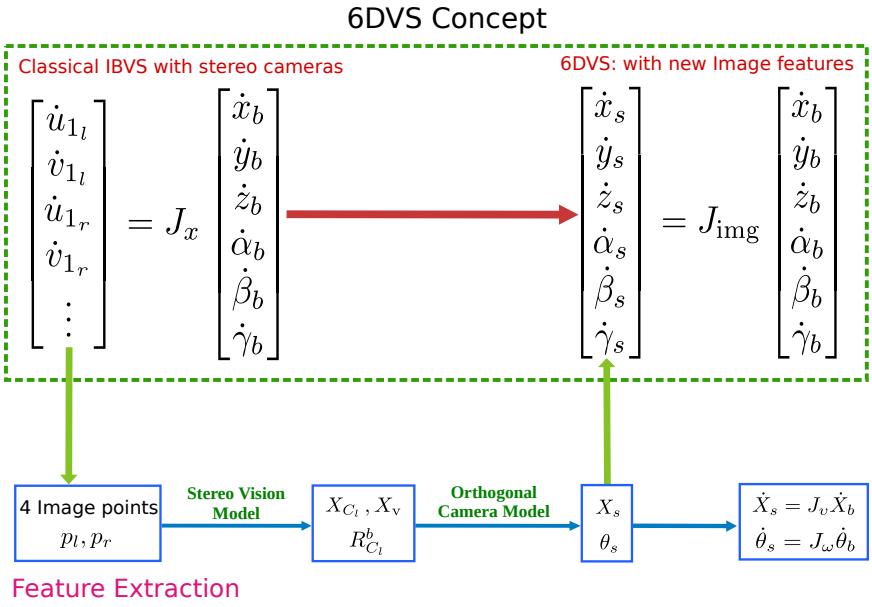


Figure 5.2: Algorithm design: mapping from the classical IBVS features to new orthogonal features in the virtual visual space.

5.2 Image Jacobian for 3D position

5.2.1 3D Camera Model

In order to get the new pixel features W_s, we propose a new camera model, where a real stereo camera model employing virtual orthogonal cameras is used to tracking the objects. First we use standard stereo vision model to compute the object position, then the position is projected onto two **virtual** orthogonal cameras, whose reference frame is located in the left stereo camera frame. Fig. 5.3 shows this new camera model.

The new camera model can construct a new *virtual visual space*, and is used to map 6D poses from Cartesian space to this virtual visual space. Each component of the 6D pose vector defined in this virtual visual space is linearly independent, leading to a full-rank 6×6 image Jacobian matrix, which allows avoiding classical problems, such as image space singularities and local minima. Furthermore, the control for rotational and translational motion of robot is decoupled due to the diagonal image Jacobian.

The obtained new *image Jacobian matrix* ($J_{\text{img}} \in \mathbb{R}^{6\times6}$) describes the relationship between the motion of selected six image features and the velocity of the robot end-effector. It has a block decoupled structure, which is divided into *position image Jacobian* ($J_v \in \mathbb{R}^{3\times3}$) and *orientation image Jacobian* ($J_\omega \in \mathbb{R}^{3\times3}$).

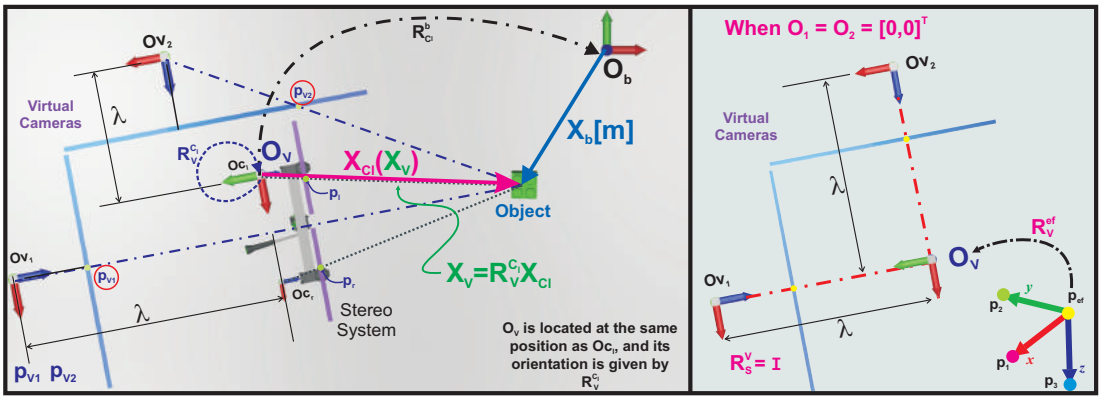


Figure 5.3: Image projections for 3D vision model to get the virtual visual space.

- **Fig. 5.3 (a)**: The figure depicts the different coordinate frames used to obtain a general 3D virtual visual space. $X_b \in \mathbb{R}^{3\times1}$ is the position in meters [m] of an *Object* with respect to the world coordinate frame (wcf) denoted by O_b. O_{C_l} and O_{C_r} are the coordinate frames for the left and right cameras, respectively. $R^b_{C_l} \in SO(3)$ represents the orientation of wcf with respect to the left camera. O_{v} is a reference coordinate frame for the virtual orthogonal cameras $O_{\mathrm{v}_{1,2}}$ where $R^{C_l}_{\mathrm{v}} \in SO(3)$ is its orientation with respect to O_{C_l}. The vectors $p_{\mathrm{v}_i} \in \mathbb{R}^{2\times1}$ represents the projection of the *Object* in the virtual cameras O_{v_i}.

- **Fig. 5.3 (b)**: 4 points (p_{ef}, p_1, p_2, p_3) are attached to the end-effector to represent the orientation frames of virtual visual space, with the optical center offsets of virtual cameras being zero ($O_1 = O_2 = [0,0]^T$).

$J_v \in \mathbb{R}^{3\times3}$ describes the relationship between the velocities of 3D Cartesian position $\dot{X}_b$ $(meters)$ and 3D visual position $\dot{X}_s$ $(pixels)$. The key idea of this model is to combine the stereo camera model with a virtual composite camera model to get a full-rank image Jacobian.

Fig. 5.3 shows the 3D camera model, which can be computed in two steps:

1. The standard stereo vision model [156] is used to analytically recover the 3D *relative* position (X_{C_l}) of an object with respect to the reference frame of the stereo system O_{C_l}.

2. The Cartesian position X_{C_l} is projected into two virtual cameras O_{v_1} and O_{v_2}.

This projection is a crucial step, since it modifies the dimension of the mapping from two 2D-image feature measurements of all p points $(s = [p_{l_1}, p_{r_1}, ...p_{l_p}, p_{r_p}]^T \in \mathbb{R}^{4p\times1})$ to a single 3D visual position vector $X_s \in \mathbb{R}^{3\times1}$ in the virtual visual space. Since s represents the position

$X_{C_l} \in \mathbb{R}^{3\times 1}$ in the image feature space, the maximum number of independent elements of s is 3. Therefore, there are $4p-3$ dependent elements in s. In this thesis, we propose a virtual projection that reduces the dimension of s and extract 3 linearly independent elements to get a full-rank image Jacobian (J_v).

5.2.2 Stereo Vision Model

The relation between right camera and left camera is given by a orientation matrix $R_r^l \in SO(3)$ and a translation vector $t_r^l \in \mathbb{R}^{3\times 1}$. Using this transformation we can define the projection matrices P_{C_l} and $P_{C_r} \in \mathbb{R}^{3\times 4}$ for each camera as

$$P_{C_l} = K_l \begin{bmatrix} I_{3\times 3} & 0_{3\times 1} \end{bmatrix} \qquad P_{C_r} = K_r \begin{bmatrix} R_r^l & t_r^l \end{bmatrix} \tag{5.9}$$

where K_l and K_r are the intrinsic camera matrices of the left and right cameras[1].

Then, defining the observed image points in each camera as $p_l = [u_l, v_l]^T$, $p_r = [u_r, v_r]^T$, we can use triangulation [156] to compute the *relative* position $X_{C_l} = [x_c, y_c, z_c]^T$ of the point X_b with respect to the left camera O_{C_l}. This can be done by solving the system

$$A \begin{bmatrix} X_{C_l} \\ 1 \end{bmatrix} = 0 \tag{5.10}$$

where

$$A = \begin{bmatrix} u_l p_{31}^l - p_{11}^l & u_l p_{32}^l - p_{12}^l & u_l p_{33}^l - p_{13}^l & u_l p_{34}^l - p_{14}^l \\ v_l p_{31}^l - p_{21}^l & v_l p_{32}^l - p_{22}^l & v_l p_{33}^l - p_{23}^l & v_l p_{34}^l - p_{24}^l \\ u_r p_{31}^r - p_{11}^r & u_r p_{32}^r - p_{12}^r & u_r p_{33}^r - p_{13}^r & u_r p_{34}^r - p_{14}^r \\ v_r p_{31}^r - p_{21}^r & v_r p_{32}^r - p_{22}^r & v_r p_{33}^r - p_{23}^r & v_r p_{34}^r - p_{14}^r \end{bmatrix} \tag{5.11}$$

where p_{ij}^l denotes the element (i,j) in P_{C_l}, and p_{ij}^r denotes the element (i,j) in P_{C_r}.

Before integrating the stereo cameras model with the virtual composite model, a re-orientation of the coordinate frame O_{C_l} to a new coordinate frame O_v with the same origin is required. The projection $X_\mathrm{v} = [x_\mathrm{v}, y_\mathrm{v}, z_\mathrm{v}]^T$ of X_{C_l} in O_v is defined as (Fig. 5.3)

$$X_\mathrm{v} = R_\mathrm{v}^{C_l} X_{C_l} \tag{5.12}$$

where $R_\mathrm{v}^{C_l}$ is the orientation of the reference frame O_v with respect to left stereo camera O_{C_l}.

Remark:

The reference frame O_v is fixed on the left camera frame and defined by the user. Therefore, it is assumed to be known. In other words, if we move the whole stereo camera system, the orientation matrix $R_\mathrm{v}^{C_l}$ is not modified. From Fig. 5.4, we can obtain $R_\mathrm{v}^{C_l}$ is a constant matrix

[1] These parameters can be computed off-line.

as

$$R_{\mathrm{v}}^{C_l} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{5.13}$$

5.2.3 Virtual Composite Camera Model

5.2.3.1 Composite Camera Analysis

We propose a *Composite Camera Model* as a minimization method for the feature point space s. The advantage of the orthogonal camera configuration is the linear independence of the visual measurements, as presented in section 4.2 (X_s in Fig. 5.4). It represents the image positions as 3 orthogonal signals which can be used as inputs for visual servoing instead of the classical features extracted from the stereo cameras.

While this orthogonal camera configuration is useful for generating independent feature vectors, it requires a complex and accurate arrangement of the cameras which is physically hard to achieve. A stereo camera arrangement, on the other hand, is easy to configure and move, but doesn't provide independent image features. We propose to combine the benefits of these two configurations by mapping the projections of the stereo configuration into a virtual composite cameras arrangement. The measurements obtained from the virtual camera model can generate a 3D pixel position.

5.2.3.2 Virtual Composite Camera Model Generation

In order to compute the virtual visual space, we define two virtual cameras attached to the stereo cameras system using the coordinate frame O_{v} (Fig. 5.4). We use the pinhole camera model [156] to project the relative position X_{v} to each of the virtual cameras O_{v_1} and O_{v_2}.

The model for the virtual camera 1 is given by

$$p_{\mathrm{v}_1} = \begin{bmatrix} u_{\mathrm{v}_1} \\ v_{\mathrm{v}_1} \end{bmatrix} = \frac{1}{-y_{\mathrm{v}} + \lambda} \alpha R(\phi) \begin{bmatrix} x_{\mathrm{v}} - o_{11} \\ z_{\mathrm{v}} - o_{12} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{5.14}$$

where ϕ is the rotation angle of the virtual camera along its optical axis, $O_1 = [o_{11}, o_{12}]^T$ is the projection position of the optical center with respect to the coordinate frame O_{v}, $C_1 = [c_x, c_y]^T$ is the position of the principal point in the image plane, λ is the distance from the virtual camera coordinate frame O_{v_1} to the reference frame O_{v} along its optical axis, α and the rotation matrix $R(\phi)$ are defined as:

$$\alpha = \begin{bmatrix} f\beta & 0 \\ 0 & f\beta \end{bmatrix} \qquad R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \tag{5.15}$$

where f is the focal length of the lens used, β is the magnification factor of the camera.

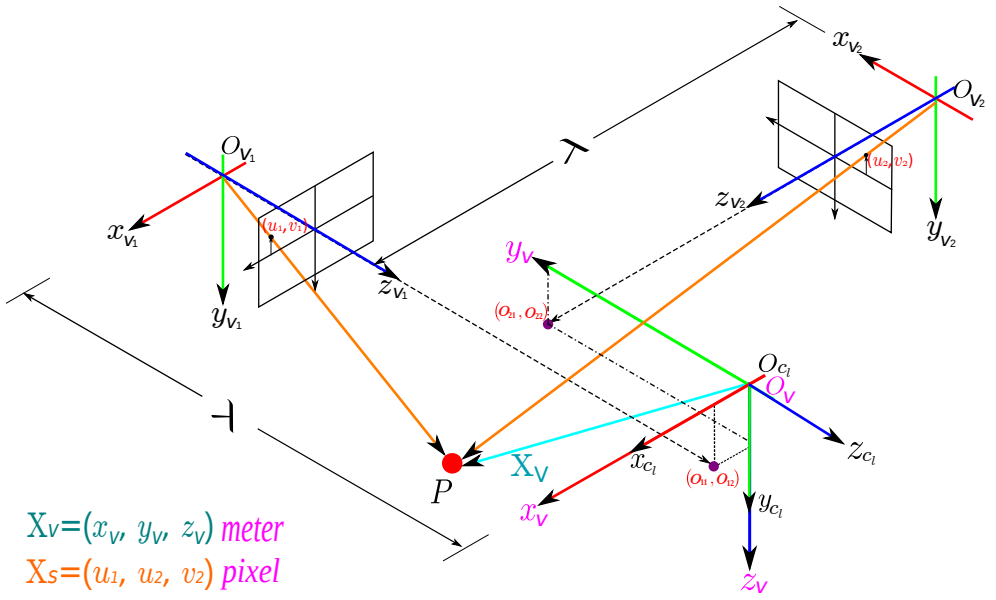


Figure 5.4: Placement of the virtual composite camera model with respect to left camera.

Since this model represents a user-defined virtual camera, all its parameters (extrinsic and intrinsic[1]) are known, in fact, in the defined configuration of the virtual cameras $\phi = 0$[2] (Fig. 5.4).

Similarly, the model for virtual camera 2 is defined as:

$$p_{v_2} = \begin{bmatrix} u_{v_2} \\ v_{v_2} \end{bmatrix} = \frac{1}{x_v + \lambda} \alpha R(\phi) \begin{bmatrix} y_v - o_{21} \\ z_v - o_{22} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{5.16}$$

In order to construct the 3D virtual visual space $X_s \in \mathbb{R}^{3\times 1}$, we combine both virtual camera models as follows.

Using the properties of the rotation matrix $R(\phi)$ and the fact that α is a diagonal matrix, from (5.14), u_{v_1} can be written in the form

$$u_{v_1} = \gamma_1 \frac{x_v - o_{11}}{-y_v + \lambda} - \gamma_2 v_{v_1} + \gamma_3 \tag{5.17}$$

where the constant parameters $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}$ are explicitly defined as

$$\gamma_1 = \frac{f\beta}{\cos\phi}, \quad \gamma_2 = \tan(\phi), \quad \text{and } \gamma_3 = c_x + c_y\gamma_2. \tag{5.18}$$

[1]Since the virtual cameras are user-defined, we can set the same intrinsic parameters and λ values for both cameras.

[2]The reason to introduce the auxiliary coordinate frame O_v is to simplify the composite camera model by rotating the coordinate frame O_{C_l} in an specific orientation, such as $\phi = 0$.

Based on (5.16) and (5.17), we define the *3D Visual Camera Model* (O_s) representation $X_s = [x_s, y_s, z_s]^T$ [1] using the orthogonal elements $[u_{v_1}, u_{v_2}, v_{v_2}]^T$ as

$$X_s = \begin{bmatrix} u_{v_1} \\ u_{v_2} \\ v_{v_2} \end{bmatrix} = \overbrace{\begin{bmatrix} \gamma_1 & 0_{1\times 2} \\ 0_{2\times 1} & \alpha R(\phi) \end{bmatrix}}^{R_\alpha} \begin{bmatrix} \frac{x_v - o_{11}}{-y_v + \lambda} \\ \frac{y_v - o_{21}}{x_v + \lambda} \\ \frac{z_v - o_{22}}{x_v + \lambda} \end{bmatrix} + \rho \tag{5.19}$$

where $\rho = [\gamma_3 - \gamma_2 v_{v_1}, c_x, c_y]^T$ and the pixel position X_s constructs the new virtual visual Space.

Given that $\phi = 0$, then $\gamma_1 = f\beta, \quad \gamma_2 = 0, \quad \gamma_3 = c_x$, implies that $\rho = [c_x, c_x, c_y]^T$ and $R_\alpha = diag(f\beta) \in \mathbb{R}^{3\times 3}$. Therefore, the mapping in (5.19) can be simplified as

$$X_s = diag(f\beta) \begin{bmatrix} \frac{x_v - o_{11}}{-y_v + \lambda} \\ \frac{y_v - o_{21}}{x_v + \lambda} \\ \frac{z_v - o_{22}}{x_v + \lambda} \end{bmatrix} + \begin{bmatrix} c_x \\ c_x \\ c_y \end{bmatrix}. \tag{5.20}$$

The velocity mapping can be obtained with the time derivative of (5.20), with combing (5.12) as follows:

$$\dot{X}_s = \overbrace{R_\alpha J_o}^{J_\alpha} \dot{X}_v = J_\alpha \dot{X}_v = (J_\alpha R_v^{C_l}) \dot{X}_{C_l} \tag{5.21}$$

where the Jacobian matrix $J_o \in \mathbb{R}^{3\times 3}$ is defined as

$$J_o = \begin{bmatrix} \frac{1}{-y_v + \lambda} & \frac{x_v - o_{11}}{(-y_v + \lambda)^2} & 0 \\ -\frac{y_v - o_{21}}{(x_v + \lambda)^2} & \frac{1}{x_v + \lambda} & 0 \\ -\frac{z_v - o_{22}}{(x_v + \lambda)^2} & 0 & \frac{1}{x_v + \lambda} \end{bmatrix}. \tag{5.22}$$

This *composite image Jacobian* J_α represents the mapping from velocities defined in the reference frame O_v to velocities in virtual visual space. In order to complete the 3D visual mapping we need to map from left camera O_{C_l} to robot base frame to O_b, see Fig. 5.3:

$$X_{C_l} = R_{C_l}^b X_b + t_{C_l}^b \tag{5.23}$$

where $R_{C_l}^b$ and $t_{C_l}^b$ are the transformation between frame O_{C_l} and O_b.

Taking the time derivative of (5.23), equation (5.21) can be rewritten in the form

$$\dot{X}_s = \overbrace{J_\alpha (R_v^{C_l} R_{C_l}^b)}^{J_v} \dot{X}_b \tag{5.24}$$

$$= J_v \dot{X}_b \tag{5.25}$$

where we define the Jacobian $J_v \in \mathbb{R}^{3\times 3}$ as the *position image Jacobian*.

[1] We use X_s instead of the classical notation s because X_s is more than an image feature measurement, in fact, it defines a position vector in the 3D virtual visual space.

Remark: Virtual Cameras.

The two virtual cameras are selected in such a way that their optical axes intersect at 90°. Since the cameras are virtual they have infinite field of view, and pixel positions X_s can be either negative or positive.

5.3 Image Jacobian for 3D Orientation

In the previous section, a point in Cartesian space X_b can be projected to a vector X_s defined in the virtual visual space. If we set the optical center offsets of the virtual cameras as $O_1 = O_2 = [0,0]^T$ (Fig. 5.3 (b)), then the mapping (5.20) can be simplified as

$$X_s = diag(f\beta) \begin{bmatrix} \frac{x_{\mathrm{v}}}{-y_{\mathrm{v}}+\lambda} \\ \frac{y_{\mathrm{v}}}{x_{\mathrm{v}}+\lambda} \\ \frac{z_{\mathrm{v}}}{x_{\mathrm{v}}+\lambda} \end{bmatrix} + \begin{bmatrix} c_x \\ c_x \\ c_y \end{bmatrix}. \tag{5.26}$$

In order to define the orientation, we need to define 4 different points rigidly attached to the robot end-effector (Fig. 5.3 (b)). These 4 points can be used to represent the orientation of the end-effector in the base frame. The 4 points: $[P_{ef}, P_1, P_2, P_3]$ expressed in the end-effector frame O_{ef}, are the origin and the canonical basis of a 3D Euclidean space, which means

$$\begin{cases} \text{origin point } P_{ef}: & X_{ef} = [0,0,0]^T \\ \text{x axis } P_1: & X_{e_1} = [1,0,0]^T \\ \text{y axis } P_2: & X_{e_2} = [0,1,0]^T \\ \text{z axis } P_3: & X_{e_3} = [0,0,1]^T. \end{cases} \tag{5.27}$$

5.3.1 Orientation Definition in Virtual Visual Space

In order to specify 3 orthogonal vectors in the *virtual visual space*, which can be used to represent visual orientation, the 4 points defined in (5.27) can be represented in frame O_{v} and O_s using (5.26)

$$\begin{cases} X_{\mathrm{v}_{ef}} = [0,0,0]^T \\ X_{\mathrm{v}_1} = [1,0,0]^T \\ X_{\mathrm{v}_2} = [0,1,0]^T \\ X_{\mathrm{v}_3} = [0,0,1]^T \end{cases} \Longrightarrow \begin{cases} X_{s_{ef}} = [c_x, c_x, c_y]^T \\ X_{s_1} = [c_x + \frac{f\beta}{\lambda}, c_x, c_y]^T \\ X_{s_2} = [c_x, c_x + \frac{f\beta}{\lambda}, c_y]^T \\ X_{s_3} = [c_x, c_x, c_y + \frac{f\beta}{\lambda}]^T. \end{cases} \tag{5.28}$$

In the same form, using (5.28), we define a basis of the 3D virtual visual space (expressed in pixels)

$$\begin{cases} V_1 = X_{s1} - X_{s_{ef}} = [\frac{f\beta}{\lambda}, \quad 0, \quad 0]^T \\ V_2 = X_{s2} - X_{s_{ef}} = [0, \quad \frac{f\beta}{\lambda}, \quad 0]^T \\ V_3 = X_{s3} - X_{s_{ef}} = [0, \quad 0, \quad \frac{f\beta}{\lambda}]^T. \end{cases} \tag{5.29}$$

According to the definition of the rotation matrix, the orientation of this 3D virtual visual space is represented as

$$R_s^{\mathrm{v}} = \begin{bmatrix} \frac{V_1}{\|V_1\|} & \frac{V_2}{\|V_2\|} & \frac{V_3}{\|V_3\|} \end{bmatrix} = I. \tag{5.30}$$

Hence, the visual rotation matrix of the end-effector with respect to O_s can be computed as

$$R_s^{ef} = R_s^{\mathrm{v}} R_{\mathrm{v}}^{ef} = R_{\mathrm{v}}^{ef} \tag{5.31}$$

$$= (R_{\mathrm{v}}^{C_l} R_{C_l}^{b}) R_b^{ef}. \tag{5.32}$$

5.3.2 Orientation Mapping J_ω

Given a rotation matrix R, the angular velocity of the rotating frame can be represented as

$$S(\omega) = \dot{R} R^T \tag{5.33}$$

in which $S(\omega)$ is a skew symmetric matrix, defined as

$$S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{5.34}$$

and $\omega = [\omega_x, \omega_y, \omega_z]^T$ is the angular velocity.

Therefore, the angular velocity of the end-effector frame with respect to the robot base frame (ω_b) is given by

$$S(\omega_b) = \dot{R}_b^{ef} (R_b^{ef})^T \tag{5.35}$$

and using (5.32) the angular velocity (ω_s) of the end-effector frame with respect to O_s is given by[1]

$$S(\omega_s) = \dot{R_s^{ef}} (R_s^{ef})^T \tag{5.36}$$

$$= (R_{\mathrm{v}}^{C_l} R_{C_l}^{b}) \dot{R_b^{ef}} (R_b^{ef})^T (R_{\mathrm{v}}^{C_l} R_{C_l}^{b})^T \tag{5.37}$$

$$= (R_{\mathrm{v}}^{C_l} R_{C_l}^{b}) S(\omega_b) (R_{\mathrm{v}}^{C_l} R_{C_l}^{b})^T \tag{5.38}$$

$$= S((R_{\mathrm{v}}^{C_l} R_{C_l}^{b}) \omega_b). \tag{5.39}$$

From (5.39) we can obtain the visual angular velocity

$$\omega_s = (R_{\mathrm{v}}^{C_l} R_{C_l}^{b}) \omega_b. \tag{5.40}$$

Now, let $\theta = [\alpha, \beta, \gamma]^T$ be a vector of Euler angles, which denotes a minimal representation for the orientation of the end-effector frame relative to the robot base frame. Then, the definition

[1] Properties of Skew Symmetric matrices show that: $RS(\alpha)R^T = S(R\alpha)$ with $R \in SO(3)$ and $\alpha \in \mathbb{R}^3$.

of the angular velocity ω is given by [155]

$$\omega = T(\theta)\dot{\theta}. \tag{5.41}$$

If $R_{ef} = R_{z,\gamma}R_{y,\beta}R_{x,\alpha}$ is the Euler angle transformation, then

$$T(\theta) = \begin{bmatrix} \cos(\gamma)\cos(\beta) & -\sin(\gamma) & 0 \\ \sin(\gamma)\cos(\beta) & \cos(\gamma) & 0 \\ -sin(\beta) & 0 & 1 \end{bmatrix}. \tag{5.42}$$

Singularities of the matrix $T(\theta)$ are called *representational singularities*. It can easily be shown that $T(\theta)$ is invertible provided $\cos(\beta) \neq 0$.

Substituting (5.41) into (5.40) we obtain

$$T(\theta_s)\dot{\theta}_s = (R_{\mathrm{v}}^{C_l} R_{C_l}^{b})T(\theta_b)\dot{\theta}_b. \tag{5.43}$$

Furthermore, this expression can be written as

$$\dot{\theta}_s = \begin{bmatrix} \dot{\alpha_s} \\ \dot{\beta_s} \\ \dot{\gamma_s} \end{bmatrix} = \overbrace{T(\theta_s)^{-1}(R_{\mathrm{v}}^{C_l} R_{C_l}^{b})T(\theta_b)}^{J_\omega}\dot{\theta}_b \tag{5.44}$$

$$= J_\omega \dot{\theta}_b \tag{5.45}$$

where the matrix $J_\omega \in \mathbb{R}^{3\times 3}$ is defined as the *orientation image Jacobian*.

5.4 Visual Jacobian

In the previous sections, we have defined the mappings for the 3D pixel position and orientation separately as position image Jacobian (J_υ) and orientation image Jacobian (J_ω). Combining equation (5.25) and (5.45) we have the full expression

$$\dot{W}_s = \begin{bmatrix} \dot{X}_s \\ \dot{\theta}_s \end{bmatrix} = \begin{bmatrix} J_\upsilon & 0 \\ 0 & J_\omega \end{bmatrix} \begin{bmatrix} \dot{X}_b \\ \dot{\theta}_b \end{bmatrix} \tag{5.46}$$

$$= J_{\mathrm{img}}\dot{W}_b \tag{5.47}$$

where $J_{\mathrm{img}} \in \mathbb{R}^{6\times 6}$ is defined as the ***image Jacobian***, which is a block diagonal Jacobian matrix.

Substituting the robot *differential kinematics* $\dot{W}_b = J_a(q)\dot{q}$, equation (5.47) can be rewritten in the form

$$\dot{W}_s = J_{\mathrm{img}}J_a(q)\dot{q} = J_s\dot{q} \tag{5.48}$$

where $J_a(q) \in \mathbb{R}^{6\times6}$ is the analytical Jacobian matrix of the robot manipulator and $J_s \in \mathbb{R}^{6\times6}$ is defined as the ***Visual Jacobian***.

The analytical Jacobian, $J_a(q)$ can be computed from the geometric Jacobian $J(q)$ as

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & T(\theta_b)^{-1} \end{bmatrix} J(q) \tag{5.49}$$

provided that $det(T(\theta)) \neq 0$.

Then the inverse differential kinematics that relates generalized joint velocities $\dot{q}$ and 6D visual velocities $\dot{W}_s$ is given by

$$\dot{q} = {J_s}^{-1}\dot{W}_s = {J_a(q)}^{-1}{J_{\text{img}}}^{-1}\dot{W}_s. \tag{5.50}$$

Equation (5.50) is the inputs of the controller, which is used to design a regressor-based adaptive control law in the next section.

Some remarks about the proposed visual Jacobian and visual servoing are presented here:

Remark: ***Image-based Visual Servoing.***
As shown in the Fig. 5.2, the general idea behind our method is that we try to extract six independent features W_s (pixel) from the classical image features s in order to get a square full-rank image Jacobian J_{img}. The mapping from s to our new features W_s can be obtained by many different methods. In this thesis, we provide one possible solution by recovering 3D positions from stereo cameras and re-projecting them into two virtual camera frames. Both these steps are intermediate steps to calculate the required mapping.

Remark: ***Singularity-free*** J_{img}**.**
From (5.46), we can see that $det(J_{\text{img}}) = det(J_\upsilon)det(J_\omega)$, therefore the set of singular configurations of J_{img} is the union of the set of position configurations satisfying $det(J_\upsilon) = 0$ and the set of orientation configurations satisfying $det(J_\omega) = 0$.

From (5.21) and (5.24), we can see that ${J_\upsilon}^{-1} = {R^b_{C_l}}^{-1}{R^{C_l}_{\text{v}}}^{-1}{J_o}^{-1}R_\alpha^{-1}$. The matrices $R^b_{C_l}, R^{C_l}_{\text{v}} \in SO(3)$ and $R_\alpha = diag(f\beta) \in \mathbb{R}^{3\times3}$ are non-singular. Then, $det(J_o) = 0 \rightarrow det(J_v) = 0$. This condition is present only when: 1) $O_{11} + \lambda = 0$ and $O_{21} - \lambda = 0$ or 2) $x_{\text{v}} = -\lambda$ and $y_{\text{v}} = O_{21}$ or 3) $y_{\text{v}} = \lambda$ and $x_{\text{v}} = O_{11}$. However, O_{11}, O_{21} and λ are also defined by the user. Then, a non-singular J_v can be obtained using the condition $O_{11} = O_{21}, \lambda > \max(x_{\text{v}_{\max}}, y_{\text{v}_{\max}})$, where $x_{\text{v}_{\max}}$ and $y_{\text{v}_{\max}}$ are delimited by the robot workspace defined with respect to O_{v}. Hence, $det(Jv) \neq 0$ and can not become infinite.

In (5.44), singularities of the matrix $T(\theta)$ are called representational singularities, which means when these happen, the orientation can not be presented as Euler angles. In our case,

Euler angles always exist by constraining those special points. By providing $det(T(\theta)) \neq 0$, J_ω^{-1} always exists. Therefore, the singularities of J_s are defined only by the singularities of $J(q)$.

Remark: *Sensitivity to Camera Orientation* $R_{C_l}^b$.
The orientation matrix $R_{C_l}^b$ requires a special attention because it can cause system instability. Instead of requiring an exact off-line calibration of this parameter, the problem is tackled in two parts: a) A coarse on-line estimation of the orientation matrix is computed using the real-time information generated by the robot (subsec 5.4.1) and b) Estimation errors for the complete Jacobian J_s are taken into account in the control design. Thus, a robust control approach is used to cope with these errors, including estimated error in $R_{C_l}^b$ and the stereo-rig intrinsic parameter uncertainties (subsection 5.5.3).

5.4.1 On-line Orientation Matrix Estimation

The stereo system is composed of two USB cameras fixed on a tripod. The stereo-rig is assumed to be known, which means that the intrinsic parameters and the transformation between two cameras are known and constant. These can be calculated once off-line. The whole stereo camera setup can be moved together during the visual tasks.

We define this system as uncalibrated because we not only assume that the calibration of the stereo cameras system (left camera O_{C_l}) with respect to the wcf (O_b) is unknown, but we also consider the possibility of *on-line modification* of the parameters that define this relationship (e.g. $R_{C_l}^b$). Nevertheless, an exact calibration of the stereo system is also not required because errors in the estimation of these visual parameters are handled in the control law (sec 5.5.3).

In order to compute the image Jacobian J_{img} in (5.46), we use an on-line orientation estimator for $R_{C_l}^b$, where two sets of position points defined in coordinate frames O_b and O_{C_l} are used. These sets are generated while the robot is moving. The estimation approach uses *Singular Value Decomposition* (SVD) [158]. More details about the on-line orientation estimation has been presented in section 4.3.3. This rotation matrix is prone to errors of estimation which are considered in the next section.

5.5 6D Visual Servoing

In this section, we describe the design of an adaptive image-based dynamic control. Here we use an adaptive controller since it allows to deal with some uncertainties from robot model and camera calibration. The proposed second order sliding mode control is chattering free. This result is obtained using the $\int sign\,(*)$ function instead of directly using the $sign\,(*)$ function.

Furthermore, the $sign(*)$ function is also replaced for $tanh(\mu*)$ to avoid discontinuous signals for the nominal reference, similar to [161]. This control method includes the robot dynamics model in its passivity proof.

5.5.1 Non Linear Robot Dynamic Model

The dynamics of a serial n-link rigid, fully actuated robot manipulator can be written as follows

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} = \tau. \tag{5.51}$$

where $q \in \mathbb{R}^{n\times 1}$ is the vector of joint positions, $\tau \in \mathbb{R}^{n\times 1}$ stands for the applied joint torques, $M(q) \in \mathbb{R}^{n\times n}$ is the symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q} \in \mathbb{R}^{n\times n}$ is the vector of centripetal and Coriolis effects, $G(q) \in \mathbb{R}^{n\times 1}$ is the vector of gravitational torques, and finally $B \in \mathbb{R}^{n\times n}$ is a diagonal matrix for the viscous frictions.

The robot model described in (5.51) can be written in terms of a known state robot regressor $Y_r = Y(q,\dot{q},\ddot{q}) \in \mathbb{R}^{n\times m}$ and an unknown robot parameter vector $\Theta \in \mathbb{R}^{m\times 1}$ by using nominal references $\dot{q_r}$ and $\ddot{q_r}$ as follows:

$$M(q)\ddot{q_r} + C(q,\dot{q})\dot{q_r} + G(q) + B\dot{q_r} = Y_r\Theta \tag{5.52}$$

Subtracting the linear parameterization equation (5.52) to (5.51), produces the open-loop error dynamics

$$M(q)\dot{S}_q + C(q,\dot{q})S_q = \tau - Y_r\Theta \tag{5.53}$$

with the *joint error surface* S_q defined as $S_q = \dot{q} - \dot{q}_r$, where $\dot{q}_r$ represents the nominal reference of joint velocities. This nominal reference can be used to design a control in the virtual visual space.

5.5.2 Joint Velocity Nominal Reference

Considering equation (5.50), $\dot{q_r}$ can be defined as

$$\dot{q_r} = {J_s}^{-1}\dot{W}_{s_r} \tag{5.54}$$

where, the 6D visual nominal reference $\dot{W}_{s_r}$ is given by

$$\dot{W}_{s_r} = \dot{W}_{s_d} - K_p\Delta W_s + S_{s_d} - K_1\int_{t_0}^{t} S_{s_\delta}(\zeta)\,d\zeta - K_2\int_{t_0}^{t} sign\left(S_{s_\delta}(\zeta)\right)d\zeta \tag{5.55}$$

$$S_{s_\delta} = S_s - S_{s_d}, \tag{5.56}$$

$$S_s = \left(\Delta\dot{W}_s + K_p\Delta W_s\right), \tag{5.57}$$

$$S_{s_d} = S_s(t_0)\,e^{-\kappa t} \tag{5.58}$$

where $\dot{W}_{s_d}$ is the desired visual velocity, $\Delta W_s = W_s - W_{s_d}$ is the visual position error, $\Delta\dot{W}_s$ is the visual velocity error, $K_p = {K_p}^T \in \mathbb{R}_+^{6\times6}$ and $K_j = {K_j}^T \in \mathbb{R}_+^{6\times6}$ (with $j = 1, 2$) and $S_{s\delta}$ is the *virtual visual error surface*.

Using (5.54-5.58) in S_q we obtain:

$$S_q = \dot{q} - \dot{q}_r = {J_s}^{-1}\left(\dot{W}_s - \dot{W}_{s_r}\right) = {J_s}^{-1} S_e \tag{5.59}$$

with

$$S_e = S_{s_\delta} + K_1 \int_{t_0}^{t} S_{s_\delta}(\zeta)\, d\zeta + K_2 \int_{t_0}^{t} sign\left(S_{s_\delta}(\zeta)\right) d\zeta \tag{5.60}$$

where S_e is the *extended virtual visual error manifold*.

5.5.3 Uncertainties in J_s

The above definition of $\dot{q}_r$ depends on the exact calibration of J_s. However, this is a very restricted assumption. Hence, the uncertainties in the Visual Jacobian J_s should be taken into account in the control design. To achieve this, the uncalibrated nominal reference is defined by

$$\widehat{\dot{q}_r} = \widehat{J_s}^{-1}\dot{W}_{s_r} \tag{5.61}$$

where $\widehat{J_s}$ is an estimate of J_s such that $\widehat{J_s}$ is full-rank. $\forall q \in \Omega_q$ defines the singularity-free workspace, where $\Omega_q = \left\{q | \det\left(J(q)\right) \neq 0, \forall q \in \mathbb{R}^{n\times1}\right\}$. Then, the uncalibrated joint error surface is:

$$\hat{S}_q = \dot{q} - \widehat{\dot{q}_r} = S_q - \Delta J_s \dot{W}_{s_r} \tag{5.62}$$

with $\Delta J_s = \widehat{J_s}^{-1} - {J_s}^{-1}$ as the estimation errors, which includes both intrinsic and extrinsic real-camera parameters and kinematic robot parameters.

5.5.4 Adaptive Control Design

Consider a robot manipulator in closed loop with the following second order sliding visual servoing scheme,

$$\tau = -K_d \hat{S}_q + \hat{Y}_r \hat{\Theta} \tag{5.63}$$

$$\dot{\hat{\Theta}} = -\Gamma {\hat{Y}_r}^T \hat{S}_q \tag{5.64}$$

where $\hat{\Theta}$ is the on-line estimation of the constant robot parameter vector, $K_d = {K_d}^T \in \mathbb{R}_+^{n\times n}$ and $\Gamma \in \mathbb{R}_+^{m\times m}$ are constant matrices. This adaptive on-line estimation together with the second order sliding mode in S_{s_δ} handle the uncertainties on the robot dynamic/kinematic and camera parameters. The framework for integration the proposed visual features with this adaptive control is shown in Fig. 5.5.

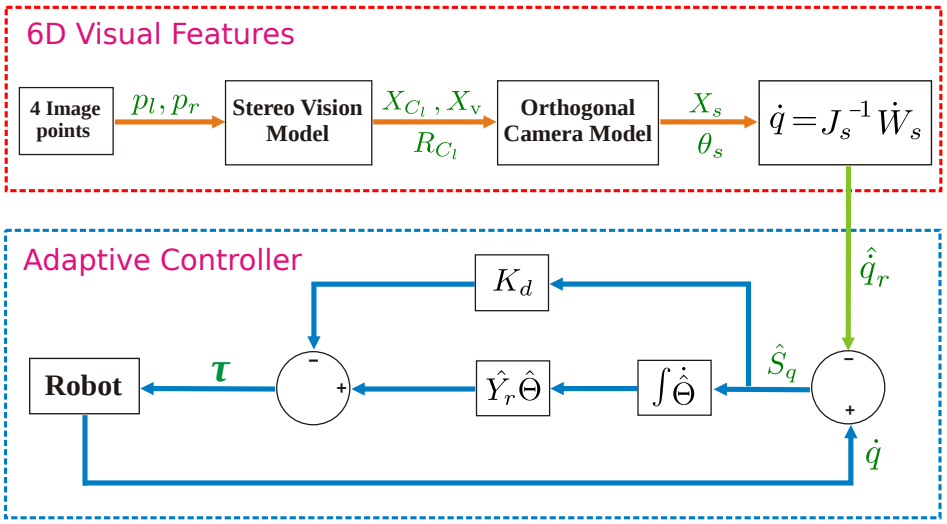


Figure 5.5: New 6D visual servoing framework.

5.5.5 Stability Proof

The stability proof is conducted in three parts:

1. Boundedness of the closed loop trajectories

The uncalibrated closed-loop error dynamics between (5.53) and (5.63-5.64) gives

$$M(q)\dot{\hat{S}}_q = \tau - \hat{Y}_r\Theta - C(q,\dot{q})\hat{S}_q \tag{5.65}$$

$$= -K_d\hat{S}_q + \hat{Y}_r\Delta\Theta - C(q,\dot{q})\hat{S}_q \tag{5.66}$$

with $\Delta\Theta = \hat{\Theta} - \Theta$.

The uncalibrated error kinematic energy can be used as a Lyapunov function in the following form as:

$$V = \frac{1}{2}\left[\hat{S}_q{}^T M(q)\hat{S}_q + \Delta\Theta^T\Gamma^{-1}\Delta\Theta\right]. \tag{5.67}$$

Considering the time derivative of (5.67) in closed loop with (5.63-5.66), $\dot{V}$ yields

$$\dot{V} = \hat{S}_q{}^T M\dot{\hat{S}}_q + \frac{1}{2}\hat{S}_q{}^T\dot{M}\hat{S}_q + \Delta\Theta^T\Gamma^{-1}\dot{\hat{\Theta}} \tag{5.68}$$

$$= -\hat{S}_q{}^T K_d\hat{S}_q + \hat{S}_q{}^T\hat{Y}_r\Delta\Theta - \Delta\Theta^T\hat{Y}_r{}^T\hat{S}_q \tag{5.69}$$

$$= -K_d\left\|\hat{S}_q\right\| \leq 0 \tag{5.70}$$

where, the property (5.52) in terms of $\hat{S}_q$ has been used.

Selecting a positive K_d, equation (5.70) becomes negative semidefinite and this proves the passivity of the robot dynamics (5.51) in closed loop with (5.63-5.64). Then, the following properties of the closed-loop state arises

$$\hat{S}_q \in L_\infty \rightarrow S_e \in L_\infty \Longrightarrow \left(S_{s_\delta}, \int_{t_0}^{t} sign\left(S_{s_\delta}\left(\zeta\right)\right) d\zeta \right) \in L_\infty \tag{5.71}$$

Which implies that all signal states are bounded, specially $(\dot{q}_r, \ddot{q}_r) \in L_\infty$ and $\left(\dot{W}_{s_r}, \ddot{W}_{s_r}\right) \in L_\infty$.

2. Second-order sliding modes

From (5.59) and (5.62) we obtain

$$\hat{S}_q = {J_s}^{-1} S_e - \Delta J_s \dot{W}_{s_r} \tag{5.72}$$

Using (5.60), (5.72) can be written as

$$J_s \left(\hat{S}_q + \Delta J_s \dot{W}_{s_r} \right) = S_e \tag{5.73}$$

$$S_e = S_{s_\delta} + K_1 \int_{t_0}^{t} S_{v_\delta}\left(\zeta\right) d\zeta + K_2 \int_{t_0}^{t} sign\left(S_{v_\delta}\left(\zeta\right)\right) d\zeta \tag{5.74}$$

then we generate an error function in terms of S_{s_δ} as

$$S_{s_\delta} = -K_1 \int_{t_0}^{t} S_{s_\delta}\left(\zeta\right) d\zeta - K_2 \int_{t_0}^{t} sign\left(S_{s_\delta}\left(\zeta\right)\right) d\zeta + S_e. \tag{5.75}$$

Taking the time derivative of (5.75) and multiplying it by $S_{s_\delta}^T$, we can prove the sliding mode regimen

$$S_{s_\delta}^T \dot{S}_{s_\delta} \leq -K_1 \left\| S_{s_\delta} \right\| - \mu \left| S_{s_\delta} \right| \tag{5.76}$$

with $\mu = K_2 - \left| \frac{d}{dt} S_e \right|$. If $K_2 \geq \left| \frac{d}{dt} S_e \right|$, then a sliding mode at $S_{v_\delta} = 0$ is induced at $t_s = \frac{\left| S_{s_\delta}(t_0) \right|}{\mu}$. Moreover, notice that for any initial condition $S_{s_\delta}\left(t_0\right) = 0$ then $t_s = 0$, which implies that the sliding mode is guaranteed for all time.

3. Exponential convergence of visual tracking errors

Since a sliding mode exists at all times at $S_{s_\delta}(t) = 0$, then $S_s = S_{s_d}$, therefore $\Delta \dot{W}_s = -K_p \Delta W_s + S_s\left(t_0\right) e^{-\kappa t}\ \forall t$, which implies that the 6D visual tracking errors converge to zero exponentially fast.

Remark: Convergence of ΔW_b without local minima.

Given that J_{img} is full-rank $\forall t$, from (5.47) can be seen that $\Delta W_s = 0 \rightarrow \Delta W_b = 0$ without local minima. This is the most important impact of designing a full-rank image Jacobian which, in general, is not obtained with the classical methods.

5.6 Simulation

In this algorithm,a torque level adaptive controller is evaluated in simulation to better illustrate the robustness of the system to uncertainties in the robot parameters and changes in the extrinsic parameters of the stereo system.

We simulate a 6DOF industrial robot with real robot dynamic parameters in closed loop with the control approach in (5.63-5.64). Real camera parameters are used to simulate the camera projections[2]. The Cartesian pose is projected into two virtual orthogonal cameras to get the new visual features as inputs for the visual servoing system. Our simulation platforms is the same as the real experiments, except we simulate the 6D visual desired pose. The robot motions are visualized in a 3D Visualization System (sub-sec 7.2.3).

In the simulation we first show the robustness of the system to uncertainties in the robot parameters and changes in the extrinsic parameters of the stereo system. Then, two simulation tasks are evaluated to show the performance of the proposed scheme.

5.6.1 Robustness to uncertainties

The task is defined as follows: the robot end-effector is commanded to draw a circle in the world coordinate frame using the Cartesian trajectory $X_{b_d} = [0.2\sin(\omega t), 0.2\cos(\omega t) - 0.8, 0.5]^T$, where $\omega = 10 rad/sec$.

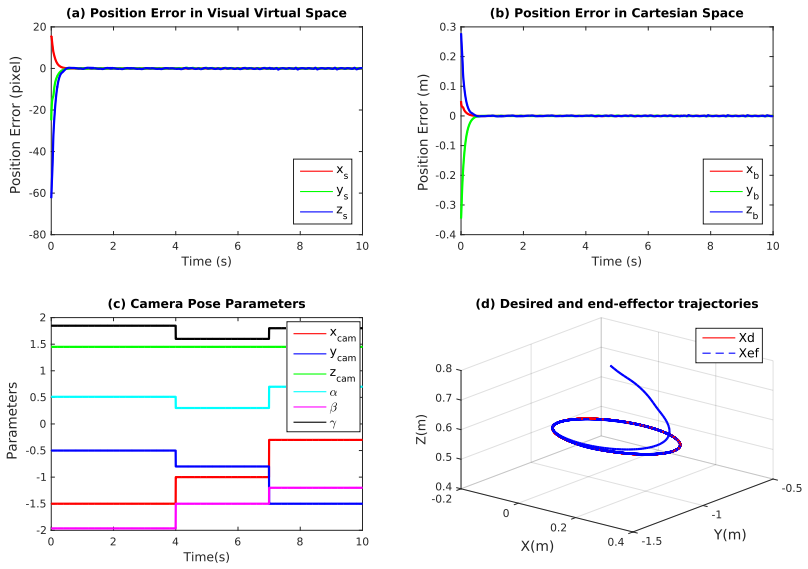


Figure 5.6: Simulation results: the robustness to uncertainties of the camera parameters.

Fig. 5.6 shows the results obtained from the simulation, where the 3D position tracking can be observed in both space. During the simulation, an estimate of robot parameters and a coarse-calibrated camera intrinsic parameters are used in the control law, and at time $t = 4s$, $t = 7s$, the extrinsic camera parameters are altered, see Fig. 5.6 (c). From the plots it can be observed that even when the parameters change, the controller is capable to cope with these uncertainties and maintain stability of the system. In the case of changes in the orientation matrix $R^b_{C_l}$, the controller can handle the uncertainty to a certain extent (approx. 20% error). Therefore, a suitable technique to generate a rough estimation of $R^b_{C_l}$ is needed to guarantee the stability (subsection 5.4.1).

5.6.2 Regulation

The robot end-effector is commanded to a desired pose as $W_{b_d} = [0.0, -0.7, 0.5, 3.0, pi/4, 2.2]^T$. We map this desired Cartesian pose to the desired 6D pixel pose W_s, which is used to design the error function for the control scheme.

Simulation results are depicted in Fig. 5.7. Each figure consists of a set of plots shown in three columns: Column (a) shows the 3D visual position and Euler angles in the virtual visual space. It also includes their corresponding errors. The same results for the Task space (Cartesian Space) are illustrated in the second column (b). The 6D trajectories of the robot end-effector are depicted in both virtual visual space and Cartesian space.

The end-effector trajectories are close to a straight line in both spaces, see Fig. 5.7.(1). The position and rotation motions exhibit exponential convergence of the visual errors. In 5.7, elements (a).3 and (b).3 are the 3D position errors, while elements (a).4 and (b).4 illustrate the orientation errors. The third column (c) in the same figures demonstrates the virtual visual sliding surface and the error surface for the second order sliding mode control. The figures also illustrate that when the 6D pixel pose W_s convergence to the desired goal in virtual visual space, the end-effector pose also converges to the goal in Cartesian space without local minima.

5.6.3 Tracking

The robot end-effector is commanded to draw a circle and rotate the end-effector in the Task space using the trajectory $W_{b_d} = [0.1\sin(\omega t) + 0.6, 0.2\cos(\omega t), 0.5, 3.0, pi/4, 0.8\sin(\omega t) + 2.0]^T$, where $\omega = 0.05 rad/sec$.

Fig. 5.8 demonstrate the tracking results. The 3D trajectories are precise tracking in both spaces, while the orientation tracking has well behavior. Fig. 5.8.(3) and (4) show 6D pose

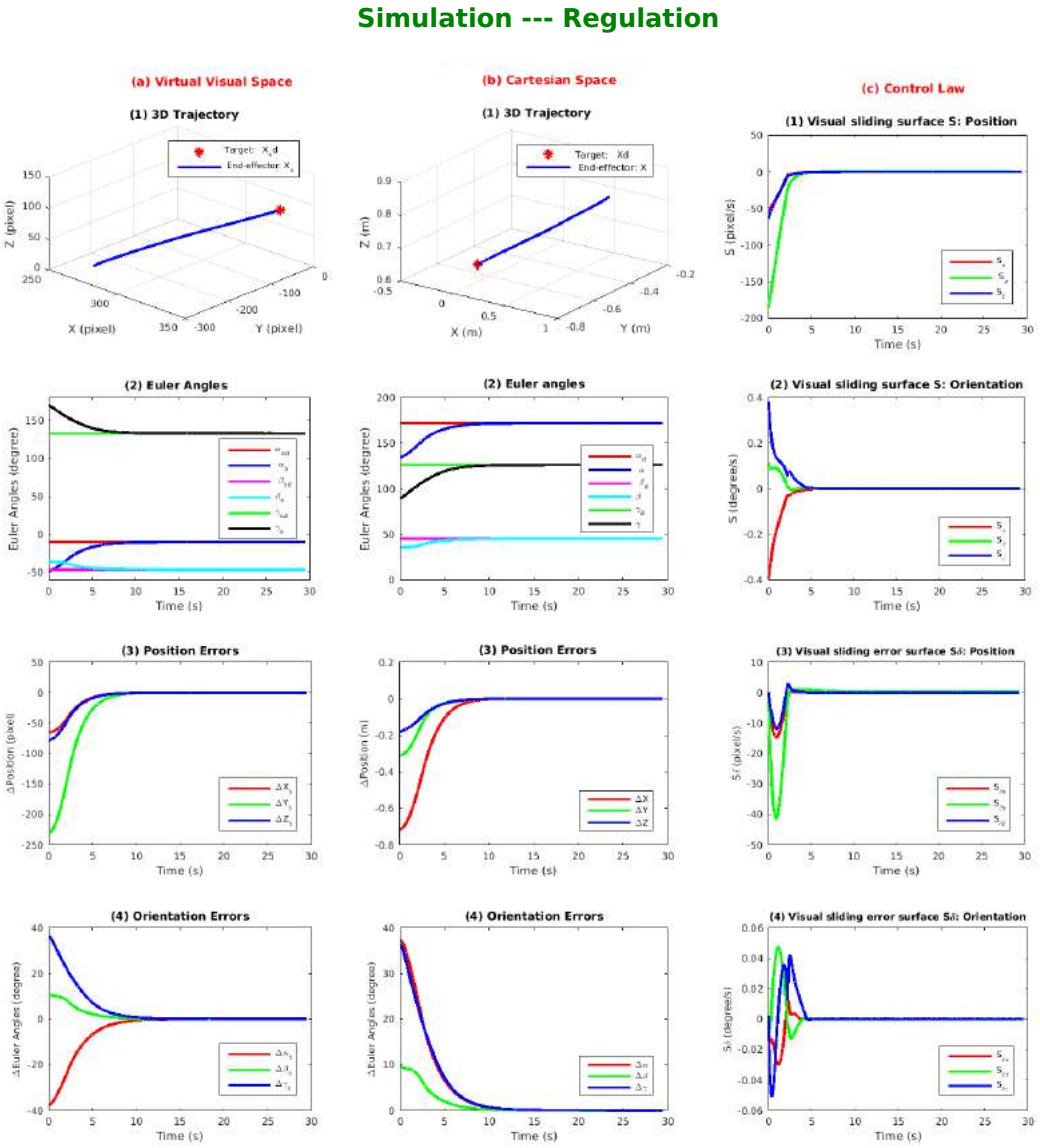


Figure 5.7: Simulation results for regulation in both Cartesian space and virtual visual space.

errors for both spaces. The results demonstrate the convergence of errors in both virtual visual space and Cartesian space without local minima. The plots in column (a) and (b) confirm that the mapping from the proposed virtual space to 3D Cartesian space is linear, and the control of rotational and translational motion of robot is decoupled due to the diagonal image Jacobian.

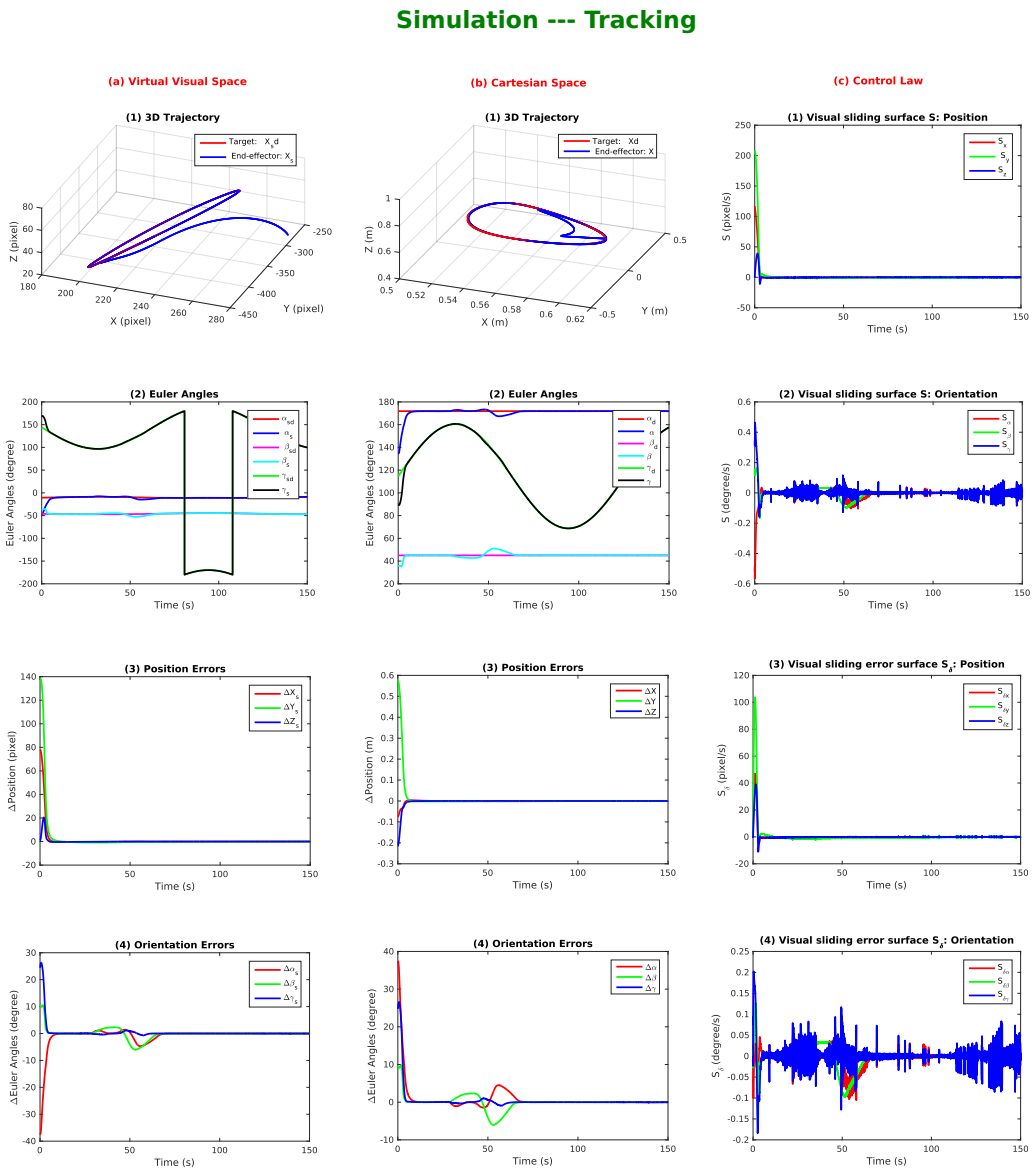


Figure 5.8: Simulation results for 6D tracking.

5.7 Discussion

In this chapter, we have investigated the control of translational and rotational motion for the end-effector of a robotic manipulator under visual feedback from fixed stereo cameras. An uncalibrated stereo vision system is used to compute the Cartesian position of feature points,

and then map these points to two virtual orthogonal cameras. The virtual cameras have a wide baseline, are orthogonal to each other and are aligned with the stereo camera frame axes. This results in a unique decoupling of the features in these two virtual images. Instead of using the classical visual features, we extract 6 orthogonal features as image features.

The key contributions of this chapter can be summarized into these aspects:

1. We have proposed a new *virtual visual space* (measured in pixels), where a 6D visual pose vector (6 orthogonal features) is defined and chosen as inputs instead of the classical visual features for IBVS method.

2. Using this 6D visual pose, we obtain a 6×6 full-rank *image Jacobian* that can avoid the well-known problems such as the image space singularities and local minima.

3. The proposed visual Jacobian is used to design an adaptive dynamic controller, which is evaluated by two simulation tests on an eye-to-hand robotic system. The results of the evaluation confirm the improvement in controller stability and motion performance. In order to evaluated the advantages of the proposed 6DVS scheme and show the novel properties of the new image features, the comparison of the behavior with the different visual servoing schemes is presented in next chapter 6.

4. Moreover, in Chapter 7 and 8, the proposed visual servoing is experimentally shown to be easy to integrate with the environment constraints such as robot singularities avoidance and (self-/obstacle-) collision avoidance in real world applications.

Chapter 6

Comparison with classical Visual Servoing Schemes

In this chapter, we compare the proposed 6D scheme (6DVS) presented in previous chapter with classical visual servoing approaches (IBVS, PBVS and HYVS). First, the selected control features for different schemes are reviewed, with which the controller is designed in closed loop with the control approach (5.63-5.64). Then five standard simulation tasks are tested to evaluate and compare the approaches in terms of steady state errors, transient systems performance, robustness to uncertainties and decoupling controlled signals. Simulation results prove that the new features in proposed 6DVS perform better than classical ones since the system combines the advantages of 2D and 3D visual servoing.

Simulations are based on a 6-DOF StaübliTX90 industrial manipulator and a target with four feature points. The real camera parameters and real robot dynamic parameters are used in the simulation.

6.1 Visual Features for different VS Approaches

As described in the visual servoing control tutorial [10], The aim of all vision-based control schemes is to minimize an error $e(t)$, which is typically defined by

$$e(t) = s - s_d, \tag{6.1}$$

and the design of the control scheme can be quite simple. Perhaps the most straightforward approach is to design a velocity controller, the relationship is given by

$$\dot{s} = L_s \, \mathrm{v}_c \tag{6.2}$$

in which s is a set of geometrical features whose time derivative $\dot{s}$ is linearly related to the spatial velocity $\mathrm{v}_c = [v_c, \omega_c]^T$ of the camera (eye-in-hand configuration[1]) through the interaction matrix L_s. Using this relationship, control schemes are designed to minimize the error e between the current value of the visual feature s and its desired value s_d.

Using (6.1) and (6.2), we immediately obtain the relationship between the robot velocity and the time variation of the error:

$$\dot{e} = L_e \, \mathrm{v}_c \tag{6.3}$$

where $L_e = L_s$. Considering v_c as the input to the robot controller, and we can solve it as

$$\mathrm{v}_c = L_e^+ \, \dot{e} \tag{6.4}$$

where $L_e^+ \in \mathbb{R}^{6\times 6}$ is chosen as the Moore-Penrose pseudo-inverse of L_e.

Visual servoing schemes mainly differ in the way that the visual feature vector s is designed, which decide the formate of control Jacobian L_e and the performance of the system. In order to compare different Visual Servoing schemes, in this section, we review three classical approaches and our proposed scheme regarding different selected s and the control Jacobian.

6.1.1 Image-based Visual Servoing

Suppose that the robot end-effector is moving with angular velocity $\omega_c = [\omega_x, \omega_y, \omega_z]^T$ and translational velocity $v_c = [v_x, v_y, v_z]^T$ both with respect to the camera frame O_c in a fixed camera system (eye-to-hand configuration). Consider the motion of a plane π attached to the end effector of a robot that rotates and translates through space in order to obtain a desired position and orientation of the end-effector. We define four target points on π denoted by P_i, $\forall i = 1, 2, 3, 4$.

Traditional *image-based* control schemes (IBVS), use the image plane coordinates of a set of points to define the set s. By considering a 3D point with coordinates $P = [X, Y, Z]^T$ in the camera frame and using a perspective projection model, the point X is projected on a 2D point x of coordinates (x, y) on image plane such that:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} = \begin{bmatrix} (u - c_x)/f\alpha \\ (v - c_y)/f \end{bmatrix} \tag{6.5}$$

where $s = [u, v]$ is the image point coordinates in pixel unit, $c = [c_x, c_y]$ is the coordinates of the principle point, f is the focal length of the camera lens and α is the ratio of pixel dimension.

[1] In our system, we use eye-to-hand configuration, hence v_c is the velocity of the robot end-effector with respect to camera frame O_c.

For 4 image points, the visual feature vector is

$$s = m = [x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4]^T \tag{6.6}$$

$$\dot{m} = \mathbf{L_x} \cdot \mathrm{v}_c \tag{6.7}$$

and the interaction matrix $\mathbf{L_x} = \begin{bmatrix} L_{\mathrm{x}_1} & L_{\mathrm{x}_2} & L_{\mathrm{x}_3} & L_{\mathrm{x}_4} \end{bmatrix}^T$, and L_{x_i} defined in (2.16) is

$$L_{\mathrm{x}_i} = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} & -xy & (1+x^2) & -y \\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -(1+y^2) & xy & x \end{bmatrix}. \tag{6.8}$$

The image-based approach may reduce interpretation and eliminate errors due to sensor modeling and camera calibration. However it does present a significant challenge for controller design since the features are nonlinear and highly coupled.

6.1.2 Position-based Visual Servoing

In *position-based* servoing (PBVS), image features are extracted as well, but a geometric model of the target and used known camera model are additionally used to estimate 3D information (pose of the target in Cartesian space). Feedback is computed by reducing errors of the estimated pose in Cartesian space. With a ZYX Euler angles representation for rotation matrix, denoted by $\theta = [\alpha, \beta, \gamma]^T$, the selected feature vector is designed as

$$s = W_b = [X, Y, Z, \alpha, \beta, \gamma]^T \tag{6.9}$$

$$\dot{W}_b = J_p \cdot \mathrm{v}_c \tag{6.10}$$

and the image Jacobian is a block diagonal matrix.

$$J_p = \begin{bmatrix} I_3 & 0 \\ 0 & J_\omega \end{bmatrix}, \qquad \begin{bmatrix} \dot{P} \\ \dot{\theta} \end{bmatrix} = J_p \begin{bmatrix} v_c \\ \omega_c \end{bmatrix}. \tag{6.11}$$

The definition of the angular velocity ω is given by [155]

$$\omega = T(\theta)\dot{\theta} \tag{6.12}$$

where $T(\theta)$ is defined in (5.42). Hence,

$$J_\omega = T^{-1}(\theta). \tag{6.13}$$

6.1.3 Hybrid Visual Servoing

The 2-1/2D visual servoing (HYVS) is one of the advanced *hybrid* methods proposed so far to combine advantages of IBVS and PBVS [11]. The 2-1/2D VS is based on decoupling the

rotational motions from the translational motions through selecting visual features defined in part in 3D Cartesian space and in part in 2D image space. Thus, the rotation is controlled directly in the Cartesian space, while its translation is controlled using image-based information.

In classical 2-1/2-D visual servoing, as described by Malis et al.[132], the selected feature vector is $h = [X_h, \Theta U^T]^T$, where X_h is the position vector, Θ and U are the rotation angle and axis of the rotation matrix R. Consider a point $P = [X, Y, Z]^T$ (called the reference point) lying on the object. If the rotation is in ZYX Euler angles representation as $\theta = [\alpha, \beta, \gamma]^T$, then the control feature vector is

$$s = h = [X_h, \theta]^T = [x, y, log(Z), \alpha, \beta, \gamma]^T \tag{6.14}$$

$$\dot{h} = J_h \cdot \mathrm{v}_c \tag{6.15}$$

The position vector $X_h = (x, y, log(Z))^T$ is defined in extended image coordinates, where x, y are directly computed from image features (2D data), and Z is the depth of the considered point (3D data).

The corresponding image Jacobian is an upper block triangular matrix, which provides interesting decoupling properties. It is represented as

$$J_h = \begin{bmatrix} J_v & J_{v\omega} \\ 0_3 & J_\omega \end{bmatrix}, \qquad \begin{bmatrix} \dot{X}_h \\ \dot{\theta} \end{bmatrix} = J_h \begin{bmatrix} v_c \\ \omega_c \end{bmatrix}. \tag{6.16}$$

Given $\dot{Z}$ in equation (2.10), the time derivative of $log(Z)$ can be represented as

$$(log(Z))' = \frac{1}{Z} \cdot \dot{Z} = [0, 0, \frac{1}{Z}, \frac{Y}{Z}, -\frac{X}{Z}, 0] \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} \tag{6.17}$$

$$= [0, 0, \frac{1}{Z}, y, -x, 0] \begin{bmatrix} v_c \\ \omega_c \end{bmatrix}. \tag{6.18}$$

Combining equation (6.8) and (6.18), we can get the Jacobian matrices

$$J_v = \frac{1}{Z} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.19}$$

and

$$J_{vw} = \begin{bmatrix} -xy & (1+x^2) & -y \\ -(1+y^2) & xy & x \\ y & -x & 0 \end{bmatrix}. \tag{6.20}$$

Since the rotational motion is directly controlled in the Cartesian space, the orientation Jacobian J_w is the same as defined in PBVS, (6.13).

6.1.4 Proposed Visual Servoing

For our proposed visual servoing (6DVS), we define a new virtual visual space (image space), where a 3D pixel position $X_s = [x_s, y_s, z_s]^T$ is extracted as a feature. All elements of this 3D position vector are linearly independent and orthogonal to each other, see equation (5.20). In order to compared with 2-1/2D VS, we simplify the orientation using a rotation matrix with ZYX Euler angles representation in Cartesian space with respect to the robot base frame, denoted by θ, which is the same as presented in PBVS and 2-1/2D VS. Thus, the new feature vector is

$$W_s = [x_s, y_s, z_s, \alpha, \beta, \gamma]^T \tag{6.21}$$

and the new mapping is given by

$$\dot{W}_s = \begin{bmatrix} \dot{X}_s \\ \dot{\theta} \end{bmatrix} = \overbrace{\begin{bmatrix} J_\upsilon & 0_3 \\ 0_3 & J_\omega \end{bmatrix}}^{J_{\text{img}}} \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} \tag{6.22}$$

where the new image Jacobian ($J_{\text{img}} \in \mathbb{R}^{6\times6}$) is a decoupling diagonal matrix that decouples the translational and rotational control.

The position image Jacobian is defined in (5.21) as

$$J_\upsilon = J_\alpha \, R_{\text{v}}^{C_l} \tag{6.23}$$

and $J_\omega = T^{-1}(\theta)$ is the same as presented in PBVS, (6.13).

The selected visual features and designed controller inputs of four visual servoing approaches are summarized in Table 6.1.

Table 6.1: Visual features and controller inputs for 4 visual servoing methods.

	Selected Visual Features s	**Controller Inputs**
IBVS	$m = [x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4]^T$	$\text{v}_c = \mathbf{L_x}^+ \dot{m}$
PBVS	$W_b = [X, Y, Z, \alpha, \beta, \gamma]^T$	$\text{v}_c = {J_p}^+ \dot{W}_b$
HYVS	$h = [x, y, log(Z), \alpha, \beta, \gamma]^T$	$\text{v}_c = {J_h}^+ \dot{h}$
6DVS	$W_s = [x_s, y_s, z_s, \alpha, \beta, \gamma]^T$	$\text{v}_c = {J_{\text{img}}}^{-1} \dot{W}_s$

6.2 Comparison of 6DVS with classical Methods

Simulations have been carried out in five different tests in terms of system performance, robustness to uncertainties and motion decoupling. Four corner points are extracted from image plane (Table 6.2), which can give us a 16×6 interaction matrix in the classical IBVS with stereo vision system and a 6×6 image Jacobian in our algorithm. In the proposed method 6DVS, the image measurements (s) are mapped to the virtual visual space to get new visual features (W_s), which is used to design the error function for the control scheme. For classical stereo IBVS, the image features (s) are directly used to design the error function and the real depth (z) obtained from the stereo vision system is used to compute the interaction matrix. The 3D Cartesian position from the stereo vision system is used to perform PBVS. Classical 2-1/2D (HYVS) with Euler angle representation is also used in comparison. The control law used for all approaches is the same.

The initial and desired pixel positions of the image features ($s = [u_1, v_1, u_2, v_2, u_3, v_3, u_4, v_4]^T$) from 4 corner points for each test are given in Table 6.2.

Table 6.2: Initial(I) and Desired(D) location of feature points on image plane (pixel) for left camera

		Point 1		Point 2		Point 3		Point 4	
		(u	v)	(u	v)	(u	v)	(u	v)
Test 1	**I**	(207	254)	(194	212)	(213	200)	(225	238)
	D	(422	379)	(406	307)	(471	290)	(486	360)
Test 2	**I**	(195	177)	(236	153)	(260	195)	(219	219)
	D	(219	219)	(195	177)	(236	153)	(260	195)
Test 3	**I**	(318	224)	(304	191)	(312	177)	(325	208)
	D	(226	228)	(200	184)	(244	159)	(269	203)
Test 4	**I**	(207	254)	(194	212)	(213	200)	(225	238)
	D	(422	379)	(406	307)	(471	290)	(486	360)
Test 5	**I**	(207	254)	(194	212)	(213	200)	(225	238)
	D	(291	444)	(304	382)	(367	376)	(354	434)

Remarks:

- **In simulation figures, only image features in the left camera are illustrated, since the results in the right camera are similar.**

For all tests, consider the motion of a plane π attached to the robot end-effector and target, with four target points on π denoted by P_i , $\forall i = 1, 2, 3, 4$. Then we obtain the translation motion using triangulation on the center of the four points while utilizing the rotational information of the end-effector through the motion of four tracked points.

6.2.1 Test 1: Large translational and rotational Motion

In the first test, we examine the convergence of each image feature point and pose errors when the desired location is far away from the initial one. The Cartesian and image trajectories of the proposed 6D visual servoing (6DVS) and the conventional approaches (IBVS, PBVS and HYVS) are compared in Fig. 6.1.

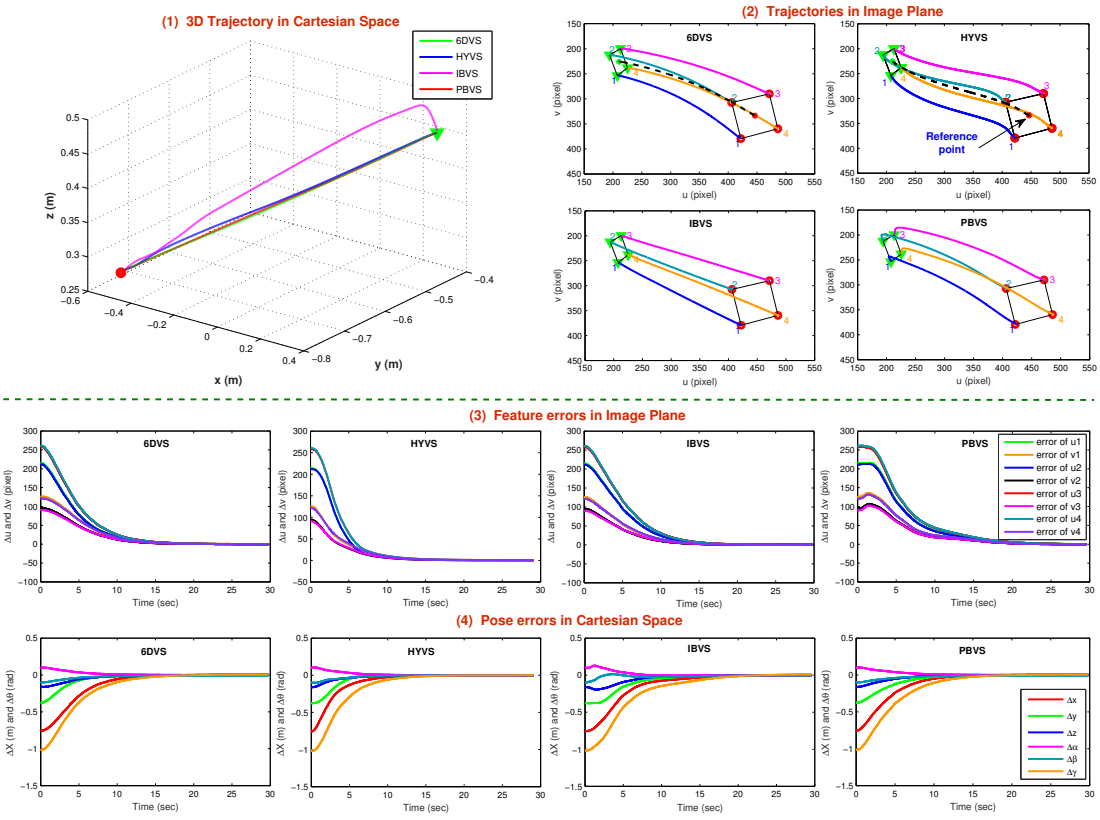


Figure 6.1: Simulation Test 1: Large translational and rotational motion.

As shown in Fig. 6.1 (1), PBVS results in a straight-line end-effector trajectory in Cartesian space. Since there is no control of the image features, the image trajectory, (as shown in Fig. 6.1 (2)), is unpredictable and may leave the camera field of view. In IBVS, straight-line image trajectory is observed while the end-effector Cartesian trajectory is not controlled.

For HYVS, the image trajectory of the reference point is a straight-line as expected, and the Cartesian trajectory is also well behaved. However, other points in image plane have curved trajectories.

Contrary to the previous approaches, the proposed 6DVS has a straight-line Cartesian trajectory (similar to that of PBVS) and all the image features are "indirectly" controlled to move approximately along straight-line trajectories like IBVS, see Fig. 6.1 (1), (2). Moreover, both the features errors and the Cartesian pose errors converge to zero very smoothly without any overshooting in 6DVS (Fig. 6.1 (3), (4)). Because of the selected features which are controlled, the Cartesian pose errors converge to zero very smoothly in PBVS while the image features errors are smooth in IBVS. Although HYVS has a similar trade-off between these properties, the proposed 6DVS is more efficient and has better performance than HYVS. Hence, 6DVS combines the advantages of PBVS in terms of controlling straight trajectories in Cartesian Space, and the advantages of IBVS in terms of controlling image trajectories.

6.2.2 Test 2: Motion around the Camera Optical Axis

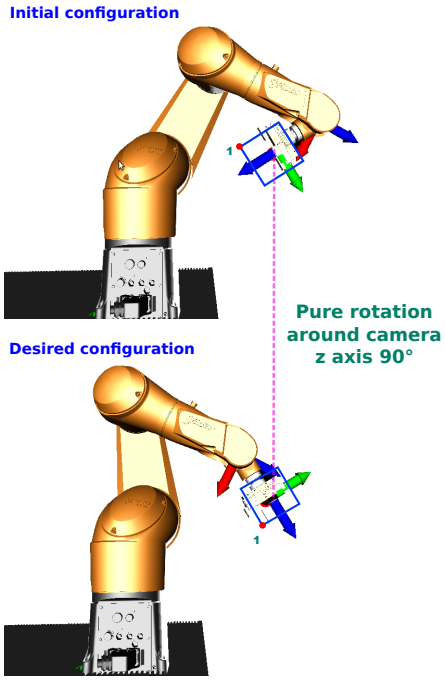


Figure 6.2: A pure rotation of features around the camera optical axis z_c by 90 degrees.

For this case we perform a pure rotation of features around the camera optical axis z_c by 90 degrees, see Fig. 6.2. The comparison results are shown in Fig. 6.3.

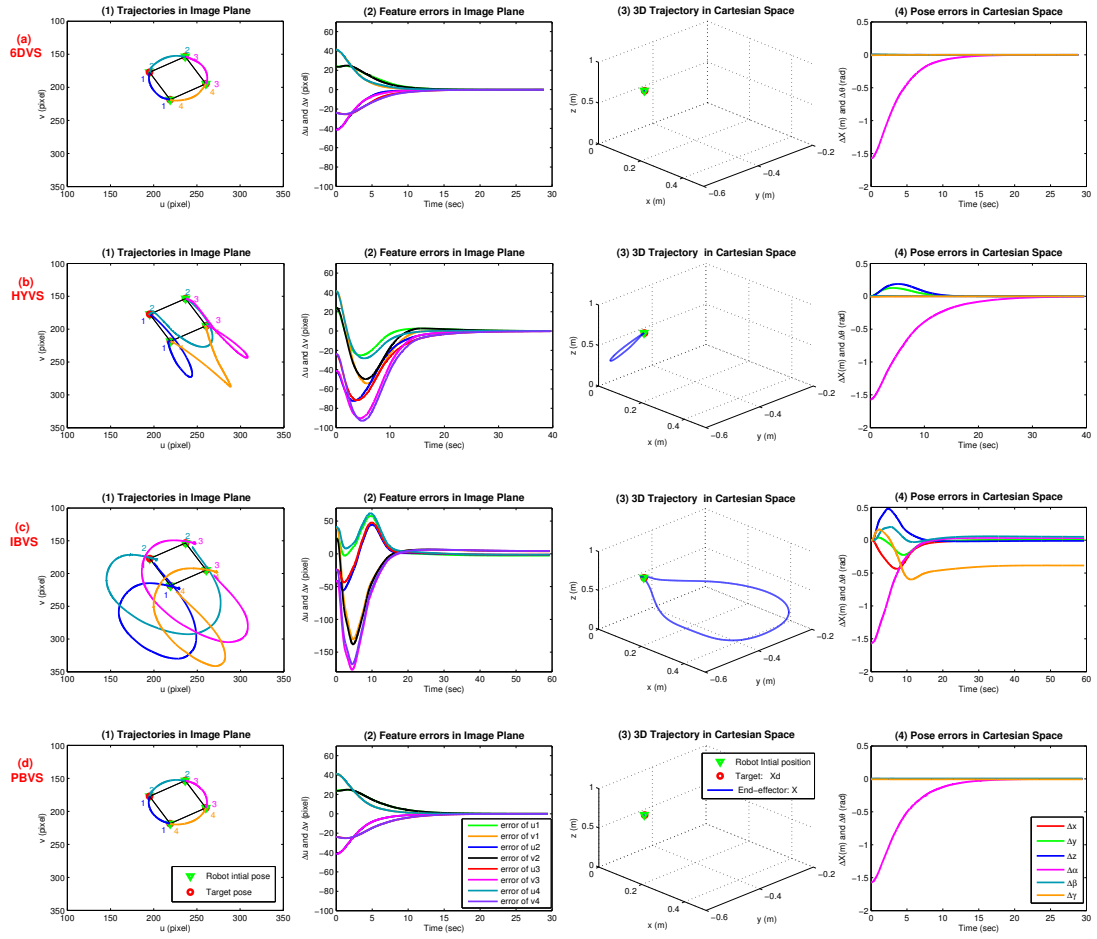


Figure 6.3: Simulation Test 2: Pure rotational motion.

The pure rotational motion is performed successfully in this test for the proposed 6DVS, IBVS, PBVS and HYVS. Fig. 6.3 (a) shows the performance of the proposed 6DVS algorithm in image plane and 3D Cartesian space, in terms of trajectories and errors. All the trajectories in difference spaces are nice and smooth. Fig. 6.3 (b), (c) and (d) illustrate the same movement using the conventional IBVS, PBVS and HYVS. From plots (1) and (2), we can see the proposed 6DVS and PBVS has nice feature trajectories, which the IBVS may leave the field of camera view. Moreover, plots (3) illustrate that, the pure rotational motion does not affect the translational position due to the block diagonal image Jacobian in 6DVS and PBVS. For IBVS, the position of end-effector is modified during the pure rotational motion performance, because the controlled features are highly coupled. This motion decoupled property for HYVS and 6DVS is also compared in Test 5.

6.2.3 Test 3: Local Minima

In this test we evaluate a common problem in classical IBVS: local minima. By definition, local minima are cases where $V = 0$ and $s \neq s_d$. So, a local minimum is reached when the point velocity on the robot end-effector is zero while its final position is far away from the desired position. At that position, the errors $s - s_d$ in image plane do not completely vanish (residual error is approximately two pixels on each u and v coordinate). Introducing noise in the image measurement leads to the same results.

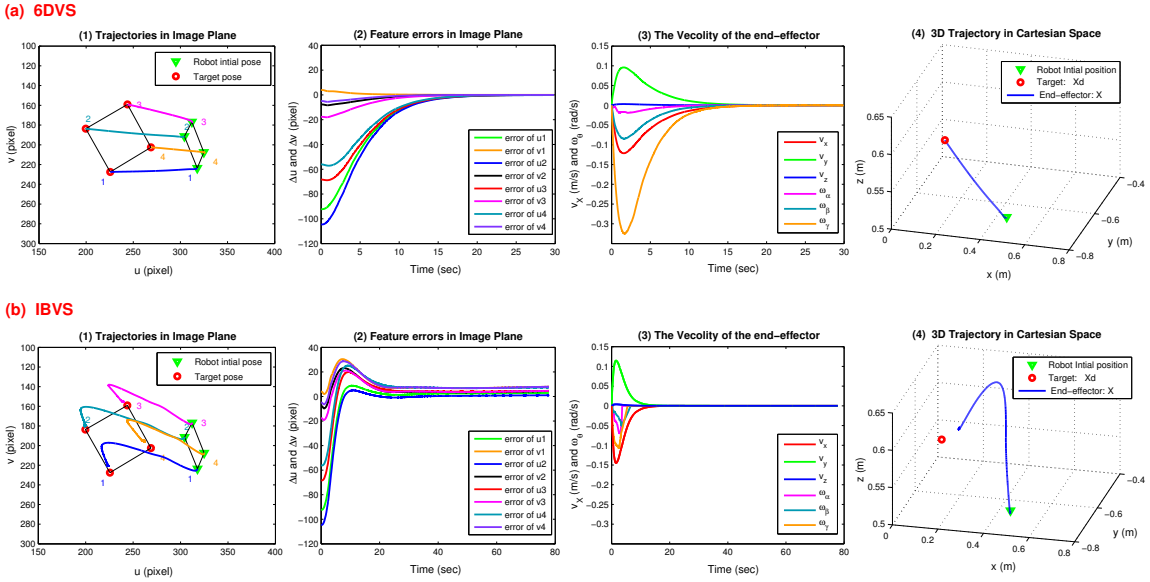


Figure 6.4: Simulation Test 3: Reaching (or not) a local minimum.

Reaching such a local minimum is illustrated in Fig. 6.4 (b) for IBVS. Each component of the feature errors e has a exponential convergence but is not exactly zero ($s \neq s_d$) in plot (2) while the robot velocity is close to zero in Fig. 6.4 (b) (3). It is clear from Fig. 6.4 (b) (4) that the system has been attracted to a local minimum far away from the desired position.

In the proposed scheme (6DVS), the image Jacobian J_{img} has full rank of 6, which implies there are no local minima. The global minimum is correctly reached from the same initial position if the proposed J_{img} is used in the control scheme (Fig. 6.4 (a)). In this case, the trajectories in image plane are straight and each component of the errors e has a exponential convergence to zero without local minima. Moreover, when the velocity reaches zero, the errors in image plane and Cartesian space are both close to zero ($V \to 0, \Delta s \to 0, \Delta X_b \to 0$).

6.2.4 Test 4: Robustness

For this test we compare the robustness of the proposed 6DVS, the conventional IBVS and PBVS to camera errors. These errors are formulated as:

- Camera intrinsic parameter errors $\hat{f} = 1.1f$.
- Camera extrinsic parameter errors $\hat{T}_c^b = 1.1T_c^b$.

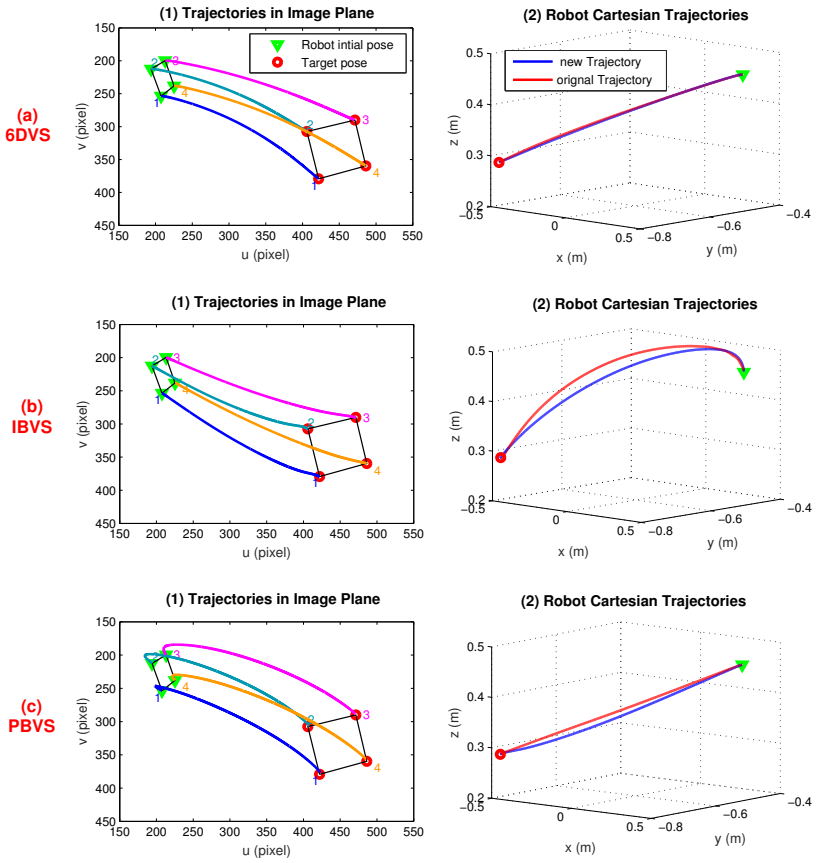


Figure 6.5: Simulation Test 4: Comparison of the robustness of 6DVS, IBVS and PBVS with effects of camera errors.

The comparison of the controllers in terms of image feature trajectories and 3D Cartesian trajectories are evaluated in Fig. 6.5. The results from this test show that despite the camera errors, the controllers for each of the evaluated approaches do not become unstable. The resulting Cartesian and image trajectories have some notable differences (see Fig. 6.5), but can be still accurately served back to the "desired" Cartesian or image locations.

Due to effects of the camera errors, the end-effector Cartesian trajectory of PBVS deviates from the original straight-line trajectory to a circular motion, and slight effects on the image trajectories can also be observed (see Fig. 6.5 (c)). However, slight differences in the end-effector Cartesian trajectory of IBVS are shown in Fig. 6.5 (b). As expected, the image trajectories for IBVS are robust to camera errors. This is because in the IBVS control loop only the image-based controller is affected by the camera modeling errors, while in the PBVS control loop both the PBVS controller and the pose estimation are affected by the camera errors.

Fig. 6.5 (a) illustrates that the effects on both Cartesian and image trajectories for the proposed 6DVS are minor. The Cartesian trajectory is still straight and the image trajectories are almost the same. Hence, 6DVS is as robust with respect to camera calibration errors as IBVS.

6.2.5 Test 5: Motion Decoupling

For standard 2-1/2D VS, the desired control input can be computed from (6.16)

$$\begin{bmatrix} \upsilon_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} J_v^+ & J_v^+ J_{vw} J_w^+ \\ 0 & J_w^+ \end{bmatrix} \begin{bmatrix} \dot{X}_h \\ \dot{\theta} \end{bmatrix}. \tag{6.24}$$

and the control input for the proposed 6DVS is given by (6.22)

$$\begin{bmatrix} \upsilon_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} J_v^{-1} & 0 \\ 0 & J_w^{-1} \end{bmatrix} \begin{bmatrix} \dot{X}_s \\ \dot{\theta} \end{bmatrix}. \tag{6.25}$$

From equation (6.24), the translational control input is $\upsilon_c = J_v^+(\dot{X}_h + J_{vw}\omega_c)$, which combines the position error with the error that would be induced by the rotational motion due to ω_c. Therefore, the rotational motion can also affect the position of the end-effector. Contrary to 2-1/2D VS, the translation control input for the proposed scheme defined in (6.25) is $\upsilon_c = J_v^{-1}(\dot{X}_s)$, which only be modified through the position error.

Motion decoupling is compared between the proposed 6DVS and conventional 2-1/2D VS in this test. Both approaches receive the same desired Cartesian position and orientation. First the desired position is given and both methods perform equally well. As shown in Fig. 6.6, the trajectories in both Cartesian space and the image plane are straight in the position task.

At around $t = 30s$, the desired orientation is modified. In the standard 2-1/2D method, a triangular interaction matrix is used for motion control, as defined in (6.13). Hence, the rotational error can affect the translation since the Jacobian J_{vw} exist, as shown in Fig. 6.6 (b). The visual signals are coupled and both position and orientation in Cartesian space are changed when only the desired orientation is modified. The same results also can be seen in the Test 2 (Fig. 6.3 (b)), where small translational motion is shown in pure rotational motion case.

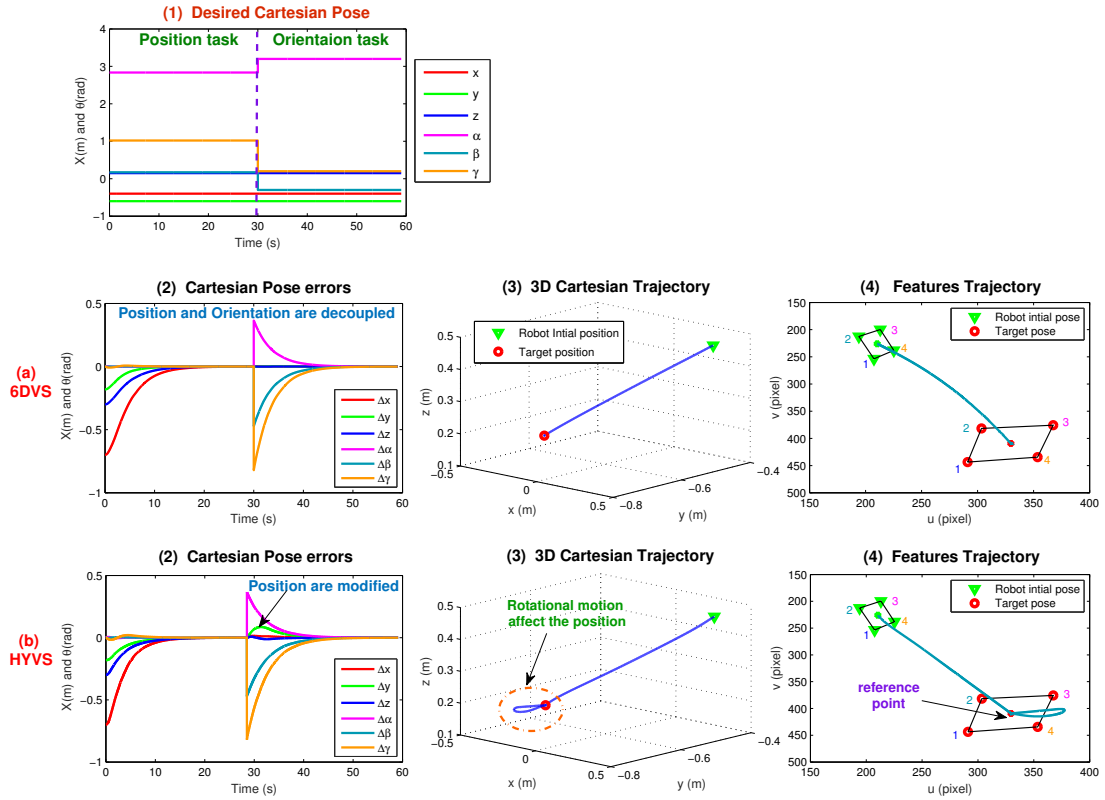


Figure 6.6: Simulation Test 5: Decoupling analysis of position and orientation.

For the proposed 6DVS, we introduce the virtual visual space to decouple the control features and get a block diagonal image Jacobian. This decoupling of rotational and translational motions allows a better control design. Fig. 6.6 (a) demonstrates the decoupled performance using the proposed method.

6.3 Conclusion

In this chapter, the simulation tests to compare the behavior of the different control schemes are presented. Simulation results of five different tests demonstrate the novel properties and better performance of the proposed 6DVS algorithm over conventional VS approaches. The simulation results are shown in Table 6.3. According to the results, 6DVS has a reliable straight 3D Cartesian trajectory like PBVS, and straight feature trajectories like IBVS. It combines the advantages of PBVS in terms of controlling straight trajectories in Cartesian Space, and the advantages of IBVS in terms of controlling image trajectories. Moreover, 6DVS allows to avoid local minima (unlike IBVS) and is robust to camera calibration errors. Contrary to classical

Table 6.3: Comparison of Visual Servoing Schemes

	IBVS	PBVS	HYVS	6DVS
Straight Cartesian Trajectory		✓	✓	✓
Straight Feature Trajectory	✓		✓	✓
No Image Singularities			✓	✓
No Local Minima				✓
Robustness to Image Noise				✓
Robustness to calibration Errors	✓		✓	✓
Motion Decoupling		✓		✓

2-1/2D visual servoing, 6DVS decouples the control of the translational and rotational motion of robot end-effector due to the diagonal image Jacobian.

According to simulation results, these new features perform better than classical ones since the system combines the advantages of 2D and 3D visual servoing. The new visual servoing does not need a precise camera calibration and presents very interesting decoupling and stability properties. Thanks to its simple structure, analytical results on its robustness with respect to calibration errors have been obtained. Furthermore, the proposed algorithm can be integrated in a practical human-robot-interaction scenario with environmental and kinematic constraints, which can generate a trajectory free of collisions and singularities. The experiments on a 6DOF industrial manipulator and real world applications are presented in the next chapter.

Chapter 7

Experiments and Real Applications

In the real world experiments, safety is a primary concern for industrial robotics. Control of the physical interaction between a robot manipulator and the environment is crucial for the successful execution of a number of practical tasks. During task execution, the environment may set constraints on the geometric paths that can be followed by the end-effector, limitations on the force or other constraints for safety. Meanwhile, the robot also has some kinematic constraints. Hence, a robot must accomplish a task while satisfying these constraints, thus requiring a control framework for combining tasks and constraints.

In this chapter, a framework for integrating the proposed visual servoing system in previous chapters, with environment constraints in a human-robot interaction (HRI) scenario is presented. Several constraints such as robot singularities and collision avoidance, have been integrated in the framework using an operational space impedance controller for the robot end-effector. The artificial potential field approach [162] is used to model these environment constraints. Several experiments are performed to validate and evaluate this work on a standard industrial robot in a realistic human-robot interaction scenario.

7.1 Control Framework with Environment Constraints

We propose a framework which supports several types of constraints, such as: robot singularities, (self-/obstacle) collision avoidance and joint limits. The constraints are modeled as forces F using artificial potential field. Fig. 7.1 shows the integration of the control with the different environmental constraints and robot's configuration constraints. The environment constraints, computed at torque level, are added to the desired tracking torque to control the robot move-

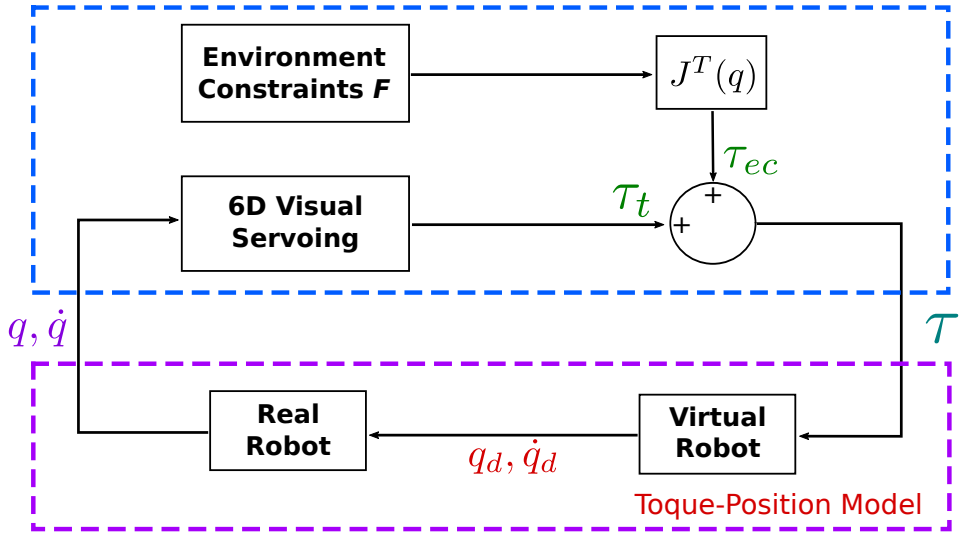


Figure 7.1: 6D visual servoing framework with environment constraints.

ment. Hence, the torque level decomposition of the global task (τ_t) and the secondary control task (τ_{ec}) is represented by

$$\tau = \tau_t + \tau_{ec} \tag{7.1}$$

$$= \tau_t + \sum_{i=1}^{N} J_i^T(q) \cdot F_i \tag{7.2}$$

where τ_t is the global task torque using the visual servoing and the virtual force F_i is represents the environment constraints modeled through artificial potential field.

In the real experiment, we integrate the visual servoing system in a HRI scenario, where environment constraints must be included to generate a safe and singularity-free trajectory for the robot. The definitions of constraint forces are presented in the following subsections.

7.1.1 Joint Limits

For industrial robotics, certain tasks may lead to uncontrollable or unwanted behavior of the robot. To guarantee the safety of the robot, the constraints imposed by the robot's structure should always be satisfied, e.g. the joint limits, robot configuration singularities. Important points that need to be respected are the limitations of joint angles, velocities and accelerations. The resulting joint position q needs to be in a certain bounding region $q_{min} \leq q \leq q_{max}$ (hard limits), and should not violate these limits to prevent physical damage to the robot. The

repelling forces are computed by

$$F_{JL} = \sum_{i}^{N} F_{JL_i} \tag{7.3}$$

and

$$F_{JL_i} = \begin{cases} K_{JL_i} P_{JL_i} D_{JL_i} - B_{JL_i} \dot{X}_{ef}, & \text{if } q_i \geq q_{\text{upper}_{\text{JL}i}} \text{ or } q_i \leq q_{\text{lower}_{\text{JL}i}} \\ 0, & \text{if } q_{\text{lower}_{\text{JL}}} \leq q_i \leq q_{\text{upper}_{\text{JL}}} \end{cases} \tag{7.4}$$

where $q_{\text{lower}_{\text{JL}}}$ and $q_{\text{upper}_{\text{JL}}}$ are the threshold vectors of safety joint position (away from the hard limits). $K_{JL_i}, B_{JL_i} \in \mathbb{R}^{3\times3}$ are constant matrices, and D_{JL_i} is the direction of gradient for the maximum manipulability factor μ. Then

$$P_{JL_i} = e^{\alpha_{JL_i} q_{\text{limit}_i}} - 1 \tag{7.5}$$

where α_r is a constant to control the stiffness of the applied force, and $q_{\text{limit}} = (q_{\text{lower}_{\text{JL}}} - q_i)$ or $(q_i - q_{\text{upper}_{\text{JL}}})$.

7.1.2 Robot Singularity

The singularity avoidance formulation is similar to that for joint limits avoidance, since some special configuration of the joint position is reached when the singularities occurs. Therefore, the robot system needs to stay away from the singularity joint position $q_{\text{singularity}}$, and the corresponding force is given by (similar as (7.4))

$$F_r = \begin{cases} K_r P_r D_r - B_r \dot{X}_{ef}, & \text{if } q_i < q_{\text{threshold}} \\ 0, & \text{if } q_i \geq q_{\text{threshold}} \end{cases} \tag{7.6}$$

where $\dot{X}_{ef}$ is the linear velocity of the end-effector, $K_r, B_r \in \mathbb{R}^{3\times3}$ are constant matrices, and $q_{\text{threshold}}$ is away from the singularity joint position $q_{\text{singularity}}$. We define Δq as the absolute value of the difference between q_i and $q_{\text{threshold}}$, D_r is the direction of gradient for the maximum manipulability factor μ. Then, $P_r = e^{\alpha_r \Delta q} - 1$, where α_r is a constant. In our case, the robot reaches the singular condition when $q_3 = 0$.

7.1.3 Collision Avoidance

Collisions need to be avoided for the static (i.e., the work-bench) and dynamic elements in the environment (i.e., the human and moving obstacles). In this task, the avoidance is done in a reactive way with a dynamically updated collision scene that is interfaceable with a variety of sensors.

We use the artificial potential field methodology to compute the forces for collision avoidance, where the obstacles are considered as repulsive surfaces for the manipulator. To compute the

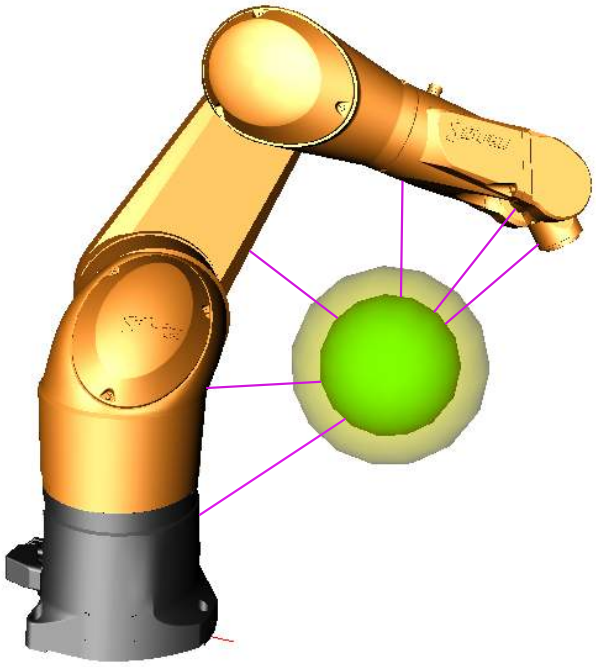


Figure 7.2: Computing the repelling forces of an obstacle.

virtual forces that repel the robot from surrounding obstacles, we need to compute the minimum distances d_c of all objects in the environment model (including self-collision) to all body parts of the robot. Fig. 7.2 depicts the body parts of the used robot with an example of the minimum distances d_c (red lines) from an obstacle to a given joint configuration. If a distance is below a chosen security threshold (transparent bubble), the distance is used to compute virtual forces on the robot using potential fields.

We measure the distances of arbitrary shapes to each joint of the robot model. After calculating the minimum distance vectors V_c in Cartesian space (i.e. the direction of the applied virtual force), we transform them to forces and find the overall motion of the robot to avoid the collision.

$$\tau_c = \sum_{i}^{N} J_i^T(q) \cdot F_{c_i} \tag{7.7}$$

$$F_{c_i} = K_{c_i} P_{c_i} V_{c_i} - B_{c_i} \dot{X}_{c_i}, \tag{7.8}$$

with N being the number of bodies of the robot and V_c is the direction of the applied force.

$K_{c_i}, B_{c_i} \in \mathbb{R}^{3\times3}$ are constant PD control parameters. The repelling potential function is

$$P_{c_i} = \begin{cases} e^{\alpha_c(\mathrm{Dist_c}-d_{c_i})} - 1, & \text{if } d_{c_i} \leq \mathrm{Dist_c} \\ 0, & \text{if } d_{c_i} > \mathrm{Dist_c} \end{cases} \tag{7.9}$$

with $\mathrm{Dist_c}$ being the distance at which the potential field function is applied (see transparent bubble in Fig. 7.2). d_c is the minimum distance between the obstacle and robot arm, and α_c is a constant to control the stiffness of the applied force.

In our case, we use the Kinect device to generate a normals map of the environment, where the normals indicate the direction of the forces. This includes the obstacles, the robot itself and the table. Note that in our design, only the position is constrained by the environment and the orientation is free. More details are provided in the experiment section 7.3.

7.1.4 Torque to Position Model

The Impedance control generates a virtual force that can be used to define a desired dynamic behavior for a robot manipulator. The changes in the robot dynamic behavior are reflected in the robot trajectory which is generated on-line and takes into account the target task and knowledge of the environment (external factors). One way of generating this trajectory is to apply the total torque to a virtual robot, whose dynamic behavior should be similar to the real robot. This virtual robot will generate the desired joint positions/velocities ($q_d/\dot{q}_d$), which will be used as the input of a position/velocity controller of the real robot. This approach is called the *Torque to Position Model* (see Fig. 7.1). The output joint position/velocity ($q/\dot{q}$) is processed by the open architecture control of an industrial robot. This approach allows the implementation of different control strategies in standard industrial robots that typically only offer position/velocity level interfaces under open architectures.

7.2 Visual Servoing System Architecture

Fig. 7.3 demonstrates robotic experimental setup in a human-robot interaction scenario which includes a 6DOF industrial robot arm, a control unit, a stereo vision system with 2 USB cameras, the user and the work station. The experimental setup consists of 3 sub-systems, 1) a visual stereo tracker, 2) the robot control system and 3) the 3D visualization System.

7.2.1 Visual Stereo Tracker

The stereo system is composed of 2 USB cameras fixed on a tripod, in a eye-to-hand configuration. The stereo rig is uncalibrated with respect to the robot base frame (the same as the world

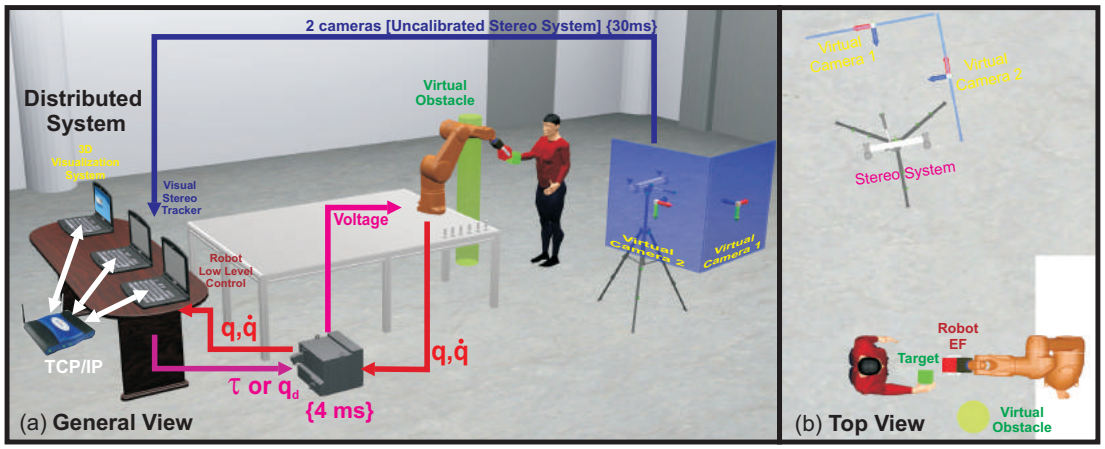


Figure 7.3: Robotic experimental setup in a Human-robot Interaction scenario.

frame) and can be manually moved. The parameters of the virtual cameras are selected such that J_α is always non-singular. In order to compute torque (τ) and avoid a multiple-sampling system, an extended Kalman filter (EKF) is used to estimate the visual position (sampling period $4ms$), whereas the reference is updated each $30ms$ with the real visual data of both cameras.

2D image features are extracted from a stereo vision system with AR markers. We use the ArUco library[1] which is based on OpenCV to detect markers. Every marker provides 2D image features for 4 corner points. 3D position and rotation with respect to the camera frame are obtained from the image features using the camera intrinsic parameters.

7.2.2 Robot Control System

The robot system comprises of a 6DOF StaübliTX90 industrial robot arm, a CS8C control unit and a Workstation running on GNU/Linux OS with real-time kernel (Fig. 7.3). The data communication between the PC and the control unit is based on TCP/IP in a local network. Here, the robot is controlled in position mode q_d using a Low Level Interface (LLI) library.

7.2.3 3D Visualization System

This module provides a real-time visualization of the static elements as well as active agents in the scene such as robot, objects, human etc. It has been developed using Coin3D with OpenGL as backbone, using the Robotics Library[2] [163]. The complete test bed environment

[1] http://www.uco.es/investiga/grupos/ava/node/26
[2] www.roboticslibrary.org.

has been modeled using Virtual Reality Modeling Language (VRML) models. This system updates the configuration of the robot arm and the positions of the target in real-time via a TCP/IP interface.

7.3 Experiments

Two experiments are performed to validate and evaluate this work on a standard industrial robot in a realistic human-robot interaction (HRI) scenario. In the first experiment, we control the robot without environment constraints to better illustrate the stability of the control scheme and the convergence of 6D visual trajectory error. It focuses on the 6D visual servoing algorithm as an application of a teaching interface, where the user defines the pose of the end-effector using a visual marker and this information is later used to define the desired visual task. The second experiment uses the 6D visual servoing algorithm for tracking a moving target in real-time. This experiment also provides an example of how this work is used in a practical human-robot interaction scenario. To this effect, several other features such as singularity avoidance, self-collision avoidance and obstacle detection and avoidance are implemented to ensure safety of the robot and human. This experiment also shows how the orientation matrix is estimated on line enabling an uncalibrated visual servoing system. Occlusions due to camera placement can be handled in a natural and intuitive way by simply moving manually the camera to a better position.

7.3.1 6D Visual Tracking

This scheme is implemented on a robotic platform with six degrees of freedom and an eye-to-hand configuration. Two AR markers are used for 6D tracking task, where one is held by the user and the other is attached on robot end-effector. 3D position and 3D orientation of AR markers are tracked by the stereo vision system. The task function of the proposed visual servoing is defined as the error of 2D image features from 4 corner points of markers.

This experiment consists of two phases: teaching and execution.

7.3.1.1 Teaching Interface

In this experiment, we provide a teaching interface for the user, see Fig. 7.4 (a), where the user is holding an AR marker, which is detected by the stereo cameras system and provides 2D image features. A red square and a marker ID (displayed in cyan) in the image shows the detection. In this task, the user moves the marker, creating some visual trajectories, such as,

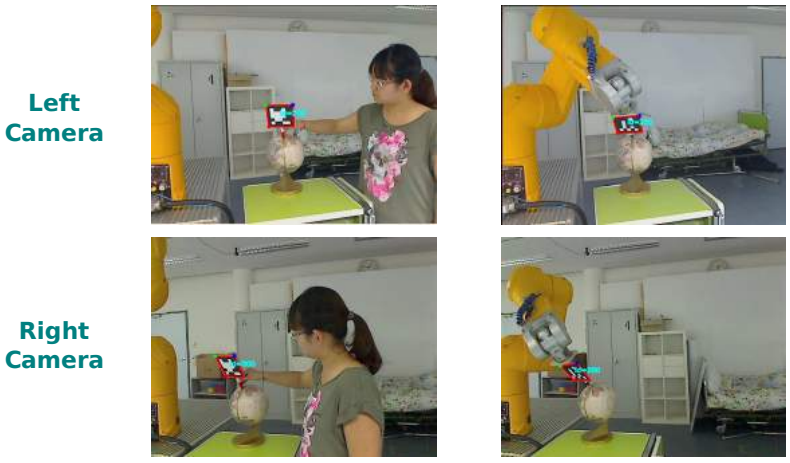


Figure 7.4: Snapshot of the 6D visual tracking.

two orthogonal straight lines on the table and two smooth curves on the surface of the globe. These trajectories include both translation and rotation motions. During the movement, the 2D features for 4 points of the target in the camera frames are recorded and saved. At certain points, when the marker is lost or can not be detected, it saves the last available data, which guarantees that the desired pose can be reached and is safe for the robot execution.

7.3.1.2 Automatic Execution

After the teaching phase, the robot can execute the recorded visual trajectories. Another AR marker with the same size is attached to the robot end-effector. The current robot position (W_s) is obtained from the visual features tracked from this marker. Target inputs to visual servoing are the 2D image features which were recorded in the teaching phase. From the recorded features we extract our desired visual feature vector $W_{s_d} = [x_{s_d}, y_{s_d}, z_{s_d}, \alpha_{s_d}, \beta_{s_d}, \gamma_{s_d}]^T$, which is used to create the error function (see Section 5.2).

Visual servoing is accomplished by driving the error function to zero. In our case, the error function is $e = W_s - W_{s_d}$. According to the properties of our image Jacobian and the control scheme, when the errors in the virtual visual space converge to zero, the errors in Cartesian space also converge to zero without local minima. Therefore during execution, the AR marker on the robot end-effector shows identical linear and angular motions as instructed in the teaching phase (Fig. 7.4). This experiment illustrates how the visual servoing system can track a given desired trajectory.

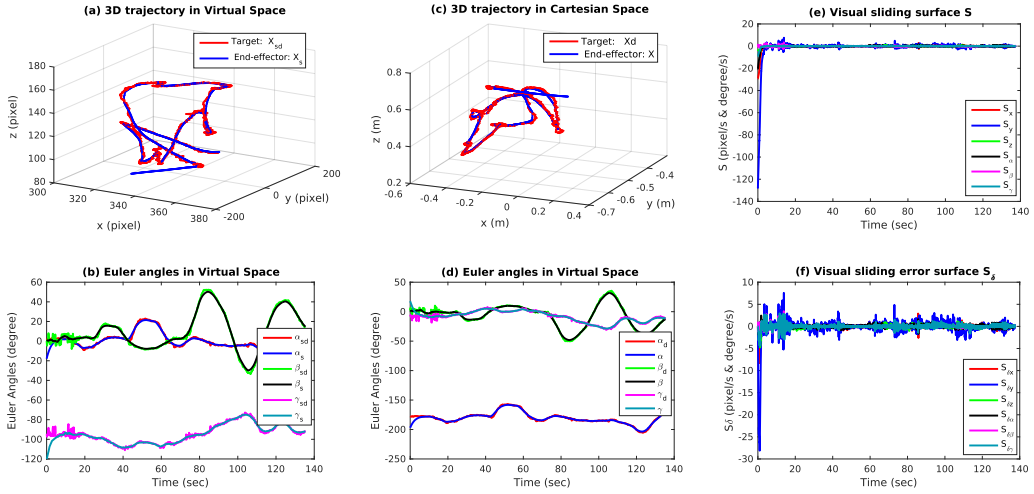


Figure 7.5: Experiment results for 6D visual tracking in both spaces.

Experimental results are depicted in Fig. 7.5. Plots in the first column (a) and (b) show the 3D linear and 3D angular tracking in the virtual visual space while the second column (c) and (d) depicts the target trajectory tracking in Cartesian space. The red lines in plots (a) and (c) are the target trajectories, which exhibit some noise and chattering due to the unsteady movement of the user. However, the blue lines which show the trajectories of the robot end-effector, are smooth and chatter free. From the plots of the last column (e) and (f), we can see that the control signals converges with a satisfactory behavior.

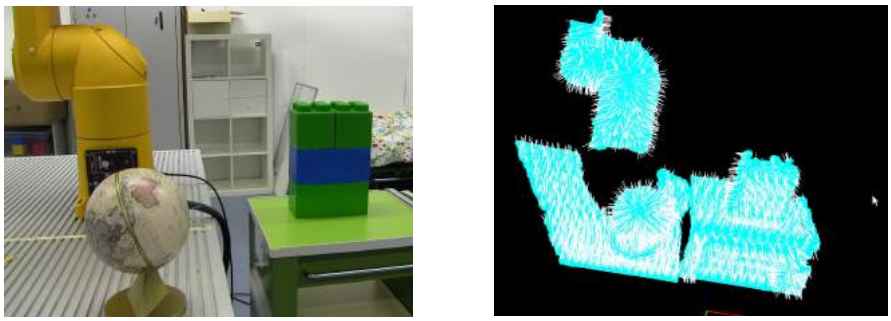
7.3.2 6D Uncalibrated IBVS in Human-Robot Interaction Scenario

In this task, we illustrate real time tracking in Human-Robot Interaction scenario. We use AR markers to identify the target pose and the current pose of the robot end-effector. The target is carried by a human, and the control goal is to make the robot end-effector follow the target placed in the human's hand. In order to maintain safety and for visualization purposes during the interaction, we keep a 0.2 meter offset between the robot end-effector and the target marker in the y_b axis of the workspace.

This experiment illustrates the 6D pose tracking in both virtual visual space and Cartesian space. Moreover, as mentioned in Section 7.1, we integrate the adaptive image-based torque controller with environment constraints to generate a safe and singularity-free trajectory for the robot.

7.3.2.1 Real-time obstacle avoidance

The raw point cloud data obtained from a depth sensor (Kinect) is processed to generate a primitive shape decomposition of the point cloud, followed by computation of a minimum volume bounding box (MVBB) for each of the primitive shapes along with their normal directions. These normal directions are stable since they are computed from a robust primitive shape decomposition algorithm [164, 165] which makes them suitable for obstacle avoidance tasks. Fig. 7.6 shows the normals map for the environment, which are used to compute the virtual forces, including (self-/obstacles/table) collision avoidance forces.



(a) (b)

Figure 7.6: Scene perception module: (a) Scene snapshot, (b) Environment normals map.

The experimental obstacle avoidance results are depicted in Fig. 7.7, where the robot end-effector trajectory (blue line), the target (red +) and the obstacle position (green *) can be seen in both, the virtual visual space (a), and in the Cartesian space (b). The visual tracking behavior in the 3 visual axes can be seen in plots (c), (d) and (e) for the 3D position and (f), (g) and (h) for the 3D orientation. Note that even when the target position is not continuous, the robot end-effector exhibits a smooth behavior. In the sections where the robot end-effector is close to the obstacle, we can see the effect of the *Obstacle Avoidance Force* in the trajectory.

7.3.2.2 Interaction results

This experiment demonstrates real-time tracking for the robot end-effector according to the moving target held by the human. We address the tracking for both translation and rotation motions, see Fig. 7.8 (a). The system proves to be stable and safe for HRI scenarios, even in situations where the target is lost (due to occlusions by the robot or the human), Fig. 7.8 (b). In this case, the robot stops and the visual tracking is resumed as soon as the target is visible again.

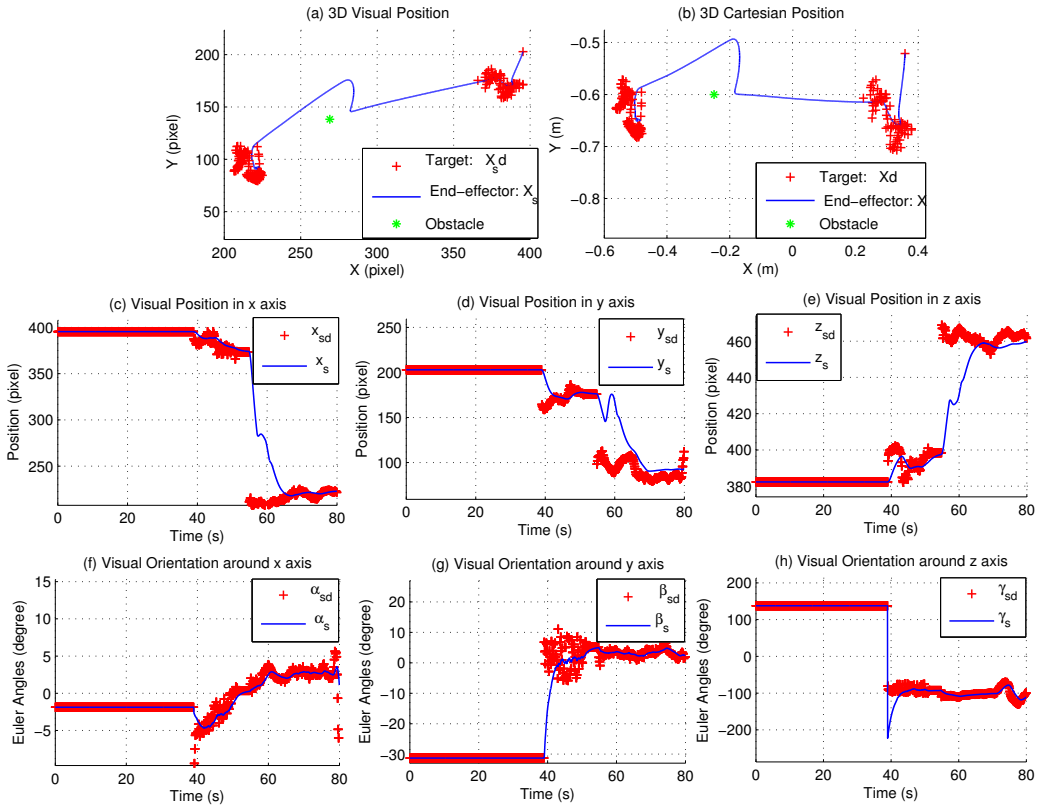


Figure 7.7: Position trajectories with obstacle avoidance.

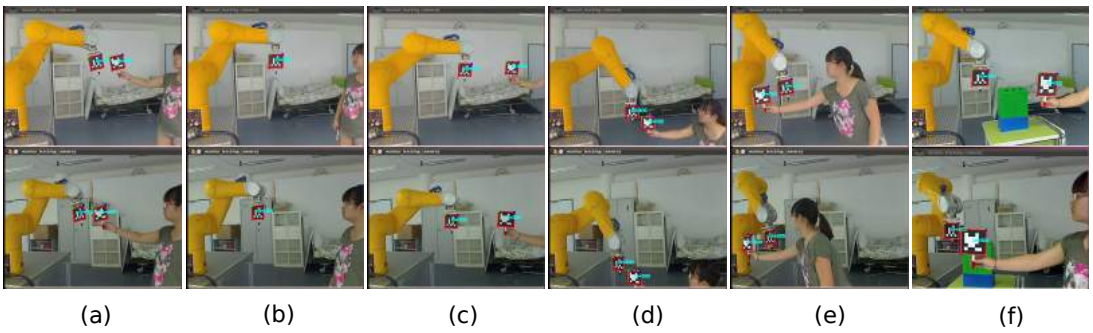


Figure 7.8: System behaviors: (a) Position and orientation tracking, (b) Case when the target is lost, (c) Case with singularity avoidance, (d) Case with table collision avoidance, (e) Case with self-collision avoidance and (f) Obstacle avoidance.

To demonstrate stability, we test our system under several environment constraints. Fig. 7.8 (c) illustrates the results of singularity avoidance, where the robot does not reach the singular condition ($q_3 = 0$), even when the user tries to force it. Using the normal directions of the

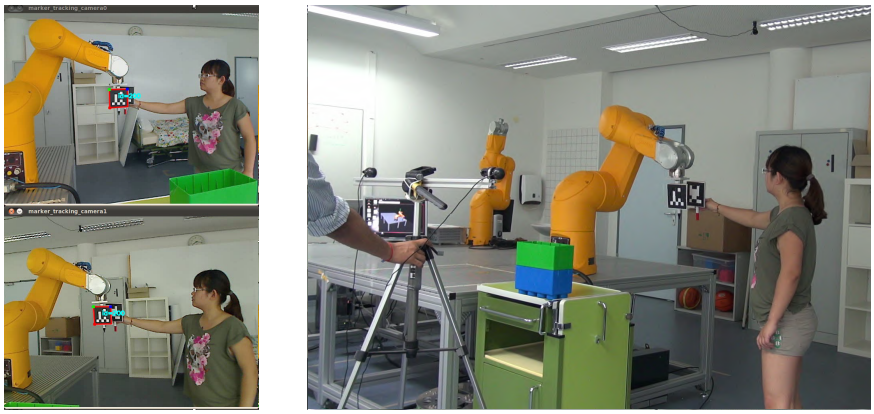


Figure 7.9: Target occlusion: (a) The target is occluded by the robot end-effector, (b) The user manually moves the stereo vision system to a pose where the occlusion is no longer present.

environment provided by the Kinect sensor, the robot can avoid collisions. Fig. 7.8 (d) depicts the table avoidance where the motion of the robot is constrained in the $z_b - axis$ by the height of the table (the end-effector is not allowed to go under the table) but it can still move in the x_b and y_b axes, and (e) shows how the robot handles self-collisions. Fig. 7.8 (f) shows obstacle avoidance while continuing to track the target.

One of the key contributions of this system is the possibility of handling situations where the target object is occluded, and the stereo system can be moved to maintain the target in the field of view. This feature is analyzed in next subsection.

7.3.2.3 On-line Orientation Matrix Estimation

As mentioned earlier in Section 5.4.1, a coarse on-line estimation of the orientation matrix is computed using the real-time information generated by the robot. The remaining estimation errors for the complete Jacobian J_s can be handled by the controller to a certain extent.

Object occlusion occurs in Fig. 7.9 (a), where the stereo system can then be moved to maintain the targets in the field of view, see Fig. 7.9 (b). The camera motion is detected by the system and a process for coarse estimation of the orientation matrix between the stereo system and the robot base frame is initiated. Based on our experiments, the control scheme can handle up to 20% error in this rotation matrix.

After the stereo camera system is moved, the robot performs a small motion and a set of points are collected, as shown in Fig. 7.10 (d). During this stage, the visual servoing is turned off,

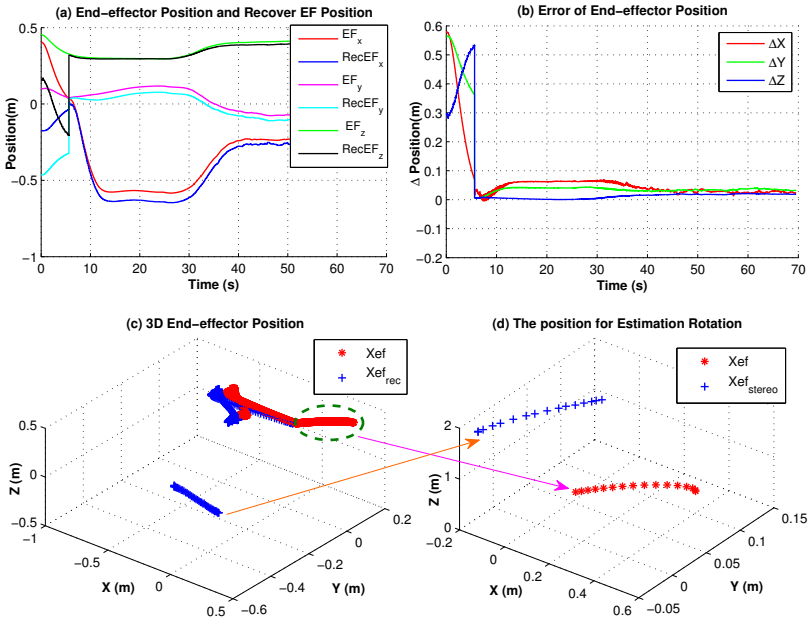


Figure 7.10: Results of the on-line orientation matrix estimation.

and the robot motion is in the free space, where the integration of the environment constraints (obstacle-, self- and singularities avoidance) play an important role for the correct/stable robot behavior. We compare the actual end-effector position with the recovered end-effector position that is obtained by using the estimated rotation. Fig. 7.10 (a) and (c) depicts the comparison of these two position trajectories. It can be observed that the error between them is closed to zero after the on-line rotation matrix estimation, illustrated in Fig. 7.10 (b).

Fig. 7.11 demonstrates the experimental results in the case where the camera is moved. Fig. 7.11 (a) and (b) show the 3D positions in the virtual visual space and the Cartesian space. In the experiment, at some instance, the robot loses the visual tracking because the target is occluded and the user manually moves the stereo vision system to a new pose, which leads to an incorrect rotation matrix for the control. In this situation, our system can handle the problem by using the on-line coarse orientation estimator and then resuming the visual tracking. Fig. 7.11 (c), (d), (e) illustrate when the cameras are moved at $t \approx 40s$, and the 3D position shows big errors. After the orientation estimation, the 3D visual position resumes tracking ($t \approx 42s$). The 3D visual orientations are depicted in Fig. 7.11 (f), (g) and (h).

A video where more details for all these experimental results are illustrated can be seen in: http://youtu.be/arNFrbJ0Lj4

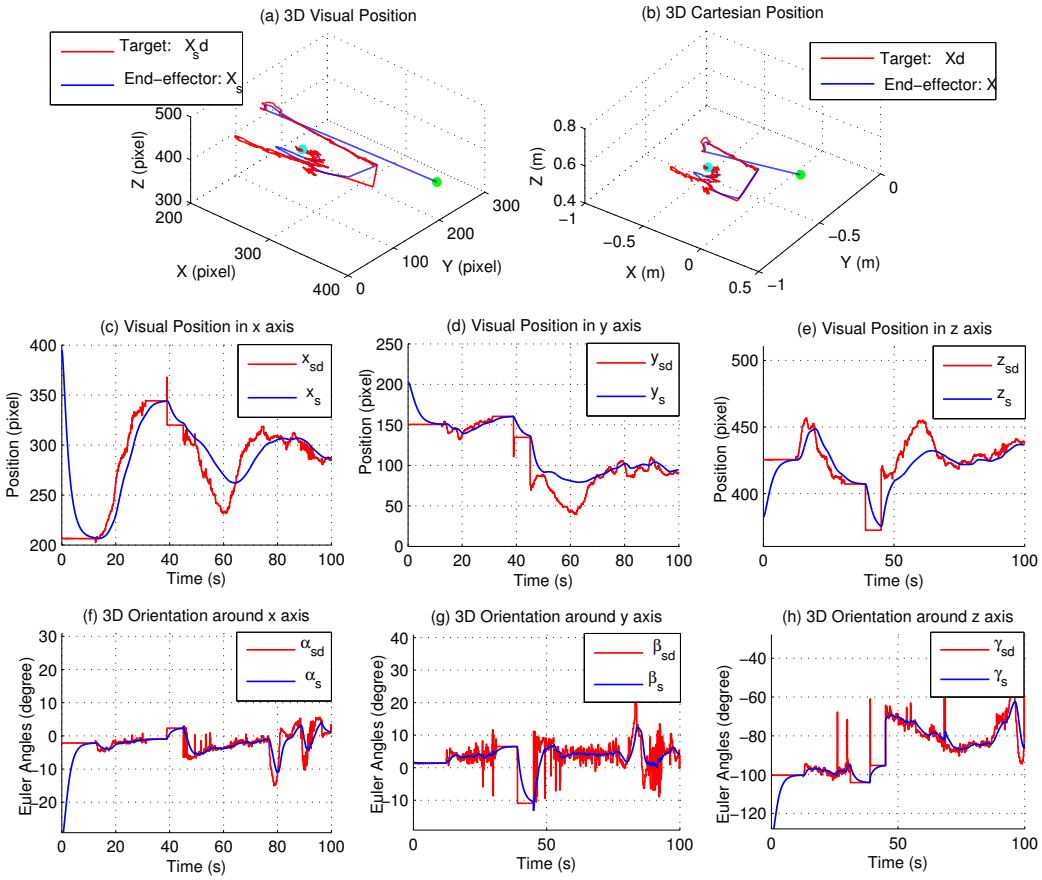


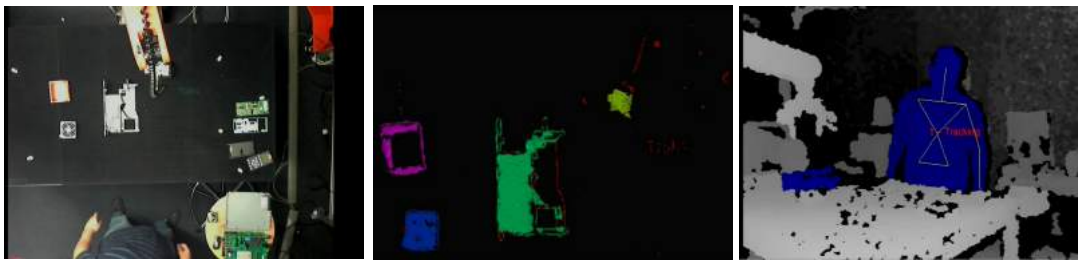
Figure 7.11: The trajectories of robot end-effector when the stereo camera system is moved by the user.

7.4 Application: Human-Robot Cooperation

We have tested a typical industrial application, assembly, to evaluate the capabilities of the system. Since our work is targeted primarily towards industrial applications, the test is conducted using a standard industrial robot, the Comau Smart Six with a C4G open architecture controller in an industrial scenario.

The application is the assembly of a power converter box. This operation consists of a number of steps, actors and objects which are identified by the perception module. Some of components are complex, high precision assembly tasks (e.g. screwing) that are suitable for the human, while some involving lifting heavy objects are more suitable for the robot. Hence, the Human-Robot cooperation (HRC) is need for this assembly application. The proposed visual servoing with multiple constraints is used to drive the robot movement safely.

7.4.1 Multiple Input Modules

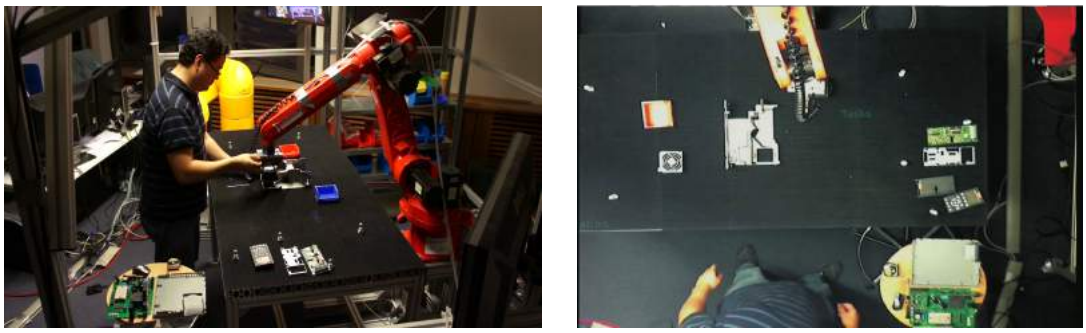


(a) (b) (c)

Figure 7.12: Multiple input modules: (a) A snapshot of the HRC application, (b) Perception module for object recognition and pose estimation, (c) Articulated human tracker.

For this application, multiple input modules are need, e.g. perception module, human tracking module. The object recognition and pose estimation algorithm used in this experiment is presented in [164, 166, 167, 168]. Such environments are typically unstructured and objects are often occluded by the human. Noisy point cloud data is obtained from the low-cost depth sensor (Microsoft Kinect) used in the experiments. Also, accurate object poses are required for precise pick-and-place tasks, due to mechanical limitations of the 2-fingered gripper. Given these constraints, an accurate algorithm which can handle occlusions, partial views and sensor noise is essential for such scenarios. Fig. 7.12 shows the object perception module and the human tracking module.

7.4.2 Automatic Assembly



(a) (b)

Figure 7.13: HRC in an assembly process: (a) interaction, (b) obstacle avoidance.

This application is aimed at automatic execution of semantic process plans in industrial scenarios. The perception module provides the objects poses, which are used to control the robot movement using the proposed visual servoing. The human tracking module provides the human hand position which is used in the cooperation with robot actions. Besides, the constraints imposed by the robot's structure and the external environment are also considered for the safety purposes, such as robot joints limits and singularity avoidance, human and obstacles avoidance. All the constraints are combined with 6D visual servoing control using the framework of Fig. 7.1. Fig. 7.13 shows snapshots of this application. Fig. 7.13 (a) demonstrates the physical interaction with the robot by using the force sensor and force control in the controller. Fig. 7.13 (b) is the top view of obstacle avoidance during the assembly task.

A video illustrating results and the process for this applications mentioned above can be found at : https://youtu.be/TQB6GsUnbDI.

7.5 Discussion: Multiple Tasks Control with Priority

In section 7.1, the system uses a force field for modeling the environment and task constraints for controlling the robot in complex and unstructured scenarios. Although this approach provides good results and can be extended easily to accommodate different constraints, the problem of local minima can not be completely solved using this approach. When there are multiple tasks or constraints, the constraints maybe conflict with and affect the global task since they are not at different priorities. Hence, the global task accomplishment can not be guaranteed in the case of conflicting tasks, as seen in the experiment results in subsection 7.3.2.

Therefore, to guarantee accomplishing the global task while satisfying several constraints, we need to design a multi-level hierarchical control structure that allows the establishment of general priorities among tasks. A prioritized, multi-task hierarchical control framework that is able to control forces will be presented in the next chapter.

Chapter 8

Prioritized Multi-Task Control Framework

In order to control robot safely when it interacts with unstructured environment, the robot manipulators are required to respond in real-time to a variety of dynamic constraints characteristic of human environments. Moreover, multiple tasks and constraints need to be simultaneously controlled while accomplishing the global task, including the constraints imposed by the robot's structure and the external environment. Hence, a prioritized, multi-task hierarchical control framework that is able to control forces for industrial is required.

In this chapter, we first review different frameworks for multi-task control of rigid robots, and analyze the individual merits of these dynamic frameworks. Thereafter, we focus on the Whole-Body Control Framework, that allows us to establish a control hierarchy for multiple prioritized tasks. Furthermore, we demonstrate the use of this framework with complicated geometric constraints by testing several types of constraints. Finally, Section 8.6 presents the evaluation of our approach on typical real-world industrial robotics applications.

8.1 Overview

Several frameworks for multi-task control of rigid robots exist in the literature. Many frameworks presented in the 80s, 90s [169, 170, 150, 171] and recently by Smits et al. (iTaSC) [172] and [173, 174] work at the kinematic level, computing the desired joint velocities ($\dot{q}$) or accelerations ($\ddot{q}$). These approaches are not suited for robots that interact with the environment, because they do not allow for force control. This motivated a more recent trend of torque control strategies [175, 176, 177, 178, 179], which consider the dynamics of the robot and compute the desired joint torques (τ). This approach can also improve tracking, as it compensates for

the dynamic coupling between the joints of the multi-body system.

Since we are interested in controlling robots that interact with the environment, we focus on frameworks that allow for force control. Peters et al. [180] demonstrated that we can derive several of these well-known torque control laws under a unifying framework, allowing for force control by setting the joint space control torques. This approach is efficient but not optimal. The Whole-Body Control Framework (WBCF) [175] allows for force control while being optimal, but is not efficient. The framework (TSID) presented by Prete et al. [181] is motivated by designing a control framework that is both optimal and efficient. However, it does not allow for inequality constraints, which are particularly important for modeling joint limits and motor torque bounds. These are very important for safety in industrial robotics.

The frameworks presented above are usually applied on humanoid robots. In this chapter, we present a prioritized, multiple-task control framework that allows for force control for industrial robotics. Although the TSID framework is efficient, we choose the WBCF framework for our work because of the fact that there are only very few DOFs for industrial manipulators. For this case, we have observed that both WBCF and TSID have similar computation times. The main reason for us to choose WBCF is that it is easy to integrate multiple tasks and model inequality constraints.

Our proposed framework is based on a composable structure where several constraints, each describing a robot task or behavior, can be combined with priorities. The framework supports several types of constraints, such as operational task (position and orientation), force (contacts), or inequality constraints (e.g., joint limits, collision avoidance). For safety and efficient control, the framework establishes a control hierarchy among behaviors by enforcing priorities among the different control categories, i.e., safety constraints, operational tasks, and postures. The priorities are accomplished by null-space projections.

8.2 Whole-Body Control Framework

In this section, we review the hierarchical task control framework based on projecting the control of lower priority tasks into the null-space of higher priority tasks. This multi-level prioritized framework allows us to establish multiple priority levels among the different control categories. In this context, we distinguish three priority levels in the hierarchy: Safety constraints (such as contacts, joint-limits, self-collisions), Operational tasks (i.e., position, orientation motion), and Postures (i.e., the residual motion), which should be controlled with different priority assignments. They are treated as independent control entities. The hierarchies are shown in

Table 8.1.

Table 8.1: The Multi-level Prioritized Control Categories

Categories	Priority level	Task Primitive
Safety constraints	1	Joint limit, Obstacle/Self collision, Contact forces
Operational tasks	2	Position, Orientation, Artificial forces
Postures	3	Residual motion

According to Table 8.1, safety constraints should always be guaranteed since they are on the highest priority level, while operational tasks should be accomplished without violating the acting constraints, and the posture should control the residual movement redundancy.

8.2.1 Integration of Constraints

This subsection describes the WBCF presented by Sentis et al. [175]. This framework is based on the *Operational Space Formulation* [182], which was introduced to address the dynamic interaction between the robot's task space motion and force, defining a dynamically consistent task null-space. We first review the fundamental mathematics and begin by describing the robot's joint space dynamics in terms of joint coordinates q with

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau\,, \tag{8.1}$$

where τ is the set of joint torques, $M(q)$ is the joint inertia matrix, $C(q, \dot{q})$ is the Coriolis and centrifugal torque vector, and $G(q)$ is the gravity torque vector.

The Operational Space Formulation describes the torque level decomposition of an operational task ($task_1$) and a secondary control task ($task_2$) according to the torque equation

$$\tau = \tau_{\text{task}_1} + \tau_{\text{task}_2}\,. \tag{8.2}$$

Based on the control algorithm projecting the control of lower priority tasks into the task null-space of higher priority tasks, the torque decomposition can be represented by

$$\tau = J_{\text{task}_1}^{\text{T}} F_{\text{task}_1} + N_{\text{task}_1}^{\text{T}} \tau_{\text{task}_2}\,, \tag{8.3}$$

where J_{task_1} is the Jacobian of task 1, F_{task_1} is a vector of forces, and $N_{\text{task}_1} = (I - J_{\text{task}_1}^{\dagger} J_{\text{task}_1})$ is the dynamically-consistent null space associated with the task$_1$. $J_{\text{task}_1}^{\dagger}$ is the dynamically-consistent generalized inverse of the task Jacobian.

8.2.2 Hierarchical Extension

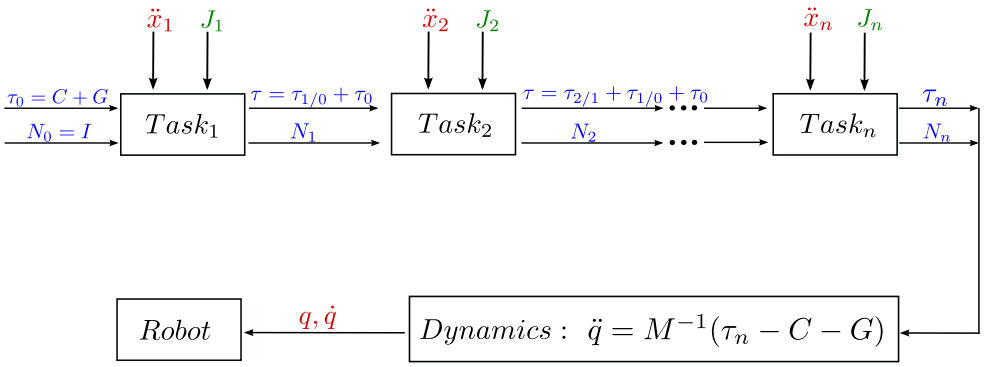


Figure 8.1: A multi-level control hierarchy framework that allows us to establish multiple priority levels among categories.

In this subsection, we propose a control hierarchy that extends the previous decomposition to multiple priority levels. This hierarchy integrates constraints and additional tasks according to desired priorities, while optimizing the execution of the global task. Given n tasks controlling the robot behavior at a given time, the multi-level hierarchy is represented as

$$\tau = \sum_{i=1}^{n} J_{p(i)}^{\mathrm{T}} \cdot F_{p(i)} \tag{8.4}$$

$$F_{p(i)} = \Lambda_{p(i)}\{\ddot{x}_i^* - \dot{J}_i\dot{q} + J_i\, M^{-1}(h - \sum_{j=1}^{i-1} J_{p(j)}^{\mathrm{T}} F_{p(j)})\} \tag{8.5}$$

$$J_{p(i)} = J_i \cdot N_{p(i)}\,, \tag{8.6}$$

where τ, $F_{p(i)}$ and $J_{p(i)}$ are prioritized controls, prioritized forces, and projected Jacobian, respectively. $\ddot{x}_i^*$ is a reference input at the acceleration level, J_i is the task Jacobian, and $h = C + G$. $\Lambda_{p(i)}$ is the task-space mass matrix and $N_{p(i)}$ is an extend null-space matrix containing the null-spaces of all preceding constraints and tasks:

$$\Lambda_{p(i)} = (J_{p(i)}\ M^{-1}\ J_{p(i)}^{T})^{-1} \tag{8.7}$$

$$N_{p(i)} = I - \sum_{j=1}^{i-1} J_{p(j)}^{\dagger}\ J_{p(j)}\,. \tag{8.8}$$

This prioritization strategy minimizes the error of each task under the constraint of not conflicting with any higher priority tasks. The hierarchical control framework is shown in Fig. 8.1.

Remark:

The presented framework is a prioritized, multi-constraints control framework, which guarantees accomplishing the global task while satisfying several constraints. The global task, which normally is a operational task, is achieved using the proposed visual servoing scheme (6DVS) while the potential field techniques are considered to handle dynamic constraints.

8.2.3 Hybrid Control

The framework allows for hybrid position/force control by setting (8.5)

$$F_{p(i)} = \Omega_f f_i^* + \Lambda_{p(i)}\{\Omega_m \ddot{x}_i^* - \dot{J}_i \dot{q} + J_i M^{-1}(h - \sum_{j=1}^{i-1} \tau_{p(j)})\}\,, \tag{8.9}$$

where the selection matrices Ω_f and Ω_m split the control space into force and motion components, respectively. f_i^* represent the constraint forces.

8.3 Types of Constraints and Control Approaches

The control framework supports several types of constraints, such as motion (position and orientation), force (contacts), or inequality constraints (e.g., joint limits, collision avoidance). The robot must accomplish a global task while satisfying several constraints. In this section, we will discuss our approach to handle some important inequality constraints including joint limits, obstacles, and self collisions.

8.3.1 Joint Limit Constraints

For industrial robotics, safety is a very important aspect. To guarantee the safety of the robot and its environment, safety related constraints (e.g., joint-limits, self-collisions) should always be guaranteed, and operational tasks should be accomplished without violating the acting constraints. Our approach handles joint-limit constraints as priority processes and executes operational tasks in the null space of joint-limits constraints.

To illustrate our approach, let us consider the control example shown in Fig. 8.2, where the robot's end-effector is commanded to move toward a target point X_d (global task), while the controller handles joint-limit constraints. When no constraints are active, the end-effector is controlled using the proposed visual servoing (6DVS) to get τ_{task}

$$\tau = J_{\text{task}}^{\text{T}}\, F_{\text{task}}\,, \tag{8.10}$$

where τ is the vector of actuation torques, F_{task} is a control force to move the end-effector toward the desired goal, and J_{task} is the end-effector's Jacobian matrix.

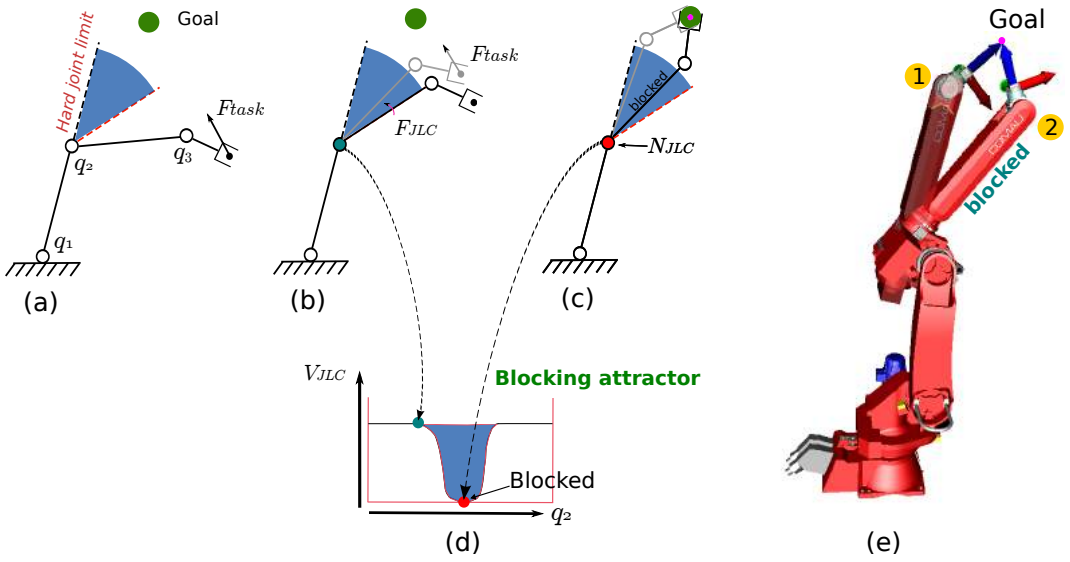


Figure 8.2: End-effector position control under joint limit constraints: In image (a), the robot's end-effector has been commanded to move toward a desired goal. The blue area defines a joint limit activation zone for the elbow joint. When this area is reached (b), a control approach is implemented to block the elbow joint while pursuing the goal (c). Image (d) depicts the attractor potential used to block the elbow joint inside the activation area. (e) shows the experiment results.

When the elbow joint enters the activation zone (shown in blue), we project the task in the constraint-consistent motion manifold, decoupling the task from the constraint. At the same time, an artificial attraction potential is implemented to prevent the elbow from penetrating further into the activation area. The simultaneous control of constraints and operational tasks is expressed as

$$\tau = J_{\text{JLC}}^{\text{T}} F_{\text{JLC}} + N_{\text{JLC}}^{\text{T}} \cdot \tau_{\text{task}} \,, \tag{8.11}$$

where N_{JLC} is the dynamically-consistent null space matrix of the constraint Jacobian, F_{JLC} is the vector of blocking forces (in the example a 1D joint space torque), J_{JLC} is the Jacobian of the violating joint (in the example it would be a constant matrix with zeros in non-violating joints and a 1 for the elbow joint). $\ddot{x}^*$ is controlled through a simple PD controller that includes velocity saturation. More details about the controller can be found in Sentis' dissertation [183].

The control of joint-limit constraints is integrated by using the top-most priority level as specified in (8.11), while the operational task is projected into the constraint null-space. Fig. 8.2

illustrates the results of the control. When the elbow joint enters the constraint activation area shown in Fig. 8.2 (b), we apply blocking forces (F_{JLC}) to block the elbow joint while pursuing the goal (Fig. 8.2 (c)). To lock the joint, we use attraction fields as shown in Fig. 8.2 (d). If there are no joint-limit constraints, the robot joints will be violated during motion, as can be seen in Fig. 8.2 (e).

We consider here potential field techniques to handle dynamic constraints in real time. For example, let us analyze in more detail the joint limit behavior shown in Fig. 8.2. When the joints enters the constraint activation area, we use attraction fields to lock the joints inside the activation area. This potential can be expressed using the following energy function

$$V_{\text{constraint}} = \|q_c - q_{\text{lock}(i,j,k,\ldots)}\|^2, \tag{8.12}$$

where each violating joint has an associated lock position represented by the values $q_{\text{lock}(i,j,k,\ldots)}$, and q_c is the joint position vector involving all violating joints,

$$q_c = \begin{bmatrix} q_i \\ q_j \\ q_k \\ \vdots \end{bmatrix}, \tag{8.13}$$

where $i, j, k, \ldots$ are violating joints. The Jacobian corresponding to this multidimensional constraint is

$$J_{\text{constraint}} = \begin{bmatrix} 0 & \cdots & 1_i & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 1_j & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1_k & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \tag{8.14}$$

which corresponds to a selection matrix selecting components corresponding to violating joints, and the velocities are

$$\dot{x}_{\text{constraint}} = J_{\text{constraint}} \cdot \dot{q}_c. \tag{8.15}$$

In this framework, every task primitive is controlled through a simple PD controller that includes velocity saturation, i.e.

$$\ddot{x}^*_{\text{constraint}} = -k_d(\dot{x}_{\text{constraint}} - \nu v_{\text{des}}), \tag{8.16}$$

$$v_{\text{des}} = \frac{k_p}{k_d} \bigtriangledown V_{\text{constraint}}, \qquad \nu = \min\left(1, \frac{v_{\max}}{\|v_{\text{des}}\|}\right), \tag{8.17}$$

Using the control expression given in (8.16), and the prioritized structure shown in (8.11) we can simultaneously handle multiple joint limits without interrupting the global task.

8.3.2 Obstacle Avoidance

To handle obstacles we apply repulsion fields in the direction of the approaching objects as shown in Fig. 8.3. Repulsion fields can be applied to desired points on the robot's body by using the control structure described in (8.5) and the velocity saturation control law described in (8.16).

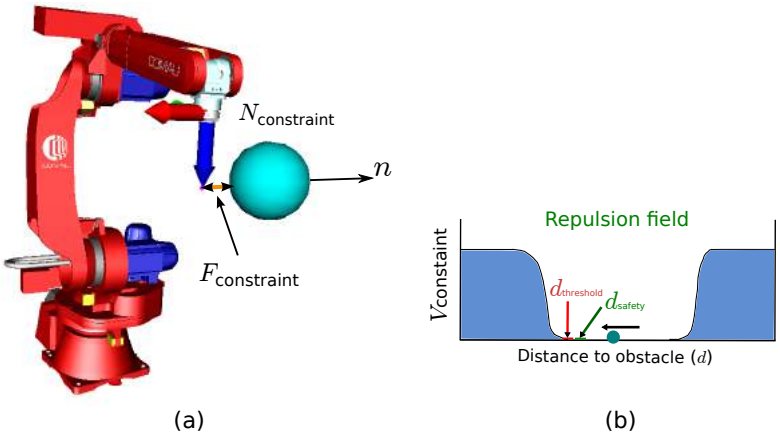


Figure 8.3: Obstacle avoidance: When an obstacle approaches the robot's body, a repulsion field is applied to the closest point on the robot's body.

When handling obstacles we use repulsion fields. A repulsion field is illustrated on image (b) of Fig. 8.3. This potential can be expressed using the following energy function

$$V_{\text{constraint}} = \|d_{\text{obstacle}} - d_{\text{safety}}\|^2, \tag{8.18}$$

where d_{obstacle} is the distance between the obstacle and the closest point on the robot's body and d_{safety} is a desired safety distance. The quantities are defined as follows

$$d_{\text{obstacle}} = x_{\text{robot}} - x_{\text{obstacle}}, \tag{8.19}$$

$$d_{\text{safety}} = K_{\text{safety}} \frac{d_{\text{obstacle}}}{\|d_{\text{obstacle}}\|}, \tag{8.20}$$

where x_{robot} is the Cartesian position of the closest point to the obstacle on the robot's body, x_{obstacle} is the position of the closest point on the obstacle, and K_{safety} is a constant gain determining a safety margin. Though we define the above distances as 3D vectors, obstacle avoidance should be a 1D task acting on the direction of the distance vector. To map a 3D task into 1D space we manipulate the Jacobian associated with the distance vector, removing unnecessary components,

$$J_{\text{constraint}} = \left(R_o^d S_n R_d^o\right) J_{\text{robot}}, \tag{8.21}$$

where R_o^d is a 3D rotation matrix between the robot base frame and a frame aligned with the distance vector, and S_n is a 3×3 selection matrix that selects the components on the normal direction. Although the constraint $J_{\text{constraint}}$ has three rows, its rank is one. As a result, when projecting constraint forces into actuation torques, only the perpendicular direction to the obstacle will be considered.

8.3.3 Self Collision Avoidance

Self collisions are especially important in industrial robotics due to the safety issues. Our approach to avoid self collisions is almost identical to avoiding obstacles. A repulsion field is created to maintain a safety distance between pairs of nearby links,

$$V_{\text{constraint}} = \|d_{\text{selfcollision}} - d_{\text{safety}}\|^2. \tag{8.22}$$

This time the distance vector corresponds to closest points on separate links,

$$d_{\text{selfcollision}} = x_{\text{link(a)}} - x_{\text{link(b)}}, \tag{8.23}$$

and the safety distance has identical form for obstacle avoidance as shown in (8.20).

8.4 Operational Task: Motion Constraints

The basic operational tasks are position and orientation movements with constraints. In the Peg-in-Hole scenario, the first step is to move the robot end-effector to the plane where the hole is on, then the robot needs to search the hole on that plane. Therefore, we test these two cases for our control framework on a 6-DOF industrial manipulator in simulation. The robot kinematics, dynamics, and low-level robot control are simulated using the Robotics Library[1] [163] with a realistic robot model.

8.4.1 Move to a Plane

In this case, the robot is commanded to reach a plane with its end-effector while controlling the movement only along the plane normal direction. The plane is defined by a point (p) and its normal vector (n). During task execution, the distance d between the robot end-effector and the plane converges to zero when it reaches the plane. The orientation of the end-effector is finally aligned with the normal direction, which means the z axis of the end-effector is parallel to the plane normal n. The constraints are described in Fig. 8.4 (a). There are two steps:

[1] http://www.roboticslibrary.org/

Step 1: Move the robot to Plane$_A$ ($d \to 0$) with

$$\ddot{x}^*_{\text{position}} = \Omega_m(\ddot{x}^*_d)\,, \tag{8.24}$$

where $\ddot{x}^*_d$ is the desired input to drive the robot end-effector to move toward the point p on Plane$_A$. This is computed using the proposed visual servoing scheme (6DVS). However, the task is to reach a plane with the movement only along the plane normal direction n, therefore we accomplish dynamic decoupling in the controllable directions according to the selection matrix Ω_m with

$$\Omega_m = R^d_o \cdot S_n \cdot R^o_d\,, \tag{8.25}$$

in which R^d_o is the transformation between the task frame O_d and the robot base frame O_o. In this case, we constrain the motion only in the n (z_d) direction, and the motion matrix S_n can be chosen as

$$S_n = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\,. \tag{8.26}$$

Step 2: After the robot has reached the plane, we rotate the end-effector until the z_d axis is perpendicular to Plane$_A$ (or $z_d//n$, which means angle (α) between z_d and n converges to zero) with

$$\ddot{x}^*_{\text{orientation}} = \Omega_m(k_p\,\theta - k_d\,\omega)\,, \tag{8.27}$$

where ω is the robot angular velocity, $\theta = (0, 0, \alpha)^{\mathrm{T}}$ and α is the angle between the robot end-effector's z_d axis and the plane normal n.

These steps are demonstrated in Fig. 8.4(1) (d). Fig. 8.4(1) (b) and (c) show the simulation results, with the robot trajectory in the Cartesian space and the control parameters d, α. Both task errors converge to zero smoothly. In step 2, the position constraint is at a higher priority than the orientation. Due to the projection of the orientation task into the null-space of the position constraint, the orientation task does not affect the position task. Fig. 8.4(1) (d) shows that the position has not changed during the orientation task in step 2.

8.4.2 Move on a Plane

In this case, we constrain the robot motion only on the plane with the robot end-effector perpendicular to the plane. Suppose we have a desired position X_d that does not lie on the plane, see Fig. 8.4(2) (d). The robot is commanded to reach the target position X_d, while staying on the plane ($x_d - y_d$ plane) and keeping the end-effector z_d axis parallel to normal n.

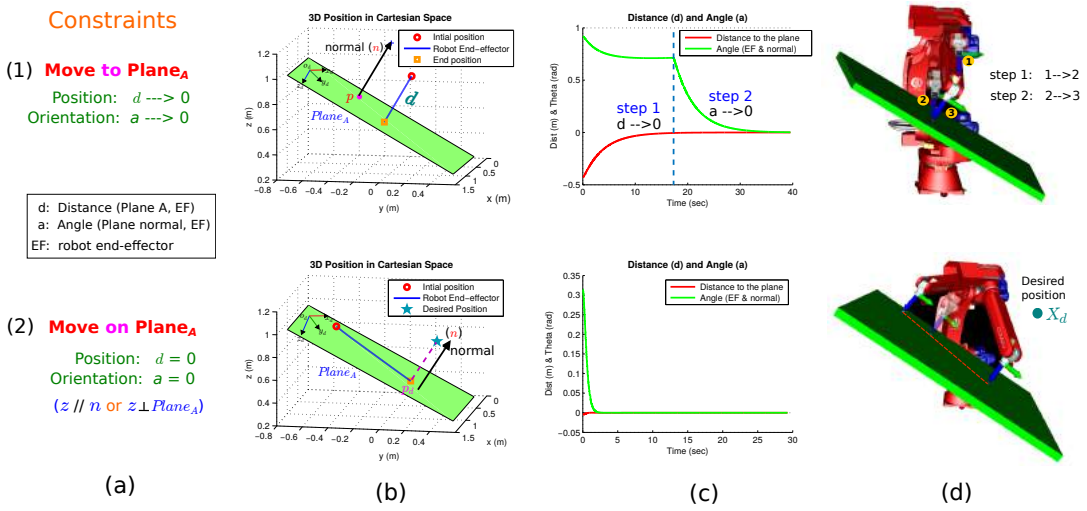


Figure 8.4: Motion constraints tests for robot manipulators in two cases: (1) Move to a plane; (2) move on a plane. (a) Constraints for tests, (b) 3D position in the task space, (c) control parameters, (d) 3D visualization of coach for simulations.

The control law is

$$\ddot{x}^*_{\text{position}} = \Omega_m \cdot (\ddot{x}^*_d), \tag{8.28}$$

where $\ddot{x}^*_d$ is the desired input to reach the target position X_d, which is computed by using the proposed visual servoing (6DVS). The selection matrix is

$$\Omega_m = R^d_o \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} R^o_d \,. \tag{8.29}$$

In this task, the control of the position constraint is integrated by using the top-most priority level, while the secondary task is the orientation constraint. The orientation task is projected into the null space of the position constraint with

$$\tau = \tau_{\text{position}} + N^{\text{T}}_{\text{position}} \cdot \tau_{\text{orientation}} \,, \tag{8.30}$$

where N_{position} is the constraint null-space matrix.

Fig. 8.4 (2) shows the results for this test. During the task execution, the distance d and the angle α remain zero since the robot is constrained to move on the plane and keeps the orientation of the end-effector (Fig. 8.4(2) (c). In this test, since the task position X_d is out of the plane, the robot moves on the plane linearly until he reaches the projection point p_d, which is the projection point of the task position X_d, see Fig. 8.4(2) (b) and (d).

8.5 Hybrid Control

In real applications, an accurate model of the environment is difficult to obtain, e.g., the Plane$_A$ in the *move to a plane* case cannot be known as model $(point, normal)$. Therefore, we require a force sensor and a dynamic contact to decide where the plane is located and what its normal direction n is. In this case, we use hybrid control to accomplish the task *move to a plane*.

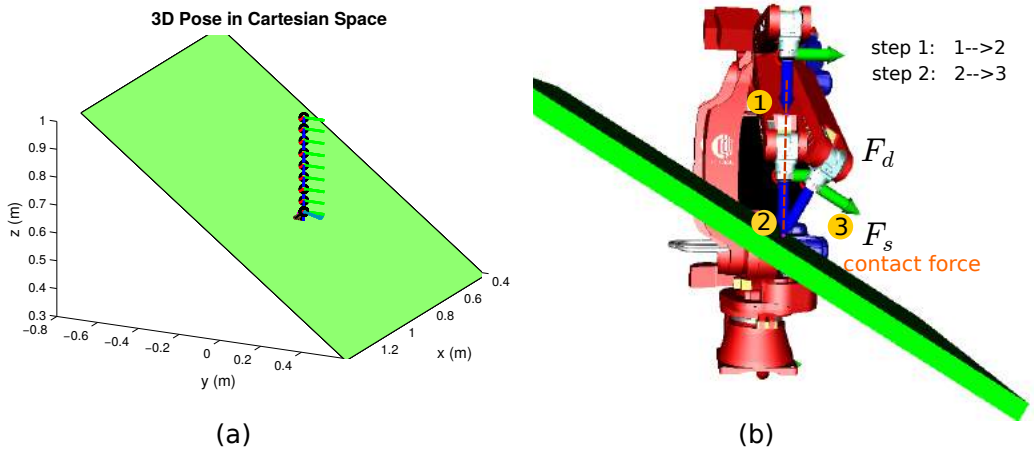


Figure 8.5: Hybrid control: (a) 3D pose for the robot end-effector. (b) Two steps for the motion, including the desired force and the contact force from the force sensor.

Step 1: We apply a desired force F_{desired} to command the robot to move while the motion is constrained in the x and y axes. All constraints are combined together to guide the robot motion solely along the z axis (Fig. 8.5) until it reaches a plane, where the contact force F_{sensor} is provided by the force sensor. The control law is

$$F_{p(i)} = \Omega_f f_i^* + \Lambda_{p(i)}\{\Omega_m \ddot{x}_i^* - \dot{J}_i \dot{q} + J_i M^{-1}(h - \sum_{j=1}^{i-1} \tau_{p(j)})\}\,, \tag{8.31}$$

with the selection matrices as

$$\Omega_m = R_o^d \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_d^o \tag{8.32}$$

and

$$\Omega_f = I - \Omega_m = R_o^d \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} R_d^o\,. \tag{8.33}$$

The force control is defined as

$$f_i^* = k_s(F_{\text{desired}} - F_{\text{sensor}})\,, \tag{8.34}$$

where k_s is the contact stiffness. Finally, the robot stays on the plane when $\Omega_f f_i^* = 0$.

Step 2: After the robot reaches the plane, we rotate the end-effector until the contact force F_{sensor} only has the z axis component, which means that the z axis of the end-effector is perpendicular to the plane.

The results and motion are depicted in Fig. 8.5.

8.6 Real-world Applications

In this section, we evaluate our approach on a selection of classical industrial robotics scenarios in real-world setups (using a 6-DOF industrial robot). We choose several typical industrial robotic applications for demonstrating and evaluating the proposed approach: grasping of cylindrical objects, welding and two force control applications (Erasing and Peg-in-Hole).

8.6.1 Grasping of Cylindrical Objects

In this scenario, an industrial manipulator is supposed to grasp a cylindrical object at its rim using a parallel gripper (Fig. 8.6). The robot end-effector is commanded to grasp the a cylindrical object at any point along the object's rim (a valid grasping pose). The orientation of the gripper is adjusted in a way that it is tangential to the cylinder's rim.

This task can be defined with the following constraints:

- **Line-Plane Coincident Constraint:** Plane_B, which contains Axis_{B_1} and Axis_{B_2}, coincident with Axis_A.
- **Line-Point Distance Constraint:** Point_B, which is the point of intersection of Axis_{B_1} and Axis_{B_2}, at a distance (Cylinder_A) from Axis_A.
- **Plane-Point Distance Constraint:** Point_B is at a distance zero from Plane_A, which is the top plane of the object.
- **Orientation Constraint:** Axis_A is parallel to Axis_{B_1} and Axis_A is perpendicular to Axis_{B_2} of the gripper.

While the above constraints need to be fulfilled exactly, a residual degree of freedom is available as a path along the rim of the cylinder (Fig. 8.6 (b)). We implemented this scenario in simulation and on an industrial robot platform (Fig. 8.6 (c) and (d)), where we solve the constraints to obtain the target pose closest to the previous waypoint of the robot.

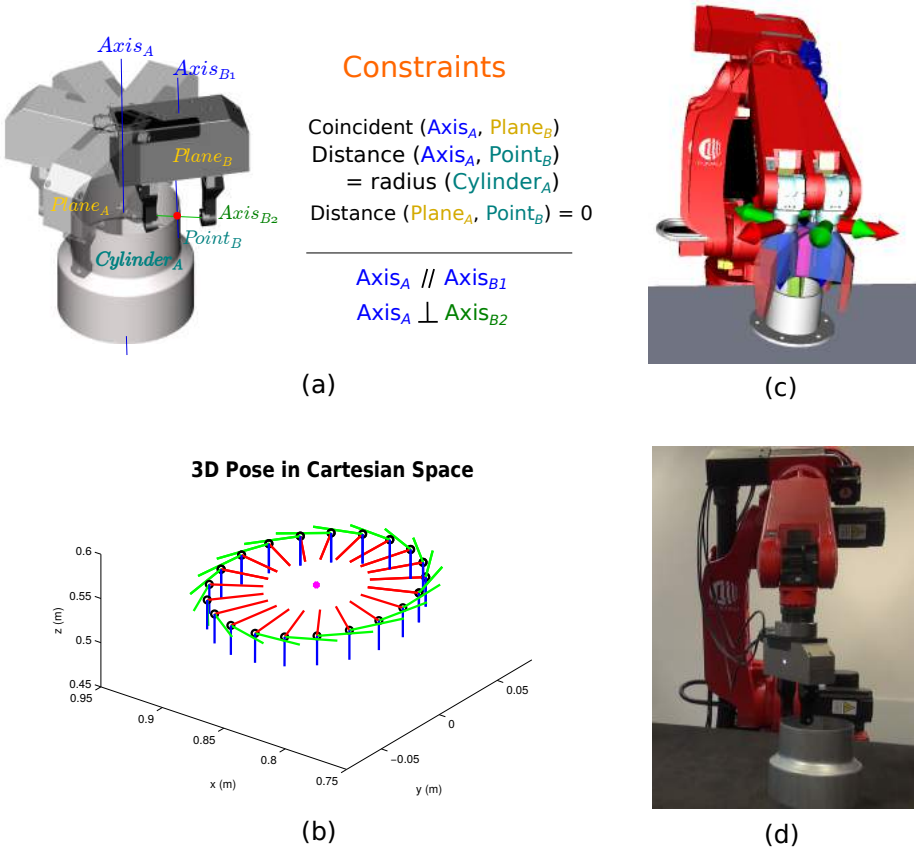


Figure 8.6: Grasping of cylindrical objects at their rim in a robotic workcell: (a) Task constraints. (b) 3D pose in Cartesian space, shows the end-effector trajectory. (c) and (d) illustrate the snapshots in simulation and real experiment.

8.6.2 Welding

For the welding application, we test two cases: point welding and seam welding.

In the point welding scenario (Fig. 8.7 (b)), the robot is supposed to weld an object at a user-specified point. This task fixes the position of the welding tool-tip. However, its orientation is not fixed and can be optimized during runtime. The tip of the welding gun must exactly coincide with the target point on the object with the position constraint:

- **Point-Point Coincident Constraint:** Point_A of workpiece is coincident with Point_B of welding gun tool (Fig. 8.7 (a)).

In this example, the orientation of the welding gun should be adjusted by an operator.

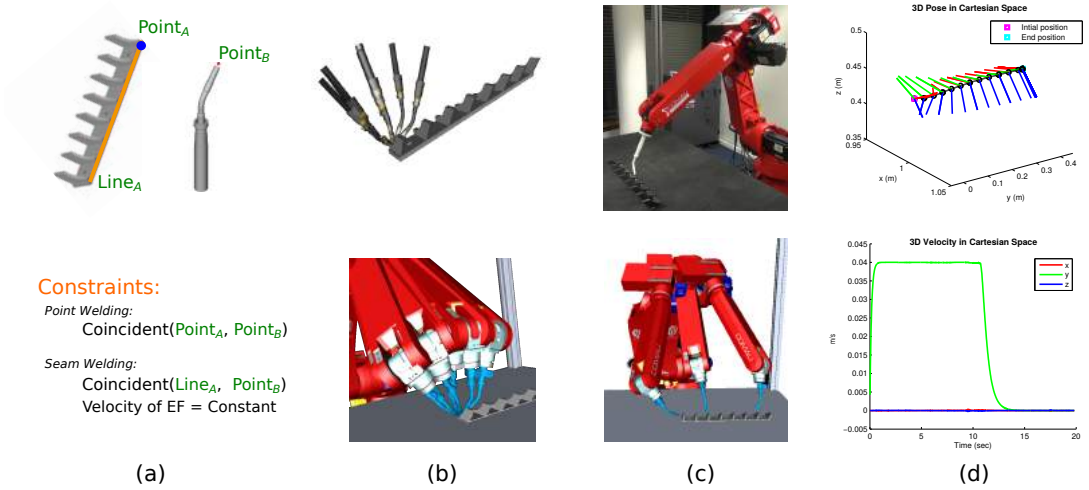


Figure 8.7: Welding in a robotic workcell: (a) Task constraints, (b) point welding, (c) seam welding in simulation and real experiment, (d) 3D end-effector trajectory and velocities for robot end-effector.

In the seam welding scenario (Fig. 8.7 (c)), the tip of the welding gun must lie on the target line on the object. The task constraints are

- **Point-Line Coincident Constraint:** Line$_A$ of workpiece is coincident with Point$_B$ of welding gun tool (Fig. 8.7 (a)).

In this simplified experiment, we add one more constraint with constant velocities for the robot movement. This velocity constraint is applied in the null space of the position constraint, which means that the motion along the line is always satisfied. Fig. 8.7 (d) illustrates the pose of the seam welding and the constant velocities of 0.04 m/s. Moreover, the robot can be jogged in the null-space to choose an orientation as required by other constraints from the workcell.

8.6.3 Erasing with Constant Force

For the erasing board application, it can be divided in 2 steps: step 1 in which the robot is move to a plane in z direction, and step 2 in which the gripper is sliding on the board with constant forces (erasing the board with given trajectory). Step 1 *Move to a Plane* is the same as presented in subsection 8.4.1. Then the step 2 erasing task can be defined with the following constraints:

- **Plane-Point Distance Constraint:** end-effector is at a distance zero from the board.

- **Orientation Constraint:** EF_z is perpendicular to the board, with fixed orientation for robot end-effector.
- **Force Constraint:** the contact force F_{sensor} should be equal to the given desired force F_{desired} along the z axis.

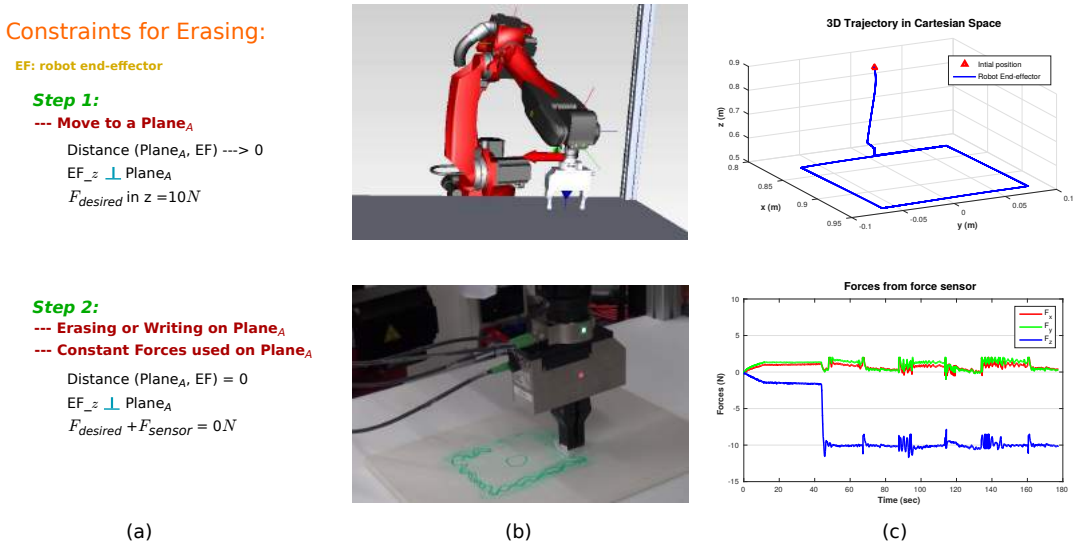


Figure 8.8: Erasing the board: (a) Task constraints, (b) Erasing task in simulation and real experiment, (c) 3D end-effector trajectory in Cartesian space and contact force from force sensor.

In this task, the hybrid control is used to guide the robot motion, where the contact force F_{sensor} is provided by the force sensor, and the $F_{\text{desired}} = 10N$. The control law is presented in equation (8.9) and the force control is defined in (8.34). When $f = (F_{\text{sensor}} + F_{\text{desired}})$ is close to $0N$, the robot is maintained on the board, where the erasing task is performed.

Fig. 8.8 (a) lists the constraints for this task and Fig. 8.8 (b) shows the scenario where the erasing task is evaluated in simulation and on a real industrial manipulator. A desired trajectory with rectangle shape is given for this task. Fig. 8.8 (c) demonstrates the experiment results in terms of the 3D trajectory in Cartesian space and the contact forces. From the force plot, it can be seen clear 2 steps ($t \approx 46s$). The forces for both x and y axes is close to zero since the robot end-effector is perpendicular to the board. However, there are still some noise and chattering due to different velocities of the robot and unsteady noise of the force sensor. Furthermore, from Fig. 8.8 (c), it can be observed that the contact force in z direction, $F_z \approx -10N$ since the desired force $F_{\text{desired}} = 10N$ and force error should be close to zero during erasing task.

8.6.4 Peg-in-Hole

In typical industrial robotics applications, the robotic peg-in-hole assembly task is a force control application, which is composed of three phases: phase 1 in which the robot is move to a surface where the hole is located, phase 2 in which the peg is sliding on the surface of the environment and searching for the hole, and phase 3 in which the peg is located straight on the hole (Peg-in-Hole phase). Different constraints should be satisfied during different phases, which are listed in Fig. 8.9 (a). During these constraints, the force constraint is the crucial one.

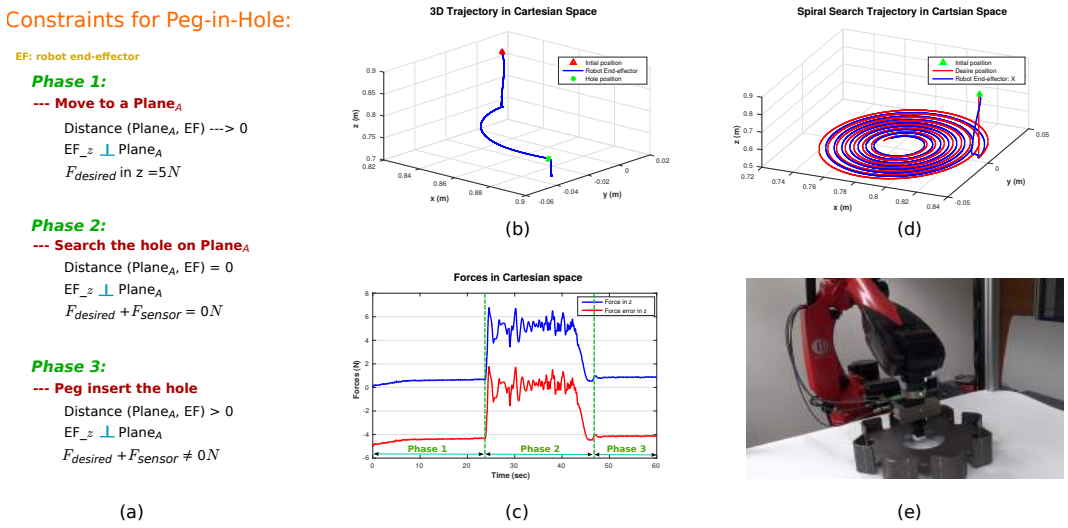


Figure 8.9: Peg-in-Hole assembly tasks: (a) Task constraints, (b) 3D trajectory in Cartesian space, (c) Force and force error in z axis, (d) Spiral search trajectory, (e) Real experiment.

Fig. 8.9 illustrates the experiment on a real industrial manipulator, where Fig. 8.9 (b-d) is the evaluation results and the (e) is a snapshot of the real experiment. Fig. 8.9 (b) is the 3D trajectory in Cartesian space for robot end-effector, where 3 phases are observed: moving down along z, searching and insert the hole. The contact force F_{sensor} which is provided by the force sensor and the force error f along z axis are demonstrated in Fig. 8.9 (c). During phases 1 and 3, the contract force is close to zero since there is no contract in z-direction. There is about $5N$ contact force in z-direction during the searching phase due to $F_{\text{desired}} = -5N$. Moreover, Fig. 8.9 (d) shows an example of 3D trajectory for searching phase. The spiral searching algorithm is used to search the hole.

8.7 Discussion

In this chapter, we presented a prioritized, multiple-task control framework that allows for force control in industrial robotics. The proposed framework is based on a composable structure where several constraints, each describing a robot task or constraint, can be combined with a priority. The priorities are accomplished by null-space projections. The robot must accomplish a global task while satisfying several constraints, where the global task is achieved using the proposed visual servoing (6DVS) while safety constraints are modeled by the potential field techniques. We tested several types of constraints, e.g., operational task (position and orientation), force (contacts), or inequality constraints (joint limits). Moreover, several typical industrial robotics applications were chosen for demonstrating and evaluating the proposed approach. We also have evaluated the proposed framework with force control applications, where the force control is a crucial aspect, such as inserting a peg into a hole, erasing the board and writing.

The constraints supported by our framework are more suitable for industrial tasks. Hence, the complexity of geometric constraints required is higher than that used in WBCF [175]. In [174, 184], even more complicated geometric constraints for industrial tasks were presented. However, these frameworks are designed to work only at the position control level, which is not suited for robots that interact with the environment. In conclusion, we use the control framework from [175], but introduce more complicated geometric constraints and adapt it for use in industrial applications.

Chapter 9

Conclusion

The individual chapters in this thesis provided a detailed insight into the research and development of a vision-based controller for manipulators in human-robot interaction scenarios. Novel techniques in extracting new image features and modeling new visual servoing control law were introduced and validated. Furthermore the possibilities of easy integration with a variety of robotic systems in human-robot interaction scenarios were demonstrated in experiments and applications.

This chapter summarizes the primary contributions of the thesis, and the resulting conclusions drawn from this work. These conclusions also include the observed shortcomings in the system. With respect to these observations, the future work in order to overcome the current shortcomings is discussed.

9.1 Primary Contributions of the Thesis

The primary contribution of this thesis, as also discussed in the earlier chapters, can be summarized as follows:

1. ***Visual Servoing:***

- **Improvement for PBVS in terms of Visual Occlusion**: Two different camera models were presented in chapter 4 to provide two different solutions for visual occlusion problems in PBVS such as out of camera field of view and visual occlusion by unstructured environments.

- **New Orthogonal Image Features for IBVS**: The choice of features directly influences on the performance of the controller. A new *virtual visual space* was designed where a

6D pixel pose vector (with 6 orthogonal features) can be extracted. The virtual visual space is obtained by first computing the 3D position of points in the metric camera frame using a stereo vision system. Then they are re-projected into two virtual camera frames. Hence, the extracted 6D pixel vector (W_s) represents the image position as 6 linearly independent and orthogonal signals, which are used as inputs for visual servoing instead of the classical image features.

- **Full-rank block diagonal Image Jacobian**: Points from the image plane are mapped to the new 6 orthogonal image features, to obtain a mapping (a 6×6 *image Jacobian*). This mapping maps velocities from Cartesian space (velocity of the robot end-effector) to the virtual visual space (velocity of selected image features). It is broken down to pure rotations and translations. Moreover, this full rank image Jacobian can avoid the well-known problems such as the image space singularities and local minima.

- **Novel Properties of the proposed Scheme**: The proposed Jacobian is used to design a new visual servoing scheme (6DVS), which was compared with several classical VS methods. The evaluation results demonstrated the novel properties and better performance of the proposed 6DVS algorithm over conventional VS approaches. It combines the advantages of PBVS and IBVS, and decouples the control of the translational and rotational motion of robot end-effector due to the diagonal image Jacobian. Furthermore, an observation from the analysis shows that errors in the virtual visual space are proportional to the error in the Cartesian space.

- **Integration into real-world Robotic Applications**: The proposed 6DVS scheme was integrated into a variety of diverse real-world robotic application scenarios. Its easy integration into different robotics system confirms its flexibility in terms of usability.

2. ***Constraint-based robot control:***

- **Environment Constraints**: The proposed scheme was experimentally shown to be easy to integrate with the unstructured environment constraints in a human-robot interaction scenario, where the environmental constraints are modeled using artificial potential fields and geometric constraints.

- **Prioritized multi-task Control Framework**: To guarantee accomplishment of the global task while satisfying several constraints, a prioritized, multi-task hierarchical con-

trol framework was designed. It allows force control and complicated geometric constraints for industrial tasks. It establishes multiple priorities among tasks. The framework supports three priority levels of constraints, safety constraints, operational tasks and posture optimization.

- **Typical industrial Robotic Applications**: Several typical industrial robotic applications, such as grasping of cylindrical objects, welding, erasing and Peg-in-Hole, were chosen for demonstrating and evaluating the proposed prioritized multi-task control framework with different constraints.

9.2 Shortcomings of the System

Although the proposed visual servoing performs robustly in unstructured environments, there are few aspects which need further optimization. This section explains these shortcomings, which shall form the basis for planning future research directions in this area.

- **Algorithm Classification**: In chapter 5, we categorize our scheme as an image-based visual servoing due to the nature of terms in the error function. As we know, in IBVS, the features used as feedback in minimizing the error are from the image space (normally measured in pixels), while PBVS uses the geometric interpretation of the information in Cartesian space (such as 3D position, measured in meter). In our case, the general idea behind our algorithm is that we try to extract 6 independent features (W_s) (pixels) from the classical image features (s) in order to get a square full-rank Image Jacobian. These new image features (in pixels) are used to design the error function to control the robot. However, as an intermediate step, recovery of the 3D Cartesian position from the stereo cameras is needed to get the new image features. This intermediate steps creates some confusion for classifying the propose visual servoing. The presented approach for finding this mapping is not the only possible way, and remains a topic for future research.

- **Orientation Estimation for uncalibrated VS**: The current method for estimating the orientation has stability issues and sometimes can result in large errors due to noise from the input visual signals. The estimation method presented in subsection 4.3.3 highly depends on the datasets generated by the vision system. Therefore, using a more stable visual tracking algorithm to improve the estimation accuracy is an important aspect for further investigation.

- **Control Schemes**: An adaptive dynamic controller, using second order sliding mode control, was chosen to control the manipulator since it allows dealing with uncertainties from robot and camera model. However, the choice of a controller was not considered as a major contribution in this thesis, since the focus was not on designing a new controller.

- **Comparison**: Although the proposed 6DVS was compared with some classical visual servoing methods in several standard tests, more tests with different situations need to be carried out. Moreover, the simulations used for comparisons should also be performed on the real robot.

- **Flexibility**: Another shortcoming of the system is the flexibility of the proposed 6DVS. Although the scheme was used in some applications, there is still a lot of scope for combining it with other modules and use in various other applications, e.g. visual navigation, objects grasping.

9.3 Proposed Future Work

From the conclusions drawn and the observed shortcomings, the future work in order to improve the performance of the system in certain aspects will be discussed in this section. The following are the planned areas for future research:

- **Algorithm Classification**: In the chapter 5, we construct a virtual visual space which has similar properties as Cartesian space (has 6 independent components), but measured in pixels. This virtual visual space is an extension of 2D image plane with z pixel direction. Therefore, we still consider it as IBVS. As future work, we can try to find another method without using 3D reconstruction information in the intermediate step, which will eliminate the confusion for algorithm classification.

- **Orientation Estimation for uncalibrated VS**: In order to improve the performance of the orientation estimation, the vision tracking algorithm will be modified such that it is more robust to signal noise.

- **Control Schemes**: This module will be researched further in order to compare the performance with different control schemes for proposed VS approach. Moreover, modeling new visual servoing control schemes to enhance the task performance and stability will be also considered.

- **Integration with more advanced vision systems**: For the camera model in chapter 5, two orthogonal virtual cameras are used to extract the new image features. This concept of the virtual visual space provides the possibilities to integrate the proposed visual servoing with different kinds of vision modules. Based on the formulation of the virtual composite camera model presented in section 5.2.3, 3D Cartesian poses obtained by any visual tracking method also can be mapped to the *virtual visual space* and the new 6 orthogonal components can be extracted as image features. Thus, use of the proposed object tracking method based on stereo cameras to obtain the 3D Cartesian position is not necessary, and can be replaced by more powerful and accurate methods for object detection and pose estimation [168, 165]. Other sensors, e.g., high frame rate cameras, depth cameras, RGBD sensors, ToF sensors, and the Leap Motion sensor can also be used to increase the accuracy of visual tracking and also facilitate many new applications.

- **Extension to different fields**: Potential applications of visual servoing are numerous. It is applicable to any task that uses visual information as input and controls the motion of a dynamic system. Given the capability of easy integration with different modules, the proposed visual servoing algorithm will be extended from the current human-robot interaction for industrial robot to a variety of different research fields and domains, such as, used in visual navigation for a mobile robot or an unmanned aerial vehicle, manipulation and grasping using a robot arm, locomotion and manipulation with a humanoid robot and precise surgery in medical robotics.

- **Constrained Problems**: This module is an important component in enabling the safe and efficient use of robotic systems in real-world applications. Given the wide variety of robotic tasks and constraints in unstructured environments, different control formulations and control strategies will be investigated.

Appendix A

Color-based Object Tracking

A.1 Image Processing

Visual Servoing is a technique which uses feedback information extracted from a vision sensor (visual feedback) to control the motion of a robot. Therefore, extraction the visual features from an image is an important step for visual servoing. In this thesis, object extraction and pose estimation is not our contribution and concern. Hence, we use an existed simple algorithm (color-based objects tracking) for testing our proposed visual servoing system. Fig. A.1 illustrates the algorithm for detection and tracking the centroid position of colored targets[1].

The algorithm segments the target objects based on their color. First, a color histogram of a mask that belongs to the target is computed off-line and used as a model. Then the back projection of a histogram is calculated and morphological operations is used for filtering and noise removal. After finding the contours in a binary image and calculating the minimal upright bounding rectangle for the specified point set, the CamShift object tracking algorithm [185, 186] is used to find the object center, size, and orientation. Outputs of the algorithm, as shown in Fig. A.2, give the pixel positions of the centroids of the detected objects.

[1]Thanks to colleague Dario Mendoza Gallegos for contributions to the implementation of this algorithm.

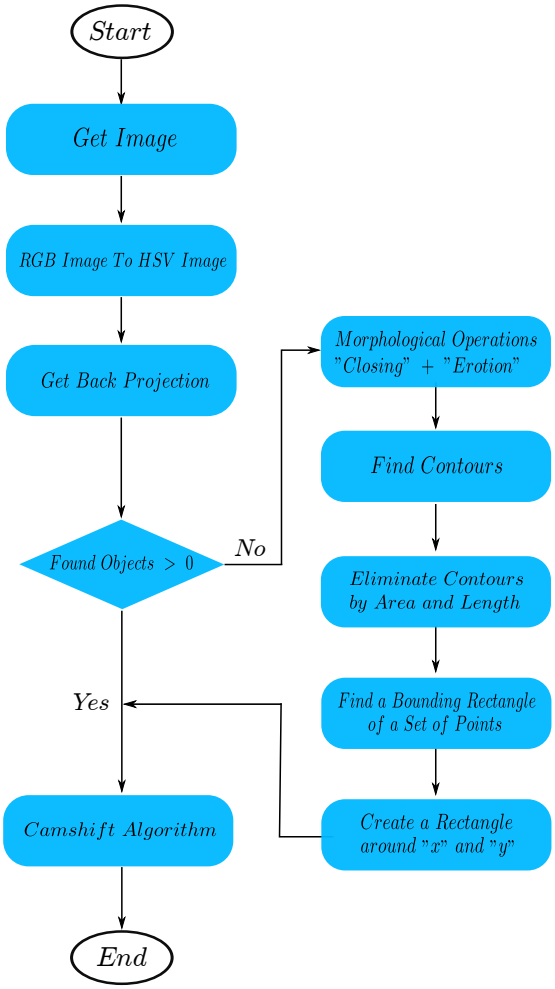


Figure A.1: Color-based objects tracking algorithm.

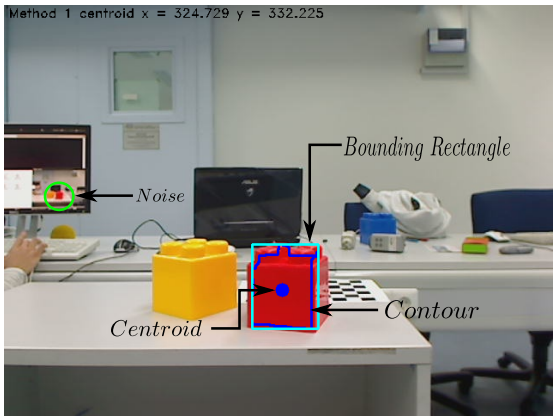


Figure A.2: The results of color-based objects detection.

Appendix B

3D Position-based Visual Servoing Experiments

A 3D PBVS approach based on the two novel camera models proposed in chapter 4 has been tested on a standard industrial robot in a Human-Robot Interaction (HRI) scenario. The real time color-based visual tracker is used to identify the target position (green cube) and the current position of robot end-effector (red cube). The target is carried by a user, and the robot end-effector tracks the target in a dynamic environment. Several robot attributes such as singularity avoidance, self-collision and obstacle avoidance are implemented to ensure safety of the robot and human. The environment constraints are modeled in 7.1.

Two different cameras configurations for tracking system as proposed in chapter 4 have been tested in the experiments, which provide some solutions for the visual occlusion problem for PBVS. Fig. B.1 (a) shows the two cameras system: four orthogonal cameras and two stereo cameras.

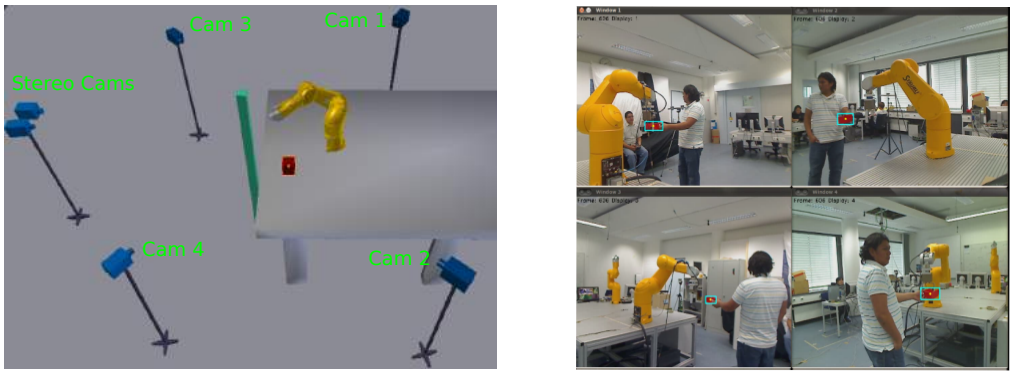


Figure B.1: Vision system with camera indexes.

B.1 3D PBVS with Four Orthogonal Cameras

In this experiment, four orthogonal camera system is used for tracking the target. The visual models are presented in Section 4.2, and can be used to avoid visual occlusion problems. Fig. B.1 (a) shows IDs (index) of four orthogonal cameras and (b) shows the snapshot of camera images.

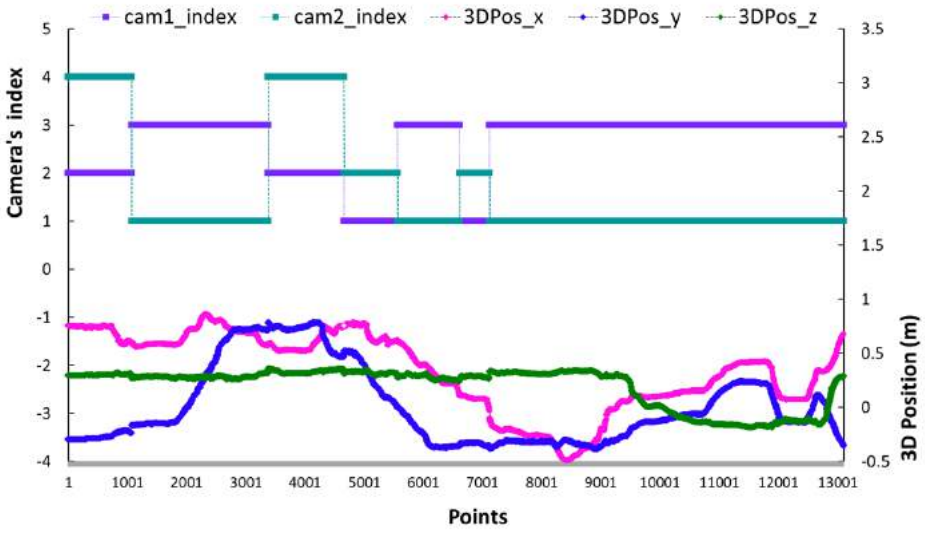


Figure B.2: Orthogonal experiment result: up part is the selected camera IDs and down part shows the 3D Cartesian position for the target.

During experiments, some of the cameras may be occluded by the user or obstacles. Then the system automatically selects two cameras that avoid visual occlusions. The two selected cameras have to be in orthogonal positions in order to get the 3D position. Although the selected cameras keep changing, the model for computing 3D position is the same, as defined in (4.29) and (4.30). The experimental process and the results are shown in Fig. B.2, where two selected cameras' IDs and the corresponding Cartesian position of the object are illustrated. It demonstrates that although the selected cameras keep changing, the Cartesian position is still continuous and smooth due to the precise camera calibration.

During the experimental validation, several behaviors are evaluated. Fig. B.3 (a) shows how the robot handles self-collisions. The system generates a collision-free trajectory (red line) instead of moving directly to the target. Fig. B.3 (b) demonstrates obstacle avoidance. The blue line is the target motion and the red line is the robot end-effector motion. The robot can avoid the obstacle while continuing to track the target.

For human avoidance, we use a Kinect sensor to detect and track the human skeleton. We

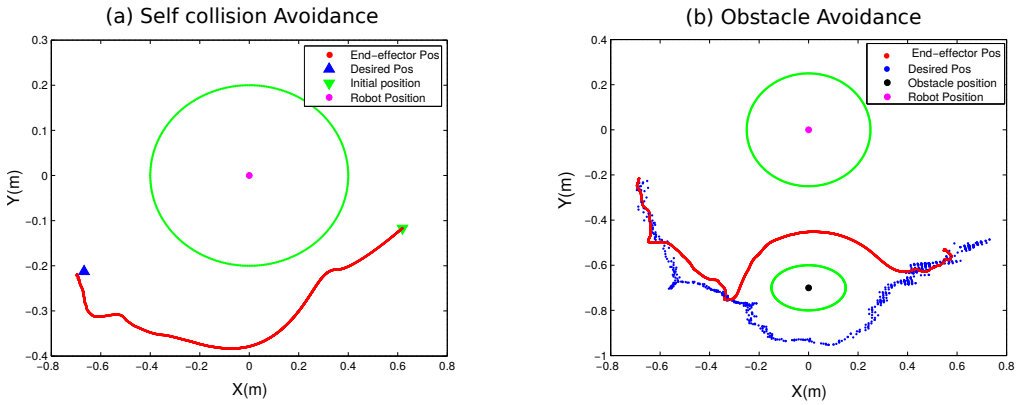


Figure B.3: Experiment results: (a) Self-collision avoidance. (b) Obstacles avoidance.

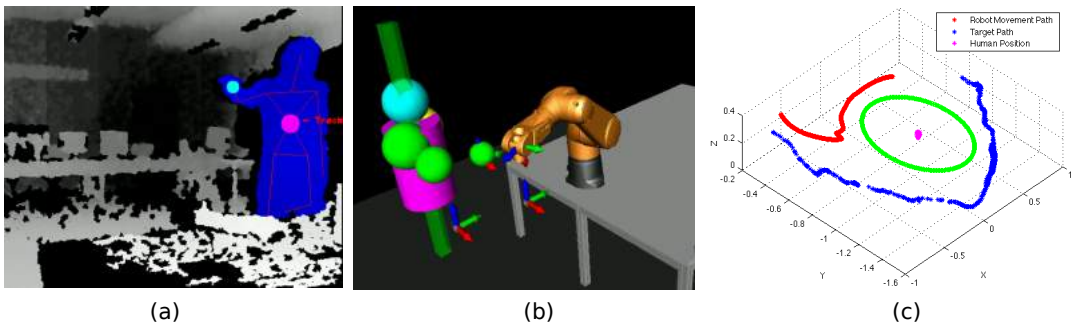


Figure B.4: Avoid Human: (a) Tracking human skeleton using Kinect; (b) Human joints model; (c) Human avoidance results: robot movement (red line) and target movement(blue line), Human position (green circle).

use this information to build our human model, as shown in Fig. B.4 (a) and (b). Fig. B.4 (c) shows the process of how the robot avoids the obstacle (human) while continuing to track the target.

B.2 3D PBVS with uncalibrated Camera System

In this case, two stereo cameras fixed on a tripod are used in tracking system instead of four orthogonal cameras, as shown in Fig. B.3 (a). The stereo vision tracking system provides the positions of both red and green cubes with respect to the stereo coordinate frame. During the experimental validation, several behaviors are evaluated. These behaviors are depicted Fig. B.5.

Fig. B.5 (f) and (a) show visual tracking with and without obstacle avoidance respectively. Fig. B.5 (c) illustrates the result of singularity avoidance, where the robot does not reach the

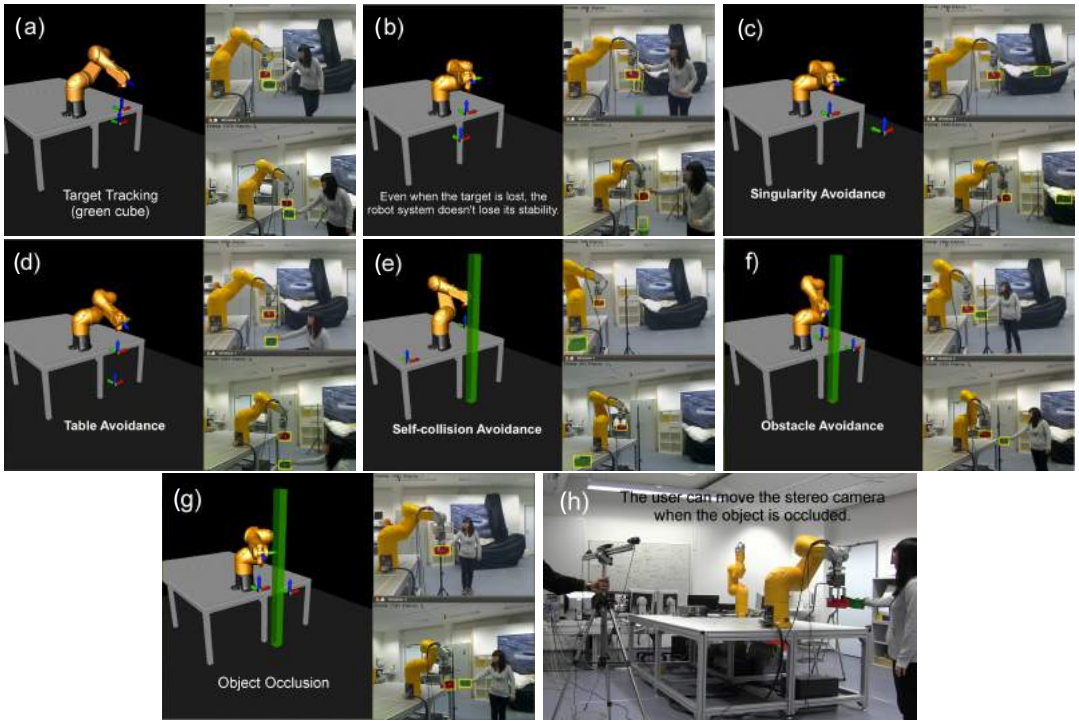


Figure B.5: System behaviors: (a) 3D Position tracking. (b) Case when the target is lost. (c) Case with singularity avoidance. (d) Table avoidance. (e) Self-collision avoidance. (f) Obstacle avoidance. (g) Case when the object is occluded. (h) The camera can be moved when the target is occluded.

singular condition ($q_3 = 0$), even when the user tries to force it. Fig. B.5 (e) shows self collision avoidance. Fig. B.5 (d) depicts table avoidance where the motion of the robot in the $z - axis$ is constrained by the height of the table (the end-effector is not allowed to go under the table). The system proves to be stable and safe for HRI scenarios, even in situations where the target is lost (due to occlusions by the robot or the human), see Fig. B.5 (b) and (g).

The primary advantage of our system is that when the target object is occluded (Fig. B.5 (g)), the stereo system can be moved to maintain the targets in the field of view, see Fig. B.5 (h). After moving the stereo system, our approach can use real data to estimate the orientation matrix on-line without needing to stop the robot and re-calibrate it. More details about the on-line orientation estimation is shown in 7.3.2.3

References

[1] Caixia Cai, Nikhil Somani, Suraj Nair, Dario Mendoza, and Alois Knoll. **Uncalibrated Stereo Visual Servoing for Manipulators using Virtual Impedance Control**. In *the 13th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, December 2014. 6

[2] Caixia Cai, Emmanuel Dean-Leon, Dario Mendoza, Nikhil Somani, and Alois Knoll. **Uncalibrated 3D Stereo Image-based Dynamic Visual Servoing for Robot Manipulators**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013. 6, 79

[3] Caixia Cai, Emmanuel Dean-Leon, Nikhil Somani, and Alois Knoll. **3D Image-based Dynamic Visual Servoing with Uncalibrated Stereo Cameras**. In *The 44th Internation Symposium on Robotics (ISR)*, October 2013. 6, 7

[4] Caixia Cai, Emmanuel Dean-Leon, Nikhil Somani, and Alois Knoll. **6D Image-based Visual Servoing for Robot Manipulators with Uncalibrated Stereo Cameras**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2014. 6, 7

[5] Caixia Cai, Nikhil Somani, and Alois Knoll. **Orthogonal Image Features for Visual Servoing of a 6-DOF Manipulator with Uncalibrated Stereo Cameras**. *IEEE transactions on Robotics*, April 2016. 6

[6] Caixia Cai, Nikhil Somani, Markus Rickert, and Alois Knoll. **Prioritized Motion-Force Control of Multi-Constraints for Industrial Manipulators**. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Zhuhai, China, December 2015. 7

[7] L.E. Weiss, A.C. Sanderson, and Charles P. Neuman. **Dynamic sensor-based control of robots with visual feedback**. *IEEE Journal of Robotics and Automation*, **3**(5):404–417, October 1987. 9, 17, 18

[8] J.T. Feddema and Owen R. Mitchell. **Vision-guided servoing with feature-based trajectory generation**. *IEEE Transactions on Robotics and Automation*, **5**(5):691–700, Oct. 1989. 9

[9] S. Hutchinson, G.D. Hager, and P.I. Corke. **A tutorial on visual servo control**. *IEEE Transactions on Robotics and Automation*, **12**(5):651–670, Oct. 1996. 9, 10, 14, 17, 18, 19, 62

[10] F. Chaumette and S. Hutchinson. **Visual servo control I. Basic approaches**. *IEEE Robotics Automation Magazine*, **13**(4):82–90, Dec. 2006. 9, 14, 17, 20, 62, 63, 85

[11] F. Chaumette and S. Hutchinson. **Visual servo control II. Advanced approaches**. *IEEE Robotics Automation Magazine*, **14**(1):109–118, Mar. 2007. 9, 10, 17, 87

[12] B. Espiau, F. Chaumette, and P. Rives. **A new approach to visual servoing in robotics**. *IEEE Transactions on Robotics and Automation*, **8**(3):313–326, June 1992. 9, 12, 17, 18, 24

[13] K. Hashimoto. *Visual Serving: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems. World Scientific Publishing Company Incorporated, 1993. 9

[14] Y. Shirai and H. Inoue. **Guiding a robot by visual feedback in assembling tasks**. *Pattern Recognition*, **5**(2):99–108, 1973. 9

[15] D. Kragic and H.I. Christensen. *Survey on visual servoing for manipulation*. Research report, Centre for Autonomous Systems. Sweden, 2002. 9, 10

[16] A. C. Sanderson and L.E. Weiss. **Image-based visual servo control using relational graph error signals**. In *IEEE International Conference on Cybernetics and Society*, pages 1074–1077, Oct 1980. 9

[17] E. Malis, F. Chaumette, and S. Boudet. **2-1/2 D visual servoing**. *IEEE Transactions on Robotics and Automation*, **15**(2):238–250, Aprl 1999. 9, 12, 21, 24

[18] P. Martinet, N. Daucher, J. Gallice, and M. Dhome. **Robot control using monocular pose estimation**. In *in Proceedings of the Workshop on New Trends in Image Based Robot Servoing, IROS*, pages 1–12, September 1997. 9, 18

[19] W.J. Wilson, C.C. Williams Hulls, and G.S. Bell. **Relative end-effector control using Cartesian position based visual servoing**. *IEEE Transactions on Robotics and Automation*, **12**(5):684–696, Oct. 1996. 9, 12, 17, 18

[20] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell. **Automatic selection of image features for visual servoing of a robot manipulator**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 83–837, May 1989. 9

[21] R. Mahony, P. Corke, and F. Chaumette. **Choice of image features for depth-axis control in image based visual servo control**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 390–395, 2002. 10

[22] R.Y. Tsai and R.K. Lenz. **A new technique for fully autonomous and efficient 3D robotics hand/eye calibration**. *IEEE Transactions on Robotics and Automation*, **5**(3):345–358, Jun. 1989. 10, 19

[23] P.K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. **Automated tracking and grasping of a moving object with a robotic hand-eye system**. *IEEE Transactions on Robotics and Automation*, **9**(2):152–165, Apr. 1993. 11

[24] G. Flandin, F. Chaumette, and E. Marchand. **Eye-in-hand/eye-to-hand cooperation for visual servoing**. In *IEEE International Conference on Robotics and Automation (ICRA)*, **3**, pages 2741–2746, Apr. 2000. 11

[25] R. Mebarki, Alexandre Krupa, and F. Chaumette. **2-D ultrasound probe complete guidance by visual dervoing using image moments**. *IEEE Transactions on Robotics*, **26**(2):296–306, Apr. 2010. 11

[26] J. Okamoto Jr and V. Grassi Jr. **Visual servo control of a mobile robot using omnidirectional vision**. In *Proceedings of Mechatronics*, pages 413–422, Jun. 2002. 11

[27] H. Hadj-Abdelkader, Y. Mezouar, and P. Martinet. **Decoupled visual servoing from a set of points imaged by an omnidirectional camera**. In *IEEE International Conference on Robotics and Automation*, pages 1697–1702, Apr. 2007. 11

[28] F. Chaumette. **Potential problems of stability and convergence in image-based and position-based visual servoing**. In *The Confluence of Vision and Control*, pages 66–78. LNCIS Series, No 237, Springer-Verlag, 1998. 11, 17, 19, 20, 23

[29] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell. **Weighted selection of image features for resolved rate visual feedback control**. *IEEE Transactions on Robotics and Automation*, **7**(1):31–47, Feb. 1991. 11, 12

[30] F. Janabi-Sharifi and W.J. Wilson. **Automatic selection of image features for visual servoing**. *IEEE Transactions on Robotics and Automation*, **13**(6):890–903, Dec. 1997. 11, 12

[31] Jianbo Shi and C. Tomasi. **Good features to track**. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994. 12

[32] N. Andreff, B. Espiau, and R. Horaud. **Visual servoing from lines**. In *IEEE International Conference on Robotics and Automation*, **3**, pages 2070–2075, 2000. 12

[33] M. Iwatsuki and N. Okiyama. **A new formulation of visual servoing based on cylindrical coordinate system**. *IEEE Transactions on Robotics*, **21**(2):266–273, April 2005. 12, 21

[34] F. Chaumette. **A first step toward visual servoing using image moments**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 378–383, 2002. 12

[35] F. Chaumette. **Image moments: a general and useful set of features for visual servoing**. *IEEE Transactions on Robotics*, **20**(4):713–723, Aug. 2004. 12, 20, 63

[36] O. Tahri and F. Chaumette. **Image moments: generic descriptors for decoupled image-based visual servo**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 1185–1190, April 2004. 12

[37] O. Tahri and F. Chaumette. **Point-based and region-based image moments for visual servoing of planar objects**. *IEEE Transactions on Robotics*, **21**(6):1116–1127, Dec 2005. 12, 20

[38] Yimin Zhao, Wen-Fang Xie, and Sining Liu. **Image-based visual servoing using improved image moments in 6-DOF robot systems**. *International Journal of Control, Automation and Systems*, **11**(3):586–596, 2013. 12, 20

[39] O. Tahri and F. Chaumette. **Application of moment invariants to visual servoing**. In *IEEE International Conference on Robotics and Automation*, **3**, pages 4276–4281, Sept 2003. 12

[40] O. Tahri and F. Chaumette. **Complex Objects Pose Estimation based on Image Moment Invariants**. In *IEEE International Conference on Robotics and Automation*, pages 436–441, April 2005. 12

[41] P. Martinet, J. Gallice, and D. Khadraoui. **Vision Based Control Law using 3D Visual Features**. In *Committees, Econometrica*, pages 497–502, 1996. 12

[42] L. Deng, W.J. Wilson, and F. Janabi-Sharifi. **Dynamic performance of the position-based visual servoing method in the Cartesian and image spaces**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 510–515, Oct. 2003. 12, 18, 19, 24

[43] Enrique Cervera, Angel P. Del Pobil, FranÃ§ois Berry, and Philippe Martinet. **Improving Image-Based Visual Servoing with Three-Dimensional Features**. *The International Journal of Robotics Research*, **22**(10-11):821–840, 2003. 12, 21

[44] J. Wang and W.J. Wilson. **3D relative position and orientation estimation using Kalman filter for robot control**. In *IEEE International Conference on Robotics and Automation*, pages 2638–2645, May 1992. 12, 19

[45] G. Hu, N. Gans, N. Fitz-Coy, and W. Dixon. **Adaptive homography-based visual servo tracking control via a quaternion formulation**. *Control Systems Technology, IEEE Transactions on*, **18**(1):128–135, Jan. 2010. 12

[46] L. Deng, F. Janabi-Sharifi, and W.J. Wilson. **Hybrid strategies for image constraints avoidance in visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 348–353, 2002. 12

[47] E. Marchand and F. Chaumette. **Feature tracking for visual servoing purposes**. *Robotics and Autonomous Systems*, **52**(1):53–70, June 2005. 12

[48] P.I. Corke, F. Spindler, and F. Chaumette. **Combining Cartesian and polar coordinates in IBVS**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5962–5967, Oct. 2009. 12

[49] K. Hashimoto, T. Ebine, and H. Kimura. **Visual servoing with hand-eye manipulator-optimal control approach**. *IEEE Transactions on Robotics and Automation*, **12**(5):766–774, Oct. 1996. 13

[50] S.K. Nayar, S.A. Nene, and H. Murase. **Subspace methods for robot vision**. *IEEE Transactions on Robotics and Automation*, **12**(5):750–758, Oct. 1996. 13

[51] Koichiro Deguchi. **A Direct Interpretation of Dynamic Images with Camera and Object Motions for Vision Guided Robot Control**. *International Journal of Computer Vision*, **37**(1):7–20, 2000. 13

[52] V. Kallem, M. Dewan, J.P. Swensen, G.D. Hager, and N.J. Cowan. **Kernel-based visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1975–1980, Oct. 2007. 13

[53] C. Collewet, E. Marchand, and F. Chaumette. **Visual servoing set free from image processing**. In *IEEE International Conference on Robotics and Automation*, pages 81–86, May 2008. 13

[54] C. Collewet and E. Marchand. **Colorimetry-based visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5438–5443, Oct. 2009. 13

[55] C. Collewet and E. Marchand. **Photometry-based visual servoing using light reflexion models**. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 701–706, May 2009. 13

[56] A. Cretual and F. Chaumette. **Visual servoing based on image motion**. *The International Journal of Robotics Research*, **20**(11):857–877, 2001. 13

[57] R. Kelly, J. Moreno, and Ricardo Campa. **Visual servoing of planar robots via velocity fields**. In *IEEE Conference on Decision and Control (CDC)*, **4**, pages 4028–4033 Vol.4, Dec. 2004. 13

[58] R. Kelly, E. Bugarin, and V. Sanchez. **Image-based visual control of nonholonomic mobile robots via velocity fields: Case of partially calibrated inclined camera**. In *IEEE Conference on Decision and Control*, pages 3071–3076, Dec. 2006. 14

[59] E. Dombre and W. Khalil. ISTE, Cambridge, Massachusetts, 2010. 14, 15

[60] Mark W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2006. 15

[61] R.T. Fomena and F. Chaumette. **Improvements on visual servoing from spherical targets using a spherical projection model**. *IEEE Transactions on Robotics*, **25**(4):874–886, Aug. 2009. 15

[62] B. K. Horn. *Robot Vision*. McGraw-Hill Higher Education, 1st edition, 1986. 15

[63] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Pearson, 2nd edition, 2012. 15

[64] L. Deng, W.J. Wilson, and F. Janabi-Sharifi. **Characteristics of robot visual servoing methods and target model estimation**. In *IEEE International Symposium on Intelligent Control*, pages 684–689, Oct. 2002. 17

[65] E. Malis. **Visual servoing invariant to changes in camera-intrinsic parameters**. *IEEE Transactions on Robotics and Automation*, **20**(1):72–81, Feb. 2004. 17, 24

[66] Ronen Basri, Ehud Rivlin, and Ilan Shimshoni. **Visual Homing: Surfing on the Epipoles**. *International Journal of Computer Vision*, **33**(2):117–137, 1999. 18

[67] C.J. Taylor and J.P. Ostrowski. **Robust vision-based pose control**. In *IEEE International Conference on Robotics and Automation*, **3**, pages 2734–2740 vol.3, 2000. 18

[68] V. Kyrki, D. Kragic, and H.I. Christensen. **Measurement errors in visual servoing**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 1861–1867, April 2004. 18, 24

[69] E. Malis and F. Chaumette. **Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods**. *Robotics and Automation, IEEE Transactions on*, **18**(2):176–186, Apr. 2002. 18

[70] P. Martinet and J. Gallice. **Position based visual servoing using a non-linear approach**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 531–536 vol.1, Oct. 1999. 18

[71] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice. **Position based visual servoing: keeping the object in the field of vision**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 1624–1629, 2002. 18

[72] R.M. Haralick, Hyonam Joo, D. Lee, S. Zhuang, V.G. Vaidya, and M.B. Kim. **Pose estimation from corresponding point data**. *IEEE Transactions on Systems, Man and Cybernetics*, **19**(6):1426–1446, Nov. 1989. 19

[73] DanielF. Dementhon and LarryS. Davis. **Model-based object pose in 25 lines of code**. *International Journal of Computer Vision*, **15**(1-2):123–141, 1995. 19

[74] M.L. Liu and K.H. Wong. **Pose estimation using four corresponding points**. *Pattern Recognition Letters*, **20**(1):69 – 74, 1999. 19

[75] A. Ansar and K. Daniilidis. **Linear pose estimation from points or lines**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(5):578–589, May 2003. 19

[76] G. Chesi. **Estimation of the camera pose from image point correspondences through the essential matrix and convex optimization**. In *IEEE International Conference on Robotics and Automation*, pages 35–40, May 2009. 19

[77] F. Dornaika and C. Garcia. **Pose estimation using point and line correspondences**. *Real-Time Imaging*, **5**(3):215 – 230, 1999. 19

[78] J.E. Mcinroy and Z. Qi. **A novel pose estimation algorithm based on points to regions correspondence**. *Journal of Mathematical Imaging and Vision*, **30**(2):195–207, 2008. 19

[79] R. Safaee-Rad, I. Tchoukanov, B. Benhabib, and K.C. Smith. **3D-pose estimation from a quadratic-curved feature in two perspective views**. In *IAPR International Conference on Pattern Recognition*, **1**, pages 341–344, Aug. 1992. 19

[80] E. Marchand and F. Chaumette. **Virtual visual servoing: a framework for real-time augmented reality**. *Computer Graphics Forum*, **21**(3):289–298, Sep. 2002. 19

[81] K. Myers and B.D. Tapley. **Adaptive sequential estimation with unknown noise statistics**. *Automatic Control, IEEE Transactions on*, **21**(4):520–523, Aug. 1976. 19

[82] M. Ficocelli and F. Janabi-Sharifi. **Adaptive filtering for pose estimation in visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 19–24, 2001. 19

[83] V. Lippiello, B. Siciliano, and L. Villani. **Visual motion estimation of 3D objects: an adaptive extended Kalman filter approach**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 957–962 vol.1, Sep. 2004. 19

[84] A. Shademan and F. Janabi-Sharifi. **Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing**. In *IEEE Conference on Control Applications*, pages 755–760, Aug, 2005. 19

[85] F. Janabi-Sharifi and M. Marey. **A kalman-filter-based method for pose estimation in visual servoing**. *IEEE Transactions on Robotics*, **26**(5):939–947, Oct. 2010. 19

[86] H.C. Longuet-Higgins. **A computer algorithm for reconstructing a scene from two projections**. *Nature*, **293**:133–135, Sep. 1981. 19

[87] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993. 19

[88] R.I. Hartley. **In defense of the eight-point algorithm**. *EEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(6):580–593, Jun. 1997. 19

[89] 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*, **37**(1):79–97, 2000. 19, 21

[90] P.I. Corke and S.A. Hutchinson. **A new partitioned approach to image-based visual servo control**. *IEEE Transactions on Robotics and Automation*, **17**(4):507–515, Aug. 2001. 19, 21

[91] Bernard Espiau. **Effect of Camera Calibration Errors on Visual Servoing in Robotics**. In *The 3rd International Symposium on Experimental Robotics III*, pages 182–192, London, UK, 1993. Springer-Verlag. 19

[92] G.D. Hager. **A modular system for robust positioning using feedback from stereo vision**. *IEEE Transactions on Robotics and Automation*, **13**(4):582–595, Aug. 1997. 19

[93] Y. Mezouar and F. Chaumette. **Path planning for robust image-based control**. *IEEE Transactions on Robotics and Automation*, **18**(4):534–549, Aug. 2002. 19, 20, 21, 22, 23, 24

[94] Lingfeng Deng. *Comparison of Image-based and Position-based Robot Visual Servoing Methods and Improvements*. PhD thesis, Waterloo, Ont., Canada, 2004. 19

[95] E. Cervera and P. Martinet. **Visual servoing with indirect image control and a predictable camera trajectory**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 381–386, 1999. 20

[96] E. Malis, Y. Mezouar, and Patrick Rives. **Robustness of Image-Based Visual Servoing With a Calibrated Camera in the Presence of Uncertainties in the Three-Dimensional Structure**. *Robotics, IEEE Transactions on*, **26**(1):112–120, Feb. 2010. 20, 25

[97] Larry Matthies, Takeo Kanade, and Richard Szeliski. **Kalman filter-based algorithms for estimating depth from image sequences**. *International Journal of Computer Vision*, **3**(3):209–238, 1989. 20

[98] L. Marce and P. Bouthemy. **Determination of a depth map from an image sequence**. In *International Conference on Advanced Robotics*, pages 221–232, Oct. 1987. 20

[99] F. Chaumette, S. Boukir, P. Bouthemy, and D. Juvin. **Structure from controlled motion**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(5):492–504, May 1996. 20

[100] A. De Luca, G. Oriolo, and P.R. Giordano. **Feature depth observation for image-based visual servoing: theory and experiments**. *The International Journal of Robotics Research*, **27**(10):1093–1116, 2008. 20

[101] W. Xie, Z. Li, X. Tu, and C. Perron. **Switching control of image-based visual servoing with laser pointer in robotic manufacturing systems**. *IEEE Transactions on Industrial Electronics*, **56**(2):520–529, Feb. 2009. 20, 21, 24

[102] J.T. Feddema, C. S G Lee, and O.R. Mitchell. **Model-based visual feedback control for a hand-eye coordinated robotic system**. *Computer*, **25**(8):21–31, Aug. 1992. 20

[103] Y. Mezouar and F. Chaumette. **Optimal camera trajectory with image-based control**. *The International Journal of Robotics Research*, **22**(10):781–804, 2003. 20, 24

[104] F. Janabi-Sharifi, Lingfeng Deng, and W.J. Wilson. **Comparison of Basic Visual Servoing Methods**. *IEEE/ASME Transactions on Mechatronics*, **16**(5):967–983, Oct. 2011. 20

[105] N.P. Papanikolopoulos and P.K. Khosla. **Adaptive robotic visual tracking: theory and experiments**. *IEEE Transactions on Automatic Control*, **38**(3):429–445, Mar. 1993. 20

[106] Eissa Nematollahi and Farrokh Janabi-Sharifi. **Generalizations to control laws of image-based visual servoing**. *International Journal of Optomechatronics*, **3**(3):167–186, 2009. 20

[107] D. Kim, A.A. Rizzi, G.D. Hager, and D.E. Koditschek. **A robust convergent visual servoing system**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 348–353, Aug. 1995. 20

[108] Yun-Hui Liu, Hesheng Wang, Chengyou Wang, and Kin Kwan Lam. **Uncalibrated visual servoing of robots using a depth-independent interaction matrix**. *IEEE Transactions on Robotics*, **22**(4):804–817, Aug. 2006. 20, 22

[109] Hesheng Wang, Yun-Hui Liu, and Dongxiang Zhou. **Dynamic Visual Tracking for Manipulators Using an Uncalibrated Fixed Camera**. *IEEE Transactions on Robotics*, **23**(3):610–617, June 2007. 20

[110] B.H. Yoshimi and P.K. Allen. **Alignment using an uncalibrated camera system**. *Robotics and Automation, IEEE Transactions on*, **11**(4):516–521, Aug. 1995. 20

[111] K. Hosoda and M. Asada. **Versatile visual servoing without knowledge of true Jacobian**. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, **1**, pages 186–193, Sep. 1994. 20

[112] J.A. Piepmeier, G.V. McMurray, and H. Lipkin. **Uncalibrated dynamic visual servoing**. *IEEE Transactions on Robotics and Automation*, **20**(1):143–147, Feb. 2004. 20

[113] Shademan Azad, Farahmand, Amir-Massoud, and M. Jägersand. **Robust Jacobian estimation for uncalibrated visual servoing**. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5564–5569, May 2010. 20

[114] P. Zanne, G. Morel, and F. Piestan. **Robust vision based 3D trajectory tracking using sliding mode control**. In *IEEE International Conference on Robotics and Automation*, **3**, pages 2088–2093 vol.3, Apr. 2000. 21

[115] P. I. Corke and S.A. Hutchinson. **A new hybrid image-based visual servo control scheme**. In *Proceedings of the 39th IEEE Conference on Decision and Control*, **3**, pages 2521–2526, 2000. 21

[116] V. Kyrki, D. Kragic, and H.I. Christensen. **New shortest-path approaches to visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 349–354 vol.1, Sep. 2004. 21

[117] J. Pages, C. Collewet, F. Chaumette, and J. Salvi. **Optimizing plane-to-plane positioning tasks by image-based visual servoing and structured light**. *IEEE Transactions on Robotics*, **22**(5):1000–1010, Oct 2006. 21

[118] Nicholas Gans, Seth Hutchinson, and Peter Corke. **Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control**. *The International Journal of Robotics Research*, **22**(10/11):955–981, October 2003. 21, 24

[119] O. Khatib. **Real-time obstacle avoidance for manipulators and mobile robots**. In *IEEE International Conference on Robotics and Automation*, **2**, pages 500–505, Mar. 1985. 21

[120] G. Chesi. **Visual servoing path planning via homogeneous forms and LMI optimizations**. *IEEE Transactions on Robotics*, **25**(2):281–291, Apr. 2009. 21

[121] A.I. Comport, E. Marchand, and F. Chaumette. **Statistically robust 2D visual servoing**. *IEEE Transactions on Robotics*, **22**(2):415–420, Apr. 2006. 21

[122] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, and C. Perez-Vidal. **Continuous visual servoing despite the changes of visibility in image features**. *IEEE Transactions on Robotics*, **21**(6):1214–1220, Dec. 2005. 21, 24

[123] N. Mansard, A. Remazeilles, and F. Chaumette. **Continuity of varying-feature-set control laws**. *Automatic Control, IEEE Transactions on*, **54**(11):2493–2505, Nov. 2009. 21

[124] A.H.A. Hafez and C.V. Jawahar. **Probabilistic integration of 2D and 3D cues for visual servoing**. In *International Conference on Control, Automation, Robotics and Vision*, pages 1–6, Dec. 2006. 21, 22

[125] N.R. Gans and S.A. Hutchinson. **An asymptotically stable switched system visual controller for eye in hand robots**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **1**, pages 735–742, Oct. 2003. 21, 22

[126] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. **Keeping features in the field of view in eye-in-hand visual servoing: a switching approach**. *IEEE Transactions on Robotics*, **20**(5):908–914, Oct. 2004. 21, 22

[127] N.R. Gans and S.A. Hutchinson. **An experimental study of hybrid switched system approaches to visual servoing**. In *IEEE International Conference on Robotics and Automation*, **3**, pages 3061–3068, Sep. 2003. 21, 22

[128] N.R. Gans and S.A. Hutchinson. **Stable visual servoing through hybrid switched-system control**. *IEEE Transactions on Robotics*, **23**(3):530–540, Jun. 2007. 21, 22, 23, 24

[129] L. Deng, F. Janabi-Sharifi, and W.J. Wilson. **Hybrid motion control and planning strategies for visual servoing**. *IEEE Transactions on Industrial Electronics*, **52**(4):1024–1040, Aug. 2005. 21, 23

[130] A.H.A. Hafez and C.V. Jawahar. **Visual servoing by optimization of a 2D/3D hybrid objective function**. In *IEEE International Conference on Robotics and Automation*, pages 1691–1696, Apr. 2007. 21

[131] A.H.A. Hafez, E. Cervera, and C.V. Jawahar. **Hybrid visual servoing by boosting IBVS and PBVS**. In *International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, Apr. 2008. 21

[132] F. Chaumette and E. Malis. **2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings**. In *IEEE International Conference on Robotics and Automation*, **1**, pages 630–635, April 2000. 21, 88

[133] K. Deguchi. **Optimal motion control for image-based visual servoing by decoupling translation and rotation**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **2**, pages 705–711, Oct. 1998. 21

[134] E. Malis. **Hybrid vision-based robot control robust to large calibration errors on both intrinsic and extrinsic camera parameters**. In *Control Conference (ECC), 2001 European*, pages 2898–2903, Sept 2001. 21

[135] S. Benhimane and E. Malis. **Homography-based 2D visual tracking and servoing**. *International Journal of Robotics Research*, **26**(7):661–676, July 2007. 21

[136] K. Hashimoto, T. Ebine, and H. Kimura. **Dynamic Visual Feedback Control For A Hand-eye Manipulator**. In *Proceedings of the 1992 lEEE/RSJ International Conference on Intelligent Robots and Systems*, **3**, pages 1863–1868, July 1992. 22

[137] E. Zergeroglu, D.M. Dawson, M.S. de Queiroz, and S. Nagarkatti. **Robust visual-servo control of robot manipulators in the presence of uncertainty**. In *IEEE Conference on Decision and Control*, **4**, pages 4137–4142, 1999. 22

[138] A. Astolfi, Liu Hsu, M.S. Netto, and R. Ortega. **Two solutions to the adaptive visual servoing problem**. *IEEE Transactions on Robotics and Automation*, **18**(3):387–392, June 2002. 22

[139] B. Bishop, S. Hutchinson, and M. Spong. **Camera modelling for visual servo control applications**. *Math. Comput. Model.*, **24**(5-6):79–102, Sep. 1996. 22

[140] L. Hsu and P.L.S. Aquino. **Adaptive visual tracking with uncertain manipulator dynamics and uncalibrated camera**. In *IEEE Conference on Decision and Control*, **2**, pages 1248–1253, 1999. 22

[141] E.C. Dean-Leon, V. Parra-Vega, A. Espinosa-Romero, and J. Fierro. **Dynamical image-based PID uncalibrated visual servoing with fixed camera for tracking of planar robots with a heuristical predictor**. In *IEEE International Conference on Industrial Informatics*, pages 339–345, June 2004. 22

[142] R. Kelly. **Robust asymptotically stable visual servoing of planar robots**. *IEEE Transactions on Robotics and Automation*, **12**(5):759–766, Oct. 1996. 22

[143] Hesheng Wang, Yun-Hui Liu, and Weidong Chen. **Uncalibrated Visual Tracking Control Without Visual Velocity**. *IEEE Transactions on Control Systems Technology*, **18**(6):1359–1370, Nov. 2010. 22

[144] R. Horaud. **New Methods for Matching 3-D Objects with Single Perspective Views**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**(3):401–412, May 1987. 23

[145] B. Nelson and P.K. Khosla. **Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance**. *International Journal of Robotics Research*, **14**(3):418–423, 1995. 24

[146] E. Marchand, F. Chaumette, and A. Rizzo. **Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, **3**, pages 1083–1090, Nov. 1996. 24

[147] C.W. Wampler. **Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods**. *IEEE Transactions on Systems, Man and Cybernetics*, **16**(1):93–101, Jan. 1986. 24

[148] Y. Nakamura and H. Hanafusa. **Inverse kinematic solutions with singularity robustness for robot manipulator control**. *Transactions ASME Journal of Dynamic Systems, Measurement, and Control*, **108**(3):163–171, Sep. 1986. 24

[149] O. Egeland, M. Ebdrup, and S. Chiaverini. **Sensory control in singular configurations application to visual servoing**. In *IEEE International Workshop on Intelligent Motion Control*, **2**, pages 401–406, Aug. 1990. 24

[150] S. Chiaverini. **Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators**. *IEEE Transactions on Robotics and Automation*, **13**(3):398–410, Jun. 1997. 24, 115

[151] M. Fruchard, P. Morin, and C. Samson. **A framework for the control of nonholonomic mobile manipulators**. *The International Journal of Robotics Research*, **25**(8):745–780, 2006. 24

[152] G. Chesi and A. Vicino. **Visual servoing for large camera displacements**. *IEEE Transactions on Robotics*, **20**(4):724–735, Aug. 2004. 24

[153] A. H. Abdul Hafez, A. K. Nelakanti, and C. V. Jawahar. **Path planning approach to visual servoing with feature visibility constraints: A convex optimization based solution**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1981–1986, Oct 2007. 24

[154] F. Janabi-Sharifi, L. Deng, and W.J. Wilson. **Comparison of basic visual servoing methods**. *IEEE/ASME Transactions on Mechatronics*, **16**(5):967–983, Oct. 2011. 25

[155] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control (2nd Edition)*. 2004. 28, 38, 39, 40, 72, 87

[156] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision (2nd Edition)*. Cambridge University Press, 2004. 47, 49, 56, 58, 65, 66, 67

[157] *Matlab Camera Calibration Toolbox*. http://www.vision.caltech.edu/bouguetj/calib_doc/. 52, 55

[158] K. S. Arun, T. S. Huang, and S. D. Blostein. **Least-Squares Fitting of Two 3-D Point Sets**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**(5):698–700, Sep. 1987. 59, 74

[159] M. Gridseth, C. P. Quintero, R. Tatsambon-Fomena, O. Ramirez, and M. Jagersand. **Bringing Visual Servoing into Real World Applications**. In *In Human Robot Collaboration workshop. Robotics Science and Systems (RSS 2013)*, June 2013. 63

[160] Junping Wang and Hyungsuck Cho. **Micropeg and Hole Alignment Using Image Moments Based Visual Servoing Method**. *IEEE Transactions on Industrial Electronics*, **55**(3):1286–1294, Mar. 2008. 63

[161] E.C. Dean-León, V. Parra-Vega, A. Espinosa-Romero, and L. García. **Visual Servoing for Constrained Robots: A New Complete TheoreticalFramework and its Experimental Validation**. In *Proc. of the IEEE/ASMEInternational Conference on Advanced Intelligent Mechatronics (AIM)*, July 2005. 75

[162] O. Khatib. **Real-Time Obstacle Avoidance for Manipulators and Mobile Robots**. **5**(1):90–98, 1986. 99

[163] Markus Rickert. *Efficient Motion Planning for Intuitive Task Execution in Modular Manipulation Systems*. Dissertation, Technische Universität München, Munich, Germany, 2011. 104, 123

[164] Nikhil Somani, Emmanuel Dean-Leon, Caixia Cai, and Alois Knoll. **Scene Perception and Recognition in industrial environments for Human-Robot Interaction**. In *9th International Symposium on Visual Computing*, July 2013. 108, 113

[165] Nikhil Somani, Alexander Perzylo, Caixia Cai, Markus Rickert, and Alois Knoll. **Object Detection using Boundary Representations of Primitive Shapes**. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Zhuhai, China, December 2015. 108, 137

[166] Nikhil Somani, Emmanuel Dean-Leon, Caixia Cai, and Alois Knoll. **Scene Perception and Recognition for Human-Robot Co-Operation**. In *ICIAP'13 workshop: First International Workshop on Assistive Computer Vision and Robotics (ACVR 2013). The 17th International Conference on Image Analysis and Processing.*, September 2013. 113

[167] Nikhil Somani, Emmanuel Dean-Leon, Caixia Cai, and Alois Knoll. **Perception and Reasoning for Scene Understanding in Human-Robot Interaction Scenarios**. In *In ICIAP'13 workshop: Recognition and Action for Scene understanding (REACTS 2013). The 15th International Conference of Computer Analysis of Images and Patterns.*, August 2013. 113

[168] Nikhil Somani, Caixia Cai, Alexander Perzylo, Markus Rickert, and Alois Knoll. **Object Recognition using constraints from Primitive Shape Matching**. In *Proceedings of the International Symposium on Visual Computing (ISVC)*, pages 783–792. Springer, December 2014. 113, 137

[169] Yoshiihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. **Task-priority Based Redundancy Control of Robot Manipulators**. *The International Journal of Robotics Research*, **6**(2):3–15, July 1987. 115

[170] B. Siciliano and J.J.E. Slotine. **A general framework for managing multiple tasks in highly redundant robotic systems**. In *Fifth International Conference on Advanced Robotics (ICAR). 'Robots in Unstructured Environments'*, pages 1211–1216, June 1991. 115

[171] P. Baerlocher and R. Boulic. **Task-priority formulations for the kinematic control of highly redundant articulated structures**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, **1**, pages 323–329, Oct. 1998. 115

[172] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter. **iTASC: a tool for multi-sensor integration in robot manipulation**. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 426–433, Aug. 2008. 115

[173] C. Lenz, M. Rickert, G. Panin, and A. Knoll. **Constraint task-based control in industrial settings**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3058–3063, Oct 2009. 115

[174] Nikhil Somani, Andre Gaschler, Markus Rickert, Alexander Perzylo, and Alois Knoll. **Constraint-Based Task Programming with CAD Semantics: From Intuitive Specification to Real-Time Control**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015. 115, 132

[175] Luis Sentis and Oussama Khatib. **Synthesis of whole-body behaviors through hierarchical control of behavioral primitives**. *International Journal of Humanoid Robotics*, pages 505–518, 2005. 115, 116, 117, 132

[176] M. de Lasa and A. Hertzmann. **Prioritized optimization for task-space control**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5755–5762, Oct. 2009. 115

[177] L. Saab, N. Mansard, F. Keith, J.-Y. Fourquet, and P. Soueres. **Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints**. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1091–1096, May 2011. 115

[178] Michael Mistry and Ludovic Righetti. **Operational Space Control of Constrained and Underactuated Systems**. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011. 115

[179] Jae Won Jeong and Pyung Hun Chang. **A task-priority based framework for multiple tasks in highly redundant robots**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5886–5891, Oct. 2009. 115

[180] Jan Peters, Michael Mistry, Firdaus Udwadia, Jun Nakanishi, and Stefan Schaal. **A unifying framework for robot control with redundant DOFs**. *Autonomous Robots*, **24**:1–12, 2008. 116

[181] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. **Prioritized motion force control of constrained fully-actuated robots: Task Space Inverse Dynamics**. *Robotics and Autonomous Systems*, **63**:150–157, 2015. 116

[182] O. Khatib. **A unified approach for motion and force control of robot manipulators: The operational space formulation**. *IEEE Journal of Robotics and Automation*, **3**:43–53, February 1987. 117

[183] Luis Sentis. *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Stanford University, July 2007. 120

[184] Nikhil Somani, Markus Rickert, Andre Gaschler, Caixia Cai, Alexander Perzylo, and Alois Knoll. **Task level robot programming using prioritized non-linear inequality constraints**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Republic of Korea, October 2016. accepted. 132

[185] G. R. Bradski. **Computer Vision Face Tracking for Use in a Perceptual User Interface**. In *OpenCV*, 1998. 139

[186] *OpenCV (Open Source Computer Vision)*. http://opencv.willowgarage.com/wiki/. 139