Pontifícia Universidade Católica
do Rio de Janeiro

Juan David Gamba Camacho

A Robust Visual Servoing Approach for Robotic Fruit Harvesting

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor : Prof. Antonio Candea Leite
Co-advisor: Prof. Pål Johan From

Rio de Janeiro
September 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Juan David Gamba Camacho

A Robust Visual Servoing Approach for Robotic Fruit Harvesting

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the undersigned Examination Committee.

Prof. Antonio Candea Leite
Advisor
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Fernando Cesar Lizarralde
Universidade Federal do Rio de Janeiro – COPPE/UFRJ

Prof. Wouter Caarls
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico – PUC-Rio

Rio de Janeiro, September 6th, 2018

Juan David Gamba Camacho

Majored in Electronics and Communications Engineering with a specialization in Automation and Control by the Universidad Latina de Costa Rica (Heredia, Costa Rica).

Bibliographic data

Gamba Camacho, Juan David

A Robust Visual Servoing Approach for Robotic Fruit Harvesting / Juan David Gamba Camacho; advisor: Antonio Candea Leite; co-advisor: Pål Johan From. – Rio de janeiro: PUC-Rio , Departamento de Engenharia Elétrica, 2018.

v., 137 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui bibliografia

1. Electrical Engineering – Teses. 2. Signal Processing, Automation and Robotics – Teses. 3. Servovisão;. 4. Robôs Agrícolas;. 5. Colheita de Frutas Automática;. 6. Controle Cinematico.. I. Candea Leite, Antonio. II. Johan From, Pål. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

To my parents and family, for their support
and encouragement.

Acknowledgments

I would like to thank my advisor, Prof. Antonio C. Leite, for his patient guidance and advice provided through this master program. Then, I wish to thank my co-advisor, Prof. Pål Johan From, for his support and interest in my research topic.

This study was financed in part by the Coordenação de Aperfeiçoamento Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Gamba Camacho, Juan David; Candea Leite, Antonio (Advisor); Johan From, Pål (Co-Advisor). **A Robust Visual Servoing Approach for Robotic Fruit Harvesting**. Rio de Janeiro, 2018. 139p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

In this work, we present different eye-in-hand visual servoing control schemes applied to a robotic harvesting task of soft fruits in the presence of parametric uncertainties in the system models. The first scheme combines position-based visual servoing (PBVS) and image-based visual servoing (IBVS) approaches in order to perform respectively an approach phase to the fruit and then a fine tuning of the end-effector to harvest. The second scheme uses a hybrid visual servoing (HVS) approach to fulfill the complete harvesting task, by designing a suitable control law which combines error vectors defined in both the image and operational spaces. For detecting the fruits, an algorithm based on the combination of the OHTA color space and Otsu's threshold method for a fast recognition of mature fruits in complex scenarios. In addition, a more accurate detection method employs a pre-trained deep encoder-decoder algorithm based on a minimized Segnet version for a fast and cheap inference during the task execution. The object localization is accomplished by employing an image triangulation technique, which combines the speeded-up-robust-features (SURF) and the-random-sample-consensus (RANSAC) or the Oriented FAST and Rotated BRIEF and the Brute-Force Matcher (BF-Matcher) algorithms to extract the fruit image feature and match it to its correspondent feature-point into the other view of the stereo camera. However, since these algorithms are computationally expensive for the task requirements, a faster estimation method uses the fruit centroid and a homogeneous transformation for discovering matching points. Finally, a vision-based sliding-mode-control scheme and a switching monitoring function are employed to cope with uncertainties in the calibration parameters of the camera-robot system. In this context, it is possible to guarantee the asymptotic stability and convergence of the image feature error, even if the misalignment angle, around the z-axis, between the camera and end-effector frames is uncertain. 3D computer simulations and preliminary experimental results, obtained with a Mitsubishi robot arm RV-2AJ carrying out a simple strawberry picking task, are included to illustrate the performance and effectiveness of the proposed control scheme.

Keywords

Visual Servoing; Agricultural Robots; Automatic Fruit Harvesting; Kinematic Control.

Resumo

Gamba Camacho, Juan David; Candea Leite, Antonio; Johan From, Pål. **Uma Abordagem de Servovisão Robusta para Colheita Robótica de Frutas**. Rio de Janeiro, 2018. 139p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Neste trabalho, apresenta-se diferentes esquemas de controle servovisuais para tarefas robóticas de colheita de fruta, na presença de incertezas paramétricas nos modelos do sistema. O primeiro esquema combina as abordagens de servovisão baseada em posição (PBVS) e servovisão baseada em imagem (IBVS) para realizar, respectivamente, a aproximação até a fruta e, em seguida, um ajuste fino para a colheita. O segundo esquema usa uma abordagem de servovisão híbrida (HVS) para realizar a tarefa de colheita completa, projetando uma lei de controle adequada que combina vetores de erro definidos no espaço operacional e no espaço da imagem. A fase de detecção utiliza um algoritmo baseado no espaço de cores OTHA e limiar da imagem Otsu para um rápido reconhecimento de frutos maduros em cenários complexos. Além disso, um método de detecção mais preciso emprega uma Rede Neural Convolucional Profunda (DCNN) pré-treinada baseada em uma versão Segnet minimizada para uma inferência rápida durante a execução da tarefa. A localização do objeto é realizada empregando uma técnica de triangulação de imagem, que combina os algoritmos SURF e RANSAC ou ORB e BF-Matcher para extrair a característica da imagem da fruta e associa-lo com o seu ponto correspondente na outra visualização. No entanto, como esses algoritmos exigem um elevado custo computacional para os requisitos da tarefa, um método de estimativa mais rápido utiliza o centróide da fruta e transformação homogênea para descobrir os pontos correspondentes. Finalmente, um esquema de controle em modos deslizantes (SMC) baseado em visão e uma função de monitoramento de comutação são empregados para lidar com incertezas nos parâmetros de calibração do sistema de câmera-robô. Nesse sentido, é possível garantir a estabilidade assintótica e a convergência do erro da característica da imagem, mesmo que o ângulo de desalinhamento, em torno do eixo z, entre os sistemas de coordenadas da câmera e do efetuador seja incerto.

Palavras-chave

Servovisão; Robôs Agrícolas; Colheita de Frutas Automática; Controle Cinematico.

Table of contents

List of figures

List of tables

List of Abreviations

CHT – Circular Hough Transform
CNN – Convolutional Neuronal Networks
DCNN – Deep Convolutional Neuronal Networks
DOF – Degree of Freedom
FCNN – Fully Convolutional Neuronal Network
GAN – Generative Adversarial Network
HVS – Hybrid Visual Servoing
IBVS – Image Based Visual Servoing
MLP – Multilayered Perceptron
ORB – Oriented FAST and Rotated BRIEF
PBVS – Position Based Visual Servoing
RANSAC – Random Sample Consensus
ROS – Robot Operating System
SLAM – Simultaneous Localization and Mapping
SMC – Switching Monitoring Controller
ST-SM – Super Twisting Sliding Mode
UVC – Unit Vector Control
VS-ABC – Variable Structure Adaptive Backstepping Controller
VSC – Variable Structure Control
WS – Watershed Segmentation

Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution.

Albert Einstein, *" What Life Means to Einstein".*

1
Introduction

Machines picking apples from trees by using vacuums in the US, octopus robots for collecting strawberries in Spain, others devices feeding and milking cows in the UK. These are some examples of how robots are being incorporated into different agricultural applications around the world. In fact, robots are expected to help and/or replace humans in repetitive and fatigue duties such as agricultural chores (Fig. 1.1).

As the world population is growing, the agricultural industry is expected to grow by around twenty-three percent annually, requiring around 90,000 workers a year by 2021 just in Europe [1].

On the economic side, with these exponential production growing, challenges as maintaining low cost and maximum profit become an obligation for any who wants to continue in business; on the other hand, food production processes need to be re-structured reducing the waste of resources and increasing the productivity of small cultivated areas, in order to decrease its negative environmental impact [2].

Figure 1.1: Strawberry pickers, mostly from Poland, in poly-tunnels on a farm in Kent, UK. Photograph: Graeme Robertson for the Guardian [2]

These concerns about food production do not belong exclusively to

Europe and can also be considered in many countries of Central and South America, particularly, in Brazil. Although, the Brazilian agricultural industry has a high degree of automation for the planting and harvesting of grains and sugarcane in large areas, farmers still do not use autonomous robotic systems to perform basic and complex agricultural tasks in small areas, such as vegetable gardens and orchards. Some basic agricultural tasks include sowing, fertilizing, and irrigation, while some complex agricultural tasks consist of harvesting fruits, killing weeds and plant phenotyping [2].

1.1 Motivation

A new trend for agricultural robotics, following the recent advances in robotic technology around the world; is to combine and integrate advanced control theory, computer vision algorithms, and machine learning techniques into visual servoing approaches, allowing robots to perform agricultural tasks with a high level of autonomy and better response to decision-making assignments into complex and very dynamic scenarios [3, 4, 5].

Figure 1.2: From left-to-right and top-to-bottom: Iron Ox Lettuce Robot, Robot Gardener, Agrobot SW6010 and Hamster Bot.

Over the last years, the technological advances of sensors and communication systems have encouraged the agricultural industry to employ and design intelligent autonomous robots to carry out a number of repetitive and dull tasks for farmers in orchards, vineyards, poly-tunnels and farms [7, 8].

Fruit harvesting and picking, weed control, autonomous mowing, pruning, seeding and spraying, plant phenotyping, sorting and packing (Fig. 1.2) are just a few examples of how robots are taking over fields around the world.

In general, these robots are equipped with complex systems with specialized accessories for navigation and different kind of tools (e.g., gripper and pickers), in order to accomplish challenging tasks successfully, safely and efficiently [8]. In addition to being mobile, the robot must contain a sensor package integrated into an autonomous navigation system so that it can move freely and autonomously in the open field or between rows of the plants [9].

Agricultural environments introduce several challenges and difficulties, particularly, for robotic harvesting and 3D navigation of mobile robots. Indeed, changes in seasons and weather conditions, crop growth, and rotation, dense vegetation, different maturity levels of fruits, the existence of diseases and fungi in plants, all these factors create a dynamic and poorly structured environment. Thus, the automatic fruit harvesting system has to incorporate perception and cognition capabilities in the gripper design [10] as well as intelligent sensors and systems for fruit detection, recognition and localization [11].

In general, crop environments include four important sources of variations for robotic harvesting tasks [8]:

- Target objects: fruits and vegetables can vary in position, size, shape, colors, and texture;
- The production conditions: orchard, greenhouse, indoor, or open field may generate a lot of variation in light conditions, visibility, and accessibility of the target objects;
- Type of crops: plants geometry differs among crops and it can drastically change technical details for robotic harvesting tasks, e.g., type of grippers;
- Obstacles: foliage, stems, other fruits, leaves may block access to the target object, reducing the visibility and creating difficulties for localization purposes.

Thus, robotic harvesting tasks for fruits and vegetables in the presence of these sources of variations and disturbances still remains an open and challenging research problem.

1.1.1 Visual-Servoing

Visual servoing is the area of robotics that concerns the pose control of robot arms or mobile robots based on the feedback of visual information extracted from one or more image features, by using a single or multiple cameras [12]. Such cameras may have different models (e.g., monocular, stereo and RGB-D) and be mounted in different configurations: fixed in the robot workspace (i.e., eye-to-hand) or attached to the robot end-effector (i.e., eye-in-hand). More advanced applications, however, can use different types and settings of cameras simultaneously [13].

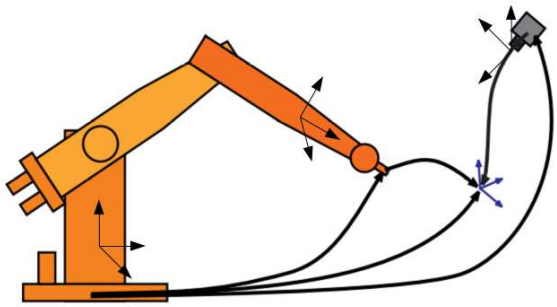

Figure 1.3: Eye-to-hand camera configuration. [13]

In general, the direct measurements provided by the vision system are related to the image feature parameters in the image space (Fig. 1.4)., while the robotic task is defined in the operating or task space in terms of the relative pose of the robot end-effector with respect to the target object (Fig. 1.3). In this context, vision-based controllers can be classified into two main

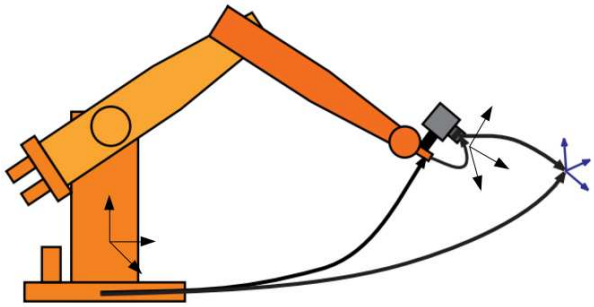

Figure 1.4: Eye-in-Hand Camera Configuration [13].

groups: position-based visual servoing (PBVS), image-based visual servoing (IBVS) as well as hybrid visual servoing (HVS) that combines the benefits of both approaches [14]. A key difference between PBVS and IBVS approaches is related to the space where the quantities used to calculate the control action are defined: task-space or image space. One of the main advantages of the

IBVS approach over the PBVS approach is its higher robustness to camera calibration errors, which results in better positioning accuracy for the vision system. For executing a successful robotic harvesting task of horticultural products there are two challenges to be considered; firstly, the recognition and localization system needs to detect the target pose (position and orientation) by employing a vision system for detecting fruits or vegetables; secondly, the control system has to move the robot end-effector - with a suitable tool - attached - towards the location of the object of interest accurately to perform the harvesting task [15].

A key operation in visual servoing is the camera calibration phase, which consists of obtaining the intrinsic and extrinsic parameters of the camera in order to compute a suitable pose (position and orientation) of the image features at the image projection plane. Another challenging aspect is to obtain the geometric model of the target object to carry out the depth control by using a single camera. On the other hand, by using stereo vision systems this information can be directly obtained by processing the images obtained from two or more cameras.

Following this motivation, in this master thesis, we aim to develop an integrated methodology for semi-autonomous fruit harvesting based on the combination of computer vision, machine learning, visual servoing and control theory techniques, to be applied to orchards, vineyards and farms [8].

1.2 Review Of The State Of The Art

Currently, crops are usually harvested when most of the plants are ready to be collected, minimizing the probability of picking fruits that do not meet a certain threshold of quality. Alternatively, using a proper selective harvesting, it is possible to collect only the matured enough fruits, leaving the others to be collected in the near future, maximizing the production performance and the quality of fruits [7, 16]. Selective harvesting applications are well known in forestry, where quality and size are key patterns for choosing the right tree and let the remaining to improve their quality and size for future harvesting.

Following this trend, picking or harvesting tasks are being carried out by using cutting tools, soft-manipulators (Fig. 1.5) or mechanized harvesting technologies. Manually-placed limb actuators can be used to apply vibrations to effect fruit release in sweet cherry crops [17]. In this context, different approaches such as SLAM, computer vision, and machine learning techniques are used to improve the performance and accuracy of the harvesting task. Moreover, a color-based object extraction method based on OHTA color space

was developed by Wei *et al.* and used to carry out robotic strawberry picking tasks under complex agricultural background [18].

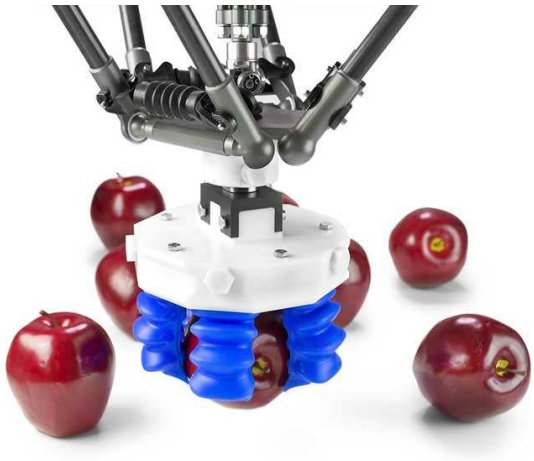

Figure 1.5: Soft Robotics demonstrates its air-actuated gripper in the robotics company's Cambridge, Massachusetts, laboratory.

For carrying out the fruit picking, it is crucial to choose a well-fit gripper that meets the task requirements. This decision may facilitate the design of the control algorithm improving the performance of robotic harvesting task. In this context, Bac *et al.*, present different results achieved for harvesting sweet peppers by using two different types of tools, say Fin Ray and Lip Type [19].

1.2.1 Computer Vision and Machine Learning

A proper image segmentation into complex scenarios has demonstrated to be one of the main challenges of using data obtained from vision systems, which may difficult the computation of an accurate 3D measurement of the object of interest. In this sense, applications involving machine learning are becoming more and more popular every day for challenging scenarios commonly found in the agricultural sector.

Lottes *et al.* have developed a precision system for selective recognition and spraying of weeds and invasive species, by using a camera mounted on a mobile field platform to obtain data from the environment and a machine learning algorithm called random forest classification to detect and remove weeds [20].

Considering the robotic harvesting problem in vineyards and fields of grape vines, a mobile robot system was developed for automatic pruning, by using a trinocular stereo camera to build a more accurate three-dimensional (3D) model of the vines. Hereafter, a machine learning algorithm decides which

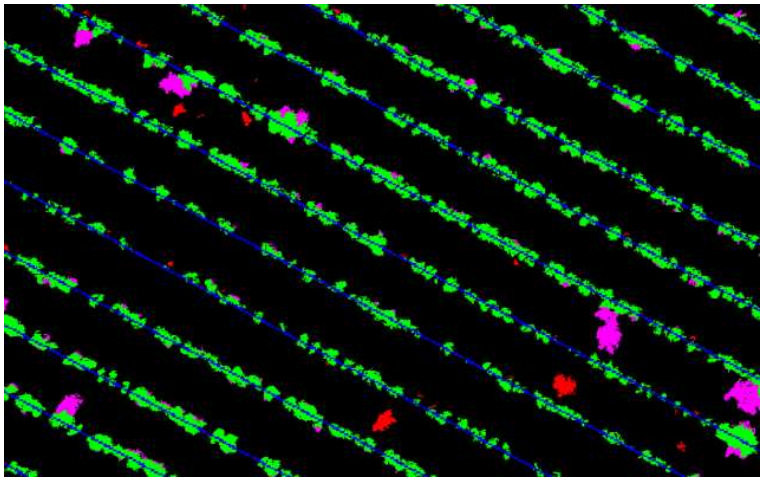

Figure 1.6: Weeds Detection and Identification [21].

canes to prune by using a six-degree-of-freedom (6-DOF) robot arm to perform the cutting assignment [22].

Botterill *et al.* have developed an artificial intelligence algorithm for pruning after recognizing the structure of the plant, wherein the AI algorithm determines where to prune, knowing the hardware limitations of the robotic arm [22].

Deep encoder-decoder architectures have been used recently to perform semantic segmentations into very complex backgrounds, due to their ability for learning textures and image features of a given interest object, these algorithms demonstrate the possibility of carrying out detection and localization tasks in non-controllable scenarios [23]. Hung *et al.* [24] introduce a segmentation scheme using multispectral images, sparse autoencoders, and support vector machine (SVM) schemes to segment leaves.

A framework for detecting and counting of apples was developed by Bargoti *et al.* , where by using machine learning schemes such as Multilayered Perceptron (MLP) and Convolutional Neuronal Networks (CNN) and classic computer vision approaches such as Watershed Segmentation (WS) and Circular Hough Transform (CHT) algorithms it is possible to detect and count individual fruits [25]. Senthilnath *et al.* have designed a fruit detection algorithm by using spectral-spatial methods in remotely sensed RGB images captured by an UAV, which could be used for tomato harvesting [30].

Dias *et al.* [26] have proposed a robust flower identification algorithm based on fully convolutional neuronal networks (FCNs), they have demonstrated how FCNs are able to deal with very challenging segmentation assignments.

1.2.2 Visual Servoing

Vision plays an important role in robotics systems, where data obtained by vision systems from the environment where the robot operates, presents a rich geometrical and qualitative information critical for achieving a successful task execution [37].

Barth *et al.* have proposed a visual servoing approach which uses the eye-to-hand camera configuration for sweet pepper harvesting in dense vegetation [27].

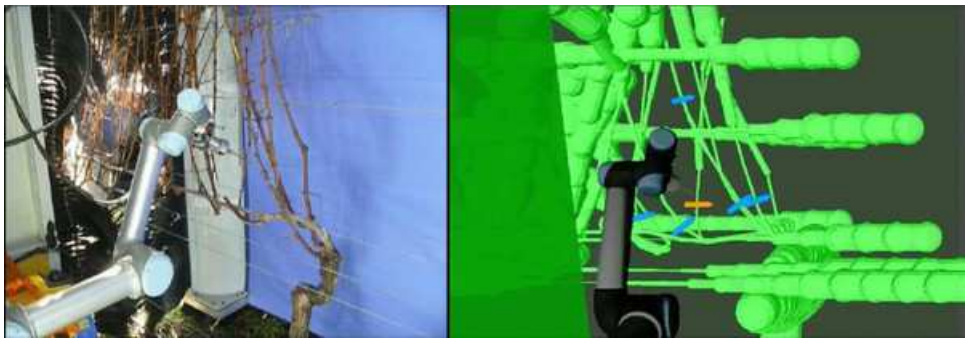

Figure 1.7: Robot arm reaching to prune the last of six cutpoints on a Sauvignon Blanc vine at the Lincoln University vineyard (left). The visualization on the right shows the model of the robot arm, the vines, and the cutpoints. [22]

.

Mehta and Burks have designed a vision-based estimation and control system for robotic citrus harvesting based on the combination of large field-of-view of a fixed camera and the accuracy of a mobile camera [28].

A detection and localisation algorithm applied to robotic apple harvesting which detects red and bicoloured fruits on tree by using an RGB-D camera was developed by Nguyen *et al.* [29].

1.2.3 Adaptive and Robust Control Schemes

Modeling inaccuracies or model imprecision at the extrinsic and intrinsic camera calibration parameter or robot kinematics due to simplified representations, may represent a real challenge from the control point of view. In particular, this inaccuracies can be classified in structure (parametric) and unstructured uncertainties(unmodeled dynamics). Two major and complementary control schemes for dealing with model uncertainties are *robust control* and *adaptive control* [60].

Robust controllers present useful techniques for rejecting external disturbances by employing a high-frequency commuted control action to constraint error trajectories on a sliding subspace. In this sense, Queiroz *et al.* presents a robust controller without envolving visual serving control schemes, the design and stability analysis of a variable structure adaptive backstepping controller (VS-ABC) for linear plants. The proposed approach uses switching laws for increasing the system robustness to parametric uncertainties and disturbances [31].

Cheah *et al.* use an adaptive visual servoing design for motion, and force tracking application in the presence of uncertainties in the surface, robot kinematics and dynamics, and camera model. This control approach intends to update online the uncertain internal and external parameters for the successful execution of the interaction task [32].

Roux *et al.* address a robust visual servoing strategy by using switching and sliding mode control techniques to deal with uncertainties in the intrinsic camera parameters and camera orientation angle with respect to the robot base frame to perform a trajectory tracking task [33].

Leite *et al.* consider a 3D visual tracking problem with uncertainties at the kinematics and dynamics model of the robot arm, and intrinsic camera parameters, they introduce an adaptive visual servoing scheme based on a kinematic approach for an image-based-look-and-move system to perform a 3D tracking task [34].

Mehta *et al.* present an autonomous citrus harvesting by employing a cooperative scheme based on eye-to-hand and eye-in-hand cameras configuration [35].

1.3 Goals and Objectives

The main goal of this work is to explore different visual servoing control technique in terms of model design, simulation, and practical implementation. The key idea is to combine different schemes from different areas like visual servoing, robust control theory, and computer vision to face the main challenges found in soft fruit manipulation applications, very common in greenhouses and farms [15].

Model, design and implement a visual-servoing based control scheme for recognizing and picking soft-fruits using a robot manipulator and different camera configurations in the presence of kinematic singularities and parametric uncertainties in system models and external perturbations.

Present verification and validation results of the proposed visual servoing

methodology for soft-fruits harvesting tasks in Matlab and Virtual Robot platform (V-REP). The author also intends (if possible) to carry out experimental tests with an autonomous robotic system for fruit picking, on a strawberry farm, in Sylling, Norway.

The objectives, regarding theoretical and practical aspects in short- and long-term scenarios, are the following:

- Carry out a prospective study on the state-of-the-art of soft fruit harvesting devices based on robotic systems to understand the main challenges involved in the design and practical implementation of the control system;
- Evaluate different computer vision techniques for soft-fruit detection, recognition, and selection at low computational cost (e.g., color segmentation, machine learning);
- Evaluate the complexity and performance trade-offs between different vision-based control schemes such as (i) position-based visual servoing (e.g., 3D reconstruction), (ii) image-based visual servoing and (iii) hybrid visual servoing.
- Image-based visual servoing: prospective study on the reliability and robustness of the existing vision-based controllers to uncertainties in the camera calibration parameters.
- Identify the main advantages and disadvantages of vision-based controllers in the presence of model uncertainties and external disturbances, in terms of theoretical and practical aspects.
- Analyse and design a robust solution for the fruit picking problem which is able to deal with these types of perturbations.
- Carry out numerical simulations, 3D graphical animations and experimental tests of a good-fit visual-servoing control scheme for robotic harvesting task of soft fruits

1.3.1 Methodology

In this work, we address the soft-fruit harvesting problem by using a visual servoing approach based on the combination of computer vision, machine learning, and control theory methodologies [36].

Basically, a visual system setup is composed of only one camera or multiple cameras. An experimental setup with two or more cameras for parameters extraction is referred as 3D vision or stereo vision, which is capable

of retrieving the depth information evaluating the distance of the target object to the vision system. 3D vision can be also implemented by using a single camera and using two images of the target object from different poses. By processing and combining both images simultaneously depth information can be extracted. If the camera images are restricted to a single pose, from the knowledge of the target geometric model it is also possible to estimate the distance from the target to the camera.

As discussed early, the task of interest in visual servoing is to control the robot end-effector, relative to the target object, using visual features extracted from the image. In particular, this task can be performed by using eye-to-hand camera configuration, where the camera is located in a fixed position of the workspace and the robot end-effector is being observed for the visual sensor for computing the desired motion (Fig. 1.3). Other camera configuration consists of mounting the camera at the robot end-effector, making the visual sensor mobile, which is called eye-in-hand camera configuration (Fig. 1.4). On the other hand, the presence of error components in the image space helps keep the image features in the camera field of view, which is a difficult task in position-based approaches [37].

A relevant constraint for visual servoing is the camera calibration problem, which consists of obtaining the intrinsic (or internal) and extrinsic (or external) parameters of the camera. The focal length of the camera lens, scaling factors, skew angle and the position of the principal point are known as intrinsic parameters for a pinhole or monocular camera model , whereas the camera pose with respect to the base frame (or inertial frame) is known as extrinsic parameters. There exist many several techniques for calculating the camera calibration parameters, and there are different camera projection models, for instance, fish-eye lens, pinhole, catadioptric and spherical. The control design uses a kinematic control approach based on a recently proposed method to deal with the existence of singular configurations for the robot arm during the harvesting task. A robust vision-based control scheme which combines the image-based visual servoing (IBVS) and the position-based visual servoing (PBVS) approaches is designed to cope with parametric uncertainties in the camera-robot system.

The hybrid approach combines the benefits of PBVS and IBVS, its name deerives from the fact that the control error is defined in the operational space for some components and in the image space for the others. This implies that the desired motion can be specified, at least partially, in the operational space so that the camera trajectory during visual servoing can be predicted in advance for some components.

In this work, the harvesting problem is achieved by using a visual servoing approach based on the combination of sliding mode control approach with a switching monitoring function. A robust vision-based control scheme which combines the IBVS and the PBVS approaches is designed to cope with parametric uncertainties in the camera-robot system. The proposed HVS control scheme ensures the asymptotic stability and convergence of the image feature error under a planar camera rotation miscalibration angle $|\tilde{\varphi}|$ greater than $\pi/2$ radian.

The color feature extraction method for fruit detection is based on OHTA color space, which is a more suitable option for real-time image processing than most of the DCNN algorithms [18]. The fruit localization method uses a combination of the fruit centroids and a Homogeneous transformation for performing a simpler and fast feature recognition and matching process, compared with other well-known algorithms as SURF, SIFT, RANSAC, ORB, BF-Matcher. Moreover, when the object of interest is out of camera focus, it is difficult to ensure the proper features extraction from the image, then the centroid of the fruit contour projected in the main camera plane is used to compute the fruit 3D position.3D Computer simulations and experimental results obtained with a Mitsubishi RV-2AJ robot, performing a simple strawberry picking task, illustrate the performance and efficiency of the proposed visual servoing scheme.

1.4 Contribution of this Thesis

In this work, a robust hybrid visual servoing (HVS) strategy combined with sliding mode control and switching monitoring function is presented to deal with intrinsic, extrinsic and kinematics uncertainties during a robotic picking application.

Some variations and/or contributions presented in this document, in contrast with [33] assumptions are:

- An eye-in-hand camera configuration with a stereo camera is used instead of an eye-to-hand setup with a monocular camera;
- In this work, we assumed parametric and non-parametric uncertainties at the length of the last link of the robot and at the rotation between the end-effector and camera frames;
- A hybrid visual servoing method is used to achieve the three-dimensional (3D) position of the object of interest rather than the image-based visual servoing for planar tracking.

– A hybrid control stability analysis is introduced, with sliding-mode and switching-based control methods. This analysis involves the presence of continuous and discrete terms into an extension of the Lyapunov Stability Theorem.

For applying the robust visual servoing control scheme to realistic situations; it was necessary to develop some other computer vision and machine learning algorithms, like:

– A segmentation algorithm based on the Otsu adaptive thresholding method of a pre-filtered OHTA color space image for fast detection and localization of fruits in real-world harvest situations.
– A circular sliding window procedure is used to resolve the problem of selecting a particular target in complex background scenarios.
– A memetic random restart strategy was developed to choose the optimal picking order when two or more objects are displayed in a scene (a traveling salesman problem).

1.5 Organization of the Thesis

This work is organized according with the following chapters:

– **Chapter 2:** Introduces dynamic and differential kinematics methods for robot controllers and robust control theory. Some practical examples involving uncertainties and/or perturbations during the execution of regulation and tracking assignments; are shown. Also, it is demonstrated how to deal with perturbed systems and possible specific solutions using adaptive and robust control theory.
– **Chapter 3:**Comments a general structure of common visual servoing systems in different camera configuration and control schemes. A rigorous stability analysis of an IBVS scheme is presented and combined with sliding mode and SMC control schemes.
– **Chapter 4:** Describes how the tasks of strawberries detection and recognition are carried out for a usual harvesting scenario. After detecting and recognizing phases, some image-processing algorithms are used to improve the features detection, to triangulate an accurate 3D target position. Finally, a combination of switching and sliding mode control schemes, presented in chapters two and three, are applied to the proposed hybrid visual servoing system (HVS), a further robustness analysis is also discussed at the end of the chapter.

2
Kinematics Modeling and Control Design

In this chapter, we consider the kinematic approach for modeling and controlling a robot arm. Most commercial robots usually have an internal velocity controller to control their joints positions and/or velocities when the control signal is applied to the manipulator space. In this context, we assume that the joint velocities can be directly controlled by a low-level control loop with high-performance, which imposes any specified reference velocity as system input [37]. Hence, if the tasks assigned to the robot end-effector are carried out at low speeds/slow accelerations and the transmissions have high-gear reduction ratios, the dynamic coupling effects can be neglected during the robot motion ensuring satisfactory performance of the overall system i.e. ignores dynamics.

2.1
Forward and Differential Kinematics

Here, we consider the kinematic model of a robot manipulator. In this framework, the *operational space* variables are related to the *joint space* variables by means of the following *forward kinematics* and *differential kinematics* mappings:

$$p = h(q)\,, \qquad \dot{p} = J_p(q)\,\dot{q}\,, \tag{2-1}$$

where $q, \dot{q} \in \mathbb{R}^n$ denote the position and velocity vectors of the robot joints, and $p, \dot{p} \in \mathbb{R}^m$ denote the position and linear velocity of the end-effector frame $\mathcal{F}_e$ with respect to the base frame $\mathcal{F}_b$. Notice that, $h(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a nonlinear function and $J_p(q) = (\partial h/\partial q) \in \mathbb{R}^{m\times n}$ is the position part of the *analytical Jacobian* [37].

The orientation of the end-effector frame $\mathcal{F}_e$ with respect to the base frame $\mathcal{F}_b$ can be described by the *unit quaternion* representation. In contrast with Euler angles quaternions approach give a global parameterization of $SO(3)$ given by $\phi = \{\eta, \epsilon\}$ where $\eta \in \mathbb{R}$ is the scalar part and $\epsilon \in \mathbb{R}^3$ is the vector part, subject to the unit norm constraint $\eta^2 + \epsilon^2 = 1$ [68]. The so-called *quaternion propagation rule* relates the time-derivative of the unit quaternion $\dot{\phi}$ to the angular velocity of the end-effector frame $\mathcal{F}_e$ with respect to the base frame $\mathcal{F}_b$, denoted by $\omega \in \mathbb{R}^3$, as:

$$\dot{\phi} = \frac{1}{2} E(\phi)\,\omega\,, \qquad E(\phi) = \begin{bmatrix} \epsilon^{\mathsf{T}} \\ \eta I - Q(\epsilon) \end{bmatrix}, \tag{2-2}$$

where $Q(\cdot) : \mathbb{R}^3 \mapsto \mathrm{so}(3)$ denotes the skew-symmetric operator($\mathrm{so}(3) \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix) and the matrix $J_r(\phi) = 2E^{\mathsf{T}}(\phi) \in \mathbb{R}^{3\times 4}$ is the well-known *representation Jacobian*.

The *differential kinematics equation* provides the relationship between the joint velocities and the corresponding linear and angular velocities of the end-effector frame $\mathcal{F}_e$ with respect to the robot frame $\mathcal{F}_b$ as:

$$\mathbf{v} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_o(q) \end{bmatrix} \dot{q} = J(q)\,\dot{q}\,, \tag{2-3}$$

where $J(q) \in \mathbb{R}^{m'\times n}$ is the *geometric Jacobian* of the robot manipulator. Notice that, in general, the orientation of the robot end-effector is given in terms of the robot joint angles as $\phi = g(q)$, where $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^4$ is a nonlinear function. Thus, considering (2-3) implies that $J_o(q) = J_r(\phi)\,J_\phi(q)$, where $J_\phi(q) = (\partial g(q)/\partial q) \in \mathbb{R}^{4\times n}$, which is the so-called orientation part of the analytical Jacobian. The kinematic model introduced in (2-3) has the following properties that will be considered in the analysis and control design for robot manipulators with revolute joints:

(P1) $J(q)$ is bounded for all possible values of $q(t)$, that is:

$$||J(q)||_\infty \leq c_0, \qquad \forall\, q \in [0, 2\pi]\,, \tag{2-4}$$

since this term depends on $q(t)$ as an argument of bounded trigonometric functions, and $c_0 \in \mathbb{R}$ is a known positive constant.

(P2) The product of the Jacobian $J(q)$ with any known vector $\nu(t) \in \mathbb{R}^m$ can be linearly expressed by,

$$W(q, \nu)\, b = J(q)\,\nu\,, \tag{2-5}$$

where $W(q, \nu) \in \mathbb{R}^{n\times n_k}$ is the differential kinematics regressor matrix, $\nu \in \mathbb{R}^n$ is a generic input signal, $b \in \mathbb{R}^{n_k}$ denotes the constant kinematic parameter vector, and n_k denotes the number of uncertain kinematic parameters of the Jacobian matrix $J(q)$. The lower and upper bounds, denoted as $b_{min}, b_{max} \in \mathbb{R}^{n_k}$, are assumed to be known for each parameter b and satisfies the following property:

$$b_{min,i} \leq b_i \leq b_{max,i}\,, \qquad i = 1, 2, \cdots, n_k\,, \tag{2-6}$$

where $b_{min,i}, b_{max,i} \in \mathbb{R}$ denote the i-th element of b_{min} and b_{max} respectively, and $b_i \in \mathbb{R}$ denotes the i-th element of b.

Notice that, the property (P1) holds not only for robot arms with revolute joints, but also for mechanisms with prismatic joints since these have physical limitations from the practical point of view [37].

2.2 Kinematic Control

Consider the kinematic control problem for a n-DoF robot manipulator. In this framework, the robot motion can be simply described by

$$\dot{q}_i \approx u_i\,, \qquad i = 1, \cdots, n\,, \tag{2-7}$$

where $\dot{q}_i$ is the velocity of the i-th robot joint and u_i is the velocity control signal applied to the i-th joint motor drive. Then, from the differential kinematic equation (2-3) and considering the kinematic control approach (2-7), we obtain the following control system:

$$\mathbf{v} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q)\,u(t)\,. \tag{2-8}$$

For the cases where the Jacobian matrix $J(q)$ is *non-square* ($m' < n$), the velocity control signal $u(t) \in \mathbb{R}^n$ can be given simply by:

$$u(t) = J^{\star}(q)\,v = J^{\star}(q) \begin{bmatrix} v_p \\ v_o \end{bmatrix}, \tag{2-9}$$

where $v = [\, v_p^{\mathsf{T}} \;\; v_o^{\mathsf{T}} \,]^{\mathsf{T}}$ is a Cartesian control signal, for position and orientation, to be designed and $J^{\star}(q) \in \mathbb{R}^{n \times m'}$ may be replaced by the right pseudo-inverse Jacobian $J^{\dagger}(q) = J^{\mathsf{T}}(JJ^{\mathsf{T}})^{-1}$ or the transpose Jacobian $J^{\top}(q)$ to avoid the inverse $J^{-1}(q)$ or pseudo-inverse $J^{\dagger}(q)$ calculations [37]. Notice that, the control signal (2-9) locally minimizes the norm of the joint velocities, provided that (i) the robot kinematics is *fully known* and (ii) the Cartesian control signal $v(t)$ does not lead the robot to *singular configurations*. The failure of the last assumptions is a fairly open-problem in robotics area, and will be discussed later on.

2.3 Inverse Kinematics Problem

In this section, we have discussed how to obtain the end-effector pose $\mathbf{x} = [\, p^{\top} \;\; \phi^{\top} \,]^{\top}$ from the joint angles q. Other interesting problem from the practical point of view consists on computing the joint angles q from the end-effector pose $\mathbf{x}$. Its solution is of fundamental importance to transform the desired motion, naturally prescribed to the end-effector in the workspace, into the corresponding joint motion. This is called the inverse-kinematics (IK)

problem and it can be written in a functional form as:

$$q = f^{-1}(\mathbf{x})\,, \tag{2-10}$$

where $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a nonlinear function, obtained by the stacking of the functions $h(\cdot)$ and $g(\cdot)$ respectively.

In general, the IK problem is much more complex for the following reasons: multiple solutions may exist; infinite solutions may exist, e.g., in the case of a kinematically redundant manipulator; there might be no admissible solutions, in view of the manipulator kinematic structure [37]. Finally, since the equations to solve are in general nonlinear, it is not always possible to find a closed-form solution. Usually, it is possible to employ analytical methods for the solution of the IK problem such as kinematic decoupling, Paden-Kahan subproblems and inverse differential kinematics algorithms, as will be introduced below [37, 62].

2.4 Inverse Kinematics-Based Algorithms

Now, suppose that a given trajectory in the Cartesian space is assigned to the robot end-effector in terms of its linear velocity $\dot{p}$ and the initial position $p(0)$. Here, without loss of generality, we assume that the orientation of the robot end-effector is of no concern or it is kept constant during the robot motion. The goal is to compute a feasible trajectory in the joint space described by $q(t)$ and $\dot{q}(t)$ which reproduces the given trajectory. From (2-3), the joint velocities can be obtained via simple inversion of the Jacobian matrix as:

$$\dot{q} = J_p^{\dagger}(q)\,\dot{p}\,, \tag{2-11}$$

and if the initial position of the robot joints $q(0)$ is *known*, joint positions can be computed by simple integrating the joint velocities over time:

$$q(t) = \int_0^t \dot{q}(\tau)\,d\tau + q(0)\,. \tag{2-12}$$

Notice that, the integration can be carried out in discrete time by using numerical integration techniques, such as the Euler method among others. However, it is well-known that reconstruction of joint variables q via numerical integration is subject to drift phenomena of the solution and, thus, the corresponding end-effector position p differs from the desired one p_d.

To overcome this inconvenience we can employ a suitable algorithm that considers the position error in the Cartesian space $e_p := p_d - p$ in the proposed solution. First, we take the time-derivative of such error e_p and according to the time-derivative of forward kinematics map:

$$\dot{p} = J_p(q)\,\dot{q}\,, \tag{2-13}$$

where $J_p(q) \in \mathbb{R}^{m\times n}$ is the position part of the geometric Jacobian, we obtain the following position error dynamics:

$$\dot{e}_p = \dot{p}_d - J_p(q)\,\dot{q}\,. \tag{2-14}$$

Notice that, to devise an *inverse kinematics-based algorithm*, it is necessary to establish the relationship between the joint velocity $\dot{q}$ and the position error e_p in order to obtain a suitable error dynamics and ensure the convergence of the error to zero or a residual set. In particular, the choice of u as a function of q and e_p allow us to find inverse kinematics-based algorithms to satisfy different task requirements [37].

2.4.1 Jacobian (Pseudo-)inverse

Under the assumptions that (i) $J_p(q)$ is a *nonsingular* matrix, (ii) the velocity of the reference trajectory $\dot{p}_d(t)$ is previously known, and considering the kinematic control approach $\dot{q}(t) \approx u(t)$, we design the following control signal

$$u(t) = J_p^{\dagger}(q)\,v_p\,, \qquad v_p = \Lambda\,e_p + \dot{p}_d\,, \tag{2-15}$$

where $J_p^{\dagger} = J_p^{\mathsf{T}}(J_p J_p^{\mathsf{T}})^{-1}$ is the right pseudo-inverse of J_p and $\Lambda_p \in \mathbb{R}^{m\times m}$ is the position gain matrix. Then, (2-15) can be substituted into (2-14) to obtain:

$$\dot{e}_p = \dot{p}_d - J_p(q)\,J_p^{\dagger}(q)\,[\,\Lambda_p\,e_p + \dot{p}_d\,]\,, \tag{2-16}$$

resulting in the following linear error system:

$$\dot{e}_p + \Lambda_p\,e_p = 0\,, \tag{2-17}$$

provided that the kinematic parameter $b_i \in \mathbb{R}$ for $i = 1, 2, \cdots, n_k$ is assumed to be fully *known*. Notice that, if Λ_p is a positive definite matrix the error system is asymptotically stable and the convergence rate depends on the eigenvalues of Λ_p, that is, the larger the eigenvalues, the faster the convergence. Moreover, the term $J_p^{\dagger}(q)$ is introduced to compensate for $J_p(q)$ and linearize the error system, while the term $\dot{p}_d(t)$ ensures the convergence of the position error e_p to zero, independently of the chosen reference trajectory $p_d(t)$ [37]. The feedback control design (2-15) can be easily implemented as illustrated in the block diagram of the inverse kinematics-based algorithm with pseudo-inverse Jacobian (Fig. 2.1).

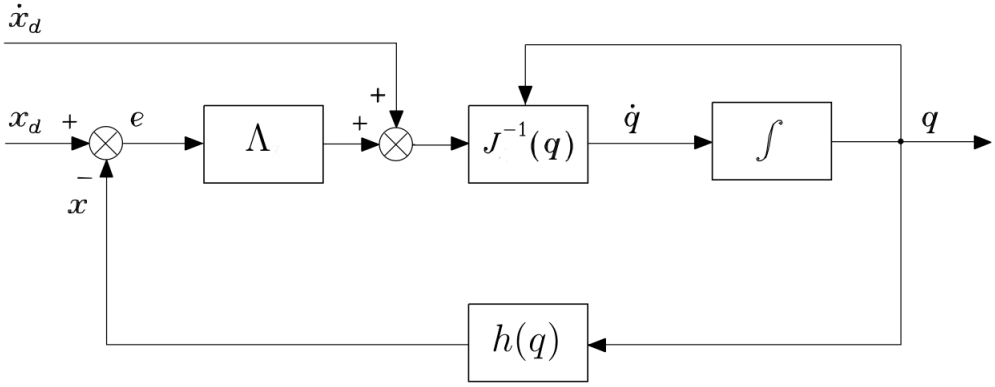

Figure 2.1: Block diagram of the inverse kinematics-based algorithm with pseudo-inverse Jacobian [37].

2.4.2 Kinematic Singularities

The solution of (2-9) can be calculated only when the Jacobian matrix has full rank, that is, when the robot arm is far from singular configurations. Otherwise, its calculation becomes meaningless and the control system (2-8) has linearly dependent equations. It is worth mentioning that, if $\mathbf{v} \in \mathcal{R}(J)$ it is possible to find a feasible solution for (2-9) by using all the linearly independent equations. This means that path assigned to the robot arm, in the Cartesian space, is physically executable even though it is in the vicinity of a singular configuration. Conversely, if $\mathbf{v} \notin \mathcal{R}(J)$, the equations have no solution and it means that the path assigned to the robot arm cannot be executed at the given posture [37]. Therefore, finding all possible singular configurations for a particular robot arm is of great interest since when a singularity occurs the Jacobian matrix $J(q)$ has deficient rank. Singularities can be classified into boundary singularities and internal singularities, as such examples can be found in [37].

It is well known that when the robot arm is in the neighborhood of a singularity, the mobility of the mechanism may be reduced (i.e., loss of degrees of freedom), infinite solutions to the inverse kinematics problem may exist or small velocities in the Cartesian space may cause large velocities in the joint space, which can compromise the successful execution of a given task. In this context, different methods have been proposed to deal with kinematic singularities such as the Damped Least-Squares (DLS) inverse method [38] and the Filtered Inverse method [39]. The DLS inverse algorithm intends to approximate the pseudo-inverse Jacobian matrix $J^{\dagger}(q)$ in the neighborhood of a kinematic singularity by

$$J^{\dagger}(q) \approx J_{dls}(q)\,, \qquad J_{dls}(q) = J^{\top}(q)\,[\,J(q)J^{\top}(q) + \alpha\, I\,]^{-1}\,, \tag{2-18}$$

where *alpha* is a damping factor which makes the inversion better conditioned from a numerical point of view, and it is given by:

$$\alpha = \begin{cases} 0\,, & w_m \geq w_0\,, \\ \alpha_0 \left(1 - \frac{w_m}{w_0}\right), & w_m < w_0\,, \end{cases} \tag{2-19}$$

where $w_m = \sqrt{\det\left(J(q)\,J(q)^\top\right)}$ is the manipulability measure, α_0 is a scaling factor and w_0 is the boundary of the singularity neighborhood [38]. Notice that, in case of kinematic singularities, it is necessary to resort to an inverse kinematics-based algorithm that does not require inversion of the Jacobian matrix.

2.4.3 Jacobian Transpose

A computationally simpler approach consists of finding a suitable relationship between $\dot{q}$ and e_p in order to ensure the convergence of the position error to zero without resorting to the system linearization as discussed in (2-15). Considering the position error dynamics (2-14), we design the following control law

$$u(t) = J_p^\top(q)\,\Lambda_p\,e_p\,. \tag{2-20}$$

Then, considering the kinematic control approach $\dot{q}(t) \approx u(t)$ and substituting (2-20) into (2-14), the position error dynamics is governed by a nonlinear differential equation:

$$\dot{e}_p = \dot{p}_d - J_p(q)\,J_p^\top(q)\,\Lambda_p\,e_p\,. \tag{2-21}$$

In this context, the Lyapunov direct method can be used to analyze the convergence and stability properties of the error system. Let us choose the following Lyapunov function candidate $2V(e_p) = e_p^\top\,\Lambda_p\,e_p$, which is positive definite, that is, $V(0) = 0$ and $V(e_p) > 0$ for $\forall\, e_p \neq 0$. From the time-derivative of $V(e_p)$ and considering (2-21) we obtain:

$$\dot{V}(e_p) = e_p^\top\,\Lambda_p\,\dot{p}_d - e_p^\top\,\Lambda_p\,J_p(q)\,J_p^\top(q)\,\Lambda_p\,e_p\,, \tag{2-22}$$

For the regulation task, that it, $\dot{p}_d = 0$, the Lyapunov function is negative definite $\dot{V}(e_p) < 0$ if and only if J_p has full rank ($\mathcal{N}(J_p^\intercal) = \emptyset$). The condition $\dot{V}(e_p) < 0$ and $V(e_p) > 0$ implies that the system trajectories converges uniformly to $e_p = 0$, that is, the position error system is asymptotically stable. When J_p has deficient rank ($\mathcal{N}(J_p^\top) \neq \emptyset$), the time derivative of $V(e_p)$ is only negative semi-definite, since $\dot{V}(e_p) = 0$ for $e_p \neq 0$ with $\Lambda_p\,e_p \in \mathcal{N}(J_p^\top)$. In such case, the inverse kinematics-based algorithm can get stuck for $\dot{q} = 0$ with $e \neq 0$, which corresponds to the situations where the assigned position $p_d(t)$ is not

reachable from the current joint configuration $q(t)$.

For the tracking task, that is, $\dot{p}_d(t) \neq 0$, it is possible to see that the first term of the right-hand side of (2-22) is not cancelled and nothing can be said about its sign. As a consequence, the asymptotic stability along the assigned position $p_d(t)$ cannot be achieved, but the position error $e_p(t)$ is norm-bounded and converges to a residual set. Then, the larger the norm of Λ_p, the smaller is the norm of e_p, but from the practical point of view there is an upper bound on the norm of Λ_p which depends on the chosen sampling time for the discrete-time implementation [37]. The feedback control design (2-20) can be easily implemented as illustrated in the block diagram of the inverse kinematics-based algorithm with transpose Jacobian (Fig. 2.2).

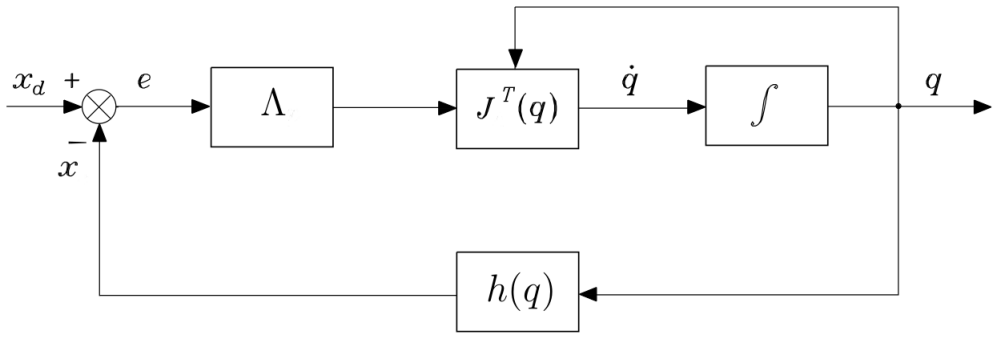

Figure 2.2: Block diagram of the inverse kinematics-based algorithm with transpose Jacobian [37].

2.5 Perturbed Systems

Modeling errors, external disturbances, aging of components, parametric and nonparametric uncertainties are some examples of perturbations that may disrupt a given control system in a realistic situation. Perturbations that do not change the order of a given system, are commonly classified into vanishing and nonvanishing perturbations and are presented later in this section. For both cases, perturbations are represented as an additive term in the right hand side of the state equation [41].

2.5.1 Vanishing Perturbation

A vanishing perturbation consist of any uncertainty or disturbance term that disappears at the origin and it is usually represented by state-dependent functions. In this case, the origin is an equilibrium point of the perturbed system. Considering the presence of kinematic uncertainties at the nominal

model of the robot arm, presented in (2-13), we obtain:

$$\dot{p} = \hat{J}_p(q)\,\dot{q} + \eta_k(q,\dot{q})\,, \tag{2-23}$$

with

$$\eta_k(q,\dot{q}) = [\,J_p(q) - \hat{J}_p(q)\,]\,\dot{q}\,, \tag{2-24}$$

where $\hat{J}_p(q) \in \mathbb{R}^{m\times n}$ denotes the uncertain Jacobian matrix and $\eta_k(q,\dot{q}) \in \mathbb{R}^m$ is the perturbation term due to modeling errors in the robot arm. Notice that, from the practical point of view, the perturbation term $\eta_k(\cdot)$ is assumed to be uncertain, norm-bounded and state-dependent. In addition, we consider that the origin is the only equilibrium point of the nominal system [41].

Then, for the regulation task, that is, $\dot{p}_d = 0$, and considering the kinematic control approach $\dot{q}(t) \approx u(t)$, we obtain the following position error dynamics:

$$\dot{e}_p = -\hat{J}_p(q)\,u(t) - \eta_k(q,\dot{q})\,, \tag{2-25}$$

and in order to perform the exact linearization of the error system, we design the following control law:

$$u(t) = \hat{J}_p^{\dagger}(q)\,\Lambda_p\,e_p\,, \tag{2-26}$$

provided that the robot arm is far away from singular configurations.

Now, in order to analyze the stability and convergence properties of the error system (2-25), we consider the Lyapunov candidate function given by $2V(e_p) = e_p^{\top}\,e_p$, that satisfies,

$$\left\| \frac{\partial V}{\partial e_p} \right\| \leq c\,, \qquad c > 0\,. \tag{2-27}$$

Then, taking the time-derivative of $V(e_p)$ along the system trajectories, we obtain,

$$\dot{V}(e_p) = -e_p^{\top}\,\hat{J}_p(q)\,u(t) - e_p^{\top}\eta_k(q,\dot{q})\,, \tag{2-28}$$

and by using the control law (2-26) yields:

$$\dot{V}(e_p) = -e_p^{\top}\Lambda_p\,e_p - e_p^{\top}\,\eta_k(q,\dot{q})\,. \tag{2-29}$$

Notice that, the first term on the right-hand side of (2-29) is a negative definite quadratic form, which converges asymptotically to zero. Now, let us analyze the second term on the right-hand side of (2-29), which consists of a non quadratic form due to the parametric uncertainty $\eta_k(q,\dot{q})$ and, in general, is indefinite. According to the property (P1), the position part of the Jacobian matrix $J_p(q)$ has a lower and upper limited norms, where the following inequality holds:

$$0 < J_{p_m} \leq ||\,J_p^{\dagger}(q)\,|| \leq J_{p_M} < \infty\,, \quad \forall q\,, \tag{2-30}$$

for some positive constants J_{p_m} and J_{bM}. Then, a choice for $\hat{J}_p(q)$ always exists

which satisfies:

$$|| I - J_p^{\dagger}(q)\,\hat{J}_p(q) || \leq \alpha_J \leq 1\,, \quad \forall q\,. \tag{2-31}$$

where α_J is a positive term. Indeed, by setting $\hat{J}_p(q)$ as:

$$\hat{J}_p = \frac{2}{J_{p_m} + J_{p_M}}\, I\,, \tag{2-32}$$

and from (2-31) implies that:

$$|| J_p^{\dagger}(q)\,\hat{J}_p(q) - I || \leq \frac{J_{p_M} - J_{p_m}}{J_{p_M} + J_{p_m} s} = \alpha_J < 1\,. \tag{2-33}$$

Notice that, if $\hat{J}_p$ is a more accurate estimate of the nominal Jacobian matrix J_p, the above inequality is satisfied with values of α_J that can be made arbitrarily small, wherein when $\alpha_J = 0$ implies that $\hat{J}_p = J_p$. In this sence, by recalling that $\hat{J}_p(q)\,\dot{q}$ in (2-24) is a function of q and $\dot{q}$, without loss of generality the following assumption is made:

$$|| \eta(q, \dot{q}) || \leq \gamma_k || e_p ||\,, \qquad \gamma_k > 0\,, \qquad \forall q, \dot{q}\,, \tag{2-34}$$

where the model-uncertainty $\eta_k(q, \dot{q})$ is a state-dependent function (2-24) that vanishes when $J_p(q) = \tilde{J}_p(q)$ and at the origin $e_p = 0$. If the nonnegative constant γ_k is bounded by

$$\gamma_k < \frac{\lambda_{min}(\Lambda_p)}{c}, \tag{2-35}$$

where $\lambda_{min}(\Lambda_p)$ is real and positive ($\lambda_{min}(\cdot)$ is the minimum eigenvalue of a matrix) since Λ_p is assumed to be symmetric and positive definite, and from (2-29) (satisfying assumptions (2-34) and (2-35)) we obtain:

$$\dot{V}(e_p) \leq -(\lambda_{min}(\Lambda_p) - \gamma_k\, c)\, ||e_p||^2, \qquad (\lambda_{min}(\Lambda_p) - \gamma_k\, c) > 0\,. \tag{2-36}$$

Hence, the origin is semiglobally exponentailly stable [41].

2.5.2 Nonvanishing Perturbation

A nonvanishing perturbation consist of any uncertainty or disturbance term that does not disappear at the origin. In this case, the origin is not an equilibrium point of the perturbed system. In general, this type of perturbation can not be represented by state-dependent functions [41].

Let us consider the same scenario presented in the previous section, as described in (2-23) and (2-24), but now we consider a time-varying reference trajectory $p_d(t)$, previously known, with the control signal $u(t)$ designed in (2-15). By using the kinematic control approach $\dot{q}(t) \approx u(t)$, we obtain the following error dynamics:

$$\dot{e}_p = \dot{p}_d - \hat{J}_p(q)\, u(t) - \eta_k(q, \dot{q})\,. \tag{2-37}$$

In order to analyze the stability and convergence properties of the error system (2-25), we consider the Lyapunov candidate function given by: $2V(e_p) = e_p^\top e_p$. Then, taking the time-derivative of $V(e_p)$ along the system trajectories, expanding the disturbance term due to the uncertain kinematics $\eta_k(q, \dot{q})$ and using the control law,

$$u(t) = J_p^\dagger(q)\, v_p\,, \qquad v_p = \Lambda\, e_p + \dot{p}_d\,, \tag{2-38}$$

yields:

$$\dot{V}(e_p) = -e_p^\top\, \Lambda_p\, e_p - e_p^\top\, [\, J_p(q)\, \hat{J}_p^\dagger(q) - I\,]\, v_p\,. \tag{2-39}$$

Again notice that, the first term on the right-hand side of (2-39) is in a negative definite quadratic form, However, nothing can be said about the sign of the second term. Indeed, due to the presence of parametric uncertainty in the position part of the Jacobian matrix $J_p(q)$ it is not possible to ensure its negative-definiteness. As a consequence, asymptotic convergence of the position error e_p along of a given trajectory $p_d(t)$ cannot be achieved. Notice that, the negative definiteness of $\dot{V}(e_p)$ will depend on the magnitudes of Λ_p, $\dot{p}_d$ and $\eta_k(q, \dot{q})$. It means that the norm of e_p can be reduced by simply increasing the norm of Λ_p as much as possible. Nonetheless , this is a very restrictive condition from the practical point of view.

In this context, to deal with this nonvanishing perturbations it is possible to design an adaptive control scheme for minimizing the error norm in the presence of uncertainties. Another promising solution consists of designing a robust control scheme to deal with this type of disturbance during the execution of the tracking task, as will be shown next.

2.5.2.1 Adaptive Controller

The following subsection illustrates the use of an adaptive control scheme for dealing with the presence of parametric uncertainties into the Jacobian matrix $J_p(q)$ for the tracking problem defined as:

$$p \to p_d(t)\,, \qquad e_p = p_d(t) - p \to 0\,. \tag{2-40}$$

Here, we first assume that the forward kinematics map can be linearly parameterized as

$$p = h(q) = W_k(q)\, b\,, \tag{2-41}$$

where $W_k(q) \in \mathbb{R}^{m \times n_k}$ is the kinematics regressor matrix, $b \in \mathbb{R}^{n_k}$ denotes the constant kinematic parameter vector, and n_k denotes the number of kinematic parameters of the forward kinematics map $h(q)$. Considering the existence of parametric uncertainties in the kinematic model (2-41), the estimated position of the end-effector $\hat{p}$ can be expressed as:

$$\hat{p} = \hat{h}(q) = W_k(q)\,\hat{b}\,, \tag{2-42}$$

where $\hat{h}$ denotes the estimated forward kinematics map and $\hat{b} \in \mathbb{R}^{n_k}$ is the estimated kinematic parameter vector. According to Slotine and Li [60] it is well-known that the linear parameterization models (2-41) and (2-42) can be used to perform an online parametric estimation provided that p and W_k can be measured from the system signals. In this context, we assume that p and q are measurable from a laser tracker and joint encoders respectively.

Now, let $\epsilon \in \mathbb{R}^m$ be the position estimation error computed from the difference between the measurable position p and the estimated position $\hat{p}$ as:

$$\epsilon = p - \hat{p}\,. \tag{2-43}$$

Then, the estimation error (2-43) can be related to the parametric error $\tilde{b}$ using:

$$\epsilon = p - W_k(q)\,\hat{b} = W_k(q)\,\tilde{b}\,, \qquad \tilde{b} = b - \hat{b}\,. \tag{2-44}$$

Due to the presence of uncertainties in the robot kinematics, the control law (2-38) can be rewritten as:

$$u(t) = \hat{J}_p^{\dagger}(q)\,v_p\,, \qquad v_p = \Lambda\,e_p + \dot{p}_d\,, \tag{2-45}$$

where $\hat{J}_p(q) \in \mathbb{R}^{3\times n}$ is the estimated Jacobian matrix related to the position part of the Jacobian matrix $J(q)$ and v_p is the position control signal choose to ensure the proper stability and zero-convergence of the position error e_p which will be proved later on.

Notice that, from the evaluation of the structure of the right pseudo-inverse Jacobian matrix $J_p^{\dagger}(q)$ it is well-known the direct adaptation method cannot be applied to solve the adaptive control problem, since the right-hand side of (2-45) cannot be linearly parameterized [60]. In this case, the kinematic parameters of the robot arm may not be directly estimated by a gradient-type adaptation law. Conversely, the indirect adaptation method is an alternative approach which consists of estimating the kinematic parameters by solving an algebraic equation and, then use the estimates indirectly to compute the control law at each instant of time [60].

Then, considering that the kinematic uncertainties affect only the position coordinates, the differential kinematic equation (2-13) can be linearly parameterized according to the property (P2) as:

$$\dot{p} = J_p(q)\,\dot{q} = W_J(q,\dot{q})\,b\,, \tag{2-46}$$

where $W_J \in \mathbb{R}^{m\times n_k}$ is a kinematic regressor matrix. From the position error (2-40) and adding and subtracting the terms $\hat{J}_p(q)\,\dot{q}$ and $\Lambda_p\,e_p$, the position error equation is given by:

$$\dot{e}_p = \dot{p}_d - \hat{J}_p(q)\,\dot{q} - [\,J_p(q) - \hat{J}_p(q)\,]\,\dot{q}\,. \tag{2-47}$$

Then, using the control law (2-45) and parameterized model (2-46), we obtain:

$$\dot{e}_p = -\Lambda_p\, e_p - W_J(q,\dot{q})\,\tilde{b}\,. \tag{2-48}$$

In this section, we consider an estimation algorithm based on the gradient method to estimate the kinematic parameters of the robot arm. The goal of the estimation algorithm is to update the uncertain parameter vector $\hat{b}$ so that the position error e_p and the estimation error ϵ can be minimized. Then, a composite adaptation law to update $\hat{b}$ can be given by:

$$\dot{\hat{b}} = -\Gamma_k\,[\,W_J^{\top}(q,\dot{q})\,e_p - W_k^{\top}(q)\,\epsilon\,]\,, \qquad \Gamma_k = \Gamma_k^{\top} > 0\,, \tag{2-49}$$

where Γ_k is the adaptation gain matrix. Notice that, the adaptation law (2-49) is said to be composite since it extracts the information about the kinematic parameters from both position and estimation errors simultaneously. In order to analyze the stability and convergence properties of the error system (2-48), we consider the following Lyapunov candidate function given by:

$$V(e_p,\tilde{b}) = e_p^{\top}\,e_p + \tilde{b}^{\top}\,\Gamma_k^{-1}\,\tilde{b}\,. \tag{2-50}$$

Then, taking the time-derivative of $V(e_p,\tilde{b})$ and by using the composite adaptation law (2-49) we obtain:

$$\dot{V}(e_p,\tilde{b}) = -e_p^{\top}\Lambda_p\,e_p - \tilde{b}^{\top}\,W_k^{\top}(q)\,W_k(q)\,\tilde{b}\,. \tag{2-51}$$

Note that, since Λ_p is assumed to be a positive definite matrix, the first term of the right-hand side of (2-51) is negative definite. Thus, $\dot{V}(e_p,\tilde{b})$ is negative definite if and only if the second term of the right-hand side of (2-51) is negative definite, that is, $W_k^{\top}(q)\,W_k(q)$ is positive definite. In fact, it is possible to show that this last condition holds provided that the robot-arm is far from singular configurations.

Therefore, since $V(e_p,\tilde{b}) > 0$ and $\dot{V}(e_p,\tilde{b}) < 0$ it is possible to guarantee that $\lim_{t\to\infty} e_p(t) = 0$ and $\lim_{t\to\infty} \epsilon(t) = 0$ over the time, ensuring a successful execution of the tracking task. Notice that, the composite adaptation scheme not only ensures the overall stability of the adaptive control system, but also leads to rapid parametric convergence and smaller tracking errors [60].

2.5.2.2 Validation Adaptive Controller

The following subsection demonstrates the proposed adaptive controller performance into a tracking task with uncertainty at the last link length of ten percent, of a two-link robot arm. Figs. 2.3-2.6 demonstrate the adaptive

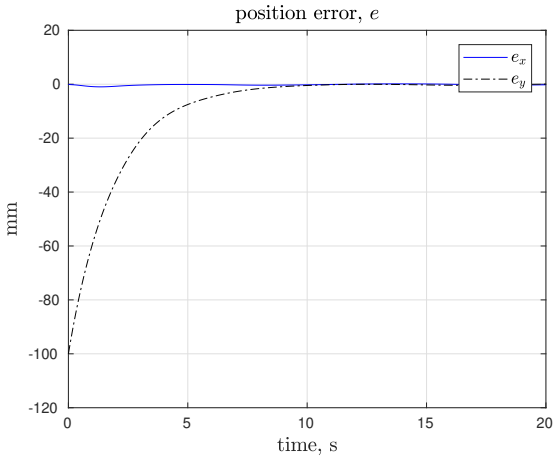

Figure 2.3: Adaptive Controller Position Error Behavior.

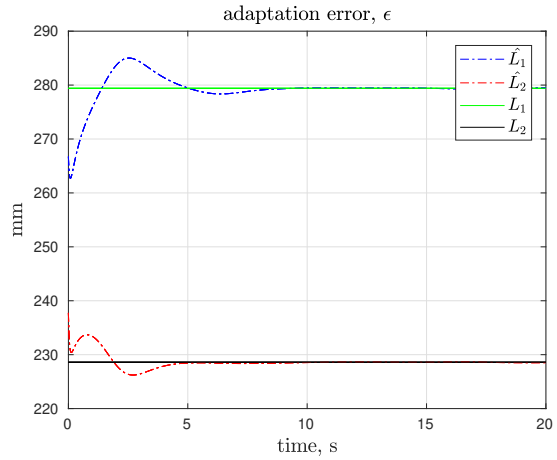

Figure 2.4: Adaptive Controller Parametric Error.

controller performance, where the Cartesian error e_p converges to zero along the time. Moreover, Fig. 2.4 demonstrates the adaptation performance of the proposed adaptation law (2-49), by analyzing the parametric error ϵ performance. Finally, Fig.2.6 demonstrates the smoothness of the velocity control signal $\dot{q}$ applied to the robot joints.

2.5.3 Robust Control

Recalling the same parametric uncertainty introduced in (2-23) and the bounding property presented in (2-33), it is possible to compute a new term Ψ to vanish the perturbation during the task execution (state e_p is out of the equilibrium point) or in the existence imperfect compensation. By augmenting the system order , we obtain

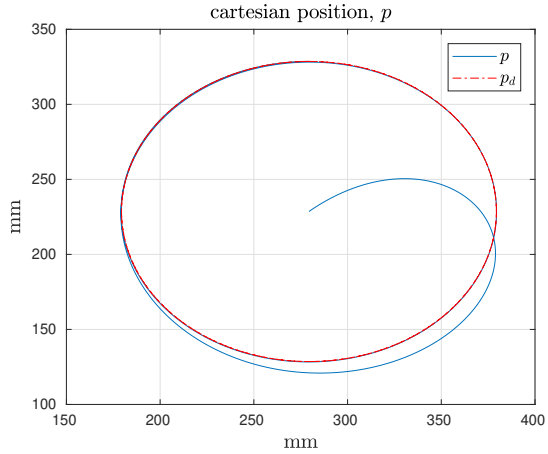

Figure 2.5: Adaptive Controller Cartesian Trajectory.

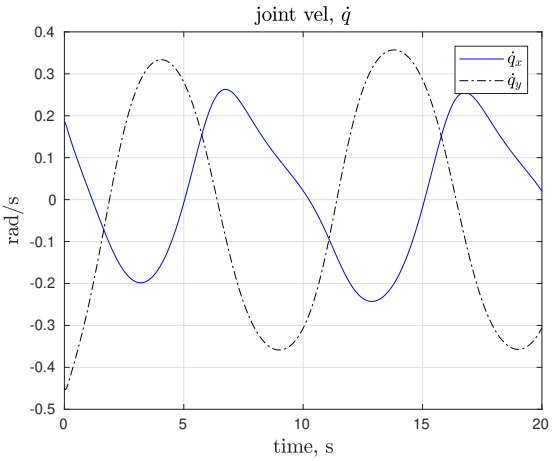

Figure 2.6: Adaptive Controller Joint Velocity.

$$\zeta = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{2-52}$$

as the system state, where $x_1 = \int_0^t e_p(\tau)d\tau$ and $x_2 = e_p$. Then, the control system can be represented by the following closed-loop differential equation

$$\dot{\zeta} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ I \end{bmatrix} \left(\dot{p}_d - \hat{J}(q)\, u(t) + \eta_k(q, \dot{q}) \right), \tag{2-53}$$

and taking the following control law

$$u(t) = \hat{J}^{\dagger}(q) \left(\dot{p}_d + \begin{bmatrix} \Lambda_{p1} & \Lambda_{p2} \end{bmatrix}^{\top} \zeta + \Psi \right), \tag{2-54}$$

where $\Lambda_{p1}, \Lambda_{p2} \in \mathbb{R}^{3\times3}$ are positive definite gain matrices, and substituting (2-54) into (2-53), yields

$$\dot{\zeta} = \begin{bmatrix} 0 & I \\ -\Lambda_{p1} & -\Lambda_{p2} \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ I \end{bmatrix} (\eta_k(q, \dot{q}) - \Psi) \,. \tag{2-55}$$

Noting that $\eta_k(q, \dot{q})$ is a function of ζ, it is possible to assume that

$$\eta_k(q, \dot{q}) < \Phi < \infty \qquad \forall \zeta \,, \tag{2-56}$$

where q and $\dot{q}$ are bounded due to the existence of joint ranges and saturation at the joints velocity. In other words infinity positions and velocities are not considered in practice.

Considering the Lyapunov candidate function

$$V(\zeta) = \zeta^\top Q \zeta > 0 \qquad \forall \zeta \neq 0 \,, \tag{2-57}$$

where $Q \in \mathbb{R}^{6\times6}$ is a positive definite matrix, deriving (2-57) we obtain

$$\dot{V}(\zeta) = \zeta^\top (A^\top Q + QA)\zeta + 2\zeta^\top QB(\eta_k(q, \dot{q}) - \Psi) \,. \tag{2-58}$$

Since A has eigenvalues with negative real part, it is necessary to compute a positive definite matrix P as:

$$A^\top Q + QA = -P \,, \tag{2-59}$$

injecting (2-59) into (2-58), $\dot{V}(\zeta)$ leads to

$$\dot{V}(\zeta) = -\zeta^\top P \zeta + 2\zeta^\top QB(\eta_k(q, \dot{q}) - \Psi) \,. \tag{2-60}$$

Regarding the left term of (2-60), it is possible to see that it is negative definite for any state ζ value, regarding the second term, it is necessary to choose Ψ for vanishing the parametric uncertain $\eta_k(q, \dot{q})$ along the task execution [42].

Adopting $\Psi(Z)$, where

$$Z = B^\top P \zeta \,. \tag{2-61}$$

The $\Psi(Z)$ vector can be computed based on a classic Siliding Mode Unit Vector Control scheme (SM-UVC) approach [42], as:

$$\Psi = \frac{\rho}{||Z||} Z \qquad \rho > 0 \,, \tag{2-62}$$

gives

$$\begin{aligned} Z^T(\eta_k(q, \dot{q}) - \Psi) &\leq ||Z||\, ||\eta_k(q, \dot{q})|| - \rho\, ||Z|| \\ &= ||Z||\, (||\eta_k(q, \dot{q})|| - \rho) \,. \end{aligned} \tag{2-63}$$

Hence, choosing $\rho \geq ||\eta_k(q, \dot{q})||$, it is possible to vanish the *perturbed term* $\eta_k(q, \dot{q})$, guaranteeing the asymptotic stability of 2-60.

$$\dot{V}(\zeta) = -\zeta^\top P \zeta + 2\zeta^\top QB \left(\eta_k(q, \dot{q}) - \frac{\rho}{||Z||} Z \right) < 0 \qquad \forall \zeta \neq 0 \,. \tag{2-64}$$

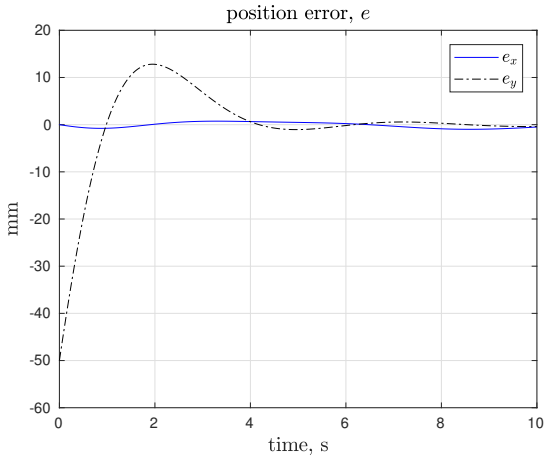

Figure 2.7: UVC Position Error Behavior.

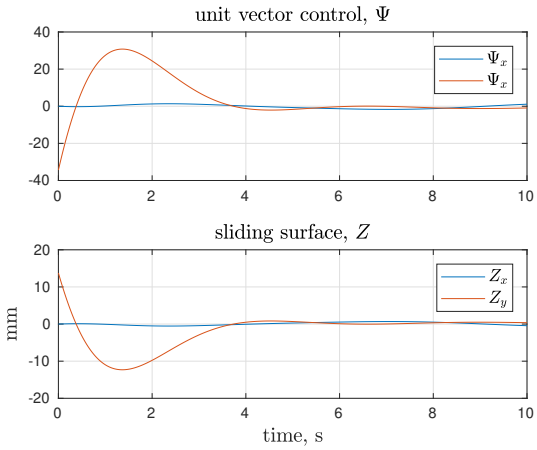

Figure 2.8: UVC Sliding Surface and unit vector control signal.

2.5.3.1 Validation SM-UVC

In this section we demonstrate the introduced UVC controller to perform a tracking task with uncertainty at the last link length of ten percent, of a two-link robot arm. Figs. 2.7-2.10 demonstrate the UVC controller performance, where the Cartesian error e_p remains close to zero along time. Moreover, Fig. 2.8 demonstrates the unit vector control $\Psi(Z)$ and the sliding surface Z performance of the proposed algorithm in (2-62). Finally, Fig.2.10 demonstrates the smoothness of the velocity control signal applied to the robot joints.

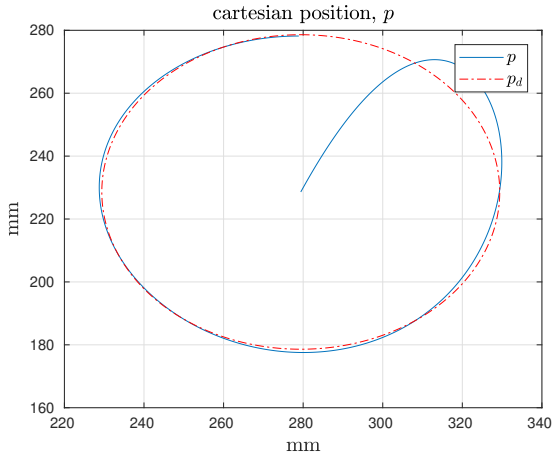

Figure 2.9: UVC Cartesian Trajectory.

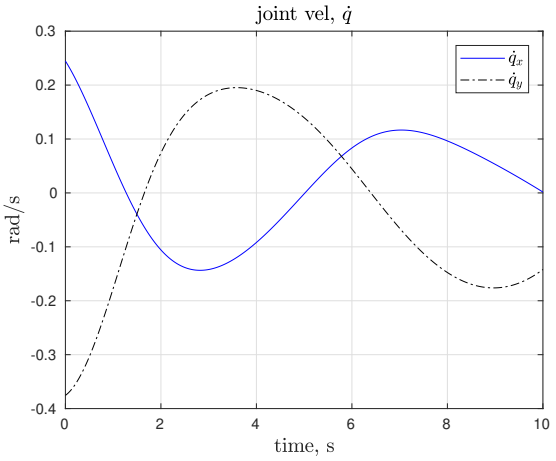

Figure 2.10: UVC Joint Velocity.

2.5.4 Nonvanishing Perturbation

Considering the Lyapunov candidate function $2V(e_p) = e_p^\top e_p$ and the nominal system introduced in (2-14), for tracking a trajectory p_d obtained by an external sensor e.g. camera, laser, etc. Where $\dot{p}_d$ is not available or very noisy, and the links-length $\eta_k(q, \dot{q})$ of the robot arm, with the following control law

$$u(t) = \hat{J}^\dagger(q)[\Lambda_p\, e_p]\,, \tag{2-65}$$

$\dot{V}(e_p)$ leads to

$$\dot{V}(e_p) = -e_p^\top\, \Lambda_p\, e_p - e_p^\top\, [J(q)\, \hat{J}^\dagger(q) - I]\, \Lambda_p\, e_p + e_p^\top\, \dot{p}_d\,. \tag{2-66}$$

From (2-29) and (2-33), it is possible to notice that the first two terms from left to right are quadratic and negative definite. In view that the tracking task considers an uncertain trajectory $\dot{p}_d$, the last term $e_p^\top\, \dot{p}_d$ cannot be canceled

by the control law proposed in (2-65). Hence, the system behavior over the time t is not asymptotically stable when performing any tracking task using the regulation control law (2-65).

2.5.5 Robust Extended Control UVC

Similarly to the tracking task of an uncertain trajectory noted in (2-66) with a second-order system denoted in (2-53), and the following control law

$$u(t) = \hat{J}^{\dagger}(q)\,\left(\left[\begin{array}{cc}\Lambda_{p1} & \Lambda_{p2}\end{array}\right]\zeta + \Psi\right), \tag{2-67}$$

we obtain

$$\dot{\zeta} = \left[\begin{array}{cc} 0 & I \\ -\Lambda_1 & -\Lambda_2 \end{array}\right]\zeta + \left[\begin{array}{c} 0 \\ I \end{array}\right](\dot{p}_d + \eta_k(q,\dot{q}) - \Psi)\,. \tag{2-68}$$

By recalling the property of $\eta_k(q,\dot{q})$ from (2-33), and assuming that $\dot{p}_d$ is a *bounded term*, as

$$\sup_{t\geq 0}||\dot{p}_d|| < Q_M < \infty \qquad \forall \dot{p}_d. \tag{2-69}$$

where infinity trajectories $\dot{p}_d$ are despised. In the same way that the Lyapunov candidate function was used in (2-57) and the P matrix in (2-59), $\dot{V}(\zeta)$ leads to

$$\dot{V} = -\zeta^{\top}P\zeta + 2\zeta^{\top}QB(\dot{p}_d + \eta_k(q,\dot{q}) - \Psi). \tag{2-70}$$

Notably the left term of (2-70) is negative definite, regarding the other term, Ψ needs to be chosen for vanishing the uncertain term $\eta_k(q,\dot{q})$ and $\dot{p}_d$ when they are different to zero.

By setting a sliding region $Z \in \mathbb{R}^{3\times 1} = 2\,B^T\,Q\,\zeta$, the *perturbed term* can be rewritten as $Z^T(\dot{p}_d + \eta_k(q,\dot{q}) + \Psi)$. The Ψ control vector can be computed as,

$$\Psi = \frac{\rho}{||Z||}\,Z + \varrho Z \qquad \rho, \varrho > 0, \tag{2-71}$$

substituting (2-71) into the *perturbed term* gives

$$Z^T((\dot{s}_d + \eta_k(q,\dot{q})) - \Psi) = Z^T(\dot{s}_d + \eta_k(q,\dot{q})) - \frac{\rho}{||Z||}\,Z^T\,Z - \varrho\,Z^T\,||Z||, \tag{2-72}$$

By choosing ρ and ϱ, as

$$||(\dot{s}_d + \eta_k(q,\dot{q}))|| <\; Z\left(\frac{\rho}{||Z||} + \varrho\right), \tag{2-73}$$

it is possible to vanish the *perturbed term* $(\dot{s}_d + \eta_k(q,\dot{q}))$, guaranteeing the asymptotic error convergence over the time t.

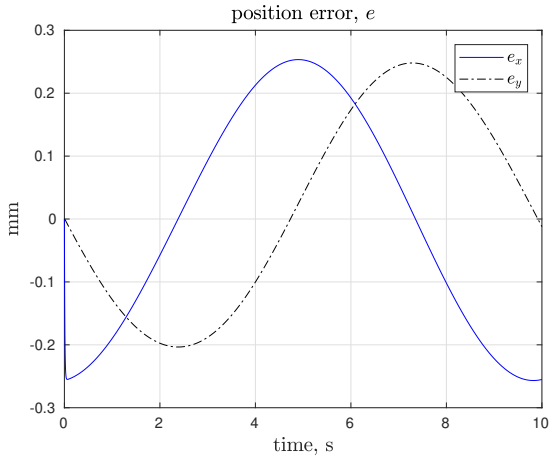

Figure 2.11: UVC Position Error Behavior.

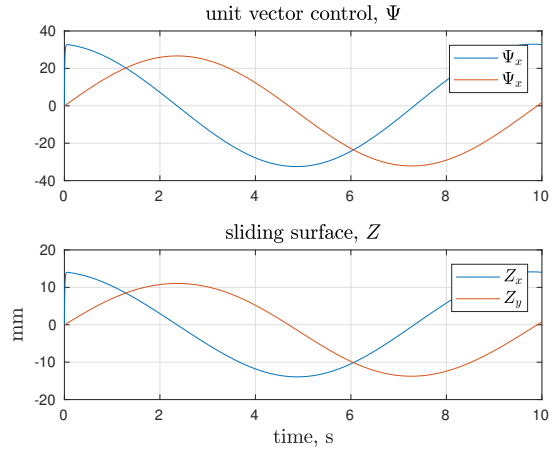

Figure 2.12: UVC Sliding Surface and unit vector control signal.

2.5.5.1 Validation Extended SM-UVC

In this section we demonstrate the introduced UVC controller to perform a tracking task with uncertainties at the trajectory $\dot{p}_d$ and the last link length of ten percent$\hat{J}(q)$, of a two-link robot arm. Figs. 2.11-2.14 demonstrate the UVC controller performance, where the Cartesian error e remains very close to zero along time even when the desired trajectory $\dot{p}_d$ is uncertain. Moreover, Fig. 2.12 demonstrates the unit vector control Ψ and the sliding surface Z performance of the proposed algorithm in (2-71). Finally, Fig.2.14 demonstrates the smoothness of the velocity control signal applied to the robot joints in order to perform the tracking task.

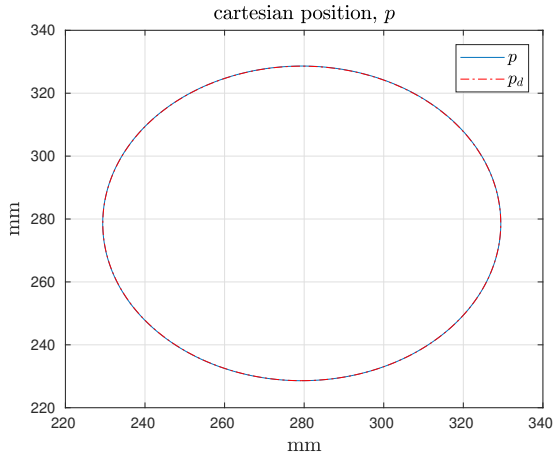

Figure 2.13: UVC Cartesian Trajectory.

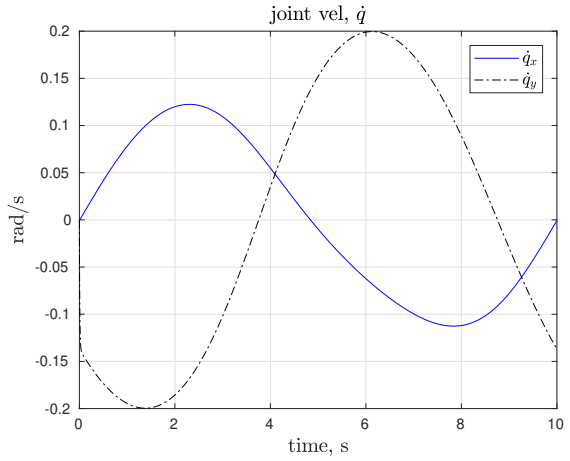

Figure 2.14: UVC Joint Velocity.

2.6 Concluding Remarks

In this chapter we have introduced robot controller methodologies based on inverse Jacobian schemes, and how the presence of parametric and non-parametric uncertainties into the robot model may cause marginal stability during a tracking task even if $\dot{p}_d$ is known (2-39). These parametric uncertainties can be minimized or vanished by introducing an adaptive controller for ensuring the error convergence to zero (2.4). On the other hand, in cases of non-parametric uncertainties or external disturbances, a robust controller is a more suitable choice for ensuring the system stability, in this chapter we have demonstrated a robust controller based on sliding surface methodology.

3
Visual Servoing Systems

Visual servoing or vision-based control consists of using visual information obtained from a single or multiple cameras to control the pose of a robot arm with respect to a target object or a set of image features. We also could consider a mobile robot, wherein the visual servoing problem is to control the pose of the vehicle with respect to a number of landmarks. In computer vision, an *image feature* may be any geometric characteristic that could be extracted from a given image. In general, an image feature can be obtained from the projection of a physical feature of the target object onto the camera image plane. Among the image features that have been used for vision-based control we have, for example, the centroid coordinates, the area of a projected surface, the parameters of lines, an ellipse or the distance between two points in the image space [12].

3.1
Visual Servoing Schemes

Visual servoing theory meets two classical approaches position-based visual servoing (PBVS) and image-based visual servoing (IBVS), there also may exist a third hybrid approach which combines both methods. The main differences between PBVS and IBVS approaches are related to how the pose of the target object is obtained (e.g., pose estimate or feature measurements) and which coordinates space the output error is computed (e.g., task or image). The main advantages of using the visual information directly into a feedback control loop are twofold: (i) there is no need to estimate the pose of the target frame $\mathcal{F}_t$ with respect to the camera frame $\mathcal{F}_c$ (i.e., the object pose) in real-time; (ii) the lower sensitivity to calibration errors in the camera [12].

3.2
Position-based Visual Servoing Control Design

Consider the pose control problem for a n-DoF robot arm. Here, we assume that the task of interest is to move the robot end-effector towards a fixed target, located at the robot workspace (i.e., regulation task) using the eye-in-hand camera configuration. The control goal is to regulate the current

pose of the robot end-effector $\mathbf{x}=[\,p^\top\ \phi^\top\,]^\mathsf{T}$ - estimated by the camera - to a constant desired pose $\mathbf{x}_d=[\,p_d\ \ \phi_d\,]^\mathsf{T}$, assumed to be bounded, that is:

$$\lim_{t\to\infty}\begin{bmatrix} p(t) \\ \phi(t) \end{bmatrix} = \begin{bmatrix} p_d \\ \phi_d \end{bmatrix} . \tag{3-1}$$

The position error e_p is defined as $e_p := p_d - p$, however the orientation error e_o should be defined in terms of the algebra of rotation groups, instead of the vector algebra [37]. Thus, considering the *unit-quaternion representation*, it is usual to define the orientation error e_o as the vector part of the error quaternion:

$$e_o := \Delta\epsilon = \eta(q)\epsilon_d - \eta_d\epsilon(q) - Q(\epsilon_d)\epsilon(q)\,, \tag{3-2}$$

where the pair $\phi_d=\{\eta_d, \epsilon_d\}$ denotes respectively the scalar and vector parts of the desired quaternion. The error quaternion is defined as $\Delta\phi := \{\Delta\eta, \Delta\epsilon\} = \phi_d * \phi^{-1}$, where the symbol "$*$" denotes the quaternion product operator. Notice that, we have $\Delta\phi = \{1, 0^\mathsf{T}\}$ if and only if the quaternions ϕ and ϕ_d are coincident, which means that the corresponding coordinates frames are aligned.

Now, we are able to compute the velocity control signal u using a control algorithm based on $J^\star$ which may denote the Jacobian pseudo-inverse or transpose:

$$u(t) := J^\star(q)\begin{bmatrix} v_p \\ v_o \end{bmatrix} = J^\star(q)\begin{bmatrix} \Lambda_p\, e_p \\ \Lambda_o\, e_o \end{bmatrix} , \tag{3-3}$$

where $\Lambda_p = \Lambda_p^\mathsf{T} > 0$ and $\Lambda_o = \Lambda_o^\mathsf{T} > 0$ are the position and orientation gain matrices, assumed to be definite positive. The stability and convergence analysis of the kinematic control approach is based on the Lyapunov stability theory and can be found in [43].

3.3 IBVS Approach

Here, we consider a visual servoing approach using an RGB-D stereo camera attached to the robot end-effector. Let $p_c = [\,x_c\ y_c\ z_c\,]^\mathsf{T}$ be the coordinates of a 3D point expressed in the camera frame $\mathcal{F}_c$. From the perspective projection model, the 3D point is projected in the image space as a 2D point with the coordinates $p_v=[\,x_v\ \ y_v]^\mathsf{T}$ expressed in pixels, say:

$$\begin{bmatrix} x_v \\ y_v \end{bmatrix} = \frac{f}{z_c}\begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix}\begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} x_{v0} \\ y_{v0} \end{bmatrix} , \tag{3-4}$$

where $\{x_{v0}, y_{v0}, f, \alpha_x, \alpha_y\}$ is the set of camera intrinsic parameters: x_{v0} and y_{v0} are the coordinates of the principal point; f is the focal length; α_x and α_y are the scaling factors in pixel per millimeter. The 3D point is projected in the

image plane as a 2D point with normalized coordinates $p_p = [\, x_p \;\; y_p]^\mathsf{T}$ given by:

$$x_p = \frac{x_v - x_{v0}}{f\alpha_x}\,, \qquad y_p = \frac{y_v - y_{v0}}{f\alpha_y}\,. \tag{3-5}$$

Now, we suppose that the robot end-effector is moving with linear velocity $v_c \in \mathbb{R}^3$ and angular velocity $\omega_c \in \mathbb{R}^3$ both expressed with respect to the (instantaneous) camera frame $\mathcal{F}_c$. Then, using the well-known relationship of *velocity transformation* between the target frame $\mathcal{F}_t$ and the camera frame $\mathcal{F}_c$, we obtain the following motion equation [12]:

$$\dot{p}_{ct} = -v_c - Q(\omega_c)\, p_{ct}\,, \tag{3-6}$$

with $v_c = \bar{R}_{bc}^\mathsf{T}\, \dot{p}_{bc}$ and $\omega_c = \bar{R}_{bc}^\mathsf{T} Q({}^b\omega_{bc})$. From the analysis of the camera-image coordinate transformation (3-4), we can observe that by using a *bidimensional image* may not be possible to obtain any explicit information on depth coordinate. Thus, we can not explicitly compute the depth between the target frame $\mathcal{F}_t$ and the camera frame $\mathcal{F}_c$ by using a single camera. This information, however, can be recovered (i) *indirectly*, using the image projected area of the object or (ii) *directly*, using a stereo vision system.In the indirect approach, the key idea is to use a target with spherical geometry so that the projected area in the image space becomes invariant with respect to the object rotations [12] Let $a_v \in \mathbb{R}^+$ be the projected area of the target object expressed in the image frame $\mathcal{F}_v$. The dynamics of the *depth-to-area transformation* is given by:

$$\dot{a}_v = \frac{-2a_v}{z_c}\, \dot{z}_c\,. \tag{3-7}$$

Here, the following two assumptions can be considered: (A3) The image projected area a_v is bounded and satisfies the inequality $0 < a_{min} < a_v(t) < a_{max}$ for all time t; (A4) The sign of z_c is assumed to be constant and known. Indirect depth estimation is employed to increase the system depth range in situations where the image feature is out of field-of-view of the right camera. Taking the time-derivative of (3-5) and using (3-6) yields:

$$\dot{s} = L_s(s)\, \mathbf{v}_c\,, \qquad \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{a}_v \end{bmatrix} = L_s(s) \begin{bmatrix} v_c \\ \omega_c \end{bmatrix}\,, \tag{3-8}$$

with

$$L_s(s) = \begin{bmatrix} -\dfrac{1}{z_c} & 0 & \dfrac{x_p}{z_c} & x_p\, y_p & -(1+x_p^2) & y_p \\ 0 & -\dfrac{1}{z_c} & \dfrac{y_p}{z_c} & (1+y_p^2) & -x_p\, y_p & -x_p \\ 0 & 0 & \dfrac{2\, a_v}{z_c} & 2\, a_v\, y_p & -2\, a_v\, x_p & 0 \end{bmatrix}\,,$$

where $L_s(s) \in \mathbb{R}^{3\times 6}$ is the *interaction matrix* related to the state vector $s \in \mathbb{R}^3$, composed of the 3D point coordinates expressed in the image space.

3.4 IBVS control design

Here, we assume that the control goal is to drive a set of features s to a desired set value s_d say:

$$s \to s_d\,, \qquad e_s := s_d - s \to 0\,, \tag{3-9}$$

where $e_s \in \mathbb{R}^3$ is the image feature error. From the differential kinematics equations (2-8) and (3-8), we obtain the following control system:

$$\dot{s} = L_s(s)\,\bar{R}_{bc}^{\mathsf{T}}\,J(q)\,u\,. \tag{3-10}$$

From the time-derivative of (3-9), we define the velocity control signal u as:

$$u := J^{\dagger}(q)\,\bar{R}_{bc}\,L_s^{\dagger}(s)\,\Lambda_s\,e_s\,, \qquad \Lambda_s = \Lambda_s^{\mathsf{T}} > 0\,, \tag{3-11}$$

where Λ_s is a positive definite gain matrix, which ensures the asymptotic convergence of the image feature error e_s to zero, that is, $\lim_{t\to\infty} e_s(t) = 0$, provided that:

- (i) the robot arm is far from singular configurations
- (ii) the calibration parameters of the camera-robot system are fully known.

3.4.1 Vanishing Perturbation

In general, this last condition regarding the fully know of the intrinsic and extrinsic camera-robot model, may not be satisfied into a practical scenario, as a result of this uncertainty, an approximation or an estimation of the interaction the matrix $L_s(s)$ as well as the Jacobian matrix $J(q)$ needs to be realized during the task execution [12]. In this context, the IBVS control system (3-10), in the presence of modeling errors or uncertainties, can be simply defined as:

$$\dot{s} = \hat{L}_s(s)\,\bar{R}_{bc}^{\mathsf{T}}\,\hat{J}(q)\,u + \eta_v(s, q, \dot{q})\,, \tag{3-12}$$

where $\eta_v(\cdot)$ is a *state-dependent* nonlinear modeling uncertainty, since $L_s(s)$ is a matrix with lower and upper limited norms, the following inequality holds,

$$L_{s_m} \leq ||L_s^{\dagger}(s)|| \leq L_{s_M} < \infty \tag{3-13}$$

Then $\hat{L}_s(s)$ always satisfies

$$\hat{L}_s(s) = \frac{2}{L_{s_m} + L_{s_M}} I, \tag{3-14}$$

now, the interaction matrix uncertainty can be bounded by positive term α_L as

$$||L_s(s)\hat{L}_s^{\dagger}(s) - I|| \leq \frac{L_{s_M} - L_{s_m}}{L_{s_M} + L_{s_m}} = \alpha_L \leq 1 \qquad \forall\, e_s. \tag{3-15}$$

Here, we also assume that $\eta_v(\cdot)$ is a nonlinear function that vanishes at origin, locally Lipschitz in $s, q, \dot{q}$ and uniformly in time t for all $t \geq 0$ [41]. Notice that, without loss of generality, we represent the perturbation term $\eta_v(\cdot)$ as an additive bounded term,

$$||\eta_v(s, q, \dot{q})|| \leq \gamma_v ||e_s|| \,, \quad \gamma_v > 0 \,, \tag{3-16}$$

on the right-hand side of the system equation (3-10). Therefore, an approximation or an estimation of the interaction matrix $L_s(s)$, the rotation matrix R_{bc} as well as the Jacobian matrix $J(q)$ must be considered for the stability analysis [12]. In this case, the velocity control signal (3-11) takes the form:

$$u := \hat{J}^*(q)\, \hat{\bar{R}}_{bc} \hat{L}_s^*(s)\, \Lambda_s\, e_s \,, \tag{3-17}$$

where $\hat{\bar{R}}_{bc} = diag\{\hat{R}_{bc}, \hat{R}_{bc}\}$, $\hat{R}_{bc} = R_{be}(q)\hat{R}_{ec}(\varphi)$, and $\varphi \in \mathbb{R}$ is the misalignment angle between the camera frame $\mathcal{F}_c$ and the end-effector frame $\mathcal{F}_e$ around the z-axis, assumed to be uncertain. Notice that, the estimated Jacobian matrix can be obtained from the DLS inverse method, say $\hat{J}^*(q) = \hat{J}^{\mathsf{T}}(q)(\hat{J}(q)\hat{J}^{\mathsf{T}}(q) + \alpha\, I)^{-1}$ where α is a damping factor that avoids the ill-conditioning problem of the Jacobian matrix.

Here, we also assume that (A3) $\hat{J}(q)\, \hat{J}^*(q)$ is bounded for all possibles values of q, that is $||\hat{J}(q)\, \hat{J}^*(q)|| = c_1$ for $\forall q \in [0, 2\pi]$ where $0 < c_1 \in \mathbb{R}$ is a known positive constant, because $J(\cdot)$ depends on $q(t)$ as the argument of bounded trigonometric functions; (A4) $R_{bc}^{\top}\, \hat{R}_{bc} \in \mathbb{SO}(3)$ is bounded for all possible values of q and φ, that is $||\bar{R}_{bc}^{\top}\, \hat{\bar{R}}_{bc}|| = c_2$, where $0 < c_2 \in \mathbb{R}$ is a known positive constant; (A5) $\hat{L}_s(s)\, \hat{L}_s^*(s)$ is bounded for all possible values of s, that is $||\hat{L}_s(s)\, \hat{L}_s^*(s)|| = c_3$, where $c_3 > 0 \in \mathbb{R}$ is a known positive constant, because we previously assumed that $\eta_v(\cdot)$ is state-dependent and vanishes at origin [41].

To analyse the stability and convergence properties of the IBVS control scheme, we employ the Lyapunov stability formalism. Here, we consider the following positive-definite candidate Lyapunov function given by $V(e_s) = e_s^{\mathsf{T}}\, P\, e_s$, with $P = P^{\top} > 0$. The time-derivative of V along the system trajectories (3-9), (3-12) and (3-17), is given by

$$\dot{V}(e_s) = -2\, e_s^{\top}\, P\, \hat{L}_s(s)\, \bar{R}_{bc}^{\top}\, J(q)\, \hat{J}^*(q)\, \hat{\bar{R}}_{bc}\, \hat{L}_s^*(s)\, \Lambda_s\, e_s + 2\, e_s^{\top}\, P\, \eta_v(s, q, \dot{q}) \,. \tag{3-18}$$

Then, for $P = (1/2)\, I$, the first term of the right-hand side of $\dot{V}(e_s)$ is negative

definite. Since the second term, due to the effect of the perturbation, satisfies (3-15) we have

$$\dot{V}(e_s) \leq -c_1\, c_2\, c_3\, \lambda_{min}(\Lambda_s)||e_s||^2 + \gamma_v\, ||e_s||^2\,. \tag{3-19}$$

Notice that, the time-derivative of V is definite negative if the matrices $\hat{J}^*(q)$ and $\hat{L}_s^*(s)$ are assumed to be full-rank, which can be guaranteed respectively if the robot arm has redundant degrees of freedom and the IBVS control scheme uses one or more than two image features. The condition $\dot{V} < 0$ with $V > 0$ implies that the system trajectories uniformly converge to the origin, that is, the error system is exponentially stable. Moreover, since the robot kinematics and camera calibration intrinsic parameters are positive values, the presence of uncertainties in any or both matrices, $J^*(q)$ and $L_s^*(s)$, is not capable to violate the condition of negative definiteness (3-18).

3.4.2
Uncertain camera-robot System IBVS

Evaluating (A4) into the first part of the system (3-18), where $\hat{J}^*(q)$ and $\hat{L}_s^*(s)$ are chosen correspondingly $\hat{J}^\dagger(q)$, $\hat{L}_s^\dagger(s)$ for simplifying purposes, obtaining:

$$\dot{V}(e_s) = -\, e_s^\top\, \bar{R}_{ce}^\top(\varphi)\, \bar{R}_{ce}(\hat{\varphi})\, \Lambda_s\, e_s \tag{3-20}$$

Now evaluating $R_{ce}^\top(\varphi)\, R_{ce}(\hat{\varphi})$at a planar misalignmenta around z-axes:

- $|\tilde{\varphi}| < \frac{\pi}{2}$ *rad*, the principal minors of $R_{ce}^\top(\varphi)\, R_{ce}(\hat{\varphi})$ are positive and have positive and negative complex parts, turning the system into stable.
- $|\tilde{\varphi}| = \frac{\pi}{2}$ *rad*, the principal minors of $R_{ce}^\top(\varphi)\, R_{ce}(\hat{\varphi})$ have zero real part and positive-negative complex parts, turning the sytem into marginally stable.
- $|\tilde{\varphi}| > \frac{\pi}{2}$ *rad*, the principal minors of $R_{ce}^\top(\varphi)\, R_{ce}(\hat{\varphi})$ are negative and have positive and negative complex parts, turning the system into unstable.

Eventually, the misalignment angle $\tilde{\varphi}$ between the camera and the end-effector frames needs to be less than $\frac{\pi}{2}$ rad,that is

$$|\tilde{\varphi}| < \frac{\pi}{2}\,, \tag{3-21}$$

in order to ensure the positive-definiteness property of the matrix $\hat{R}_{ec}(\hat{\varphi})$.

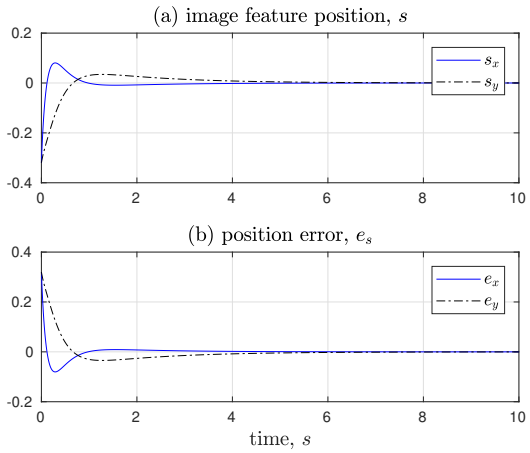

Figure 3.1: IBVS Position Error Behavior.

3.4.2.1 Validation Perturbed IBVS System

In this section, we demonstrate the introduced IBVS controller, configured as:

$$u := \hat{J}^{\top}(q)\,\hat{R}_{bc}\hat{L}_s^{\top}(s)\,\Lambda_s\,e_s\,, \tag{3-22}$$

to perform a regulation task with:

- Misalignment angle $\tilde{\varphi}$ between the camera and the end-effector frames around the z-axis of $-\pi/6$ as commented in property (3-21).
- Uncertainty of ten percent into the intrinsic camera parameters.
- Uncertainty of ten percent at last link-length of the robot arm.

Simulations were carried out with a Denavit-Hartenberg representation of the Mitsubishi RV-2AJ robot arm [37]. Figs. 3.1-3.3 demonstrate the IBVS controller performance, where the feature error e_s converge to zero over the time. Moreover, Fig. 3.2 demonstrates the feature trajectory into the image plane and the successful performance of the algorithm proposed in (3-22). Finally, Fig.3.3 demonstrates the smoothness of the velocity control signal applied to the robot manipulator.

3.4.3 Experimental Validation PBVS - IBVS

In this section, we present experimental results for a fruit harvesting task carried out in a strawberry farm (Sylling, Norway) in order to illustrate the effectiveness of the proposed visual servoing scheme (Fig. 3.4). We consider a RGB-D stereo camera attached to the end-effector of a 5-DoF robot arm, which

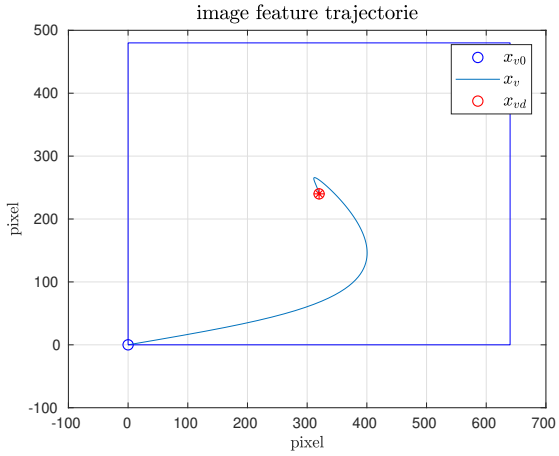

Figure 3.2: IBVS Image Trajectory: initial position: blue "∘", desired position: red "∗" and final position: red "∘".

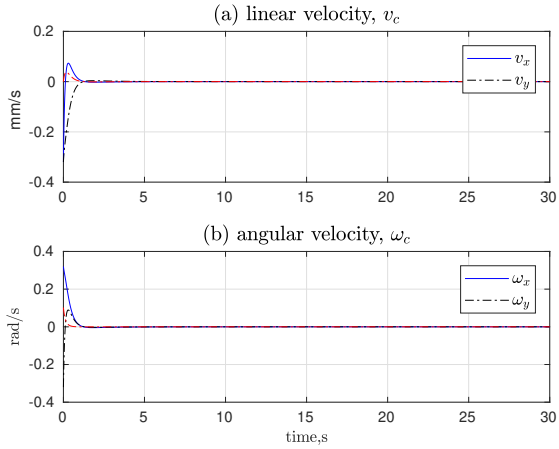

Figure 3.3: IBVS Manipulator Velocity.

is visually controlled at the velocity level by an image-based visual servoing approach in the neighborhood of singular configurations.

Numerical simulations were carried out on a Lenovo laptop with Intel Core i7-6500U Processor (4M Smart Cache, 2.5 GHz) 16 GB RAM, running Windows OS 64 bits. The control algorithm was implemented in MATLAB/Simulink (The MathWorks Inc.) Release 2017a and V-REP PRO EDU version 3.4.0 used as a robotic simulation platform. A set of scripts and function blocks were created to perform all necessary calculations and execute the control loop. Experiments were carried out using the Mitsubishi robot RV-2AJ and the controller Mitsubishi Melfa CR1. The communication channel between the laptop and the controller was established through RS-232C command protocol by means of an own-built cable, based on manufacturer manual. The camera attached to the end-effector was the ZED 2K Stereo Camera dis-

tributed by STEREOLABS and simple gripper was built ad-hoc using a pair of blades driven with a 24-VDC ABB motor. The parameters of the visual servoing system are $f = 10\ mm$, $\alpha_x = \alpha_y = 100\ pixel\, mm^{-1}$, $x_{v0} = y_{v0} = 500\ pixel$, $z_0 = 0.4640\ m$. The Denavit-Hartenberg parameters [37] of the robot arm are: $\alpha_1 = -\pi/2$, and $\alpha_4 = \pi/2$ and the offsets of joints two and four are $-\pi/2$ and $\pi/2$, where all angles are expressed in radians; $a_2 = 0.25$, $a_3 = 0.16$; $d_1 = 0.3$, and $d_5 = 0.072$, where all lengths are expressed in meters. The control gains were chosen as: $\Lambda_p = 7 \times 10^2\, I$, $\Lambda_o = 8 \times 10^4\, I$, $\Lambda_s = 3 \times 10^4\, I$. All other parameters are assumed to be zero.

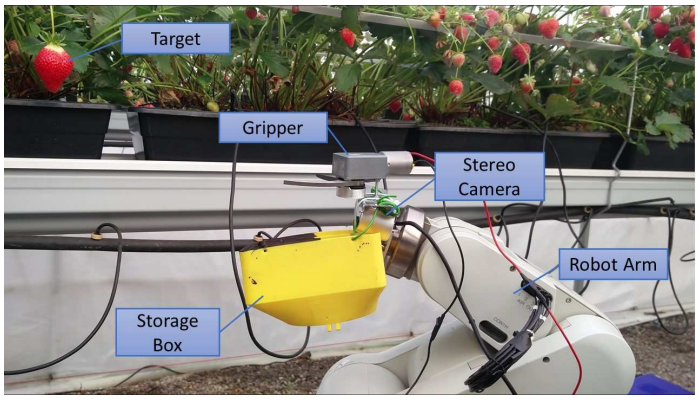

Figure 3.4: Experimental setup at the strawberry farm.

In this section, we describe some aspects of design and practical implementation of the proposed visual servoing approach. The flowchart of the fruit harvesting algorithm is shown in Fig. 3.5. From the calculation of the same image feature in both cameras of the stereo vision, we can compute the 3D point coordinates of the target in the operational space by using a triangulation technique [37]. In computer vision, a well-known problem consists on recognizing matching points that belong to the same image feature in different scenes, along with object extraction or image segmentation [44].

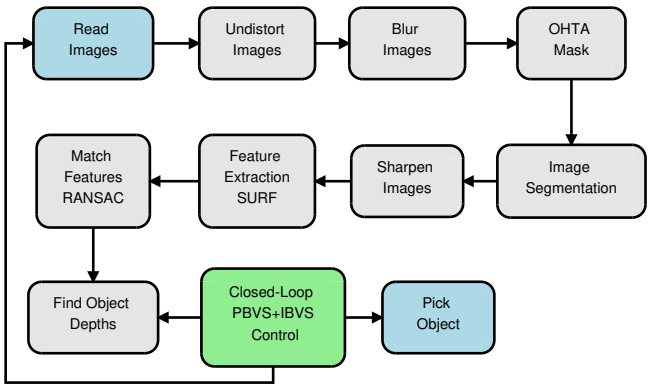

Figure 3.5: Fruit harvesting algorithm flowchart.

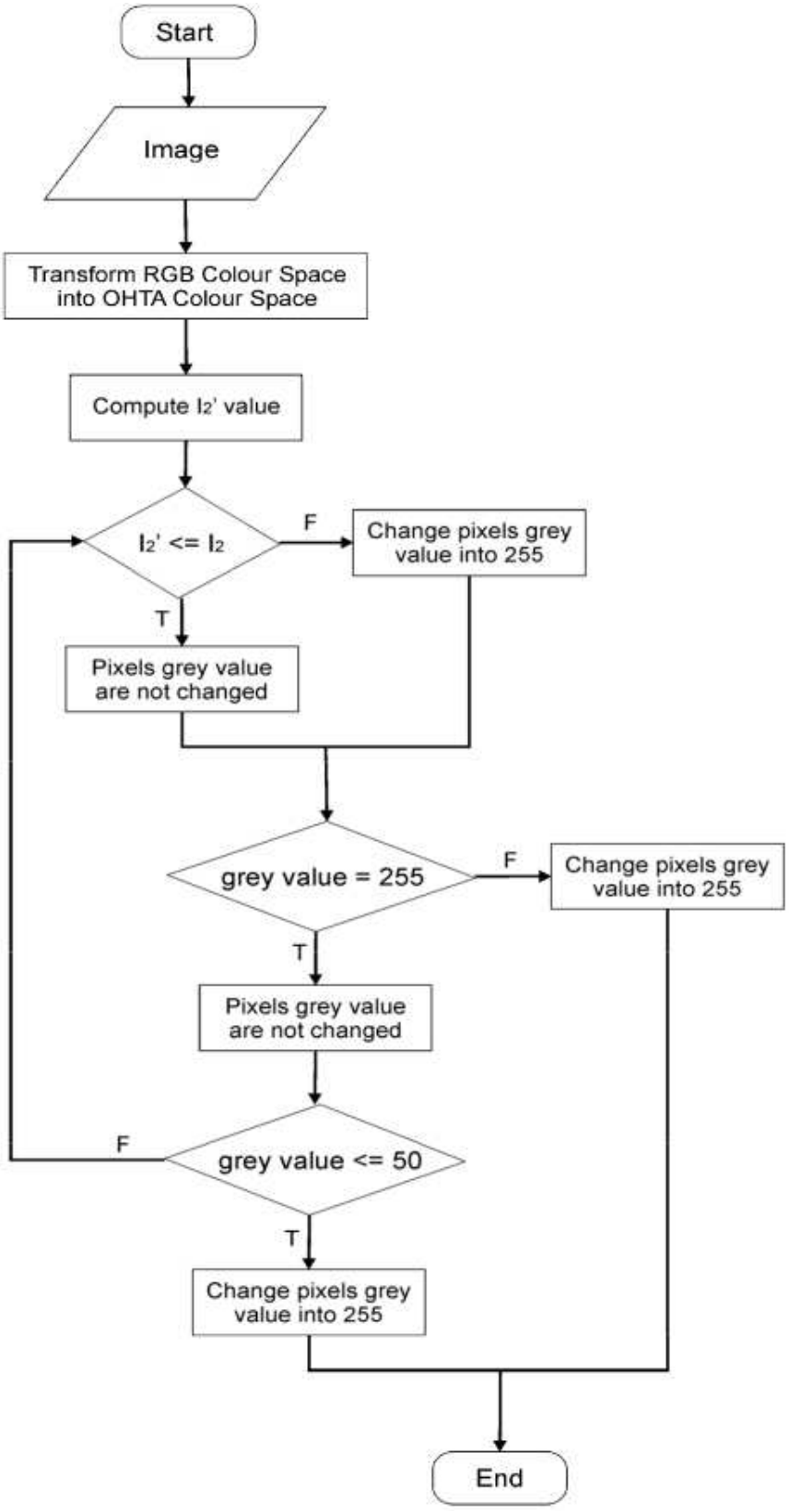

Figure 3.6: OTHA Segmentation Flowchart [64].

In this context, the OHTA color space has been chosen due to its simple implementation and accurate performance in comparison with HSV and RGB color spaces. An approach introduced by Vasthi *et al.* has demonstrated to be accurate and a computationally simple solution for fruit recognition by linearly transforming the image from an RGB space to an OHTA color space denoted by,

$$I_1 = \frac{R+G+B}{3}\,, \qquad I_2 = \frac{R-B}{2}\,, \qquad I_3 = \frac{2G-R-B}{4}\,, \tag{3-23}$$

where R, G and B are the red, green and blue channels from the RGB space, during the algorithm execution two specific OHTA features are chosen,

$$I_2 = \frac{R-B}{2}\,, \qquad I_2' = R-B\,, \tag{3-24}$$

for executing the OHTA Segmentation algorithm as presented in Fig. 3.6 [64].

Before running the segmentation algorithm, the two OHTA color features (I_2 and I_2') are preprocessed with a Median filter (available at the OpenCV library) to obtain a smoothed image and more consistent bounds at the mask obtained from the segmentation phase. Then a gray-scale image obtained from the original image is thresholded with the segmentation-mask obtained from the filtering phase to complete the fruit-image extraction.

After extracting the object of interest from the scene, a smoothed image (obtained from a Gaussian smoothing filter) is subtracted from the original image to obtain sharper edges facilitating the image-features extraction which is crucial for matching and triangulation phases. The SURF algorithm [45] is used to obtain image-features from both images and, then, the RANSAC algorithm [44] for identifying the corresponding or matching points. It is worth noting that when performing the extraction of image features after object segmentation, a lower computational cost and more simple calculations are required than if the process is performed in the reverse order. On the other hand, it is well-known that RANSAC algorithm is a stochastic algorithm which does not guarantee the total recognition of the inliers features. In addition, by performing a triangulation with outliers features from a considerable distance will result in a target position which is very different from expected. Therefore, it is not always possible to ensure the system stability by performing an end-point open-loop control based on the PBVS approach, using as a target object the corresponding 3D point coordinates obtained from the visual system. Under this constraint, a closed-loop control system seems to be the more appropriate to ensure the reliability and safety of tasks performed by vision-based controllers. The proposed visual servo system involves different control techniques: (i) PBVS scheme for approaching the target object (approaching

phase); (ii) IBVS scheme to maintain the object in the field of view of the stereo vision system (fine tuning phase); (iii) quaternion-based orientation control to ensure the proper pose of the robot end-effector for harvesting the selected fruit (picking phase). To deal with the minimum and maximum depth range presented in the stereo vision system as well as the depth needed for collecting the fruit properly, a depth estimation method based on the object projected area is used to calculate the object depth with respect to the camera. From a proper camera calibration, it is possible to compute the 3D point coordinates of the target using a single camera. After reaching the desired distance, a final image-based height adjustment is carried out for positioning the gripper close to the fruit stem, cut it and store the fruit in a storage box, completing the harvesting task successfully.

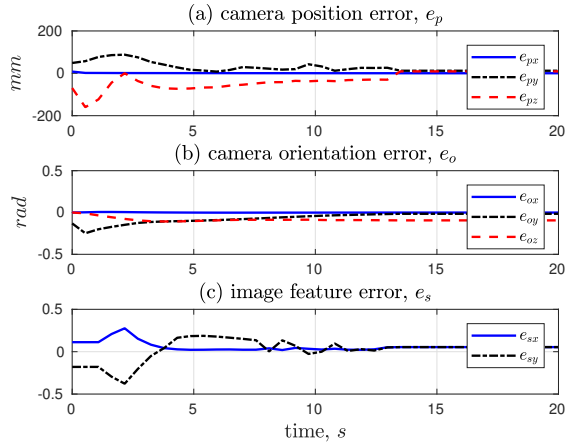

Figure 3.7: PBVS+IBVS: camera pose and image feature errors.

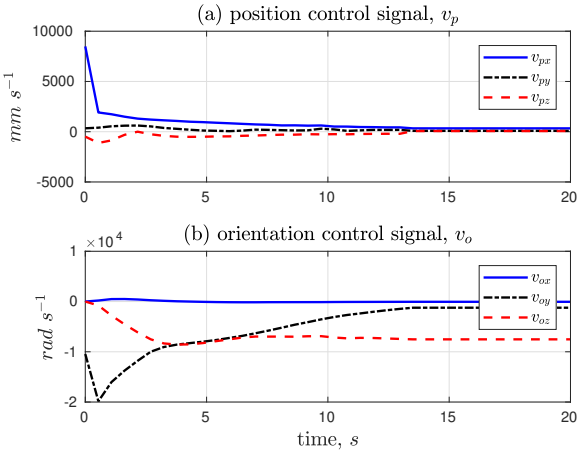

Figure 3.8: PBVS: pose control signals.

The *experimental results*[1] for a fruit picking task, performed with the Mitsubishi robot RV-2AJ at the strawberry farm in Sylling, Norway (Fig. 3.4)

are presented in Figures 3.7-3.10. The behavior in time for position, orientation and image regulation errors can be observed in Fig. 3.7, where it is possible to see the asymptotic convergence of the errors. From Fig. 3.8, we can verify the time history of the position and orientation control signals provided by the PBVS scheme during the approaching phase. Conversely, the behavior in time of the position and orientation control signals provided by the IBVS scheme during the picking phase is shown in Fig. 3.9. The time history of the velocity control signals applied to the joints is shown in Fig. 3.10(a). We can observe the smooth behavior of the joint velocities, and the satisfactory performance of the orientation control scheme. The angular position of joint 5, q_5, is critical for the successful execution of the fruit harvesting task and the orientation controller ensures that the values assumed by q_5 remain close to zero as seen Fig. 3.10(b).

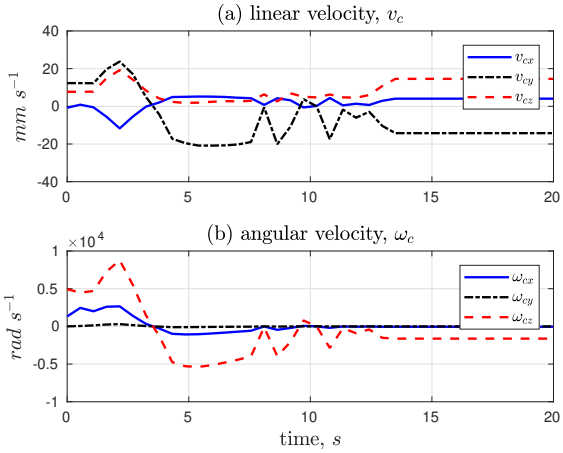

Figure 3.9: IBVS: linear and angular velocities.

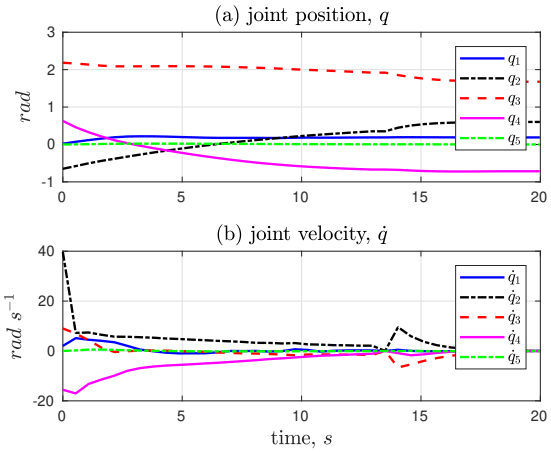

Figure 3.10: PBVS+IBVS: joint positions and velocities.

[1]The preliminary experiments can be viewed in the accompanying video clip in: `https://youtu.be/Hi10oiuG0_I`

3.4.4
Uncertain and Unconstrained robot-camera System IBVS

In the following section, it is assumed that property (3-21) cannot be satisfied into (3-18), thus the stability of the system cannot be guaranteed because $R_{ce}^{\top}(\varphi)\, R_{ce}(\hat{\varphi})$ becomes negative definite for $|\tilde{\varphi}| > \frac{\pi}{2}$ or indefinite for $|\tilde{\varphi}| = [\frac{\pi}{2}, \pi]$.

Here, the key idea is to employ a recently proposed robust visual servoing scheme based on a sliding mode control (SM) and a switching monitoring (SMC) function to cope with the performance degradation due to modeling uncertainty and camera calibration errors [46]. The proposed switching mechanism selects a suitable discrete pre-compensator out of a finite indexed set of matrices S_j , according to an appropriate *monitoring function* $\psi_m(t)$, to correct any mismatch from the nominal orientation $R_{ec}(\hat{\varphi})$ of the camera and the real one $R_{ec}(\varphi)$; with respect to the end-effector.

Here, we consider that the plant control direction is *unknown* (and constant) as Fig. 3.11 shows, in the sense that the uncertain parameter of the matrix

$$K_P = L_s(s)\, \bar{R}_{bc} \tag{3-25}$$

where $\bar{R}_{bc} = \text{diag}\{R_{bc}, R_{bc}\}$ belong to some compact set Ω_p where we assumed that: $\det(K_P K_P^{\top}) \neq 0$, and there exists a finite index set Ω of known matrices $S_j \in \mathbb{R}^{3\times 3}$ such that the matrix $-K_P\, S_j K_P^{\top}$ is Hurwitz stable for some $j \in \Omega$.

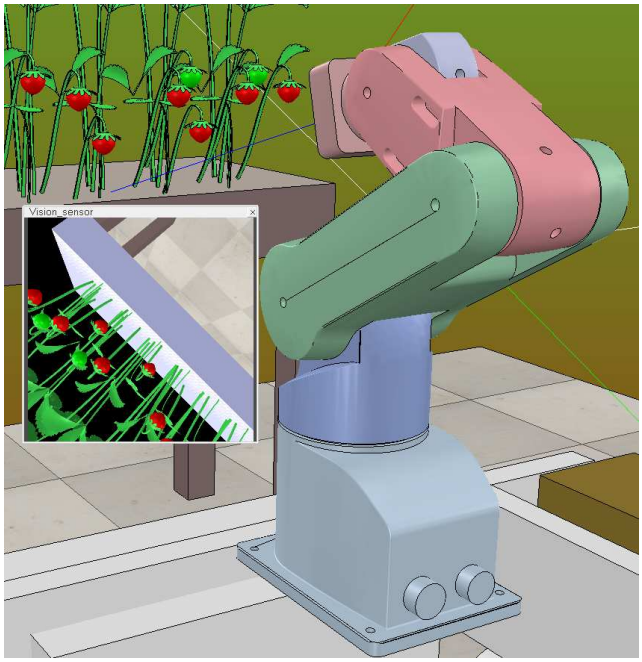

Figure 3.11: Robotic fruit picking tasks on V-REP robot simulator.

Hybrid Control Systems

Based on the SM and SMC controller introduced by [33], where a hybrid control scheme is used to deal with the calibration problem presented between the camera and end-effector frames alignment, the following SMC algorithm is proposed for an eye-to-hand visual servoing application.

Bringing the System to a Hybrid control system formalism and using a basic switching approach of four operating regions X_l, $l = \{0, 1, 2, 3\}$; we obtain:

$$f_l(\dot{e}_s) = -K_P\, S_l\, K_P^{\top}\, \Lambda\, e_s \qquad l \in L = \{0, 1, 2, 3\} \tag{3-26}$$

where the vector $f_l(\dot{e}_s)$ represents the dynamics of the $l - th$ mode into a continuous function, and L is a finite index set. For a state-dependent switching task, the switching times are denoted as k and it is assumed that for every discrete index N_L, exists an operating region X_l limited by a guard set $G(l, l')$. Note that, a discrete transition to l_{k+1} occurs when the continuous state e_s into operating region X_l meets its corresponding guard set $G(l, l')$ [47].

By defining the discrete state S_l as:

$$\mathcal{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \mathcal{S}_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},\ \mathcal{S}_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},\ \mathcal{S}_3 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},$$

and setting four discrete state values under the constraint (3-21), the following operating regions into the radiant space are obtained $X_0 = [\frac{-\pi}{2}, ..., \frac{\pi}{2}]$, $X_1 = [0, ..., \pi]$, $X_2 = [\pi, ..., 2\,\pi]$ and $X_3 = [\frac{\pi}{2}, ..., \frac{-\pi}{2}]$ as shown in Fig.3.12.

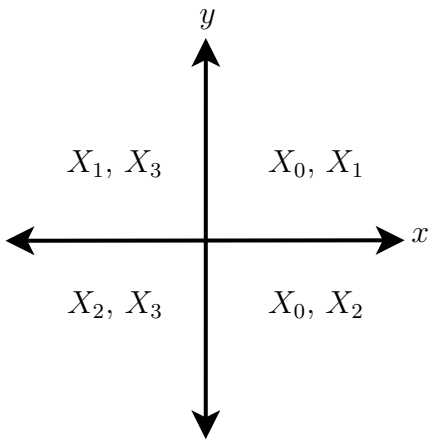

Figure 3.12: Operating Regions.

Analyzing the continuous system state variable we have that for any $t_i \in [0, t_M]$ and, by using the comparison lemma [41], we obtain:

$$||e_s(t)|| \leq \xi(t)\,, \qquad \forall t \in [t_i, t_M]\,, \tag{3-27}$$

where

$$\xi(t) := ||e_s(t_i)||\, e^{-\lambda_m(t-t_i)}\,, \tag{3-28}$$

where λ_m is a positive constant.

Constructing the guard set $G(l, l')$ as a monitoring function $\psi_m(\cdot)$ based on the norm bound for the feature error e_s given in (3-27) following the ideas introduced in [48, 46]. Notice that, (3-27) holds when the matrix S_l is correct ($S_l = S$), it seems natural to use the term $\xi(t)$ as a benchmark to decide whether a switching of S_l is needed, that is, the switching occurs only when the condition (3-27) is violated. On the other hand, since S_l is *unknown*, the following function ψ_k is defined in the interval $[t_k, t_{k+1}]$, to replace the term $\xi(t)$ as:

$$\psi_k(t) = ||e_s(t_k)||e^{\lambda(t-t_k)} + \gamma\, a(t_k)\, e^{\frac{-t}{a(t_k)}}\,, \tag{3-29}$$

where the switching time t_k sets the change of index $l \in L$, cycling through the S_l matrices for $l = 1, 2$, and $a(t_k)$ is any positive monotonically increasing unbounded sequence. The monitoring function ψ_m can be defined as:

$$\psi_m(t) := \psi_k(t)\,, \quad \forall t \in [t_k, t_{k+1}] \subset [t_0, t_M]\,. \tag{3-30}$$

Note that, from (3-29) and (3-30), we have $||e_s(t_k)|| < \psi_k(t_k)$ at $t = t_k$. Hence, the switching time t_k is defined as the time instant when the state $||e_s(t)||$ meets the guard set $G(l, l')$ (monitoring function $\psi_m(t)$) , that is,

$$t_{k+1} := \begin{cases} t_M, & \text{if} \quad ||e_s(t)|| \geq \psi_m(t), \\ t_k, & \text{otherwise}, \end{cases} \tag{3-31}$$

where $k = 0, 1, \cdots$ and $t_0 := 0$. From (4-30), the following inequality can be obtained:

$$||e_s(t)|| \leq \psi_m(t)\,, \quad \forall t \in [0, t_M]\,. \tag{3-32}$$

3.4.4.1 Validation Uncertain and Unconstrained IBVS System

In this section, we demonstrate the introduced Hybrid Controller(Switching Monitoring Controller, SMC), configured as

- $\lambda = 1$,
- $\gamma = 0.05$,

to perform a faster respond of the SMC, two independent monitoring function ψ_m have been employed. The regulation task was performed with:

- Misalignment angle $\tilde{\varphi}$ between the camera and the end-effector frames around the z-axis of $2.056\,\pi$ as commented in (3-21).
- Uncertainty of ten percent into the intrinsic camera parameters.

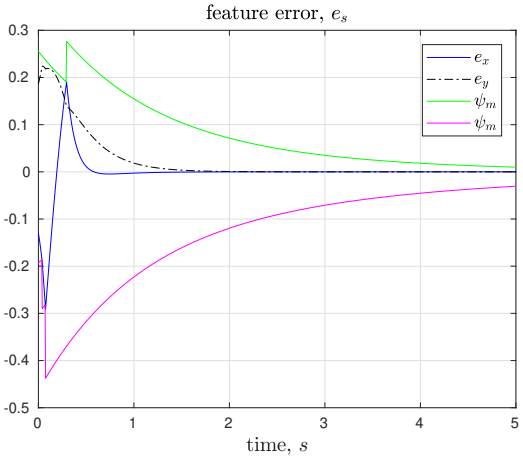

Figure 3.13: IBVS/SMC Position Error Behavior.

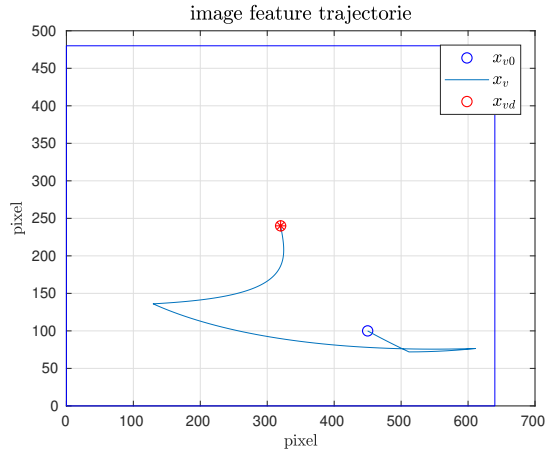

Figure 3.14: IBVS/SMC Image Trajectory: initial position: blue "∘", desired position: red "∗" and final position: red "∘".

– Uncertainty of ten percent at last link-length of the robot arm.

Simulations were carried out with a Denavit-Hartenberg representation of the Mitsubishi RV-2AJ robot arm [37]. Figs. 3.13-3.15 demonstrate the SMC controller performance, where the feature error e_s converge to zero over the time, Fig. 3.13 demonstrates the switching times (3-26) of the SMC controller, the algorithm starts in the region X_0 then it switches three times to reach X_3. Moreover, Fig. 3.14 illustrates the feature trajectory into the image plane and the successful performance of the algorithm for maintaining the image feature s into the screen. Finally, Fig.3.15 shows the velocity control signal applied to the robot manipulator.

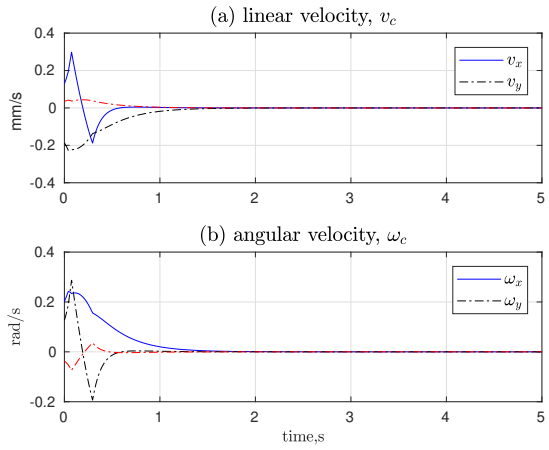

Figure 3.15: IBVS/SMC Manipulator Velocity.

3.4.5 Nonvanishing Perturbations IBVS

The following section analyses an IBVS task exposed to external perturbations $d(t)$, a robust approach is presented to guarantee the system stability during the control execution. Computing the closed-loop differential equation,

$$\dot{\zeta} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ I \end{bmatrix} \left(d(t) - L_s(s)\, R_{bc}^{\top}\, J(q)\, u(t) + \eta_v(s, q, \dot{q}) \right), \qquad (3\text{-}33)$$

where $\zeta = [x_1, x_2]^T$, $x_1 = \int_0^t e_s(\tau) d\tau$ and $x_2 = e_s = s_d - s$; $d(t)$ is a non-state dependent bounded function, thus it is not possible to analyze stability at the equilibrium point because the perturbed system may have not an equilibrium point at the origin [41]. Taking the control law

$$u = \hat{J}^{\dagger}\, \hat{R}_{bc} \hat{L}_s^{\dagger} \left(\begin{bmatrix} \Lambda_1 & \Lambda_2 \end{bmatrix} \zeta + \Psi \right), \qquad (3\text{-}34)$$

into (3-33), we obtain

$$\dot{\zeta} = \begin{bmatrix} 0 & I \\ -\Lambda_1 & -\Lambda_2 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ I \end{bmatrix} (d(t) + \eta_v(s, q, \dot{q}) - \Psi). \qquad (3\text{-}35)$$

By recalling that $\eta_v(s, q, \dot{q})$ in (3-12) is a function of ζ, the following assumptions are made:

$$\sup_{t \geq 0} ||d(t)|| < Q_M < \infty \qquad \forall d(t) \qquad (3\text{-}36)$$

$$\eta_v(s, q, \dot{q}) < \Phi < \infty \qquad \forall x_1, x_2 \qquad (3\text{-}37)$$

Assumption (3-36) is practically satisfied since any s_d trajectory cannot require an infinite velocity. Regarding assumption (3-37), $\eta_v(s, q, \dot{q})$ is a function of q and $\dot{q}$, where q and $\dot{q}$ are bounded due to the existence of joint ranges and saturation at the joints velocity. In other words infinity positions and velocities

are not considered in practice.

To determine Ψ, consider the Lyapunov candidate function

$$V = \zeta^\top Q\zeta > 0 \qquad \forall\zeta \neq 0, \tag{3-38}$$

where Q is a positive definite matrix. Then, obtaining the derivative of (3-38)

$$\dot{V} = \zeta^\top(A^\top Q + QA)\zeta + 2\zeta^\top QB(d(t) + \eta_v(s, q, \dot{q}) - \Psi), \tag{3-39}$$

Since A has eigenvalues with negative real part, it is necessary to compute a positive definite matrix P as:

$$A^\top Q + QA = -P, \tag{3-40}$$

then (3-39) becomes in

$$\dot{V} = -\zeta^\top P\zeta + 2\zeta^\top QB(d(t) + \eta_v(s, q, \dot{q}) - \Psi). \tag{3-41}$$

Regarding the left term of (3-41), it is possible to see that it is negative definite, now the Ψ term needs to be chosen in order to vanish the uncertain term $\eta_v(s, q, \dot{q})$ and $d(t)$ when they are different to zero. this term can be modeled based on a Variable Structure Control (VSC) our Unit Vector Control (UVC) approaches.

By setting a sliding region $Z \in \mathbb{R}^{3\times1} = 2\,B^T\,Q\,\zeta$, the *perturbed term* can be rewritten as $Z^T(d(t) + \eta_v(s, q, \dot{q}) + \Psi)$.

3.4.5.1 Unit Vector Control (UVC)

The nonlinear robust control design that relies on *unit vector control* (UVC) based on the *norm* function instead of the usual *sign* switching control of *variable structure controller* (VSC)[49]. In this context, the Ψ vector can be computed as,

$$\Psi = \frac{\rho}{||Z||}\,Z + \varrho Z \qquad \rho, \varrho > 0, \tag{3-42}$$

gives

$$Z^T((d(t) + \eta_v(s, q, \dot{q})) - \Psi) = Z^T(d(t) + \eta_v(s, q, \dot{q})) - \frac{\rho}{||Z||}\,Z^T\,Z - \varrho\,Z^T\,||Z||, \tag{3-43}$$

By choosing ρ and ϱ, as

$$||(d(t) + \eta_v(s, q, \dot{q}))|| < Z\left(\frac{\rho}{||Z||} + \varrho\right), \tag{3-44}$$

it is possible to vanish the *perturbed term* $(d(t) + \eta_v(s, q, \dot{q}))$, guaranteeing the asymptotic error convergence over the time t.

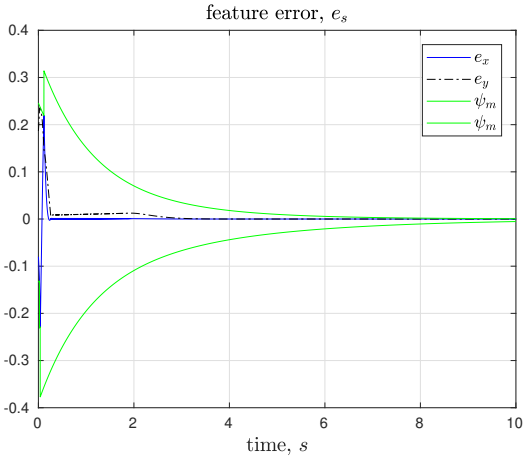

Figure 3.16: UVC Position Error Behavior.

3.4.5.2 Validation UVC

In this section we demonstrate the introduced UVC controller to perform a tracking task of an unknown trajectory with the following control law,

$$u = \hat{J}^{\top} \hat{R}_{bc} \hat{L}_s^{\dagger} \left(\begin{bmatrix} \Lambda_1 & \Lambda_2 \end{bmatrix} \zeta + \Psi \right), \tag{3-45}$$

whit the following assumptions,

- A ten percent uncertainty at the length of the last link.
- A ten percent uncertainty at the intrinsic camera parameters.
- Misalignment angle $\tilde{\varphi}$ between the camera and the end-effector frames around the z-axis of $2.056\,\pi$ as commented in (3-21).
- An unknown circular trajectory of 2π rad/s.

Figs. 3.16-3.19 demonstrate the UVC controller performance, where the Cartesian error e remains close to zero along time, even when the desired trajectory $d(t)$ is unknown. Moreover, Fig. 3.17 demonstrates the unit vector control Ψ and the sliding surface Z performance of the proposed algorithm in (3-42), it is possible to see how the unit vector Ψ presents a very discontinues performance. Finally, Fig.3.19 demonstrates how the inconsistency of the unit vector Ψ control is propagated to the robot joints velocities. In fact, due to the irregularity of the control signal, a practical application could end in damaging the robot motors.

In this sense, a second-order sliding mode controller may be able to minimize common robust-control problems like chattering very common in first-order sliding mode controllers [63].

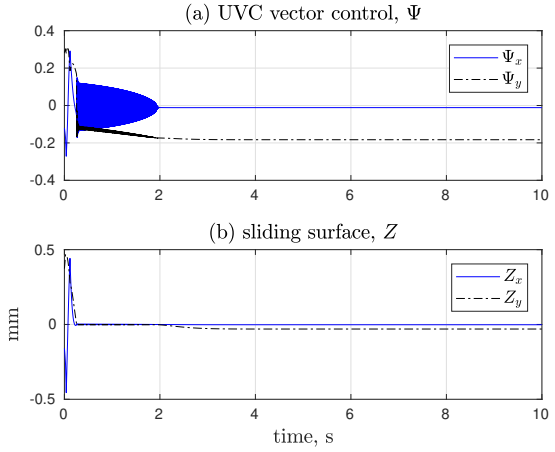

Figure 3.17: UVC Sliding Surface and vector control signal.

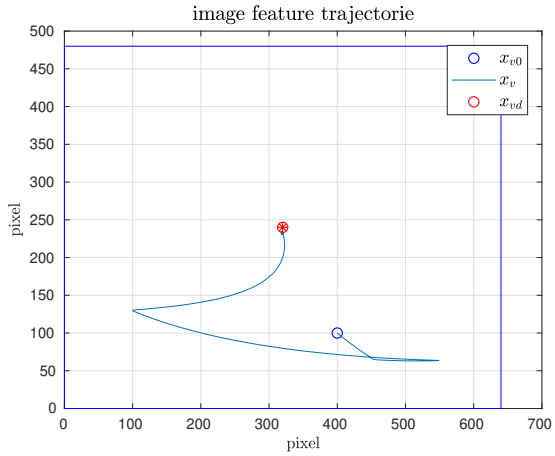

Figure 3.18: UVC Cartesian Trajectory.

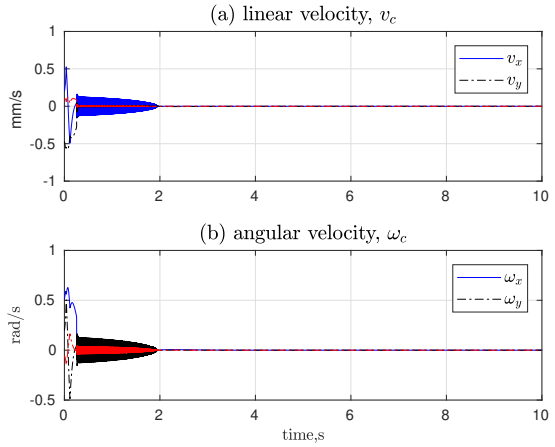

Figure 3.19: UVC Joint Velocity.

3.4.5.3 Super Twisting Sliding Mode (ST-SM)

Chattering are very dangerous effects presented into conventional sliding mode controllers due to the infinite discontinuous responses (switching) because of the *sign* function. Modified first-order sliding mode controllers substitute the conventional *sign* function by *tanh* or *sat* functions to minimize chattering effects but compromising performance and robustness of the algorithm [49].

Second order sliding mode controllers have demonstrated a more robust alternative for solving the chattering problem without compromising robustness properties [63]. Following this trend, the vector Ψ is computed as,

$$\Psi = \rho_1 \sqrt{|Z|}\, sign(Z) + \upsilon + \varrho Z \qquad \varrho > 0, \tag{3-46}$$

$$\dot{\upsilon} = \rho_2\, sign(Z). \tag{3-47}$$

where,

$$Z^\top((d(t) + \eta_v(s,q,\dot{q})) - \Psi) = Z^\top(d(t) + \eta_v(s,q,\dot{q})) - \rho_1 \sqrt{|Z|}\, sign(Z) - \rho_2 \int_0^t sign(Z)dt - \varrho Z\,. \tag{3-48}$$

By choosing ρ and ϱ, as

$$||(d(t) + \eta_v(s,q,\dot{q}))|| < \rho_1\, ||\sqrt{|Z|}\, sign(Z)|| + \rho_2 \left|\left|\int_0^t sign(Z)dt\right|\right| + \varrho\, ||Z||\,, \tag{3-49}$$

and for simplicity, the parameters $\rho 1$ and $\rho 2$ can be set as

$$\rho_1 = \sqrt{\rho} \qquad \rho_2 = 1.1\,\rho \qquad \rho > 0, \tag{3-50}$$

where ρ and ϱ are positive and sufficient large constants for vanishing any perturbation added to the nominal system [49].

3.4.5.4 Validation ST-SM

In this section we demonstrate the introduced ST-SM controller to perform a tracking task of an unknown Cartesian trajectory with the same control law and assumptions commented in the last subsection.

Figs. 3.20-3.23 demonstrate the ST-SM controller performance, where the projection error e_s remains close to zero along time, even when the desired trajectory $d(t)$ is unknown. Moreover, Fig. 3.21 demonstrates the unit vector control Ψ and the sliding surface Z performance of the proposed algorithm in (3-42), it is possible to see how the unit vector Ψ presents a

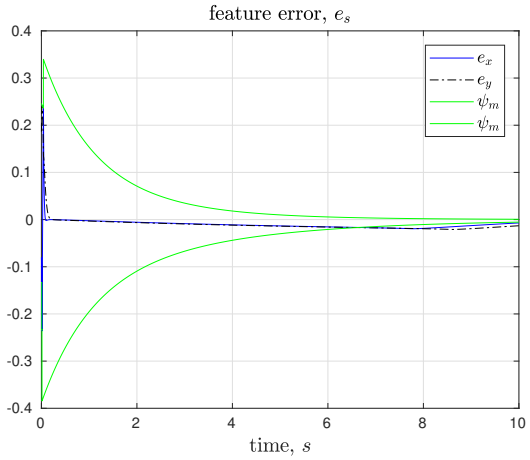

Figure 3.20: ST-SM Feature Error.

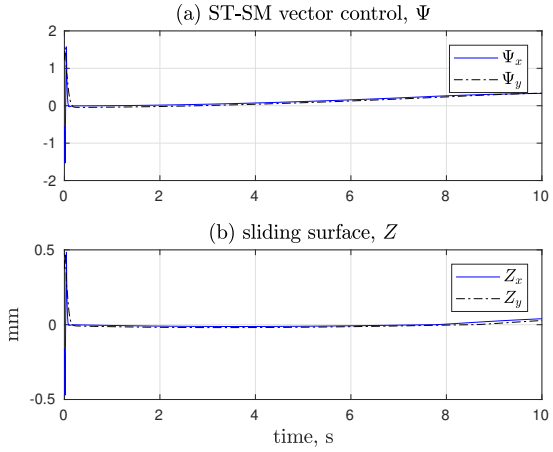

Figure 3.21: ST-SM Sliding Surface and vector control signal.

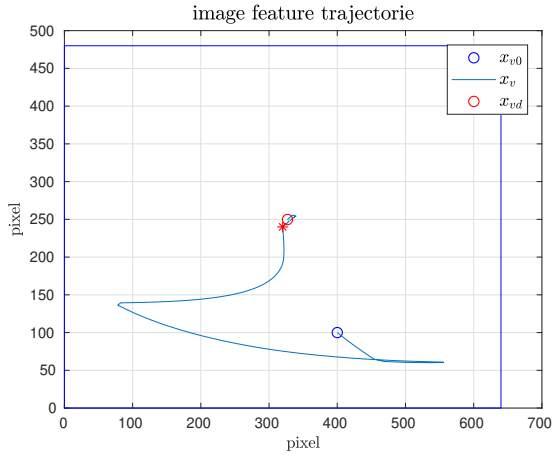

Figure 3.22: ST-SM Image Trajectory.

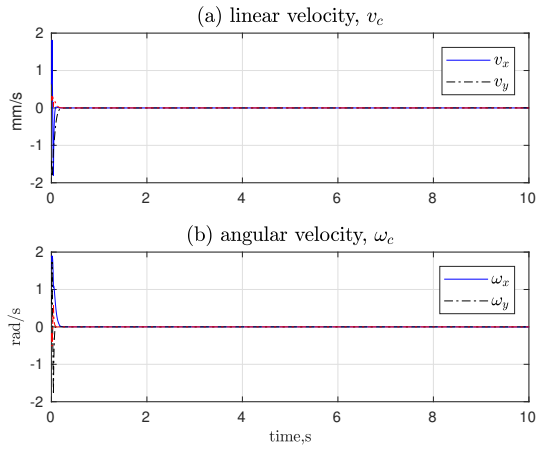

Figure 3.23: ST-SM Joint Velocity.

more continues performance in comparison with the UVC algorithm. Finally, Fig.3.23 demonstrates the smooth performance the robot joints velocities.

3.5 Concluding Remarks

In this chapter, we have demonstrated how PBVS and IBVS control schemes can be used to approach or reach a desired 3D position with respect to an object of interest and how the IBVS approach demonstrates more robustness to uncertainties than the PBVS approach.

In this sense, a critical uncertainty to the IBVS control scheme belongs to the rotation between the camera an end-effector frames, a hybrid control scheme inspired in the SMC and SM controllers presented by Roux *et al.* [33] is used to deal with an uncertain planar rotation along the z-axes of the camera and end-effector frames. However, being that the switching order is fixed it is not possible to guarantee $V_{l_{(k+1)}}(e_s(k+1)) \leq V_{l_{(k)}}(e_s(k))$ in certain switching times k.

Another strategy presented is an IBVS sliding mode design for tracking a desired feature s_d where $d(t)$ is unavailable or very noisy. In cases where $||d(t)||$ is large enough to cause instability into the system (3-39), the UVC scheme demonstrates a very chattered response for preserving the system stability, this chattering return can be very problematic at real applications damaging the robot motors. In this context, the chattering problem is treated with an ST-SM scheme, which by combining continuous and discontinuous and continuous control laws and an appropriate gains configuration, it is possible to vanish the chattering problem without compromising robustness properties.

4
Robust Hybrid Visual Servoing (HVS)

The following section describes our proposal in terms of detection, recognition, and collection of many strawberries presented into a scene (crop). Eventually, the task starts by gathering information from the scene with a vision-system (stereo camera). Then, the obtained images are processed concerning object identification and correspondence points matching to construct an accurate 3D position. Finally, this 3D position and the data obtained from the camera in real time is used to run a Hybrid Visual Servoing algorithm for reaching the fruit position completing the harvesting task (Fig. 4.1).

Wei *et al.* have presented a fruit segmentation scheme which transforms the fruit image from RGB space to OHTA color space and adds an adaptive thresholding based on the Otsu proposal [67], obtaining as result an image segmentation process more robust to light changes and able to extract strawberries from complex backgrounds [18].On the other hand, color segmentation algorithms are not able to extract specific objects in clusters configurations, with occlusion or different maturity levels. To solve the different maturity levels presented into a single fruit, a pre-trained deep encoder-decoder algorithm based on the "SegNet" architecture proposed by Badrinarayanan *et al.* [61] is used due to its capability to learn image features from the object of interested. Nowadays, learning multiple image features demands very complex algorithms, turning the deep network into a computationally expensive solution at training and inference phases. In this way, Deep encoder-decoder algorithms are a non-viable option where the required computational resources are not available or inexistent. The experiment presented in this chapter employs a modified

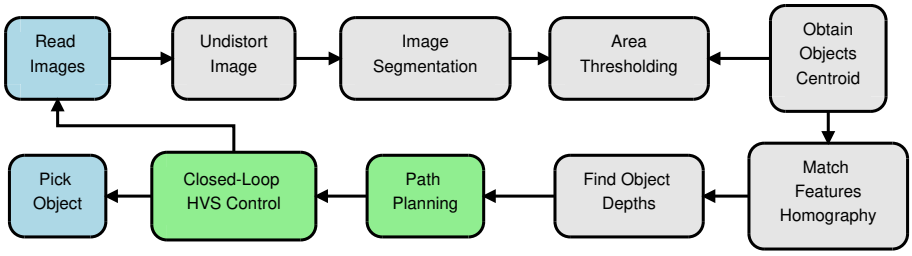

Figure 4.1: Fruit picking algorithm flowchart.

version of the Wei approach [18] to segment the fruit from the background.

Correspondence points matching has been considered as the base for 3D position reconstruction from two or more views of the same scene, where the data from views is usually gathered by a single camera or a vision-system (stereo camera). In this sense, recent researches in feature detection from images come up with robust, complex and/or computationally expensive solutions such as Harris Detector, Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Oriented FAST and Rotated BRIEF (ORB) and others [65], [66], [45], [54].

SURF and ORB algorithms have demonstrated to be the more suitable tools for discovering image features at soft-fruit images over SIFT and Harris detector, In view of fast image processing task requirements (features detection algorithm is intended to run into a closed-loop robot controller) and the lack of sharp edges and corners (rounded shape). Image features obtained by using SURF and ORB algorithms come with an image descriptor wichs adds histograms information that surrounds the image feature ensuring invariance at the location, scale, orientation, and illumination. In this way, this feature-description information is used by mathing algorithms like Random Sample Consensus (RANSAC), Brute-Force Matcher (BF Matcher) and others to discover corresponded points.

At the experiment of chapter three where a harvesting task involving a single strawberry into the scene is considered, it is possible to demonstrate the feasibility of detecting and matching image features thru the use of SURF and RANSAC algorithms. In applications where harvesting more than one strawberry into a scene, the feature descriptor information tends to be very similar to other features (various similar fruits presented into the scene) which means that is not possible to ensure a proper features correspondence nor an accurate 3D reconstruction. Alternatively, a simpler feature detection and matching solutions are proposed, assuming that:

- fruits presented into the scene are widely separated (non-cluster configurations);
- images obtained from the vision-system do not present occlusion configurations.

In this sense, the fruit centroid is used as an image feature and the correspondence problem is carried out by finding the closest point to an estimated pixel position, which is calculated with the scene homogeneous transformation (Homography). By this way, the correspondence points matching problem is achieved by simply discovering fruit centroids and estimating a pixel position

into the other view,

$$\hat{p}_r = \hat{H}\, p_l. \tag{4-1}$$

After segmenting and finding matching points, fruits 3D positions are obtained by employing a triangulation technique.

To achieve the a desired cartesian position into the task operating space for harvesting a fruit of interest, a visual-robot-controller based on the well-known Visual Servoing methodology is proposed, which combines the strengths from classical approaches as PBVS and IBVS. In this sense, the most simple technique to reach the fruit position may be use an open-loop PBVS control scheme under the strict assumption that there are not any model uncertainties into the robot and camera models (robot and camera well calibrated) nor external perturbations. Nevertheless, many control problems involve uncertain parameters due to slow time variation of the parameters (e.g., ambient air pressure during an aircraft flight), an abrupt change in parameters (e.g., internal parameters at robot grasping application) and/or hard-modeling nonlinearities at the control system [60]. On the other hand, by employing a closed-loop PBVS control scheme to minimize a given Cartesian error vector e_p. Since this error vector only belongs to the Cartesian space, it is not possible to guarantee the presence of the image feature into the camera Field Of View (FOV) during task execution. Then, it may not be possible to feed the closed-loop controller by estimating the object 3D position (feedback from the environment) breaking the control loop [13].

For ensuring a successful task execution in terms of VS controller, the proposed algorithm needs to contemplate a closed-loop strategy for adding robustness to model uncertainties and controlled external perturbations. Moreover, the algorithm requires to be able to perform 3D tasks and to ensure at the same time that the object image feature always remains into the FOV along the task execution. In this context, the proposed Hybrid Visual Servoing scheme (HVS) combines the planar image vector e_s employed into the classic IBVS approach and the depth parameter from the PBVS strategy to obtain a Hybrid controller that meets the discussed requirements [37].

Given the task of collecting more than one object, a memetic random restart strategy is employed to minimize the collecting time, also a sliding window is used to satisfactory reach the desired fruit. Finally, the robust algorithm combines Sliding Mode and Switching Monitoring control methods to execute a particular picking-task successfully into the presence of non-parametric uncertainties.

4.1 Problem Formulation

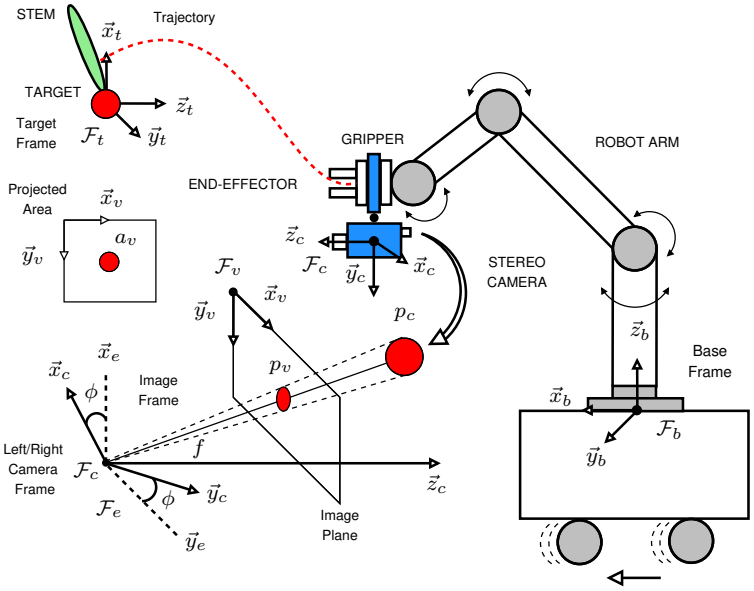

Figure 4.2: Visual servoing system for robotic harvesting tasks.

In this work, we address the robotic fruit harvesting problem using a visual servoing scheme with an RGB-D stereo camera mounted on the robot end-effector (Fig. 4.2). Here, the following notation is considered: $p_{ij} \in \mathbb{R}^3$ and $R_{ij} \in \mathbb{SO}(3)$ denote respectively the position vector and orientation matrix of the frame $\mathcal{F}_j$ with respect to frame $\mathcal{F}_i$; $T_{ij} \in \mathbb{R}^{4\times4}$ is the homogeneous transformation matrix, which denotes the pose of the frame $\mathcal{F}_j$ with respect to frame $\mathcal{F}_i$. In this context, the pose of the camera frame $\mathcal{F}_c$ with respect to the base frame $\mathcal{F}_b$ is given by $T_{bc} = T_{be}\, T_{ec}$, say:

$$T_{bc} = \begin{bmatrix} R_{bc} & p_{bc} \\ 0^\intercal & 1 \end{bmatrix} = \begin{bmatrix} R_{be}\, R_{ec} & R_{be}\, p_{ec} + p_{be} \\ 0^\intercal & 1 \end{bmatrix}. \tag{4-2}$$

Here, we assume that (A1) the homogeneous transformation matrix T_{be} can be obtained from the *forward kinematics* map by using, for example, the Denavit-Hartenberg convention. In this case, implies that $p_{be} = p_{be}(q)$ and $R_{be} = R_{be}(q)$. For simplicity, we also assume that (A2) the camera frame $\mathcal{F}_c$ and the end-effector frame $\mathcal{F}_e$ are aligned only with respect to z-axis, but the relative translation between their origins and the relative orientation of their z-axes, denoted by ϕ, may be uncertain. In this context, implies that $R_{ec} = R_{ec}(\phi)$.

The fruit harvesting task we consider consists of moving the robot arm to the vicinity of the fruit, cut the stem using a suitable device attached to the robot end-effector and store the fruit in a storage device.

In this context, the harvesting execution-loop is presented in Fig. 4.1, where the first goal is to solve the image segmentation and interpretation problem, that is, to detect and recognize the target object located in the robot workspace by using a deep encoder-decoder algorithm. Once the target object is tracked by using a stereo vision system, the next step is to solve the correspondence and 3D reconstruction problem, that is, to compute the 3D coordinates of the fruit with respect to the camera by using, for example, a simple triangulation technique [37]. Thus, the pose of the target frame $\mathcal{F}_t$ with respect to the base frame $\mathcal{F}_b$ is given by $T_{bt} = T_{bc}\, T_{ct}$, say:

$$T_{bt} = \begin{bmatrix} R_{bt} & p_{bt} \\ 0^{\intercal} & 1 \end{bmatrix} = \begin{bmatrix} R_{bc}\, R_{ct} & R_{bc}\, p_{ct} + p_{bc} \\ 0^{\intercal} & 1 \end{bmatrix}. \tag{4-3}$$

where T_{ct} is the homogeneous transformation matrix whose entries can be computed from the application of the segmentation algorithm and the triangulation technique. Finally, since the homogeneous transformation matrix T_{bt} is computed, we can employ an *inverse kinematics-based algorithm* to transform the motion specifications, assigned to the robot end-effector in the task space, into the corresponding joint space motions, allowing for the successful execution of the desired motion.

4.2 Detection and Recognition

It is well-known that by detecting and recognizing matching points of two or more images taken from different perspectives of the same scene, we can compute a 3D point coordinates of a given object of interest into the operational space by using a triangulation technique [37]. Moreover, recognizing and matching points that belong to the same image feature in different views is difficult along with, object extraction or image segmentation in complex backgrounds [44].

Here, a color based segmentation algorithm combined with an adaptive threshold obtaining segmentation-robustness to different light conditions [18]. A machine learning solution is also proposed based on a deep encoder-decoder network structure, where the main idea is to perform a wise-segmentation based on shape, image-texture, and color. Finally, due to the high computational cost demanded by the deep encoder-decoder algorithm during execution, a modified version of the Wei *et al.* proposal is presented by performing an image filtering with a Gaussian smoothing filter before running the adaptive thresholding step, this to obtain more uniform segmentation.

4.2.1 OHTA-Otsu Image Segmentation

In this context, Wei *et al.* have presented a more robust image segmentation scheme based on the OHTA color space and Otsu adaptive thresholding [18]. As a result the algorithm is able to deal with light variations at crops,

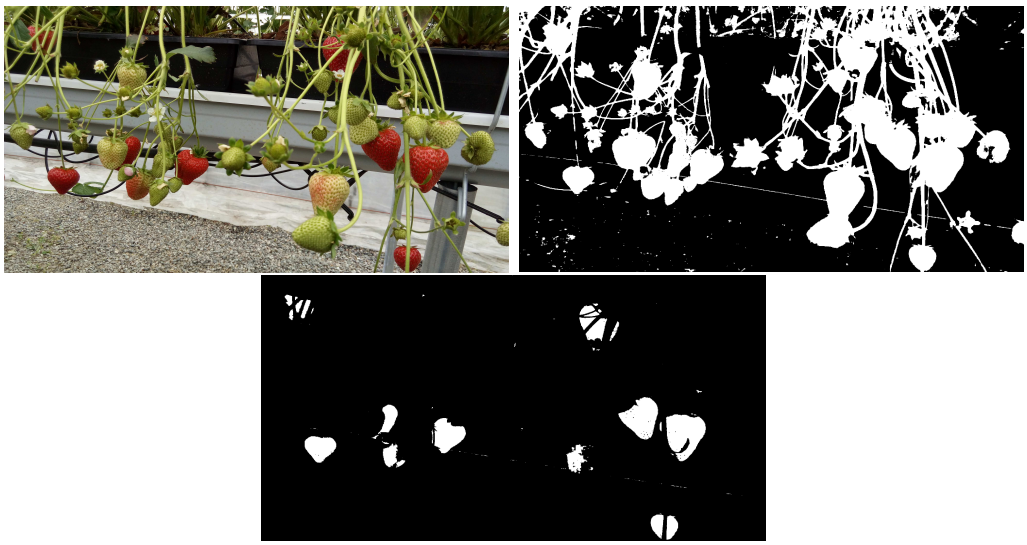

Figure 4.3: from left-to-right and top-to-bottom: Original Image, Result processed by second normalized OHTA channel, Result processed by first OHTA normalized OHTA channel.

detect and segment fruits and vegetation in complex agriculture situations (Fig. 4.3) by a linear transformation of a given image from an RGB space to an OHTA color space, there are two kinds of expressions for converting this image to the OHTA color space denoted by,

$$I_1 = \frac{R+G+B}{3}, \qquad I_2 = \frac{R-B}{2}, \qquad I_3 = \frac{2G-R-B}{4}, \tag{4-4}$$

and,

$$I'_1 = R-G, \qquad I'_2 = R-B, \qquad I'_3 = \frac{2G-R-B}{2}, \tag{4-5}$$

where R, G and B are the red, green and blue channels from the RGB space [18]. As presented in Fig 4.4, the main advantages of this algorithm are its simplicity, low computational and fast execution. An area-thresholding step has been added at the end of the segmentation phase for removing objects with an area less than a certain threshold (ten pixels) which are considered as noise.

4.2.2 Deep Encoder-Decoder Image Segmentation

Traditional encoder-decoders minimize a given loss function

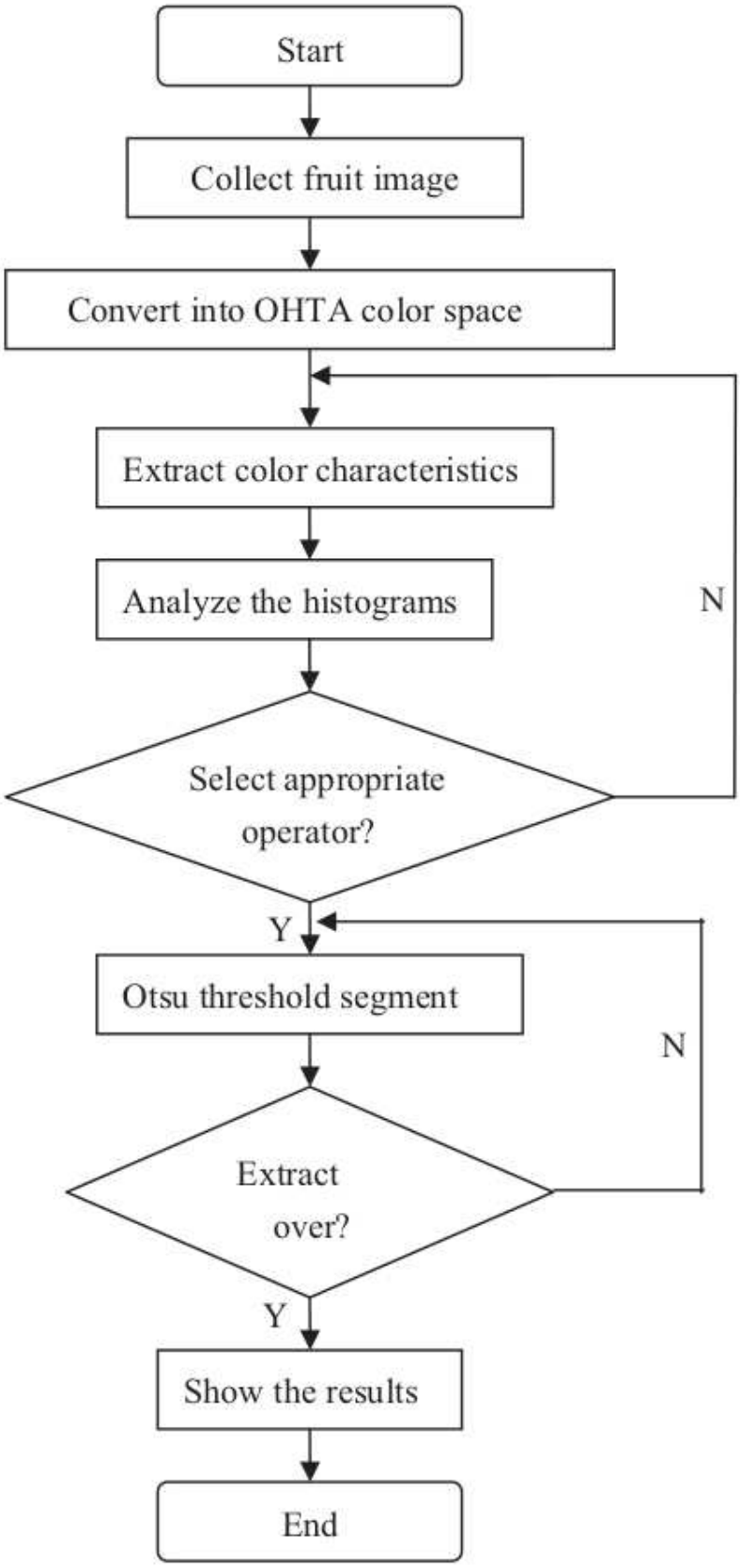

Figure 4.4: Flow chart of the automatic method of fruit object extraction for vision system of fruit picking robot [18].

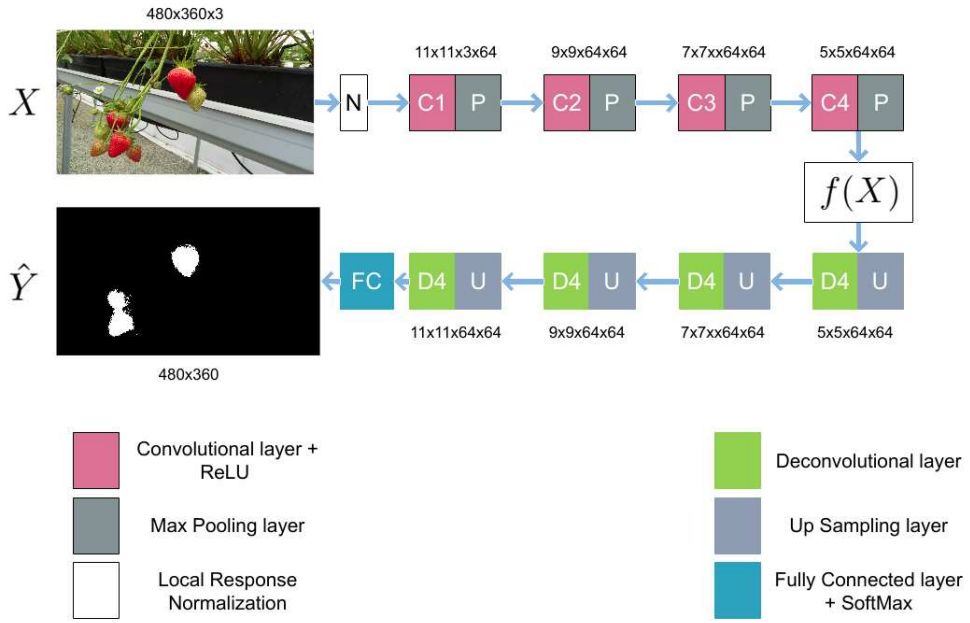

Figure 4.5: Deep Encoder-Decoder algorithm.

$$L(Y, \hat{Y}), \qquad \hat{Y} = g(f(X))), \tag{4-6}$$

where $L(\cdot)$ is a loss function regarding the dissimilarity from a desired output (ground of truth) Y and the output $g(f(X))$ where $f(\cdot)$ and $g(\cdot)$ are the encoder and decoder stages respectively.

In this context, deep convolutional encoder-decoders algorithms may become a very useful tool for semantic segmentation by learning a certain set of features of a given object (pixel-wise classification task). Being that the semantic segmentation is intended to be performed at every frame during the task execution, a simplified encoder-decoder version based on SegNet architecture [61] (Fig. 4.5) (four convolutional and deconvolutional layers) is introduced to reach a faster inference than the original network. Subsequently, dropout layers were added to the simplified model for improving training results in very small datasets [52]. At the encoder side, there are four convolutional layers with different kernel dimension (Fig. 4.5) and an activation function *ReLU* [50], there are also a max-pooling layer after every convolutional layer to add small translation invariance to the model. Decoder side has four deconvolutional layers with same kernel dimensions to preserve the model-symmetry into the model, then an upsampling layer is added before every deconvolutional layer. Finally, in aims of improving algorithm performance but also maintaining a computationally cheap solution, the convolutional kernels dimension of the encoder-decoder layers were modified based on Chao *et al.* proposal [70].

4.2.2.1 Data-set

The following custom dataset (4.6) was created from images available at the Internet (4.6), for exploring the DCNN performance in terms of accuracy, complexity and frame-rate. A manual selection process with super-pixels [51]

Figure 4.6: Data-set samples.

was used to create the different images annotations. An annotation relates to the desired output for each input image, it contains every pixel membership to a certain class.

The custom data-set contains fifty images of 480x360 dimensions, where ninety and ten percent of it are used for training and validation purposes.

4.2.2.2 Training

During segmentation, pixels are classified into background, strawberry and strawberry leaf. By adopting a specific class for background, it is possible to facilitate the fruit extraction from background. The encoder-decoder model is not only able to learn textures and features information from fruits but also from the background, that can become very complex in some situations. The algorithm obtains information from both classes which helps to make a more accurate segmentation in comparison with the OHTA cascade segmentation method introduced by Wei *et al.* [18]. Due to the small number of samples obtained from the custom dataset a *Dropout* layer was added after every convolutional and deconvolutional layer, to avoid interdependent learning among the neurons and take more advantage of the $encoder - decoder$ model [52].

Network training was carried out on a Desktop PC with an Intel Core i7-7700 Processor, 8 GB RAM and a Nvidia Geforce GTX 1080 GPU. Training was executed by running 220,000 steps, with four images by step, a dropout of 0.3 and a learning rate of 0.001, *Adam* optimizer was chosen due to its computationally efficient architecture and ability to deal with very noisy and/or sparse gradients [53], training took around of twenty hours approximately into an Ubuntu 18.04 OS, Python, and TensorFlow-GPU framework.

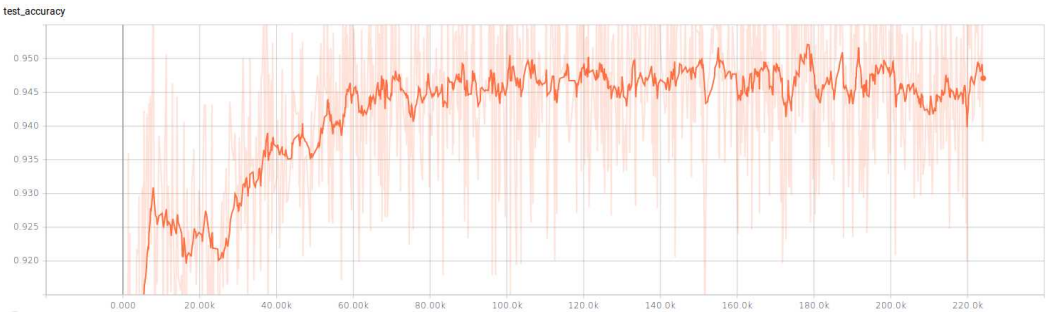

Figure 4.7: Training-Test Accuracy.

	Confidence Bounds ($\alpha = 0.05$)		
	Lower Bound	Upper Bound	Median
Trainig Dataset	0.945	0.969	0.957
Validation Dataset	0.922	1.011	0.967

Table 4.1: Accuracy Results

After the training phase, the algorithm obtained the accuracy results dysplayed in table 4.1. Notice that results presented in Fig. 4.7 very noisy and to facilitate their proper interpretation a smoothing filter has been applied to the shaded signal, which denotes the real value, the orange signal represents the smoothed output.

4.2.2.3 Results

The following section presents the results obtained with the simplified SegNet model at the strawberry segmentation task. Figure 4.8 demonstrates the results obtained from the deep encoder-decoder algorithm, images at:

- left side demonstrate the algorithm input X,
- middle side are the expected output Y,
- right side denote the algorithm output $\hat{Y}$.

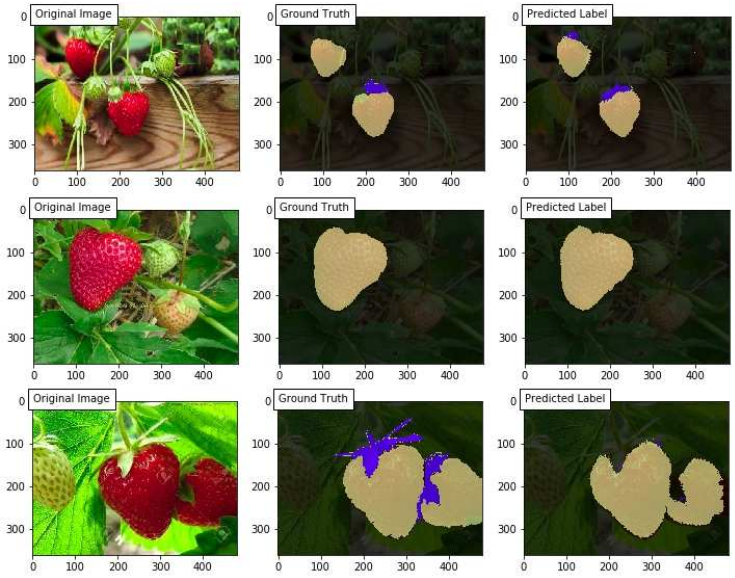

Figure 4.8: Results test samples.

The background class is denoted by black color, the strawberry class by yellow color and the strawberry leaves class by blue color. Figure 4.8 demonstrates the algorithm achievement with test images (unknown images for the network). It also possible to see the difficulties to extract the strawberry leaves due to its big similarity to the background class. Furthermore, The algorithm is able to recognize and segment strawberries successfully, which demonstrates its capacity for generalizing over unknown data to make accurate predictions.

For improving the algorithms accuracy during the strawberry identification, it is necessary to train the model with a larger dataset of images obtained from the application scenario (Fig. 4.9). Running the network inference process into a Dell, Alienware laptop with an Intel Core i5-6300 Processor, 8 GB RAM and a Nvidia Geforce GTX 1060 GPU reached five frames-per-second for segmenting a live video stream from the stereo camera.

4.2.3 Modified OHTA and Otsu Segmentation

After obtaining a successful output from the Ohta-Otsu algorithm in terms of performance and time-execution, a modified version has been introduced by filtering each Ohta channel with a Gaussian filter before of the adaptive thresholding calculation, this to obtain a better histogram distribution, facilitating the adaptive threshold estimation(Otsu).

Figure 4.9: Strawberries samples.

4.2.4 Segmentation Algorithms Comparison

By using the introduced strawberries database to compute different outputs for each algorithm, it is possible to note which algorithm achieves better results than the others based on the Structural SIMilarity (SSIM) index [69] of the image ground of truth and its corresponding output.

In this context, fig. 4.10 demonstrates that color segmentation algorithms are capable of extracting mature (red) strawberries from the background and discriminate them from immature (green) strawberries, strawberries which are not completely matured have demonstrated to be a big limitation of this algorithm. On the other hand, the encoder-decoder algorithm was able to extract mature (red) strawberries and eliminate strawberries that are not completely mature.

	Confidence Bounds ($\alpha = 0.05$)		
Algorithm	Lower Bound	Upper Bound	Median
OHTA + Otsu	0.878	0.929	0.904
OHTA + G. Filtering + Otsu	0.892	0.94	0.916
Deep Encoder-Decoder	0.945	0.968	0.957

Table 4.2: SSIM Results of the Segmentation Algorithms

Table 4.2 demonstrates that the encoder-decoder algorithm obtained the highest SSIM and the lowest variance rate. In this sense, deep learning algorithms achieved a more stable and accurate response by using its capability to learn images features for image segmentation tasks in complex scenarios. The modified OTHA have obtained a small improvement by adding the Gaussian

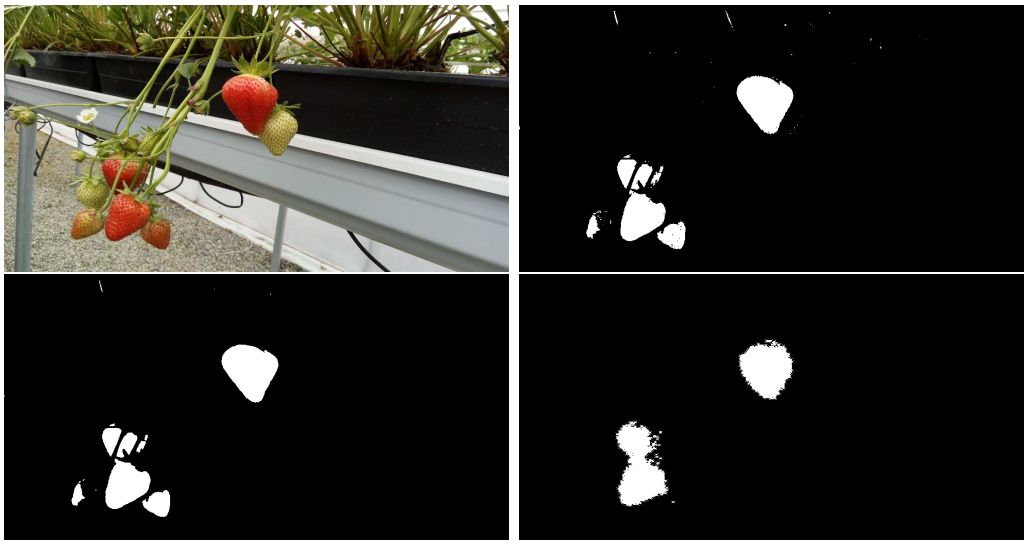

Figure 4.10: from left-to-right and top-to-bottom: Original Image, Result processed by OHTA and Otsu algorithm, Result processed by OHTA and Otsu modified algorithm, Result processed by encoder-decoder algorithm.

filter than the Wei proposal, but it still experiencing the same problem related to occlusion and false positives as shown in Fig 4.10.

4.2.5 Fruit Localization

In this section, we describe some aspects of design and practical implementation of the proposed visual servoing approach. By obtaining the same image feature in both cameras of the stereo vision system, we can compute the 3D Cartesian position of the target into the operational space using a triangulation technique [37]. In computer vision, a well-known problem consists on recognizing and matching points that belong to the same image feature in different scenes, along with object extraction or image segmentation [44]. Due to the irregular feature recognition performance presented by the ORB [54] and SURF [45] algorithm for extracting interest points into the images obtained by the stereo system, resulting into null results from the matching algorithm and other inconsistencies, it was necessary to simplify the feature extraction phase by using the fruit centroid to compute an approximate 3D position. Ensuring a fast and computationally cheap localization method. In the presence of many isolated fruits in the scene (non-fruits-clusters), the matching process was carried out by computing a homogeneous transformation (homography matrix) between both camera images. This homography matrix is calculated every time that the scene changes based on a similarity measure (mean squared error).

As noted, the object 3D position is an estimation, therefore, it is not

always possible to ensure the system stability by performing an end-point open-loop PBVS control scheme, to reach the corresponding 3D point coordinates obtained from the triangulation. Under this constraint, a HVS control scheme seems to be the more appropriate to ensure the reliability and safety of tasks performed by vision-based controllers.

To deal with the minimum depth range presented in the stereo vision system as well as the depth needed for collecting the fruit properly, a depth estimation method based on the object projected area is used to calculate the object depth with respect to the main camera.

4.3 HVS modeling approach

The control goal is to drive a set of features w to the desired values of the hybrid features w_d say:

$$w \to w_d\,, \qquad e_w = w_d - w \to 0\,, \tag{4-7}$$

where $e_w \in \mathbb{R}^3$ is the hybrid image feature error. The key idea consists of computing the position in the scene of the 3D points projected on the image plane of the two cameras using a triangulation technique [37]. Let $\bar{z}_c := \ln(z_c/z_d) \in \mathbb{R}$ be a supplementary normalized depth coordinate, where $\ln(\cdot)$ denotes the natural logarithm function. Taking the time-derivative of (3-5) and $\bar{z}_c$, and using (3-6) yields:

$$\dot{w} = L_w(w)\,\mathbf{v}_c\,, \qquad \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\bar{z}}_c \end{bmatrix} = L_w(w) \begin{bmatrix} v_c \\ \omega_c \end{bmatrix}, \tag{4-8}$$

with

$$L_w(w) = \begin{bmatrix} -\frac{1}{z_c} & 0 & \frac{x_p}{z_c} & x_p\,y_p & -(1+x_p^2) & y_p \\ 0 & -\frac{1}{z_c} & \frac{y_p}{z_c} & (1+y_p^2) & -x_p\,y_p & -x_p \\ 0 & 0 & -\frac{1}{z_c} & -y_p & x_p & 0 \end{bmatrix}$$

where $L_w(w) \in \mathbb{R}^{3\times 6}$ is the *interaction matrix* related to $w \in \mathbb{R}^3$, which denotes the 3D point coordinates expressed in the image and operational spaces.

Notice that, since the target object is assumed to be fixed with respect to the base frame $\mathcal{F}_b$ the desired values for image features are assumed to be constant, and changes in s depends only on camera motion.

Since the camera is attached to the robot end-effector (i.e., eye-in-hand configuration) and $\mathbf{v}_c = R_{bc}^{\mathsf{T}}\,\mathbf{v}$, we can combine (2-3), (2-7) and (3-8) obtaining the following control system:

$$\dot{w} = L_w(w)\, R_{bc}^{\mathsf{T}}\, J(q)\, u(t)\,. \tag{4-9}$$

From the equation (4-9), we can design the velocity control signal u as:

$$u(t) := J^{*}(q)\, R_{bc}\, L_w^{*}(w)\, \Lambda_w\, e_w\,, \qquad \Lambda_w = \Lambda_w^{\mathsf{T}} > 0\,, \tag{4-10}$$

where Λ_w is a proportional gain matrix, $J^{*}(q)$ and $L_w^{*}(w)$ are generic matrices to be properly defined in order to guarantee the asymptotic convergence of the image feature error e_w to zero, that is, $\lim_{t\to\infty} e_w(t)=0$.

Notice that, a computationally simple control algorithm can be derived by finding a suitable relationship between u and e_w that ensures error convergence to zero. The algorithm can be designed using the pseudo-inverse of Jacobian and interaction matrices, $J^{\dagger}(q)$ and $L_w^{\dagger}(w)$, in order to compensate $J(q)$ and $L_w(w)$, making the error system linear. In this case, implies that: (i) if $J^{*}(q) \equiv J^{\dagger}(q)$ the robot arm must be far from singular configurations, which is difficult to ensure, in practice, using the interaction matrix $L_w(w)$ represented in Cartesian or polar coordinates; (ii) if $L_w^{*}(w) \equiv L_w^{\dagger}(w)$, the HVS system must use one image feature to perform a 3-DoF taks or more than two image features, by stacking all interaction matrices, to perform a 6-DoF task and avoid some configurations for which $L_w(w)$ has deficient rank and

On the other hand, to avoid linearization of the error system, the algorithm can be designed using the *transpose* of the Jacobian and interaction matrices, $J^{\top}(q)$ and $L_w^{\top}(w)$. As a consequence, the algorithm is computationally more efficient and the error dynamics will be governed by a first-order nonlinear differential equation. In this case, implies that: (i) if $J^{*}(q) \equiv J^{\top}(q)$, it is possible to deal with kinematic singularities since the control algorithm does not require matrix inversion; (ii) if $L_w^{*}(w) \equiv L_w^{\top}(w)$, it is possible to cope with the ill-conditioning problem of the interaction matrix $L_w(w)$, even if the HVS system uses only two image features to perform a 6-DoF task.

4.4 HVS Robust Controller Design

In this section, the control schemes introduced in chapter two and three are mixed to obtain a robust HVS controller for achieving a picking task. The velocity performance of a given image feature $w \in \mathbb{R}^3$ into an uncertain camera-robot system and subject to external perturbations can be represented as

$$\dot{w} = \hat{L}_w(w)\, \bar{R}_{ec}^{\top}(\varphi)\, \bar{R}_{be}^{\top}\, \hat{J}(q)\, \dot{q} + \eta(w, q, \dot{q}) + d(t), \tag{4-11}$$

where $L_w(w) \in \mathbb{R}^{3\times 6}$ is the *interaction matrix* related to $w \in \mathbb{R}^3$, which denotes the 3D point coordinates expressed in the image and operational spaces; $\bar{R}_{ec}(\varphi) = diag(R_{ec}(\varphi), R_{ec}(\varphi)) \in \mathbb{R}^{6\times 6}$,$R_{ec}(\varphi) \in SO(3)$ means the *unknown*

planar rotation misalignment between end-effector and camera frames around the z-axis; $\bar{R}_{be} = diag(R_{be}, R_{be}) \in \mathbb{R}^{6\times6}$, R_{be} means the *known rotation* obtained by direct-kinematics ($h(q)$) between robot-base and end-effector frames; $J(q) \in \mathbb{R}^{6\times n}$ of a *n-DOF* robot arm, the Robot Jacobian provides the relationship between the joint velocities and the corresponding linear and angular velocities of the end-effector frame $\mathcal{F}_e$ with respect to the robot frame $\mathcal{F}_b$; and $\dot{q} \in \mathbb{R}^n$ denotes the robot joints velocity; $d(t) \in \mathbb{R}^3$ is a non-state dependent bounded function that describes common external disturbances that can perturb the proper functionality of the nominal system, commonly this external perturbations do not have an equilibrium point [41]; the camera-robot system uncertainties $\eta(w, q, \dot{q}) \in \mathbb{R}^{3\times n}$ are represented by

$$\begin{aligned}\eta(w,q,\dot{q}) = \hat{L}_w(w)\,\bar{R}_{ec}^{\top}(\varphi)\,\bar{R}_{be}^{\top}(J(q)-\hat{J}(q))\,\dot{q} \\ + (L_w(w)-\hat{L}_w(w))\,\bar{R}_{ec}^{\top}(\varphi)\,\bar{R}_{be}^{\top}\,\hat{J}(q)\,\dot{q} \\ + (L_w(w)-\hat{L}_w(w))\,\bar{R}_{ec}^{\top}(\varphi)\,\bar{R}_{be}^{\top}(J(q)-\hat{J}_b(q))\,\dot{q}\,, \quad (4\text{-}12)\end{aligned}$$

and can be separated as

$$\eta(w,q,\dot{q},\dot{}) = \eta_k(\dot{q}) + \eta_v(w) + \eta_{kv}(w,q,\dot{q})\,, \quad (4\text{-}13)$$

from left to right, the first and second terms $\eta_k(\dot{q})$, $\eta_v(w)$ are assumed to be bounded as mentioned in (2-33) and (3-15), respectively. For simplification purposes, a new term is introduced $J_w(w, \varphi, q) \in \mathbb{R}^{3\times n}$ denoted as the *image-jacobian*

$$J_w(w,\varphi,q) = \hat{L}_w(w)\,\bar{R}_{ec}^{\top}(\varphi)\,\bar{R}_{be}^{\top}\,\hat{J}(q)\,, \quad (4\text{-}14)$$

based on assumptions (2-33) and (3-15) the term $\eta_{kv}(w, q, \dot{q})$ can be denoted as

$$||\eta_{kv}(w,q,\dot{q})|| \leq \alpha_L\,\alpha_J J_w(w,\varphi,q)\,\dot{q} \qquad \forall q,\dot{q}\,, \quad (4\text{-}15)$$

and bounded as,

$$||\eta_{kv}(w,q,\dot{q})|| \leq \gamma\,||e_w|| \qquad \gamma \geq 0. \quad (4\text{-}16)$$

Then, defining the state variables as

$$w_1 = \int_0^t e_w(\tau)d\tau \qquad w_2 = e_w, \quad (4\text{-}17)$$

where $w_1, w_2 \in \mathbb{R}^3$ represent the continuous system states and $e_w \in \mathbb{R}^3$ denotes the system error vector the actual position w to the desired position w_d.

By taking

$$\dot{\zeta} = \begin{bmatrix} \dot{w}_1 \\ \dot{w}_2 \end{bmatrix}, \quad (4\text{-}18)$$

as the system state vector $\zeta \in \mathbb{R}^6$, the following differential equation is obtained:

$$\dot{\zeta} = A\,\zeta + B\,(\eta(w,q,\dot{q}) + d(t) - J_w(w,\varphi,q)\,u(t)) \quad (4\text{-}19)$$

where $A \in \mathbb{R}^{6\times6}$ and $B \in \mathbb{R}^{6\times3}$ seted as:

$$A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{4-20}$$

with reference to (4-11), the following control signal $u(t) \in \mathbb{R}^3$ is noted as

$$\dot{q} := u(t) = \hat{J}_w^{\top}(s, \hat{\varphi}, q)\ (\Lambda_w\, \zeta + \Psi)\,, \tag{4-21}$$

and

$$\hat{J}_w(s, \hat{\varphi}, q) = \hat{L}_w(w)\, \bar{S}_l(\hat{\varphi})\, \bar{R}_{be}^{\top}\, \hat{J}(q), \tag{4-22}$$

where $\Lambda_w \in \mathbb{R}^{3\times6} = [\Lambda_{w1}\ \Lambda_{w2}]$ and Λ_{w1} and Λ_{w2} are positive definite gain matrices; $\bar{S}_l(\hat{\varphi}) = diag(S_l(\hat{\varphi}), S_l(\hat{\varphi}) \in \mathbb{R}^{6\times6}$ and $S_l(\hat{\varphi}) \in SO(3)$ is a switching matrix which depends of a discrete state $l \in \{0, 1\}$; and $\Psi \in \mathbb{R}^3$ represents a compensation term that will be explained later. Introducing the control signal $u(t)$ to the nominal system (4-19),

$$\dot{\zeta} = \tilde{A}_l\, \zeta + B(\eta(w, q, \dot{q}) + d(t) - \Psi) \tag{4-23}$$

where

$$\tilde{A}_l = \begin{bmatrix} 0 & I \\ -J_w(s, \varphi, q)\, \hat{J}_w^{\top}(s, \hat{\varphi}, q)\, \Lambda_{w1} & -J_w(s, \varphi, q)\, \hat{J}_w^{\top}(s, \hat{\varphi}, q)\, \Lambda_{w2} \end{bmatrix}, \tag{4-24}$$

$\tilde{A}_l \in \mathbb{R}^{6\times6}$ and needs to be Hurwitz to guarantee the asymptotic error convergence of the camera-robot system $\lim_{t\to\infty} e_w(t) = 0$.

4.4.1 HVS controller Verification and Validation

In this section, we present simulation results for a robotic fruit harvesting task. The simulations tests were carried out considering the presence of parametric uncertainties in the Jacobian matrix J, interaction matrix L_w and rotation matrix R_{ec}. We also assumed that robot end-effector is moving in the neighborhood of singular configurations.

The robustness of the control algorithm will be evaluated by choosing the misalignment angle between the camera and the end-effector frames around the z-axis as $\phi = \pi/6\ rad$ and 10% of uncertainty in the length of the last link of the robot arm as well as in the camera intrinsic parameters. The control goal is to drive the object image feature w to the desired image feature w_d located at the camera center point (x_{v0}, y_{v0}) with desired depth $z_d = 0.1\ m$. The numerical simulations were executed in MATLAB and V-REP robot simulator (see Fig.4.11). To illustrate the performance and effectiveness of the HVS control scheme, simulations results are shown in Figures 4.12-4.15.

Figure 4.12(a) and (b) shows the behavior over time of the position of

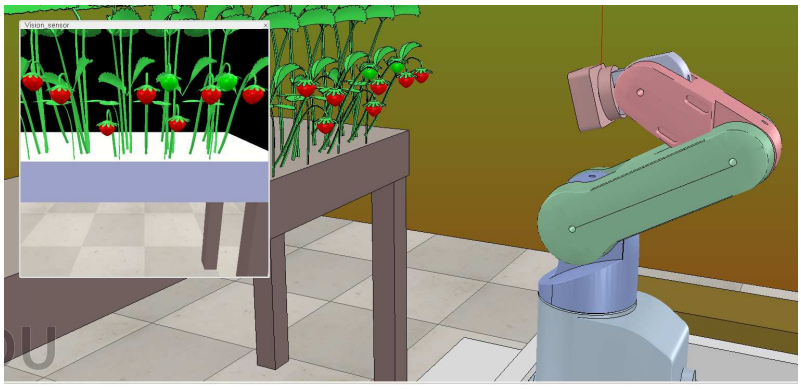

Figure 4.11: Robotic fruit picking tasks on V-REP robot simulator.

the image feature and the position error during the regulation tasks, where we can observe a slight disturbance at the beginning of the simulation due to the existence of the misalignment between the camera and end-effector frames in the z-axis. We can also note the asymptotic convergence of both signals to zero. The time history of the HVS control signal is illustrated in Fig. 4.13(a)

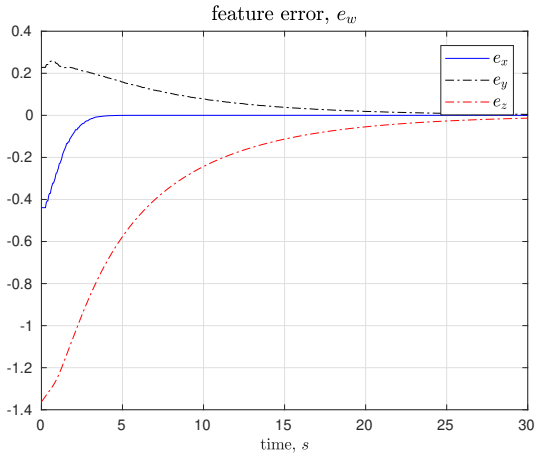

Figure 4.12: Regulation task: feature position error.

and (b), where it is possible to verify the stable behavior of the linear and angular velocity signals, obtained using the HVS control scheme. Figure 4.14, (a)-(e), presents the behavior in time of the angular velocities of the robot joints, where we can see the smoothness of the control signals even if the robot is close to singular configurations. Figure 4.15 shows the motion of multiple image features (strawberries) from a given initial position "$*$" to a desired final position "$\circ$", it is possible to verify the satisfactory performance of the proposed control scheme, in spite of the existence of parametric uncertainties in the camera-robot system.

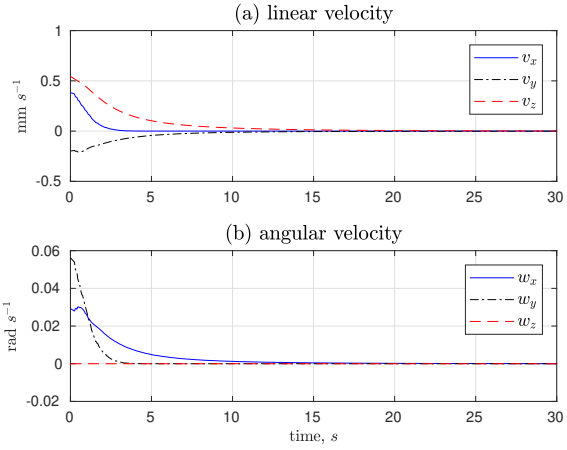

Figure 4.13: HVS control signal: (a) linear velocity; (b) angular velocity.

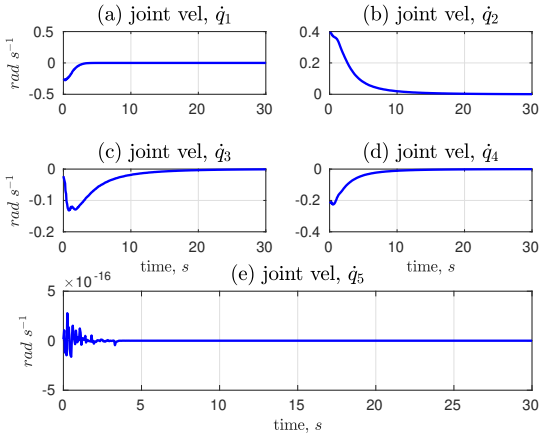

Figure 4.14: Joint angular velocities.

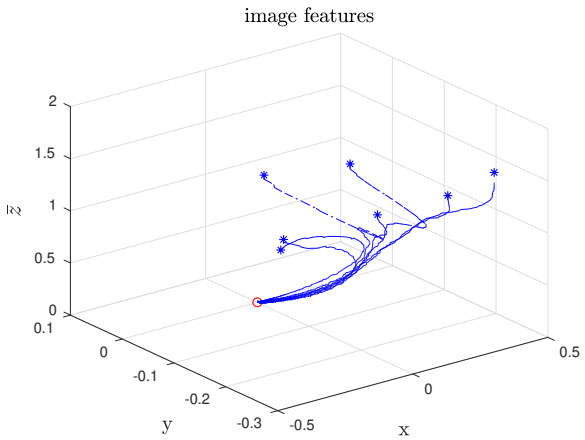

Figure 4.15: Trajectory of the image features: initial position "$*$" and final position "$\circ$".

4.4.2 Hybrid Control Design

Regarding the discrete state $S_l(\hat{\varphi})$ for assuring the asymptotic stability of the system even when $|\varphi - \hat{\varphi}| \geq |\frac{\pi}{2}|$, which violates the Hurwitz property of the matrix $\tilde{A}_l$ (4-24) turning the system into unstable.

Bringing the System to a Hybrid control system formalism and using a basic switching approach of two operating regions X_l, $l = \{0, 1\}$; we obtain

$$f_l(\dot{\zeta}) = \tilde{A}_l\,\zeta + B(\eta(w, q, \dot{q}) + d(t) - \Psi)\,, \qquad l \in L = \{0, 1\}\,, \tag{4-25}$$

where the vector $f_l(\dot{\zeta})$ represents the dynamics of the $l - th$ mode into a continuous function, and L is a finite index set. For a state-dependent switching task, the switching times are denoted as k and it is assumed that for every discrete index N_L, exists an operating region X_l limited by a guard set $G(l, l')$. Note that, a discrete transition to l_{k+1} occurs when the continuous state ζ into operating region X_l meets its corresponding guard set $G(l, l')$ [47].

By defining the discrete state $S_l(\hat{\varphi})$ as

$$\mathcal{S}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathcal{S}_1(\hat{\varphi}) = \begin{bmatrix} cos(\hat{\varphi}) & -sin(\hat{\varphi}) & 0 \\ sin(\hat{\varphi}) & cos(\hat{\varphi}) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4-26}$$

where $\hat{\varphi}$ is an estimation of φ to ensure the Hurwitz property of $\tilde{A}_l$ (4-24), this $\hat{\varphi}$ estimation will be discussed later, and setting two discrete state values under the constraint (3-21), the following operating regions into the radiant space are obtained $X_0 = [\frac{-\pi}{2}, ..., \frac{\pi}{2}]$, $X_1 = [\hat{\varphi} - \frac{\pi}{2}, ..., \hat{\varphi} + \frac{\pi}{2}]$.

Analyzing the continuous system state ζ we have that for any $t_i \in [0, t_M]$ and, by using the comparison lemma [41], we obtain:

$$||\zeta(t)|| \leq \xi(t)\,, \qquad \forall t \in [t_i, t_M]\,, \tag{4-27}$$

where

$$\xi(t) := ||\zeta(t_i)(t_i)||\, e^{-\lambda_m(t - t_i)}\,, \tag{4-28}$$

where λ_m is a positive constant.

Constructing the guard set $G(l, l')$ as a monitoring function $\psi_m(\cdot)$ based on the norm bound for the feature error ζ given in (4-27) following the ideas introduced in [48, 46]. Notice that, (4-27) holds when the matrix S_l is correct ($S_l = S$), it seems natural to use the term $\xi(t)$ as a benchmark to decide whether a switching of S_l is needed, that is, the switching occurs only when the condition (4-27) is violated. On the other hand, since S_l is *unknown*, the following function ψ_k is defined in the interval $[t_k, t_{k+1}]$, to replace the term $\xi(t)$ as:

$$\psi_k(t) = ||\zeta(t_k)||e^{\lambda(t - t_k)} + \gamma\, a(t_k)\, e^{\frac{-t}{a(t_k)}}\,, \tag{4-29}$$

where the switching time t_k sets the change of index $l \in L$, cycling through the S_l matrices for $l = 1, 2$, and $a(t_k)$ is any positive monotonically increasing unbounded sequence. The monitoring function ψ_m can be defined as:

$$\psi_m(t) := \psi_k(t)\,, \quad \forall t \in [t_k, t_{k+1}] \subset [t_0, t_M]\,. \tag{4-30}$$

Note that, from (4-29) and (4-30), we have $||\zeta(t_k)|| < \psi_k(t_k)$ at $t = t_k$. Hence, the switching time t_k is defined as the time instant when the state $||\zeta(t)||$ meets the guard set $G(l, l')$ (monitoring function $\psi_m(t)$) , that is,

$$t_{k+1} := \begin{cases} t_M, & \text{if} \quad ||\zeta(t)|| \geq \psi_m(t), \\ t_k, & \text{otherwise}, \end{cases} \tag{4-31}$$

where $k = 0, 1, \cdots$ and $t_0 := 0$. From (4-30), the following inequality can be obtained:

$$||\zeta(t)|| \leq \psi_m(t)\,, \quad \forall t \in [0, t_M]\,. \tag{4-32}$$

4.4.3 Stability Analysis

Considering the Lyapunov candidate function

$$V_l(\zeta) = \zeta^\top Q \zeta > 0 \qquad \forall \zeta \neq 0\,, \zeta \in X_l\,, \tag{4-33}$$

where $Q \in \mathbb{R}^{6\times 6}$ is a positive definite matrix, deriving (2-57) we obtain

$$\dot{V}_l(\zeta) = \zeta^\top (\tilde{A}_l^\top Q + Q\tilde{A})_l \zeta + 2\zeta^\top QB(\eta(w, q, \dot{q}) + d(t) - \Psi)\,. \tag{4-34}$$

Since $\tilde{A}_l$ is Hurwitz, it is necessary to compute a positive definite matrix P_l as:

$$\tilde{A}_l^\top Q + Q\tilde{A}_l = -P_l\,, \tag{4-35}$$

injecting (4-35) into (4-34), $\dot{V}_l(\zeta)$ leads to

$$\dot{V}_l(\zeta) = -\zeta^\top P_l \zeta + 2\zeta^\top QB(\eta(w, q, \dot{q}) + d(t) - \Psi)\,. \tag{4-36}$$

Regarding the left term of (4-36), it is possible to see that it is negative definite $\forall \zeta \in X_l$, for the second term, it is necessary to choose Ψ for vanishing the *perturbed term* $(\eta(w, q, \dot{q}) + d(t))$ along the task execution [42]. Adopting the vector Ψ as

$$\Psi = \rho_1 \sqrt{|Z|}\, sign(Z) + \upsilon + \varrho Z \qquad \varrho > 0\,, \tag{4-37}$$

$$\dot{\upsilon} = \rho_2\, sign(Z)\,, \tag{4-38}$$

and for simplicity, the parameters $\rho 1$ and $\rho 2$ can be set as

$$\rho_1 = \sqrt{\rho} \qquad \rho_2 = 1.1\,\rho \qquad \rho > 0, \tag{4-39}$$

where ρ and ϱ are positive and sufficient large constants for vanishing any perturbation added to the nominal system [49]. In this sense the *perturbed term* is denoted as

$$||(\eta(w, q, \dot{q}) + d(t))|| < ||\Psi|| \tag{4-40}$$

and the Lyapunov function (4-36) can be rewritten as

$$\dot{V}_l(\zeta) \leq -\zeta^\top P_l \zeta - ||\Psi||\, 2\, \zeta^\top Q\, B \qquad \forall \zeta \in X_l\,. \tag{4-41}$$

Being that the rotation matrix $R_{ec}(\varphi)$ is constant during the task execution ($\dot{\varphi} = 0$) and by using the switching scheme introduced in 4-25, only one switching time $k+1$ is expected in case that $||\varphi - \hat{\varphi}_k|| > \pi/2$ for ensuring $||\varphi - \hat{\varphi}_{(k+1)}|| < \pi/2$ after switching, then

$$V_{l_{(k+1)}}(\zeta(k+1)) \leq V_{l_{(k)}}(\zeta(k)) \tag{4-42}$$

guaranteeing the system stability and error convergence to zero $\zeta = 0$ $\lim_{t\to\infty}$ before and after the switching occurs $k+1$.

4.4.4 Validation HVS Hybrid and ST-SM

In this section we demonstrate the introduced ST-SM controller to track an image feature w_d which is assumed to have an unknown and bounded $\dot{w}_d$ with the following assumptions,

- A ten percent uncertainty at the length of the last link.
- A ten percent uncertainty at the intrinsic camera parameters.
- Misalignment angle $\tilde{\varphi}$ between the camera and the end-effector frames around the z-axis of 3.32 *rad* as commented in (3-21).
- An unknown circular trajectory of π rad/s.

Figs. 4.16-4.19 demonstrate the ST-SM controller performance, where the projection error e_w remains close to zero along time, even when the desired trajectory $\dot{s}_d$ is unknown. Moreover, Fig. 4.17 demonstrates the unit vector control Ψ and the sliding surface Z performance of the proposed algorithm in (3-42), it is possible to see how the unit vector Ψ presents a more continues performance in comparison with the UVC algorithm. Finally, Fig.4.19 demonstrates the smooth performance the robot joints velocities.

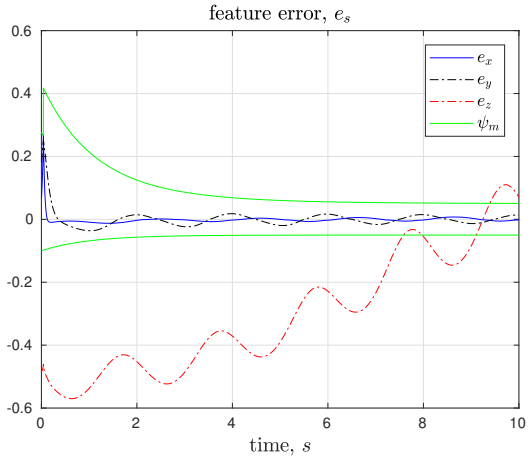

Figure 4.16: ST-SM Feature Error.

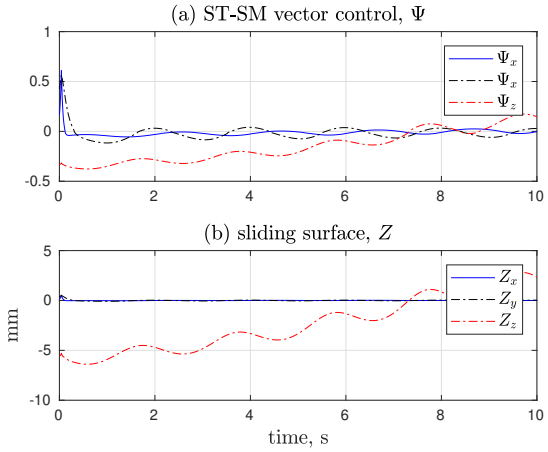

Figure 4.17: ST-SM Sliding Surface and unit vector control signal.

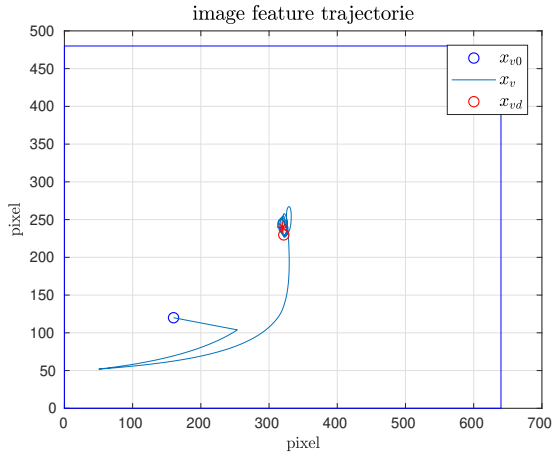

Figure 4.18: ST-SM Image Trajectory.

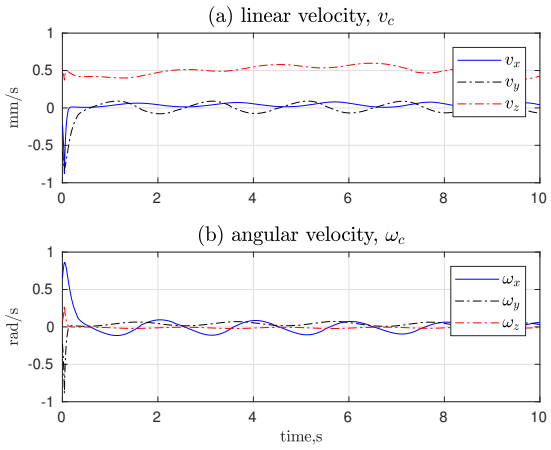

Figure 4.19: ST-SM End-Effector Velocity.

4.5 Task Optimizer

By having a stable control loop for collecting one strawberry, a practical situation yields on collecting more than one strawberry in a complex real-world scenario, where many strawberries may be on the screen and a harvest-planning phase may not be trivial for collecting the strawberries as fast as possible. As expected, while performing the HVS control scheme to approach an object of interest, the others objects are considered as external disturbances to the control loop, which can lead to system instability.

In order to discover the best-path for collecting the objects, an optimization scheme is introduced. The main idea is to obtain an initial path and then optimize it until obtaining an approximation to the best-path. This optimization procedure is accomplished based on the cost function, which is defined as path cost, which means the sum of the distances between each object following the test-path. The candidate path, the numbers in red shown in Fig. 4.20, is calculated by choosing the closest point starting with the first image centroid obtained from the segmentation phase.

Path optimization is carried out by using a learning exploration/exploitation strategy using a memetic random restart strategy [55], which tries to minimize the cost function (or path-cost) by choosing a random permutation at an exploration phase or modifying randomly the test path for exploitation. Then, the optimization process stops when the cost function converges ("Fully Optimized") as shown in Fig. 4.21. Since this is a stochastic optimization process, running the algorithm a certain number of times may improve the obtained results.

In order to deal with the disturbances presented during the approaching

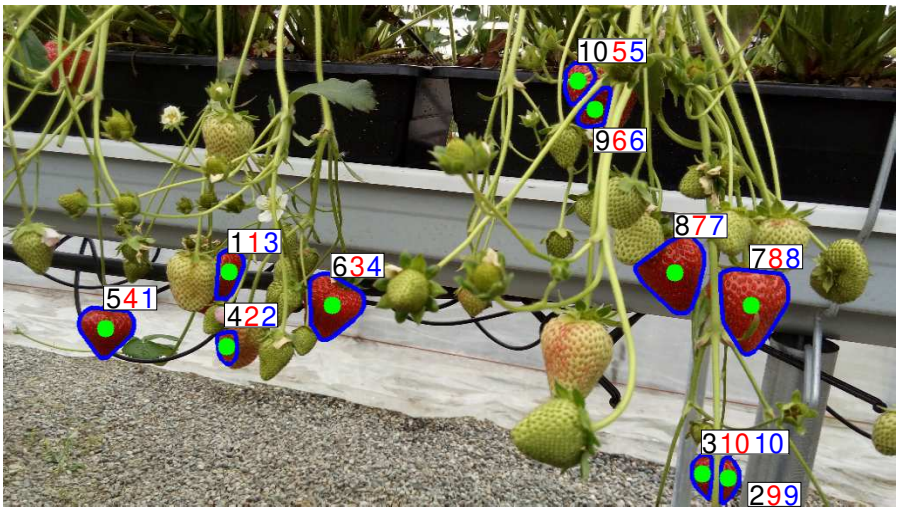

Figure 4.20: Harvesting paths: The numbers in black, red and blue, refer respectively to initial, candidate and optimized paths.

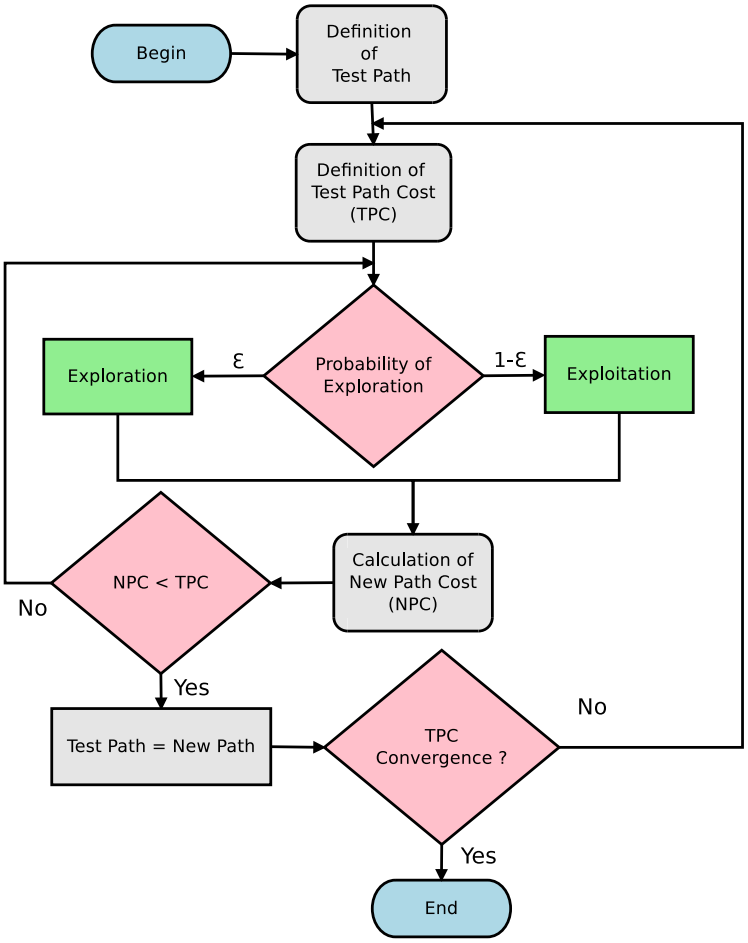

Figure 4.21: Memetic Algorithm Ramdom Restart.

phase towards a single object of interest in the presence of other fruits, a self-updating tracking window [56] is used to ensures the object visibility and be able to compute its corresponding image features. After reaching the desired distance, a final image-based height adjustment could be carried out for positioning the gripper close to the fruit stem, cut it and store the fruit in a storage box, completing the harvesting task successfully.

4.6 Experimental Results HVS

In this section, we present experimental results for a fruit harvesting task carried out in order to illustrate the effectiveness of the proposed control scheme (Fig. 4.22). We consider a RGB-D stereo camera attached to the end-effector of a 5-DoF robot arm, which is visually controlled at the position level by a hybrid-based visual servoing approach in the neighborhood of singular configurations.

Numerical simulations were carried out on a Lenovo laptop with Intel Core i7-6500U Processor (4M Smart Cache, 2.5 GHz) 16 GB RAM, running Windows OS 64 bits. The control algorithm was implemented in MATLAB/Simulink (The MathWorks Inc.) Release 2017a and V-REP PRO EDU version 3.4.0 used as a robotic simulation platform. A set of scripts and function blocks were created to perform all necessary calculations and execute the control loop. Experiments were carried out using the Mitsubishi robot RV-2AJ and the controller Mitsubishi Melfa CR1. The communication channel between the laptop and the controller was established through RS-232C command protocol by means of an own-built cable, based on manufacturer manual. The camera attached to the end-effector was the ZED Mini Stereo Camera distributed by STEREOLABS. The intrinsic camera parameters of the visual servoing system were provided by the manufacturer (ROS package) The Denavit-Hartenberg parameters [37] of the robot arm are: $\alpha_1 = -\pi/2$, and $\alpha_4 = \pi/2$ and the offsets of joints two and four are $-\pi/2$ and $\pi/2$, where all angles are expressed in radians; $a_2 = 0.25$, $a_3 = 0.16$; $d_1 = 0.3$, and $d_5 = 0.072$, where all lengths are expressed in meters.

Figure 4.22: Laboratory Experimental Setup.

4.6.1 ROS Architecture

The HVS algorithm was implemented with the Robot Operating System (ROS) framework, in this way it was possible to use the camera manufacturer ROS package and communicate different Python scripts in parallel. Figs 4.23 and 4.24 presents the system nodes and topics implemented for ensuring a successful execution. The ROS implementation contains four scripts, three in Python (image segmentation, homography generator, and the robot controller) and one in Matlab (HVS controller), all of them have been added to a single launch file for simplicity purposes their initialization.

4.6.1.1 Feature Matching Script

This script intends to solve the matching points problem by employing the scene Homography, the Homogeneous transformation (Homography) H is

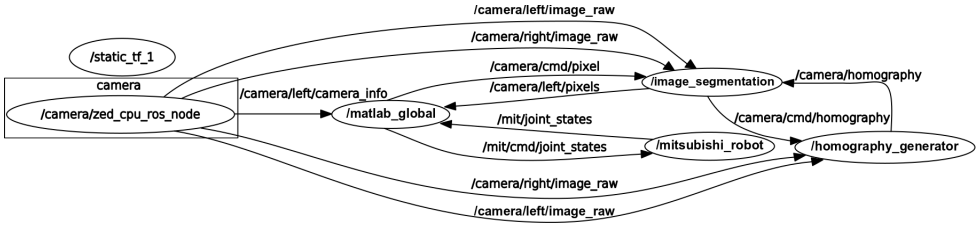

Figure 4.23: ROS nodes.

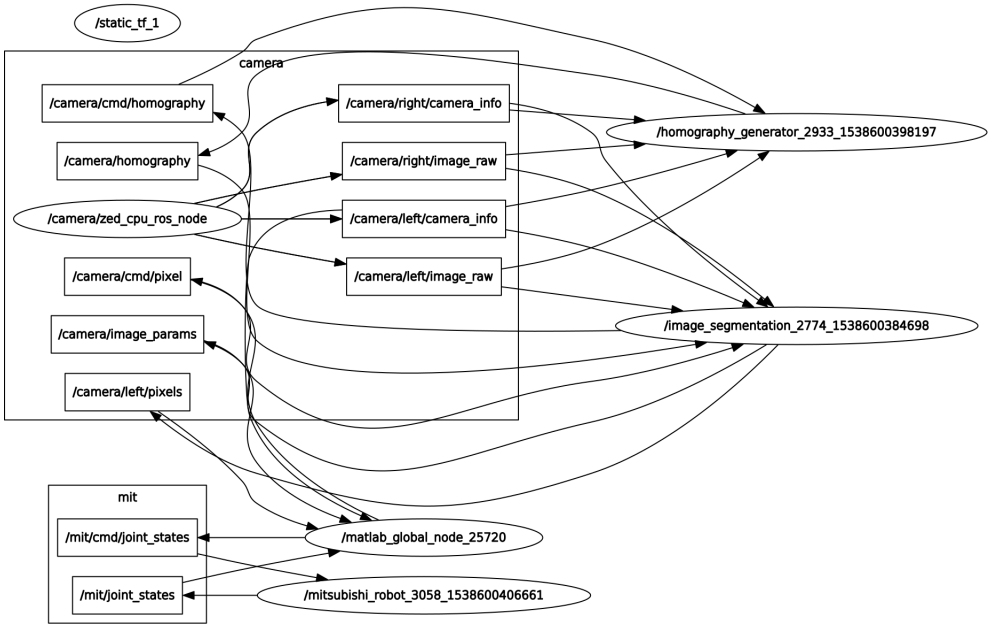

Figure 4.24: ROS nodes and topics.

used to map a given pixel location p_l from one view (left camera) to another p_r (right camera), this Homography mapping is an approximation hence the corresponding point is assumed to be the closest to this estimation $\hat{p}_r$ (4.25). In consequence that the Homography $\hat{H}$ is an estimation of H which is unknown, we assumed that the distance between $\hat{p}_r$ and p_r should be the zero in theory but in practice, this distance needs to be restricted by a certain threshold for wrong $\hat{H}$ estimations; provided that there are not any close point p_r to $\hat{p}_r$, the algorithm estimates a new homography $\hat{H}$ until reaching p_r [44].

4.6.1.2 Image Segmentation

This script (4.26) is intended to extract the desired object to track (specified by the HVS controller), during the task execution; and by using the scene Homography estimation $\hat{H}$ triangulate accurately the 3D position of the object/s of interest presented in the scene (4.26). Figures 4.27 and 4.28, demonstrate the input images before and after a specific object is chosen by

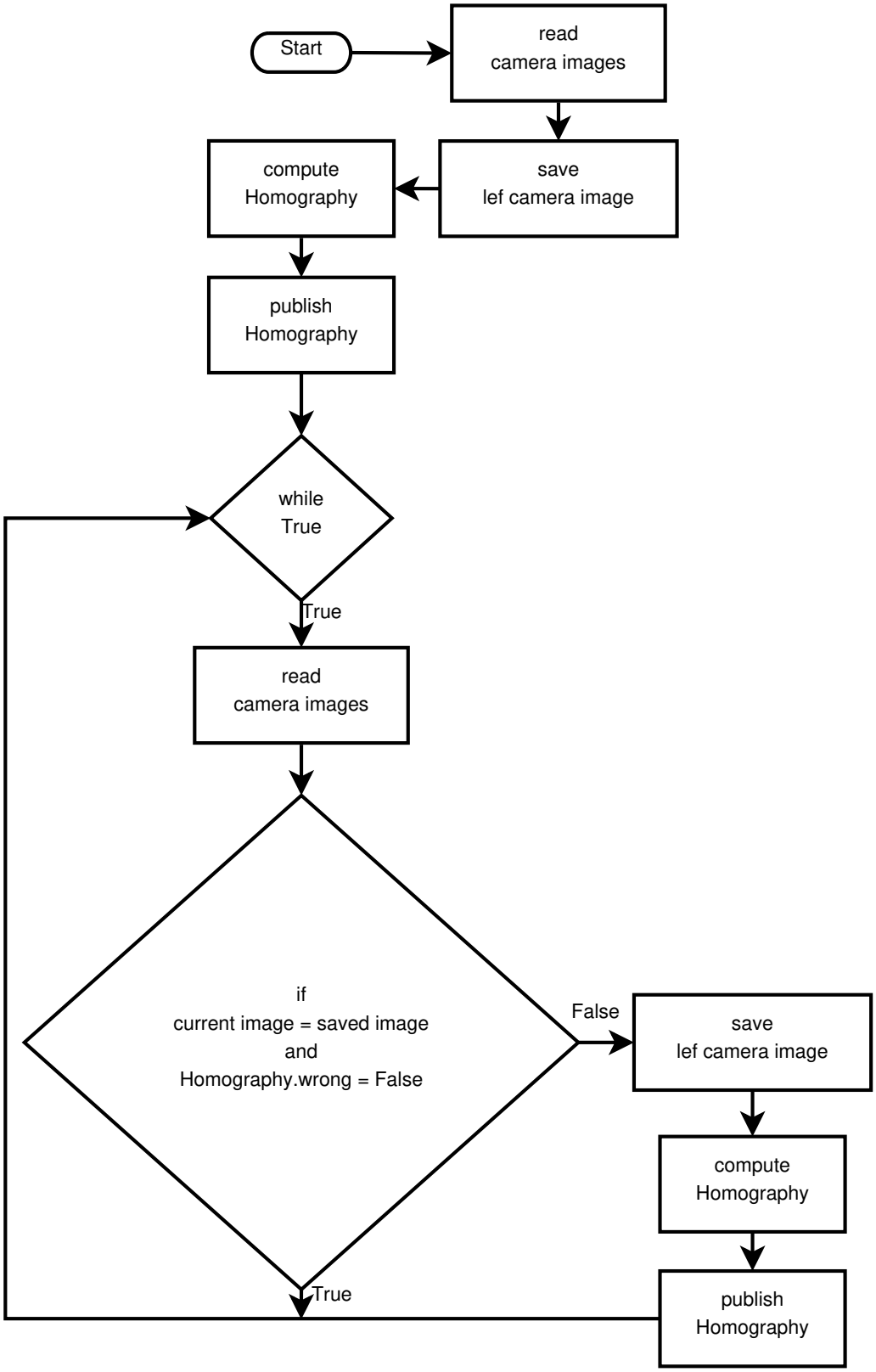

Figure 4.25: Homogeneus Transformation Script.

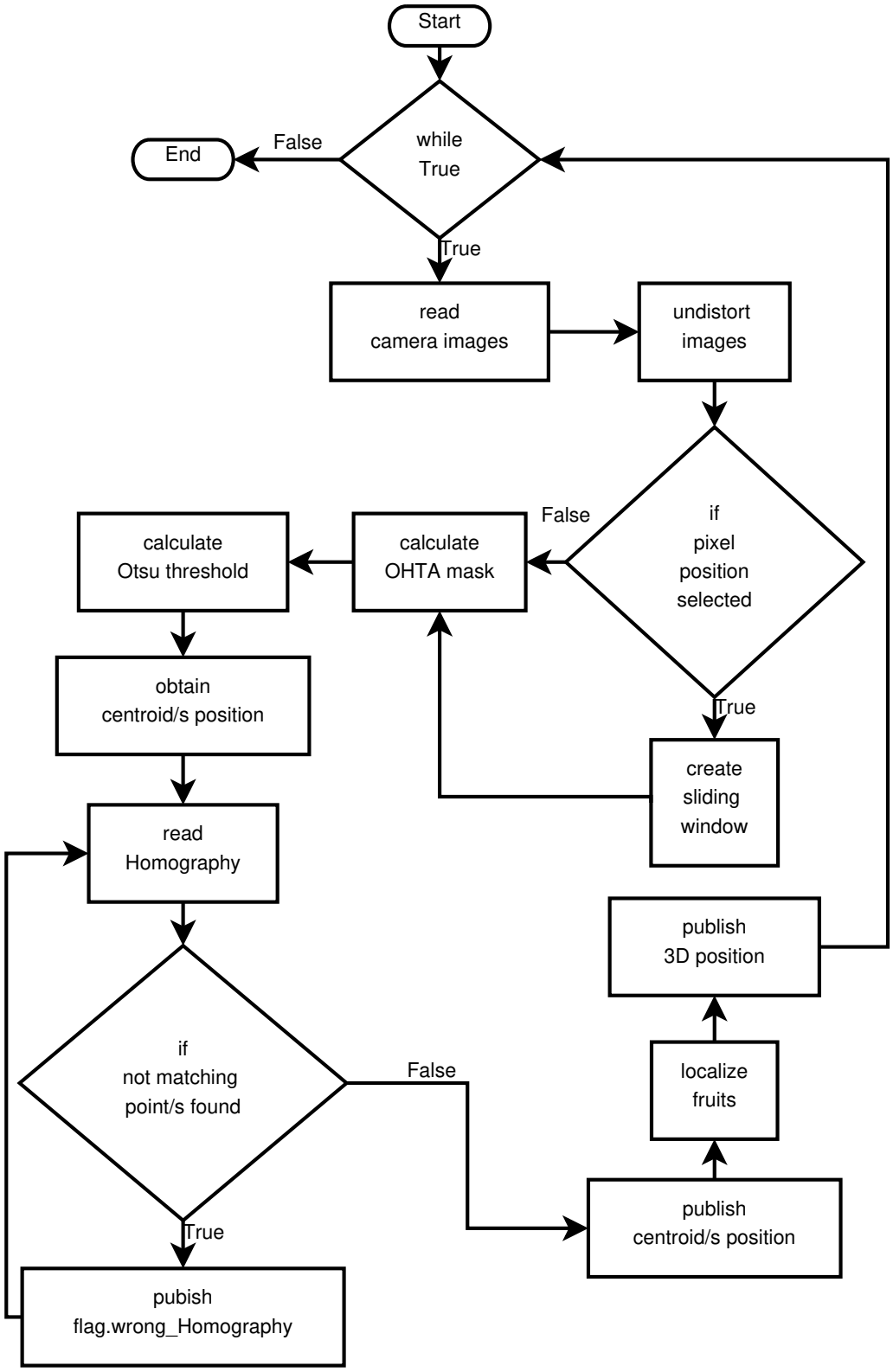

Figure 4.26: Image Segmentation Script.

Figure 4.27: Image Before Selecting an Object (left and right camera images).

Figure 4.28: Image After Selecting an Object (left and right camera images).

the HVS controller, the sliding window is employed to remove similar objects from the scene to execute the hybrid visual servoing application for an specific fruit.

4.6.1.3 HVS Controller

This algorithm (Fig. 4.29) estimates the desired end-effector trajectory to achieve the visual servoing task, the loops starts by reading the identified features from the image segmentation algorithm (4.26) and after choosing the object of interest into the scene, the algorithm starts computing the manipulator trajectory until reaching the desired position w_d.

4.6.1.4 Experimental Results

In this section we present the proposed HVS controller performance with the following control law,

$$u(t) := \dot{q} = J^{\top}(q) R_{bc} L_w^{\top}(w) \Lambda_w \, e_w \,, \qquad \Lambda_w = \Lambda_w^{\top} > 0 \,, \tag{4-43}$$

with uncertainties at the Jacobian $J(q)$ and Iteration $L_w(w)$ matrices and in presence of similar objects into the scene.

Figures 4.30 - 4.35 demonstrate the proposed algorithm performance; figure 4.31 shows the estimate 3D position convergence of a fixed object, this 3D estimation (object position referred to robot base frame) was calculated

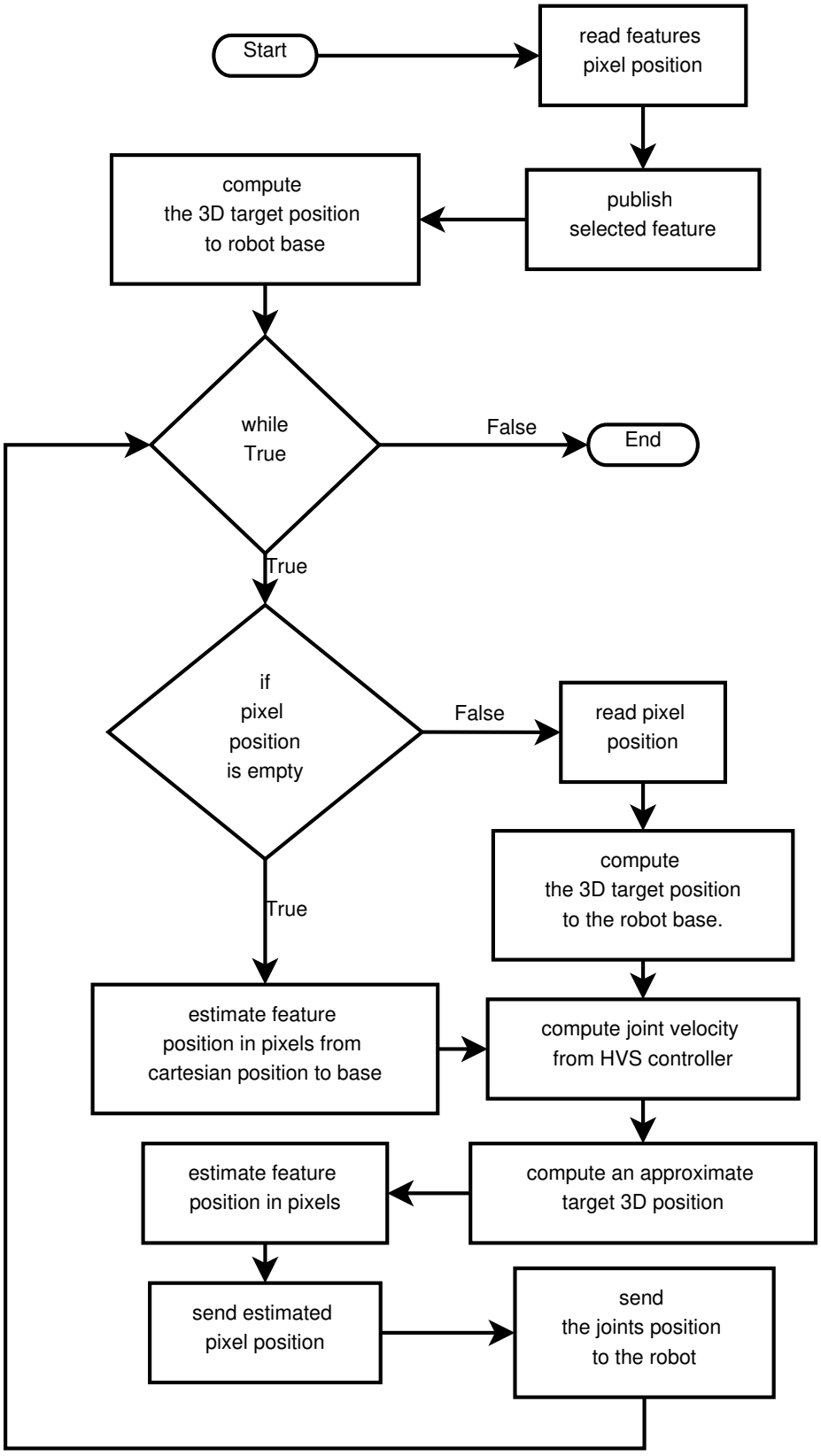

Figure 4.29: HVS Controller Script.

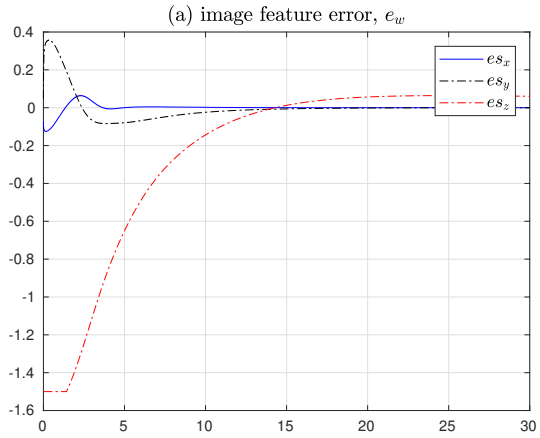

Figure 4.30: HVS Controller Error Performance.

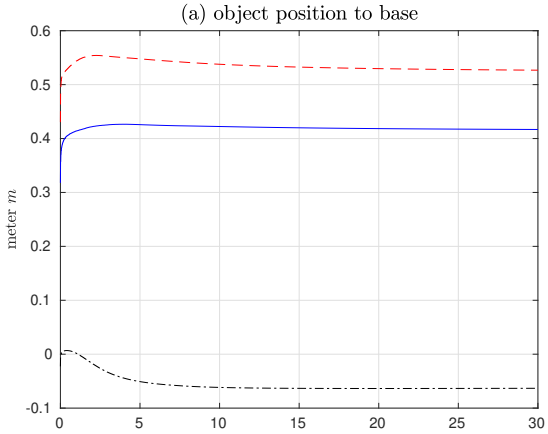

Figure 4.31: HVS Controller Object Position.

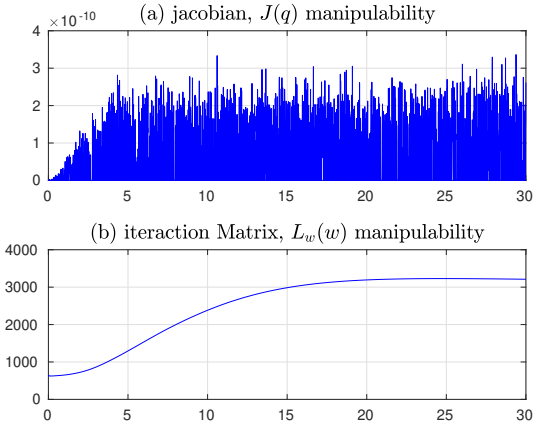

Figure 4.32: HVS Controller Jacobian and Iteration Matrices Manipulability.

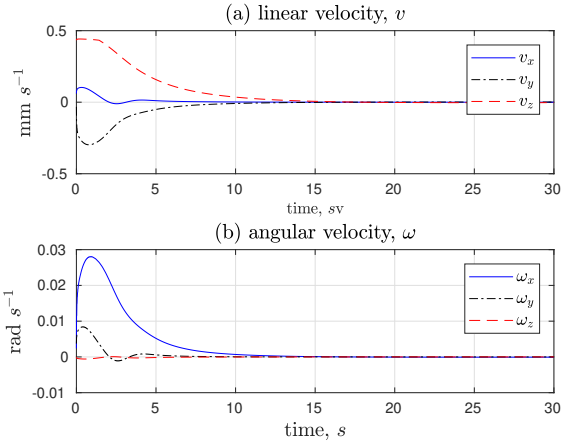

Figure 4.33: HVS Controller End-Effector Velocity.

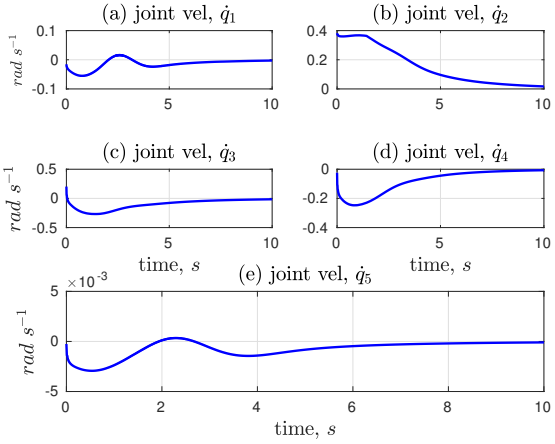

Figure 4.34: HVS Controller Joints Velocity.

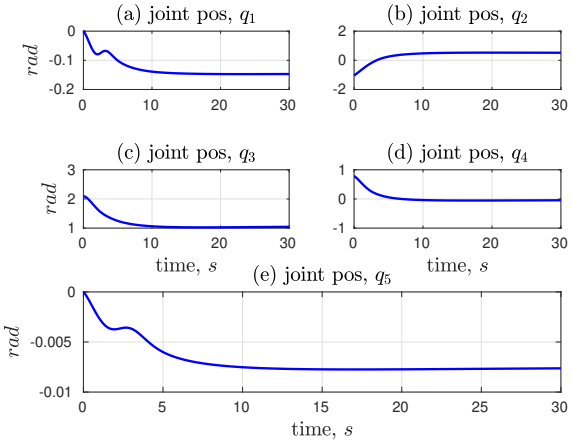

Figure 4.35: HVS Controller Joints Position.

with an uncertain distance between the camera and manipulator frames; figure 4.31 present the Jacobian an Iteration matrices manipulability during the task execution, note that the HVS controller is able to deal with singular configurations during the task execution, performing the a smooth control signals (Figs. 4.33 and 4.34).

4.7 Concluding Remarks

In this chapter, we have demonstrated that the HVS is a suitable and robust option to cope with the ill-conditioning problem of the Jacobian and interaction matrices, and also to deal with parametric uncertainties for regulation tasks. Moreover, a sliding mode control (SMC) combined with a switching monitoring algorithm and state observer have been added to the HVS controller to deal with the usual limitation of the camera misalignment angle $|\tilde{\varphi}| < \pi/2$ rad and external unmodeled perturbations.

The segmentation algorithm based on the OTHA color space combined with the Otsu method has demonstrated a good and fast segmentation performance for online applications where time execution is critical for the control reliability. On the other hand, DCNN algorithms have shown a more accurate and computationally expensive segmentation into very dynamic and complex scenarios.

In situations where harvesting task turns into collecting more than one fruit, by using a path planning algorithm, a Matching algorithm based on the scene Homography and a self-updating tracking window approaches, it was possible to demonstrate a satisfactory performance of the proposed solution into a test harvesting setup.

Finally, the ROS framework was employed to achieve a real harvesting task with the Mitsubishi robot, where many parallel tasks or nodes are needed to run in parallel threads and to communicate with each other. Image-processing and Homography scripts were implemented in Python and for time-consuming reasons, the HVS controller was implemented in Matlab.

5
Conclusions

In this work, we have developed a hybrid visual servoing control scheme based on PBVS and IBVS approaches, for robotic fruit harvesting tasks in the presence of parametric uncertainties in the camera-robot system.

In chapter two we comment how convenient is to choose the Jacobian-transpose algorithm for solving the inverse-kinematics problem at singular joints q configuration into a regulation task. Furthermore, adaptive and robust control (sliding mode) methodologies where compared into a tracking task in presence of parametric uncertainties of a two-link planar robot, these to determine the advantages and disadvantages of both methods with uncertain models. In this context, the UVC robust control scheme have demonstrated a better performance compared with the adaptive controller for performing a regulation control task in presence of constant external perturbations $\dot{p}_d \neq 0$.

Chapter three introduces the PBVS and IBVS controllers for archiving a regulation task by using a camera attached to the robot end-effector, experimental results of the first prototype based on PBVS and IBVS indicate the PBVS low-robustness properties to modeling-errors compared to the IBVS scheme. Subsequently, a more robust IBVS controller is presented based on ST-SM and SMC to ensure a successful task execution under vanishing and non-vanishing perturbations existence. Moreover, the chattering phenomenon is studied and minimized by employing an ST-SM which combines continuous and non-continuous control laws.

Based on robustness properties of the IBVS scheme over the PBVS approach commented in chapter three, chapter four presented an HVS design is proposed for achieving a desired 3D position with respect to an object of interest. In this ST-SM was added to the control law for obtaining robustness to model uncertainties and external perturbations. In addition, a semi-adaptive switching SMC controller based on the state observer strategy has demonstrated a better performance for minimizing the switchin times in comparison with the approach used in chapter three.

Numerical simulations and preliminary practical results have shown the efficiency and feasibility of using robot arms to perform semi-autonomous harvesting tasks for soft fruits in orchards and poly-tunnels. We expected

that in the near future other types of crops - that introduce different types of challenges - may be harvested by the robot arm, and others kinds of agricultural tasks can be also implemented by using robotic arms, such as weed control.

Some proposed topics for further investigation involve:

- Improve the segmentation algorithm to obtain a stem detection and segmentation to perform fruit harvesting under different stems configurations and rotations. Since just by finding the top of the fruit and adding a small value to the height it is possible to pick only fruits with a stem in the upright position;
- Perform the undistortion process to the extracted image features instead of the whole image to obtain a computationally cheaper segmentation algorithm;
- Examine other DCNN algorithms for fruits recognition and image segmentation based on atrous convolution schemes, in order to avoid the loss of information at downsampling phases, also test instance segmentation to deal with recognition problems presented in occlusion and clustering situations [57];
- Create a larger training data-set for training a Deep Learning algorithm for semantic and instance segmentation.
- Examine other methods for fruits recognition and image segmentation based on machine learning and deep learning techniques for dealing with clustering and occlusion scenarios such GANs [58].
- Explore other localization methods, like 3D reconstruction techniques (e.g., disparity maps and position triangulation) and different vision sensors (e.g., stereo camera, RGB-D sensor, Kinect).
- Analyze other vision-based control approaches, combining different camera models and mounting configurations, that demonstrate more robustness to parametric uncertainties and kinematic singularities.
- Improve the performance of the control scheme by adding an adaptive control algorithm to estimate the gain values of the Sliding Mode Controller.
- Examine the feasibility of the proposed Hybrid-ST-SM HVS control scheme carrying out experimental tests.
- Analyze how the proposed controller deals with uncertainties at robot encoders q which is non-parametric.
- Propose a dynamic cascade control scheme for performing the task much faster.

Bibliography

[1] DUROCHER, K.. **Should We Fear the Rise of Farm Robots? These Experts Say No**. In: FUTURE OF FOOD A PROJECT OF RED CUP AGENCY, November 2017.

[2] BUTLER, S.. **If EU workers go, will robots step in to pick and pack Britain's dinners?** In: THE GUARDIAN, Feb 2017.

[3] KAPACH, K.; BARNEA, E.; MAIRON, R. ; EDAN, Y.. **Computer Vision for Fruit Harvesting Robots - State of the Art and Challenges Ahead**. International Journal Computational Vision and Robotics, 3(1/2):4–34, 2012.

[4] MEHTA, S. S.; MACKUNIS, W. ; BURKS, T. F.. **Robust Visual Servo Control in the Presence of Fruit Motion for Robotic Citrus Harvesting**. Computers and Electronics in Agriculture, 123:362–375, 2016.

[5] MANN, M. P.; ZION, B.; SHMULEVICH, I.; RUBINSTEIN, D. ; LINKER, R.. **Combinatorial Optimization and Performance Analysis of a Multi-arm Cartesian Robotic Fruit Harvester–Extensions of Graph Coloring**. Journal of Intelligent & Robotic Systems, 82(3-4):399–411, Jun 2016.

[7] EDAN, Y.; HAN, S. ; KONDO, N.. **Automation in Agriculture**. In: Nof, S. Y., editor, SPRINGER HANDBOOK OF AUTOMATION, p. 1095–1128. Springer-Verlag Berlin Heidelberg, 2009.

[8] BAC, C. W.; VAN HENTEN, E. J.; HEMMING, J. ; EDAN, Y.. **Harvesting Robots for High-value Crops: State-of-the-art Review and Challenges Ahead**. Journal of Field Robotics, 31(6):888–911, 2014.

[9] ZAIDNER, G.; SHAPIRO, A.. **A Novel Data Fusion Algorithm for Low-cost Localisation and Navigation of Autonomous Vineyard Sprayer Robots**. Biosystems Engineering, 146:133–148, 2016. Special Issue: Advances in Robotic Agriculture for Crops.

[10] EIZICOVITS, D.; VAN TUIJL, B.; BERMAN, S. ; EDAN, Y.. **Integration of Perception Capabilities in Gripper Design using Graspability**

Maps. Biosystems Engineering, 146:98–113, 2016. Special Issue: Advances in Robotic Agriculture for Crops.

[11] GONGAL, A.; AMATYA, S.; KARKEE, M.; ZHANG, Q. ; LEWIS, K.. **Sensors and Systems for Fruit Detection and Localization: A Review**. Computers and Electronics in Agriculture, 116:8–19, 2015.

[12] CHAUMETTE, F.; HUTCHINSON, S.. **Visual Servo Control - Part I: Basic approaches**. IEEE Robotics Automation Magazine, 13(4):82–90, Dec 2006.

[13] CORKE, P. I.. **Robotics, vision and control : fundamental algorithms in MATLAB / Peter Corke** . Springer Berlin, 2011.

[14] CHAUMETTE, F.; HUTCHINSON, S.. **Visual Servo Control - Part II: Advanced approaches**. IEEE Robotics and Automation Magazine, 14(1):109–118, 2006.

[15] PLEBE, A.; GRASSO, G.. **Localization of Spherical Fruits for Robotic Harvesting**. Machine Vision and Applications, 13(2):70–79, Nov 2001.

[16] BILLINGSLEY, J.; VISALA, A. ; DUNN, M.. **Robotics in Agriculture and Forestry**. In: Siciliano, B.; Khatib, O., editors, SPRINGER HANDBOOK OF ROBOTICS, p. 1065–1077. Springer Berlin Heidelberg, 2008.

[17] AMATYA, S.; KARKEE, M.; GONGAL, A.; ZHANG, Q. ; WHITING, M. D.. **Detection of Cherry Tree Branches with Full Foliage in Planar Architecture for Automated Sweet-cherry Harvesting**. Biosystems Engineering, 146:3–15, 2016. Special Issue: Advances in Robotic Agriculture for Crops.

[18] WEI, X.; JIA, K.; LAN, J.; LI, Y.; ZENG, Y. ; WANG, C.. **Automatic Method of Fruit Object Extraction under Complex Agricultural Background for Vision System of Fruit Picking Robot**. Optik - International Journal for Light and Electron Optics, 125(19):5684–5689, 2014.

[19] BAC, C. W.; ROORDA, T.; RESHEF, R.; BERMAN, S.; HEMMING, J. ; VAN HENTEN, E. J.. **Analysis of a Motion Planning Problem for Sweet-pepper Harvesting in a Dense Obstacle Environment**. Biosystems Engineering, 146:85–97, 2016.

[20] LOTTES, P.; HÖRFERLIN, M.; SANDER, S. ; STACHNISS, C.. **Effective Vision-based Classification for Separating Sugar Beets and Weeds for Precision Farming**. Journal of Field Robotics, 34(6):1160–1178, 9 2016.

[21] BAH, M. D.; HAFIANE, A. ; CANALS, R.. **Deep learning with unsupervised data labeling for weeds detection on UAV images**. CoRR, abs/1805.12395, 2018.

[22] BOTTERILL, T.; PAULIN, S.; GREEN, R.; WILLIAMS, S.; LIN, J.; SAXTON, V.; MILLS, S.; CHEN, X. ; CORBETT-DAVIES, S.. **A Robot System for Pruning Grape Vines**. Journal of Field Robotics, 34(6):1100–1122, 10 2016.

[23] LONG, J.; SHELHAMER, E. ; DARRELL, T.. **Fully Convolutional Networks for Semantic Segmentation**. In: THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), June 2015.

[24] HUNG, C.; NIETO, J.; TAYLOR, Z.; UNDERWOOD, J. ; SUKKARIEH, S.. **Orchard fruit segmentation using multi-spectral feature learning**. In: 2013 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, p. 5314–5320, Nov 2013.

[25] BARGOTI, S.; UNDERWOOD, J. P.. **Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards**. CoRR, abs/1610.08120, 2016.

[26] DIAS, P. A.; TABB, A. ; MEDEIROS, H.. **Multispecies Fruit Flower Detection Using a Refined Semantic Segmentation Network**. IEEE Robotics and Automation Letters, 3(4):3003–3010, Oct 2018.

[27] BARTH, R.; HEMMING, J. ; VAN HENTEN, E. J.. **Design of an Eye-in-hand Sensing and Servo Control Framework for Harvesting Robotics in Dense Vegetation**. Biosystems Engineering, 146:71–84, 2016.

[28] MEHTA, S. S.; BURKS, T. F.. **Vision-based Control of Robotic Manipulator for Citrus Harvesting**. Computers and Electronics in Agriculture, 102:146–158, 2014.

[29] NGUYEN, T. T.; VANDEVOORDE, K.; WOUTERS, N.; KAYACAN, E.; BAERDEMAEKER, J. G. D. ; SAEYS, W.. **Detection of Red and**

Bicoloured Apples on Tree with an RGB-D Camera. Biosystems Engineering, 146:33–44, 2016.

[30] SENTHILNATH, J.; DOKANIA, A.; KANDUKURI, M.; K.N., R.; ANAND, G. ; OMKAR, S.. **Detection of Tomatoes using Spectral-spatial Methods in Remotely Sensed RGB Images Captured by UAV**. Biosystems Engineering, 146:16–32, 2016. Special Issue: Advances in Robotic Agriculture for Crops.

[31] KURIOS, Q.; ALDAYR, A. ; SAMAHERNI, D.. **Design and stability analysis of a variable structure adaptive backstepping controller**. Asian Journal of Control, 14(3):641–651, 2011.

[32] CHEAH, C. C.; HOU, S. P.; ZHAO, Y. ; SLOTINE, J. J. E.. **Adaptive Vision and Force Tracking Control for Robots With Constraint Uncertainty**. IEEE/ASME Transactions on Mechatronics, 15(3):389–399, June 2010.

[33] ROUX, O. T.; CANDEA, L. A.; JACOUD, P. A. ; LIU, H.. **Overcoming Limitations of Uncalibrated Robotics Visual Servoing by means of Sliding Mode Control and Switching Monitoring Scheme**. Asian Journal of Control, 16(3):752–764, 2014.

[34] LEITE, A. C.; LIZARRALDE, F.. **Passivity-based Adaptive 3D Visual Servoing without Depth and Image Velocity Measurements for Uncertain Robot Manipulators**. International Journal of Adaptive Control and Signal Processing, 30(8-10):1269–1297, 2016.

[35] MEHTA, S.; MACKUNIS, W. ; BURKS, T.. **Nonlinear Robust Visual Servo Control for Robotic Citrus Harvesting**. IFAC Proceedings Volumes, 47(3):8110 – 8115, 2014. 19th IFAC World Congress.

[36] GAMBA, J.; FROM, P. J. ; LEITE., A. C.. **A Visual Servoing Approach For Robotic Fruit Harvesting in the Presence of Parametric Uncertainties**. CBA Proceedings Volumes, 2018. XXII Congresso Brasileiro de Automática.

[37] SICILIANO, B.; SCIAVICCO, L.; VILLANI, L. ; ORIOLO, G.. **Robotics: Modelling, Planning and Control**. Springer Publishing Company, Inc., 2009.

[38] NAKAMURA, Y.; HANAFUSA, H.. **Inverse Kinematic Solution with Singularity Robustness for Robot Manipulator Control**. ASME

Journal of Dynamic Systems, Measurement, and Control, 108(3):163–171, 1986.

[39] VARGAS, L. V.; LEITE, A. C. ; COSTA, R. R.. **Overcoming Kinematic Singularities with the Filtered Inverse Approach**. In: PROCEEDINGS OF THE 19TH IFAC WORLD CONGRESS, p. 8496–8502, Cape Town, South Africa, Aug 2014.

[41] KHALIL, H. K.. **Nonlinear Systems**. Prentice-Hall Inc., 2002.

[42] YU, X.; MAN, Z.. **Variable Structure Systems with Terminal Sliding Modes**, p. 109–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[43] LEITE, A. C.; LIZARRALDE, F. ; HSU, L.. **Hybrid Adaptive Vision-Force Control for Robot Manipulators Interacting with Unknown Surfaces**. The International Journal of Robotics Research, 28(7):911–926, 2009.

[44] FORSYTH, D. A.; PONCE, J.. **Computer Vision - A Modern Approach**. Pearson Inc., 2nd edition, 2012.

[45] BAY, H.; ESS, A.; TUYTELAARS, T. ; VAN GOOL, L.. **Speeded-Up Robust Features (SURF)**. Computer Vision and Image Understanding, 110(3):346–359, June 2008.

[46] OLIVEIRA, T. R.; LEITE, A. C.; PEIXOTO, A. J. ; HSU, L.. **Overcoming Limitations of Uncalibrated Robotics Visual Servoing by means of Sliding Mode Control and Switching Monitoring Scheme**. Asian Journal of Control, 16(3):752–764, 2014.

[47] PAPACHRISTODOULOU, A.; PRAJNA, S.. **Robust Stability Analysis of Nonlinear Hybrid Systems**. IEEE Transactions on Automatic Control, 54(5):1035–1041, May 2009.

[48] YAN, L.; HSU, L.; COSTA, R. R. ; LIZARRALDE, F.. **A Variable Structure Model Reference Robust Control without a Prior Knowledge of High Frequency Gain Sign**. Automatica, 44(4):1036–1044, 2008.

[49] PERRUQUETTI, W.. **Sliding Mode Control in Engineering**. Marcel Dekker, Inc., New York, NY, USA, 2002.

[50] LECUN, Y.; BENGIO, Y. ; HINTON, G.. **Deep learning**. Nature, 521(7553):436–444, 5 2015.

[51] CHANG, S.-F.. **Segmentation Using Superpixels: A Bipartite Graph Partitioning Approach**. In: PROCEEDINGS OF THE 2012 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), CVPR '12, p. 789–796, Washington, DC, USA, 2012. IEEE Computer Society.

[52] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[53] KINGMA, D. P.; BA, J.. **Adam: A Method for Stochastic Optimization**. CoRR, abs/1412.6980, 2014.

[54] RUBLEE, E.; RABAUD, V.; KONOLIGE, K. ; BRADSKI, G.. **Orb: An efficient alternative to sift or surf**. p. 2564–2571, Nov 2011.

[55] Neri, F.; Cotta, C. ; Moscato, P., editors. **Handbook of Memetic Algorithms**, volumen 379 de **Studies in Computational Intelligence**. Springer, 2012.

[56] QIAN, H.; MAO, Y.; GENG, J. ; WANG, Z.. **Object tracking with self-updating tracking window**. In: Yang, C. C.; Zeng, D.; Chau, M.; Chang, K.; Yang, Q.; Cheng, X.; Wang, J.; Wang, F.-Y. ; Chen, H., editors, INTELLIGENCE AND SECURITY INFORMATICS, p. 82–93, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[57] CHEN, L.; PAPANDREOU, G.; SCHROFF, F. ; ADAM, H.. **Rethinking Atrous Convolution for Semantic Image Segmentation**. CoRR, abs/1706.05587, 2017.

[58] GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A. ; BENGIO, Y.. **Generative Adversarial Nets**. In: Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D. ; Weinberger, K. Q., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 27, p. 2672–2680. Curran Associates, Inc., 2014.

[60] SLOTINE, J.-J. E.; LI, W.. **Applied nonlinear control**. Pearson, Upper Saddle River, NJ, 1991. The book can be consulted by contacting: BE-ABP-CC3: Pfingstner, Juergen.

[61] BADRINARAYANAN, V.; KENDALL, A. ; CIPOLLA, R.. **Segnet: A deep convolutional encoder-decoder architecture for image segmentation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12):2481–2495, Dec 2017.

[62] DIMOVSKI, I.; TROMPESKA, M.; SAMAK, S.; DUKOVSKI, V. ; CVETKOSKA, D.. **Algorithmic approach to geometric solution of generalized paden–kahan subproblem and its extension**. International Journal of Advanced Robotic Systems, 15(1):1729881418755157, 2018.

[63] SWIKIR, A.; UTKIN, V.. **Chattering analysis of conventional and super twisting sliding mode control algorithm**. In: 2016 14TH INTERNATIONAL WORKSHOP ON VARIABLE STRUCTURE SYSTEMS (VSS), p. 98–102, June 2016.

[64] VASTHI, P. I.; KUSUMANINGRUM, R.. **Object segmentation for fruit images using ohta colour space and cascade threshold**. In: 2015 INTERNATIONAL CONFERENCE ON SCIENCE IN INFORMATION TECHNOLOGY (ICSITECH), p. 321–325, Oct 2015.

[65] HARRIS, C.; STEPHENS, M.. **A combined corner and edge detector**. In: IN PROC. OF FOURTH ALVEY VISION CONFERENCE, p. 147–151, 1988.

[66] LOWE, D. G.. **Distinctive image features from scale-invariant keypoints**. Int. J. Comput. Vision, 60(2):91–110, Nov. 2004.

[67] OTSU, N.. **A Threshold Selection Method from Gray-level Histograms**. IEEE Transactions on Systems, Man and Cybernetics, 9(1):62–66, 1979.

[68] MURRAY, R. M.; SASTRY, S. S. ; ZEXIANG, L.. **A Mathematical Introduction to Robotic Manipulation**. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.

[69] WANG, Z.; BOVIK, A. C.; SHEIKH, H. R. ; SIMONCELLI, E. P.. **Image quality assessment: From error visibility to structural similarity**. IEEE TRANSACTIONS ON IMAGE PROCESSING, 13(4):600–612, 2004.

[70] PENG, C.; ZHANG, X.; YU, G.; LUO, G. ; SUN, J.. **Large kernel matters - improve semantic segmentation by global convolutional network**. CoRR, abs/1703.02719, 2017.

A
ROS Script Files

Listing A.1: ROS System Launch File

```
<launch>

    <include
    file="$(find␣zed_cpu_ros)/launch/zed_cpu_ros.launch"/>

    <node pkg="mitsubishi_robot"
          name="mitsubishi_robot"
          type="mit.py">
    </node>

    <node pkg="image_segmentation"
          name="image_segmentation"
          type="seg.py">
    </node>

    <node pkg="homography_generator"
          name="homography_generator"
          type="trian.py">
    </node>

</launch>
```

Listing A.2: ROS Camera Launch File

```
<launch>
<arg name="config_file_location"
                default="$(find␣zed_cpu_ros)/config
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/SN10026505.conf"/>
<arg name="camera_namespace" default="camera"/>

<node pkg="zed_cpu_ros" type="zed_cpu_ros"
          name="zed_cpu_ros_node" output="screen"
          ns="$(arg␣camera_namespace)" required="true">
        <param name="resolution" value="3"/>
        <param name="frame_rate" value="100"/>
        <param name="config_file_location"
               value="$(arg␣config_file_location)"/>
        <param name="show_image" value="false"/>
```

```
        <param name="left_frame_id" value="left_frame"/>
        <param name="right_frame_id" value="right_frame"/>
        <param name="load_zed_config" value="true"/>
</node>

<node pkg="tf" type="static_transform_publisher"
name="static_tf_1" args="0.25 0 0.4 0 0 0 1
base_link left_frame 30"/>

</launch>
```

A.1 Python Scripts

Listing A.3: Segmentation Script File

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2, math,copy, rospy
from img_pros.msg import point_array,pixel_params
from sensor_msgs.msg import Image, CameraInfo, JointState
from geometry_msgs.msg import Point
from cv_bridge import CvBridge, CvBridgeError
import numpy as np
import imutils
import time
from std_msgs.msg import Bool

bridge = CvBridge()
#img_params = ImageParams()
img_params = point_array()
pixl = pixel_params()
pixr = pixel_params()
chng_flag = Bool()
AREA_FLAG = 10


def callback_iml (msg):
 global bridge
 global il

 try:
  # Convert your ROS Image message to OpenCV2
  il = bridge.imgmsg_to_cv2(msg, "bgr8")
 except CvBridgeError, e:
  print(e)

def callback_imr (msg):
 global bridge
 global ir

 try:
  # Convert your ROS Image message to OpenCV2
  ir = bridge.imgmsg_to_cv2(msg, "bgr8")
 except CvBridgeError, e:
  print(e)

def callbackhom(data):
```

```
 global H

 H = np.array([[data.X[0],data.X[1],data.X[2]],
        [data.X[3],data.X[4],data.X[5]],
        [data.X[6],data.X[7],data.X[8]]])

def callbackcaml(data):
 global projMatrl
 global Dl
 global Kl

 projMatrl = data.P
 Dl = np.array([data.D[0],data.D[1],data.D[2],data.D[3],
 data.D[4]])*0.1
 Kl = np.array([[data.K[0],data.K[1],data.K[2]],
        [data.K[3],data.K[4],data.K[5]],
        [data.K[6],data.K[7],data.K[8]]])

def callbackcamr(data):
 global projMatrr
 global Dr
 global Kr

 projMatrr = data.P
 Dr = np.array([data.D[0],data.D[1],data.D[2],data.D[3],
 data.D[4]])*0.1
 Kr = np.array([[data.K[0],data.K[1],data.K[2]],
        [data.K[3],data.K[4],data.K[5]],
        [data.K[6],data.K[7],data.K[8]]])

def callbackpixel(data):
 global pixel_cmdl
 pixel_cmdl = np.array([data.x,data.y,data.z],dtype=np.int)

def pross(imagerw,K,D,cx,cy,r):
 h,  w = imagerw.shape[:2]
 newcameramtx, roi=
 cv2.getOptimalNewCameraMatrix(K,D,(w,h),1,(w,h))

 image = cv2.undistort(imagerw, K, D, None, newcameramtx)

 mask = 255*np.ones((376,672),np.uint8)
 if cx!=0 and cy!=0:
  mask = 0*mask
  mask = cv2.circle(mask, (cx,cy), r, (255,255,255), -1)

 image[:,:,0][mask == 0] = 0
```

```
 image[:,:,1][mask == 0] = 0
 image[:,:,2][mask == 0] = 0

 graym,closing = seg(image)

 # get objects
 im2,contours,hierarchy = cv2.findContours(closing, 1, 2)
 #sort by area
 contours = sorted(contours, key=cv2.contourArea, reverse=True)

 points = np.empty((0, 7))
 for contour in contours:
  area = cv2.contourArea(contour)
  if area > AREA_FLAG:
   M = cv2.moments(contour)
   cX = int(M["m10"] / M["m00"])
   cY = int(M["m01"] / M["m00"])
   x,y,w,h = cv2.boundingRect(contour)
   points = np.append(points,[np.array([cX,cY,x,y,h,w,area])],
    axis=0)


 return image,closing,points,mask

def cos_sim(a, b):
 """Takes 2 vectors a, b and returns the similarity according
 to the definition of the dot product
 """
 c = np.subtract(a, b)
 return np.linalg.norm(c)

def geo(a):
 b = np.array(a)
 return b.prod()**(1.0/len(b))

def seg(image):
 global segp1t
 global segp2t

 # divide image
 b,g,r = cv2.split(image)
 #gray image
 gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

 #Ohta color espace
 I2 = cv2.subtract(r,b)/2
 I2l = cv2.subtract(r,b)
```

```
I1l = cv2.subtract(r,g)

#noise removing
#I2[I2 < 25] = 0
#I2l[I2l < 50] = 0

#Adaptive Threshold Otsu
I1l = cv2.GaussianBlur(I1l,(5,5),0)
segp1,th1tmp = cv2.threshold(I1l,50,255,cv2.THRESH_BINARY+
cv2.THRESH_OTSU)

I2l = cv2.GaussianBlur(I2l,(5,5),0)
segp2,th2tmp = cv2.threshold(I2l,50,255,cv2.THRESH_BINARY+
cv2.THRESH_OTSU)

segp1t.pop(0)
segp1t.append(segp1)

segp2t.pop(0)
segp2t.append(segp2)

nth1= np.maximum(35,geo(segp1t))
nth2= np.maximum(35,geo(segp2t))

_,th1 = cv2.threshold(I1l,nth1,255,cv2.THRESH_BINARY)
_,th2 = cv2.threshold(I2l,nth2,255,cv2.THRESH_BINARY)

th2[th2 > th1] = 0

_,contours,_ = cv2.findContours(th2, 1, 2)

if len(contours)== 1:
 th2tmp[th2tmp > th1] = 0
 th2 = th2tmp.copy()


#Improve images
kernel = np.ones((5,5),np.uint8)
# closing = grayl.copy()

for i in range (1,5):
 th2 = cv2.morphologyEx(th2, cv2.MORPH_CLOSE, kernel) #abetura
 th2 = cv2.morphologyEx(th2, cv2.MORPH_OPEN, kernel) #fechamento

#image mask
gray[th2 != 255] = 0
```

```
 return gray,th2

def main():

 rospy.init_node('segdeep', anonymous=True) #node name
 rospy.Subscriber('/camera/left/image_raw', Image,
 callback_iml)
 rospy.Subscriber('/camera/right/image_raw', Image, callback_imr)

 rospy.Subscriber("/camera/left/camera_info",CameraInfo,
 callbackcaml)
 rospy.Subscriber("/camera/right/camera_info",CameraInfo,
 callbackcamr)

 rospy.Subscriber("/camera/cmd/pixel",Point,callbackpixel)

 rospy.Subscriber("/camera/homography",point_array,callbackhom)
 chng_flag_pub = rospy.Publisher('/camera/cmd/homography', Bool,
 queue_size=10)

 img_params_pub = rospy.Publisher('/camera/image_params',
 point_array, queue_size=10)
 img_pixels_publ = rospy.Publisher('/camera/left/pixels',
  pixel_params, queue_size=10)
 img_pixels_pubr = rospy.Publisher('/camera/right/pixels',
 pixel_params, queue_size=10)

 #pixel_cmdl = np.zeros((3,1)).copy()
 rate = rospy.Rate(30)
 rospy.sleep(1)


 #camera Matrices
 Pl = np.array([[projMatrl[0], projMatrl[1], projMatrl[2],
  projMatrl[3]],
   [projMatrl[4], projMatrl[5], projMatrl[6], projMatrl[7]],
   [projMatrl[8], projMatrl[9], projMatrl[10], projMatrl[11]]])
 Pr = np.array([[projMatrr[0], projMatrr[1], projMatrr[2],
 projMatrr[3]],
     [projMatrr[4], projMatrr[5], projMatrr[6], projMatrr[7]],
     [projMatrr[8], projMatrr[9], projMatrr[10], projMatrr[11]]])

 val_a = 10000

 while not rospy.is_shutdown():
  try:
   try:
```

```
        prr =np.array([pxr[:,0],pxr[:,1]],dtype=np.int)
        prl =np.array([pxl[:,0],pxl[:,1]],dtype=np.int)
      except:
        prr =[]
        prl =[]

      if pixel_cmdl[0] != 0 and pixel_cmdl[1] != 0:
       #get right point left camera
       px_cmd_candl = np.array([pixel_cmdl[0],pixel_cmdl[1]])
       prl=np.transpose(prl)
       al=val_a
       pcl = np.array([0,0,0],dtype=np.int)
       for px_l in prl:
        c =cos_sim(px_cmd_candl,px_l)
        if c < al:
         pcl = px_l
         al =c
       print(al,'left')
       print(pcl,'left')
       print(pixel_cmdl,'left')

       if al < val_a:
        px_cmd_candr = np.dot(np.linalg.inv(H), [pcl[0], pcl[1],
        pcl[1]/pcl[1]])
        px_cmd_candr = np.array([px_cmd_candr[0]/px_cmd_candr[2],
        px_cmd_candr[1]/px_cmd_candr[2]],dtype=np.int)

        #get right point right camera
        prr=np.transpose(prr)
        a=10*(al+4)
        pcr = np.array([0,0,0],dtype=np.int)
        for px_r in prr:
         c =cos_sim(px_cmd_candr,px_r)
         if c < a:
          pcr = px_r
          a =c
        print(a,'right')
        print(pcr,'right')

        if pcr[0] == 0 and pcr[1] == 0:
         chng_flag.data = True
        else:
         chng_flag.data = False

        chng_flag_pub.publish(chng_flag)
```

```
    else:
     pcr = np.array([0,0,0])
     pcl = np.array([0,0,0])

   else:
    pcr = np.array([0,0,0])
    pcl = np.array([0,0,0])

   iml, closingl, pxl, ml = pross(il, Kl, Dl, pcl[0], pcl[1],
   pixel_cmdl[2])
   imr, closingr, pxr, mr = pross(ir, Kr, Dr, pcr[0], pcr[1],
   np.int(pixel_cmdl[2]*1.1))
  except:
   iml, closingl, pxl, ml = pross(il, Kl, Dl,0,0,0)
   imr, closingr, pxr, mr = pross(ir, Kr,
    Dr,0,0,0)

  #more than one point, need to be organized
  if (len(pxl[:,0])>1) and (len(pxr[:,0])>1):
   px_candr = np.dot(np.linalg.inv(H), [pxl[:,0],
   pxl[:,1], pxl[:,1]/pxl[:,1]])
   px_candr = np.array([px_candr[0,:]/px_candr[2,:],
   px_candr[1,:]/px_candr[2,:]], dtype=np.int)
   px_candr=np.transpose(px_candr)

   #Organize points
   pr =np.array([pxr[:,0], pxr[:,1]])
   pr=np.transpose(pr)
   pr_ord = np.empty((0, 2))
   for px_c in px_candr:
    a=500
    #pc = [];
    for px_r in pr:
     c =cos_sim(px_c,px_r)
     if c < a:
      pc =
      px_r
      a =c
    pr_ord = np.append(pr_ord,[np.array([pc[0],pc[1]])], axis=0)

   pr_ord = np.transpose(pr_ord)
   pl =np.array([pxl[:,0], pxl[:,1]])

   #Triangulation with Homography points.
   X = cv2.triangulatePoints(Pl,Pr,pl,pr_ord)

   img_params.X = X[0,:]
```

```
 img_params.Y = X[1,:]
 img_params.Z = X[2,:]

 img_params_pub.publish(img_params)
elif (len(pxl[:,0])==0) or (len(pxr[:,0])==0):
 img_params_pub.publish()
else:
 ppl =np.array([pxl[0,0],pxl[0,1]])
 ppr =np.array([pxr[0,0],pxr[0,1]])

 X = cv2.triangulatePoints(Pl,Pr,ppl,ppr)

 X = np.array([X[0,:]/X[3,:],X[1,:]/X[3,:],X[2,:]/X[3,:]]);

 img_params.X = X[0,:]
 img_params.Y = X[1,:]
 img_params.Z = X[2,:]

 img_params_pub.publish(img_params)


pixl.cx = pxl[:,0];
pixl.cy = pxl[:,1];
pixl.x = pxl[:,2];
pixl.y = pxl[:,3];
pixl.h = pxl[:,4];
pixl.w = pxl[:,5];
pixl.area = pxl[:,6];

pixr.cx = pxr[:,0];
pixr.cy = pxr[:,1];
pixr.x = pxr[:,2];
pixr.y = pxr[:,3];
pixr.h = pxr[:,4];
pixr.w = pxr[:,5];
pixr.area = pxr[:,6];

img_pixels_publ.publish(pixl)
img_pixels_pubr.publish(pixr)

cv2.imshow('closingl',closingl)
cv2.imshow('closingr',closingr)

cv2.imshow('iml',iml)
cv2.imshow('imr',imr)

cv2.waitKey(3)
```

```
    rate.sleep()

if __name__ == '__main__':
 main()
```

Listing A.4: Homography Script File

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from pyimagesearch.panorama import Stitcher
import cv2, math,copy, rospy

from img_pros.msg import point_array,pixel_params
from sensor_msgs.msg import Image, CameraInfo
from cv_bridge import CvBridge, CvBridgeError
from std_msgs.msg import Bool
import numpy as np
import imutils


bridge = CvBridge()
img_homo = point_array()
stitcher = Stitcher()
iml_last = Image()

def callbackbool (msg):
 global chng_flag

 chng_flag = msg.data
 print(chng_flag)

def callback_iml (msg):
 global bridge
 global il

 try:
  # Convert your ROS Image message to OpenCV2
  il = bridge.imgmsg_to_cv2(msg, "bgr8")
 except CvBridgeError, e:
  print(e)

def callback_imr (msg):
 global bridge
 global ir

 try:
```

```
    # Convert your ROS Image message to OpenCV2
    ir = bridge.imgmsg_to_cv2(msg, "bgr8")
  except CvBridgeError, e:
    print(e)

def callbackcaml(data):
  global projMatrl
  global Dl
  global Kl

  projMatrl = data.P
  Dl = np.array([data.D[0],data.D[1],data.D[2],
  data.D[3],data.D[4]])*0.1
  Kl = np.array([[data.K[0],data.K[1],data.K[2]],
        [data.K[3],data.K[4],data.K[5]],
        [data.K[6],data.K[7],data.K[8]]])

def callbackcamr(data):
  global projMatrr
  global Dr
  global Kr

  projMatrr = data.P
  Dr = np.array([data.D[0],data.D[1],data.D[2],
  data.D[3],data.D[4]])*0.1
  Kr = np.array([[data.K[0],data.K[1],data.K[2]],
        [data.K[3],data.K[4],data.K[5]],
        [data.K[6],data.K[7],data.K[8]]])



def undistort(imagerw,K, D):
  h,  w = imagerw.shape[:2]
  newcameramtx, roi=
  cv2.getOptimalNewCameraMatrix(K,D,(w,h),1,(w,h))

  image = cv2.undistort(imagerw, K, D, None, newcameramtx)

  return imutils.resize(image, width=672)

def mse(imageA, imageB):
  # the 'Mean Squared Error' between the two images is the
  # sum of the squared difference between the two images;
  # NOTE: the two images must have the same dimension
  err = np.sum((imageA.astype("float") -
  imageB.astype("float")) ** 2)
  err /= float(imageA.shape[0] * imageA.shape[1])
```

```
    # return the MSE, the lower the error, the more "similar"
    # the two images are
    return err

def main():

    rospy.init_node('segdeep', anonymous=True) #node name
    rospy.Subscriber('/camera/left/image_raw', Image, callback_iml)
    rospy.Subscriber('/camera/right/image_raw', Image, callback_imr)

    rospy.Subscriber("/camera/left/camera_info",CameraInfo,
    callbackcaml)
    rospy.Subscriber("/camera/right/camera_info",CameraInfo,
    callbackcamr)

    img_homo_pub = rospy.Publisher('/camera/homography', point_array,
     queue_size=10)
    rospy.Subscriber("/camera/cmd/homography",Bool,callbackbool)

    iml_last = 255*np.ones((376,672,3),np.uint8)

    rate = rospy.Rate(30)
    rospy.sleep(1)
    while not rospy.is_shutdown():

     iml = undistort(il, Kl, Dl)
     imr = undistort(ir, Kr, Dr)

     #Homography is calculated every movement
     s = mse(il, iml_last)
     try:
      if s > 700 or chng_flag:
       zed,H = stitcher.stitch([iml, imr], Movement=True)
       iml_last = il.copy()
       print('error',s)
      else:
       zed,H = stitcher.stitch([iml, imr], Movement=False)

      img_homo.X = np.array([H[0,0],H[0,1],H[0,2],H[1,0],
      H[1,1],H[1,2],H[2,0],H[2,1],H[2,2]])
      img_homo_pub.publish(img_homo)

      cv2.imshow('zed',zed)
      cv2.waitKey(3)
     except:
      print('NO Homography')
```

```
  rate.sleep()
if __name__ == '__main__':
 main()
```

Listing A.5: Mitsubishi Script File

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import rospy, serial, atexit
import numpy as np
from sensor_msgs.msg import JointState
from geometry_msgs.msg import Point, Twist
from std_msgs.msg import Header

global ser
global jp
global jp_last
global Xbe
global Xbe_last

jp = [np.float(0), np.float(0),np.float(120),
np.float(-15),np.float(0)]
jp_last = [np.float(0), np.float(-60),
np.float(120),np.float(0),np.float(0)]
Xbe = Twist()

Xbe_last = Xbe

def callback_JointState (msg):
 global jp

 jp = msg.position

def callback_pose (msg):
 global Xbe

 Xbe = msg


def exit_handler():
    print 'My application is ending!'

def config_com(port='/dev/ttyUSB0'):
 ser = serial.Serial()
 ser.port = port
 ser.baudrate = 9600
```

```
    ser.stopbits = serial.STOPBITS_TWO
    ser.parity = serial.PARITY_EVEN
    ser.timeout = 0.2
    return ser


def open_cont(ser):
    ser.write('1;1;NEW')
    ser.write('\r'.encode())

    ser.write('1;1;LOAD=1')
    ser.write('\r'.encode())

    ser.write('1;1;PRTVERLISTL')
    ser.write('\r'.encode())

    ser.write('1;1;PRTVEREMDAT')
    ser.write('\r'.encode())

    ser.write('1;1;CNTLON')
    ser.write('\r'.encode())

    ser.write('1;1;SRVON')
    ser.write('\r'.encode())

    ser.write('1;1;OVRD=40')
    ser.write('\r'.encode())

    ser.reset_input_buffer()
    ser.reset_output_buffer()
    rospy.sleep(1)


def close_com(ser):
    print('closing com')
    ser.write('1;1;SRVOFF')
    ser.write('\r'.encode())

    ser.write('1;1;CNTLOFF')
    ser.write('\r'.encode())
    ser.close()

def joint_pos_cmd(jp,joint_states,joint_states_pub,ser):
    global jp_last

    if len(jp) == 5:
     ser.reset_output_buffer()
```

```
    pos_cmd = '1;9;EXECJCOSIROP␣=␣('+'%.3f' % jp[0]+',␣'+
    '%.3f' % jp[1]+',␣'+'%.3f' % jp[2]+',␣0.000,␣'+
    '%.3f' % jp[3]+',␣'+'%.3f' % jp[4]+')'
    #print(pos_cmd)
    print('writing',pos_cmd)
    ser.write(pos_cmd)
    ser.write('\r'.encode())

    rospy.sleep(0.1)
    ser.write('1;9;EXECMOV␣JCOSIROP')
    ser.write('\r'.encode())
    ser.write('\r'.encode())

    joint_states.position = [jp[0], jp[1],jp[2],jp[3],jp[4]]

    joint_states.header.stamp = rospy.Time.now()
    joint_states_pub.publish(joint_states)
   else:
    print('wrong␣cmd␣dimension', len(jp))

   jp_last = jp

def pose_cmd(Xbe,ser):
   global Xbe_last

   ser.reset_output_buffer()
   Xbe.linear.x
   pose_cmd = '1;9;EMDATP1=('+'%.2f' % Xbe.linear.x+',
␣'+'%.2f' % Xbe.linear.y+',␣'+'%.2f' % Xbe.linear.z+',
␣'+'%.2f' % Xbe.angular.x+',␣'+'%.2f' % Xbe.angular.y+
   ',␣0.00)(6,0)'
   print('writting',pose_cmd)

   ser.write(pose_cmd)
   ser.write('\r'.encode())

   rospy.sleep(0.8)
   ser.write('1;9;EXECMOV␣P1')
   ser.write('\r'.encode())

   Xbe_last = Xbe

def joint_read_pub(joint_states_pub,joint_states,ser):
   ser.write('\r'.encode())
   ser.write('1;1;JPOSF')
   ser.write('\r'.encode())
   #print('reading')
```

```
 data = ''
 d = 'a'
 while data.count(';;') < 1 and d != '': #16
  d = ser.read(1)
  if 'K' in d or 'K' in data:
   data = data+d
  elif 'R' in data:
   print(data,"d",d)

 if 'K' in data:
  data = data.split(";")

  joint_states.position = [np.float(data[1]), np.float(data[3]),
  np.float(data[5]),np.float(data[9]),np.float(data[11])]

  joint_states.header.stamp = rospy.Time.now()
  joint_states_pub.publish(joint_states)

 #return joint_states.position

def pose_read_pub(pose_pub,pose_state,ser):
 ser.write('\r'.encode())
 ser.write('1;1;PPOSF')
 ser.write('\r'.encode())
 #print('reading')

 data = ''
 d = 'a'
 while data.count(';;') < 1 and d != '': #16
  d = ser.read(1)
  if 'K' in d or 'K' in data:
   data = data+d
  elif 'R' in data:
   print(data,"d",d)

 if 'K' in data:
  data = data.split(";")

  pose_state.linear.x = np.float(data[1])
  pose_state.linear.y = np.float(data[3])
  pose_state.linear.z = np.float(data[5])
  pose_state.angular.x = np.float(data[7])
  pose_state.angular.y = np.float(data[9])

  pose_pub.publish(pose_state)
```

```
 #return joint_states.position

def main():
 rospy.init_node('joint_states', anonymous=True) #node name

 joint_states_pub = rospy.Publisher('/mit/joint_states',
 JointState, queue_size=10)
 pose_pub = rospy.Publisher('/mit/pose', Twist, queue_size=10)

 joint_states = JointState()
 joint_states.name = ['joint0','joint1','joint2','joint3',
 'joint4']
 joint_states.header = Header()
 joint_states.effort = []

 pose_state = Twist()


 #open serial comunication
 ser = config_com('/dev/ttyUSB0')
 ser.open()
 open_cont(ser)

 joint_read_pub(joint_states_pub,joint_states,ser)

 rospy.Subscriber('/mit/cmd/joint_states',JointState,
 callback_JointState)
 rospy.Subscriber('/mit/cmd/pose',Twist, callback_pose)

 rate = rospy.Rate(5) #Hz

 while not rospy.is_shutdown():

  if jp != jp_last:
   joint_pos_cmd(jp,joint_states,joint_states_pub,ser)
  elif Xbe != Xbe_last:
   pose_cmd(Xbe,ser)
  else:
   joint_read_pub(joint_states_pub,joint_states,ser)
   pose_read_pub(pose_pub,pose_state,ser)

  rate.sleep()
  ser.reset_input_buffer()
  ser.reset_output_buffer()

if __name__ == '__main__':
 main()
```