Constraint-Aware Visual Servoing for Teaching Practical Robot Motion

by

Ambrose Chan

B.A.Sc., The University of Waterloo, 2006

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

The Faculty of Graduate Studies

(Mechanical Engineering)

The University Of British Columbia

(Vancouver)

April, 2009

Abstract

In this thesis, a *constraint-aware* visual servoing control law is proposed. The control law is designed for a robot manipulator with an uncalibrated camera mounted on its end-effector. This control law allows the robot to execute large, collision-free motions with closed-loop positional accuracy. A reference image visually describes the desired end-effector position with respect to a target object whose location is initially unknown. The control law uses this reference image with online feedback from the camera to direct the trajectory of robot towards the completion of the positioning task. The control law generates feasible and realistic robot trajectories that respect the robot's joint position and velocity limits, even in the presence of large control gains. The control law also explicitly keeps the target object within the camera's field of view to provide uninterrupted visual feedback. The control law avoids potential whole-arm collisions with workspace obstacles via planning and control strategies.

The visual servoing control law is implemented in a nonlinear model predictive control framework, using an estimated model of the eye-in-hand configuration and an estimated location of the target object. Two methods of approximating the object's location for joint-space path planning are demonstrated in simulations and experiments. The first uses homography estimation and decomposition on an un-modelled object. The second uses an extended Kalman filter with a prior object model to improve robustness against image noise and disturbances.

Two planning and control strategies are presented. The first strategy uses an offline *plan-then-servo* approach that integrates probabilistic roadmaps with visual servoing. A method to construct paths between two robot configurations that keep the target object within the camera's field of view is demonstrated, allowing feasible transitions from planned motion to visual servoing. A method to address pose uncertainty to ensure collision-free, closed-loop motion is statistically tested with multiple positioning tasks. The second strategy uses an online iterative *plan-and-servo* approach that dynamically updates its estimate of the collision-free space while visual servoing. Experiments using an uncalibrated eye-in-hand plat-

form demonstrate the ability of the visual servoing control law to achieve closed-loop positioning via collision-free trajectories, even when the object location is highly uncertain.

Table of Contents

Appendices

List of Tables

List of Figures

Nomenclature

$\mathbf{A}_{\text{control}}$ The system matrix of the open-loop plant in PBVS

$\widetilde{\mathbf{A}}$ The system matrix used in EKF pose estimation, representing the linearized state difference equations of the observer model

α Pitch; the angle of rotation about the y-axis, applied in the order z-y-x about current frames

b Shorthand for the robot base frame, $\mathcal{F}_b$, when used in a homogeneous transformation, for example, ${}^{m}\mathbf{T}_b$

$\mathbf{B}_{\text{control}}$ The input matrix of the open-loop plant in PBVS

β The perpendicular skew angle between imaging plane and optical axis

$\mathbf{C}$ The camera calibration matrix

c Shorthand for the camera frame, $\mathcal{F}_c$, when used in a homogeneous transformation, for example, ${}^{m}\mathbf{T}_c$

$\widetilde{\mathbf{C}}$ The output matrix used in EKF pose estimation, representing the linearized state output equations matrix of the observer model

$\mathbf{C_q}$ The robot inertial matrix

CAD Computer Aided Design

CCD Charge-Coupled Device

$\mathcal{C}$ The cost function to be minimized in MPC

CPU Central Processing Unit

CT Computed Tomography

d the signed perpendicular distance of the virtual plane π to the current camera frame; (*) ... in the desired camera frame

$\widetilde{d}(\mathbf{q})$ The shortest distance between the target object and the camera frustrum in configuration $\mathbf{q}$; used in the DVC algorithm

$d(\mathbf{q})$ The shortest distance between an obstacle and the robot in configuration $\mathbf{q}$; used in the DCC algorithm

$\mathbf{D}_{\text{adaptive}}$ An adaptive gains matrix used to address camera field-of-view constraints

$\mathbf{D}_{\mathbf{q}}$ The robot coriolis and centrifugal matrix

$\mathbf{d}_k$ The disturbance (model) correction term at time k in the general MPC framework

DCC Dynamic Collision Checking

δ_t Time-step; the sampling period for a discrete-time controller or observer

$\boldsymbol{\delta}_{\mathbf{q}}$ The joint position perturbation vector resulting from closed-loop motion; describes the difference between the estimated *open-loop* joint position and the actual *closed-loop* joint position

$\boldsymbol{\delta}_{\mathbf{q}}^{max}$ The upper bound vector on perturbations in joint positions resulting from closed-loop motion

$\boldsymbol{\delta}_{\mathbf{q}}^{min}$ The lower bound vector on perturbations in joint positions resulting from closed-loop motion

$\boldsymbol{\delta}_{xyz\phi\alpha\psi}$ An upper bound on the magnitude of the object pose perturbation vector

$Distance_{Key}$ A SIFT parameter describing the image distance between the *key feature* and the neighbour

$Distance_{Match}$ A SIFT parameter describing the image distance between the *key's matched feature* and its neighbour

DoF Degree(s) of Freedom

DVC Dynamic Visibility Checking

e Shorthand for the robot end-effector frame, $\mathcal{F}_e$, when used in a homogeneous transformation, for example, ${}^{m}\mathbf{T}_e$

$\mathbf{e}_{2^1/_2\text{-D}}$ The error vector for $2^1/_2$-D visual servoing

$\mathbf{e}_{\text{IBVS}}$ The error vector for IBVS

$\mathbf{e}_{\text{PBVS}}$ The error vector for PBVS

EKF Extended Kalman Filter

$\boldsymbol{\epsilon}_{i|k}$ The errors at time i *predicted* at time k in the general MPC framework

$\boldsymbol{\eta}$ Zero-mean Gaussian noise vector in the nonlinear output equations of the observer model

$\mathbf{F}(\mathbf{x}, \mathbf{u})$ The nonlinear state difference equations of the observer model used for pose estimation

f The focal length of lens

$\mathcal{F}_b$ The robot base frame

$\mathcal{F}_c$ The camera frame

$\mathbf{F}_{\text{control}}$ The state feedback matrix of the closed-loop plant in PBVS

$\mathcal{F}_{c^*}$ The desired camera frame

$\mathcal{F}_e$ The robot end-effector frame

$\mathcal{F}_o$ The object frame

$f_{\text{robot}}(\mathbf{q})$ The forward kinematic model of the robot in terms of joint angles

$\mathbf{G}(\mathbf{x})$ The nonlinear output equations of the observer model used for pose estimation

$\mathbf{g}_{\mathbf{q}}$ The robot gravity vector

$\boldsymbol{\gamma}$ Zero-mean Gaussian disturbance vector in the nonlinear state difference equations of the observer model, consisting of ${}_1\gamma$ and ${}_2\gamma$

GPC Generalized Predictive Control

$\mathbf{H}$ A homography matrix associated with points on plane π, describing the transformation of image coordinates in the desired image to image coordinates in the current image

$\mathbb{H}_j$ The set of 3-D points representing the hull of link j of the robot manipulator; used for collision tests

IBVS Image-Based Visual Servoing

$\mathbf{J}_{\text{robot}}$ The Jacobian relating robot joint velocities to camera velocities expressed in the camera frame

$\mathbf{J}_{\phi\alpha\psi}$ The Jacobian relating the linear and angular velocities of the object expressed in the camera frame to the rate of change of X-Y-Z and roll-pitch-yaw angles expressed in the camera frame

$\widetilde{\mathbf{K}}$ The Kalman gain matrix used in EKF pose estimation

k_u The horizontal scale of imaging array in number of pixels per unit length

k_v The vertical scale of imaging array in number of pixels per unit length

κ A damping factor, used in the damped least-squares inverse

$\ell(\mathbf{q}(t))$ The longest distance traveled by any point on the robot during trajectory $\mathbf{q}(t)$; used in the DCC algorithm

$\widetilde{\ell}(\mathbf{q}(t))$ The longest distance traveled by any point on the target object with respect to the camera's frame of reference during trajectory $\mathbf{q}(t)$; used in the DVC algorithm

$\mathbf{L}_{2^1/_2\text{-D}}$ the interaction matrix for $2^1/_2$-D visual servoing relating the camera velocity to the rate of change of the $2^1/_2$-D visual servoing error vector

$\mathbf{L}_{\text{IBVS}}$ The interaction matrix for IBVS relating the camera velocity to the rate of change of the IBVS error vector

$\mathbf{L}_{\text{PBVS}}$ The interaction matrix for PBVS relating the camera velocity to the rate of change of the PBVS error vector

λ The convergence rate of a visual servoing control law

LQG Linear Quadratic Gaussian

LQR Linear Quadratic Regulator

$\mathbf{m}_j$ The normalized metric image coordinates of feature point j expressed in the current camera frame in homogeneous form; (*) ... in the desired camera frame

MIMO Multiple-Input Multiple-Output

MIS Minimally Invasive Surgeries

MPC Model Predictive Control

MRI Magnetic Resonance Imaging

$\mathbf{n}$ The normal of plane, π, expressed in the current camera frame; (*) ... in the desired camera frame

N_c The control horizon in the general MPC framework

N_p The prediction horizon in the general MPC framework

o Shorthand for the object frame, $\mathcal{F}_o$, when used in a homogeneous transformation, for example, ${}^m\mathbf{T}_o$

Ori_{Key} A SIFT parameter describing the orientation at which the *key feature* is detected

$Ori_{KeyMatch}$ A SIFT parameter describing the orientation at which the *key's matched feature* is detected

$Ori_{Neighbor}$ A SIFT parameter describing the orientation at which the *neighbour feature* is detected

$Ori_{NeighborMatch}$ A SIFT parameter describing the orientation at which the*neighbour's matched feature* is detected

$(\mathbf{p}_j)_{i|k}$ The image coordinates (u,v) of feature j at time i *predicted* at time k in the MPC visual servoing framework

$\mathbf{p}_j$ The image coordinates of feature point j in homogeneous form

$\widetilde{\mathbf{P}}$ The state covariance matrix obtained from EKF pose estimation, representing the covariance of the estimated states

${}^m\overline{\mathbf{P}}$ An object model consisting of the 3-D coordinates of all feature points expressed in frame m,

${}^m\mathbf{P}_j$ The 3-D coordinates of feature point j expressed in frame m in homogeneous form

PBVS Position-Based Visual Servoing

PCI Periheral Component Interconnect Bus

ϕ Roll; the angle of rotation about the z-axis, applied in the order z-y-x about current frames

$\boldsymbol{\pi}$ A plane defined by three 3-D points

PID Proportional Intergal Derivative

$\overline{\mathbf{p}}$ A vector of the image coordinates of all feature points in the MPC visual servoing framework

$\overline{\mathbf{p}}^d$ A vector of the *desired* image coordinates of all features in the MPC visual servoing framework

$\overline{\mathbf{p}}_{i|k}$ A vector of the image coordinates of all features at time i *predicted* at time k in the MPC visual servoing framework

$\overline{\mathbf{p}}_k$ A vector of the *observed* image coordinates of all features at time k in the MPC visual servoing framework

PRM Probabilistic Roadmap

ψ Yaw; the angle of rotation about the x-axis, applied in the order z-y-x about current frames

$\mathbf{Q}$ A weighting matrix whose elements determine the weight given to the errors associated with each image feature in the MPC visual servoing control law

$\mathbf{q}$ The robot joint position vector

$\mathbf{q}_{DCC}^{max}$ The upper bound vector on the hyperrectangular collision-free space returned from DCC

$\mathbf{q}_{DCC}^{min}$ The lower bound vector on the hyperrectangular collision-free space returned from DCC

$\mathbf{q}_{DCC_k}^{max}$ The upper bound vector on the dynamically updated hyperrectangular collision-free space returned from DCC at time k

$\mathbf{q}_{DCC_k}^{min}$ The lower bound vector on the dynamically updated hyperrectangular collision-free space returned from DCC at time k

$\mathbb{Q}_{\text{desired}}$ The set of all robot joint configurations that satisfy the desired end-effector position; Set of all inverse kinematic solutions to the robot

$\mathbf{q}_{\text{desired}}$ The desired robot joint configuration; the goal configuration in path planning

$\mathbb{Q}_{free}$ The collision-free space, representing the set of all robot configurations that do not result in physical collisions with workspace obstacles

$\mathbf{q}_{\text{IBVS}}(\zeta)$ The joint space path, parameterized by ζ, resulting from IBVS

$\mathbf{q}_{i|k}$ The robot joint positions at time i *predicted* at time k in the MPC visual servoing framework

$\mathbf{q}_{\text{initial}}$ The initial robot joint configuration; the start configuration in path planning

$\widetilde{\mathbf{Q}}$ The disturbance covariance matrix used in EKF pose estimation, representing of the covariance of the disturbances in the state difference equations

$\dot{\mathbf{q}}$ The robot joint velocity vector

$\dot{\mathbf{q}}_{i|k}$ The robot joint velocities at time i *predicted* at time k in the MPC visual servoing framework

$\ddot{\mathbf{q}}$ The robot joint acceleration vector

Δq_j The change in position of joint j between two robot configurations

$\dot{\mathbf{q}}^{max}$ An upper bound vector on feasible robot joint velocities

$\dot{\mathbf{q}}^{min}$ A lower bound vector on feasible robot joint velocities

$\hat{\mathbf{q}}$ The estimated *open-loop* joint position, resulting from MPC path planning

$\mathbf{q}^{max}$ An upper bound vector on feasible robot joint positions

$\mathbf{q}^{min}$ A lower bound vector on feasible robot joint positions

QP Quadratic Programming

$\mathbf{r}$ The unit vector defining the axis of rotation of ${}^{m}\mathbf{R}_{n}$ in axis-angle representation

r_j The greatest distance between the line defined by the axis of joint rotation $\mathbf{z}_j$ and a point $\mathbf{P}$; a "moment arm"

$\widetilde{\mathbf{R}}$ The noise covariance matrix used in EKF pose estimation, representing the covariance of the noise in the output equations

${}^{m}\mathbf{R}_{n}$ A rotation matrix expressing the coordinates of frame n in the coordinates of frame m

ρ_1 A parameter used in the $2^1/_2$-D visual servoing control law, defined as $\frac{Z}{d^*}$

ROI Region of Interest

RTX Real Time Extension

$\mathbf{s}$ A feature vector selected to define the error vector to be minimized in visual servoing

$Scale_{Key}$ A SIFT parameter describing the image scale at which the *key feature* is detected

$Scale_{KeyMatch}$ A SIFT parameter describing the image scale at which the *key's matched feature* is detected

$Scale_{Neighbor}$ A SIFT parameter describing the image scale at which the *neighbour feature* is detected

$Scale_{NeighborMatch}$ A SIFT parameter describing the image scale at which the *neighbour's matched feature* is detected

SIFT Scale-Invariant Feature Transform

σ_i The i^th singular value of a matrix, resulting from SVD

$\boldsymbol{\sigma}_{xyz\phi\alpha\psi}$ The standard deviation vector corresponding to the object pose perturbation vector

SQP Sequential Quadratic Programming

SVD Singular Value Decomposition

${}^{m}\mathbf{T}_{n}$ A homogeneous transformation expressing the coordinates of frame n in the coordinates of frame m

${}^{m}\mathbf{t}_{n}$ A translation matrix expressing the coordinates of frame n in the coordinates of frame m

$\mathbf{T}_{Rx}(\psi)$ A homogeneous transformation expressing a rotation of ψ about x

$\mathbf{T}_{Ry}(\alpha)$ A homogeneous transformation expressing a rotation of α about y

$\mathbf{T}_{Rz}(\phi)$ A homogeneous transformation expressing a rotation of ϕ about z

$\mathbf{T}_{xyz}(x, y, z)$ A homogeneous transformation expressing a translation of x, y, z

$\boldsymbol{\tau}$ The robot joint torque vector

θ The angle of rotation about $\mathbf{r}$ of ${}^{m}\mathbf{R}_{n}$ in axis-angle representation

θ_{error} The rotation estimation error defined as the angle of rotation of ${}^{c}\mathbf{R}_{c^*}\,{}^{c}\widehat{\mathbf{R}}_{c^*}^{-1}$ in axis-angle representation

$\mathbb{U}$ The set of possible inputs in the general MPC framework

$\overline{\mathbf{u}}$ A vector of the horizontal image coordinates of all feature points

u_0 The horizontal image coordinate of the principal point

$\mathbf{u}_{i|k}$ The system input at time i *predicted* at time k in the general MPC framework

u_j The horizontal image coordinate of feature point j

$\mathbf{u}_{\text{obsv}}$ The inputs to the observer model used for pose estimation

u^{max} The maximum pixel coordinate of the camera CCD array in the horizontal direction

u^{min} The mininum pixel coordinate of the camera CCD array in the horizontal direction

$\overline{\mathbf{v}}$ A vector of the vertical image coordinates of all feature points

v_0 The vertical image coordinate of the principal point

v_j The vertical image coordinate of feature point j

ε The size of the singular region for singularity avoidance; used in the damped least-squares inverse to determine the damping factor

$\mathbf{v}_c$ The spatial velocity of the camera with respect to the global frame

$\mathbf{v}_o$ The spatial velocity of the target object with respect to the global frame, tracked as states in the pose observer

$\mathbf{W}$ A weighting matrix whose elements determine the weight given to the velocities associated with each robot joint in the MPC visual servoing control law

${}^{c}\boldsymbol{\omega}_{o}$ The pose vector of the target object expressed in the current camera frame, tracked as states in the pose observer

$\mathbb{X}$ The set of possible states in the general MPC framework

$\mathbf{x}_{i|k}$ The system state at time i *predicted* at time k in the MPC framework

x_j The normalized metric image coordinates of feature point j in the horizontal direction, expressed in the current camera frame; (*) ... in the desired camera frame

${}^{m}X_j$ The x-coodinate of the origin of frame j (or feature point j) expressed in frame m

$\mathbf{x}_{\text{obsv}}$ The states of the observer model used for pose estimation

$\mathbf{X}$ The 3-D coordinates of a feature point expressed in the current camera frame; (*) ... in the desired camera frame

$\mathbf{y}^{d}$ The desired system output in the general MPC framework

$\mathbf{y}(\mathbf{x}_{i|k})$ The system output at time i *predicted* at time k in the general MPC framework

y_j the normalized metric image coordinates of feature point j in the vertical direction, expressed in the current camera frame; (*) ... in the desired camera frame

${}^{m}Y_j$ The y-coodinate of the origin of frame j (or feature point j) expressed in frame m

$\mathbf{z}_j$ The axis of rotation of revolute joint j

Z_j The "depth" of feature point j; the Z-coodinate of feature point j expressed in the current camera frame; (*) ... in the desired camera frame

${}^{m}Z_j$ The z-coodinate of the origin of frame j (or feature point j) expressed in frame m

$\mathbf{z}_{\text{obsv}}$ The outputs of the observer model used for pose estimation

ζ	A path variable where $\zeta = 0$ corresponds to the beginning of the path and $\zeta = 1$ corresponds to the end of the path

Acknowledgements

I would like to thank my supervisors, Dr. Elizabeth A. Croft and Dr. James J. Little, for their guidance and support throughout the thesis. I am very grateful for all the useful advice and learning opportunities that have resulted from their patient mentoring.

I would also like to thank Dr. Ryozo Nagamune and Dr. Bob Woodham for their guidance and suggestions on the thesis.

I would like to thank Dr. Simon Leonard for the many hours of fruitful discussions on visual servoing, and for help on setting up the experiments.

To my colleagues in the CARIS lab, thank you for your friendship and support, and for giving me a healthy perspective on life, when things get a little hectic.

A special thanks goes to my parents for their love, support and encouragement, and for always giving me the freedom to pursue my interests. To my sister, thank you for being there for the family, while I am temporarily away for my studies.

Finally, I would like to thank Theresa for her love, support, encouragement. The many hours spent on editing my thesis, listening to my presentations, and patiently waiting for me to come home on late nights, are greatly appreciated.

To the faculty and staff of the Mechanical Engineering Department, thank you for your assistance and support throughout my learning experience here at UBC. I would also like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada.

Chapter 1

Introduction

The ability to teach real robots how to carry out physical positioning tasks safely and accurately, using a set of natural human commands, has been a life-long goal of many robotic system designers. Vision is naturally used by humans for demonstrating to each other spatial locations and multi-body motions. As a natural extension of this *teach-by-showing* paradigm, a human user can present a robot with a visual description of a positioning task, which can be readily captured using an imaging sensor. The approach removes the need for tedious and cumbersome numerical robot programming, so that complex tasks can be easily taught to robots by non-technical users.

Often, a robot manipulator is required to position its end-effector with respect to a target object for grasping. A digital camera can be used by the user to teach the required task to the robot. Given that this task requires relative positioning, it is often advantageous to mount the camera near the end-effector in an *eye-in-hand* configuration, to provide a close-up, non-occluded view of the target object for feedback purposes. The user initiates the visual command by showing the robot a reference image of the desired grasping position. This image describes what the camera should see when the end-effector is correctly positioned relative to the target object. The robot then compares the current image against the reference image, and generates a trajectory that efficiently reduces the difference between the two images at each time step. The task is complete when the observed image is in agreement with the reference image, resulting in the correct positioning of the end-effector relative to the target object. This method of end-effector position control using feedback from an arm-mounted camera is known as *eye-in-hand visual servoing*. An example of a positioning task achieved via eye-in-hand visual servoing is shown in Figure 1.1.

This *teach-by-showing* method is not only intuitive to human users, but also offers several technical advantages over traditional, position-based robot programming. Vision-based feedback allows the generalization of manipulator programming to accommodate frequent changes in the target object's location. Exact knowledge of the object's location and precise camera-robot calibrations are not required, due to the closed-loop control structure. Ex-

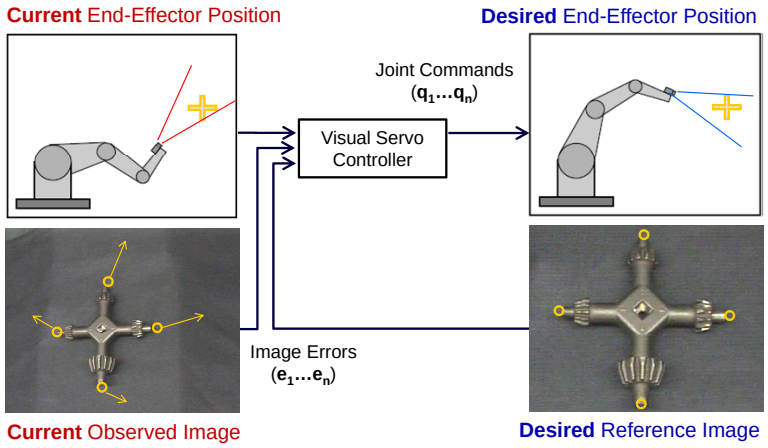


Figure 1.1: Visual servoing uses the image differences observed by the camera as feedback to drive the end-effector towards a desired position relative to the target object.

pressing the Cartesian positioning task using an image-based metric effectively cancels out calibration errors in the intermediate reference frames. The position of the end-effector relative to the target object is guaranteed to be accurate, provided the currently observed image corresponds to the reference image.

Despite a large body of existing work in this area [1], a fundamental gap still exists between the current capability of visual servoed robotic systems and the requirements of real world robotic applications. Beyond its use for correcting small planar motions, there has been limited commercial adoption of visual servoing for manipulator control, primarily due to concerns around reliability, safety, and stability. Most research to-date in visual servoing has focused on the control aspect from the perspective of a free-floating camera, paying little or no attention to the physical robot that is used to drive camera motion. However, robots have mechanical joint limits and finite performance capabilities which restrict where a camera mounted to the robot can go and how fast it can move. Cameras themselves have limited fields and depths of view. During servoing, the image-based control law can become unstable if the camera loses sight of the target object, or if the required trajectory exceeds the robot's physical limits. Furthermore, the image-based control law could result in dangerous whole-arm collisions, if the system is unaware of the physical relationships between the robot

manipulator and its environment. These issues must be addressed before visual servoing can be reliably used in applications that demand a full range of robot motions.

This work aims to address the aforementioned practical issues which impede the application of manipulator visual servoing in real life applications. For non-trivial robot motions, the stability of a visually servoed system depends upon a prudent awareness of the robot's sensing, control, and physical limits. Successful design of a visual servoing control law, that can effectively manage these constraints while carrying out its positioning task, will lead to a versatile robotic system that can execute large-range, safe motions while adapting to observed changes in the target object's pose.

1.1 Motivation

As described above, manipulator visual servoing provides a powerful method to compensate for uncertainties that exist in robotic systems and their environments. In an ideal world where system parameters are precisely known and there are no expectations of disturbances, one can simply use an open-loop approach without the need to servo. However, many engineering systems cannot be simply realized with this approach. In manufacturing applications, costs tend to increase exponentially with increases in required open-loop accuracy, stemming from the need to reduce tolerances recursively at each design layer. In surgical applications involving complex biological systems, it may not be possible to obtain an accurate open-loop model. Finally, for human-in-the-loop applications, it is impractical to design a system that relies upon precise numeric inputs from human users. For such systems, visual servoing provides a natural way for humans to functionally communicate with machines without demanding meticulous precision. Some specific examples that illustrate the need for reliable visual servoing approaches are provided in this section.

1.1.1 Industrial Automation: Reduced Setup Costs

Industrial robots perform a variety of automated tasks such as component assembly, pick-and-place, inspection, cutting, painting, welding, and lifting of heavy loads. All of these tasks require the robot end-effector to be accurately positioned with respect to a target object prior to the execution of the task. Traditionally, manufacturers have employed *record-and-playback* programming methods, which rely on mechanical fixtures, positioning devices and human labour to present objects to the robot's pre-programmed

interfacing locations. However, the use of structured work-cells demand substantially higher costs stemming from engineering development, capital investment, and system maintenance. For the manufacture of goods with high volumes and long lifecycles, these costs are still justified. However, the growing demand for mass-customization of durable and non-durable consumer goods poses a major challenge to this approach. Where production volume is sensitive to changing customer demands, manufacturers must move towards flexible and adaptive production systems that can be quickly reconfigured to produce a wide array of products using the same infrastructure. Reduction in equipment costs and work-cell setup time are the main advantages for using visual servoing in manufacturing environments.

Manipulator visual servoing provides a low-cost, highly adaptive, and calibration-free solution to the relative end-effector positioning problem. Commercial off-the-shelf cameras are available at a fraction of the cost of a robot, while providing sufficiently rich sensory data for the robot to make decisions for intelligent control. Using software to detect and correct for positional errors allows the system to be quickly adapted to suit a variety of different robots, cameras and industrial parts without extensive retooling. The use of direct image-based feedback for trajectory correction makes the robot's positional accuracy robust to calibration errors.

Unfortunately, the reliability of manipulator visual servoing methods must be proven before manufacturers will consider adopting them for industrial use. Manufacturers do not have the luxury of employing solutions that have appreciable chances of failure, since frequent downtimes can interrupt workflow and negate the productivity that is gained by automation. A necessary requirement is that the chance of visual servo failure must remain near zero. One frequently voiced concern is that visual servoing in uncertain conditions may generate unsafe robot motions resulting in costly damage to equipment. The visual servo control law must be made aware of its position, velocity, and acceleration limits and provide a guarantee for safe, collision-free robot motions. The system must be designed such that if visual input is interrupted due to unforeseen conditions, the position task can continue to operate in a stable manner until visual input is restored.

1.1.2 Medical Robotics: Adaptive Trajectories

The ability to generate robot trajectories that can adapt to patient movement is one of the main motivations for using manipulator visual servoing in medical applications. Advanced robotic surgical systems, such the Intuitive Surgical da Vinci© shown in Figure 1.2, assist surgeons in performing min-

Figure 1.2: The da Vinci© surgical system uses robotic arms to control the position of cameras and surgical tools to assist surgeons in performing minimally invasive procedures [2].

imally invasive surgeries (MIS) by using its robotic manipulators to control cameras and laparoscopic tools. The use of a master-slave robotic system eliminates the awkward inverted manipulation of laparoscopic instruments that is required of the surgeon. However, in MIS, the surgeon cannot directly see the surgical field and must rely on scans of the patient's anatomical structures to plan the surgery. Magnetic resonance imaging (MRI) and computed tomography (CT) are generally used to provide high resolution images of the surgical site prior to the operation. During the surgery, however, any previously identified structures may have shifted in location due to the patient's movements. Real-time imaging devices, such as ultrasound and endoscopic cameras, are used to provide visual feedback to the surgeons to correct for such deviations.

Throughout the surgery, the position of these imaging devices must be actively managed in order to provide useful information to the surgeon. The imaging planes of ultrasound transducers have finite effective thicknesses and cameras have limited fields of view. The position of these sensors must be frequently adjusted to compensate for patient motion which may bring the surgical field outside of the sensor's effective limits. Visual servo control of these sensors can free the surgeon from tedious adjustments, so that attention can be redirected towards the surgical task. When a new sensor viewpoint is required, the surgeon needs only to move the sensor in the

correct position once to obtain a reference image. The sensor maintains this viewpoint through visual servoing, providing the surgeon with a consistent frame of reference, despite patient motions.

The proposed visual servoing system for sensor position control must be safe and reliable. In medical applications, any chance of visual servo failure is simply not acceptable because of the inherent dangers to the patient. At the most basic level, the manipulator and sensor should be able to maintain a safe distance from its defined limits during servoing. In the case of an in-vivo sensor, the movement of the sensor should not cause further damage to the surgical site. The ideal servo system should incorporate information from high-fidelity models built from offline CT or MRI images to constrain the robot's movement from operating in unsafe regions.

1.1.3 Domestic Service Robots: Ease of Programming

The ease of robot programming and re-programming is one of the main advantages of using visual servoing for domestic robot applications. Domestic robots must be designed with usability in mind. Unlike industrial robots, domestic robots do not have the luxury of precise calibration, nor the dedicated support of engineering teams when robot re-programming is required. Without resorting to low-level numerical programming, the user must be able to efficiently re-program the robot when new interaction scenarios arise. The robot may need to interact with objects that it has not encountered before. For example, the user may want to command the robot to push a light switch, turn a door knob, or pick up a household object, the precise location of which is unknown. The robot must be able to achieve sufficient positional accuracy in the absence of precise camera-robot calibrations and without exact *a priori* knowledge of the target object's location. If the robot has visual servoing capabilities, the user can train the robot using a reference image. The user obtains this image first, by showing the robot what the finished task looks like. This image captures the desired relative position between the robot and the object of interest. Coupled with visual servoing, this *teach-by-showing* method allows the robot to generalize its programming to accommodate for changes in the target object's location, permitting task generalization. As in the case with manufacturing and medical surgery, the visually servoed manipulator must be aware of its physical, sensing, and control limits in order to ensure safe operation.

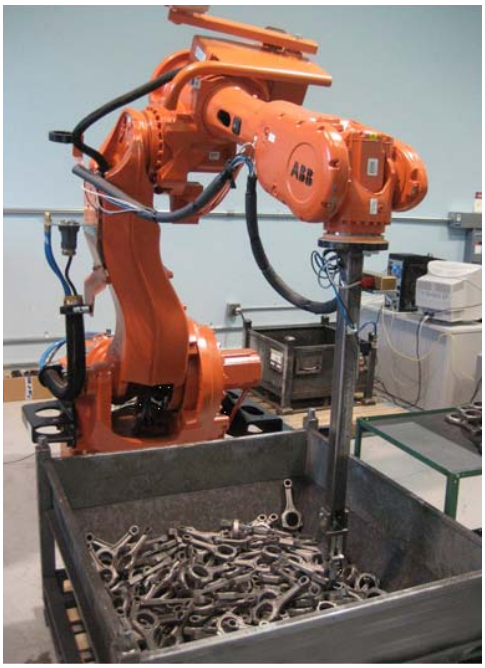


Figure 1.3: Robot bin-picking requires the accurate positioning of the robot end-effector with respect to randomly located parts, without causing collision with workspace objects, such as the container shown. Image courtesy of Braintech Inc. [3].

1.2 Robot Bin-Picking: An Example

This thesis will return frequently to the following example of robot bin-picking to provide a useful context for manipulator visual servo control.

Consider the common industrial task that is required when a container of unfinished parts is presented to a robot for further handling and processing. This task is typically referred to as robot bin-picking. Bins and containers are widely used to store, protect and transport products ranging from automotive parts to agricultural produce. Manufacturers spend considerable effort and energy on the loading and unloading of parts into and out of bins and constainers, so automation of this repetitive task using programmable robotic manipulators is highly desirable. An example of a robot bin-picking prototype is shown in Figure 1.3.

Three different robot bin-picking implementations are presented below

(in the order of increasing use of vision as feedback) to compare the advantages and disadvantages of visual servoing with other commonly used approaches. These illustrative examples are representative of the continuum of robotic automation in bin-picking applications.

The first implementation uses a *record-and-playback* approach, where the robot blindly executes a set of pre-programmed bin-picking motions. This requires prior organization of parts, using pallets or carriages to structure the parts into rows and columns inside the container. Mechanical fixtures must be designed for each part variant; as a result, robotic automation costs are very high. Such automation is limited to long-term, high-volume production lines with limited part changes. Robot manipulation will fail if the part-to-part positional tolerances are too large.

The second implementation uses a *look-then-move* approach, where a fully calibrated camera and a precise CAD model of the part are used to estimate the part's pose (position and orientation) from a single sensed image. In this implementation, a fully calibrated robot is commanded to move its end-effector into a calculated configuration to pick up the part. The robot's workspace must be free of obstacles, in order to guarantee collision-free motion. Robot manipulation will fail if there are significant calibration errors in the camera or in the robot (resulting in errors in pose estimation or in robot position control), or if the part has shifted in location since the last image was taken.

The third implementation uses a *visual servoing* approach to further eliminate the dependency on precise camera and robot calibration. Exact offline computations of the part's pose and the robot's desired configuration are not required. Rather, the robot uses online visual feedback to continuously correct its trajectory as it approaches the part. The robot has full control of its trajectory, though it must not exceed the sensing, control, and physical collision limits of the system in order to provide safe robot motion. This approach ensures that the manipulator achieves a positional accuracy that is robust to modelling errors and disturbances.

1.3 Problem Statement

The goal of this research is to enable the use of manipulator visual servoing for teaching practical robot motions. While traditional visual servoing methods perform well for correcting small planar deviations, they cannot be used for teaching general positioning tasks because, by the design of their control law, they are inherently unaware of the sensing, control, and physical limits

of the robotic-camera system. Naive application of traditional visual servoing methods to large robot motions could result in dangerously unstable trajectories, due to: (i) the target object leaving the sensor's field-of-view; (ii) the robot being commanded to move beyond its mechanical limits, or beyond it performance limits; or (iii) the robot arm physically colliding into other objects within its workspace. This work will address all three of the above problems that are typically encountered when using visual servoing to drive non-trivial robot motions.

1.4 Research Objectives

The objectives of this work are to design a visual servo control law that:

1. Keeps the target object within the camera's field-of-view at all times during servoing.
2. Generates realistic camera-robot motions that remain within the robot's dexterous workspace and mechanical limits.
3. Generates time-efficient, but safe robot trajectories. These trajectories should not exceed the robot's velocity limits, but should allow the use of high feedback gains while servoing.
4. Avoids manipulator singularities, which affect the controllability of the robot's end-effector position.
5. Generates collision-free robot motion, given some knowledge of the robot's environment.

The above objectives are integrated, where possible, in a unified framework. Further, the research aims to provide insight into the following questions:

- For an eye-in-hand robot, how should the trajectory be specified to provide a good balance between an ideal robot trajectory (joint coordinates follow a straight line) and an ideal image trajectory (features move in straight lines on the image plane), while ensuring feasibility and task convergence?
- How does one design a robotic system that incorporates both model-based robot path planning methods (which are computationally efficient, but calibration-dependent) and reactive visual servoing methods

(which are robust to calibration errors, but may get stuck in local minima)?

- Should planning and servoing take place simultaneously, or in separate stages? If separate stages are required, then when should the positioning task be handed off from one stage to the next?
- When using vision-based path planning methods, how does one ensure that the target object stays within the camera's field of view, when the target object's location is uncertain?

1.5 Thesis Outline

The thesis is organized as follows:

This chapter (Chapter 1) outlines the main research questions and gives concrete examples to motivate why the constraints of an eye-in-hand sytem must be addressed by the visual servo control law, when it is used to drive large robot motions.

Chapter 2 contains a review of the state of the art in constraint-aware visual servoing, and in other related work that has influenced the contents of this thesis.

The core of the thesis is divided into four main chapters. Chapters 3 and 4 contain minor contributions to improvements in visual servoing. In addition, these chapters contain methodological work (on the *partial* and *complete* pose estimation a target object from eye-in-hand images) that is the basis of the joint-space path planning and visual servoing methods presented in Chapters 5 and 6. Chapters 5 and 6 contain the main contributions of the thesis, as outlined in Section 7.2. Experiment and simulation results are presented within each individual chapter after the appropriate discussions on methods. The core chapters of this thesis contain work in each of the following areas:

- Chapter 3 discusses the use of the image homography to improve the predictability of visual servoing motion. Scaled Euclidean parameters are recovered using feature point correspondences from images only, and are used in visual servoing to provide a decoupled control law. A method of estimating a homography when observing planar and non-planar objects is discussed. A method of decomposing the homography into its constituent parameters, and of selecting the correct solution amongst many that are possible, is also discussed. The performance of the proposed hybrid visual servo controller is evaluated in

simulation against the image-based visual servoing (IBVS) approach. Experiments are performed using images obtained from an eye-in-hand camera, to evaluate the accuracy of homography estimation and homography decomposition.

- Chapter 4 discusses the use of an extended Kalman filter (EKF) with a model of the target object, to improve the robustness of pose estimation with respect to image noise and disturbances. Position-based visual servoing (PBVS) is implemented with an adaptive gains controller to manage the camera's field-of-view limits. A method of generating acceptable robot trajectories while visual servoing near joint singularities is described. The performance of the adaptive gains PBVS controller working in conjunction with the EKF is evaluated in the presence of severe image noise and model disturbances.

- Chapter 5 discusses use of path planning as a prerequisite to visual servoing for managing robot-related constraints. A method for evaluating inverse kinematic solutions that considers the trajectory of the eye-in-hand camera is presented. An offline *plan-then-servo* collision-avoidance method is discussed, integrating probabilistic roadmaps (PRM) with visual servoing. A method to construct paths between two robot configurations that keep the target object within the camera's field of view is demonstrated to allow feasible transitions from planned motion to visual servoing. The performance of this path planning method is evaluated through simulations. A dynamic collision checking (DCC) algorithm and a dynamic visibility checking (DVC) algorithm is presented for ensuring that system constraints are satisfied during the motion between two robot configurations.

- Chapter 6 presents a unifying framework for constraint-aware visual servo control. An image-based visual servo controller is proposed using the model predictive control (MPC) framework, to allow planning and servoing to be executed simultaneously. Simulation results are presented to demonstrate the ability of the MPC visual servoing controller to manage eye-in-hand constraints while completing large-motion positioning tasks in closed-loop. A discussion on the tuning parameters in the visual servoing controller is given. A method to address pose uncertainty to ensure collision-free, closed-loop motion is statistically tested with multiple positioning tasks. An online iterative *plan-and-servo* collision-avoidance method is implemented using dynamically updated estimates of the collision-free space. Experimental results are

shown on a CRS-A465 robot with a Sony XC-HR70 camera performing an end-effector positioning task without calibration, demonstrating collision-free robot motions in the presence of pose uncertainty.

Chapter 7 presents the conclusions of this research. It summarizes the main contributions of the thesis and provides recommendations for future work in this area.

Chapter 2

Literature Review

This chapter provides a summary of related work done by other researchers to improve the performance characteristics of visually servoed systems. Visual servo control of robotic manipulators has been an active area of research for over 30 years and it is a topic that spans many disciplines. A good introduction for visual servo control can be found in [4], while a more recent survey paper covering many classes of visual servoing methods can be found in [1].

This literature review will focus on visual servoing methods that are most relevant to the goals of this research; that is, methods that are capable of handling large-range robot motions without violating system constraints. In this context, the review commences with a description of the major advances in the two main domains of visual servoing, namely image-based and position-based servoing. Next, two visual servoing methods that exploit the known camera structure to improve servo motion are discussed: epipolar-based servoing and homography-based servoing. Then, image path planning and image trajectory tracking methods are presented as extensions to classical robot control for minimizing system constraint violation. The chapter ends with a discussion of explicit constraint-avoidance in visual servoing for redundant manipulators and predictive control.

2.1 Advances in Image-Based Visual Servoing

Image-based visual servoing (IBVS) uses direct image measurements as feedback to control the motion of the robot. The robot's positioning task is expressed as an image-based error function to be minimized using a suitable control law. Because IBVS does not explicitly solve for the Cartesian pose of the target object, its performance does not depend on the accuracy of *a priori* models. However, since the domain of the control law is in image-space, there is no direct control over the Cartesian or joint-space trajectory of the robot end-effector. Application of IBVS to control large-range robot motions typically results in trajectories that are desirable in image space, but are convoluted in the robot's joint-space and may result in physical

trajectories exceeding the workspace of the robot. To address this problem, researchers have designed some image features and error functions that are tailored specifically for certain classes of positioning tasks and target objects, which are described below.

Classical IBVS uses the displacement between corresponding image feature points expressed in planar Cartesian coordinates to define the error function. Chaumette [5] describes the *camera retreat* problem encountered when classical IBVS is used to control a positioning task that requires a rotation about the camera's optical axis (denoted as the z-axis in the frame centered at and moving with the camera). Rather than commanding camera rotation while holding the camera position constant, the control law forces the camera to translate backward and forward along its optical axis while completing the rotation. Not only is such motion not time-optimal, it greatly increases the potential for robot collision, since this type of motion is neither intended nor expected by the user. In the worst case when the rotation required is exactly 180 degrees about the optical axis, the camera retreats to a distance of infinity and the servoing never reaches its goal. Iwatsuki *et al.* [6] propose the use of cylindrical coordinates in the definition of the image error function to avoid camera retreat. Unfortunately, this method forces the camera to rotate even for pure translational tasks, resulting in a problem similar to the original camera retreat, but for translational motions. An improvement to the cylindrical coordinate method is presented in [7] that allows the position of the origin to shift. However, determining the correct shift parameters requires an estimation of the rotational motion from the differences between the initial and the desired image, resulting in a partial pose estimation problem.

If the target object is known to be planar (and the object and camera planes are parallel at the desired position), then several other image features can be chosen for IBVS to achieve decoupled control. Corke *et al.* [8] introduce a partitioned approach that decouples the control of z-axis rotational and translational components from the remaining degrees of freedom. Z-axis translation is controlled by the square root of the area of the regular polygon whose vertices are the image feature points, while z-axis rotation is controlled by the angle subtended by the line segment connecting the two chosen feature points that are furthest apart. The use of image moments as visual features is proposed in [9] to avoid the problems of singularities and local minima. Good decoupling and stability properties are obtained through careful selection of combinations of moments to control the six degrees of freedom of the camera, although moments of higher orders tend to be more sensitive to image noise.

Cervera *et al.*[10] introduce the use of three-dimensional features in IBVS via a stereo camera mounted on the robot end-effector. A linear control law is obtained, resulting in a motion of the object along a straight path in the camera frame. For target objects that are symmetric (e.g., tetrahedron, square, cube), an *exact* computation of the velocity screw for any rotation can be obtained. In this formulation, the resulting motion of the camera is the same regardless of the 3-D model of the object.

2.2 Advances in Position-Based Visual Servoing

Position-based visual servoing (PBVS) uses an estimation of the pose of the target object with respect to the camera as feedback to control the motion of the robot. The robot positioning task is expressed as an error function composed of pose parameters. Computing that pose from a set of measurements in one image requires full knowledge of the camera intrinsic parameters and a 3-D model of the target object. This is closely related to the *3-D localization problem* in classical computer vision and many solutions have been presented in the literature [11] [12]. PBVS allows the decoupled control of translational and rotational motions. The resulting camera trajectory is a straight line in Cartesian space, while the image trajectories are less satisfactory than those in IBVS. The domain of the control law is in Cartesian-space, so there is no direct control over the trajectory of feature points on the image plane.

From the perspective of practical robot servoing, one of the main drawbacks of PBVS is that the target object (and its image features) may partially or entirely exit the camera's field of view during the servoed motion. Servoing can still continue when the feature losses are partial and where redundant features are still available for pose estimation. Wilson *et al.* [13] suggest the detection of feature losses to avoid errors in pose estimation and appropriate modification to the extended Kalman filter for tracking. The lost feature's entry in the covariance matrix is increased to a very large value, which has the effect of eliminating its influence on the pose estimate. The entry is returned to its normal value when the feature is detected to have returned back into the field of view. Lippiello *et al.* [14] [15] suggest the selection of an optimally-defined subset of image features for PBVS that is updated online to minimize the loss of image features. Several quality indices are proposed to take into account the spatial distribution, angular distribution, visibility and non-coplanarity of the image features, and the product of these indices is used as the basis of selection. However, both

of these approaches are passive and they do not prevent the target object from leaving the field of view *entirely*, resulting in pose estimation failure and servo failure. Servo failure occurs quite frequently in PBVS when large translations and large out-of-plane rotations are simultaneously required.

Chesi *et al.* [16] propose a switching control approach to keep the target within the field of view of the camera. PBVS is selected when all feature points are inside the field of view, implemented as a region of interest (ROI) in the image. When at least one feature point lies on the ROI boundary, a set of rotational and translational control laws is applied to push that feature point back into the region. If all of the above methods fail, the camera is sent away from the target object through a backward translational motion along the optical axis by a specified gain. One drawback of this method is that appropriate gains and hysteresis bands must be used to avoid chattering, a phenomenon resulting from frequent switching when feature points reside near the image boundary. Thuilot *et al.* [17] propose the use of a reference trajectory to coordinate the rate of convergence between translation and rotation motions in PBVS. Rotation errors are allowed to decrease exponentially without any reference trajectory, while the translational reference frame is adjusted to keep the geometric center of the feature points in the camera field of view.

Cervera [10] and Deng [18] explore the use of alternative coordinate frames for defining the PBVS error vector to obtain improved servoing characteristics. Two methods are discussed. The first method defines the pose error vector in the stationary target frame. Here, the shortest straight-line translation of the camera is achieved at the expense of a highly nonlinear target pose sequence where image trajectories can easily leave the camera's field of view. The second method defines the pose error vector in the current camera frame. If a reference point close to the geometric center of all feature points is chosen as the target frame origin, the chances of image trajectories leaving the camera field of view is significantly minimized. However, the ability to generate straight-line Cartesian trajectories is lost.

Another difficulty associated with PBVS is that the stability of the servoing system is difficult to study, since it is sensitive to pose estimation errors. Zanne *et al.* [19] suggest using sliding mode control to design a PBVS controller that is robust to bounded parametric estimation errors. A switching controller is proposed using an appropriate selection of the sliding surface, based on the quaternion representation for rotations, to ensure stability.

2.3 Epipolar-Based Visual Servoing

When a camera views a 3-D object from two distinct positions, there exists a number of geometric relations between the 3-D points and their projections onto the 2-D images that lead to some additional constraints between the two sets of observed image points. The epipolar geometry describes such relationships between the two resulting views. Conversely, if a set of corresponding features in the current image and in the desired image is available, the epipolar geometry can be recovered and used for visual servoing [20]. Piazzi [21] and Marotta [22] propose a visual servoing algorithm that uses the observed epipolar geometry to construct a series of translations and rotations that iteratively bring the camera to the desired position. An advantage of this method is that no model of the scene is required and that the servoing is compatible with uncalibrated cameras. However, it is shown that recovering epipoles is often sensitive to image noise. Moreover, near the servo target, the epipolar geometry becomes degenerate and it is not possible to estimate accurately the partial pose between the two views. An alternative is to use homography-based methods which provide more stable estimations when the current and desired images are similar.

2.4 Homography-Based Visual Servoing

A *homography* is an invertible transformation that maps points from one 3-D plane to another 3-D plane. Since a camera sensor is also a projective plane in space, each 3-D plane that is observed by the camera in the current image is related to its projection in the desired image by an *image homography*. If all the feature points of a target object lie on a 3-D plane, then there is a single homography matrix that holds for all the observed feature points. This homography can be estimated using a minimum of four point correspondences between the current and desired image. If all feature points do not belong to the same 3-D plane, then three points can be used to define a *virtual* plane, and a minimum of five additional supplementary points are needed to estimate the homography [23].

Knowledge of this homographic relationship is advantageous because it can be decomposed to give a partial pose estimation that is useful for visual servoing. The camera rotation and the camera translation (up to a scale factor) can be recovered from the homography matrix via decomposition [24] [25] without requiring a model of the target object. Thus, some of the useful properties of PBVS such as decoupled control of camera rotation and

translation can be obtained using only point correspondences from images. Another advantage of this method is that it works universally for both planar and non-planar target objects.

There exist several related approaches that use the recovered rotation and scaled translation parameters for visual servoing. Deguichi *et al.* [26] use the recovered direction of translation with feedback control to bring the camera towards its goal in a straight line. Rotation is controlled to compensate the effects on the image caused by translation, to keep the target object in the field of view, and at the same time, to minimize the difference between the current image and the desired image. Malis *et al.* [27] propose a method known as $2^1/_2$-D visual servoing with a decoupled control law similar to PBVS. Image errors of a chosen reference point are used to control in-plane camera translation, while camera approach and rotation are controlled using parameters recovered from the homography matrix. Due to the simple structure of the system, the conditions for global asymptotic stability can be established. An alternative hybrid visual servoing scheme proposed in [28] is similar to $2^1/_2$-D visual servoing, but can tolerate a larger amount of calibration errors. However, this method is more sensitive to measurement errors since the task function is not directly computed as a difference of image data. In [29], the camera is controlled using the recovered direction of translation, while a single image point is chosen to control the x-axis and y-axis rotation, with the rotation matrix controlling the z-axis rotation. This approach results in the image point having a straight line trajectory so that it always remains in the field of view.

Homography estimation and decomposition based on three points defining a virtual plane (and additional points to solve for the relationship) can be extremely sensitive to image noise. In the above cases, the recovered motion is used as an approximation for servoing only, rather than for one-step position control. All servoing methods must continuously update the estimate on the homography as new images become available, in order to minimize the impact of estimation errors on the final servoing accuracy. As a result, these hybrid approaches tend to be more computationally intensive than PBVS and IBVS.

2.5 Image Path Planning

Many classical visual servoing methods work well when the initial and desired robot positions are close together. The idea behind image path planning methods is to use intermediate reference images to influence the path

of the servoing and to divide the required servoing task into smaller incremental tasks. Such intermediate reference images are typically not available from the camera (unless the user has the patience to teach the robot at every intermediate reference position!) so, in practice, the intermediate feature positions are generated via partial Euclidean reconstruction of intermediate camera poses using the initial image and the desired image. In the continuous case, an image-based reference trajectory is formed using interpolation in the image, combined with a suitable timing law to produce a time-varying reference for the visual servo controller to track. This significantly improves the robustness of visual servoing with respect to calibration and modelling errors, since the feedback error is always kept small.

An image-based path planning method that uses artificial potential fields and homography-based reconstruction to help avoid camera field-of-view limits and robot joint limits is used in [18] [30] [31] [32]. The path planning takes place at the beginning of the motion, starting with an estimation of a homography from the initial image to the desired image. The desired camera pose is found by decomposing the homography matrix into its rotation and translation components. A potential-fields approach [33] [34] is used to generate a path from the initial camera pose to the desired camera pose. An attractive potential field is used to bring the camera to its goal, while a repulsive potential field pushes the camera away from robot joint limits and another repulsive potential field pushes the camera away from positions that bring the target object out of view. The planned path is then re-projected back into the image and interpolated to give a reference image trajectory that is tracked using IBVS.

One disadvantage of the above approach is that the effectiveness of the potential field planning for camera and robot limit avoidance is highly dependent on the accuracy of the initial homography estimation. As discussed above, homography decomposition for scaled Euclidean reconstruction is known to be very sensitive to image noise. Unlike the hybrid visual servoing methods, there is no update to the homography estimation upon subsequent image observations, as there is no intermediate re-planning step. The actual camera path may deviate significantly from the initially reconstructed camera path, so that any repulsive potentials applied to the latter may not be effective in pushing the actual camera path away from the real robot joint limits and the real camera field-of-view limits.

If only the camera field-of-view visibility is of concern, Schramm *et al.* [35] propose an alternative path planning approach that does not require homography decomposition. The algorithm uses the properties of affine spaces to design a path that ensures the target object is always visible within the

camera's field of view. In [36], a helical trajectory is proposed to harmonize translation with rotation to maintain target visibility, without the use of potential field-based planning.

2.6 Image Trajectory Tracking

Close tracking of the image reference trajectory is important to ensure that the actual image trajectory does not exit the camera's field of view during servoing, despite the original path planning efforts. Close tracking of the planned trajectory is also important for ensuring predictable camera motion for avoiding robot joint limits. To minimize tracking errors, the visual servo control law must be modified to correctly anticipate the time-variation of the reference trajectory. Morel [37] and Zanne [38] propose a robust tracking controller for visual servoing that guarantees bounded tracking errors, with a bound that can be specified by the user to ensure visibility. Rather than naively increasing the control gains to achieve the desired precision, the reference velocity is modulated to satisfy an inequality relationship between the error bound, the parametric uncertainties in calibration, and the control gain. This method achieves bounded tracking errors without the use of large control gains and is robust to calibration errors. The only real drawback is that the resulting reference velocity may be conservative at times, so this method may be more suited to tasks that require precise geometric path tracking, but at low speeds.

2.7 Constraint Avoidance with Redundant Manipulators

If the required positioning task does not use up all degrees of freedom (DoF) available to the robot, the remaining DoF can be exploited by the visual servoing controller to avoid camera field-of-view limits, robot joint limits, singularities and obstacles. Redundancy-based solutions for eye-in-hand positioning and for visual tracking are discussed below.

For highly redundant robots completing 6-DoF positioning tasks, a gradient projection method is proposed for avoiding constraints related to visual servoing [39][40]. The constraints imposed by the environment are embedded as a cost function, and the gradient of this cost function is computed as a secondary task that moves the robot away from these constraints. This gradient is then projected onto the set of motions that keep the main task invariant and added to the control law that executes the main task. Because

the secondary task cannot have any effect on the main task in this formulation, only the degrees of freedom not controlled by the main task can be exploited to perform constraint avoidance. Mansard *et al.* [41] propose a method to improve the performance of the secondary task by enlarging the number of DoF available. An equality relationship in the classical redundancy formalism is replaced with a norm inequality relationship, to allow the execution of motions produced by the secondary control law that also *help* the main task to be completed faster. The amount of secondary motion that is gained by using this approach depends greatly on the degree of opposition between the task and the constraint.

Several methods have been proposed for handling 2-DoF tracking tasks via visual servoing while avoiding constraints. In [42], an object is visually tracked with an eye-in-hand camera while the robot simultaneously avoids kinematic singularities and joint limits by moving in directions along which the tracking task space is unconstrained or redundant. A manipulability measure is introduced into the visual tracking task function, such that the control law is able to take into account the configuration of the robot while the object is visual tracked. It is shown that for tracking objects with planar motions, the tracking region of the robot can be greatly increased using this method. In [43], a frequency-based framework is used to control a 5-DoF robot (a 2-DoF pan-tilt unit plus a 3-DoF Cartesian gantry) for tracking an operator as he walks around a workcell. A partitioning control law is used to exploit the kinematic and dynamic attributes of each DoF. The high-bandwidth pan-tilt unit is employed for tracking so that the fast-moving human target stays within camera's field of view, while the low-bandwidth gantry is used to reduce the bearing angle between the camera and the target, reducing the demand on the pan-tilt unit's range of motion. Analysis of phase characteristics show that the large bandwidth DoF that are visually servoed act as lead compensators for the remaining DoF with slow response times.

2.8 Predictive Control for Visual Servoing

Visual servoing can be formulated as an optimization procedure that computes the appropriate control input to the robot given a cost function that defines measured image errors. A linear quadratic Gaussian (LQG) control design can also be used to choose gains that minimize a linear combination of state and control input [44]. This approach explicitly balances the trade-off between tracking errors and robot motion. A more recent develop-

ment is the use of predictive control in visual servoing so that path planning and control can be solved simultaneously. Predictive control makes explicit use of a process model to obtain a control signal by minimizing an objective function. Generalized Predictive Control (GPC) is principally suited for single-variable linear control, since the process model is presented in the form of a polynomial transfer function, while Model Predictive Control (MPC) is formulated in state space and is designed for multi-variable control. In [45], GPC is used to provide improved tracking characteristics for a one-DoF profile-following task, while the remaining DoF are controlled using visual servoing. Sauvee *et al.* [46] presents a framework for controlling a robot positioning task using MPC with image feedback from a stationary camera. Successful avoidance of field-of-view limits (for the stationary camera observing the robot) and avoidance of joint limits (for the robot completing the task) are demonstrated in simulation. This approach is extended in [47] to a medical robot controlling a surgical instrument using MPC with image feedback from an stationary ultrasound transducer. In both [46] and [47], the imaging sensor is stationary with respect to the robot, and the geometry of the end-effector tool and the transformation from the camera frame to the robot base frame are assumed to be known.

2.9 Summary

The chapter presented an overview of the state-of-the-art in constraint-aware visual servoing. A description of the major advances in the two main domains of visual servoing are discussed. When large-range motions are required, it is shown that representing a positioning task as a minimization of image feature errors, as in IBVS, results in camera motions that naturally keep the target object within the field of view, but are convoluted in the joint-space of the robot. On the other hand, representing the positioning task as a minimization of rotation and translation errors, as in PBVS, produces efficient Cartesian trajectories (but the resulting joint-space motion can still be poor), which may cause the target object to exit the field of view. Some approximate knowledge of the pose of the target object, obtained through homography decomposition or pose estimation, is beneficial for improving visual servo motion since it allows path planning to occur. However, path planning must occur in the joint-space of the robot, if avoidance of robot joint limits and avoidance of whole-arm collisions with workspace obstacles is required. An explicit representation of system constraints in the visual servoing control law helps to ensure that robot positioning and

constraint avoidance can be simultaneously achieved.

This thesis will build on top of the strengths and weaknesses of the above approaches to arrive at a method for achieving collision-free and constraint-aware visual servo control of an eye-in-hand robot. Homography decomposition and online pose estimation are discussed in Chapters 3 and 4, as practical methods of obtaining an approximate pose of a target object for servoing and planning purposes. Path planning methods are discussed in Chapter 5, using a method inspired by IBVS to address camera field-of-view and robot joint-limit constraints, and combined with probablistic road maps to address collision constraints. Chapter 6 presents a unifying visual servoing framework that solves the problem of planning and servoing simultaneously, resulting in a visual servo control law that achieves close-loop positioning while being explicitly aware of system constraints. The cumulative goal is a visual servo control law that can be used to drive large-range robot motions without losing sight of the target object, violating robot joint limits or velocity limits, or colliding with workspace obstacles.

Chapter 3

Image Homography

3.1 Introduction

The objective of this chapter is to evaluate the suitability of homography-based methods for constraint-aware visual servoing. In particular, the ability of homography-based methods to extract Euclidean motion parameters without requiring a model of the observed object is quite attractive for use in the joint-space path planning methods that are presented in Chapter 5 and Chapter 6. The method of visual servoing presented and implemented in this chapter is known as $2^1/_2$-D visual servoing and it is based on methods proposed in [27]. The goal is to take advantage of the properties of the homography (as described in Section 2.4) to improve the predictability of visual servoing and to minimize the chance that the resulting motion will violate system contraints. The first section gives an overview of the visual servo control law. The second section describes how the requirements of the control law are met using images with matched feature points. In particular, the methods of homography estimation from observations of non-planar objects and of homography decomposition to recover scaled Euclidean parameters are discussed. Simulation results are shown for several positioning tasks that require large visual servoing motions. The performance of this homography-based visual servoing method is compared to that of traditional image-based visual servoing (IBVS). Finally, experimental results are obtained to assess the accuracy of homography estimation and decomposition using images from an eye-in-hand camera.

3.2 Homography-Based Visual Servoing

3.2.1 Overview

As explained in Chapter 1, the goal of eye-in-hand visual servoing is to control the 6-DoF position and orientation of the wrist-mounted camera relative to a target object, through the regulation of an error function defined by a set of visual features. This error metric describes the difference between

the current camera frame, $\mathcal{F}_c$, and the desired camera frame, $\mathcal{F}_{c^*}$. When expressed purely in image space, this error is the difference in the observed pixel locations of the visual features. When expressed purely in Cartesian space with respect to the current camera frame, this error is the translation, ${}^c\mathbf{t}_{c^*}$, defined in $\mathcal{F}_c$ and the rotation, ${}^c\mathbf{R}_{c^*}$, that describes $\mathcal{F}_{c^*}$ in the coordinates of $\mathcal{F}_c$.

In 2½-D visual servoing, the error metric consists of a combination of 2-D image features and 3-D parameters. To take advantage of the accuracy and the robustness of image-based techniques, the x-y translation of the camera is controlled by aligning the projected image coordinates of a real-world reference point P to that of the reference image. To capitalize on the decoupled nature of position-based techniques, the required camera rotation and the rate of approach in the z direction of the camera frame are estimated from the image using scaled Euclidean reconstruction. This approach does not require a model of the target object. All it requires is a minimum of eight matched points between the two images to estimate the homography of a virtual plane attached to the object defined by any three points.

3.2.2 Feature Selection

It is assumed that the object of interest remains rigid during visual servoing and that it can be represented as a distribution of feature points in 3-D space. A target point ${}^c\mathbf{P}_j$ is selected on the object to provide the required 2-D image coordinates (x, y) for measuring in-plane camera translations. An additional three points ${}^c\mathbf{P}_1$, ${}^c\mathbf{P}_2$, ${}^c\mathbf{P}_3$ are selected on the object to define a virtual plane π to act a 3-D reference for measuring the rotation ${}^c\mathbf{R}_{c^*}$ and the depth Z.

In 2½-D visual servoing, the feature vector $\mathbf{s}$ consists of 6 parameters that derived are from a combination of 2-D and 3-D metrics:

$$\mathbf{s} = \begin{bmatrix} \frac{X}{Z} & \frac{Y}{Z} & \log Z & \theta\mathbf{r}^T \end{bmatrix}^T = \begin{bmatrix} x & y & \log Z & \theta\mathbf{r}^T \end{bmatrix}^T, \tag{3.1}$$

where:

- (X, Y, Z) are the real-world coordinates of a 3-D target point ${}^c\mathbf{P}_j$ with respect to the camera frame.
- (x, y) are the normalized metric coordinates of the image point corresponding to ${}^c\mathbf{P}_j$.
- θ and $\mathbf{r}$ are, respectively, the angle and axis of rotation matrix ${}^c\mathbf{R}_{c^*}$ associated with the virtual plane π defined by three target points ${}^c\mathbf{P}_1$, ${}^c\mathbf{P}_2$, ${}^c\mathbf{P}_3$.

Knowledge of the intrinsic camera parameters is required, since the normalized metric coordinates $\mathbf{m}_j = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ of point ${}^c\mathbf{P}_j$ are related to its *measured* pixel coordinates $\mathbf{p}_j = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ by the camera matrix, $\mathbf{C}$:

$$\mathbf{m}_j = \mathbf{C}^{-1}\mathbf{p}_j, \tag{3.2}$$

where,

$$\mathbf{C} = \begin{bmatrix} fk_u & fk_u \cot\beta & u_0 \\ 0 & fk_v(\frac{1}{\sin\beta}) & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

where (u_0, v_0) are the images coordinates of the principal point, f is the focal length, β is the perpendicular skew angle, and k_u and k_v are the number of pixels per unit distance in x and y, respectively.

The current camera frame $\mathcal{F}_c$ is aligned with the desired camera frame $\mathcal{F}_{c^*}$ when the error function $\mathbf{e}_{2\frac{1}{2}\text{-D}} = (\mathbf{s} - \mathbf{s}^*)$ is zero in all of its components. Here, $\mathbf{s}^*$ corresponds to the feature measurements that are observed at the desired camera pose:

$$\mathbf{e}_{2\frac{1}{2}\text{-D}} = (\mathbf{s} - \mathbf{s}^*) = \begin{bmatrix} x - x^* & y - y^* & \log(\frac{Z}{Z^*}) & \theta\mathbf{r}^T \end{bmatrix}^T. \tag{3.4}$$

The depth component Z of the reference point ${}^c\mathbf{P}_j$ cannot be recovered from a single image. However, if the projective homography $\mathbf{H}$ of the virtual plane π is available (relating the observed image points of ${}^c\mathbf{P}_1$, ${}^c\mathbf{P}_2$, ${}^c\mathbf{P}_3$ in the two camera frames), then the ratio $\log(\frac{Z}{Z^*})$ and the rotation ${}^c\mathbf{R}_{c^*}$ (or $\theta\mathbf{r}$) can be recovered to complete the last 4 components of the error function. In summary, the goal of the $2\frac{1}{2}$-D visual servo control scheme is to regulate this error function, Equation 3.4, to zero.

3.2.3 Control Scheme

Most vision-based control approaches consider only robot kinematics, since the bandwidth for visual control is usually limited by the camera frame rate and image processing times. Robot dynamics are generally handled with a low-level PID controller using joint encoder feedback.

The $2\frac{1}{2}$-D visual servoing controller is designed at the velocity level, based on Jacobian matrices which relate the rate of change between the visual measurements and the robot joint angles. The Jacobian relationship between the spatial velocity of the camera $\mathbf{v}_c = \begin{bmatrix} {}_xv_c & {}_yv_c & {}_zv_c & {}_xw_c & {}_yw_c & {}_zw_c \end{bmatrix}^T$ and the time variation of the error function $\dot{\mathbf{e}}_{2\frac{1}{2}\text{-D}}$ is described by the interaction matrix $\mathbf{L}_{2\frac{1}{2}\text{-D}}$:

$$\dot{\mathbf{e}}_{2\frac{1}{2}\text{-D}} = \mathbf{L}_{2\frac{1}{2}\text{-D}}\mathbf{v}_c, \tag{3.5}$$

where,

$$\mathbf{L}_{2^1/_2\text{-D}} = \begin{bmatrix} \frac{1}{d^*}\mathbf{L}_{\mathbf{v}} & \mathbf{L}_{(\mathbf{v},\boldsymbol{\omega})} \\ 0 & \mathbf{L}_{\boldsymbol{\omega}} \end{bmatrix}. \tag{3.6}$$

The $2^1/_2$-D error function gives rise to an interaction matrix that is upper triangular, meaning that translation and rotation are decoupled in the control law. Expressions for the sub-matrices $\mathbf{L}_{\mathbf{v}}$, $\mathbf{L}_{(\mathbf{v},\boldsymbol{\omega})}$, $\mathbf{L}_{\boldsymbol{\omega}}$ are derived as follows, where the parameter ρ_1 is recovered from the homography $\mathbf{H}$ of a virtual plane π:

$$\mathbf{L}_{\mathbf{v}} = \frac{1}{\rho_1}\begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix}, \tag{3.7}$$

where

$$\rho_1 = \frac{Z}{d^*}, \tag{3.8}$$

$$\mathbf{L}_{(\mathbf{v},\boldsymbol{\omega})} = \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{bmatrix}, \tag{3.9}$$

$$\mathbf{L}_{\boldsymbol{\omega}} = \mathbf{I}_3 - \frac{\theta}{2}\left[\mathbf{r}\right]_\times + \left(1 - \frac{\operatorname{sinc}(\theta)}{\operatorname{sinc}^2(\frac{\theta}{2})}\right)\left[\mathbf{r}\right]_\times^2, \tag{3.10}$$

where $\left[\mathbf{r}\right]_\times$ is defined as the skew of the vector $\mathbf{r} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T$:

$$\left[\mathbf{r}\right]_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{3.11}$$

Here, a proportional velocity control law is designed to ensure the exponential convergence of each component in the error function towards zero by imposing:

$$\dot{\mathbf{e}}_{2^1/_2\text{-D}} = -\lambda \mathbf{e}_{2^1/_2\text{-D}}, \tag{3.12}$$

where λ is the convergence rate. Substituting the Jacobian relationship $\dot{\mathbf{e}}_{2^1/_2\text{-D}} = -\lambda \mathbf{e}_{2^1/_2\text{-D}}$ into the rate of change of error, an expression for the desired camera velocity is obtained:

$$\mathbf{v}_c = -\lambda \mathbf{L}^{-1}_{2^1/_2\text{-D}} \mathbf{e}_{2^1/_2\text{-D}}. \tag{3.13}$$

The above expression is always valid because $\mathbf{L}_{2^1/_2\text{-D}}$ is free of singularities in the entire workspace in front of the virtual plane (and is always

invertible). The desired camera velocity command is converted to robot joint velocities $\dot{\mathbf{q}}$ using the robot Jacobian relationship $\mathbf{J}_{\text{robot}}(\mathbf{q})$. The inverse relationship is valid when $\mathbf{J}_{\text{robot}}(\mathbf{q})$ is non-singular in the robot joint space:

$$\dot{\mathbf{q}} = -\lambda \mathbf{J}_{\text{robot}}^{-1}(\mathbf{q}) \mathbf{L}_{2^1/_2\text{-D}}^{-1} \mathbf{e}_{2^1/_2\text{-D}}. \tag{3.14}$$

Finally, the robot joint velocities $\dot{\mathbf{q}}$ commanded to the robot controller can be written as a function of the measured error components in $\mathbf{e}_{2^1/_2\text{-D}}$:

$$\dot{\mathbf{q}} = -\lambda \mathbf{J}_{\text{robot}}^{-1}(\mathbf{q}) \begin{bmatrix} d^* \mathbf{L}_{\mathbf{v}}^{-1} & -d^* \mathbf{L}_{\mathbf{v}}^{-1} \mathbf{L}_{(\mathbf{v},\boldsymbol{\omega})} \\ 0 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} x - x^* \\ y - y^* \\ \log(\frac{Z}{Z^*}) \\ \theta \mathbf{r} \end{bmatrix}. \tag{3.15}$$

The parameter d^* represents the distance from virtual plane π to the desired camera frame $\mathcal{F}_{c^*}$. Unfortunately, it cannot be recovered from the images, so it must be approximated off-line when the desired training image is acquired. Within reasonable bounds, this parameter only affects the relative convergence rate of the camera's translational and rotational components.

3.2.4 Servoing Requirements from Images

To summarize, the parameters that must be estimated from the images to be used as input into the control law, discussed in Section 3.2.3, are:

1. ${}^c\mathbf{R}_{c^*}$ (or in angle-axis form, $[\theta, \mathbf{r}]$)
2. $\log(Z/Z^*)$
3. $\rho_1 = Z/d^*$

The first two metrics define the error in 3-D rotation and the error in depth, respectively, between the current camera frame $\mathcal{F}_c$ and the desired camera frame $\mathcal{F}_{c^*}$. They are required in order to evaluate the error function $\mathbf{e}_{2^1/_2\text{-D}}$. The last metric is used to control the rate of translational convergence of the camera and it is required in the sub-matrix $\mathbf{L}_{\mathbf{v}}$ of the interaction matrix $\mathbf{L}_{2^1/_2\text{-D}}$. These parameters can be recovered from two images if the homography $\mathbf{H}$ for a common virtual plane π is available.

Obtaining an accurate estimate of $\mathbf{H}$ is not trivial for images of non-planar objects, where a virtual plane is defined by three feature points, and additional feature points that do not necessarily lie on the virtual plane

are used to recover the homography relationship. The interested reader is directed to Appendix A for a discussion on the homography estimation method implemented in the simulations and experiments.

Once $\mathbf{H}$ is determined, it is decomposed to recover the Euclidean parameters that are used in the visual servoing control law. The interested reader is directed to Appendix B for a discussion on the homography decomposition methods implemented in the simulations and experiments. The decomposition solution has the following form, where up to 8 possible solutions exist:

$$\mathbf{H} = d^*({}^c\mathbf{R}_{c^*}) + ({}^c\mathbf{t}_{c^*}){\mathbf{n}^*}^T. \tag{3.16}$$

where

- ${}^c\mathbf{R}_{c^*}$ is the rotation matrix from frame $\mathcal{F}_{c^*}$ to $\mathcal{F}_c$,
- $\mathbf{n}^*$ is the unit vector normal to π expressed in $\mathcal{F}_{c^*}$,
- ${}^c\mathbf{t}_{c^*}$ is the direction of translation from $\mathcal{F}_{c^*}$ to $\mathcal{F}_c$, and
- d^* is the signed distance from π to $\mathcal{F}_{c^*}$.

In Appendix B.3, a method is presented that uses visibility constraints to reduce the number of decomposition solutions from 8 to 2. A method that is specific for visual servoing, to determine which of the two remaining solutions is correct, is presented in Appendix B.4. Once ${}^c\mathbf{R}_{c^*}$ is extracted from the homography, θ and $\mathbf{r}$ can be computed to complete the last three components of the error function $\mathbf{e}_{2\frac{1}{2}\text{-D}}$. Since $\mathbf{H}$ is only defined up to a scale, the vector ${}^c\mathbf{t}_{c^*}$ only gives the direction of translation (but not its magnitude), so ${}^c\mathbf{t}_{c^*}$ cannot be used to directly control camera translation. The remaining parameters ρ_1 and $\log(Z/Z^*)$ that are required in the control law are determined from $\mathbf{H}$ as follows:

$$\rho_1 = \frac{Z}{d^*} = \frac{1}{d^*}\left(\frac{d}{\mathbf{n}^T\mathbf{m}}\right) = \frac{d}{d^*}\frac{1}{\mathbf{n}^T\mathbf{m}} = \det(\mathbf{H})\left(\frac{1}{\mathbf{n}^t\mathbf{m}}\right) \tag{3.17}$$

$$\log\left(\frac{Z}{Z^*}\right) = \log\left(\frac{d}{d^*}\frac{{\mathbf{n}^*}^T\mathbf{m}^*}{\mathbf{n}^T\mathbf{m}}\right) = \log\left(\det(\mathbf{H})\left(\frac{{\mathbf{n}^*}^T\mathbf{m}^*}{\mathbf{n}^T\mathbf{m}}\right)\right) \tag{3.18}$$

In addition to the direct use of these recovered parameters in the $2\frac{1}{2}$-D visual servoing control law, an accurate knowledge of ${}^c\mathbf{R}_{c^*}$, $\mathbf{n}^*$, ${}^c\mathbf{t}_{c^*}$ and d^* is useful for robot path planning, as discussed in Chapter 5.

3.3 Simulation Results

3.3.1 Purpose and Setup

The purpose of these simulations is to evaluate the image trajectories, the Cartesian trajectories and the robot trajectories that result from visual servoing using homography-based Euclidean reconstruction. These trajectories are compared against those obtained using IBVS to discern any improvements to the observed motion. Two different positioning tasks with significant displacements are considered. The first is a translational task that requires the robot end-effector to approach the target object with some out-of-plane rotations. The second is a pure rotational task of 180 degrees about the optical axis. The observed target is a non-planar object with 9 identifiable feature points. The target object is positioned above the robot in these experiments to increase the effective range of motion available to the robot for servoing, since robot joint limits are not explicitly managed in the servoing control law. The simulations are performed on a 6-DoF CRS-A465 robot with a Hitachi KP-D8 camera amounted on the end-effector to reflect the equipment available in the Collaborative Advanced Robotics and Intelligent Systems (CARIS) Lab at UBC.

3.3.2 Image-Based Visual Servoing

The error to be minimized in image-based visual servoing (IBVS) is the difference between current image feature location (u, v) and the desired image feature location (u^*, v^*), for n chosen feature points:

$$\mathbf{e}_{\text{IBVS}} = \begin{bmatrix} (u_1 - u_1^*) & (v_1 - v_1^*) & \cdots & (u_n - u_n^*) & (v_n - v_n^*) \end{bmatrix}. \tag{3.19}$$

A proportional control law is used to drive image coordinates exponentially towards their desired locations, with λ as the convergence rate:

$$\dot{\mathbf{e}}_{\text{IBVS}} = -\lambda \mathbf{e}_{\text{IBVS}}. \tag{3.20}$$

To achieve the above closed-loop behaviour, the control law for IBVS has the following form:

$$\dot{\mathbf{q}} = -\lambda \mathbf{J}^{+}_{\text{robot}}(\mathbf{q}) \mathbf{L}^{+}_{\text{IBVS}}(\mathbf{x}, \mathbf{y}, \mathbf{Z}) \mathbf{e}_{\text{IBVS}} \tag{3.21}$$

where $\mathbf{J}_{\text{robot}}(\mathbf{q})$ is the robot Jacobian corresponding to the eye-in-hand configuration, and $\mathbf{L}_{\text{IBVS}}$ is the image Jacobian of the feature points. For mul-

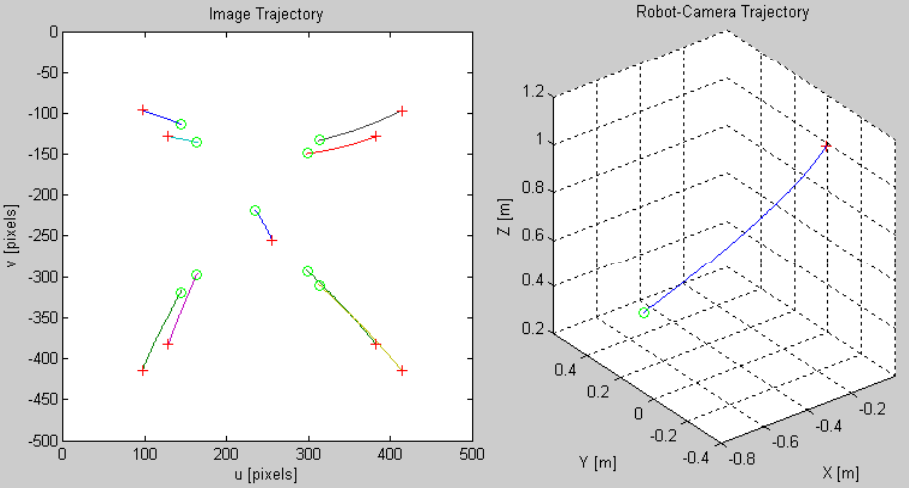


Figure 3.1: IBVS of a task requiring translations and out-of-plane rotations: image trajectory (left) and Cartesian trajectory (right).

tiple feature points, $\mathbf{L}_{\text{IBVS}}$ is a stacked matrix composed of the following [4]:

$$\mathbf{L}_{\text{IBVS}} = \begin{bmatrix} \mathbf{L}_1^T & \cdots & \mathbf{L}_n^T \end{bmatrix}^T \tag{3.22}$$

where

$$\mathbf{L}_i(x_i, y_i, Z_i) = \begin{bmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -(1+y_i^2) & y_i \\ 0 & -\frac{1}{Z_i} & \frac{y_i}{Z_i} & 1+x_i^2 & -x_i y_i & -x_i \end{bmatrix}. \tag{3.23}$$

The distance Z of a feature point to the image plane is obtained from the model of the object. The normalized coordinates (x, y) are calculated from (u, v) using the camera matrix from Equation 3.2.

Figure 3.1 shows the results from the first robot positioning task as described in Section 3.3.1 using IBVS for control. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. The image trajectories of the feature points are near-straight lines from the start-position to the end-position which keeps the target within the field of view. The Cartesian trajectory is also acceptable, due to the simplicity of the task, which mainly involved translations with some out-of-plane rotations. These results show that the trajectories generated from IBVS are sometimes acceptable for large motions.

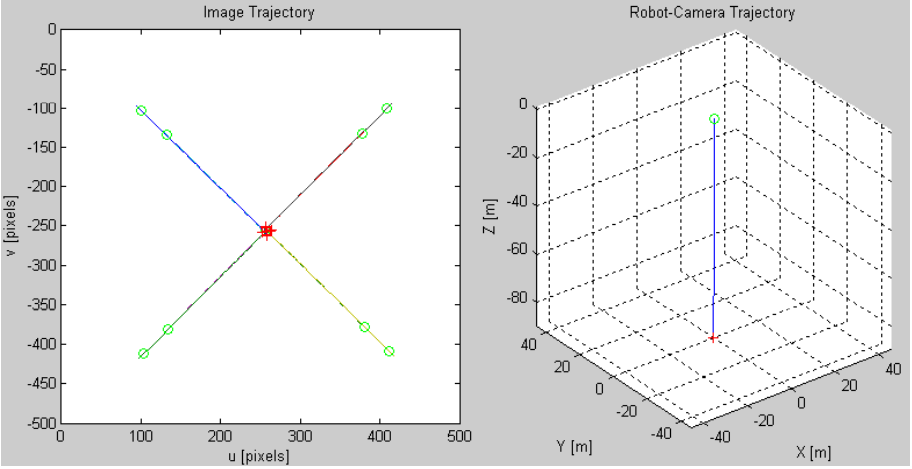


Figure 3.2: IBVS of a task requiring a rotation of 180 degrees about the optical axis: image trajectory (left) and Cartesian trajectory (right). The task is not completed successfully due to the camera retreat problem, which violates robot joint limits.

Figure 3.2 shows the results from the second robot positioning task as described in Section 3.3.1 using IBVS for control. The feature points approach their desired positions in straight-line trajectories. However, the visual servoing stops before the feature points reach their goal, and the positioning task does not converge. Examination of the Cartesian trajectory shows that under IBVS control, the camera is forced to retreat away from the target object until a joint limit is reached. These results demonstrate the problem of *camera retreat*, and illustrate why IBVS cannot be used for positioning tasks involving significant rotation about the optical axis.

3.3.3 $2^1/_2$-D Visual Servoing

The method of $2^1/_2$-D visual servoing is described in Sections 3.2, while the methods of homography estimation and decomposition are described in Appendix A and Appendix B, respectively. Figure 3.3 shows the results from the first robot positioning task as described in Section 3.3.1 using $2^1/_2$-D visual servoing for image feedback control. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. In $2^1/_2$-D visual servoing, the image trajectories of the feature points are more curved than the near-straight-line trajectories that are obtained using IBVS.

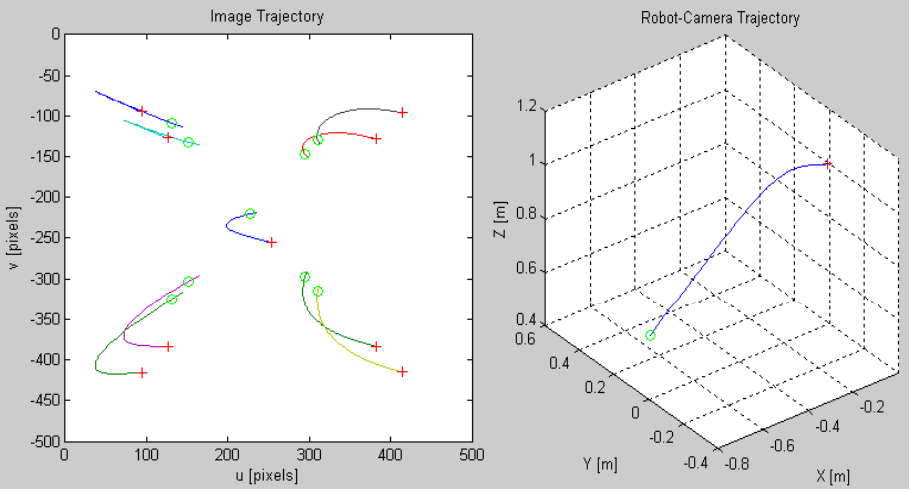


Figure 3.3: 2½-D visual servoing of a task requiring translations and out-of-plane rotations: image trajectory (left) and Cartesian trajectory (right).

This is due to the decoupling between image-based translation control and homography-based rotation control. The Cartesian trajectory is shown to be acceptable, as it does not deviate too far from a simple straight-line motion. The robot joint positions and velocities of the resulting motion are shown in Figure 3.4. They exhibit exponential convergence towards the desired robot configuration, at a rate that is proportionally to the control gain. These results show that while the path generated from 2½-D visual servoing is acceptable for large translational tasks with some out-of-plane rotations, its timing must be addressed through the use of a reference trajectory, since it is sub-optimal over large-range motions. The joint velocities decreases exponentially with the image error, resulting in high velocities at the start and negligible velocities towards the end.

Figure 3.5 shows the results from the second robot positioning task as described in Section 3.3.1 using 2½-D visual servoing for image feedback control. Unlike IBVS, which forces the camera to retreat (resulting in the violation of joint limits), 2½-D control is able to correctly recognize that a 180-degree rotation is required, using the homography-based methods outlined in Appendix A and Appendix B. The Cartesan trajectories show that 2½-D visual servo control executes the required rotational motion without requiring any extra translational motion. This property of decoupled control illustrates one of the many benefits of using partial pose information

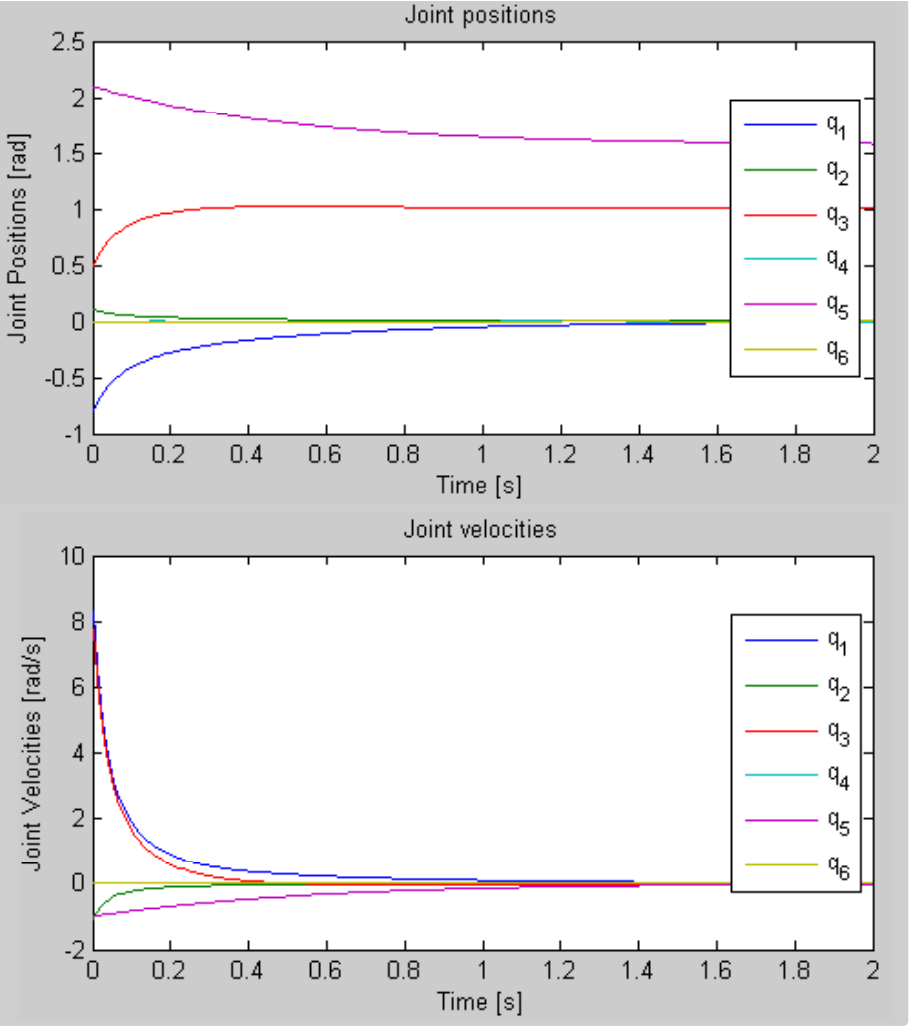


Figure 3.4: 2½-D visual servoing of a task requiring translations and out-of-plane rotations: robot joint positions (top) and robot joint velocities (bottom).

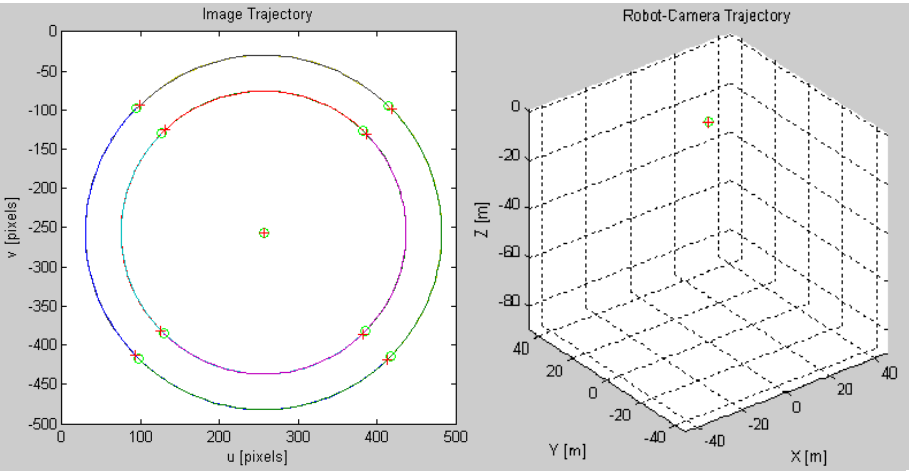


Figure 3.5: $2^1/_2$-D visual servoing of a task requiring a rotation of 180 degrees about the optical axis: image trajectory (left) and Cartesian trajectory (right).

to supplement visual servoing in order to simplify motion. An additional advantage of homography estimation and decomposition is that it does not require a 3-D model of the observed target object.

3.4 Experimental Results

3.4.1 Purpose and Setup

This section outlines a method for evaluating the accuracy of homography estimation and decomposition from image observations of real target objects. This step is important for two reasons. First, the accuracy of the partial pose estimation from the image homography affects the convergence rate of all visual servoing methods that use these parameters as part of their control schemes. Secondly, the recovered Euclidean parameters must be sufficiently accurate if they are to be used for path planning for robot joint-limit avoidance and camera limit avoidance, as proposed by several authors [18] [30] [31] [32].

Experiments are performed using real images obtained from an eye-in-hand camera to evaluate the accuracy of the scaled Euclidean parameters $({}^c\mathbf{R}_{c^*}, {}^c\mathbf{t}_{c^*})$ recovered from the homography $\mathbf{H}$ of a virtual plane. In these

experiments, a Hitachi KP-D8 miniature CCD camera is attached the end of a CRS-A465 robot arm to observe a non-planar target object. As the images are captured, the corresponding robot joint encoder measurements are also recorded so that the change in camera position and orientation are known. The scaled Euclidean parameters $({}^{c}\mathbf{R}_{c^*}, {}^{c}\mathbf{t}_{c^*})$ that are recovered from vision are compared against the ground truth provided by the kinematics of the robot.

3.4.2 Extracting and Matching Image Features

The Scale Invariant Feature Transform (SIFT) [48] is used to characterize the target object into a set of features points. These SIFT feature points are used to establish point-to-point correspondences between subsequent images. SIFT is particularly well-suited to this task, since it generates well-localized features that work well on textured objects. An added advantage is that they are invariant to scale changes and are somewhat invariant to small changes in viewpoint, so that the camera is not restricted to move in a planar 2-D fashion. Features between two images are matched using the nearest-neighbor algorithm and outliers are removed using semi-local constraints. For each matched candidate, K out of its N nearest neighbors must agree with its scale, orientation, and location in order for it to be considered a valid match. $K = 8$ is chosen because the subsequent estimation of the homography requires at least eight matched points. $N = 16$ is chosen experimentally to give good detection rates while minimizing computation times. The thresholds for scale, orientation, and location consistency are chosen as follows:

$$\frac{2}{3} \leq \frac{\left(\frac{Scale_{Key}}{Scale_{KeyMatch}}\right)}{\left(\frac{Scale_{Neighbor}}{Scale_{NeighborMatch}}\right)} \leq \frac{3}{2}, \tag{3.24}$$

$$-30^\circ \leq \text{diff}\begin{pmatrix} \text{diff}(Ori_{key}, Ori_{KeyMatch}), \\ \text{diff}(Ori_{Neighbor}, Ori_{NeighborMatch}) \end{pmatrix} \leq 30^\circ, \tag{3.25}$$

$$\frac{2}{3} \leq \text{Scale}\frac{(Distance_{match\&neighbor})}{(Distance_{key\&neighbor})} \leq \frac{3}{2}. \tag{3.26}$$

These parameters were experimentally tested with images and shown to provide good outlier removal.

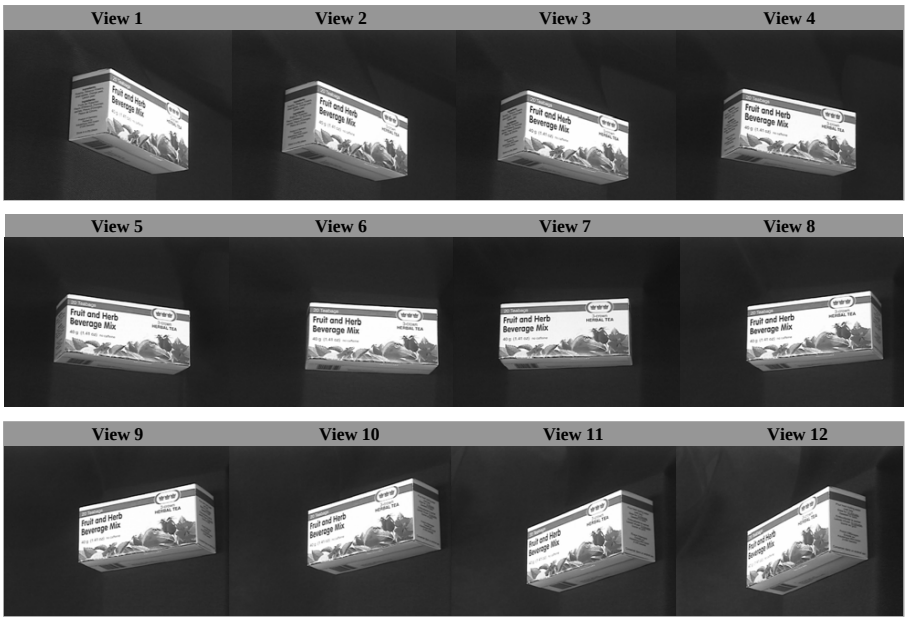


Figure 3.6: Sequence of images used to assess the accuracy of homography-based methods for visual servoing.

3.4.3 Error Assessment

The method of estimating ${}^{c}\mathbf{R}_{c^*}$, ${}^{c}\mathbf{t}_{c^*}$ from the homography of a virtual plane is tested on several household and industrial 3-D objects such as coffee mugs, key chucks, motors, and computer cooling fans. The most comprehensive results are obtained from a series of twelve images of a tea box taken at approximately 10- to 15-degrees increments of out-of-plane 3-D rotations. This sequence of images is shown in 3.6 for visual reference. Although the object has several planar surfaces, no effort is spent on ensuring that all matched features belong to a common plane. In fact, many of the SIFT features matched between the images belong to different faces on the tea box. The advantage of homography estimation based on the virtual parallax is that it applies to both planar and non-planar objects. Regardless of the structure of the object, three points are automatically selected to define a reference plane which maximizes the surface area of corresponding triangles in both images.

Figure 3.7 shows the number of matched SIFT features for each image

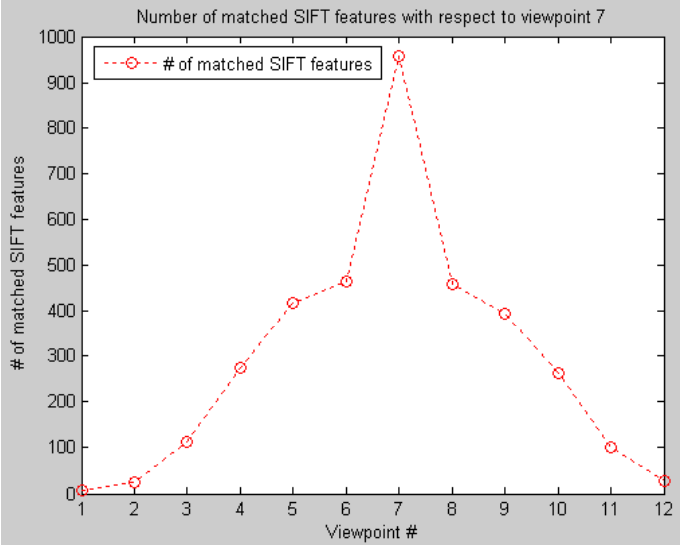


Figure 3.7: Number of feature matches available for non-planar homography estimation for the image sequence shown.

pair with viewpoint 7 as the reference. The number of matched SIFT features between two images decreases quickly as the camera deviates from the reference viewpoint. For out-of-plane rotations greater than 25 degrees, it is difficult to obtain the required number of matching features to accurately estimate $\mathbf{H}$. In fact, because only six SIFT features are matched between viewpoint 1 and viewpoint 7, in this case, the rotational and translational parameters cannot be recovered. The quality of the matched SIFT features also decreases in the sense that the locations of the matches are less accurate. Since SIFT features are not truly affine invariant, distortions due to changes in viewpoints (even for planar features) will cause the location of the SIFT features to shift, resulting in an incorrect estimate of $\mathbf{H}$. This is problematic since in the virtual plane homography estimation method [23] since, regardless of the number of matched features available for homography estimation, only three feature points are used to define the virtual plane.

Figure 3.8 show the magnitude of errors in the estimated rotation ${}^{c}\widehat{\mathbf{R}}_{c^*}$ for the image sequence shown in Figure 3.6. The magnitude of the actual rotation ${}^{c}\mathbf{R}_{c^*}$ corresponding to each viewpoint is also shown for comparison. Viewpoint 7 is used as the reference image for establishing the canonical

frame of reference. The rotation error is defined as the shortest length of the geodesic starting at ${}^{c}\mathbf{R}_{c^*}$ and ending at ${}^{c}\widehat{\mathbf{R}}_{c^*}$, which corresponds to the rotation angle θ_{error} of the rotation matrix ${}^{c}\mathbf{R}_{c^*}{}^{c}\widehat{\mathbf{R}}_{c^*}^{-1}$. Note that the rotation errors *cannot* be simply subtracted from the actual camera rotation to obtain the rotation estimate, since their respective axes of rotation do not align. Finite 3-D rotations are, in general, not vectors and therefore cannot be summed up vectorially. It is found that the magnitude of errors in ${}^{c}\widehat{\mathbf{R}}_{c^*}$ increases quickly as the camera deviates from the reference viewpoint. For small camera motions throughout viewpoints 4-9, the rotation estimation errors are less than 5°. The largest error is 15° for viewpoint 12, which also has very few matched features.

Figure 3.9 show the magnitude of errors in the estimated translation ${}^{c}\widehat{\mathbf{t}}_{c^*}$ for the image sequence shown in 3.6. The magnitude of the actual translation ${}^{c}\mathbf{t}_{c^*}$ corresponding to each viewpoint is also shown for comparison. Viewpoint 7 is used as the reference image for establishing the canonical frame of reference. Since the recovered translation is valid only in direction (not magnitude), the translational error is defined as the angle θ_t between the normalized vectors ${}^{c}\mathbf{t}_{c^*}/\|{}^{c}\mathbf{t}_{c^*}\|$ and ${}^{c}\widehat{\mathbf{t}}_{c^*}/\|{}^{c}\widehat{\mathbf{t}}_{c^*}\|$. It is found that errors in the direction of translation ($10° - 20°$) are generally greater than the errors in rotation (average $3° - 8°$). However, the magnitudes of the errors in ${}^{c}\widehat{\mathbf{t}}_{c^*}$ appear to settle at a constant level (20 degrees on the left, 10 degrees on the right) even as the camera deviates from the reference viewpoint, except for viewpoint 12, for which there is a low number of matched features.

One direction for future work is to investigate whether affine invariant image features would improve the estimate of $\mathbf{H}$ for large changes in camera viewpoint. If not, then multiple training images of the target object taken at 10- to 20-degree increments of rotation may be required to generate good estimates of $\mathbf{H}$ over large viewing angles. Another direction is to investigate whether Kalman filtering would improve the estimates of ${}^{c}\mathbf{R}_{c^*}$ and ${}^{c}\mathbf{t}_{c^*}$ by incorporating tracking through successive frames. The current implementation only uses the previous solution to ${}^{c}\mathbf{R}_{c^*}$ and ${}^{c}\mathbf{t}_{c^*}$ to help determine the correct solution to the decomposition of $\mathbf{H}$. It does not track the pose of the virtual plane, nor the rotation and translation parameters of the camera. It remains to be verified whether additional information provided by the Kalman filter would improve the robustness of the estimations in the presence of image noise.

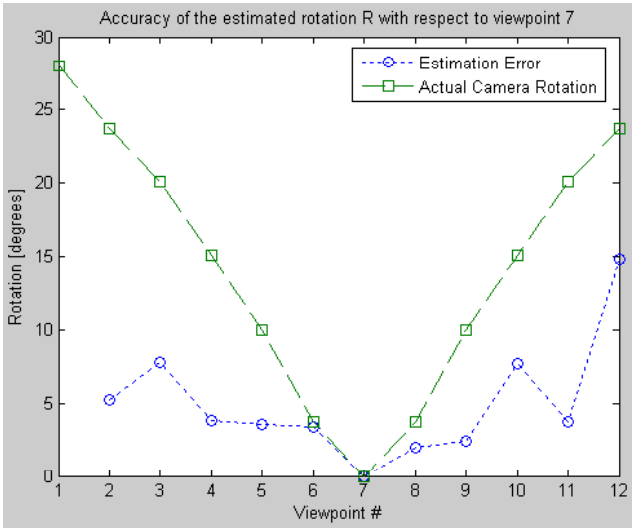


Figure 3.8: Rotational accuracy of homography decomposition for visual servoing.

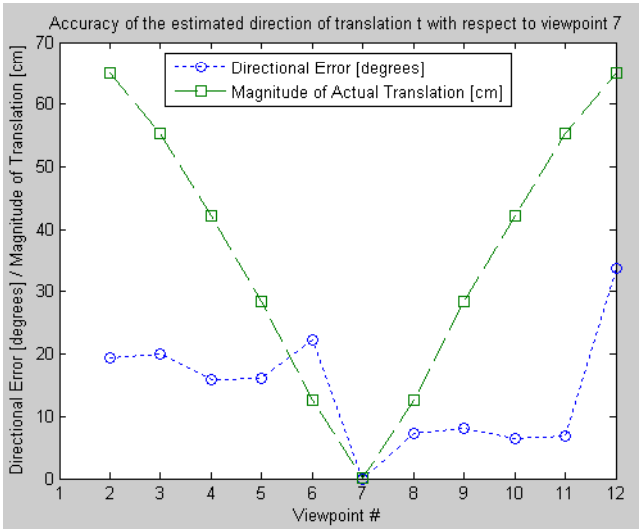


Figure 3.9: Translational accuracy of homography decomposition for visual servoing.

3.5 Summary

Experiments using an eye-in-hand camera validate homography-based Euclidean reconstruction as a useful method for recovering from images, rotational and scaled translation parameters that can be used to improve the predictability of visual servoed motion. The definition of a virtual plane formed by any three feature points on an object allows an image-to-image homography to be associated with any 3-D object, planar or non-planar. The virtual parallax method uniquely solves for that homography when 5 or more additional feature points are available. Analysis and simulations of visual servoing tasks in this chapter validate homography-based visual servoing as a useful method for positioning a robot to target objects without *a priori* models. The 2-D/3-D hybrid definition of the error metric in $2^1/_2$-D visual servoing results in an upper triangular Jacobian so that camera translation and rotation can be decoupled in the control law, resulting in efficient camera motion, as shown in simulation reults.

However, experimental results using images also show the limits of using homography decomposition for open-loop path planning, especially when precise Cartesian or joint-space trajectories are required for obstacle avoidance. The accuracy of the rotation and translation estimated from a homography tends to suffer when the camera displacement is large due to: i) an insufficient number of matched feature points between very different viewpoints; and ii) the sensitivity of homography decomposition to image noise and to real-image deviations from projections produced by the classical pinhole-camera model. Regardless of the number of matched features available for homography estimation, only three feature points are used by the homography estimation to define the virtual plane. The reliance of the virtual parallax method on these three chosen feature points explains why the recovered parameters are so sensitive to image noise. The next chapter discusses an improved method of pose estimation, using a model of the target object with an extended Kalman filter (EKF) to obtain pose estimates that are more robust against image noise and modelling errors.

Chapter 4

Robust Pose Estimation

4.1 Introduction

In the previous chapter, it was shown that the partial recovery of Euclidean parameters from images could be used to significantly improve the performance of visually servoed motion. Specifically, knowledge of the required rotation and the direction of translation (recovered from a homography) allowed the visual servoing controller to execute the positioning task in a near straight-line motion in Cartesian space, resulting in efficient camera motions. Not only did this approach minimize the chance of exceeding robot joint limits during servoing, it improved the predictability of the servoed motion and avoided the problem of camera retreat. However, the rotation and translation parameters recovered from homography estimation and homography decomposition were found to be extremely sensitive to image noise. One reason for this sensitivity is that the partial pose estimates were obtained without tracking and that the estimation was executed without any assumptions regarding the structure of the observed scene (except that a virtual plane could be formed by any three feature points).

This chapter discusses the use of an *extended Kalman filter* (EKF), in conjunction with a model of the target object, to improve the robustness of the pose estimates against image noise and model disturbances. The EKF is an nonlinear extension of the classical Kalman filter, which is a linear optimal estimator. The EKF directly estimates the states of a nonlinear system using a bootstrap method. The nonlinear system is linearized around the current Kalman filter estimate, while the Kalman filter estimate of the nominal trajectory is, in turn, based on the linearized system. A nonlinear observer is required for pose tracking, due to the nonlinearity in the camera photogrammetric equations and in the parametric representations of 3-D rotations. An added benefit of the EKF is that it can be implemented to track the velocity of the target object, thus enabling the use of position-based visual servoing (PBVS) with respect to a moving object.

The focus of the chapter is the control of an eye-in-hand robot using the PBVS approach. The EKF is used to estimate the pose of a (potentially)

moving target object from images, while a state-feedback controller is used to drive the eye-in-hand robot to a desired pose with respect to the object. Analysis and synthesis of the observer and the controller is performed within the framework of state-space control. The PBVS controller is described in the first section, while the observer is derived in the second section. Following, the performance of PBVS is compared against that of standard methods using a number of visual servoing tasks. A section is devoted to the discussion of the transient performance of the EKF in the presence of image noise and modelling errors, and its effect on the PBVS controller.

A well-known problem encountered in PBVS is that the position-controlled camera trajectory may cause the target object to exit the field of view during servoing, resulting in pose estimation failure and servo failure. A method using adaptive control gains for PBVS (borrowed from hybrid servoing [27]) is proposed in this chapter to address the field-of-view problem. This method exploits the decoupled control of camera translation and rotation available in PBVS to ensure that the target stays within the camera's field of view, under the assumption of a sufficiently fast image frame rate.

Another common problem that is encountered in PBVS is that full controllability of the camera is lost when its position is at a robot singularity. Near the singularity, the required servoing motion may also result in large joint velocities. A practical method of restoring the controllability of visual servoing methods near robot singularities is discussed in this chapter.

4.2 Controller Design

4.2.1 Plant Model

In position-based visual servoing (PBVS), the 6-dimensional state vector is an error function, $\mathbf{e}_{\text{PBVS}}(t)$, consisting of the following Euclidean parameters:

$$\mathbf{e}_{\text{PBVS}}(t) = \left[{}^{c^*}\mathbf{t}_c, \theta\mathbf{r}\right]^T \tag{4.1}$$

where

- ${}^{c^*}\mathbf{t}_c$ are the x-y-z coordinates of the current camera frame expressed in the desired camera frame.
- θ and $\mathbf{r}$ are, respectively, the angle and axis of rotation extracted from ${}^{c^*}\mathbf{R}_c$, the rotation matrix that expresses the coordinates of the current camera frame in the coordinates in the desired camera frame.

When the norm of the error function, $\mathbf{e}_{\text{PBVS}}(t)$, is zero, the current camera frame is aligned with the desired camera frame. Since the translation and orientation errors between the current and the desired camera frame do not naturally correct themselves, the *autonomous system* has the following form:

$$\dot{\mathbf{e}}_{\text{PBVS}}(t) = \mathbf{A}_{\text{control}}\mathbf{e}_{\text{PBVS}}(t), \tag{4.2}$$

where

$$\mathbf{A}_{\text{control}} = \mathbf{0}_{6\times 6}. \tag{4.3}$$

The inputs to the system are the rate of change of the joint angles controlled by the motors of the 6-DoF eye-in-hand robot:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 & \dot{q}_5 & \dot{q}_6 \end{bmatrix}^T. \tag{4.4}$$

Most vision-based control approaches consider only robot kinematics, since the bandwidth for visual control is usually limited by the camera frame rate (in Hertz) and image processing times (in milliseconds). Robot dynamics are generally taken into account with a low-level PID controller using joint encoder feedback.

To solve for the state differential equations, it is necessary to derive the relationship between the input, $\dot{\mathbf{q}}$, and the rate of change of the error function $\dot{\mathbf{e}}_{\text{PBVS}}(t)$. At the position level, the error space, the camera space and the robot space are all related by nonlinear mappings that are neither *surjective* nor *injective*. Fortunately, the controller is designed at the velocity level where they are related by linear Jacobian matrices that are functions of only the positional state variables. The velocity of the camera has the following three translational and three rotational components:

$$\mathbf{v}_c = \begin{bmatrix} {}_x v_c & {}_y v_c & {}_z v_c & {}_x w_c & {}_y w_c & {}_z w_c \end{bmatrix}^T. \tag{4.5}$$

The velocity of the camera $\mathbf{v}_c$ is related to the velocity of the joint angles $\dot{\mathbf{q}}$ by the robot-manipulator Jacobian:

$$\begin{bmatrix} \mathbf{v}_c \end{bmatrix} = \mathbf{J}_{\text{robot}}|_{\mathbf{q}} \begin{bmatrix} \dot{\mathbf{q}} \end{bmatrix}. \tag{4.6}$$

The exact form of $\mathbf{J}_{\text{robot}}$ is a function of the kinematic structure of the eye-in-hand robot. Methods for determining $\mathbf{J}_{\text{robot}}$ can be found in texts such as [49]. The rate of change of the error function $\dot{\mathbf{e}}_{\text{PBVS}}(t)$ is related to the velocity of the camera $\mathbf{v}_c$ by the interaction matrix $\mathbf{L}_{\text{PBVS}}$:

$$\begin{bmatrix} \dot{\mathbf{e}}_{\text{PBVS}} \end{bmatrix} = \mathbf{L}_{\text{PBVS}}|_{\mathbf{e}} \begin{bmatrix} \mathbf{v}_c \end{bmatrix}. \tag{4.7}$$

For this definition of $\mathbf{e}_{\text{PBVS}}(t)$, the interaction matrix has a simple decoupled form between translational and rotational motions [13]:

$$\mathbf{L}_{\text{PBVS}} = \begin{bmatrix} {}^{c^*}\mathbf{R}_c & 0 \\ 0 & \mathbf{L}_{\theta\mathbf{r}} \end{bmatrix}, \tag{4.8}$$

where

$$\mathbf{L}_{\theta\mathbf{r}} = I_{3\times 3} - \frac{\theta}{2}\left[\mathbf{r}\right]_\times + \left(1 - \frac{\text{sinc}(\theta)}{\text{sinc}^2(\frac{\theta}{2})}\right)\left[\mathbf{r}\right]_\times^2. \tag{4.9}$$

Substituting for $\dot{\mathbf{e}}_{\text{PBVS}}(t)$ in terms of $\dot{\mathbf{q}}$ using the Jacobian relationships, the state differential equation for the *non-autonomous system* is obtained as follows:

$$\dot{\mathbf{e}}_{\text{PBVS}}(t) = \mathbf{A}_{\text{control}}\mathbf{e}_{\text{PBVS}}(t) + \mathbf{B}_{\text{control}}\mathbf{F}_{\text{control}}\dot{\mathbf{q}}(t), \tag{4.10}$$

where

$$\mathbf{A}_{\text{control}} = \mathbf{0}_{6\times 6}, \tag{4.11}$$

$$\mathbf{B}_{\text{control}} = \mathbf{I}_{6\times 6}, \tag{4.12}$$

$$\mathbf{F}_{\text{control}} = \left[\mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}\right]\left[\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\right]. \tag{4.13}$$

4.2.2 Analysis of Controllability

The system is completely controllable if, and only if,

$$\begin{bmatrix}\lambda\mathbf{I}_{6\times 6} - \mathbf{A}_{\text{control}} & \mathbf{B}_{\text{control}}\mathbf{F}_{\text{control}}\end{bmatrix} \tag{4.14}$$

has rank $n = 6$. Since the eigenvalues of $\mathbf{A}_{\text{control}}$ are all zero, this matrix evaluates to:

$$\begin{bmatrix}\mathbf{0}_{6\times 6} & \mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\end{bmatrix}. \tag{4.15}$$

Thus, $\mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}$ and $\mathbf{J}_{\text{robot}}|_{\mathbf{q}}$ are required to have full rank in order for the system to be completely controllable. Due to the nature of the decoupled form of $\mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}$ and the definition of the rotation matrix ${}^{c^*}\mathbf{R}_c$ and its constituents $\theta\mathbf{r}$, $\mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}$ has no singularities for the entire error space. In fact, the inverse of $\mathbf{L}_{\text{PBVS}}|_{\mathbf{e}}$ always exists and is constant:

$$\mathbf{L}_{\text{PBVS}}^{-1} = \begin{bmatrix} {}^{c^*}\mathbf{R}_c^T & 0 \\ 0 & \mathbf{I}_{3\times 3} \end{bmatrix}. \tag{4.16}$$

However, the same cannot be said for the robot-manipulator Jacobian. For example, when the configuration of the robot is such that two rotational

axes (or two translational axes) are collinear, there is redundant motion along the collinear direction. Since the robot only has 6 DoF in total, a DoF must be lost in another direction, reducing the rank of the Jacobian. The Jacobian relationship is also undefined for camera poses that are outside the dexterous workspace of the robot. Therefore, the system is completely controllable if and only if the commanded camera pose is not at a manipulator singularity (i.e. $\mathbf{J}_{\text{robot}}|_{\mathbf{q}}$ has full rank) and does not exceed the robot's joint limits or reach. A method to address this partial loss of controllability is discussed in Appendix C.

4.2.3 State-Feedback Controller with Adaptive Gains

The purpose of PBVS is to introduce state feedback to ensure exponential convergence towards zero of each component in the error function:

$$\dot{\mathbf{e}}_{\text{PBVS}} = -\lambda \mathbf{e}_{\text{PBVS}}, \tag{4.17}$$

where $\lambda > 0$ is the convergence rate. The input to the system from the state-feedback controller is chosen to achieve the closed-loop behaviour defined by Equation 4.17 for the open-loop system defined by Equation 4.10. Combining the two equation results, the required input from the state-feedback controller is:

$$\dot{\mathbf{q}} = -\lambda \left[\mathbf{J}^{-1}_{\text{robot}}|_{\mathbf{q}}\right] \left[\mathbf{L}^{-1}_{\text{PBVS}}|_{\mathbf{e}}\right] \mathbf{e}_{\text{PBVS}}(t) \tag{4.18}$$

From Equation 4.17, $-\lambda$ corresponds to the desired eigenvalues of the system, which must be in the open left-half plane to achieve asymptotic stability. Due to visual constraints, however, a constant value for λ is typically insufficient over the entire course of visual servoing. When the initial camera pose differs significantly from the desired camera pose, the target object has a large chance of leaving the camera's field of view during the servoing trajectory, leading to a discontinuity and subsequent failure in the control law. An adaptive control law is implemented here to address this issue:

$$\dot{\mathbf{e}}_{\text{PBVS}} = -\lambda \mathbf{D}_{\text{adaptive}} \mathbf{e}_{\text{PBVS}} \tag{4.19}$$

where $\mathbf{D}_{\text{adaptive}}$ is a positive diagonal gains matrix of the following form:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & f_u(u) & 0 & 0 \\ 0 & 0 & f_u(u)f_v(v) & 0 \\ 0 & 0 & 0 & \mathbf{I}f_u(u)f_v(v) \end{bmatrix}. \tag{4.20}$$

Each diagonal element of $\mathbf{D}_{\text{adaptive}}$ is a function of the pixel coordinates (u, v) of the object's center. The bell-curve function $f(x)$ is defined as:

$$f(x) = \begin{cases} \exp\left(-\frac{\left(x - \frac{1}{2}[x^{\min} + x^{\max}]\right)^{2n}}{(x - x^{\min})^m (x^{\max} - x)^m} \right) & \text{if } x^{\min} < x < x^{\max} \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

with $0 \leq f(x) \leq 1$ and two parameters m and n are used to design the form of the bell-curve. The parameters $u^{\min}$, $u^{\max}$, $v^{\min}$, $v^{\max}$ correspond to the size of the CCD array of the camera.

Combining Equation 4.19 with Equation 4.10, the input to the system from the state-feedback controller *with adaptive gains* has the following form:

$$\dot{\mathbf{q}} = -\lambda \left[\mathbf{J}^{-1}_{\text{robot}}|_{\mathbf{q}}\right] \left[\mathbf{L}^{-1}_{\text{PBVS}}|_{\mathbf{e}}\right] \left[\mathbf{D}_{\text{adaptive}}\right] \mathbf{e}_{\text{PBVS}}(t) \tag{4.22}$$

4.2.4 Camera Field-of-View Constraints

The elements of $\mathbf{D}_{\text{adaptive}}$ are used to establish hierarchical control for each component in the error function. They act as a band-pass filter, ensuring that the image coordinates (u, v) of the target frame are well away from the periphery of the image before large changes in depth and orientation are allowed to occur. In other words, if at any point in time during servoing the target object is about to leave the field of view, then the degrees of freedom corresponding to camera rotation and approach in the z direction are temporarily frozen, so that $x - y$ translational correction can bring the object back towards the center of the image.

The resulting system using $\mathbf{D}_{\text{adaptive}}$ is stable in the sense of Lyapunov for a sufficiently fast camera frame rate. $0 \leq f(x) \leq 1$ does not affect the sign of the original eigenvalues. The largest Jordan block is of size 1, so even if more than one eigenvalue is set to zero, the system will not become unstable. At worst, all gains may be severely attenuated and visual servoing may be brought to a halt. The transient performance of the control may suffer due to the adaptive gains, but at least it will not become unstable due to the object exiting the field of view.

4.3 Observer Design

4.3.1 Plant Model

Given the decoupled form of the control law in Equation 4.22, the challenge of PBVS is in obtaining an accurate estimate of the target pose. Even when

the complete calibration of the camera is given and an accurate model of target object is known, the pose estimate can still be very sensitive to image noise in the components related to out-of-plane translations and rotations. Since the image measurements are only available on a frame-by-frame basis, a discrete-time state-space dynamic model is proposed to track the pose over successive image frames to increase the robustness of the pose estimates to image noise and disturbances.

The relative pose of the target object is affected independently by the motion of the camera and by the motion of the target object. Although the motion characteristic of the target object is generally unknown, the camera self-motion can be calculated from robot kinematics and encoder measurements. Therefore, only 12 states are required to represent the relative object pose and the motion of the target object:

$$\mathbf{x}_{\text{obsv}} = \begin{bmatrix} {}^{c}\mathbf{w}_o & \mathbf{v}_o \end{bmatrix}^T \tag{4.23}$$

where ${}^{c}\mathbf{w}_o$ is the *relative* pose of the object with respect to the camera, parameterized as x-y-z translation and roll-pitch-yaw orientation angles:

$${}^{c}\mathbf{w}_o = \begin{bmatrix} {}^{c}X_o & {}^{c}Y_o & {}^{c}Z_o & {}^{c}\phi_o & {}^{c}\alpha_o & {}^{c}\psi_o \end{bmatrix}^T. \tag{4.24}$$

The second part of the state vector consists of $\mathbf{v}_o$, the velocity of the target object with respect to a *global* reference frame. It is expressed in terms of its linear and angular velocity in the x-y-z directions:

$$\mathbf{v}_o = \begin{bmatrix} {}_{x}v_o & {}_{y}v_o & {}_{z}v_o & {}_{x}w_o & {}_{y}w_o & {}_{z}w_o \end{bmatrix}^T. \tag{4.25}$$

The velocity of the camera $\mathbf{v}_c$ is modelled as an input into the system, since the camera self-motion can be determined with high certainty from the robot kinematics and the joint encoder measurements:

$$\mathbf{u}_{\text{obsv}} = \mathbf{v}_c = \begin{bmatrix} {}_{x}v_c & {}_{y}v_c & {}_{z}v_c & {}_{x}w_c & {}_{y}w_c & {}_{z}w_c \end{bmatrix}^T. \tag{4.26}$$

Note that $\mathbf{v}_o$ and $\mathbf{v}_c$ have the same form. From linear superposition, they can be subtracted from one another to obtain relative velocities:

$${}^{c}\mathbf{v}_o = \begin{bmatrix} \mathbf{v}_o \end{bmatrix} - \begin{bmatrix} \mathbf{v}_c \end{bmatrix}. \tag{4.27}$$

During visual servoing, the measurement outputs used for pose estimation are the image pixel coordinates of the feature points of the object that

are visible in the camera's field of view. If n feature points are being used, then the output vector is:

$$\mathbf{z}_{\text{obsv}} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & \cdots & u_n & v_n \end{bmatrix}^T . \tag{4.28}$$

The state difference equations and output equations are derived in the next section based on these definitions of states, inputs and outputs.

4.3.2 State Difference Equations

The relative pose $({}^c\mathbf{w}_o)_k$ of the object with respect to the camera at iteration k is a function of its pose at iteration $k-1$ and its higher order derivatives. Using a Taylor series expansion and a sampling period δ_t, a constant velocity model is chosen where derivatives of 2^{nd} order or higher are approximated as a zero-mean Gaussian disturbance vector ${}_1\boldsymbol{\gamma}_{k-1}$:

$$(({}^c\mathbf{w}_o)_k) \approx (({}^c\mathbf{w}_o)_{k-1}) + \delta_t(({}^c\dot{\mathbf{w}}_o)_{k-1}) + {}_1\boldsymbol{\gamma}_{k-1}. \tag{4.29}$$

The above approximation is valid under the assumption that both the object and the camera have smooth motions and experience low accelerations. It is assumed that most accelerations remain near zero and that the probability of high accelerations fall off with a Gaussian profile. This model generally results in dynamic modelling errors when there are large changes in the relative velocity. These errors are represented by disturbance inputs ${}_1\boldsymbol{\gamma}_{k-1}$ to the state equations.

The following Jacobian relationship exists between the change of the pose parameters $({}^c\dot{\mathbf{w}}_o)_{k-1}$ and the relative velocity of the object with respect to the camera :

$$({}^c\dot{\mathbf{w}}_o)_{k-1} = \left(\mathbf{J}^{-1}_{\phi\alpha\psi}|_{({}^c\mathbf{w}_o)_{k-1}}\right)({}^c\mathbf{v}_o)_{k-1} \tag{4.30}$$

$$\begin{bmatrix} {}^c\dot{X}_o \\ {}^c\dot{Y}_o \\ {}^c\dot{Z}_o \\ {}^c\dot{\phi}_o \\ {}^c\dot{\alpha}_o \\ {}^c\dot{\psi}_o \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\phi\frac{\sin\alpha}{\cos\alpha} & \sin\phi\frac{\sin\alpha}{\cos\alpha} & 1 \\ 0 & 0 & 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & \frac{\cos\phi}{\cos\alpha} & \frac{\sin\phi}{\cos\alpha} & 0 \end{bmatrix} \begin{bmatrix} {}^c_x v_o \\ {}^c_y v_o \\ {}^c_z v_o \\ {}^c_x w_o \\ {}^c_y w_o \\ {}^c_z w_o \end{bmatrix} . \tag{4.31}$$

The Jacobian is nonlinear in terms of the state variables, and its inverse is undefined for $\alpha = \pm\frac{\pi}{2}$, known as the *gimbal lock* configuration. This issue is further addressed Section 4.3.5. Using the Jacobian relationship, the state

difference equation can be expressed in terms of the states $(\mathbf{x}_{\text{obsv}})_{k-1}$, inputs $(\mathbf{u}_{\text{obsv}})_{k-1}$ and disturbance variables $\boldsymbol{\gamma}_{k-1}$:

$$(({}^c\mathbf{w}_o)_k) \approx (({}^c\mathbf{w}_o)_{k-1}) + \delta_t\Big(\mathbf{J}^{-1}_{\phi\alpha\psi}|_{({}^c\mathbf{w}_o)_{k-1}}\Big)(({}^c\mathbf{v}_o)_{k-1}) + {}_1\boldsymbol{\gamma}_{k-1} \tag{4.32}$$

$$= (({}^c\mathbf{w}_o)_{k-1}) + \delta_t\Big(\mathbf{J}^{-1}_{\phi\alpha\psi}|_{({}^c\mathbf{w}_o)_{k-1}}\Big)((\mathbf{v}_o)_{k-1} - (\mathbf{v}_c)_{k-1}) + {}_1\boldsymbol{\gamma}_{k-1}. \tag{4.33}$$

Similarly, the velocity of the object with respect to the global reference frame at iteration k is a function of its pose at iteration $k-1$ and its higher order derivatives. Using a Taylor series expansion and a sampling period δ_t, a constant velocity model is chosen where derivatives of 2^{nd} order or higher are approximated as a zero-mean Gaussian disturbance vector ${}_2\boldsymbol{\gamma}_{k-1}$. The second part of the state difference equations is as follows:

$$(\mathbf{v}_o)_k = (\mathbf{v}_o)_{k-1} + {}_2\boldsymbol{\gamma}_{k-1}. \tag{4.34}$$

4.3.3 Output Equations

To derive the output equations for pose estimation, a 3-D model of the target object is required. An object frame must be established so that its pose can be defined and so that the 3-D geometric relationships between its feature points can be described. Let $({}^oX_j, {}^oY_j, {}^oZ_j)$ be the coordinates of feature point j with respect to the object frame in the model. The coordinates $({}^oX_j, {}^oY_j, {}^oZ_j)$ of feature point j in the camera frame is a function of the relative target object pose ${}^c\mathbf{w}_o$ with respect to the camera:

$$\begin{bmatrix} {}^cX_j \\ {}^cY_j \\ {}^cZ_j \\ 1 \end{bmatrix} = \begin{bmatrix} c_\phi c_\alpha & c_\phi s_\alpha s_\psi - s_\phi c_\psi & c_\phi s_\alpha c_\psi + s_\phi s_\psi & {}^cX_o \\ s_\phi c_\alpha & s_\phi s_\alpha s_\psi + c_\phi c_\psi & s_\phi s_\alpha c_\psi - c_\phi s_\psi & {}^cY_o \\ -s_\alpha & c_\alpha s_\psi & c_\alpha c_\psi & {}^cZ_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^oX_j \\ {}^oY_j \\ {}^oZ_j \\ 1 \end{bmatrix}. \tag{4.35}$$

Given the camera calibration parameters, a predictive model for the pixel coordinates (u_j, v_j) corresponding to the projection of feature point j onto the image plane can be found:

$$u_j = \frac{f({}^cX_j)}{k_u({}^cZ_j)} + u_0; \quad v_j = \frac{f({}^cY_j)}{k_v({}^cZ_j)} + v_0 \tag{4.36}$$

where (u_0, v_0) are the images coordinates of the principal point, f is the focal length, and k_u and k_v are the number of pixels per unit distance in x and y, respectively.

Finally, an output model describing the relationship between the output measurement $(\mathbf{z}_{\text{obsv}})_k$ and the states $(\mathbf{x}_{\text{obsv}})_k$ is obtained, with the image noise $\boldsymbol{\eta}_k$ modeled with zero-mean Gaussian probability distributions:

$$\mathbf{z}_k = \mathbf{G}((\mathbf{x}_{\text{obsv}})_k) + \boldsymbol{\eta}_k, \tag{4.37}$$

where

$$\mathbf{G}(\mathbf{x}_{\text{obsv}}) = \begin{bmatrix} u_1 & v_1 & \cdots & u_n & v_n \end{bmatrix}^T, \tag{4.38}$$

and the subscript k denotes the k^{th} observation. Since the measurable pixel coordinates of the image features are nonlinear functions of the 6-DoF pose of the target object, a nonlinear observer is required. Since the measurements arrive frame-by-frame at fixed sampling intervals, the nonlinear observer must be implemented in discrete-time.

4.3.4 Observer Model Summary

In summary, the nonlinear discrete state-space model for pose estimation and tracking has the following parts:

States:	$(\mathbf{x}_{\text{obsv}})_k = \begin{bmatrix} ({}^c\mathbf{w}_o)_k & (\mathbf{v}_o)_k \end{bmatrix}$
Input:	$(\mathbf{u}_{\text{obsv}})_k = \begin{bmatrix} (\mathbf{v}_c)_k \end{bmatrix}$
Disturbance:	$\boldsymbol{\gamma}_k = \begin{bmatrix} {}_1\boldsymbol{\gamma}_k & {}_2\boldsymbol{\gamma}_k \end{bmatrix}^T$
Noise:	$\boldsymbol{\eta}_k = \begin{bmatrix} \boldsymbol{\eta}_k \end{bmatrix}^T$

Nonlinear State Difference Equations:

$$(\mathbf{x}_{\text{obsv}})_k = \mathbf{F}((\mathbf{x}_{\text{obsv}})_{k-1}, (\mathbf{u}_{\text{obsv}})_{k-1}) + \boldsymbol{\gamma}_{k-1} \tag{4.39}$$

$$= \begin{bmatrix} (({}^c\mathbf{w}_o)_{k-1}) + \delta_t \Big(\mathbf{J}^{-1}_{\phi\alpha\psi}|_{({}^c\mathbf{w}_o)_{k-1}} \Big) ((\mathbf{v}_o)_{k-1} - (\mathbf{v}_c)_{k-1}) \\ ((\mathbf{v}_o)_{k-1}) \end{bmatrix} + \begin{bmatrix} {}_1\boldsymbol{\gamma}_{k-1} \\ {}_2\boldsymbol{\gamma}_{k-1} \end{bmatrix} \tag{4.40}$$

Nonlinear Output Equations:

$$(\mathbf{z}_{\text{obsv}})_k = \mathbf{G}((\mathbf{x}_{\text{obsv}})_k) + \boldsymbol{\eta}_k. \tag{4.41}$$

4.3.5 Analysis of Observability

It is difficult to obtain a closed form analytic solution for the conditions that guarantee complete observability for this nonlinear discrete system. Nonetheless, conditions for which it is *not* completely observable can be derived from the nonlinear difference equations. Specifically, the system will not be fully observable when $\alpha = \pm\frac{\pi}{2}$, since the Jacobian $\left(\mathbf{J}^{-1}_{\phi\alpha\psi}|_{({}^c\mathbf{w}_o)_{k-1}}\right)$ loses rank and its inverse does not exist. Physically, when $\alpha = \pm\frac{\pi}{2}$, the axis of rotation of ϕ and the axis of rotation of ψ are collinear. From the image output point of view, the two orientation states are indistinguishable from each other. Therefore, not all states are completely observable in this singular configuration. To retain observability near singularities, a damped-least squares inverse [50] of the Jacobian is used, similar in form to the method of restoring controllability (Appendix C).

4.3.6 Extended Kalman Pose Estimation

An extended Kalman filter is implemented to provide a near-optimal recursive update for the nonlinear discrete observer for the system presented in this section. The complete set of equations for the implementation of EKF is described in Appendix D for the reader's convenience. For this EKF pose estimation model, both the state difference equations $\mathbf{F}((\mathbf{x}_{\text{obsv}})_{k-1}, (\mathbf{u}_{\text{obsv}})_{k-1})$ and the output equations $\mathbf{G}((\mathbf{x}_{\text{obsv}})_k)$ must be linearized about the current state estimate at each iteration k.

4.4 Relationship Between Observer & Controller States

The states of the controller and the states of the observer are expressed using different parameterizations, each in their own reference frames to best suit its own development and analysis. Recall:

$$\begin{array}{ll} \text{Controller States:} & \mathbf{e}_{\text{PBVS}} = \begin{bmatrix} {}^{c^*}\mathbf{t}_c & \theta\mathbf{r} \end{bmatrix} \\ \text{Observer States:} & \mathbf{x}_{\text{obsv}} = \begin{bmatrix} {}^c\mathbf{w}_o & \mathbf{v}_o \end{bmatrix}^T \end{array}$$

The final step is to derive the relationship between the two sets of states. The reference signal for the controller is defined as ${}^c\mathbf{w}^*_o$, the desired relative pose of the object with respect to the camera frame:

$${}^c\mathbf{w}^*_o = \begin{bmatrix} {}^cX^*_o & {}^cY^*_o & {}^cZ^*_o & {}^c\phi^*_o & {}^c\alpha^*_o & {}^c\psi^*_o \end{bmatrix}^T. \tag{4.42}$$

The observed states can be converted into the states required by the controller using the following relationships:

$$ {}^{c^*}\mathbf{t}_c = \begin{bmatrix} {}^cX_o^* \\ {}^cY_o^* \\ {}^cZ_o^* \end{bmatrix} - \mathbf{R}_{\text{diff}} \begin{bmatrix} {}^cX_o \\ {}^cY_o \\ {}^cZ_o \end{bmatrix}, \tag{4.43} $$

$$ \theta\mathbf{r} = \frac{\cos^{-1}\left(\frac{1}{2}Tr(\mathbf{R}_{\text{diff}}) - 1\right)}{2\sin\left(\cos^{-1}\left(\frac{1}{2}Tr(\mathbf{R}_{\text{diff}}) - 1\right)\right)}, \tag{4.44} $$

where

$$ \mathbf{R}_{\text{diff}} = \mathbf{R}^{-1}({}^c\phi_o, {}^c\alpha_o, {}^c\phi_o)\mathbf{R}({}^c\phi_o^*, {}^c\alpha_o^*, {}^c\phi_o^*) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{4.45} $$

Note that $\mathbf{v}_o$ does not appear in the calculation of $\mathbf{e}_{\text{PBVS}}(t)$. Strictly speaking, the state-feedback controller only requires ${}^c\mathbf{w}_o$ to be observed. The purpose of including the velocity of the target object $\mathbf{v}_o$ as part of the states in the observer is to allow the observer to track a potentially moving target object. Camera self-motion is also incorporated as input into the Kalman state prediction, so that it can better track the pose of the object during fast camera motions. Experimental results show that this method offers an improvement over the traditional approach of modeling all motions as random disturbances.

4.5 Simulation Results

4.5.1 Purpose and Setup

The purpose of these simulations is to evaluate the image trajectories, the Cartesian trajectories and the robot trajectories that result from visual servoing using online pose estimation. The observed target is a non-planar object with 9 identifiable feature points. The target object is positioned above the robot in these experiments to increase the effective range of motion available to the robot for servoing, since robot joint limits are not explicitly managed in the servoing control law. The simulations are performed on a 6-DoF CRS-A465 robot with a Hitachi KP-D8 camera amounted on the end-effector to reflect the equipment available in the lab.

The first two robot positioning tasks are similar to those defined in Section 3.3.1. The first is a translational task that requires the robot end-effector to approach the target object with small out-of-plane rotations. The

second is a rotational task of 180 degrees about the optical axis. Both tasks are executed by applying PBVS with respect to the stationary target frame. The resulting trajectories are compared against those obtained using IBVS, as presented in Section 3.3.2, to discern any improvements to the observed motion.

The third robot positioning task presents the camera field-of-view problem that commonly affects PBVS. This task is executed by applying PBVS with respect to the camera frame, and involves significant out-of-plane rotations and coupled with large translational motions. Due to an initially poor pose estimate, the target object is brought outside of the camera's field of view during PBVS. An adaptive gains controller is tested with PBVS to determine whether the camera field-of-view constraints can be satisfied, while simultaneously completing the servoing task.

The fourth robot positioning task evaluates the tracking performance of PBVS with an EKF in the presence of severe image noise. The robot is commanded to maintain a relative pose with respect to a moving target object. The target object moves with a constant velocity of $0.1m/s$ in the x direction and the image noise has a standard deviation of ± 2 pixels.

4.5.2 Position-Based Visual Servoing

The form of the state-feedback adaptive gains controller and that of the EKF observer for position-based visual servoing (PBVS) is described in Sections 4.2 and 4.3 respectively.

Figure 4.1 shows the results from the first robot positioning task as described in Section 4.5.1 using PBVS for feedback control. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. For this task (mainly translation with some small out-of-plane rotations), PBVS generates acceptable image trajectories and Cartesian trajectories. Using PBVS, the image trajectories of the feature points are more curved than the near-straight-line trajectories that are obtained using IBVS, so there is a significant chance that the image trajectories will exit the field-of-view. Recall that in PBVS, the control metric is expressed purely in the position domain, so there is no direct control over the image trajectories. Nonetheless, the resulting Cartesian trajectory is extremely efficient, as it is a straight line connecting the camera start-position to the camera end-position. This property makes it easier to predict the motion of the camera using PBVS for the purpose of collision avoidance and joint-limit avoidance. These results show that the trajectories generated with PBVS are sometimes acceptable, for large translational tasks with some out-of-plane rotations.

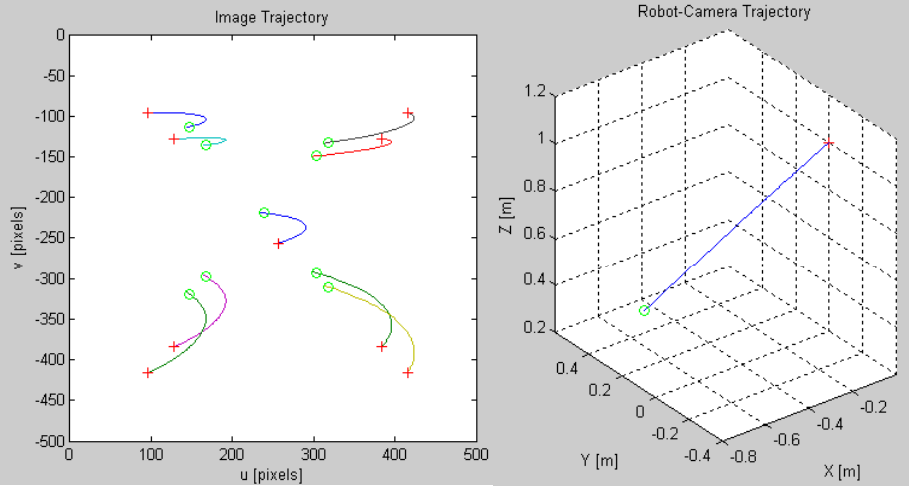


Figure 4.1: PBVS of a task requiring translations and out-of-plane rotations: image trajectory (left) and Cartesian trajectory (right).

Figure 4.2 shows the results from the second robot positioning task as described in Section 4.5.1 using PBVS for feedback control. Unlike IBVS which forces the camera to retreat (resulting in the violation of joint limits), PBVS is able to correctly recognize that a 180-degree rotation is required, using the pose estimation obtained from the EKF. PBVS successfully avoids the problem of camera retreat. The Cartesan trajectories show that PBVS executes the required rotational motion, without inducing any extra translational motion. This property of decoupled control illustrates one of the many benefits of using pose estimation, when a model of the target object is available, to simplify visual servoing motion. An additional advantage of obtaining an *accurate* pose estimation through a robust method like EKF is that it can be subsequently used for path planning.

4.5.3 Camera Field-of-View Constraints

Results from the third robot positioning task show that the adaptive-gains PBVS controller presented in Section 4.2.3 works well for keeping the target object within the camera's field of view. Figure 4.3 shows a visual servoing task in which the target object exits the camera's field of view when PBVS is used. In contrast, when the adaptive gains are applied, the rotational velocities of the camera become severely attenuated as the object moves towards

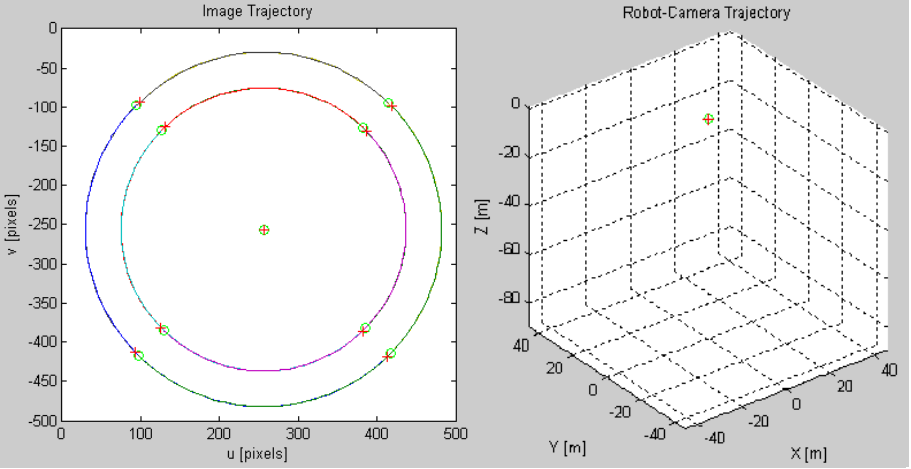


Figure 4.2: PBVS of a task requiring a rotation of 180 degrees about the optical axis: image trajectory (left) and Cartesian trajectory (right).

the camera's periphery. The object is brought back towards the center of the image by the remaining translational degrees of freedom. Control of the camera's rotation resumes when the target object returns back to the center of the region of interest. Although there are partial losses of visual features during the motion, pose estimation is still possible through the modification to the EKF. The entries in the covariance matrix that correspond to the lost features are increased to eliminate their influence on the pose estimate. The entries are returned to their normal values when the features return back into the field of view. In order for this adaptive-gains approach to work, the visual updates from the camera must be sufficiently fast, in order to capture the violating image point as it enters into the high penalty region of the camera's periphery prior to its exit from the field of view. Using an excessively wide penalty region tends to slow down the convergence of the overall trajectory. The bell-curve parameters $m = 5$ and $n = 5$ were chosen experimentally to give acceptable results.

4.5.4 Visual Servoing Near Robot Singularities

Simulations show that the visual servo control schemes become ill-conditioned and may result in large joint velocities when the camera is driven near the kinematic singularities of the robot. Path planning methods can be used to set up singularity-free waypoints for visual servoing. However, this requires

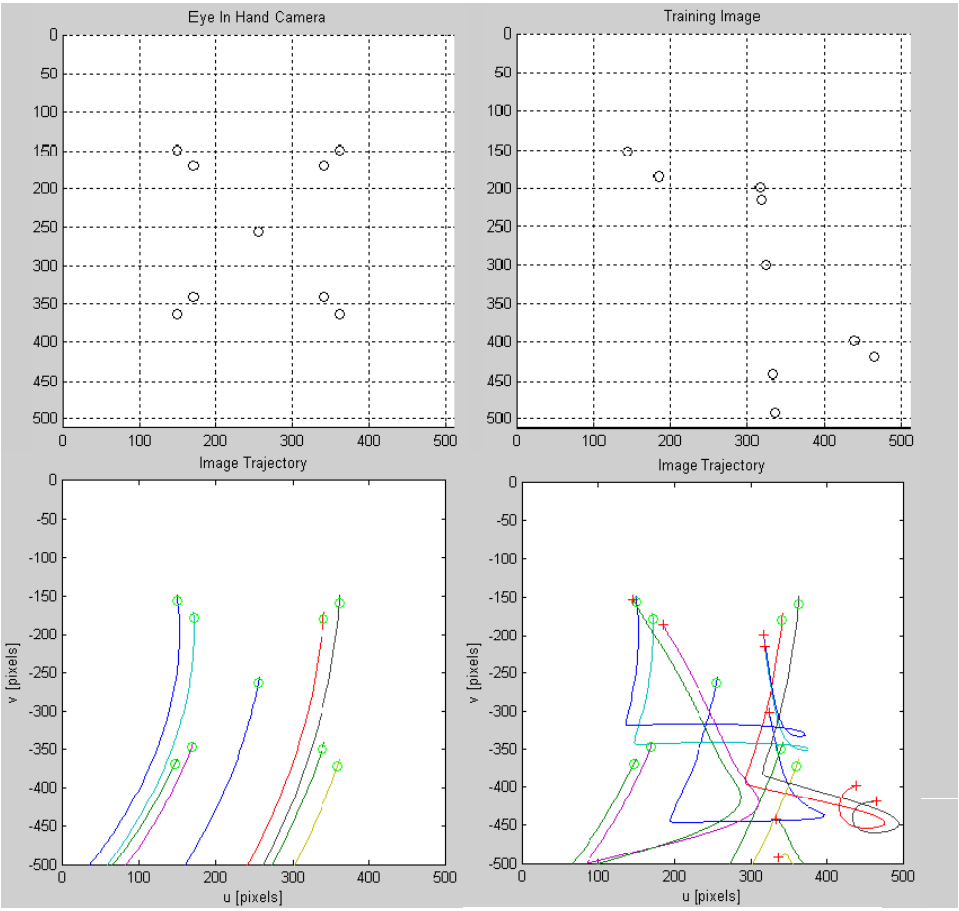


Figure 4.3: Using adaptive gains with hierarchical control of camera translation and rotation to keep the target object within the field of view: initial image (top left); desired image (top right); target object leaves the field of view under PBVS control (bottom left); target object remains in view using adaptive gains with PBVS (bottom right).

the robot trajectory to be known ahead of time. In many visual servoing applications, the trajectory of the camera is not known prior to servoing, so measures must be taken to account for unexpected trajectories through robot singularities. In the simulations, robot singularities were addressed for visual servoing using the damped least squares inverse method using Equation C.1 and Equation C.3 (from Appendix C). The maximum damping factor, $\kappa_{\max}$, was experimentally chosen to be 0.02, with the size of the singular region, ε, defined to be 0.04 to give acceptable results. Larger values tend to bring the target object out of the field of view near singularities. This method did not suffer from numerical drift even when the inverse kinematics was implemented in discrete time. The visual inputs acted as feedback to correct for positional errors, so that additional feedback correction terms were not required.

4.5.5 Tracking performance of EFK and PBVS

To further evaluate the tracking performance of PBVS, a tracking task is simulated with a moving target object. The target object moves with a constant velocity of $0.1m/s$ in the x direction and the image noise has a standard deviation of ± 2 pixels.

The transient behavior of the EKF observer in the presence of image noise is shown in Figure 4.4. At the start of the task, the pose estimate tends to oscillate about the true solution, indicating that the initialization for the state covariance matrix is inaccurate. As the covariance matrix is refined over the next iterations, the Kalman gain becomes more conservative, and the estimates become more robust as they gradually converge towards the real solution. It is found that an arbitrary initiation of the state covariance matrix to the identity matrix is sufficient to achieve convergence. Figure 4.5 shows the evolution of errors in the pose estimation.

The transient behavior of the state-feedback adaptive gains PBVS controller while tracking a moving object is shown in 4.6. The oscillations in the camera motion are directly related to the oscillations in the estimate of the object pose, caused by measurement noise in the images. It can be seen that the accuracy of the camera pose (controller) is limited by the accuracy of the pose estimates (observer). The pose estimates converge quickly and the camera motion becomes smoother as the true pose solution is reached. Unfortunately, a state-steady error still exists in the X component of the camera pose due to the form of the proportional velocity controller. This error can be reduced by the use of a proportional-integral controller in visual servoing. Figure 4.7 shows the evolution of pose-following errors.

Finally, the transient behavior in the EKF estimate of the target object's velocity is shown in Figure 4.8. Because the velocity states are not directly observable from the output equations, the estimates take many more iterations to converge. The target object's velocity estimate is shown to converge to 0.1 m/s in the X direction.

4.6 Summary

An extended Kalman filter (EKF) is implemented in this chapter to provide robust estimates of the target's 6-D pose from 2-D image measurements. Simulation results show the ability of the EKF observer to obtain accurate pose estimates despite the presence of large image noise and disturbances. The use of an object model and state tracking to regularize the pose estimation problem shows that significant improvements can be obtained over the method of homography estimation and decomposition. The EKF observer also produces a state covariance matrix, which is useful for quantifying the level of uncertainty in the target object's pose. In contrast to the EKF observer implemented by Wilson *et al.* in [13], the EKF observer implemented here incorporates camera self-motion as an input into the state prediction to improve tracking during fast camera movements. The absolute velocity of the target object is also tracked as a state, so that the object does not have to be assumed to be stationary.

Simulation results validate position-based visual servoing (PBVS) as a useful method for positioning an eye-in-hand robot with respect to a target object. The definition of the error metric based on 3-D parameters results in an upper triangular Jacobian that permits the decoupled control of camera translation and rotation. Decoupled control allows an adaptive control law to be designed to ensure that the target object remains within the field of view during visual servoing. PBVS results in efficient Cartesian motions, although the robot's joint-space may still be unfeasible. A method of handling robot singularities during visual servoing is also presented, using a damped least-squares inverse kinematic solution to restore controllability. However, the remaining issues related to the robot-related constraints such as joint-limits constraints, joint-velocity constraints, and whole-arm collision constraints still need to be addressed. In the next chapter, the pose estimate that is obtained from EKF will be used to enable joint-space path planning for satisfying these constraints.

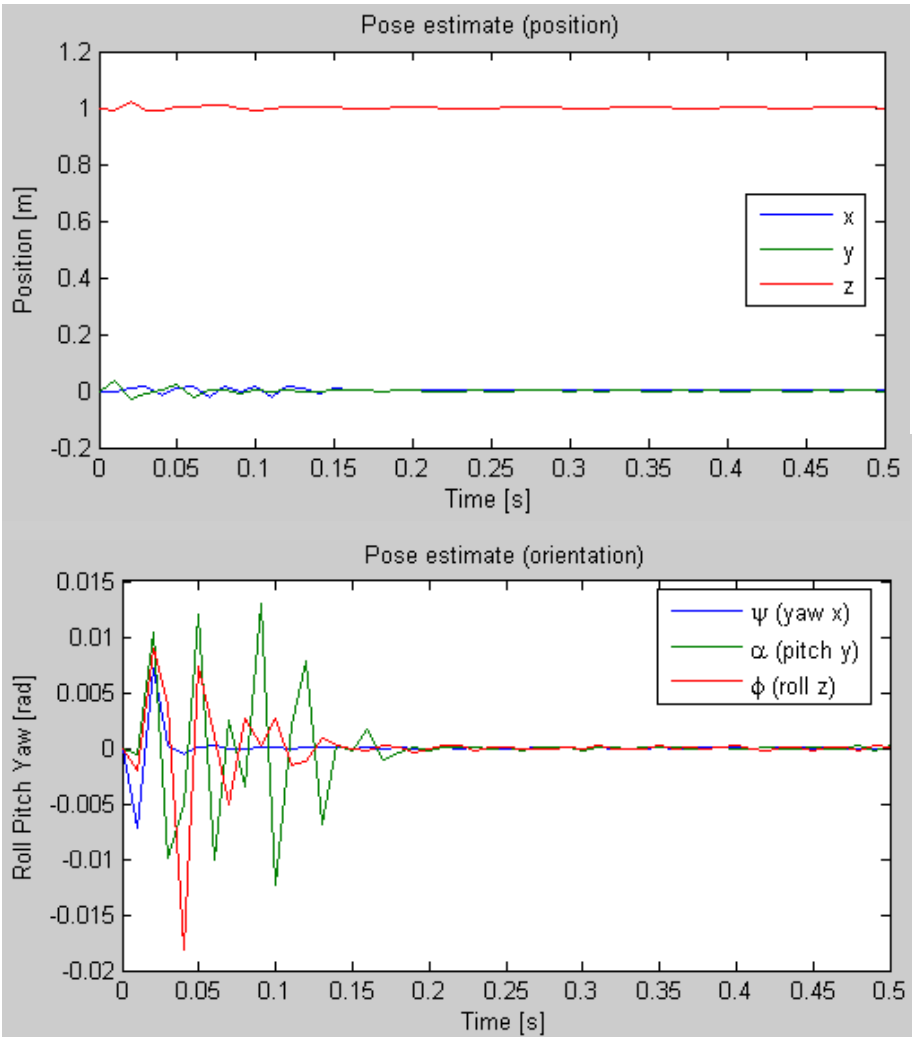


Figure 4.4: Transient behaviour of the EKF pose estimate: translation parameters (top) and rotation parameters (bottom).

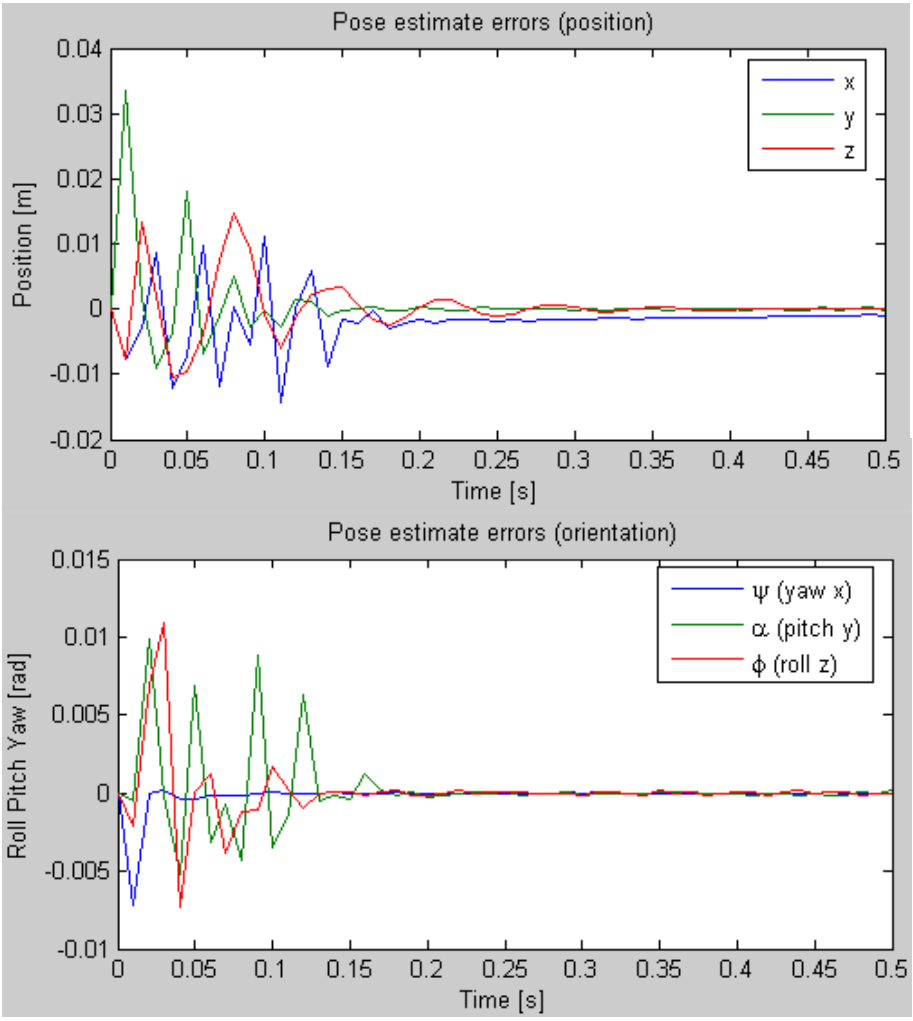


Figure 4.5: Transient behaviour of pose estimation errors in the EKF: translation errors (top) and rotation errors (bottom).

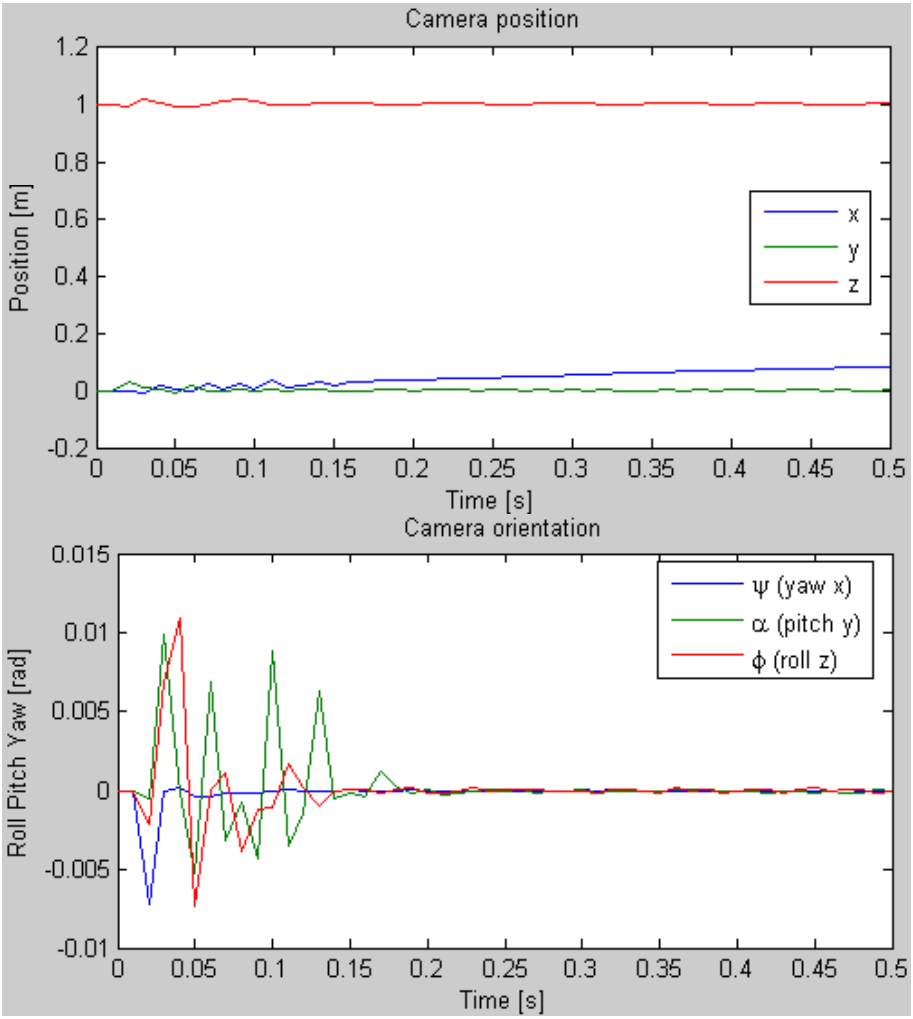


Figure 4.6: Transient behaviour of the PBVS controller with an EKF observing a target object moving at $0.1m/s$ in the X direction: camera position (top) and camera orientation (bottom).

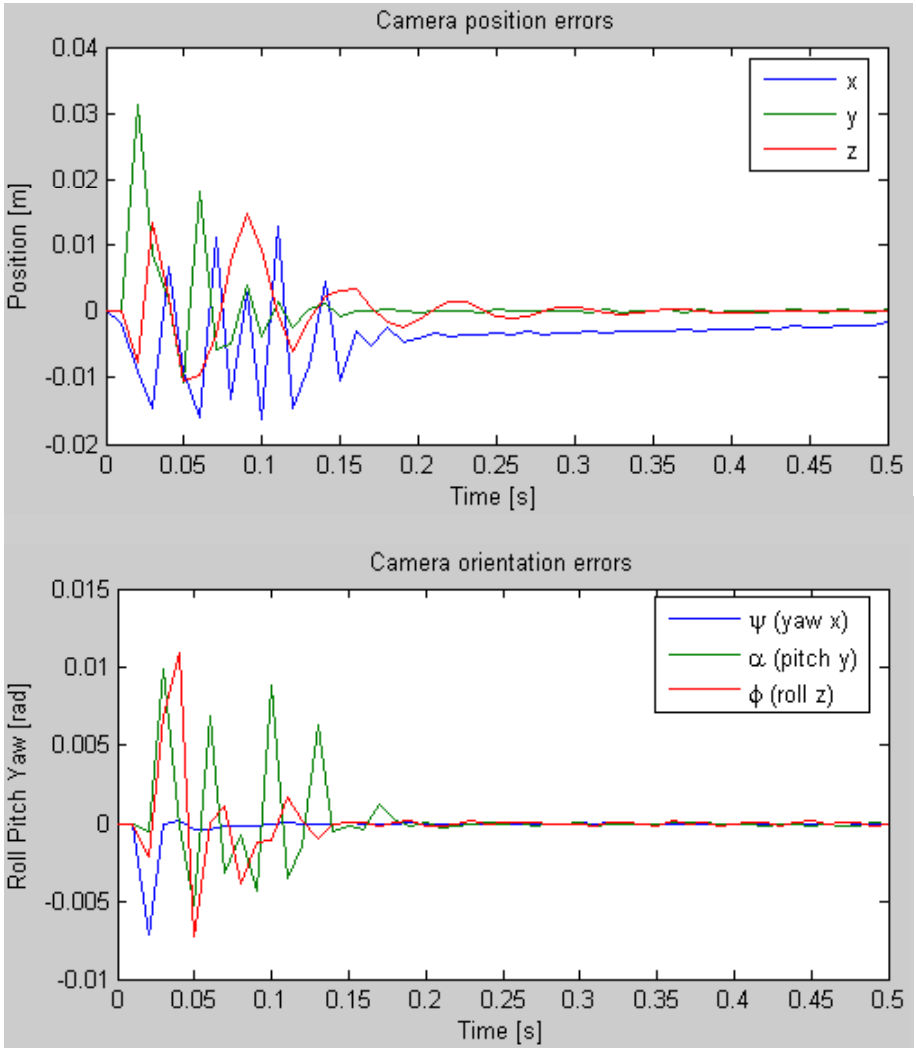


Figure 4.7: Pose-following errors in the PBVS controller with an EKF observing a target object moving at $0.1m/s$ in the X direction: camera position errors (top) and camera orientation errors (bottom).

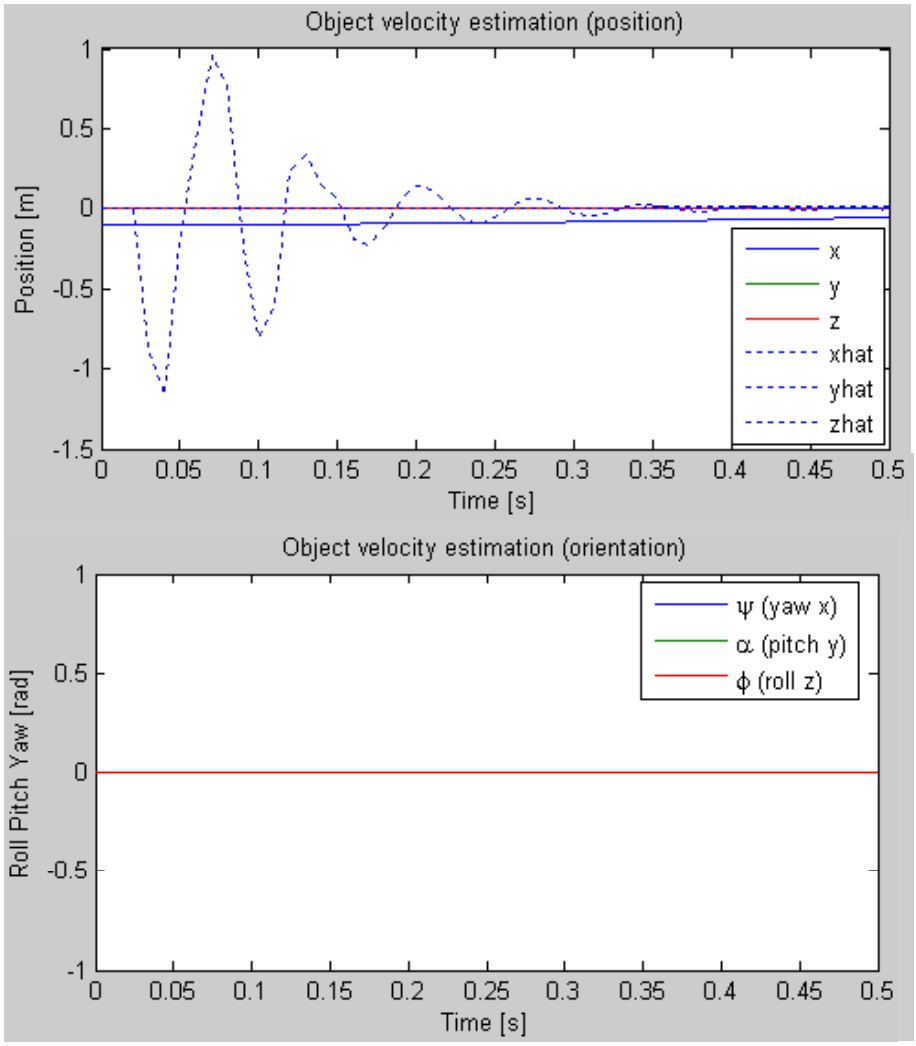


Figure 4.8: Transient behaviour in the EKF estimate of the target object's velocity: translation rate of change (top) and rotational rate of change (bottom).

Chapter 5

Path Planning for Constraint Avoidance

5.1 Introduction

In the previous chapters, it was shown that pose estimation can be used to provide significant improvements to the predictability of the visual servoed robot motion. Camera field-of-view constraints were addressed through the use of an adaptive controller that acted as a filter to establish hierarchical control of the six degrees of freedom of the camera to restore visibility. A damped least-squares inverse Jacobian method was implemented to generate acceptable trajectories while servoing near robot joint singularities. However, despite these improvements to the visual servo control law, the avoidance of constraints that are related to the robot, such as joint limits and whole-arm collisions, could not be formally guaranteed.

The chapter explores the use of path planning as a prerequisite to visual servoing for managing robot-related constraints. The idea is to execute as much of the required motion as possible using planned motion, while using visual servoing to compensate for uncertainties in the target object's pose, executed in two separate stages. A path planning stage is useful since there are a variety of techniques [51] [52] for handling constraints when the robot's joint-space motion is known *a priori*, while the same problem is much more difficult for visual servoing since the generated motion is difficult to predict. Using this two-stage approach, the servoing motion can be kept to a minimum to avoid possible violations of constraints. The path planning methods presented in this chapter work well for solving difficult problems in collision avoidance, such as navigating a manipulator arm in a tightly cluttered workspace. In contrast, the next chapter considers a method that allows planning and servoing to be executed simultaneously using model predictive control (MPC), to avoid whole-arm collisions with obstacles that do not tightly constrain the robot motion.

5.2 Path Planning Requirements

The objective of path planning is to define a sequence of configurations that a robot should follow in order to accomplish its positioning task, while satisfying field-of-view, joint-limit, and collision constraints. Path planning is typically executed offline based on some knowledge of the relationship between the robot and its environment. Typically, path planning occurs in the robot's joint-space, rather than in image-space, since many of the physical constraints (such as whole-arm collision and robot joint-limits) are less sensitive to errors when they are expressed in the joint-space of the robot. Image-based planning techniques [30] [31] [32] based on homography decomposition, potential fields planning in Cartesian-space, and reprojection back into the image, may work well for keeping the image trajectory in the field of view, but they are insufficient for the avoidance of complicated robot-related constraints. Homography decomposition is shown to suffer in accuracy when the camera displacement is large. In the presence of errors, motion that is planned in the image-space may not correspond to the desired motion in the robot's joint-space. Even if the homography decomposition is completely accurate, repulsive potential fields applied to the position of the camera can only prevent camera collisions with obstacles, while whole-arm collisions can still occur. Discussions on path planning in this chapter will concern the joint-space of the robot.

The planning of robot joint-space motion for camera positioning requires a rough estimate of the target object's pose. The methods for *partial* pose estimation using homography decomposition or *complete* pose estimation using an extended Kalman filter were discussed in the previous chapters. The estimated pose may or may not be accurate, so visual servoing must be used at the end of the planned motion to compensate for any modelling errors. However, visual servoing can only be initiated with the target object in view. The ideal scenario for the two-stage approach is where the target object is in the camera's field of view *at the end* of the planned motion. However, there is no way of knowing this at the planning stage, since by the nature of the problem, the object's *actual location* is uncertain. The desired path must be planned to keep the target object in view (based on its best known position) while reaching the goal, such that if the target object exits a region of interest representing the field of view (due to poor initial assumptions about it pose), the robot controller can immediately switch to visual servoing to resume control and "close the loop" to the end goal. To allow switching at anytime, the robot uses a series of linearly interpolated waypoints to achieve its planned motion, which can be interrupted between the

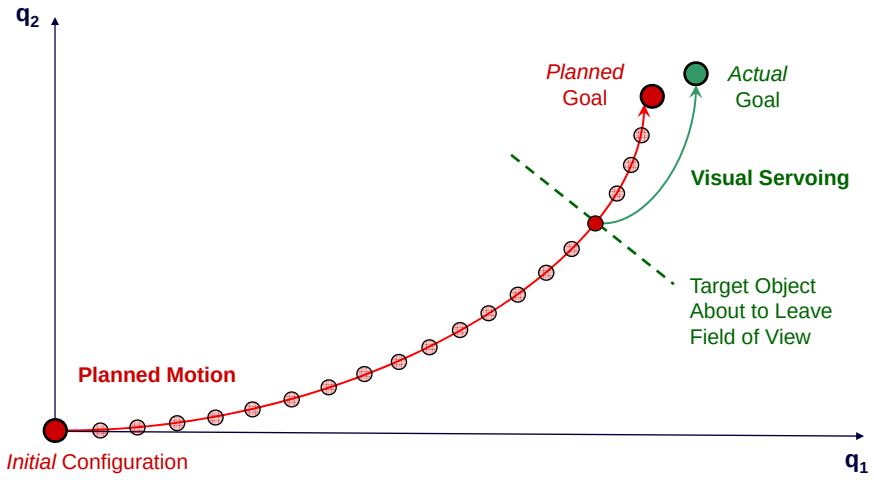


Figure 5.1: Two-stage approach for planning and servoing: joint-space path planning addresses collision and joint-limit constraints; visual servoing completes the motion correcting for uncertainties.

execution of waypoints. This two-stage approach for planning and servoing is summarized in Figure 5.1.

5.3 Pose Calculations and Modeling

Let $\mathcal{F}_o$ be the canonical frame attached to the target object. The eye-in-hand robot starts at an initial configuration $\mathbf{q}_{\text{initial}}$ and provides an initial view of the target object with a pose, represented by $\mathcal{F}_o$, that may vary for each instance of robot servoing. For the purpose of the planning algorithm, it is assumed that the target object remains stationary with respect to the robot base frame $\mathcal{F}_b$ during all robot motion. The goal is to position the camera $\mathcal{F}_c$ at a desired pose with respect to the target object $\mathcal{F}_o$. This desired pose is specified via an offline teach-by-showing method. The pose of the object in the camera frame in the initial and desired images, respectively $({}^c\mathbf{T}_o)_{\text{initial}}$ and $({}^c\mathbf{T}_o)_{\text{desired}}$, are calculated using methods described in the previous chapters (EKF pose estimation, homography decomposition with known scale). There exists a fixed geometric relationship between the end-effector frame $\mathcal{F}_e$ and the camera frame $\mathcal{F}_c$, described by the homogeneous transformation ${}^e\mathbf{T}_c$. The target object is within the field of view of the

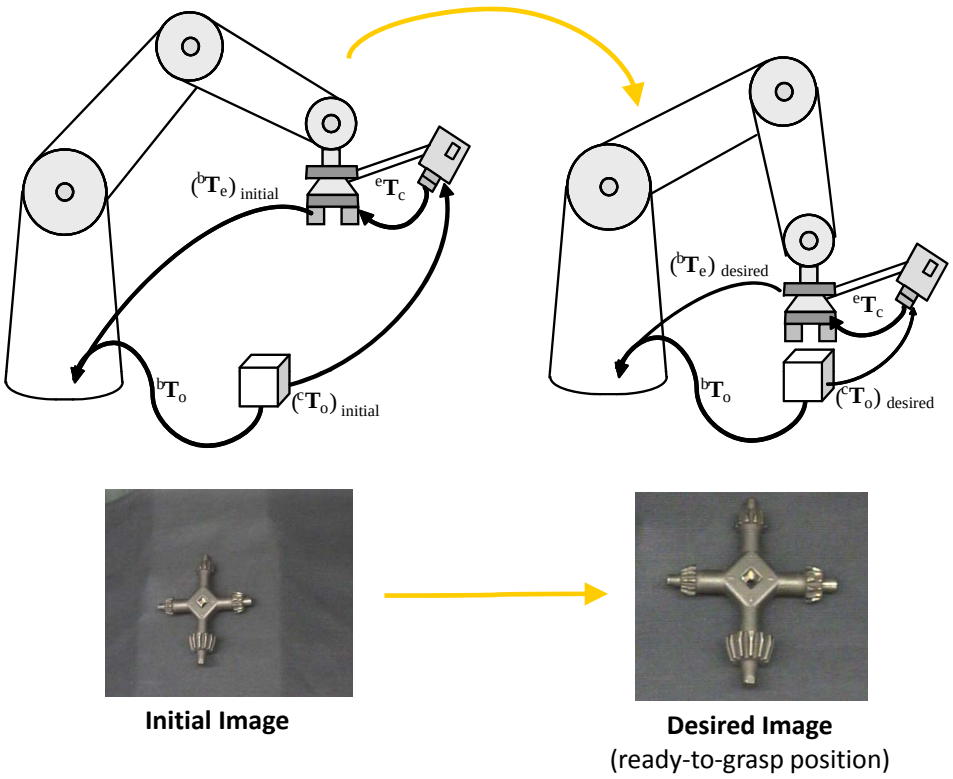


Figure 5.2: Homogeneous transformations between coordinate frames used for path planning: initial end-effector position (left) and desired end-effector position (right). The corresponding images observed by the camera are shown at the bottom.

camera at the start and at the end of the robot motion. A diagram of the relationship between coordinate frames is shown in Figure 5.2.

The pose of the object with respect to the robot frame, ${}^{b}\mathbf{T}_{o}$, can be computed by the following:

$$ {}^{b}\mathbf{T}_{o} = ({}^{b}\mathbf{T}_{e})_{\text{initial}} {}^{e}\mathbf{T}_{c} ({}^{c}\mathbf{T}_{o})_{\text{initial}}, \tag{5.1} $$

$$ {}^{b}\mathbf{T}_{o} = ({}^{b}\mathbf{T}_{e})_{\text{desired}} {}^{e}\mathbf{T}_{c} ({}^{c}\mathbf{T}_{o})_{\text{desired}}, \tag{5.2} $$

where ${}^{b}\mathbf{T}_{e}$ is the homogeneous transformation mapping the robot base frame $\mathcal{F}_b$ to the end-effector frame $\mathcal{F}_e$. This relationship is a function of the well-known forward kinematics of the industrial robot. The initial transformation

is computed from the vector of the measured joint angles, $\mathbf{q}$, as:

$$({}^{b}\mathbf{T}_{e})_{\text{initial}} = f_{\text{robot}}(\mathbf{q}_{\text{initial}}). \tag{5.3}$$

It is necessary to determine the set of joint configurations $\mathbb{Q}_{\text{desired}}$ which correctly position the camera relative to the target object. As discussed in the following section, typical industrial robots have closed-form analytic solutions to the inverse kinematics of its end-effector. Let there be p distinct inverse kinematic solutions available:

$$\mathbb{Q}_{\text{desired}} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_p\} \tag{5.4}$$

$$= f^{-1}_{\text{robot}}({}^{b}\mathbf{T}_{e})_{\text{desired}} \tag{5.5}$$

where $({}^{b}\mathbf{T}_{e})_{\text{desired}}$ is computed through the following expression:

$$({}^{b}\mathbf{T}_{e})_{\text{desired}} = ({}^{b}\mathbf{T}_{e})_{\text{initial}}\,{}^{e}\mathbf{T}_{c}({}^{c}\mathbf{T}_{o})_{\text{initial}}({}^{c}\mathbf{T}_{o})^{-1}_{\text{desired}}({}^{e}\mathbf{T}_{c})^{-1} \tag{5.6}$$

5.4 Inverse Kinematic Solutions for Path Planning

The inverse kinematic calculations allow the path planning problem to be expressed in the joint-space of the robot. For anthropomorphic robots with with revolute joints such as the one depicted in Figure 5.2, there may be multiple robot joint-space configurations that satisfy the desired relative camera pose. If collision constraints are satisfied for the motion between the start and end robot configurations (using a dynamic collision checking method presented in Section 5.5.2), then a linear joint interpolation method can be used to naturally satisfy robot joint-limit constraints, while ensuring efficient robot motion towards the chosen robot configuration. For given upper-bounds on joint velocities set by the manufacturers of industrial robots, this solution minimizes the travel time required for the motion.

The method presented here is similar to the *look-then-move* approach discussed in Section 1.2 for robot bin-picking, with the exception that visual servoing is used at the end of the planned motion to correct for any modelling errors or pose estimation errors. In the presence of errors, the chosen inverse kinematic solution must attempt to provide continuous visibility of the target object throughout the interpolated path, so that if the target object exits a region of interest (ROI) representing the camera's field of view

at any time when the planned motion is executed, visual servoing can be immediately activated to "close the loop" to the end goal.

It is well known that, due to the robot's mechanical joint limits, not all closed-form inverse kinematic solutions correspond to physically realizable configurations. Once a mathematical solution is identified, it must be further checked to determine whether it satisfies all constraints on the robot's range of joint motions. An extension of this requirement, for eye-in-hand robots visual servoing, is to check whether the planned interpolated trajectory for each inverse kinematic solution satisfies the field-of-view constraints of the camera for a given target object. Although the final camera pose is identical for all inverse kinematic solutions, the resulting interpolated camera trajectories are quite different for each of these joint configurations, and some of the interpolated trajectories may bring the target object outside of the field of view.

A criterion for selecting an appropriate inverse kinematic solution for joint-space path planning is that its interpolated trajectory must keep the target object within the camera's field of view, preferably in manner that is robust to disturbances. The most *robustly visible* inverse kinematic solution is defined as the one that maximizes the distance between the target object and the camera field-of-view limits during the interpolated motion. The aim is to select a robot configuration, which provides the largest buffer against target pose estimation errors, to ensure that a continuously visible path will most likely result during the execution of the planned path. This selection criteria is described more rigorously in the next section.

5.4.1 Visibility Modelling

Let the target object consist of a number of visual features. The image coordinates (u, v) of the n visual features are functions of the joint coordinates of the robot, where:

$$\mathbf{u} = \begin{bmatrix} u_1(\mathbf{q}) & \cdots & u_n(\mathbf{q}) \end{bmatrix}^T, \tag{5.7}$$

$$\mathbf{v} = \begin{bmatrix} v_1(\mathbf{q}) & \cdots & v_n(\mathbf{q}) \end{bmatrix}^T. \tag{5.8}$$

Let the robot path be parameterized by $\mathbf{q}(\zeta)$ with ζ as the path variable, where $\mathbf{q}_{\text{initial}} = \mathbf{q}(0)$ and $\mathbf{q}_{\text{desired}} = \mathbf{q}(1)$. Continuous visibility constraints require that all features remain within the field of view of the camera throughout the interpolated robot path. The problem of selecting the most *robustly visible* inverse kinematic solution is formulated as follows:

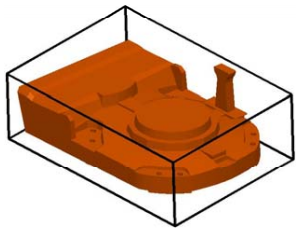


Figure 5.3: Oriented bounding box used to represent the target object for field-of-view visibility planning.

$$\underset{\mathbf{q}_{\text{desired}} \in \mathbb{Q}_{\text{desired}}}{\text{argmax}} \left(\min \begin{pmatrix} |\mathbf{u}(\mathbf{q}(\zeta)) - u^{\min}|, \\ |\mathbf{u}(\mathbf{q}(\zeta)) - u^{\max}|, \\ |\mathbf{v}(\mathbf{q}(\zeta)) - v^{\min}|, \\ |\mathbf{v}(\mathbf{q}(\zeta)) - v^{\max}| \end{pmatrix} \right), \tag{5.9}$$

such that

$$u^{\min} \leq \mathbf{u}(\mathbf{q}(\zeta)) \leq u^{\max}, \quad \forall \zeta, \tag{5.10}$$

$$v^{\min} \leq \mathbf{v}(\mathbf{q}(\zeta)) \leq v^{\max}, \quad \forall \zeta, \tag{5.11}$$

where $u^{\min}$, $u^{\max}$, $v^{\min}$, $v^{\max}$ are defined by the image region of interest (ROI) representing the field of view of the camera.

To extend this method for handling objects with complicated features (such as areas, lines, and image moments), an oriented bounding box is defined with respect to the target object frame $\mathcal{F}_o$ such that it encloses the target object and all its relevant visual features, as shown in Figure 5.3. A union of bounding boxes is used to increase the planning resolution where possible. In the presence of possible self-occlusion by the target object, this method implicitly assumes that minimum subsets of visual features can always be observed around the object (when it is within the field of view) for use in visual servoing. The location of the corners of the bounding box(es) are tracked to bound the visual location of the target object. The target object is guaranteed to remain within the field of view of the camera if the corners of the bounding box(es) are continuously visible.

5.4.2 Simulation Results

A vision-guided positioning task is simulated using a model of the CRS-A465 robot with a Sony XC-HR70 camera mounted on its end-effector. The target object is a 20cm×20cm×20cm box made up of eight corner features plus the geometric centroid. The goal is to select an inverse kinematic solution that keeps all these features inside the camera's field of view during the interpolated robot motion. The camera starts from an initial overhead view, and is required to approach the target object while performing significant out-of-plane rotations. Of the eight closed-form inverse kinematic solutions that are geometrically available to the CRS-A465 robot to achieve the desired camera pose, four are outside of the robot's mechanical joint limits. The image trajectories corresponding to the remaining four inverse kinematic solutions are shown in Figure 5.4. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. The motion is executed using linear joint-space interpolation. Note that only two of the four solutions give satisfactory visual trajectories. Following the constrained optimization criteria defined in Section 5.4.1, the solution in the bottom left is chosen for joint-space path planning.

If no feasible trajectory (for ensuring target visibility) exists among the inverse kinematic solutions, then additional intermediate robot configuration must be specified to guide the robot towards its goal, while positioning the target object within the camera's field of view at each configuration. The insertion of an additional *visible* robot configuration divides the field-of-view problem into two sub-problems of the same form but of lesser difficulty, since the robot's interpolation distance is reduced. The problem of generating intermediate robot configurations that position the target object within the field of view is discussed in Section 5.6 within the framework of probabilistic roadmaps (PRM).

5.5 Probabilistic Roadmaps

Probabilistic roadmaps (PRM) [51] [52] have emerged, over the past decade, as one of the popular methods for robot path planning to simultaneously manage multiple constraints. The intuition behind PRM consists of randomly generating a large sample of robot configurations in joint-space, so that various paths can be planned between these configurations for constraint avoidance. The sampled robot configurations make up the vertices in an undirected graph. In the graph, two vertices are connected by an edge if the trajectory between the two vertices satisfies a set of constraints. For

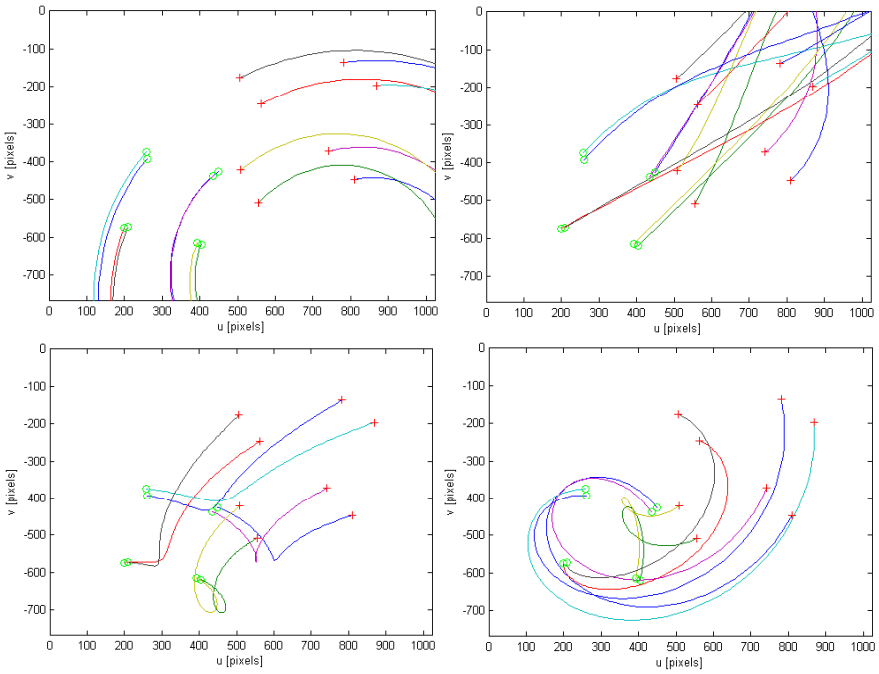


Figure 5.4: Image trajectories resulting from four different inverse kinematic solutions: two solutions result in the target object leaving the field of view (top), two solutions result in a satisfactory image trajectories (bottom).

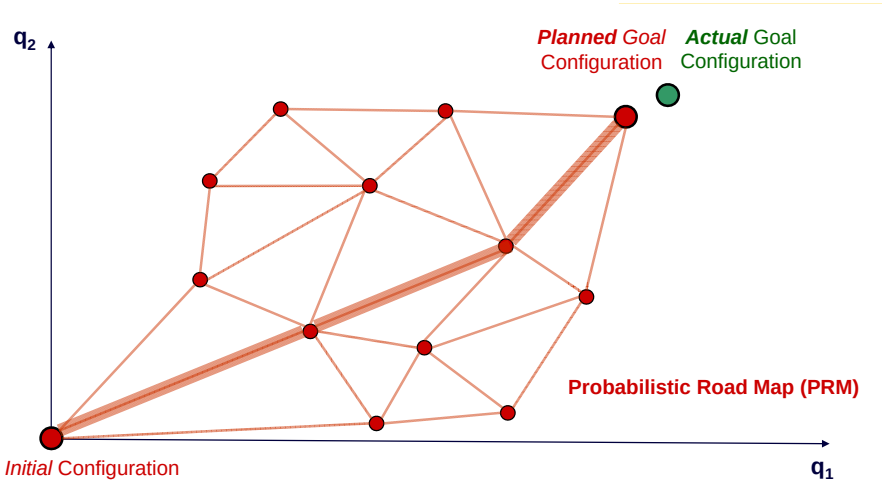


Figure 5.5: PRM representation of the robot path planning problem: nodes represent feasible robot configurations, edges represent feasible robot motions.

each type of constraint, there exists an efficient algorithm to check whether the constraint is satisfied, as presented in the next section. Once several connections of vertices exist between the starting robot configuration and the goal robot configuration, the problem becomes a path search in a undirected graph with edges weighted by joint-space distance or some other metric. Dijkstra's algorithm, a well-known graph search technique, is used to provide the most efficient path through the roadmap. An illustration of the PRM framework in two-dimensional joint-space is presented in Figure 5.5.

5.5.1 Robot Joint-Limit Constraints

Path planning in the joint-space of the robot allows a natural representation of joint-limits as a hyper-rectangular region in which feasible robot configurations exists. Robot joint-limit constraints are handled by restricting the sampling of vertices to within the hyper-rectangular region. A separate algorithm is not required to check whether robot-limit constraints are satisfied between two adjacent vertices.

5.5.2 Whole-Arm Collision Constraints

A dynamic collision checking (DCC) algorithm proposed by Schwarzer *et al.* [53] is used to test for collisions between the robot and obstacles in the workspace. DCC determines if a robot collides with the environment *during a motion* between two configurations. Specifically, given the initial configuration $\mathbf{q}_i$ and the final configuration $\mathbf{q}_f$, DCC determines if the trajectory defined by $\mathbf{q}(t) = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i)t$ for $0 \leq t \leq 1$ results in a collision.

As a prerequisite, DCC requires the robot configurations at the two vertices to be collision-free. To guarantee collision-free motion in between the configurations, DCC exploits the relationship between the shortest distance between the robot and any obstacle, and the longest distance traveled by any point on the robot during a trajectory. Let $d(\mathbf{q}_i)$ be the shortest distance between an obstacle and the robot in the configuration $\mathbf{q}_i$. Similarly, let $d(\mathbf{q}_f)$ be the shortest distance between an obstacle and the robot in the configuration q_f. Also, given a trajectory $\mathbf{q}(t)$, let $\ell(\mathbf{q}(t))$ be the the longest distance traveled by any point on the robot during the trajectory. That is, as the robot moves according to $\mathbf{q}(t)$ each point on the hull of the robot describes a 3-D curve and $\ell(\mathbf{q}(t))$ represents the length of the longest of these curves. Then, given $d(\mathbf{q}_i)$, $d(\mathbf{q}_f)$ and $\ell(\mathbf{q}(t))$, DCC determines that the manipulator does not collide with the obstacle if

$$\ell(\mathbf{q}(t)) < d(\mathbf{q}_i) + d(\mathbf{q}_f) \tag{5.12}$$

The intuition behind this inequality is that if every point on the robot travels a short distance during $\mathbf{q}(t)$ and the robot is far enough from an obstacle, then it is impossible for the robot to collide with the obstacle. Equation (5.13) is a sufficient condition, but it is not a necessary condition. If Equation (5.13) is not satisfied, the algorithm proceeds by dividing the trajectory $\mathbf{q}(t)$ in two trajectories $\mathbf{q}(u)$ for $0 \leq u < t/2$ and $\mathbf{q}(v)$ for $t/2 \leq v \leq 1$ and Equation (5.13) is evaluated for both trajectories. This procedure is applied recursively until every trajectory satisfies Equation 5.13 or a collision is detected in the process.

Further details on the DCC algorithm are presented in the next chapter, where the specific form of $\ell(\mathbf{q}(t))$ is discussed within the context of model predictive control (MPC).

5.5.3 Camera Field-of-View Constraints

A dynamic visibility checking (DVC) algorithm proposed by Leonard *et al.* [54] is used to test for visibility of a target object within the camera's field of

view. The hollow frustum formed by the camera's field of view is modelled as an physical extension of the robot manipulator. Using a method based on the DCC algorithm, DVC determines if the hollow frustum "collides" with the target object of interest, bringing the target object outside of the camera's field of view. Specifically, given the initial configuration $\mathbf{q}_i$ and the final configuration $\mathbf{q}_f$, DVC determines if the trajectory defined by $\mathbf{q}(t) = \mathbf{q}_i + (\mathbf{q}_f - \mathbf{q}_i)t$ for $0 \leq t \leq 1$ results in a "collision" between the camera's frustum and the target object.

A *visible* vertex is defined here as a static robot configuration that has the target object within the camera's field of view. DVC requires the two vertices of interest to be visible, similar in form to the prerequisite of the DCC algorithm for collision-free motion. To guarantee visibility between the configurations, DVC exploits the relationship between the shortest distance between the target object and the camera frustum, and the longest distance traveled by the target object with respect to the camera's moving frame of reference during a trajectory. Let $\widetilde{d}(\mathbf{q}_i)$ be the shortest distance between the target object and the camera frustum in configuration $\mathbf{q}_i$. Similarly, let $\widetilde{d}(\mathbf{q}_f)$ be the shortest distance between the target object and the camera frustum in configuration q_f. Also, given a trajectory $\mathbf{q}(t)$, let $\widetilde{\ell}(\mathbf{q}(t))$ be the the longest distance traveled by any point on the target object with respect to the camera's frame of reference during the trajectory. That is, as the camera moves according to $\mathbf{q}(t)$, each point on the target object describes a 3-D curve with respect to the camera and $\widetilde{\ell}(\mathbf{q}(t))$ represents the length of the longest of these curves. Then, given $\widetilde{d}(\mathbf{q}_i)$, $\widetilde{d}(\mathbf{q}_f)$ and $\widetilde{\ell}(\mathbf{q}(t))$, DVC determines that the target object does not "collide" with the frustum formed by the camera's field of view if

$$\widetilde{\ell}(\mathbf{q}(t)) < \widetilde{d}(\mathbf{q}_i) + \widetilde{d}(\mathbf{q}_f) \tag{5.13}$$

The intuition behind this inequality is that if every point on the target object travels a short distance with respect to the camera during $\mathbf{q}(t)$ and the target object is far enough from frustum representing the field of view limits, then it is impossible for the target object to "collide" with the frustum and leave the camera's field of view. Equation (5.13) is a sufficient condition, but it is not a necessary condition. If Equation (5.13) is not satisfied, the algorithm proceeds by dividing the trajectory $\mathbf{q}(t)$ into two trajectories $\mathbf{q}(u)$ for $0 \leq u < t/2$ and $\mathbf{q}(v)$ for $t/2 \leq v \leq 1$ and Equation (5.13) is evaluated for both trajectories. This procedure is applied recursively until every trajectory satisfies Equation 5.13 or a "collision" with the frustum is detected in the process.

5.6 Dynamic Generation of *Visible* Robot Configurations

5.6.1 Motivation

In many PRM implementations, a static roadmap is constructed only once (or very infrequently) to capture the relationship between the robot and the configuration of obstacles in the workspace. Assuming that the robot's workspace does not change and that is not extremely cluttered, it is not difficult to find static robot configurations (vertices) that are collision-free by random sampling. Many implementations of PRM generate their vertices by method of uniform random sampling, or by biased random sampling based on a proximity metric. Collision-free vertices are prerequisites to the DCC algorithm for determining if the motions in between the vertices are also collision-free. So, only vertices that correspond to collision-free static robot configurations are retained in the graph.

Visible vertices are prerequisites to the DVC algorithm, which determines if the motions in between vertices keep the target object within the camera's field of view. This prerequisite is more difficult to satisfy in practice, especially if the target object's location changes very frequently. In many vision-based applications, such as robot bin-picking as described in Section 1.2, the target object's location changes at every pick instance, while the workspace of the robot remains static. Removal of all vertices that are *not visible* with respect to a specific target object typically results in a rather sparse graph for path planning. Continued use of the PRM may require the generation of new vertices (robot configurations) that are specific to the pose of the target object. These new vertices correspond to robot configurations that position the target object within the camera's field of view, while providing a visible joint-space path to connect the start configuration to the goal configuration. A method of generating and selecting these new vertices *dynamically* for a target object in any given pose is presented in the next section.

5.6.2 Vertices Generation using Virtual IBVS

The goal of dynamic vertex generation is to provide a systematic method of creating, in the PRM, robot configurations that:

1. Keep the target object within the camera's field of view;

2. Provide a connecting path from the starting configuration to the end

goal configuration.

These vertices must be subsequently checked for the satisfication of collision constraints, using the DCC algorithm discussed in Section 5.5.2. A secondary purpose of these vertices is to act as location biases for further configuration sampling in the surrounding joint-space region if the collision constraints cannot be met using the original vertices.

With respect to camera field-of-view constraints, the ideal image trajectory of a given feature point is one that travels in a straight line from its location within the initial image to its location within the desired image. If such a trajectory could be realized, then the convexity of the image-space guarantees that the feature point remain inside the camera's field of view; namely the IBVS approach.

Here, the idea is to apply IBVS *virtually* with respect to the target object, in order to generate candidate *visible vertices* for PRM planning. However, the joint-space motion imposed by IBVS can be extremely convoluted; the sequence of configurations do not necessarily bring the robot closer to its joint-space goal configuration, as in the problem of camera retreat. Thus, it is desirable to "short circuit" the IBVS path in joint-space whenever possible, in order to generate feasible and efficient robot motion. A discussion of virtual IBVS failure is given in Section 5.6.4

IBVS is *virtually* applied with respect to the object's feature points to generate a set of candidate visible vertices for selection. In the case of complex objects, the corners of the oriented bounding box enclosing the target object are used as the image features, as discussed in Section 5.4.1. The control law for IBVS is described in Section 3.3.2.

5.6.3 Vertices Selection using a Joint-Space Metric

Let the IBVS trajectory be parameterized by $\mathbf{q}_{\text{IBVS}}(\zeta)$ with ζ as the path variable, where $\mathbf{q}_{\text{initial}} = \mathbf{q}_{\text{IBVS}}(0)$ and $\mathbf{q}_{\text{desired}} = \mathbf{q}_{\text{IBVS}}(1)$. The problem of vertex selection can be formulated as:

$$\underset{\mathbf{q}_{\text{IBVS}}(\zeta)}{\text{argmax}} \quad \zeta \tag{5.14}$$

such that

$$\|\mathbf{q}_{\text{IBVS}}(\zeta) - \mathbf{q}_{\text{IBVS}}(0)\| = \frac{1}{2}\|\mathbf{q}_{\text{IBVS}}(1) - \mathbf{q}_{\text{IBVS}}(0)\|. \tag{5.15}$$

A backward search algorithm is initiated from the end robot configuration, selecting the first point along the IBVS trajectory where the joint-space error is half of that between the start and end configuration. If the joint-space path is convoluted, the waypoint is asymmetrically biased towards to the end configuration rather than the start configuration. Due to the linear approximation used in the proportional control law in IBVS, joint-space trajectories near the end configuration are typically less convoluted than those at the start of the servoing path. A graphical illustration of this vertex selection method in two-dimensional joint-space is shown in Figure 5.6.

The two joint-space segments created by the selected vertex are checked for continuous visibility using DVC. The backward search algorithm is applied recursively for each segment, generating new visible vertices with respect to a joint-space metric, until the continuous visibility criterion is met in DVC. Typically, one or two visible vertices (selected from the IBVS path) are all that are required to bring the target back into view and to provide continuous target visibility between robot configurations. In the presence of uncertainties in the pose estimate and in the planning model, the robustness of this planning method is increased by reducing the size of the region of interest (ROI) representing the camera's field of view. The reduced field of view creates additional vertices to sub-divide the joint-space motion, forcing the camera to take more conservative motions. The advantage of the method of vertex selection over the execution of the entire IBVS path is that the joint-space motion can be shortened whenever possible, resulting in feasible and efficient robot motion.

5.6.4 Addressing Virtual IBVS Failure

Simulations show that virtual IBVS fails to converge if the trajectory passes near robot singularities, even though actuation limits do not exist in simulation. In the presence of numerical integration errors, the high joint velocities that are generated near singularities tend to bring the trajectory (both in image-space and in joint-space) away from the desired solution, bringing the target object outside of the field of view. A damped least-squares inverse kinematics solution [50] is implemented to allow virtual IBVS near robot singularities. Details on the implementation of this method is discussed in Section 4.5.4.

Nonetheless, virtual IBVS will fail if joint limits are reached (due to *infinite* camera retreat), or if the trajectory gets stuck in a local minima (due to poor selection of image features). In all planning scenarios, the camera pose that is achieved by virtual IBVS is compared with the desired camera

pose to determine whether the solution converges. In the event of virtual IBVS failure, the vertex selection algorithm is modified and applied to the *partial* IBVS trajectory. Let the *partial* IBVS trajectory be parameterized by $\mathbf{q}_{\text{IBVS}}(\zeta)$ with ζ as the path variable, where $\mathbf{q}_{\text{initial}} = \mathbf{q}_{\text{IBVS}}(0)$ and $\mathbf{q}_{\text{fail}} = \mathbf{q}_{\text{IBVS}}(1)$. The problem of vertex selection from a *partial* IBVS trajectory is formulated as:

$$\underset{\mathbf{q}_{\text{IBVS}}(\zeta)}{\text{argmin}} \quad \|\mathbf{q}_{\text{IBVS}}(\zeta) - \mathbf{q}_{\text{desired}}\| \tag{5.16}$$

A forward (regular) search algorithm is applied to the partial IBVS trajectory, selecting the configuration that is closest to the goal configuration in joint-space as the new vertex. The two joint-space segments created by the selected vertex are checked for continuous visibility using DVC. Where a full trajectory exists, Equation 5.14 is used to select the visible vertex; where a partial trajectory exists, Equation 5.16 is used. Because the selection of visible vertices is based on a joint-space metric, useful vertices can still be found amongst candidates that bring the robot closer to its configuration goal from a partial IBVS trajectory. A graphical illustration of this vertex selection method in two-dimensional joint-space is shown in Figure 5.7.

5.7 Simulation Results

The vision-guided positioning task is simulated using a model of the CRS-A465 robot with a Sony XC-HR70 camera mounted on its end-effector. The target object is a 20cm×20cm×20cm box made up of eight corner features plus the geometric centroid. The goal is to plan a path, via the insertion of visible robot configurations, to keep all these features inside the camera's field of view during the interpolated robot motion. The camera starts from an initial overhead view, and is required to approach the target object while performing significant out-of-plane rotations. To account for uncertainties in the pose estimate and in the planning model, the focal length of the camera model used for planning is modified to reduce the camera's field of view by 1/3 in each dimension. This has the same effect as defining a region of interest (ROI) representing the camera's reduced field of view for visibility planning in the presense of increased uncertainty.

With the reduced field of view, all inverse kinematic solutions fail to maintain continuous visibility on the target object, when using linear interpolation to generate robot motion. The inverse kinematic solution that

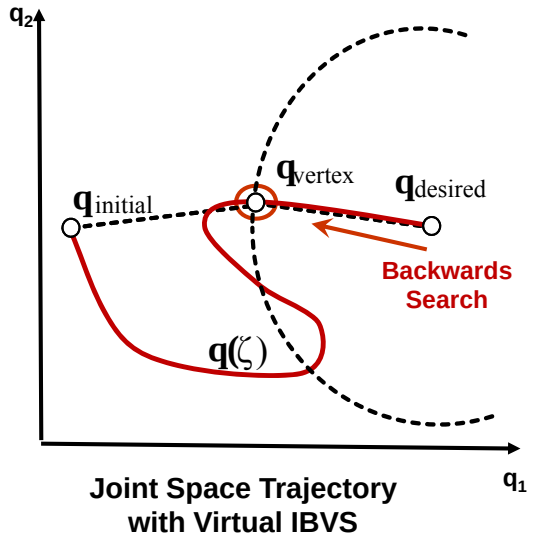


Figure 5.6: Using the IBVS joint-space trajectory to select candidate visible vertices while reducing joint-space motion.

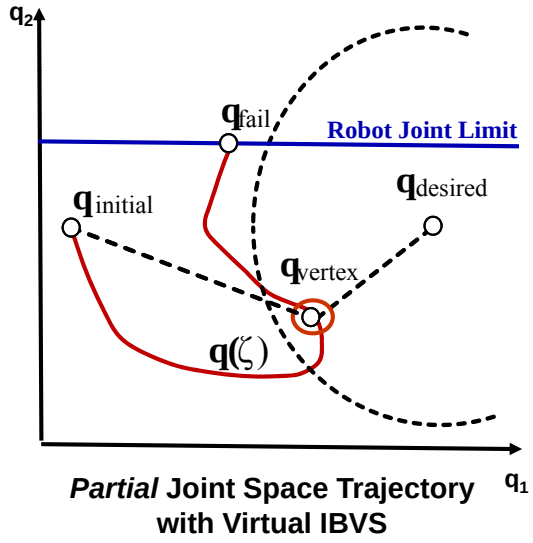


Figure 5.7: Using a *partial* IBVS joint-space trajectory to select candidate visible vertices while reducing joint-space motion.

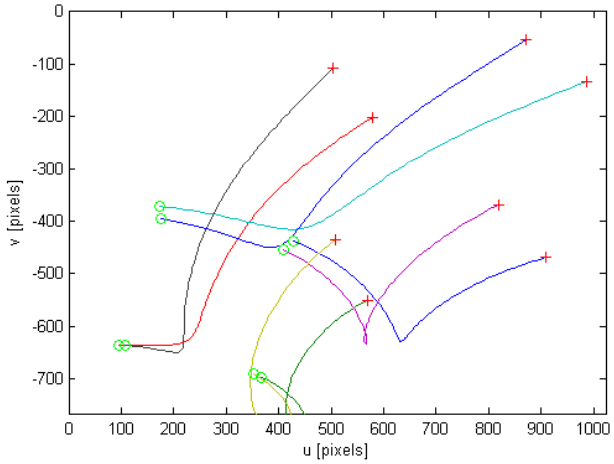


Figure 5.8: Image trajectories resulting from direct joint-space interpolation: target object partially exits the field of view

best keeps the target with the camera's field of view during *most* of the robot motion is shown in Figure 5.8. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. Note that several feature trajectories exit the field of view at the bottom of the image. Virtual IBVS is applied to the target object to generate visible robot configurations for vertices selection in PRM planning. The corresponding IBVS image trajectory is shown in Figure 5.9. All features trajectories remain inside the field of view by the design of the control. Using the vertices selection method outlined in Section 5.6.3, a visible robot configuration from the IBVS trajectory is chosen as a vertex for path planning. This vertex brings the target object back into the field of view. In this example, the chosen vertex also provides continuous target visibility during the interpolated robot motion between the three robot configurations (start, intermediate, end). The image trajectory resulting from the insertion of this intermediate robot configuration is shown in Figure 5.10.

5.8 Summary

The use of path planning is discussed in this chapter, as a prerequisite to visual servoing, for managing the multiple constraints that an eye-in-hand

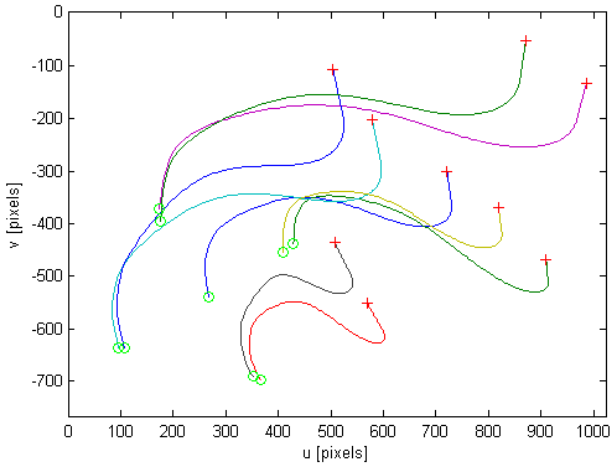


Figure 5.9: Image trajectories resulting from virtual IBVS: target object stays within the field of view

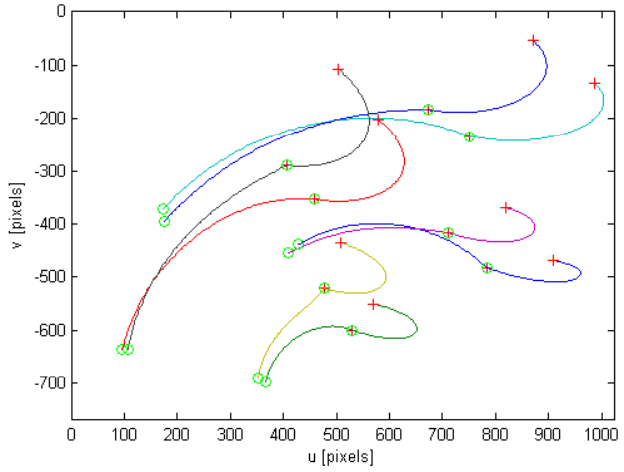


Figure 5.10: Image trajectories resulting from the insertion of a visible vertex, chosen from the virtual IBVS trajectory using the proposed joint-space metric: target object stays within the field of view.

system may encounter when the required robot motion is large. A two-stage approach is presented for planning and servoing. Path planning is used to address collision and joint-limit constraints, while visual servoing is used at the end of the planned motion to correct for uncertainties. Since visual servoing can only be initiated with the target object in view, the planned path must continuously keeps the target object within the camera's field of view to allow a feasible transition. In the presence of modelling errors, continuous visibility allows the robot controller to switch to visual servoing to "close the loop" to the end goal, in the event that the target object unexpectedly begins to leave the camera's field of view during the execution of the planned path. A *visible* robot configuration is defined in this chapter as a robot configuration that positions the target object of interest within the camera's field of view.

Path planning must occur in the robot's joint-space, rather than in image-space, in order to accurately capture the constraints related to workspace collision and robot joint-limits. Selection of an appropriately *visible* inverse kinematic solution allows vision-guided path planning to occur in the joint-space of the robot. Probabilistic roadmaps (PRM) provide an efficient framework for path planning in the presence of multiple constraints. For a target object in any given pose, a method of generating new vertices in the PRM that correspond to *visible* robot configurations is presented. These vertices act as location biases for further configuration sampling in the surrounding joint-space region if collision constraints are not met. This method of vertices generation is inspired by image-based visual servoing (IBVS), and it uses a joint-space metric to select *visible* robot configurations that also bring the robot closer to its end goal configuration. Simulations show that the insertion of a small number of *visible* intermediate robot configurations provide the sufficient conditions to ensure continuous target visibility throughout the entire planned path. The resulting continously visible path allows a feasible transition from planned motion to visual servoing.

In summary, the idea behind this two-stage approach is to execute as much of the positioning task as possible using planned joint-space motion, so that the remaining visual servoing motion can be kept to a minimum, thus reducing unexpected motions that may violate robot joint constraints. However, two problems still exist in this approach:

1. If the target object exits the field of view early on during the planned path (due to an extremely poor initial estimate of the target object's pose), then all path planning is voided, since the remaining robot motion must be executed using visual servoing;

2. When standard visual servoing techniques are used at the end of the planned motion to "close the loop" to the end goal, there is still no guarantee that the resulting motion will not violate constraints.

These two issues are addressed in the next chapter, using a visual servo control law based on Model Predictive Control (MPC).

Chapter 6

Model Predictive Control

6.1 Introduction

In the previous chapters, it was shown that pose estimation coupled with an inverse kinematic solution that satisfied path visibility constraints, allowed a part of the visual servoing task to be planned in joint-space to address robot-related constraints. One drawback with the proposed two-stage approach was that the uncertainty in the pose of the target object makes it impossible to predict how much of the vision-guided task will be executed using planned motion and how much will be executed using visual servoing. More importantly, despite the constant availability of visual feedback and availability of constraint-based models, errors still remained uncorrected throughout the first stage (planned motion), and avoidance of constraints was still not guaranteed in the second stage (visual servoing).

This chapter presents a novel visual servo control law, based on Model Predictive Control (MPC), to address the deficiencies of previous planning and servoing approaches. MPC visual servoing provides a unified framework for managing both camera-related (field of view) and robot-related (joint-limit, joint-velocity, whole-arm collision) constraints associated with eye-in-hand robotic systems. MPC solves the problem of joint-space path planning and eye-in-hand visual servoing simultaneously, by formulating the visual servoing task as a *finite horizon, open-loop, optimal control problem with constraints* that is solved online. At each control iteration, visual feedback is used to compensate for errors and to refine the prediction model. The iterative process of planning and servoing in MPC results in trajectories that are far less sensitive to the propagation of modelling errors, compared with the two-stage approach. The MPC framework produces a truly constraint-aware visual servo control law, which can be used to drive large-range robot motions to achieve close-loop positioning.

Although the use of predictive control for visual servoing has been proposed recently in [45], [46], [47], none of these methods address the problem of constraint avoidance for eye-in-hand visual servoing. To the author's knowledge, the use of MPC for simultaneous visual servoing and joint-space

planning to avoid whole-arm collisions (amongst other system constraints related to an eye-in-hand robot) is a first contribution in the field. This chapter also presents original work in methods for addressing uncertainty in the target object's location when using vision-guided path planning methods (such as PRM, as presented in Chapter 5) with MPC.

This chapter opens with a basic review of the MPC formulation for the control of a multiple-input multiple-output (MIMO) system. Following, the modelling of the system and the implementation of a MPC controller for eye-in-hand visual servoing is described. Simulation results are given for an eye-in-hand positioning task to demonstrate the successful avoidance of constraints during MPC visual servoing. The effects of the various tuning parameters are discussed. The next section describes a method which exploits the DCC algorithm (previously discussed in 5.5.2) to obtain a useful representation of the collision-free space for MPC visual servoing. Integration of MPC with PRM, including a method to address uncertainty in the target object's pose is presented next. Integration of MPC with DCC to arrive at an online iterative planning and servoing method is also presented. Experimental results are given for an eye-in-hand visual servoing task in the presence of workspace obstacles and uncertainty in the target object's pose to demonstrate the efficiency of the approach.

6.2 MPC Formulation

The basic idea of MPC is to use the predictive power of a model to choose a sequence of control actions that is optimal (according to some criterion) for achieving a desired output, while compensating for any prediction errors using feedback generated from the observation of the real plant. MPC is formulated to solve, online, a finite horizon, open-loop, optimal control problem. This optimization is subject to the system dynamics and the constraints related to the states and inputs of the system. An illustration of the basic principle of MPC is shown in Figure 6.1.

The MPC problem for computing an optimal control sequence is formulated as follows:

$$\min_{\mathbf{u}_{i|k}} \quad \mathcal{C}(\boldsymbol{\epsilon}_{i|k}, \mathbf{u}_{i|k}), \tag{6.1}$$

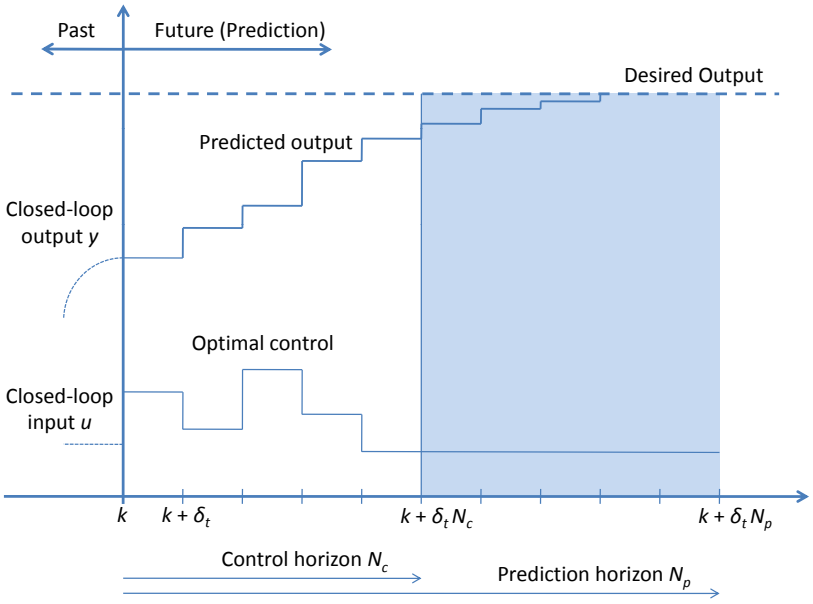


Figure 6.1: The basic principle of model predictive control.

subject to

$$\mathbf{x}_{i+1|k} = \mathbf{f}(\mathbf{x}_{i|k}, \mathbf{u}_{i|k}) \tag{6.2}$$

$$\boldsymbol{\epsilon}_{i|k} = (\mathbf{y}(\mathbf{x}_{i|k}) - \mathbf{y}^d) + \mathbf{d}_k, \qquad i \in [1, N_p] \tag{6.3}$$

$$\mathbf{x}_{i|k} \in \mathbb{X}, \qquad i \in [1, N_p] \tag{6.4}$$

$$\mathbf{u}_{i|k} \in \mathbb{U}, \qquad i \in [1, N_c] \tag{6.5}$$

where

$$\mathbf{x}_{0|k} = \mathbf{x}_k \tag{6.6}$$

$$\mathbf{u}_{i|k} = \mathbf{u}_{N_c|k}, \qquad \forall i \geq N_c \tag{6.7}$$

$$\mathbf{d}_k = \mathbf{y}(\mathbf{x}_{k|k-1}) - \mathbf{y}_k \tag{6.8}$$

At each iteration k, the MPC controller uses an internal dynamic model (Equation 6.2) to predict the behaviour of the plant over a future prediction horizon N_p of fixed length. The predicted behaviour depends on the current state $\mathbf{x}_k$ and on the assumed sequence of inputs $\mathbf{u}_{i|k}$ that are applied over a control horizon N_c, where $\mathbb{X}$ and $\mathbb{U}$ are the sets of feasible inputs and

states, respectively. The idea is to select the sequence of inputs that best forces the predicted outputs $\mathbf{y}(\mathbf{x}_{i|k})$ to converge to the desired set-point $\mathbf{y}^d$, subject to a set of constraints related to the plant. When evaluating the sequence of prediction errors $\boldsymbol{\epsilon}_{i|k}$, a disturbance (model) correction term $\mathbf{d}_k$ is applied to capture the discrepancy between the latest plant output $\mathbf{y}_k$ and the latest model output $\mathbf{y}(\mathbf{x}_{k|k-1})$. Once an optimal sequence of inputs is chosen, only the first element of the sequence is applied as the input signal to the plant. The whole cycle of output measurement, prediction, and input trajectory selection is repeated one sampling interval later. The prediction horizon remains the same length as before, but slides along by one sampling interval.

The main advantage of MPC lies in its ability to routinely handle constraints. MPC is the only advanced control technique (that is, more advanced than standard PID control) to have had a significant impact in industrial process control, especially where control update rates are infrequent and operation near constraints is necessary to obtain efficient processes. However, it should be noted that proof of closed-loop stability of MPC control of a nonlinear system is still an open research problem. Further details on MPC theory and analysis can be found in [55].

6.3 Eye-in-Hand Visual Servoing using MPC

The MPC framework is implemented to produce a constraint-aware control law for eye-in-hand visual servoing. This section describes the various components of the MPC visual servoing controller, including the modelling (robot model, camera model, object model), the constraints of the system, and the cost function to be optimized.

6.3.1 System Modelling

The following shorthand notation is used to refer to coordinate frames that are used for system modelling:

- Robot base frame, $\mathcal{F}_b$;
- Robot end-effector frame, $\mathcal{F}_e$;
- Camera frame, $\mathcal{F}_c$;
- Target object frame, $\mathcal{F}_o$.

Object Model

The object model consists of number of feature points, whose coordinates $({}^{o}X_j, {}^{o}Y_j, {}^{o}Z_j)$ are define with respect to a canonical object frame. For a target object made up of n feature points, the object model used for prediction is:

$$ {}^{o}\mathbf{P}_j = \begin{bmatrix} {}^{o}X_j \\ {}^{o}Y_j \\ {}^{o}Z_j \\ 1 \end{bmatrix}, \qquad j \in [1, n]. \tag{6.9} $$

Let ${}^{b}\mathbf{T}_o$ define the homogeneous transformation that expresses the coordinates of the object frame in the coordinates of the robot base frame. Alternatively, where it is more convenient for eye-in-hand pose estimation, ${}^{c}\mathbf{T}_o$ can be used to define the homogeneous transformation that expresses the coordinates of the object frame in the coordinates of the camera frame.

Robot Model

A forward kinematic model of the robot is used to describe the homogeneous transformation ${}^{b}\mathbf{T}_e$ that expresses the coordinates of the end-effector frame in the coordinates of the robot base frame. ${}^{b}\mathbf{T}_e$ is a function of robot joint positions:

$$ {}^{b}\mathbf{T}_e = f_{\text{robot}}(\mathbf{q}). \tag{6.10} $$

For MPC controller design, it is assumed that a low-level controller exists to stabilize the internal dynamics of the robot. This is achieved through the control of motor torques to satisfy the generalized equations of motion for robots with revolute joints and serial links:

$$ \boldsymbol{\tau} = \mathbf{D}_{\mathbf{q}}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}_{\mathbf{q}}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}). \tag{6.11} $$

For industrial robots with highly-geared joints to reject external torque disturbances, independent PID controllers are used to generated the appropriate torque τ to track a smooth position reference signal for each joint. For proposed service robots, which have back-drivable joints that are not geared (for safety reasons), an inverse dynamic model of the robot is used to provide feedback linearization. Within this context, the objective of the MPC controller is to generate feasible robot trajectories in the form of joint positions, velocities, or accelerations that efficiently drive the robot towards

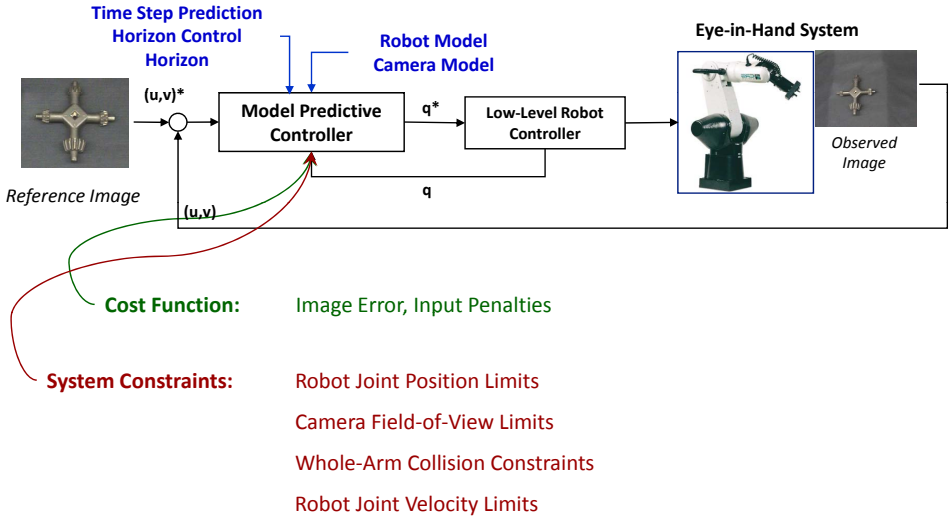


Figure 6.2: A system diagram illustrating the input and output requirements of the MPC controller.

completion of the positioning task using visual feedback. These joint commands generated by MPC are used as reference signals for the low-level torque controllers to track. A system diagram for each of the controllers described is shown in Figure 6.2. Taking into account the low frame-rate of the camera with respect to the high bandwidth of the inner stabilization loop, most visual servoing researchers use a kinematic model for controller design. The following first-order discretization is used for MPC prediction:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \delta_t \dot{\mathbf{q}}_k. \tag{6.12}$$

Camera Model

A classical pin-hole camera model is used to describe the projection of the target object on the camera CCD array. Let $\mathbf{C}$ be the camera matrix defined as:

$$\mathbf{C} = \begin{bmatrix} fk_u & fk_u \cot\beta & u_0 \\ 0 & fk_v(\frac{1}{\sin\beta}) & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.13}$$

where (u_0, v_0) are the images coordinates of the principal point, f is the focal length, β is the perpendicular skew angle, and k_u and k_v are the number of pixels per unit distance in x and y, respectively. The camera is mounted on the robot's distal link, where the position and orientation of its canonical frame is controlled by the robot. Let ${}^{e}\mathbf{T}_{c}$ define the homogeneous transformation that expresses the coordinates of the end-effector frame in the coordinates of the camera frame. The images coordinates (u_j, v_j) of the feature points can be modelled as:

$$(\mathbf{p}_j) = \begin{pmatrix} u_j \\ v_j \\ 1 \end{pmatrix} = \mathbf{C}[\mathbf{I}_{3\times3}|\mathbf{0}_{3\times1}]{}^{c}\mathbf{T}_{e}{}^{e}\mathbf{T}_{b}{}^{b}\mathbf{T}_{o}{}^{o}\mathbf{P}_{j}, \quad j \in [1, n]. \tag{6.14}$$

Finally, a vector consisting of the image coordinates of all feature points is used for image-based feedback control:

$$\overline{\mathbf{p}} = \begin{bmatrix} (\mathbf{p}_1)^T & (\mathbf{p}_2)^T & \cdots & (\mathbf{p}_n)^T \end{bmatrix}^T. \tag{6.15}$$

6.3.2 Constraint Modelling

Robot Joint-Limit Constraints

For a robot with N independent joint actuators, the robot joint-limit constraints are modelled as lower bounds $\mathbf{q}^{min}$ and upper bounds $\mathbf{q}^{max}$ on the range of feasible joint positions:

$$\mathbf{q} \in [\mathbf{q}^{min}, \mathbf{q}^{max}], \qquad \mathbf{q}^{min}, \mathbf{q}^{max} \in \mathbb{R}^N. \tag{6.16}$$

Robot Joint-Velocity Constraints

For a robot with N independent joint actuators, the robot joint-velocity constraints are modelled as lower bounds $\dot{\mathbf{q}}^{min}$ and upper bounds $\dot{\mathbf{q}}^{max}$ on the range of feasible joint velocities:

$$\dot{\mathbf{q}} \in [\dot{\mathbf{q}}^{min}, \dot{\mathbf{q}}^{max}], \qquad \dot{\mathbf{q}}^{min}, \dot{\mathbf{q}}^{max} \in \mathbb{R}^N. \tag{6.17}$$

Camera Field-of-View Constraints

For a camera with a rectangular imaging sensor array, whose maximum and minimum coordinates are u^{min} and u^{max} in the horizontal axis and v^{min} and v^{max} in the vertical axis, the camera field-of-view constraints for a target object with n features are:

$$u_j(\mathbf{p}_j) \in [u^{min}, u^{max}] \qquad \forall j \in [1, n], \tag{6.18}$$

$$v_j(\mathbf{p}_j) \in [v^{min}, v^{max}] \qquad \forall j \in [1, n]. \tag{6.19}$$

Whole-Arm Collision Constraints

For a robot with N independent joint actuators, let $\mathbb{Q}_{free}$ be the collision-free space, where $\mathbb{Q}_{free} \in \mathbb{R}^N$. $\mathbb{Q}_{free}$ is the set of joint configurations which *do not* cause the robot to occupy the same physical space as a workspace object. In this context, whole-arm collision constraints are represented as:

$$\mathbf{q} \in \mathbb{Q}_{free}. \tag{6.20}$$

A exact closed-form solution to $\mathbb{Q}_{free}$ is typically not available, except in the case of trivially simple workspace objects (such as points) with kinematically simplified robots (such as Cartesian gantry robots, or robots where N is limited). The discussion of a representation of collision-free space that is useful for MPC visual servoing is defered until Section 6.6, where it is discussed within the context of the dynamic collision checker (DCC).

6.3.3 Control Law Design

The error function for MPC visual servoing is chosen to be similar in form to IBVS. At time k, the predicted error at time $k+i$ is the difference between the predicted image coordinates $\overline{\mathbf{p}}_{i|k}$ and the desired image coordinates, plus a disturbance (model) correction term $\mathbf{d}_k$. The correction term is also expressed in terms of image errors, capturing the difference between latest plant output $\overline{\mathbf{p}}_k$ and the latest model output $\overline{\mathbf{p}}(\mathbf{q}_{k|k-1})$:

$$\boldsymbol{\epsilon}_{i|k} = (\overline{\mathbf{p}}_{i|k} - \overline{\mathbf{p}}^d) + \mathbf{d}_k, \tag{6.21}$$

$$\mathbf{d}_k = \overline{\mathbf{p}}(\mathbf{q}_{k|k-1}) - \overline{\mathbf{p}}_k. \tag{6.22}$$

The control law for MPC visual servoing is designed to take advantage of the robustness of IBVS for error correction while penalizing large joint-space motions to give efficient trajectories. Similar to the visible vertex generation and selection algorithm presented in Chapter 5, the idea is to take advantage of the natural ability of IBVS to keep the target within the field of view (to avoid local minimas resulting from the nonlinear field-of-view constraints)

while minimizing joint motion. The cost function $\mathcal{C}$ to be minimized is a quadratic function of image errors $\boldsymbol{\epsilon}$ and of joint velocities $\mathbf{q}$:

$$\mathcal{C} = \frac{1}{2}\left(\sum_{i=1}^{N_p-1} \boldsymbol{\epsilon}_{i|k}^T \mathbf{Q} \boldsymbol{\epsilon}_{i|k} + \dot{\mathbf{q}}_{i|k}^T \mathbf{W} \dot{\mathbf{q}}_{i|k}\right), \tag{6.23}$$

where $\mathbf{Q}$ and $\mathbf{W}$ are two symmetric positive definite matrices, weighting the predicted image errors against the predicted joint control efforts at time instant k as in [47]. The penalization of joint velocities helps to avoid joint singularities where the control and the prediction of eye-in-hand camera motion is more sensitive to errors.

6.4 Summary of the MPC Visual Servo Control Law

From the perspective of manipulator visual servoing, the proposed MPC visual servoing control law can be seen *simultaneously* as:

1. An optimal controller using IBVS with penalization on large joint motions;

2. An online open-loop motion planner that takes into account the future dynamics and constraints of the robotic system, based on an approximate pose of the target object.

The approximate object pose that is required for online open-loop planning in MPC is obtained from one of the following:

- From *the user*, in the form of a rough "ballpark" pose of the target object, when initiating the robot positioning task;

- From *homography decomposition*, in the form of scaled Euclidean parameters with an estimated depth provided by the user (Chapter 3);

- From *EKF pose estimation*, in the form a 6-D pose vector and a covariance matrix describing the uncertainty of the estimated pose in each dimension (Chapter 4).

In this formulation, the cost function can be minimized over a sequence of $\mathbf{q}_{i|k}$ or $\dot{\mathbf{q}}_{i|k}$, depending on the type of input that is accepted by the low-level robot controller. Typical industrial controllers accept only joint positions as input, so minimization over $\mathbf{q}_{i|k}$ is used for experimental validation here.

An added advantage of optimizing over $\mathbf{q}_{i|k}$ is that the sequence of planned joint configuration inputs can be checked for collision-free motion, using the DCC algorithm as outlined in Section 6.8.2. Furthermore, by using $\mathbf{q}_{i|k}$ as the input for optimization, the MPC controller can be programmed to solve off-line for the *approximate* robot joint configuration that completes the positioning task, based only on the prediction model. The MPC visual servoing formulation is summarized as:

$$\min_{\mathbf{q}_{i|k}} \frac{1}{2}\Big(\sum_{i=1}^{N_p-1} \boldsymbol{\epsilon}_{i|k}^T \mathbf{Q} \boldsymbol{\epsilon}_{i|k} + \dot{\mathbf{q}}_{i|k}^T \mathbf{W} \dot{\mathbf{q}}_{i|k}\Big), \tag{6.24}$$

subject to

$$\mathbf{q}_{i|k+1} = \mathbf{q}_{i|k} + \delta_t \dot{\mathbf{q}}_{i|k}, \tag{6.25}$$

$$\boldsymbol{\epsilon}_{i|k} = (\overline{\mathbf{p}}_{i|k} - \overline{\mathbf{p}}^d) + \mathbf{d}_k, \tag{6.26}$$

$$\dot{\mathbf{q}}_{i|k} \in [\dot{\mathbf{q}}^{min}, \dot{\mathbf{q}}^{max}], \tag{6.27}$$

$$\mathbf{q}_{i|k} \in [\mathbf{q}^{min}, \mathbf{q}^{max}], \tag{6.28}$$

$$\mathbf{q}_{i|k} \in \mathbb{Q}_{free}, \tag{6.29}$$

$$u_j((\mathbf{p}_j)_{i|k}) \in [u^{min}, u^{max}], \qquad j \in [1, n] \tag{6.30}$$

$$v_j((\mathbf{p}_j)_{i|k}) \in [v^{min}, v^{max}], \qquad j \in [1, n] \tag{6.31}$$

$$\tag{6.32}$$

where

$$\mathbf{q}_{0|k} = \mathbf{q}_k, \tag{6.33}$$

$$\overline{\mathbf{p}}_{i|k} = \begin{bmatrix} (\mathbf{p}_1)_{i|k}^T & (\mathbf{p}_2)_{i|k}^T & \cdots & (\mathbf{p}_n)_{i|k}^T \end{bmatrix}^T, \tag{6.34}$$

$$(\mathbf{p}_j)_{i|k} = \mathbf{C}[\mathbf{I}_{3\times3}|\mathbf{0}_{3\times1}]{}^c\mathbf{T}_e\, {}^e\mathbf{T}_b\, {}^b\mathbf{T}_o\, {}^o\mathbf{P}_j, \qquad j \in [1, n] \tag{6.35}$$

$${}^e\mathbf{T}_b = f_{\text{robot}}(\mathbf{q}_{i|k}), \tag{6.36}$$

$$\mathbf{d}_k = \overline{\mathbf{p}}(\mathbf{q}_{k|k-1}) - \overline{\mathbf{p}}_k. \tag{6.37}$$

6.5 Simulation Results

6.5.1 Setup

The MPC eye-in-hand visual servoing controller is implemented in simulation to test its ability to perform large-motion robot positioning tasks in the presence of constraints and pose estimation errors. The simulations are

designed to reflect, as closely as possible, the parameters of the actual experimental test-bed that is used for performing eye-in-hand visual servoing experiments. In order to avoid repeating the same material, details on the simulation model (and on the experimental test-bed) can be found in Section 6.9.1. The minimization of the cost function for predictive control is solved using a sequential quadratic programming (SQP) optimization algorithm [56][55]. This method solves a quadratic programming (QP) subproblem at each iteration using the method of active sets. To speed up computations, the gradients to the cost function and to the constraints are provided via a closed-form analytic function composed of the robot Jacobian, the image Jacobian and the measured errors. In the simulations, the *actual* pose of the target object is:

$$^{b}\mathbf{T}_{o} = \mathbf{T}_{xyz}(-0.28, 0.51, 0.14)\mathbf{T}_{Rz}(145^{\circ})\mathbf{T}_{Rx}(-55^{\circ})[m],$$

whereas the *estimated* pose of the target object used by the MPC controller for joint-space planning, is:

$$^{b}\mathbf{T}_{\hat{o}} = \mathbf{T}_{xyz}(-0.30, 0.50, 0.15)\mathbf{T}_{Rz}(135^{\circ})\mathbf{T}_{Rx}(-45^{\circ})[m].$$

A desired view of the target object is shown to the robot to describe the required positioning task, and the robot starts the visual servoing task with an initial view containing the target object in sight. The initial view and desired view corresponding to the positioning task are shown in Figure 6.3 and 6.4, respectively. The prediction horizon N_p is set to 5 and the control horizon N_c is set to 1. The time step δ_t of 10ms is used for simulations. $\mathbf{Q}$ is chosen to be the identity matrix, while $\mathbf{W}$ is chosen to be a diagonal matrix with its diagonal entries equal to 2,500. Note that collision constraints are not modelled in these simulations. The purpose of these simulations is to demonstrate the ability of MPC visual servoing to manage constraints that are represented in the controller: field-of-view constraints, joint-limit constraints and joint-velocity constraints. An appropriate representation of collision constraints $\mathbf{q}_{i|k} \in \mathbb{Q}_{free}$ that is useful for MPC visual servoing will be discussed in Section 6.6. The demonstration of collision avoidance with MPC visual servoing will be shown in experiments with a physical robot setup in Section 6.9.2).

6.5.2 Camera Field-of-View Constraints

Figure 6.5 shows the trajectory of the target object (as seen by the eye-in-hand camera) for the positioning task achieved via MPC visual servoing.

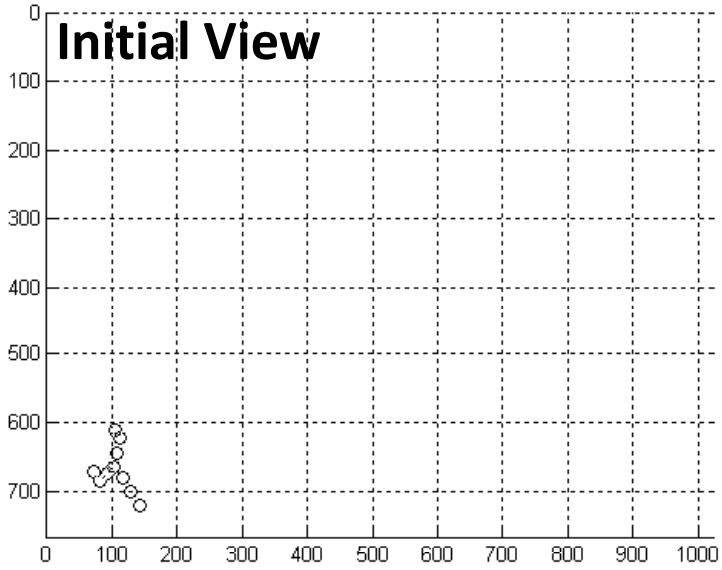


Figure 6.3: Overhead image describing the initial position of the eye-in-hand camera with respect to a target object.

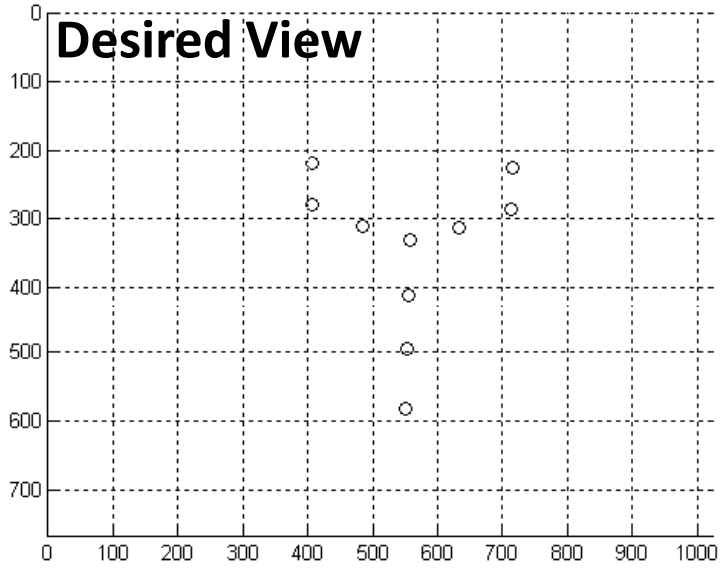


Figure 6.4: Reference image describing the desired position of the eye-in-hand camera with respect to the target object.

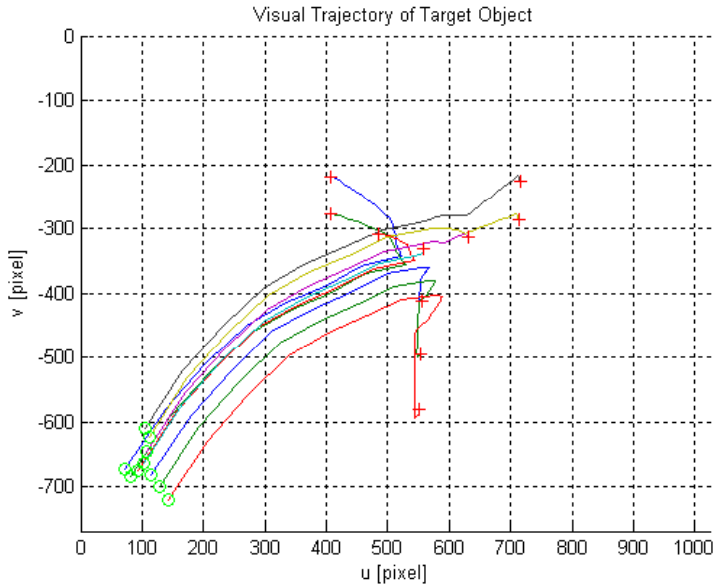


Figure 6.5: Image trajectory generated by MPC visual servoing, demonstrating the avoidance of camera field-of-view limits and the completion of the visual servoing task

The target object, which resembles the conrod shown in Figure 6.18, consists of ten non-coplanar feature points. The point 'o' designates the start of the motion and the point 'x' designates the end of the motion. Through the use of an image-based cost function with an eye-in-hand robot configuration, the control law generates motions that naturally satisfy the camera's field-of-view constraints. In contrast, this is not true for a stationary camera observing a robot completing a position task. Figure 6.5 shows the completion of the task in Cartesian space. The positioning task requires significant translation towards the target object, and additional in-plane and out-of-plane rotations to correct for the orientation errors. All of the above is completed in an efficient manner, without camera retreat, despite the use of an image-based control law. With a high visual update rate, the prediction model is quickly corrected to account for the initial pose estimation errors.

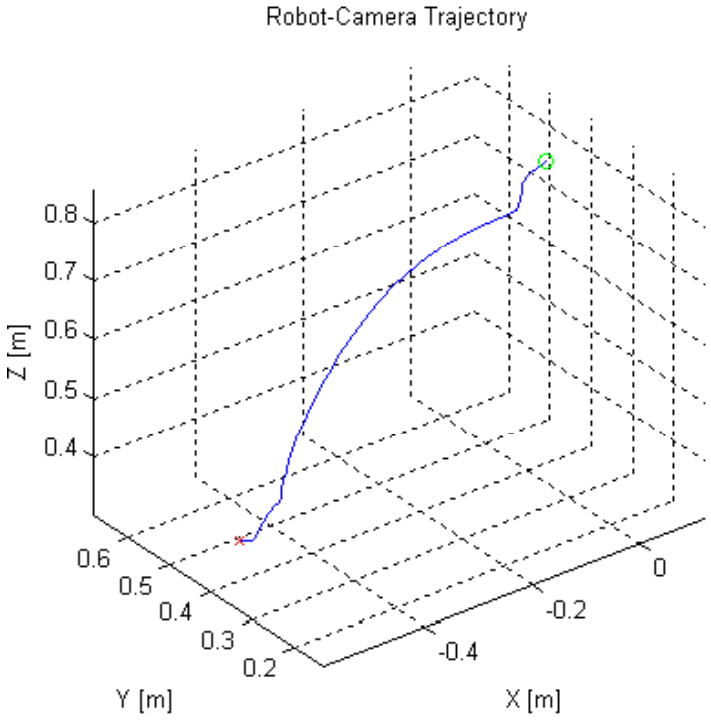


Figure 6.6: Camera Cartesian trajectory generated by MPC visual servoing, demonstrating reasonable camera motions.

6.5.3 Robot Joint-Limit Constraints

Figure 6.7 shows the joint-space trajectory of the 6-DoF CRS-A465 robot for the positioning task achieved via MPC visual servoing. The joint limits of the robot are shown as dotted lines, where as the joint trajectories are shown as solid lines. The joint trajectories respond to the robot's joint limits by approaching them conservatively, while not violating them. The result is that the robot's joint-limit constraints are always satisfied throughout the servoing motion. Note that the avoidance of joint-limit constraints cannot be naively achieved with the use of a joint-position saturator. The difference between the constraint-aware MPC motion and a simple joint-position saturator is that the MPC motion continues to complete the task while managing constraints, while a joint-position saturator will likely bring the target object outside of the field of view, causing instability in the control law.

6.5.4 Robot Joint-Velocity Constraints

Figure 6.8 shows the joint velocities of the 6-DoF CRS-A465 robot for the positioning task achieved via MPC visual servoing. The velocity limits of the robot are shown as dotted lines, where as the actual joint velocities are shown as solid lines. Unlike the exponentially decreasing velocity profiles observed in other visual servoing control laws (which tend to violate joint velocity limits. See Figure 3.4), MPC visual servoing makes maximum use of the robot's output capabilities by keeping the joint velocities near their limits, while not violating them. This is a rather aggressive controller. The insufficient weight given to the velocity weighting matrix $\mathbf{W}$ allows the magnitude of pixel errors to dominate the magnitude of joint velocities in the cost function. Note that the avoidance of velocity-limit constraints cannot be naively achieved with the use of a joint-velocity saturator. The difference between the constraint-aware MPC motion and a simple joint-velocity saturator is that the MPC motion continues to complete the task while managing constraints, while a joint-velocity saturator will likely bring the target object outside of the field of view, causing instability in the control law. The joint velocities still tend to change abruptly in the given simulations. Where acceleration limits are available for the robot, a higher-order model can be used in the MPC controller, to provide a smoother trajectory for tracking. This is not a major concern in the actual experimental implementation for visual servoing, since the update rate of the camera is limited, and a trajectory interpolator must be used, in any case, to match the update rate of the inner PID loop.

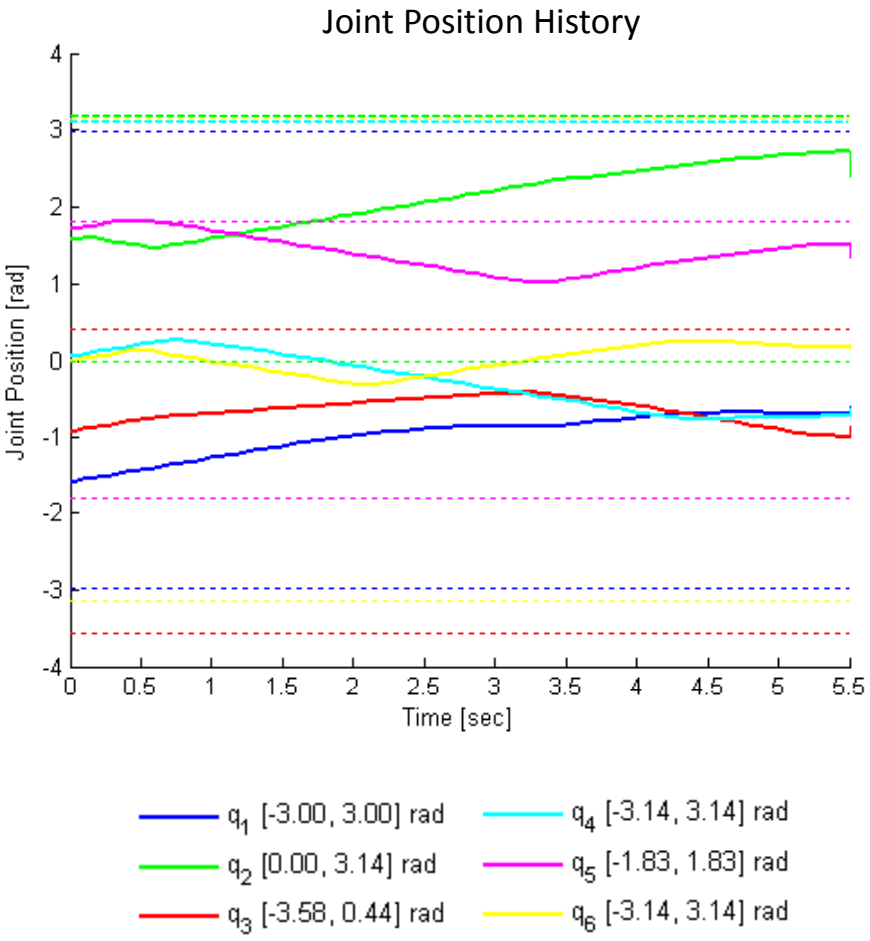


Figure 6.7: Robot joint trajectory generated by MPC visual servoing, demonstrating the avoidance of robot joint position limits.

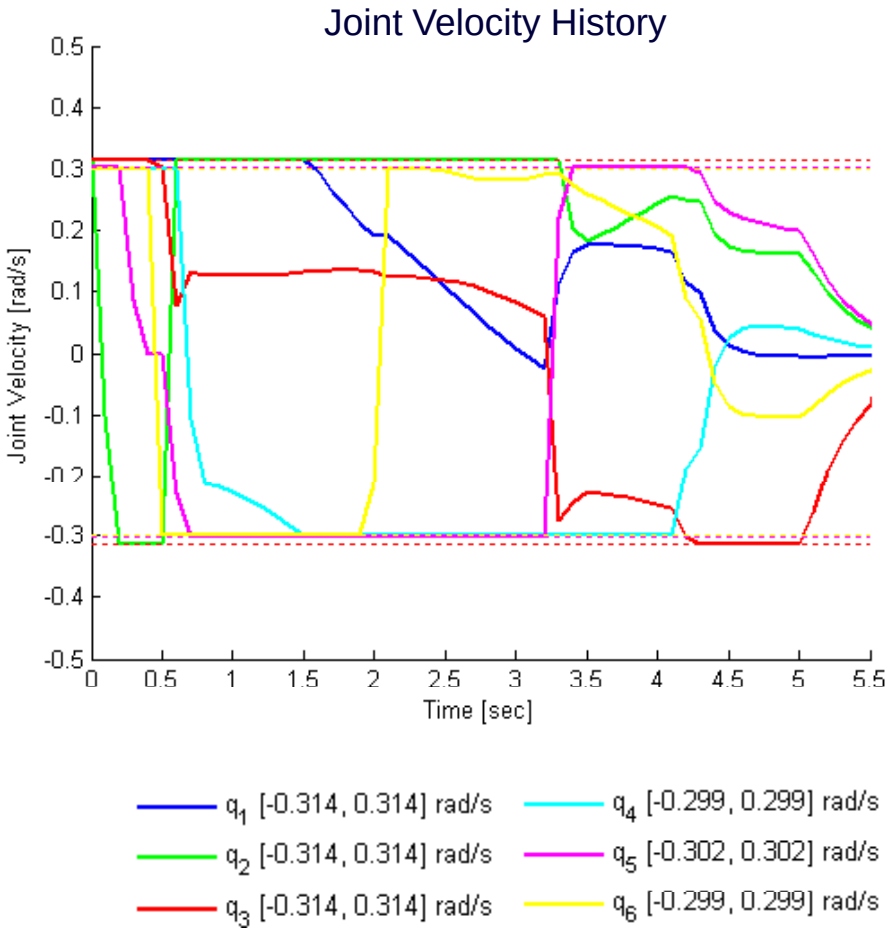


Figure 6.8: Robot joint velocities generated by MPC visual servoing, demonstrating the avoidance of robot joint velocity limits.

6.5.5 Tuning Parameters

The following section describes insights on tuning parameters obtained from simulations.

Switching Control vs. MPC

The problem with expressing robot joint-limits, field-of-view limits, collision limits, as *explicit* constraints in the form of a *typical switching visual servo control law* is that the controller is not aware of the constraints until they are active. The need to avoid immediate constraints requires large controller efforts. Since the controller does not plan ahead to avoid constraints, many constraints will indeed become active, so the modelled constraints must closely match the boundaries of the real limits of the plant in order to guarantee safe operation. In the presence of nonlinear constraints, without planning, the system trajectory can easily get stuck in a local minima within an infeasible region.

The motivation for predictive control is to allow the controller to plan ahead to compute the best sequence of control inputs to avoid these constraints, while completing the task. Although many of the constraints will become active *in the prediction model*, the control input produced by the predictive controller that is applied to the plant several steps ahead may actually prevent them from becoming active *in the real plant*. The prediction in MPC allows significant margins of errors in the modelling of the plant for control. The MPC visual servoing framework allows gains to be tuned to match the controller with the degree of uncertainty in the model (i.e. robot calibration, camera calibration, target object pose). For example, the prediction horizon is a gain that can be tuned to control the aggressiveness of the control scheme to match the precision of the model that is used for planning. If the model is precise, then a very aggressive control scheme can be used (short prediction horizon, long time step). The MPC controller can be interpreted as a constraint-aware switching controller in the limiting case where the prediction horizon and control horizon are zero. The following section describes the effect of each tuning parameter within the MPC visual servoing controller when the cost function is optimized over a sequence of joint positions $\mathbf{q}_{i|k}$.

Prediction Horizon

The length of the prediction horizon N_p determines the aggressiveness of the control law with respect to constraints. When the prediction horizon is long,

the controller predicts further ahead to become aware of constraints that may become active in the near future. The controller reacts by producing control actions that avoid those constraints, many steps before they actually become active in the real plant. The result is a conservative control law that maintains a wide distance from constraints. The prediction horizon also affects the control efforts, since it is typically chosen to be much longer than the control horizon. For a prediction horizon that is longer than a given control horizon, the last control input is held constant for the remainder of the prediction horizon. Therefore, a very long prediction horizon tends to produce more conservative control efforts.

Control Horizon

The length of control horizon N_c determines the degrees of freedom that the controller has, for choosing how a task is to be completed. When the control horizon is long, the controller can afford to be more aggressive in minimizing the image errors *when required*, without fear of violating constraints. But the controller can also afford to be more conservative *when required*, so as to minimize joint actuation efforts in the cost function, when given the option to complete the task in several steps with different control inputs. The control horizon is typically chosen to be much shorter than the prediction to increase the speed of computations. When the prediction horizon is longer than the control horizon, the last control input is held constant for the remainder of the prediction horizon. A short control horizon (in the presence of a much longer prediction horizon) tends to produce more conservative control efforts, since the last control input must be held constant for the remainder of the prediction horizon without overshooting the goal, or violating constraints.

Time step

The length of the time step determines the aggressiveness of the control law with respect to the certainty in the target object's pose. The length of the time step is inversely proportionally to the frequency of the visual update. When the time step is large, the controller has the opportunity to execute large robot motions in the absence of feedback correction (since planning and servoing are executed iteratively). In the extreme case where the time step *very* large, the controller can be programmed to execute the position task in a one-step open-loop motion. This is analogous to an open-loop constraint-aware path planner. However, this one-step planner suffers the same problem as the PRM planner presented in Chapter 5. When the

pose of the target object is uncertain, a large time step tends to bring the target object outside of the field of view, before close-loop corrections can have any effect on the prediction model. MPC visual servoing addresses this field-of-view problem by allowing the frequency of re-planning and servoing to be tuned (by adjusting the length of the time step) to match the level of uncertainty in the target object's pose.

Cost Function: Q and W

The weighting matrices **Q** and **W** determine the aggressiveness of the control law with respect to the servoing task and control efforts, as inspired from LQR (Linear Quadric Regulator) control theory. The elements with the matrix **Q** determine the weight given to the errors associated with each image feature, while the elements with the matrix **W** determine the weight given to the velocities associated with each robot joint.

6.6 Whole-Arm Collisions Constraints

Performance evaluations of the proposed MPC visual servoing controller (Section 6.5) shows that it is capable of visual servoing over large robot motions without violating the constraints related to an eye-in-hand system. System limits can be avoided as long as the corresponding constraint is included the model. The following section describes the modelling of whole-arm collisions constraints, which must be addressed and included in the MPC controller to ensure collision-free visual servoed motion.

6.6.1 Representation of the Collision-Free Space

The representation of whole-arm collision constraints in joint-space from known Cartesian-space geometry is a non-trivial problem. Unlike the inverse kinematic solutions for the robot configurations that solve for a Cartesian frame of a *particular* point on the robot, there is typically no closed-form solution to the set of robot configurations that are collision-free. Collisions can occur between any combination of points on the obstacle and on the robot. The collision boundary is a complex manifold that depends not only on the geometry of the obstacle, but also on the relative geometry of the robot arm. It is easy to check for the absence or presence of collision at a *particular* robot configuration, but further information regarding the topology of the manifold is difficult to obtain. Unfortunately, MPC requires the use of gradients to effectively solve the constrained optimization problem.

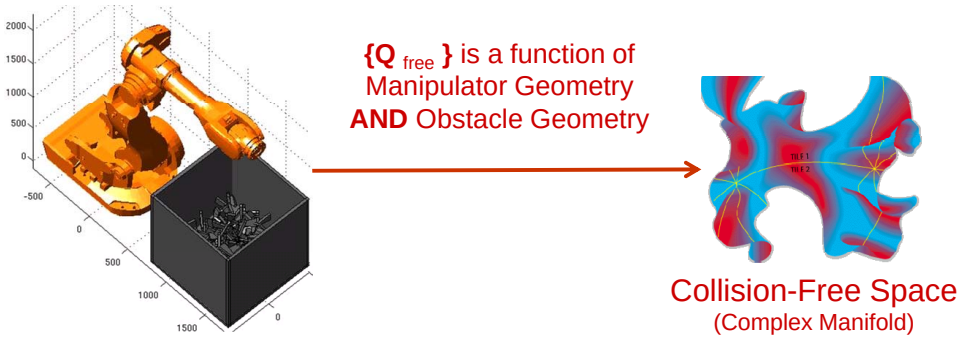


Figure 6.9: Joint-space representation of the collision-free space (right) for a particular manipulator and obstacle configuration (left). The collision-free space is represented in 3-dimensional space for 3 joints only.

Therefore, point-based representations of the collision-free space, derived from sampling robot configurations that are free of collisions, cannot be used as constraints for MPC visual servoing. Figure 6.9 depicts a hypothetical set of collision-free robot configurations, represented as a three-dimensional manifold in joint-space for 3 out of the 6 available robot joints.

6.6.2 Exploiting DCC for Collision Bounds

This section presents a method to exploit the properties of the dynamic collision checking (DCC) algorithm to obtain a closed-form *approximation* of the collision-free space, which can be used for MPC visual servoing. The DCC algorithm was previously presented in Section 5.5.2 to test for the feasibility of whole-arm collisions with obstacles during the trajectory $\mathbf{q}(t)$ between two robot configurations, $\mathbf{q}_i$ and $\mathbf{q}_f$. The DCC inequality test has the following form:

$$\ell(\mathbf{q}(t)) < d(\mathbf{q}_i) + d(\mathbf{q}_f) \tag{6.38}$$

where

- $\ell(\mathbf{q}(t))$ is the the longest distance traveled by any point on the manipulator during the trajectory $\mathbf{q}(t)$.
- $d(\mathbf{q}_i)$ is the shortest distance between an obstacle and the manipulator in the configuration $\mathbf{q}_i$.

- $d(\mathbf{q}_f)$ is the shortest distance between an obstacle and the manipulator in the configuration $\mathbf{q}_f$.

Collision-free motion can be guaranteed if the inequality in Equation 6.38 holds for a chosen trajectory $\mathbf{q}(t)$. Given a trajectory $\mathbf{q}(t)$ and the set of 3-D points $\mathbb{H}$ representing the hull of the manipulator, $\ell(\mathbf{q}(t))$ is determined by

$$\ell(\mathbf{q}(t)) = \max_{\mathbf{P}\in\mathbb{H}} \int_0^1 |\dot{\mathbf{P}}|_2 dt. \tag{6.39}$$

Solving Equation 6.39 is tedious but much can be gained given that Equation 5.13 is an inequality and it still holds if $\ell(\mathbf{q}(t))$ is replaced by an upper bound [53]. For a manipulator with N actuators, suppose that the trajectory $\mathbf{q}(t)$ only involves rotating the j^{th} actuator by an amount Δq_j. Then, it is possible to find a configuration for the actuators $q_{j+1}, \ldots, q_N$ such that

$$\begin{aligned} \ell(\mathbf{q}(t)) &\leq \max_{q_{j+1},\ldots,q_N} \ell(\mathbf{q}(t)) \\ &= \ell_j(\mathbf{q}(t)). \end{aligned}$$

If the j^{th} actuator is a revolute joint around the axis $\mathbf{z}_j$, then

$$\ell_j(\mathbf{q}(t)) = \max_{\substack{q_{j+1},\ldots,q_N \\ \mathbf{P}\in\mathbb{H}_j\cup\cdots\cup\mathbb{H}_N}} \int_0^1 |\dot{\mathbf{P}}|_2 dt \tag{6.40}$$

$$= r_j|\Delta q_j| \tag{6.41}$$

where $\mathbf{P} \in \mathbb{H}_j \cup \cdots \cup \mathbb{H}_N$ are the points on the hulls of the links j to N and r_j is the greatest possible distance between the line $\mathbf{z}_j$ and a point $\mathbf{P}$. Finally, generalizing Equation 6.40 to trajectories involving N actuators we obtain

$$\ell(\mathbf{q}(t)) \leq \sum_{j=1}^{N} r_j|\Delta q_j|. \tag{6.42}$$

Notice that the right hand side of Equation 6.42 is "trajectory-free". That is, it does not depend on $\mathbf{q}_i$, $\mathbf{q}_f$ or the trajectory $\mathbf{q}(t)$. One notable constraint is that any trajectory must be bounded within the hyperrectangle with corners $\mathbf{q}_i$ and $\mathbf{q}_f$.

Replacing Equation 6.42 in Equation 5.13 results in

$$\sum_{j=1}^{N} r_j|\Delta q_j| < d(\mathbf{q}_i) + d(\mathbf{q}_f). \tag{6.43}$$

In [53], Equation 6.43 is used to determine if a manipulator collides with obstacles during a trajectory $\mathbf{q}(t)$. The conservativeness of the upper bound, however, can be used for trajectories other than the linear interpolation defined by $\mathbf{q}(t)$. Indeed, if Equation 6.43 is true, then any trajectory bounded by the hyperrectangle with corners $\mathbf{q}_i$ and $\mathbf{q}_f$ is collision-free. This is determined from Equation 6.40 where any trajectory that is bounded between q_{j_i} and q_{j_f} will result in the same upper bound ℓ_j.

Equation 6.43 is used to find the collision-free hyperrectangle from the initial configuration $\mathbf{q}_i$. The algorithm is based on evaluating Equation 6.43 with the trajectories $\mathbf{q}(t/2^i)$ for $0 \leq i$. Starting with $i = 0$, the entire trajectory is tested. If DCC succeeds, then the entire hypperrectangle from $\mathbf{q}_i$ to $\mathbf{q}_f$ is collision-free. If it fails, then the interval is split in half and DCC is evaluated between $0 \leq t \leq 0.5$. The procedure is applied recursively until DCC succeeds, and a hypperrectangluar region of collision-free joint space is found.

The region of collision-free space between q_{j_i} and q_{j_f}, as obtained from DCC, can be expressed in the form of joint-limit constraints in the formulation of the MPC controller, such that the resulting visual servoed motion can be guaranteed to be collision-free. In the following sections, this characteristic of Equation 6.43 will be used by the MPC controller in two implementations: MPC with PRM for offline path planning and MPC with DCC for online collision-avoidance.

6.7 MPC with Offline Path Planning

6.7.1 Motivation

The ability of the MPC visual servoing controller to handle constraints suggests that it should be used in the two-stage *planning-then-servoing* approach discussed in Chapter 5 to avoid collisions during visual servoing. However, the appropriate bounds on the constraints must still be determined in order for the controller to be effective. Specifically, it is desirable to guarantee that the robot will not collide with obstacles or violate joint-limits when visual servoing is activated to "close the loop". Unfortunately, this closed-loop position is typically unknown. The next section discusses a method to determine the appropriate joint-motion constraints to be imposed on the MPC controller, given a measure of uncertainty in the object's estimated pose, to achieve collision-free closed-loop motion.

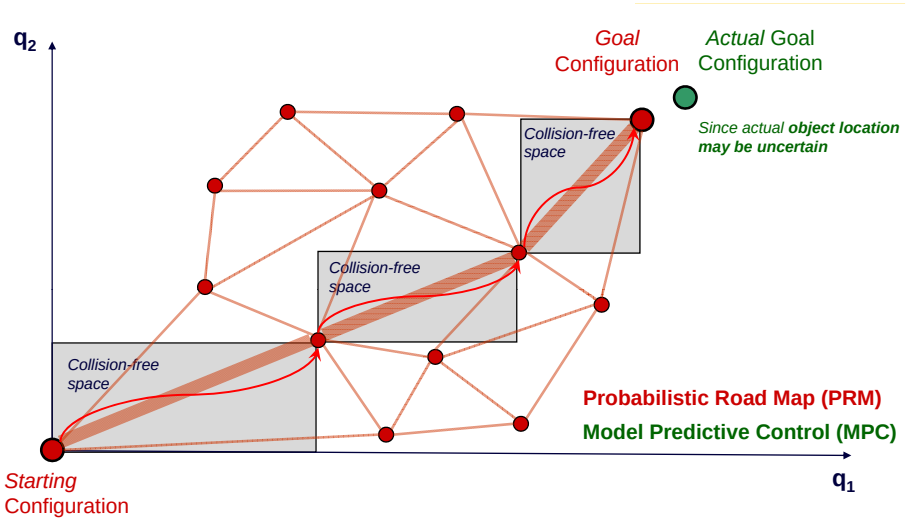


Figure 6.10: Hyperrectangular collision-free spaces obtained from DCC as a byproduct of the form chosen for $\ell(\mathbf{q}(t))$.

6.7.2 Pose Uncertainty Modelling

Let $\boldsymbol{\delta}_{xyz\phi\alpha\psi} = [\delta_x, \delta_y, \delta_z, \delta_\phi, \delta_\alpha, \delta_\psi]^T$ such that $[-\boldsymbol{\delta}_{xyz\phi\alpha\psi}, \boldsymbol{\delta}_{xyz\phi\alpha\psi}]$ represents the bounds on the uncertainty in the 6-D pose of the target object, expressed with respect to the *estimated* target object frame. If the pose variation has a Gaussian distribution, this bound can be conservatively estimated in terms of a number of standard deviations from the pose estimate, derived from the state covariance matrix from EKF pose estimation.

Let $\hat{\mathbf{q}}$ be the *estimated* robot joint position which completes the positioning task. Let $\hat{\mathbf{q}} + \delta_{\mathbf{q}}$ be the *actual* robot position which completes the positioning task, where $\delta_{\mathbf{q}}$ is the change in robot joint positions when the loop is closed. These joint positions are used to define the transformation from the robot end-effector frame to the robot base frame:

$$^b\mathbf{T}_{\hat{e}} = \mathbf{f}_{\text{robot}}(\hat{\mathbf{q}}), \tag{6.44}$$

$$^b\mathbf{T}_e = \mathbf{f}_{\text{robot}}(\mathbf{q}) = \mathbf{f}_{\text{robot}}(\hat{\mathbf{q}} + \delta_{\mathbf{q}}). \tag{6.45}$$

Let $^b\mathbf{T}_{\hat{o}}$ represent the *estimated* pose of the target object with respect to the base frame. Let $^b\mathbf{T}_o$ represent the *actual* pose of the target object with respect to the base frame. Then,

$$^{b}\mathbf{T}_{\hat{o}} = {}^{b}\mathbf{T}_{\hat{e}}\,{}^{e}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{\hat{o}}, and \tag{6.46}$$

$$^{b}\mathbf{T}_{o} = {}^{b}\mathbf{T}_{e}\,{}^{e}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{o}. \tag{6.47}$$

$$\tag{6.48}$$

Let $^{\hat{o}}\mathbf{T}_{o}$ represent the deviation of the *actual* target object frame from the *estimated* target object frame:

$$^{\hat{o}}\mathbf{T}_{o} = {}^{b}\mathbf{T}_{\hat{o}}^{-1}\,{}^{b}\mathbf{T}_{o}, \tag{6.49}$$

$$= \begin{bmatrix} c_\phi c_\alpha & c_\phi s_\alpha s_\psi - s_\phi c_\psi & c_\phi s_\alpha c_\psi + s_\phi s_\psi & {}^{\hat{o}}X_o \\ s_\phi c_\alpha & s_\phi s_\alpha s_\psi + c_\phi c_\psi & s_\phi s_\alpha c_\psi - c_\phi s_\psi & {}^{\hat{o}}Y_o \\ -s_\alpha & c_\alpha s_\psi & c_\alpha c_\psi & {}^{\hat{o}}Z_o \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.50}$$

The parameters $^{\hat{o}}X_o, {}^{\hat{o}}Y_o, {}^{\hat{o}}Z_o, {}^{\hat{o}}\phi_o, {}^{\hat{o}}\alpha_o, {}^{\hat{o}}\psi_o$ describe the deviation of the *actual* target object frame from the *estimated* target object frame, which causes a change in the robot joint positions $\boldsymbol{\delta}_{\mathbf{q}}$ when the loop is closed. These parameters can be checked to determine whether they are within the bounds defined by $\boldsymbol{\delta}_{xyz\phi\alpha\psi}$. Note that $^{c}\mathbf{T}_{o} = {}^{c}\mathbf{T}_{\hat{o}}$ when the eye-in-hand visual servoing is complete, regardless of the actual pose of the target object $^{b}\mathbf{T}_{o}$ with respect to the robot base.

6.7.3 Bounds on Closed-Loop Motion

Let $^{b}\mathbf{J}_{\hat{o}}(\hat{\mathbf{q}})$ be the Jacobian for $^{b}\mathbf{T}_{\hat{o}}$ evaluated at $\hat{\mathbf{q}}$. Then an estimate of the lower bounds and upper bounds on the change in joint positions (resulting from the closed-loop motion) can be found using the following first-order approximation:

$$\delta_{\mathbf{q}} = {}^{b}\mathbf{J}_{\hat{o}}^{-1}(\hat{\mathbf{q}})\boldsymbol{\delta}_{xyz\phi\alpha\psi}. \tag{6.51}$$

To solve for an exact solution for the lower bounds and the uppers bounds on the change in joint positions resulting from the closed-loop motion, a nonlinear constrained optimization problem is formulated and solved using the solution to linear Jacobian approximation as the initial state. This optimization is solved twice for each joint q_i (once for the lower bound and once for the upper bound):

$$\min_{\mathbf{q}} \quad \delta_{q_i}, \qquad i \in [1, N], \tag{6.52}$$

$$\max_{\mathbf{q}} \quad \delta_{q_i}, \qquad i \in [1, N], \tag{6.53}$$

subject to,

$$^{\hat{o}}X_o \in [-\delta_x, \delta_x], \tag{6.54}$$

$$^{\hat{o}}Y_o \in [-\delta_y, \delta_y], \tag{6.55}$$

$$^{\hat{o}}Z_o \in [-\delta_z, \delta_z], \tag{6.56}$$

$$^{\hat{o}}\phi_o \in [-\delta_\phi, \delta_\phi], \tag{6.57}$$

$$^{\hat{o}}\alpha_o \in [-\delta_\alpha, \delta_\alpha], \tag{6.58}$$

$$^{\hat{o}}\psi_o \in [-\delta_\psi, \delta_\psi]. \tag{6.59}$$

Therefore, bounds on the change in joint positions resulting from the closed-loop motion can be expressed in the form of a hyperrectangular region $\boldsymbol{\delta}_{\mathbf{q}} \in [\boldsymbol{\delta}_q^{min}, \boldsymbol{\delta}_q^{max}]$. Unlike the symmetric bounds estimated from the linear approximation, the nonlinear bounds describing this hyperrectangular region can be asymmetric about the estimated open-loop joint position $\hat{\mathbf{q}}$.

6.7.4 Simulation Results

Simulations were performed on a 6-DoF CRS-A465 robot with a Sony XC-HR70 camera in an eye-in-hand configuration, completing a positioning task using MPC visual servoing. The visual servoing task was repeated 100 times, while the pose of the target object was perturbed about its initially estimated position by a random pose vector with a zero-mean Gaussian distribution and standard deviation $\boldsymbol{\sigma}_{xyz\phi\alpha\psi}$. To test the performance limits of the methods presented in Section 6.7, the value of $\boldsymbol{\sigma}_{xyz\phi\alpha\psi}$ was chosen to represent a fairly high level of uncertainty in each of the pose dimensions:

$$\boldsymbol{\sigma}_{xyz\phi\alpha\psi} = \begin{bmatrix} 0.025m & 0.025m & 0.050m & 20^\circ & 20^\circ & 10^\circ. \end{bmatrix}^T / 3$$

The bounds on the maximum uncertainty in the pose are set as $\boldsymbol{\delta}_{xyz\phi\alpha\psi} = \pm 3\boldsymbol{\sigma}_{xyz\phi\alpha\psi}$ to encapsulate 99.7% of all samples. The methods outlined in Section 6.7 are applied to determine the appropriate bounds on the change in joint positions that result from the closed-loop motion (executed to compensate the perturbation in the target object's pose). The results are presented

on 2-D plots for easy visualization. Figure 6.11 shows the results for joints 1 and 2, Figure 6.12 shows the results for joints 3 and 4, and Figure 6.13 shows the results for joints 5 and 6. The open-loop estimates of the desired joint positions are shown in the center of the figures (as dotted red lines). The closed-loop joint positions resulting from the object pose perturbation are shown as sample points (in green). The bounds obtained from the linear Jacobian approximation (shown as dotted green lines) provide good initial estimates, though they fail to encapsulate possible joint positions that result from close-loop motion. These bounds are used to influence the search region in the nonlinear constrained optimization problems. Results from the nonlinear constrained optimization formulation (shown in solid green) show that this method provides bounds that closely match the shape of the sample distribution. More importantly, the bounds encapsulate all samples generated. The bounds obtained from nonlinear constrained optimization are used to define the hyperrectangular region $\boldsymbol{\delta}_{\mathbf{q}} \in [\boldsymbol{\delta}_q^{min}, \boldsymbol{\delta}_q^{max}]$.

6.7.5 Integration with PRM

The hyperrectangular region $\boldsymbol{\delta}_{\mathbf{q}} \in [\boldsymbol{\delta}_q^{min}, \boldsymbol{\delta}_q^{max}]$ obtained from Equations 6.52 and 6.53 describing the bounds on the change in joint positions resulting from closed-loop motion, can be used with the PRM path planning methods (previously discussed in Section 5) to guarantee collision-free visual servoing during the second stage of the *planning then servoing* approach. Since the closed-loop position $\mathbf{q}_f$ is unknown during planning, setting $d(\mathbf{q}_f) = 0$ in Equation 6.43 results in a modified form of the inequality:

$$\sum_{j=1}^{N} r_j |\Delta q_j| < d(\mathbf{q}_i). \tag{6.60}$$

If appropriate collision-free joint bounds can be established by the DCC algorithm (Equation 6.60) to include the hyperrectangular region $\boldsymbol{\delta}_{\mathbf{q}} \in [\boldsymbol{\delta}_q^{min}, \boldsymbol{\delta}_q^{max}]$, then collision-free motion can be achieved in closed-loop by expressing these bounds as the *new* joint-limit constraints for MPC servoing. In the MPC implementation for closed-loop visual servoing, $\mathbf{q}_{i|k} \in \mathbb{Q}_{free}$ (Equation 6.29 from Section 6.4) is replaced with the following collision-free joint bounds obtained from DCC:

$$\mathbf{q}_{i|k} \in [\mathbf{q}_{DCC}^{min}, \mathbf{q}_{DCC}^{max}]. \tag{6.61}$$

The integration of MPC with PRM using the updated collision-free joint boundaries is illustrated in Figure 6.14 in 2-dimensional joint space. If

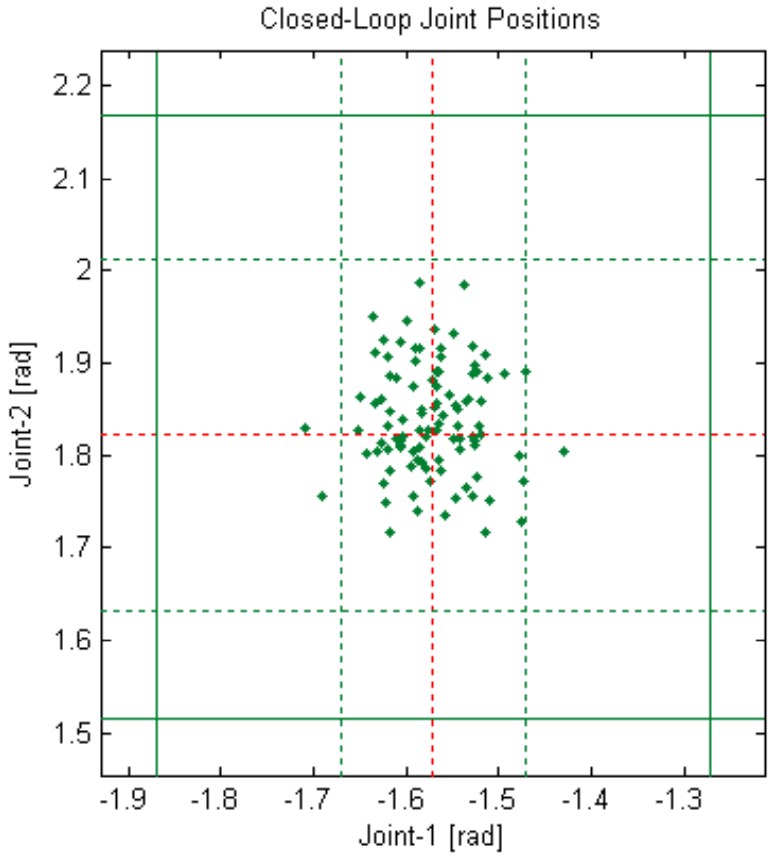


Figure 6.11: Closed-loop positions for joints 1 & 2 resulting from MPC visual servoing with object pose perturbations: closed-loop positions (green points, n = 100), open-loop position (dotted red), linear bounds (dotted green), nonlinear bounds (solid green).

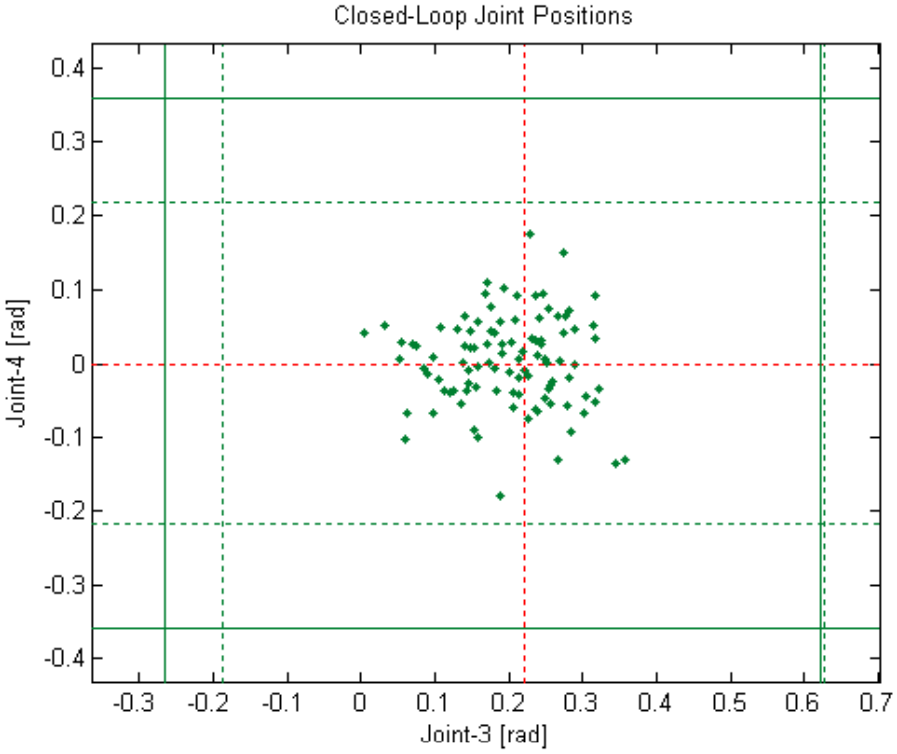


Figure 6.12: Closed-loop positions for joints 3 & 4 resulting from MPC visual servoing with object pose perturbations: closed-loop positions (green points, n = 100), open-loop position (dotted red), linear bounds (dotted green), nonlinear bounds (solid green).

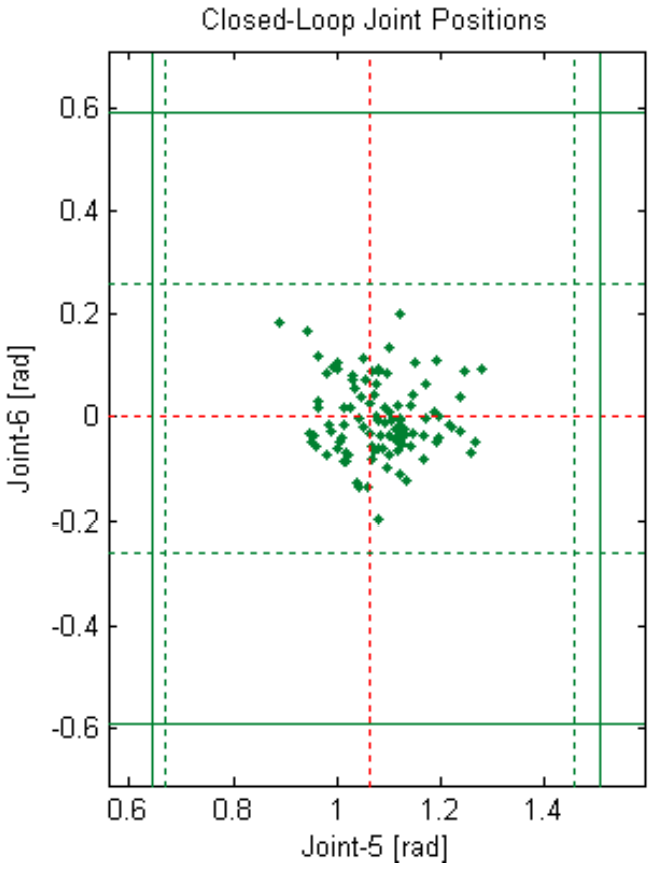


Figure 6.13: Closed-loop positions for joints 5 & 6 resulting from MPC visual servoing with object pose perturbations: closed-loop positions (green points, n = 100), open-loop position (dotted red), linear bounds (dotted green), nonlinear bounds (solid green).

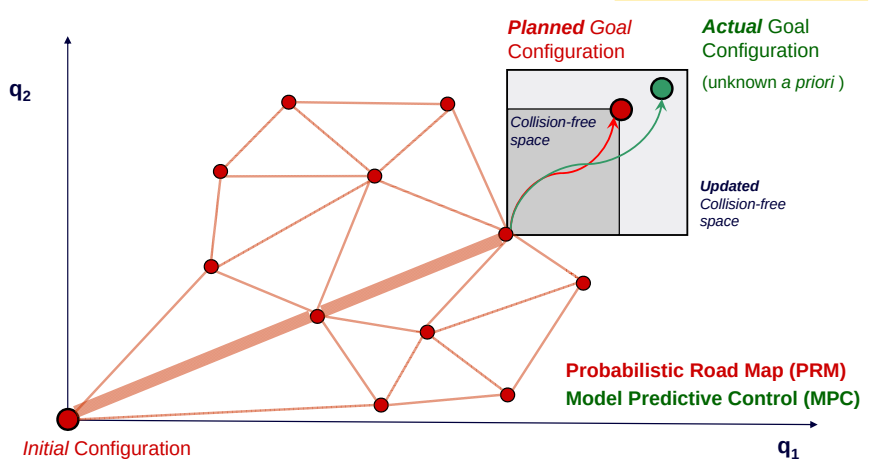


Figure 6.14: Integration of PRM path planning with MPC visual servoing. Visual servoing is restricted to stay within the collision-free space determined by DCC algorithm.

appropriate collision-free joint bounds *cannot* be established by the DCC algorithm (Equation 6.60) to include the hyperrectangular region $\boldsymbol{\delta}_{\mathbf{q}} \in [\boldsymbol{\delta}_q^{min}, \boldsymbol{\delta}_q^{max}]$, then the current pose estimate of the target object is too uncertain for the execution of the two-stage *planning then servoing* approach. Subsequent observations of the target object are required to reduce the uncertainty in the target object's pose in order to guarantee collision-free motion when closed-loop positioning is achieved. Pose uncertainty can be reduced through additional observations with the EKF using images generated from new camera positions (as previously discussed in Chapter 4).

6.8 MPC with Online Collision-Avoidance

6.8.1 Motivation

The two-stage *planning then servoing* approach (using PRM with MPC) allows difficult collision-avoidance problems to be solved *offline* in the absence of computational time constraints. In contrast, for positioning tasks where collision-avoidance is necessary, but not the primary issue at hand, an *online* approach can be used. This *online* approach uses MPC with DCC for *iterative* planning and servoing to ensure collision-free visual servoing over

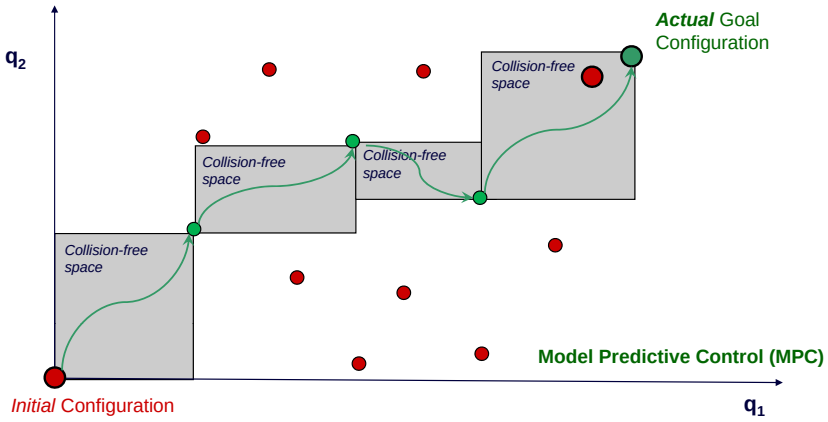


Figure 6.15: Integration of MPC, DCC and adaptive bisection for collision-free visual servoing.

large robot motions.

6.8.2 Integration with Online DCC

The integration of MPC with DCC using a dynamically updated collision-free joint boundary is illustrated in Figure 6.15. During MPC prediction at time instance k, the sequence of planned joint input configurations $\mathbf{q}_{i|k}$ is checked to determine if it satisfies collision constraints using the DCC algorithm (Section 6.6.2). If DCC determines that the planned motion is collision-free, then the first element of the sequence is applied as the input signal to the plant. If DCC determines that the planned motion cannot be guaranteed to be collision-free, then the hyperrectangular region that is determined to be collision-free by the DCC bisection algorithm, is applied as new joint-limit constraints for MPC optimization. Specifically, Equation 6.29 from Section 6.4 is replaced with the following collision-free joint bounds obtained from the hyperrectangular region returned by DCC bisection algorithm:

$$\mathbf{q}_{i|k} \in [\mathbf{q}^{min}_{DCC_k}, \mathbf{q}^{max}_{DCC_k}]. \tag{6.62}$$

The bounds on the collision-free space are dynamically updated at each iteration k, as denoted by the subscript.

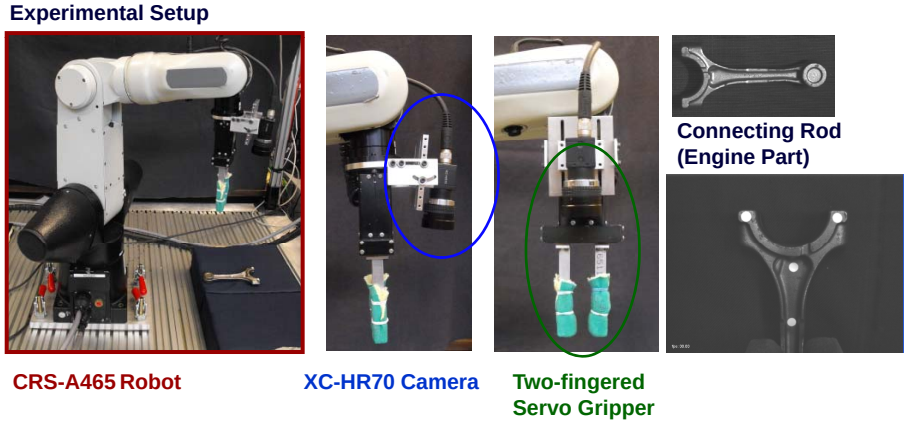


Figure 6.16: Eye-in-hand platform for MPC visual servoing experiments.

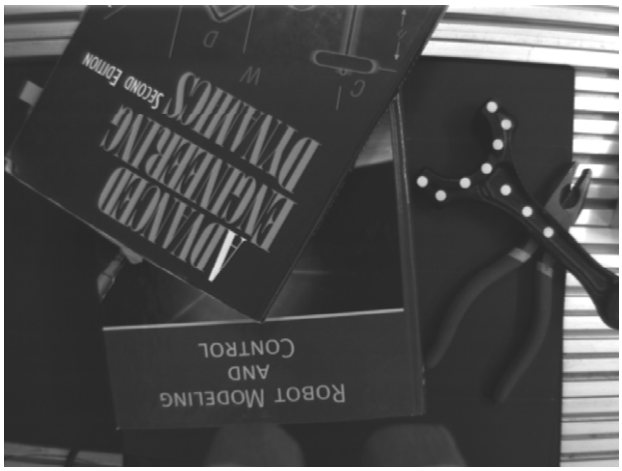
6.9 Experiments: Collision-Free Visual Servoing

6.9.1 Experiment Setup

An overview of the experimental test-bed is shown in Figure 6.16. The goal of the experiment is to position the eye-in-hand robot with respect to the connecting rod (which is located at some random location in the workspace), such that the two-fingered gripper is in the correct position for grasping the connecting rod. MPC visual servoing allows this positioning task to be completed, without requiring prior hand-eye calibration of the system. The following sections describe the robot and camera subsystem, as well as the approximate models that are used in the MPC visual servoing controller.

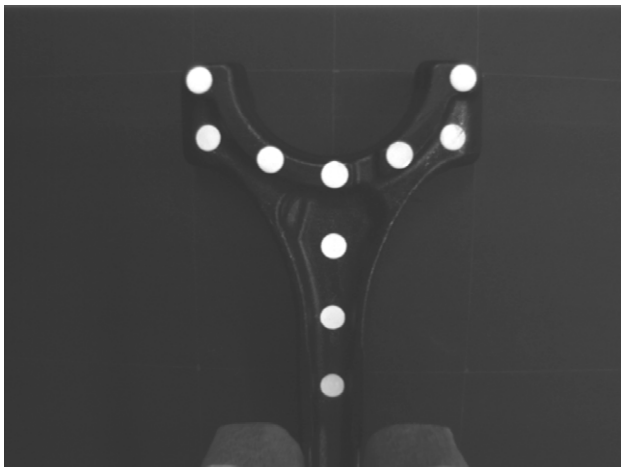
Initial Image and Desired Image

Figure 6.17 shows the initial image that the robot observes, prior to the start of the positioning task. The robot initially observes the workspace from a "bird's eye" view, with its gripper positioned away from the target object. The robot positioning task is communicated to the robot via an image that shows the correct relative positioning between the camera and the target object, when the desired positioning task is achieved. This reference image is shown in Figure 6.18. Note that this image also corresponds to the correct pre-grasp position for the two-fingered gripper, which is used to pick up the conrod. The two-fingered gripper can be seen at the bottom of the desired image, enclosing the neck of the conrod.



Initial Camera View
[Initial Robot Configuration]

Figure 6.17: Overhead image describing the initial position of the camera and end-effector.



"Ready to Grasp" View
[Desired Robot Configuration]

Figure 6.18: Reference image describing the desired position of the end-effector with respect to a target object.

Table 6.1: Uncalibrated Camera Parameters

f	0.0048	[m]
(u^{min}, u^{max})	(1,1023)	
(v^{min}, v^{max})	(1,767)	
(u_0, v_0)	(512,384)	
k_u	$(4.65 \times 10^{-6})^{-1}$	[pixels/m]
k_v	$(4.65 \times 10^{-6})^{-1}$	[pixels/m]
β	90°	

Camera

The imaging sensor that is used in the experiments is a Sony XC-HR70 monochrome CCD camera, with a 4.8mm lens, as shown in Figure 6.16. The camera acquires images at a resolution of 1024×768 at a maximum frame rate of 30fps. An estimate of the camera's intrinsic parameters is required by the MPC controller to model the image projections in the camera for predictive control. The camera parameters that are used in the MPC controller for experiments can be found in Table 6.1. These parameters are obtained from the product manual without calibration.

An estimate of the homogeneous transformation ${}^e\mathbf{T}_c$ from the camera frame to the end-effector frame is also required by the MPC controller, to model the position and orientation of the camera for prediction. This typically requires extrinsic camera calibration. An *approximate* camera-to-end-effector transformation is obtained using measurements by hand and roughly estimating the location of the camera's optical center. The following transformation is used in MPC controller for experiments:

$$ {}^e\mathbf{T}_c = \mathbf{T}_{Rz}(90^\circ)\mathbf{T}_{Rx}(-12.5^\circ)\mathbf{T}_{xyz}(0, -0.085, 0.070)[m] $$

Robot

The robot used in this experiment is a CRS-A465 anthropomorphic robot. It has six independently controlled revolute joints that are highly geared and not backdrivable. A low-level PID controller controls the output torque for each joint to track reference trajectories at 1kHz. The low-level trajectory interpolator accepts inputs in the form of joint positions at 10Hz and accepts velocities when specified as end conditions for smooth trajectory interpolation. A forward kinematic model of the robot is used in the MPC controller to estimate for the location of the camera frame, given a set of

Table 6.2: Joint-position limits of the CRS-A465 6-DoF robot

	q_1 [rad]	q_2 [rad]	q_3 [rad]	q_4 [rad]	q_5 [rad]	q_6 [rad]
$\mathbf{q}^{min}$	-2.9845	0.0000	-3.5605	-3.1416	-1.8151	-3.1416
$\mathbf{q}^{max}$	2.9845	3.1416	0.4189	3.1416	1.8151	3.1416

Table 6.3: Joint-velocity limits of the CRS-A465 6-DoF robot

	q_1 [rad/s]	q_2 [rad/s]	q_3 [rad/s]	q_4 [rad/s]	q_5 [rad/s]	q_6 [rad/s]
$\dot{\mathbf{q}}^{min}$	-3.14	-3.14	-3.14	-2.99	-3.02	-2.99
$\dot{\mathbf{q}}^{max}$	+3.14	+3.14	+3.14	+2.99	+3.02	+2.99

joint configurations. Geometric models of the robot, the camera and the gripper (constructed using geometric primitives) are used for the collision tests. The joint-position limits of the CRS-A465 robot can be found in Table 6.2. The joint-velocity limits of the CRS-A465 robot can be found in Table 6.3. For the purpose of the experiment, the velocities of the CRS-A465 robot have been limited to 5% of their maximum.

Target Object

The target object is a conrod (a connecting rod used in a reciprocating piston engine) consisting of ten feature points as shown in Figures 6.17 and 6.18. A rough point-based model is constructed by measuring the centroid location of the feature points on the object. The *approximate* target object model that is used by the MPC controller can be found in Table 6.4.

An estimate of the pose of the target object is required by the MPC controller for joint-space path planning. In the experiments, a human user determined that the target object has the following "ball-park" pose, expressed with respect to the robot's base:

- Translation: $(10cm, 40cm, 15cm)$ in (x, y, z).
- Rotation: 90° about z.

Therefore, the following *approximate* target object pose is used by the MPC controller for joint-space path planning:

Table 6.4: Feature point coordinates of the target object model

	${}^{o}X_j$ [m]	${}^{o}Y_j$ [m]	${}^{o}Z_j$ [m]
${}^{o}\mathbf{P}_1$	0.076	0.031	0
${}^{o}\mathbf{P}_2$	0.064	0.031	-0.003
${}^{o}\mathbf{P}_3$	0.058	0.015	0
${}^{o}\mathbf{P}_4$	0.054	0	0
${}^{o}\mathbf{P}_5$	0.058	-0.015	0
${}^{o}\mathbf{P}_6$	0.064	-0.031	-0.003
${}^{o}\mathbf{P}_7$	0.076	-0.031	0
${}^{o}\mathbf{P}_8$	0.037	0	-0.008
${}^{o}\mathbf{P}_9$	0.019	0	-0.008
${}^{o}\mathbf{P}_{10}$	0	0	-0.008

$$ {}^{b}\mathbf{T}_{\hat{o}} = \mathbf{T}_{Rz}(90^\circ)\mathbf{T}_{xyz}(0.100, 0.400, 0.150)[m] $$

Significant errors have been introduced to this pose estimate in order to demonstrate the efficacy of MPC visual servoing in correcting for modelling errors. The actual (x, y, z) location of the conrod with respect to the robot based is closer to $(0.130, 0.460, 0.170)[m]$. Also, note that one side of the conrod actually rests on top of another object, as shown in Figure 6.17, resulting in significant out-of-plane rotation in the y axis with respect to the robot's base. However, this is not captured in the pose estimate that is used by the MPC controller, which only has a rotation of 90° about the z axis. Also, the rotation the conrod about the z axis is actually closer to 135°. These significant pose estimation errors (in addition to the approximation errors introduced by the rounded "ball-park" figures, estimated by a human user) are designed to test the ability of the MPC controller to compensate for large errors in its prediction model.

Workspace Obstacles

The obstacles in the workspace include the two textbooks, the black binder and the set of pliers, which are located around the connecting rod. The obstacles are modelled as simple polygons for collision detection. The dimensions of the workspace obstacles can be found in Table 6.5. The ho-

Table 6.5: Dimensions of workspace obstacles

	Length [m]	Width [m]	Height [m]
Textbook 1	0.255	0.175	0.025
Textbook 2	0.235	0.190	0.025
Binder	0.290	0.270	0.040
Pliers	0.165	0.075	0.020

Table 6.6: Location of workspace obstacles

	Homogeneous Transformation [m]
Textbook 1	${}^{b}\mathbf{T}_{wo_1} = \mathbf{T}_{Rz}(60°)\mathbf{T}_{xyz}(-0.030, 0.570, 0.185)$
Textbook 2	${}^{b}\mathbf{T}_{wo_2} = \mathbf{T}_{Rz}(95°)\mathbf{T}_{xyz}(-0.025, 0.450, 0.160)$
Binder	${}^{b}\mathbf{T}_{wo_3} = \mathbf{T}_{xyz}(-0.040, 0.440, 0.120)$
Pliers	${}^{b}\mathbf{T}_{wo_4} = \mathbf{T}_{Rz}(60°)\mathbf{T}_{xyz}(0.110, 0.440, 0.150)$

mogeneous transformations from the workspace obstacle centroidal frame to the robot base frame can be found in Table 6.6.

Control System Implementation

The MPC controller is implemented on Pentium Dual-Core 2.0 GHz CPU running Windows XP. A Matrox Meteor II acquisition board acquires images from the Sony XC-HR70 at a frame rate of 30Hz. A visual tracker using adaptive windowing and thresholding is implemented to track the centroid location of the feature points. The maximum control rate that is achieved, with SQP optimization running on one thread and feature tracking running on another, is 10Hz. The low-level PID, independent joint controller is implemented on a Pentium 4, 2.8 GHz computer operating with a Windows RTX extension. The hardware is controlled through a Quanser Multi-Q PCI card and WinCon software. The inner control loop runs at 1kHz. Quintic polynomial interpolation is used to provide smooth reference signals for PID tracking, to account for the difference in control rates between the inner and the outer control loops. Communication between controllers is handled via the Quanser serial block.

Tuning Parameters

For the eye-in-hand visual servoing experiments, the prediction horizon N_p is set to 10 and the control horizon N_c is set to 1. The use of a longer prediction horizon in the MPC controller is intended to account for the (possibly significant) discrepancy between the real eye-in-hand system and the approximate model used by the MPC controller for planning. The MPC controller time step, δ_t, is set at 0.1 seconds. $\mathbf{Q}$ is an identity matrix and $\mathbf{W}$ is a diagonal matrix with its diagonal entries equal to 10,000.

6.9.2 Experiment Results

Figure 6.19 shows the trajectory of the target object, seen from the eye-in-hand camera, as the robot completes the visual servoing task. The experiment is designed to test the ability of the image-based visual servoing controller to compensate for uncertainties in the model and to efficiently complete the positioning task without violating constraints. The image-based control law successfully handles both the significant rotation that is required about the camera's optical axis and the translational motion that is required towards the target object. The camera approaches the target efficiently without invoking the camera retreat phenomenon observed in other image-based control laws. The MPC controller is able to recognize the out-of-plane rotation required to properly align the gripper with the conrod.

Figure 6.20 shows the trajectory of the robot as the visual servoing task is executed. The initial robot motions in frames 1 to 7 are very aggressive; the DCC algorithm is able to guarantee collision-free motion several prediction steps ahead. Robot motions are more conservative when the robot gripper is in proximity to the workspace obstacles next to the conrod. In frames 8 to 12, collision-free motion can no longer be guaranteed by the DCC algorithm between large changes in robot configurations. As the regular DCC algorithm fails (inequality not satisfied), the MPC controller replaces its joint-limits with the dynamically updated collision-free bounds provided by the DCC bisection algorithm. The MPC controller is optimized over a smaller region in joint-space where collision-free robot motion is feasible. The last few frames show the robot taking very small steps and maintaining a good distance away from the obstacles. The final gripper position is achieved conservatively, despite the incorrect object pose and the incorrect eye-in-hand calibration that is used in the MPC prediction model.

With such a poorly estimated initial target object pose, the same task cannot be executed using a one-step *look-then-move* approach (i.e., using

Figure 6.19: Sequence of camera motions generated by MPC visual servoing to complete a difficult positioning task. The resulting image feature trajectories satisfy the camera field-of-view constraints.

a very large time step δ_t in the MPC controller). Not only does the DCC algorithm fail to guarantee collision-avoidance for such a large robot motion, it actually detects robot-collision at the *open-loop* end goal. The poorly estimated location of the target object puts the robot in virtual collision with the surrounding workspace obstacles, so the robot is unable to execute its motion. This problem can be solved by: (i) improving the pose estimate with the EKF through further observations; (ii) using a smaller time step δ_t, thus allowing the MPC-DCC controller to gradually correct the pose estimate while approaching the target object, as demonstrated above.

A similar problem affects the two-stage planning-then-servoing approach (PRM and MPC, Section 6.7) when the pose of the target object is highly uncertain.

6.9.3 Discussion on Stability

As previously noted, the proof of closed-loop stability for MPC control of a nonlinear system with constraints is still an open research problem. Of particular interest is the proof of stability when the model is *not* assumed to be perfect. Stability is surprising easy to prove under the assumption of a

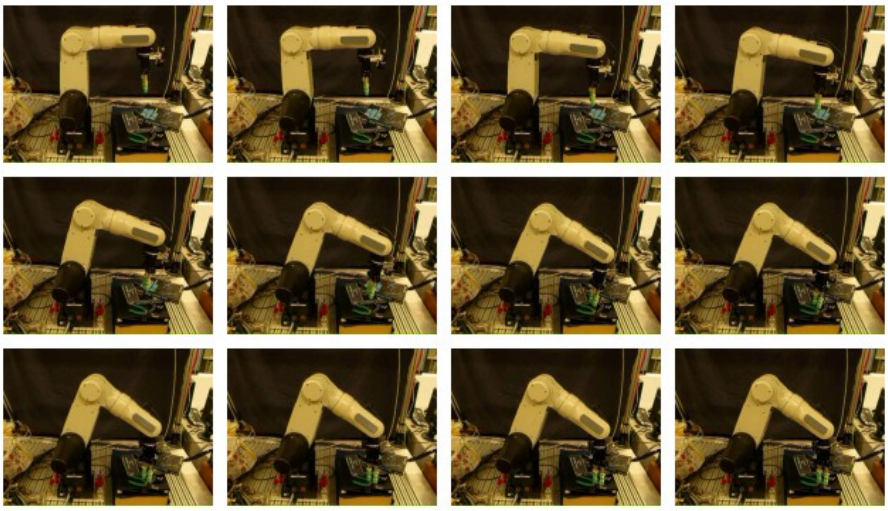


Figure 6.20: Sequence of robot motions generated by MPC visual servoing to complete a difficult positioning task. The resulting robot trajectory satisfies joint position, joint velocity, and workspace collision constraints.

perfect model. Keerthi and Gilbert [57] show that the addition of a "terminal constraint" forces the state to take a particular value at the end of the prediction horizon. The equilibrium point can then be proved stable via a given Lyapunov function, provided that the optimization problem is feasible and is solved at each step. In MPC visual servoing, the fact that a nonlinear eye-in-hand model is used for predictions makes the robust stability analysis of the system quite challenging. In contrast, predictive controllers that are based on *linear models* behave linearly provided that the plant is operating safely away from constraints (however, they behave nonlinearly when constraints are nearby). Unfortunately, for eye-in-hand systems that exhibit severe nonlinearities, the usefulness of predictive control based on a *linearized* model is very limited. For short prediction horizons, the MPC controller can be approximated as a family of switching controllers, with each controller designed to correspond to the set of constraints which are active. In this approximation, the key is the find a common Lyapunov function for all switching controllers to guarantee stability.

In practice, the nominal stability of the closed-loop system is *not* an issue *when an accurate model of the plant is available.* It is quite easy to obtain

stability by tuning the parameters in the problem formulation and very easy to check that the designed system is stable (assuming the correct plant model). This is the typical situation with current applications of predictive control in industrial processes [55].

6.9.4 Insights from Simulations and Experiments

Local Minima from Nonlinear Optimization

The eye-in-hand visual servoing experiments demonstrate that parameter tuning and feedback correction in the MPC controller provide a significant degree of robustness against modelling errors. Stability of the MPC visual servoing controller is demonstrated for a real robotic system completing positioning tasks that require large-range robot motions, in the presence of object pose uncertainty and system constraints. However, each iteration of the MPC eye-in-hand visual servoing requires the optimization of a quadratic cost function, subject to constraints that are non-convex. Specifically, the camera field-of-view constraints are nonlinear functions of the joint configurations of the robot, over which the cost function is minimized. Fortunately, the image-based cost function is designed to naturally keep the image trajectories away from the camera's field-of-view constraints, as in the case with IBVS. Simulations show very few instances where the robot controller runs into local minima due to the nonlinear field-of-view constraints.

Local Minima from Collision Avoidance

The avoidance of collision constraints is also a problem that may cause the visual servoing trajectory to get stuck in local minima. Specifically, the avoidance of collision constraints may be in direct conflict with the visual servoing goal. In the absence of a random sampling of additional collision-free robot configurations (such as in PRM path planning) or random perturbations applied to the current robot configuration (such as in simulated annealing), the control law may not be able to "side-step" the obstacle, since it requires the trajectory to temporary diverge away from the goal. The collision constraints obtained from DCC are dynamically updated at each iteration. Using DCC as part of the MPC formulation, the magnitude of the robot's joint-space motion is naturally scaled to match the robot's proximity to obstacles. Collision constraints obtained from DCC are initially designed to be conservative to keep the robot away from obstacles, in order to prevent the MPC trajectory from getting into a local minimum. On the other hand, integration of MPC with the DCC adaptive bisection

algorithm is designed to allow the robot to take smaller steps in an attempt to get out of a local minimum, before the trajectory becomes enclosed by the collision-constraints in the direction of the goal. In cases where the obstacle avoidance problem is very complex, as in maze solving, the two-stage planning and servoing approach (using PRM for obstacle avoidance and MPC for closed-loop correction) is recommended over the MPC-DCC approach. An analysis of the MPC-DCC approach within the framework of the Active Set method is proposed for future work in Section 7.3.2.

6.10 Summary

Experiment and simulation results in this chapter demonstrate the efficacy of MPC visual servoing in managing the contraints related to an eye-in-hand robot. Using MPC visual servoing formulation, large-range positioning tasks can be completed without violation of field-of-view, joint-limit, joint velocity, or collision constraints, even in the presence of large uncertainties in the initial estimation of the target object's pose. Closed-loop positional accuracy is achieved with an uncalibrated eye-in-hand system, using the *teach-by-showing* method with a rough model of the system for prediction. Eye-in-hand visual servoing experiments show that parameter tuning and feedback correction in the MPC controller provide a significant degree of robustness against modelling errors. Stability of the MPC visual servoing controller is demonstrated for a real robotic system completing a difficult positioning task.

Insights from simulations have been provided for each of the tuning parameters in the MPC visual servoing. The prediction horizon N_p determines the aggressiveness of the control law with respect to constraints. The control horizon N_c determines the degrees of freedom the controller has for choosing how to complete the task. The length of the time step δ_t determines the aggressiveness of the controller with respect to the uncertainty in the target object's location. The matrices $\mathbf{Q}$ and $\mathbf{W}$ weight the priority given to image error minimization and control effort in terms of joint velocities.

A method of modelling an eye-in-hand robot and its associated contraints is presented. A method of modelling collision constraints that is used for MPC visual servoing is also presented. This method exploits a property of the DCC algorithm to obtain a hyperrectangular region in joint-space that is collision-free. The form of these bounds allows them to be included as constraints in the MPC visual servoing controller for trajectory optimization.

Two approaches have been demonstrated for handling collision con-

straints from a path planning perspective. The first approach integrates MPC with a PRM path planner to allow it to handle complicated obstacle avoidance scenarios. The generation of visible vertices within the PRM allows a feasible transition from the PRM path planner to MPC visual servoing. A method of estimating the upper bounds on the changes in joint positions resulting from closed-loop motion is discussed. Simulations of a large number of MPC positioning tasks with perturbations to the target object's pose validate the accuracy of these upper bounds. The bounds are used with the DCC algorithm to ensure collision-free visual servoing after the planned motion is executed. The second approach integrates MPC with DCC to iteratively plan and servo. This method uses DCC to check for collisions and uses the joint bounds obtained from the DCC adaptive bisection algorithm to modify the constraints expressed in the MPC visual servoing controller. Experiments demonstrate the ability of this method to provide efficient visual servoing over large motions, without collisions with workspace obstacles, and without violating joint-limit, joint-velocity and camera field-of-view constraints, even when the target pose's location is uncertain.

To provide a point of reference, typical eye-in-hand systems using the look-then-move approach (discussed in Section 1.2) rely on complete calibration of the robot, the camera, the eye-in-hand transformation, and a completely accurate object pose to achieve its positioning task. The MPC visual servoing controller is able to achieve the same with an uncalibrated system, while compensating for uncertainties in the object's pose and avoiding workspace collisions.

Chapter 7

Conclusions and Recommendations

7.1 Summary Remarks

This thesis represents the design process for a constraint-aware visual servoing control law that can be used to drive large-range robot motions. Unique insights can be gained by examining the improvements that were made at each stage of the research. First, an estimation of pose (through homography decomposition or extended Kalman filter pose estimation) is necessary for improving the predictability of image-based control laws when the visually servoed motion is large. However, the direct use of these Euclidean parameters in the visual servoing control law (as implemented in $2^1/_2$-D visual servoing and position based visual servoing) may not result in the best robot motion. Path planning must occur in the joint-space of the robot, rather than in image-space, to produce feasible trajectories that successfully avoid joint limits, velocity limits, and whole-arm collisions with workspace obstacles. A two-stage planning and servoing approach (using probabilistic roadmaps with MPC visual servoing) allows difficult collision-avoidance problems to be solved, but suffers from the accumulation of errors during the execution of the planning stage. An online iterative planning and servoing approach (using MPC visual servoing with dynamic collision checking) results in trajectories that are far less sensitive to the propagation of modelling errors. Only a rough estimate of the target object's pose is required for planning purposes. In applications where collisions with workspace obstacles are of concern and the target object's pose is uncertain, MPC visual servoing with dynamic collision checking can be implemented to generate collision-free robot motions while closing the visual loop. The final result of this research is a versatile, constraint-aware visual servoing control law that can be used to teach practical robot motions.

7.2 Contributions

The contributions of this thesis are in three main areas:

7.2.1 MPC Eye-in-Hand Visual Servoing

Contribution 1: *A visual servo control law for eye-in-hand robots that effectively manages camera field-of-view, robot joint-limit, robot velocity-limit, and whole-arm collision constraints while visual servoing over large motions.*

1. A method to include whole-arm collision constraints in the MPC optimization problem.
2. A method to correct for uncertainty in the target object's pose using image-based feedback.
3. A method to optimize the predictions over joint positions as inputs, such that future inputs can be checked against collision constraints.

7.2.2 Integration of MPC Visual Servoing with PRM

Contribution 2: *A method to address complicated obstacle avoidance problems while ensuring closed-loop eye-in-hand accuracy, using a two-stage planning-then-servoing approach.*

1. A method to ensure a feasible transition between the PRM path planner and MPC visual servoing controller in the presence of large uncertainties in the pose of the target object.
2. A method to generate new vertices in a PRM that provide *visible* paths connecting the initial robot configuration to the goal robot configuration. The *visible* path consists of robot configurations that keep the target object within the camera's field of view throughout the entire planned path.
3. A method to address uncertainty in the pose of the target object to guarantee collision-free closed-loop motion during the visual servoing stage.

7.2.3 Integration of MPC Visual Servoing with DCC

Contribution 3: *A method to address peripheral obstacle avoidance problems while enabling the use of visual servoing over large motions, using an online iterative planning and servoing approach.*

1. A method to ensure collision-free motion using DCC with MPC visual servoing.
2. A method to obtain dynamically updated collision-free bounds for MPC visual servoing using the results from the adaptive DCC bisection algorithm.

7.2.4 Minor Contributions

Other minor contributions of this thesis include:

- A method of selecting the most *robustly visible* kinematic solution for joint-space path planning for eye-in-hand robots.
- A method of keeping the target within the field of view using adaptive gains for PBVS.
- A method of selecting the correct homography decomposition solution for visual servoing.
- An experimental evaluation of the accuracy of homography estimation and decomposition techniques.
- An PBVS implementation with EKF which also tracks the velocity of the target object.

7.3 Future Work

7.3.1 Interruption of Visual Loop

One advantage of MPC visual servoing is its ability to handle interruptions to its visual feedback loop, without becoming unstable like other visual servoing controllers. In the presence of temporary object occlusions, the MPC controller can be programmed to continue its positioning task by operating in open-loop (using the latest updated prediction model) until the temporary occlusions have passed and the controller regains sight of the target object. The same also applies if the target object temporarily falls outside of the camera's field of view. However, in the presence of a poorly estimated initial target pose and an extended interruption to the visual feedback loop, there is no guarantee that the latest prediction model is correct and that the controller will regain sight of the target object. (It may very well be looking at the wrong place!). In the event that the controller cannot see the

target object for an extended period of time, it must begin an active search for the target object based on its last known location while continuing its positioning task. A method to handle to these two conflicting goals for an eye-in-hand robot remains to be explored.

7.3.2 Stability Analysis of MPC-DCC

A method to formally guarantee the convergence of the eye-in-hand positioning task when facing difficult collision-avoidance problems is desirable. The current MPC-DCC method for handling collision constraints is similar to the Active Set method that is used for solving QP optimization problems. The Active Set method assumes that a feasible solution is initially available, where some inequality constraints are active and some are inactive. (For example, collision constraints are *inactive* when the robot is far away from obstacles). At each iteration, the Active Set method finds an improved solution by taking a step in the direction which minimizes the cost while satisfying the *active* contraints, without worrying about the *inactive* constraints. If this new solution is feasible, then it is accepted as the next iteration. (This is analogous to when the DCC inequality holds and the MPC controller is able to directly apply the input signal to the plant.) If it is not feasible, then a line-search is made in the direction of the step to locate the point at which feasibility is lost - namely the point at which one of the inactive constraints becomes active. (This is analogous to when the DCC inequality fails, and the bisection algorithm is applied to determine the collision-free bounds.) The solution at this point is accepted as the next iteration and the newly active constraint is added to the active set (just as the collision-free bounds are added as new constraints to the MPC controller.) Note that the collision-free bounds are *approximations* to the actual collision-free boundaries in the robot's joint-space. These collision-free bounds are updated dynamically at each iteration. In this aspect, the MPC-DCC formulation is most similar to the SQP algorithm for solving *nonlinear* optimization problems. The SQP algorithm solves a QP sub-problem at each iteration by making a local linear approximation of the nonlinear constraints, with a modification to the original cost function using a quadratic approximation of the Lagrangian. The current MPC-DCC algorithm does *not* make modifications to the cost function when using dynamically updated approximations to the collision-free joint bounds. Further analysis is required to determine how the MPC-DCC formulation can be modified to take advantage of its similarities to the SQP and QP algorithms.

7.3.3 Model Correction Methods

In addition to using an image-based approach for model correction in MPC, the model correction can also take the form of a partial pose correction using homography decomposition or a full pose correction using EKF. The former method requires a very good estimation of depth and relatively noise-free images, while the later requires a calibrated camera. Both increase the computation burdens of the MPC controller. These three differing approaches (image-based model correction, homography-based model correction, pose-based model correction) remain to be validated in experiments to determine their practical merits as well as stability requirements.

7.3.4 Further Evaluation Metrics

To date, there has been little research on how to evaluate the relative strengths and weaknesses of the different visual servoing approaches presented in the field. Specifically, there is a lack of a formal method for comparing the performance of visual servoing controllers in a *quantitative manner* for practical systems that must satisfy constraints. The difficulty in evaluation lies in the large combination of configurations that are possible in the parameter-space and state-space (positioning tasks, robot configurations, target object poses, workspace obstacles, and tuning parameters), and the dependency of visual servoing performance on the specific configuration chosen. In [58], Gans et al. posit a set of preliminary servoing tests and metrics to measure quantitatively the performance of a visual servo controller. However, these metrics are more concerned with the *transient* performance of the controllers, rather than their ability to react in a stable manner to system constraints. The limits imposed by the physical eye-in-hand system (field of view, joint-limits, collision, etc.) in practical applications often render the transient performance of visual servo controllers of secondary importance. For example, for visual servoing systems that are not constraint-aware, very conservative control gains must be chosen to ensure safe operation, which results in poor transient performance. Future work remains to derive common metrics to evaluate the performances of constraint-aware visual servoing approaches against the performance of those that are not constraint-aware.

Bibliography

[1] Danica Kragic and Henrik I. Christensen. Survey on visual servoing for manipulation. Technical report, Computational Vision and Active Perception Laboratory, 2002. `http://www.societyofrobots.com/robottheory/Survey_on_Visual_Servoing_for_Manipulation.pdf`.

[2] Intuitive Surgical Inc., 2009. `http://www.intuitivesurgical.com/`.

[3] Braintech Inc., 2009. `http://www.braintech.com/`.

[4] Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[5] François Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G. Hager, and A.Morse, editors, *The Confluence of Vision and Control*, volume 237 of *Lecture Notes in Control and Information Systems*, pages 66–78. Springer-Verlag, 1998.

[6] Masami Iwatsuki and Norimitsu Okiyama. Rotation-oriented visual servoing based on cylindrical coordinates. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 4198–4203, 2002.

[7] Masami Iwatsuki and Norimitsu Okiyama. A new formulation of visual servoing based on clindrical coordinate system. *IEEE Transactions on Robotics*, 21(2):266–273, April 2005.

[8] Peter I. Corke and Seth A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, August 2001.

[9] François Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4):713–723, August 2004.

[10] E. Cervera, A. P. del Pobil, F. Berry, and P. Martinet. Improving image-basaed visual servoing with three-dimensional features. *The International Journal of Robotics Research*, 22(10-11):821–839, October-November 2003.

[11] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[12] D. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[13] William J. Wilson, Carol C. Williams Hulls, and Graham S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions or Robotics and Automation*, 12(5):684–696, October 1996.

[14] Vincenzo Lippiello, Bruno Siciliano, and Luigi Villani. An experimental setup for visual servoing applications on an industrial robotic cell. In *Proceedings of the 2005 IEEE International Conference on Advanced Intelligent Mechatronics*, pages 1431–1436, Monterey, California, USA, July 2005.

[15] Vincenzo Lippiello, Bruno Siciliano, and Luigi Villani. Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. *IEEE Transactions on Robotics*, 23(1):73–86, February 2007.

[16] Graziano Chesi, Koichi Hashimoto, Domenico Prattichizzo, and Antonio Vicino. A switching control law for keeping features in the field of view in eye-in-hand visual servoing. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 3929–3934, Taipei, Taiwan, September 2003.

[17] Benoit Thuilot, Philippe Martinet, Lionel Cordesses, and Jean Gallice. Position based visual servoing : keeping the object in the field of vision. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1624–1629, Washington, DC, May 2002.

[18] Lingfeng Deng, Farrokh Janabi-Sharifi, and William J. Wilson. Hybrid motion control and planning strategies for visual servoing. *IEEE Transactions on Industrial Electronics*, 52(4):1024–1040, August 2005.

[19] P. Zanne, G. Morel, and F. Plestan. Robust vision based 3d trajectory tracking using sliding mode control. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 2088–2093, San Francisco, CA, April 2000.

[20] R. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 33:117–137, September 1999.

[21] Jacopo Piazzi, Domenico Prattichizzo, and Antonio Vincino. *Visual Servoing Along Epipoles*. Springer-Verlag, 2003.

[22] A. Marotta, J. Piazzi, D. Prattichizzo, and A. Vicino. Epipole-based 3d visual servoing. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 366–371, Lausanne, Switzerland, October 2002.

[23] Ezio Malis and François Chaumette. 2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *International Journal of Computer Vision*, 37(1), June 2000.

[24] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[25] Z. Zhang and A. R. Hanson. Scaled euclidean 3d reconstruction based on externally uncalibrated cameras. In *Proceedings of the 1995 IEEE Symposium on Computer Vision*, pages 37–42, Coral Gables, Florida, USA, November 1995.

[26] Koichiro Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 705–711, October 1998.

[27] Ezio Malis, François Chaumette, and Sylvie Boudet. 2-1/2-d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, April 1999.

[28] Ezio Malis and François Chaumette. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *International Journal of Robotics and Automation*, 18(2):176–186, 2002.

[29] Ville Kyrki, Danica Kragic, and Henrik I. Christensen. New shortest-path approaches to visual servoing. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 349–354, Sendai, Japan, September 2004.

[30] Youcef Mezouar and François Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, August 2002.

[31] Youcef Mezouara and François Chaumette. Avoiding self-occlusions aand preserving visibility by path planning in the image. *Robotics and Autonomous Systems*, 41:77–87, 2002.

[32] Youcef Mezouar and François Chaumette. Optimal camera trajectory with imaged-based control. *International Journal of Robotics Research*, 22(10-11):781–803, October-November 2003.

[33] O. Khatib. Real-time obstacle avoidance for manipulataors and mobile robots. *International Journal of Robotics Reseach*, 5(1):90–98, 1986.

[34] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[35] Florian Schramm, Franck Geffard, Guillaume Morel, and Alain Micaelli. Calibration free image point path planning simultaneously ensuring visibility and controlling camera path. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2074–2079, Roma, Italy, April 2007.

[36] Benedetto Allotta and Duccio Fioravanti. 3d motion planning for image-based visual servoing tasks. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2173–2178, Barcelona, Spain, April 2005.

[37] Guillaume Morel, Philippe Zanne, and Franck Plestan. Robust visual servoing: Bounding the task function tracking errors. *IEEE Transactions on Control Systems Technology*, 13(6):998–1009, November 2005.

[38] P. Zanne, G. Morel, and F. Plestan. Sensor based robot control in the presence of uncertainties: Bounding the task function tracking errors. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 3775–3780, Washington, DC, May 2002.

[39] A. Castano and S. Hutchinson. Visual compliance: task directed visual servo control. *IEEE Transactions on Robotics and Automation*, 10(3):334–342, June 1994.

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

[41] Nicolas Mansard and François Chaumette. A new redundancy formalism for avoidance in visual servoing. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 468– 474, Edmonton, Canada, August 2005.

[42] Bradley J. Nelson and Pradeep K. Khosla. Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. *International Journal of Robotics Research*, 14(3):255–269, June 1995.

[43] Paul Y. Oh and Peter K. Allen. Visual servoing by partitioning degrees of freedom. *IEEE Transactions on Robotics and Automation*, 17(1):1–17, Febuary 2001.

[44] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, February 1993.

[45] Jacques A. Gangloff and Michel F. De Mathelin. Visual servoing of a 6-dof manaipulator for unknown 3-d profile following. *IEEE Transactions on Robotics and Automation*, 18(4):511–520, August 2002.

[46] M. Sauvée, P. Poignet, E. Dombre, and E. Courtial. Image based visual servoing through nonlinear model predictive control. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1776–1781, San Diego, CA, USA, December 2006.

[47] Michaël Sauvée, Philippe Poignet, and Etienne Dombre. Ultrasound image-based visual servoing of a surgical instrument through nonlinear model predictive control. *The International Journala of Robotics Research*, 27(1):25–40, January 2008.

[48] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[49] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Inc., 2006.

[50] S. Chiaverini, B. Siciliano, and O. Egeland. Review of the damped least-squares inverse kinematics withexperiments on an industrial robot manipulator. *IEEE Transactions on Control Systems Technology*, 2(2):123–134, 1994.

[51] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[52] David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundation of probabilistic roadmap planning. *International Journal of Robotics Research*, 25(7):627–643, July 2006.

[53] Fabian Schwarzer, Mitul Saha, and Jean-Claude Latombe. Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. *IEEE Transactions on Robotics*, 21(3):338–353, June 2005.

[54] Simon Leonard, Elizabeth A. Croft, and James J. Little. Dynamic visibility checking for vision-based motion planning. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2283–2288, Los Angeles, California, May 2008.

[55] J. M. Maciejowski. *Predictive Control with Constraints*. Pearson Education Limited, 2002.

[56] M.J.D. Powell. *Variable Metric Methods for Constrained Optimization*. Springer-Verlag, 1983.

[57] S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.

[58] Nicholaas R. Gans, Seth A. Hutchinson, and Peter I. Corke. Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control. *International Journal of Robotics Reseach*, 22(10-11):955–981, 2003.

[59] R. Hartley and A. Zisserman. *A Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

Appendix A

Homography Estimation for Non-Planar Objects

A.1 Virtual Plane Definition

Recall that three 3-D points ${}^c\mathbf{P}_1$, ${}^c\mathbf{P}_2$, ${}^c\mathbf{P}_3$ are selected on the target object to define a common virtual plane π for each pair of images to act as references for visual servoing. In order to ensure numeric stability, these points are chosen (among all possible combination of available feature points) such that they maximize the surface of the corresponding triangles in both images. The resulting three image points $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{m}_3$ in the current camera frame $\mathcal{F}$ are related to the corresponding image points $\mathbf{m}_1^*$, $\mathbf{m}_2^*$, $\mathbf{m}_3^*$ in the desired camera frame $\mathcal{F}^*$ by a projective homography $\mathbf{H}$ such that:

$$\mathbf{m}_i \propto \mathbf{H}\mathbf{m}_i^*, \tag{A.1}$$

where $\mathbf{H}$ is a homogeneous 3×3 matrix. Note that $\mathbf{H}$ is only defined up to a scale factor, so that one of its elements can be set to 1 without loss of generality.

A.2 Planar vs. Non-Planar Objects

If the target object is known to be planar, then *all* pairs of image points belonging to the object are related exactly by the projective homography $\mathbf{H}$. Using an additional feature point on this plane, the 8 unknown elements of $\mathbf{H}$ can be estimated by solving the simple linear homogeneous system of the form:

$$\mathbf{m}_i \times \mathbf{H}\mathbf{m}_i^* = \mathbf{0}, \qquad \forall i = 1, 2, 3, 4. \tag{A.2}$$

Standard methods exist for solving the homography related to planar objects where at least 4 feature points are available [59]. However, this planar assumption severely restricts the type of target objects that can be used for visual servoing.

In the general case where the structure of the object is not planar, the estimation of $\mathbf{H}$ is a nonlinear problem which is difficult to solve online at the rate required for visual servoing. However, if at least 8 points are available (3 to define π, and 5 not belonging to π), then it is possible to estimate the homography matrix $\mathbf{H}$ using a linearized algorithm based on the principle of virtual parallax, exploiting the property that all epipolar lines intersect at the epipole. (A point in one view projects onto an epipolar line in the other view. An epipole is the image in one view of the camera centre of the other view.) The next section briefly describes the application of this linear algorithm, as developed in [23].

For each 3-D target point ${}^c\mathbf{P}_j$ which does not belong to the commonly defined virtual plane π, its projection $\mathbf{m}_j^*$ in the desired camera frame and its projection $\mathbf{m}_j$ in the current camera frame are *not* related by the homography $\mathbf{H}$. If the homography relationship $\mathbf{H}$ is *virtually* applied to $\mathbf{m}_j^*$ (that is, by *virtually* moving ${}^c\mathbf{P}_j$ to the plane π while preserving its projection $\mathbf{m}_j^*$ in the first camera frame), the *virtual* projection $\mathbf{H}\mathbf{m}_j^*$ would not coincide with the *observed* projection $\mathbf{m}_j$ in the second camera frame, due to the parallax effect. But since $\mathbf{m}_j^*$ is preserved in the first camera frame, the *virtual* projection $\mathbf{H}\mathbf{m}_j^*$ must lie on its epipolar line $\mathbf{l}_j$ in the second camera frame. By definition, the same is also true for the *observed* projection $\mathbf{m}_j$. Therefore, the equation of the epipolar line $\mathbf{l}_j$ can be written as the cross-product of the *observed* coordinates $\mathbf{m}_j$ and the *virtual* coordinates $\mathbf{H}\mathbf{m}_j^*$ in projective space:

$$\mathbf{l}_j = \mathbf{m}_j \times \mathbf{H}\mathbf{m}_j^*, \qquad \forall j \notin \pi. \tag{A.3}$$

A.3 Estimation using the Virtual Parallax

The estimation of $\mathbf{H}$ is based on the constraint that all the epipolar lines must meet at the epipole. For each set of three epipolar lines, a constraint can be generated by setting the area of the bounded triangle to be zero:

$$\det\begin{bmatrix}\mathbf{l}_j & \mathbf{l}_k & \mathbf{l}_l\end{bmatrix} = 0, \qquad \forall j \neq k \neq l. \tag{A.4}$$

Using the results from the virtual parallax analysis, this constraint can be expressed in terms of the 8 unknown elements of the homography matrix $\mathbf{H}$:

$$\det\begin{bmatrix}(\mathbf{m}_j \times \mathbf{H}\mathbf{m}_j^*) & (\mathbf{m}_k \times \mathbf{H}\mathbf{m}_k^*) & (\mathbf{m}_l \times \mathbf{H}\mathbf{m}_l^*)\end{bmatrix} = 0, \qquad \forall i,j,k \notin \pi. \tag{A.5}$$

Although the above constraint is non-linear in $\mathbf{H}$, a change of projection coordinates can be performed to reduce the number of unknowns to a minimum, such that the solution can be solved linearly in two steps. Let $\mathbf{M}$ and $\mathbf{M}^*$ be 3×3 transformation matrices formed by the projective coordinates of the three reference points which define the virtual plane:

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \end{bmatrix}, \tag{A.6}$$

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{m}_1^* & \mathbf{m}_2^* & \mathbf{m}_3^* \end{bmatrix}. \tag{A.7}$$

$$\tag{A.8}$$

In the new coordinate systems, the projective coordinates of all remaining feature points $\widetilde{\mathbf{m}}_j$ and $\widetilde{\mathbf{m}}_j^*$ are given by:

$$\widetilde{\mathbf{m}}_j = \mathbf{M}^{-1}\mathbf{m}_j, \tag{A.9}$$

$$\widetilde{\mathbf{m}}_j^* = \mathbf{M}^{*-1}\mathbf{m}_j^*. \tag{A.10}$$

By definition, the coordinates of the three reference points form a canonical basis in the new coordinate systems:

$$(\widetilde{\mathbf{m}}_1, \widetilde{\mathbf{m}}_2, \widetilde{\mathbf{m}}_3) = (\widetilde{\mathbf{m}}_1^*, \widetilde{\mathbf{m}}_2^*, \widetilde{\mathbf{m}}_3^*) = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right). \tag{A.11}$$

The homography relationship relating the three reference points in the two images can be re-expressed in the new coordinate systems as:

$$\widetilde{\mathbf{m}}_i \propto \widetilde{\mathbf{H}}\widetilde{\mathbf{m}}_i^*, \qquad \forall i = 1, 2, 3. \tag{A.12}$$

Since $\widetilde{\mathbf{m}}_1$, $\widetilde{\mathbf{m}}_2$, $\widetilde{\mathbf{m}}_3$ are orthogonal (and similarly $\widetilde{\mathbf{m}}_1^*$, $\widetilde{\mathbf{m}}_2^*$, $\widetilde{\mathbf{m}}_3^*$), the number of unknowns to solve for in $\widetilde{\mathbf{H}}$ are reduced from the original 8 to 3 in the new coordinate systems. That is:

$$\widetilde{\mathbf{m}}_i = \widetilde{h}_i \widetilde{\mathbf{m}}_i^*, \tag{A.13}$$

and

$$\widetilde{\mathbf{H}} = \mathbf{M}^{-1}\mathbf{H}\mathbf{M}^* = \begin{bmatrix} \widetilde{h}_1 & 0 & 0 \\ 0 & \widetilde{h}_2 & 0 \\ 0 & 0 & \widetilde{h}_3 \end{bmatrix}. \tag{A.14}$$

The constraint requiring all the epipolar lines to meet at the epipole can be written in the new coordinate system as:

$$\det \begin{bmatrix} (\widetilde{\mathbf{m}}_j \times \widetilde{\mathbf{H}}\widetilde{\mathbf{m}}_j^*) & (\widetilde{\mathbf{m}}_k \times \widetilde{\mathbf{H}}\widetilde{\mathbf{m}}_k^*) & (\widetilde{\mathbf{m}}_l \times \widetilde{\mathbf{H}}\widetilde{\mathbf{m}}_l^*) \end{bmatrix} = 0. \tag{A.15}$$

Since the above constraint is homogeneous and of polynomial degree three, it can be rearranged to form a measurement matrix $\mathbf{C}_{\widetilde{\mathbf{h}}}$ relating the seven unknown degree-three polynomials:

$$\mathbf{C}_{\widetilde{\mathbf{h}}}\mathbf{x} = \mathbf{0}, \tag{A.16}$$

where

$$\mathbf{x}^T = \begin{bmatrix} \widetilde{h}_1\widetilde{h}_2 & \widetilde{h}_2\widetilde{h}_1 & \widetilde{h}_1\widetilde{h}_3 & \widetilde{h}_2\widetilde{h}_3 & \widetilde{h}_3\widetilde{h}_1 & \widetilde{h}_3\widetilde{h}_2 & \widetilde{h}_1\widetilde{h}_2\widetilde{h}_3 \end{bmatrix}. \tag{A.17}$$

Given n matched feature points that do not belong to the virtual plane, there are $C(n,3)$ ways to choose the three different epipolar lines. This method generates $C(n,3)$ equations for the 7 unknowns, where:

$$C(n,3) = \frac{n!}{3!(n-3)!}. \tag{A.18}$$

Therefore, a minimum of eight matched feature points are required to uniquely solve for $\mathbf{x}$ (with five points not belonging to the plane). Since the constraints are homogeneous, the solution is found by performing a singular value decomposition (SVD) of $\mathbf{C}_{\widetilde{\mathbf{h}}} = \mathbf{U}\mathbf{S}\mathbf{V}^T$. The solution to $\mathbf{x}$ is the column of $\mathbf{V}$ corresponding to the smallest singular value. The unknowns $\widetilde{h}_1$, $\widetilde{h}_2$, $\widetilde{h}_3$ are found by solving the following linear homogeneous system, again using SVD:

$$\begin{bmatrix} -\widetilde{x}_2 & \widetilde{x}_1 & 0 \\ \widetilde{x}_5 & 0 & -\widetilde{x}_3 \\ -\widetilde{x}_7 & \widetilde{x}_3 & 0 \\ \widetilde{x}_7 & 0 & -xt_1 \\ -\widetilde{x}_4 & \widetilde{x}_7 & 0 \\ \widetilde{x}_4 & 0 & -\widetilde{x}_2 \\ \widetilde{x}_6 & 0 & -\widetilde{x}_7 \\ 0 & -\widetilde{x}_6 & \widetilde{x}_4 \end{bmatrix} \begin{bmatrix} \widetilde{h}_1 \\ \widetilde{h}_2 \\ \widetilde{h}_3 \end{bmatrix} = \mathbf{0}. \tag{A.19}$$

Finally, the homography matrix $\mathbf{H}$ is obtained by transforming $\widetilde{\mathbf{H}}$ back into the original coordinate system:

$$\mathbf{H} = \mathbf{M}\widetilde{\mathbf{H}}\mathbf{M}^{*-1} = \mathbf{M} \begin{bmatrix} \widetilde{h}_1 & 0 & 0 \\ 0 & \widetilde{h}_2 & 0 \\ 0 & 0 & \widetilde{h}_3 \end{bmatrix} \mathbf{M}^{*-1}. \tag{A.20}$$

Appendix B

Homography Decomposition

B.1 Geometric Interpretation

A 3-D target point ${}^{c}\mathbf{P}_{j}$ expressed in the current camera frame $\mathcal{F}_c$ as $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}$ projects onto the image point $\mathbf{m} = \begin{bmatrix} x & y & 1 \end{bmatrix}$ in the current image through the following relation:

$$\frac{X}{x} = \frac{Y}{y} = Z. \tag{B.1}$$

When expressed in the desired camera frame $\mathcal{F}_{c^*}$, the *same* 3-D target point ${}^{c^*}\mathbf{P}_{j}$ expressed as $\mathbf{X}^* = \begin{bmatrix} X^* & Y^* & Z^* \end{bmatrix}$ projects onto the image point $\mathbf{m}^* = \begin{bmatrix} x^* & y^* & 1 \end{bmatrix}$ in the desired image through the following relation:

$$\frac{X^*}{x^*} = \frac{Y^*}{y^*} = Z^*. \tag{B.2}$$

The projection $\mathbf{m}^*$ in the desired image is related to the projection $\mathbf{m}$ in the current image by a linear homography $\mathbf{H}$:

$$\mathbf{m} = \mathbf{H}\mathbf{m}^*, \tag{B.3}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix}. \tag{B.4}$$

Let the 3-D target point be located on the plane π, defined by the plane normal $\mathbf{n}^*$ expressed in the desired camera frame $\mathcal{F}_{c^*}$. The signed perpendicular distance d^* of the plane to the desired camera frame $\mathcal{F}_{c^*}$ is given by:

$$d^* = {\mathbf{n}^*}^T \mathbf{X}^*. \tag{B.5}$$

The 3-D coordinates $\mathbf{X}$ and $\mathbf{X}^*$ of the 3-D target point in the two canonical coordinate frames $\mathcal{F}_c$ and $\mathcal{F}_{c^*}$ are related by a rotation ${}^{c}\mathbf{R}_{c^*}$ and a translation ${}^{c}\mathbf{t}_{c^*}$:

$$\mathbf{X} = {}^{c}\mathbf{R}_{c^*}\mathbf{X}^* + {}^{c}\mathbf{t}_{c^*}. \tag{B.6}$$

Substituting the relationship $\frac{\mathbf{n}^*\mathbf{X}^*}{d^*}$=1:

$$\mathbf{X} = \left({}^c\mathbf{R}_{c^*} + \frac{({}^c\mathbf{t}_{c^*})\mathbf{n}^{*T}}{d^*} \right) \mathbf{X}^*. \tag{B.7}$$

The above equation can be rewritten in projective coordinates as:

$$\mathbf{m} = (d^*({}^c\mathbf{R}_{c^*}) + ({}^c\mathbf{t}_{c^*})\mathbf{n}^{*T})\mathbf{m}^*. \tag{B.8}$$

By comparing the above projective relationship to the homography relationship between $\mathbf{m}$ and $\mathbf{m}^*$, the homography matrix $\mathbf{H}$ can be decomposed into an outer product and sum of four Euclidean entities:

$$\mathbf{H} = d^*({}^c\mathbf{R}_{c^*}) + ({}^c\mathbf{t}_{c^*})\mathbf{n}^{*T}. \tag{B.9}$$

where

- ${}^c\mathbf{R}_{c^*}$ is the rotation matrix from frame $\mathcal{F}_{c^*}$ to $\mathcal{F}_c$,
- $\mathbf{n}^*$ is the unit vector normal to π expressed in $\mathcal{F}_{c^*}$,
- ${}^c\mathbf{t}_{c^*}$ is the direction of translation from $\mathcal{F}_{c^*}$ to $\mathcal{F}_c$, and
- d^* is the signed distance from π to $\mathcal{F}_{c^*}$.

B.2 Decomposition Solutions

The inverse problem of matrix decomposition (to obtain ${}^c\mathbf{R}_{c^*}$, $\mathbf{n}^*$, ${}^c\mathbf{t}_{c^*}$ and d^* from $\mathbf{H}$) may contain multiple solutions and is a non-trivial problem. The method was originally proposed by Faugeras [24] and Zhang [25]. To simplify subsequent decompositions, $\mathbf{H}$ is first re-expressed (using SVD) in a diagonalized form:

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T, \tag{B.10}$$

where $\mathbf{\Lambda}$ is a diagonal matrix consisting of the singular values of $\mathbf{H}$, which are positive and sorted in decreasing order $\lambda_1 \geq \lambda_2 \geq \lambda_3$:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}. \tag{B.11}$$

The new decomposition equation is

$$\mathbf{\Lambda} = d'\mathbf{R}' + \mathbf{t}'\mathbf{n}'^T. \tag{B.12}$$

where d', $\mathbf{R}'$, $\mathbf{t}'$, $\mathbf{n}'$ are related to the original d^*, ${}^c\mathbf{R}_{c^*}$, ${}^c\mathbf{t}_{c^*}$, $\mathbf{n}^*$ by:

$$
\begin{aligned}
{}^c\mathbf{R}_{c^*} &= s\mathbf{U}\mathbf{R}'\mathbf{V}^T, && \text{(B.13)}\\
{}^c\mathbf{t}_{c^*} &= \mathbf{U}\mathbf{t}', && \text{(B.14)}\\
\mathbf{n}^* &= \mathbf{V}\mathbf{n}', && \text{(B.15)}\\
d^* &= sd', && \text{(B.16)}\\
s &= \det(\mathbf{U})\det(\mathbf{V}). && \text{(B.17)}
\end{aligned}
$$

Writing $\mathbf{n}'$ in terms of its components $\mathbf{n}' = \begin{bmatrix} n'_1 & n'_2 & n'_3 \end{bmatrix}$ and defining a set of canonical basis:

$$
(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right). \qquad \text{(B.18)}
$$

The decomposition of $\mathbf{\Lambda}$ gives three vector equations:

$$
\lambda_i \mathbf{e}_i = d'\mathbf{R}'\mathbf{e}_i + \mathbf{t}'n'_i, \qquad \forall i = 1, 2, 3. \qquad \text{(B.19)}
$$

Eliminating $\mathbf{t}'$ through multiplication by n'_j followed by subtraction gives:

$$
d'\mathbf{R}'(n'_j\mathbf{e}_i - n'_i\mathbf{e}_j) = \lambda_i\, n'_j\mathbf{e}_i - \lambda_j n'_i\mathbf{e}_j. \qquad \text{(B.20)}
$$

Since $\mathbf{R}'$ preserves the vector norm, taking the norm of the above equation $\forall i, j = 1, 2, 3$ gives a set of linear equations in the unknowns n'^2_1, n'^2_2, n'^2_3:

$$
\begin{aligned}
(d'^2 - \lambda_2^2)n'^2_1 + (d'^2 - \lambda_1^2)n'^2_2 &= 0, && \text{(B.21)}\\
(d'^2 - \lambda_3^2)n'^2_2 + (d'^2 - \lambda_2^2)n'^2_3 &= 0, && \text{(B.22)}\\
(d'^2 - \lambda_1^2)n'^2_3 + (d'^2 - \lambda_3^2)n'^2_1 &= 0. && \text{(B.23)}
\end{aligned}
$$

Using the property that $\mathbf{n}'$ has unit norm, the system of equations can be combined to give 4 different solutions to $\mathbf{n}'$ depending on the chosen signs of n'_1 and n'_3:

$$
\begin{aligned}
n'_1 &= \varepsilon_1 \sqrt{\frac{\lambda_1^2 - \lambda_2^2}{\lambda_1^2 - \lambda_3^2}}, && \text{(B.24)}\\
n'_2 &= 0, \qquad\qquad \varepsilon_1, \varepsilon_3 = \pm 1, && \text{(B.25)}\\
n'_3 &= \varepsilon_3 \sqrt{\frac{\lambda_2^2 - \lambda_3^2}{\lambda_1^2 - \lambda_3^2}}. && \text{(B.26)}
\end{aligned}
$$

To solve for d', note that the determinant of the system of equations must be zero to give a non-trivial (non-zero) solution to $n_1'^2$, $n_2'^2$, $n_3'^2$:

$$(d'^2 - \lambda_1^2)(d'^2 - \lambda_2^2)(d'^2 - \lambda_3^2) = 0. \tag{B.27}$$

However, the solutions $d' = \pm\lambda_1$ or $d' = \pm\lambda_3$ are impossible, since they require $n_1' = n_2' = n_3' = 0$ in the system of equations (due to the relative sizes of $\lambda_1 \geq \lambda_2 \geq \lambda_3$). Therefore, only two solutions to d' remain:

$$d' = \pm\lambda_2. \tag{B.28}$$

If $d' = +\lambda_2$, then the solutions to $\mathbf{R}'$ and $\mathbf{t}'$ are:

$$\mathbf{R}' = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}; \quad \mathbf{t}' = (\lambda_1 - \lambda_3) \begin{bmatrix} n_1' \\ 0 \\ -n_3' \end{bmatrix}, \tag{B.29}$$

where

$$\sin\theta = \frac{\lambda_1 - \lambda_3}{\lambda_2} n_1' n_3' = \varepsilon_1\varepsilon_3 \frac{\sqrt{(\lambda_1^2 - \lambda_2^2)(\lambda_2^2 - \lambda_3^2)}}{(\lambda_1 + \lambda_3)\lambda_2}, \tag{B.30}$$

$$\cos\theta = \frac{\lambda_1 n_3'^2 + \lambda_3 n_1'^2}{\lambda_2} = \frac{\lambda_2^2 + \lambda_1\lambda_3}{(\lambda_1 + \lambda_3)\lambda_3}. \tag{B.31}$$

If $d' = -\lambda_2$, then the solutions to $\mathbf{R}'$ and $\mathbf{t}'$ are:

$$\mathbf{R}' = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & -\cos\phi \end{bmatrix}; \quad \mathbf{t}' = (\lambda_1 + \lambda_3) \begin{bmatrix} n_1' \\ 0 \\ n_3' \end{bmatrix}, \tag{B.32}$$

where

$$\sin\phi = \frac{\lambda_1 + \lambda_3}{\lambda_2} n_1' n_3' = \varepsilon_1\varepsilon_3 \frac{\sqrt{(\lambda_1^2 - \lambda_2^2)(\lambda_2^2 - \lambda_3^2)}}{(\lambda_1 - \lambda_3)\lambda_2} \tag{B.33}$$

$$\cos\phi = \frac{\lambda_3 n_1'^2 - \lambda_1 n_3'^2}{\lambda_2} = \frac{\lambda_1\lambda_3 - \lambda_2^2}{(\lambda_1 - \lambda_3)\lambda_2}. \tag{B.34}$$

The indeterminate signs of n_1, n_3 and d^* give a total of 8 different possible solutions to the decomposition of $\mathbf{H}$. The solution to the original Euclidean components d^*, ${}^c\mathbf{R}_{c^*}$, ${}^c\mathbf{t}_{c^*}$, $\mathbf{n}^*$ can be obtained by transforming d', $\mathbf{R}'$, $\mathbf{t}'$, $\mathbf{n}'$ back into the original coordinate system using $\mathbf{U}$ and $\mathbf{V}$ from the SVD of $\mathbf{H}$.

B.3 Reinforcing Visibility Constraints

Fortunately, the knowledge of one point $m^* = \begin{bmatrix} x^* & y^* & 1 \end{bmatrix}$ on the plane, combined with visibility constraints, can be used to reduce the overall number of solutions from 8 to 2. The visibility constraints of a physical camera require $Z > 0$ and $Z^* > 0$, so that the sign of d^* can be determined from the following relationship:

$$\frac{Z}{Z^*} = \frac{h_{31}x^* + h_{32}y^* + h_{33}}{d^*} > 0. \tag{B.35}$$

Only 4 solutions remain once the sign of d^* is determined. The following constraint for $Z^* > 0$ leaves only 2 of the 4 possible solutions for $\mathbf{n}^*$:

$$\frac{\mathbf{n}^{*T}\mathbf{m}}{d^*} > 0. \tag{B.36}$$

B.4 Solution for Visual Servoing

Only one of the two remaining solutions correctly describes the relative geometry between the desired camera frame $\mathcal{F}_{c^*}$, the current camera frame $\mathcal{F}_c$, and the plane π. If the incorrect decomposition of $\mathbf{H}$ is chosen as input into the control law, there is no guarantee that visual servoing will converge. Therefore, a robust method must be devised to eliminate the incorrect solution.

One method is to use a third image to determine the common plane normal $\mathbf{n}^*$. Unfortunately, this strategy cannot be used in the first iteration of the control law, since the eye-in-hand robot must be commanded to move based on the currently observed image and the original training image. To overcome this deficiency, another virtual reference plane is chosen among the available feature points, so that a common solution to ${}^c\mathbf{R}_{c^*}$ and ${}^c\mathbf{t}_{c^*}$ can be determined. In subsequent iterations, the solution nearest to the previous one is chosen, assuming that the camera motion is small between image frames.

Appendix C

Damped Least-Squares Inverse

In section 4.5.4, the problem of robot singularities arose, reducing the complete controllability of visual servoing methods. A damped least-squares inverse kinematics solution [50] is implemented to address robot singularities. This method provides the possibility of *ensuring complete controllability* throughout the entire robot workspace at the expense of increased tracking errors. Recall that complete controllability is lost when the inverse of the robot Jacobian $\mathbf{J}^{-1}_{\text{robot}}|_{\mathbf{q}}$ does not exist. $\mathbf{J}^{-1}_{\text{robot}}|_{\mathbf{q}}$ can be redefined as a pseudo-inverse with the insertion of a small positive diagonal matrix $\kappa^2\mathbf{I}$ before the matrix inversion to restore its rank. The solution to the modified inverse always exists, but in a slightly inaccurate form:

$$\mathbf{J}^{-1}_{\text{robot}}|_{\mathbf{q}} \equiv \left(\left[\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\right]^T \left[\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\right] + \kappa^2\mathbf{I} \right)^{-1} \left[\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\right]^T . \tag{C.1}$$

The damping factor κ determines the degree of approximation introduced with respect to the pure least-squares solution. Small values of κ give accurate solutions but are not very robust to near-singular configurations, while large values are robust but result in large tracking errors. Using a constant value for κ is, therefore, inadequate for obtaining a satisfactory performance over the entire workspace. κ can be adjusted as a function of a measure of closeness to the singularity. The singular value decomposition of the Jacobian matrix gives insight into the condition of the Jacobian relationship:

$$\left[\mathbf{J}_{\text{robot}}|_{\mathbf{q}}\right] = \sum_{i=1}^{6} \sigma_i \widetilde{\mathbf{u}}_i \widetilde{\mathbf{v}}_i^T, \tag{C.2}$$

where $\widetilde{\mathbf{u}}_i$ and $\widetilde{\mathbf{v}}_i$ are the input and output singular vectors and σ_i are the singular values, ordered by size so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_6$. The damping factor κ is adjusted based on the proximity of the smallest singular value σ_6 to ε (the size of the singular region). A smooth function is chosen so that

continuity of joint velocity is ensured during the transition at the border of the singular region [50]:

$$\kappa^2 = \begin{cases} 0 & \text{when } \sigma_6 \geq \varepsilon, \\ \left(1 - (\frac{\sigma_6}{\varepsilon})^2\right)\kappa_{\max}^2 & \text{otherwise.} \end{cases} \tag{C.3}$$

Appendix D

Extended Kalman Filter

A nonlinear system can be approximated by linearizing it about the current state from which infinitesimally small changes occur. The *linearized* Kalman filter is based on finding a linear system whose states represent the deviations from a nominal trajectory of a nonlinear system. After linearization, a Kalman filter can be used to estimate the deviations from the nominal trajectory and obtain an estimate of the states of the nonlinear system. An improved version of the linearized Kalman filter is the *extended* Kalman filter (EKF), which directly estimates the states of a nonlinear system using a bootstrap method. That is, the nonlinear system is linearized around the Kalman filter estimate, and the Kalman filter estimate of the nominal trajectory is, in turn, based on the linearized system. The complete set of equations used for the implementation of EKF pose estimation is outlined below:

Prediction of State Estimate:

$$(\widehat{\mathbf{x}}_{\text{obsv}})_{k,-} = \mathbf{F}((\widehat{\mathbf{x}}_{\text{obsv}})_{k-1}, (\mathbf{u}_{\text{obsv}})_{k-1}) \tag{D.1}$$

Linearization of State Difference Equations:

$$\widetilde{\mathbf{A}}_k = \left.\frac{\partial \mathbf{F}(\mathbf{x}_{\text{obsv}}, \mathbf{u}_{\text{obsv}})}{\partial \mathbf{x}_{\text{obsv}}}\right|_{\mathbf{x}_{\text{obsv}}=(\widehat{\mathbf{x}}_{\text{obsv}})_{k,-}} \tag{D.2}$$

Prediction of Covariance Estimate:

$$\widetilde{\mathbf{P}}_{k,-} = \widetilde{\mathbf{A}}_{k-1}\widetilde{\mathbf{P}}_{k-1}\widetilde{\mathbf{A}}_{k-1}^T + \widetilde{\mathbf{Q}}_{k-1} \tag{D.3}$$

Linearization of Output Equations:

$$\widetilde{\mathbf{C}}_k = \left.\frac{\partial \mathbf{G}(\mathbf{x}_{\text{obsv}})}{\partial \mathbf{x}_{\text{obsv}}}\right|_{\mathbf{x}_{\text{obsv}}=(\widehat{\mathbf{x}}_{\text{obsv}})_{k,-}} \tag{D.4}$$

Correction of Covariance Estimate:

$$\widetilde{\mathbf{P}}_k = \left[\widetilde{\mathbf{P}}_{k-1}^{-1} + \widetilde{\mathbf{C}}_k^T\widetilde{\mathbf{R}}_k^{-1}\widetilde{\mathbf{C}}_k\right]^{-1} \tag{D.5}$$

Kalman Gain:

$$\widetilde{\mathbf{K}}_k = \widetilde{\mathbf{P}}_k \widetilde{\mathbf{C}}_k \widetilde{\mathbf{R}}_k^{-1} \tag{D.6}$$

Correction of State Estimate:

$$(\widehat{\mathbf{x}}_{\text{obsv}})_k = (\widehat{\mathbf{x}}_{\text{obsv}})_{k,-} + \widetilde{\mathbf{K}}_k((\mathbf{z}_{\text{obsv}})_k - \mathbf{G}((\widehat{\mathbf{x}}_{\text{obsv}})_{k,-})) \tag{D.7}$$