Control of Mobile Manipulation with Networked Sensing

by

Haoxiang Lang

M. A. Sc., The University of British Columbia, 2008
B. A. Sc., Ningbo University, 2003

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

DOCTOR OF PHILOSOPHY

in
THE FACULTY OF GRADUATE STUDIES

(MECHANICAL ENGINEERING)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)
September, 2012

Abstract

This thesis addresses the manipulation control of a mobile robot with the support of a sensor network, for carrying out dynamically challenging tasks. Such tasks are defined as those where the robot is required to first identify objects, approach and grasp the needed objects, and transport them to goal locations in an environment that is dynamic, unstructured and only partially known. In the present work, a robotic system with these capabilities is developed and implemented for use in tasks of search and rescue, and homecare robotics. To this end, this thesis makes significant original contributions in developing a scheme of adaptive nonlinear model predictive control (ANMPC) and a sensor network with dynamic clustering capability for mobile manipulation under challenging conditions.

Two object tracking algorithms for color tracking and feature tracking are developed for object identification and tracking. A system that uses Q-learning is developed for mobile robot navigation, which allows the robot to learn and operate in an unknown and unstructured dynamic environment.

A traditional approach of image-based visual servo control is developed and demonstrated. The scheme of ANMPC is developed, which incorporates a multi-input multi-output (MIMO) control system that can accommodate constraints, including environmental constraints and physical constraints of the robots. In implementing ANPC scheme, the nonlinear and time-variant model is linearized on line with respect to the current position of the image feature and robot joints, using an adaptive approach. The corresponding control architecture predicts the system outputs and generates optimized control actions according to a cost function.

In order to extend the mobile manipulation system to a wider workspace such as that found in cities and home scenarios, a sensor network is designed and developed employing PFSA (Probabilistic Finite State Automata). The developed PFSA is utilized in both modeling the sensor data and organizing and representing the sensor network. An application of object identification and tracking is presented; and a heterogeneous sensor network is developed along with a simulation platform in MATLAB. A self-organized

and clustered sensor network, which is based on PFSA, is demonstrated. In conclusion, directions for further research and development are indicated.

Table of Contents

List of Figures

List of Tables

Nomenclature

B	Blue
c	Column
$C(k)$	Cost function
D	Distance measurement of the laser
$D(x, y, \sigma)$	Difference of Gaussian
f	Focal length
$g(x, y)$	Magnitude of the gradient
G	Green
$G(x, y, \sigma)$	Gaussian kernel
G^k	Pattern of interest
H	Hue
$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}$	Homogeneous transformation
H_p	Prediction horizon
H_u	Control horizon
$\bar{h}$	Average hue value
I	Intensity
$I(x, y)$	Original Image
J	Jacobian
L	Interaction matrix
$L(x, y, \sigma)$	Convoluted image
$NCC(A, B)$	Normalized cross correlation
p	Position and orientation of the robot
p'	New position and orientation of the robot
$P(a_k)$	Probabilistic action selection function
q^0	Stat state
Q	Finite set
$Q(s_i, a_j)$	An entry in the Q-table

$Q \times \Sigma_i \to Q$	State transition function
$Q \times \Sigma_o \to [0,1]$	Output mapping
r	Row (see Chapter 2)
r	Reward (see Chapter 3)
R	Red
$R: S \times A \to \Re$	Reward function
s_x	Horizontal dimension of the CCD sensor
s_y	Vertical dimension of the CCD sensor
S	Saturation
$T: S \times A \to \Pi(S)$	Transition function
α	Learning rate
β	Discount factor
σ	Standard deviation
ϵ_i	Detection region
$\theta(x, y)$	Orientation of the gradient
Σ_i	Input alphabet
Σ_o	Output alphabet
(u_L, v_L)	Coordinates of the left image
(u_R, v_R)	Coordinates of the right image
Δx	Position difference on the *x*-axis
Δy	Position difference on the *y*-axis
$\Delta\theta$	Orientation difference

List of Abbreviations

ANMPC	Adaptive Nonlinear Model Predictive Controller
ANNs	Artificial Neural Networks
CCD	Charge-coupled Device
CMOS	Complementary Metal–oxide–semiconductor
CNC	Computer Numerical Control
DFSM	Deterministic Finite State Machine
DH	Denavit-Hartenberg
DOF	Degree of Freedom
DoG	Difference of Gaussian
DSTC	Dynamic Space-time Clustering
FSA	Finite State Automata
FSM	Finite State Machine
GPC	Generalized Predictive Control
GUI	Graphic User Interface
HSI	Hue Saturation Intensity
IBVS	Image-based Visual Servoing
LoG	Laplacian of Gaussian
MDP	Markov Decision Processes
MPC	Model Predictive Control
MV	Machine Vision
NDFSA	Non-deterministic Finite State Automata
NCC	A Normalized Cross-correlation
NF	Neuro-fuzzy
OES	Optical Emission Spectroscopy
PA	Probability Automata
PBVS	Position-based Visual Servoing
PFSA	Probabilistic Finite State Automata
PID	Proportional, Integral, Derivative

RGA	Residual Gas Analysis
RGB	Red Green Blue
RNN	Recurrent Neural Network
SIFT	Scale Invariant Feature Transform
SQKF	Sequential Q-learning Algorithm with Kalman Filtering
TSNNs	Time Series Neural Networks
UGVs	Unmanned Ground Vehicles
WTC	World Trade Center

Acknowledgements

First, I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Clarence W. de Silva whose constant supervision and guidance have enabled me to effectively and successfully complete the Master's and Ph.D. studies at UBC. Particular acknowledgement should be made of the research grants held by Dr. de Silva from the Natural Sciences and Engineering Research Council (NSERC) of Canada, Canada Research Chairs Program, Canada Foundation for Innovation (CFI), British Columbia Knowledge Development Fund (BCKDF) which funded the research and provided generous research assistantships for me.

In the same vein, I also want to thank my colleagues, Dr. Tahir Khan, Dr. Ying Wang, Mr. Edward Wang and all other colleagues at the Industrial Automation Laboratory (IAL) for their friendship and help. For those who have been working with me, and have paved the way to my success, please allow me to say a sincere "thank you." I also want to thank my friends who have supported me and given me their coaching and mentoring in the past two years.

Finally, I would like to express my deepest thankfulness to my father Junsheng Lang and my mother Xiutian Zheng who have given me life, and my wife Wanling Li who always supports me.

CHAPTER 1 Introduction

In recent years, the emphasis of robotic research appears to have shifted from the development of robots for structured industrial environments to the development of autonomous and cooperative mobile robots operating in unstructured and partially-known natural environments, such as homes, planet surfaces and disaster scenes. These autonomous mobile robots are applicable in a number of challenging practical tasks such as cleaning of hazardous material, surveillance, rescue, and reconnaissance in unstructured environments which can be too hazardous for humans; and taking care of the elderly and the disabled in home environments where human caregivers are costly and in short supply. It is expected that this new class of mobile robots will have extensive applications in activities where human capabilities are needed, yet not suitable or impractical for human presence (Siegwart and Nourbakhsh, 2004).

As the main objective, this research aims to complement the existing research activities in this area, leading to the development of an autonomous mobile robotic grasping system and associated methodologies and technologies that particularly incorporate visual servoing with advanced control and networked sensing. The application domains for these developments are search and rescue situations, and homecare, where mobile robots are employed.

1.1 Motivation

1.1.1 Search and Rescue Robotics

The necessity of autonomous search and rescue robots was highlighted in the 2001 attack and destruction of the World Trade Center (WTC) in New York City which resulted in the death of 343 firefighters and 65 other rescuers. This additional loss of life was mainly caused by the search and rescue operations in an unsafe environment. Generally speaking, rescue workers have about 48 hours to retrieve trapped humans

subsequent to a disaster. However, much time is wasted because of the lack of necessary equipment and resources for accessing collapsed buildings or generally unsafe areas. Robotic rescuers will be able to carry out rescue operations more efficiently without further endangering human life.

Ten years after the WTC incident, the tsunami caused by the earthquake in Sendai, Japan, resulted in a nuclear reactor crisis. It was another situation where robotic search and rescue would have been effective. Unfortunately, there was no report of the use of robots in that situation even though Japan is a leader in robotic applications. These facts highlight that there are still serious challenges and unsolved technical difficulties in the field of autonomous mobile robotic manipulation.

1.1.2 Homecare Robotics

Another good application of mobile manipulation systems is in homecare environments. Since the overall average age of the world population is growing, the percentage of non-working elders is increasing with respect to the working population. In Canada, statistics show that the percentage of the senior population will reach about 25% of the overall population in 2050, while that percentage was a mere 8% in 1971. This dramatic change will cause financial problems (i.e., more public funds will be needed for healthcare and homecare) as well as social problems (i.e., short of human caregivers, effect on families). An effective solution to this problem is to employ autonomous mobile robots with adequate task capabilities and intelligence. These robots can be designed for taking care of elders (and the disabled) in their own home environments, providing assistance in daily activities, medical assistance, surveillance, and so on. Not having to remove the care receiver from the familiar home environment is a clear benefit in this regard. Furthermore, round-the-clock care can be provided by a robotic system, in a consistent manner. Also, the family members will have the freedom and peace of mind to pursue their own activities such as employment and education while being able to monitor the home scenario on line with the use of the system sensors and communication links.

1.2 Scope and Goals of the Research

As investigated in the present work, a mobile robotic manipulation system is rather complex and involves multi-domain technologies such as computer vision, artificial intelligence, mechanical and electrical design, signal processing, sensing and control. The objective of this thesis is to develop methodologies and technologies that will lead to an autonomous mobile robot manipulation system for important practical applications. The two real-word application scenarios which are targeted in the present developments and implementations are robotic search and rescue, and homecare robotics. In view of the needed capabilities for such a robotic system, the scope of the thesis spans four main areas: 1. Machine vision, specifically object identification and tracking, 2. Robotic navigation, 3. Manipulation, and 4. Networked sensing. The thesis will particularly address the following primary challenges:

- Develop methodologies and vision systems for detecting, identifying, and tracking objects of interest in both global and local sensing areas of a robot.
- Develop a robotic navigation system that has the ability to autonomously guide a mobile robot to a goal location in an unknown, unstructured and dynamic environment.
- Develop effective control strategies for robust control of motion and manipulation of mobile robots in the focused application scenarios.
- Incorporate into the control system a sensor network with both static and dynamic sensors along with an information fusion technology to enhance the performance of the robotic tasks.
- Study such performance issues as robustness, cooperative behavior, self-learning, and adapting capability of the developed robotic system.

1.3 Problem Definition

The primary objective of the present work is to develop an autonomous mobile manipulation system which has abilities to identify objects, navigate in the workspace, grasp an object of interest and finally transport it to a goal location. A schematic representation of the system developed in the present work is shown in Figure 1.1.

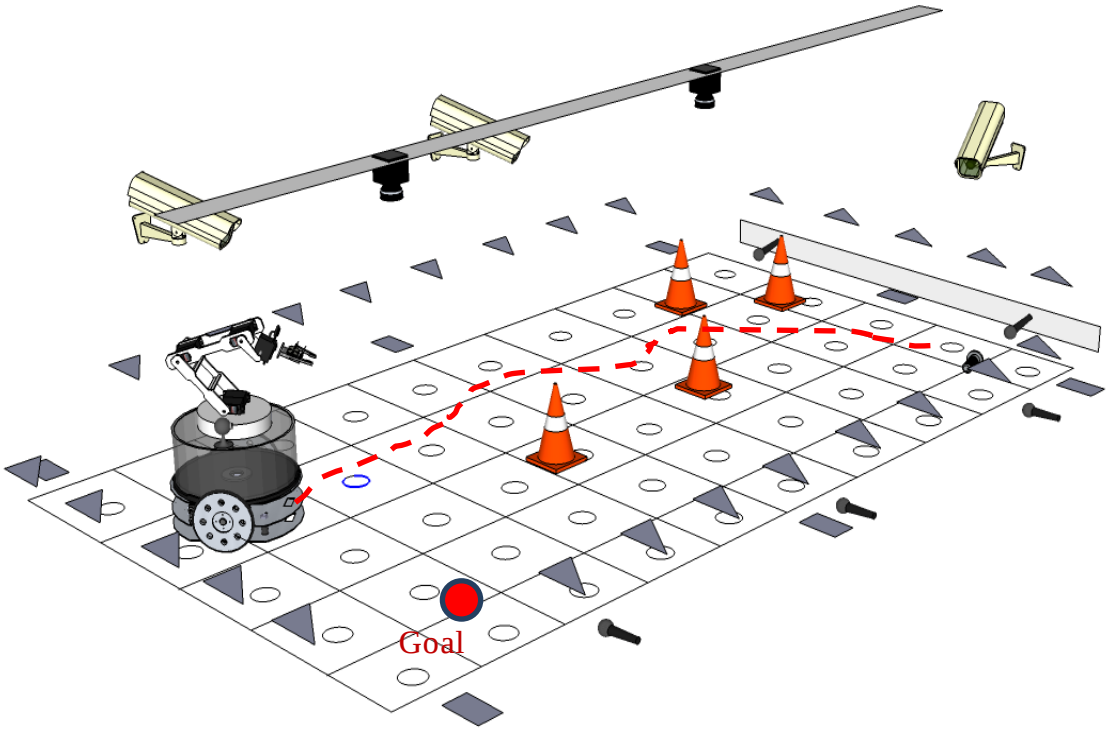


Figure 1.1: Schematic representation of the developed system.

In this figure, there is a sensor network which contains a set of different type sensors, such as pressure sensors (circles in the middle of each square block), acoustic sensors, magnetic sensors (triangles), cameras, and laser distance finders. Some sophisticated sensors are “dynamic” (e.g., the cameras have the ability to move along the track under the ceiling; some cameras, magnetic sensors and laser distance finders are carried by mobile robots). The working procedure of the system is shown in Figure 1.2. The robot receives a task (e.g., find a specific object, go and grasp it, and transport it to a goal location) while it is navigating in the workspace. Then, it will try to find the object of interest in the environment with the help of the sensor network. As it finds the object, the sensor network will provide to the robot the coordinates of the object in the workspace, and also the location and the orientation of the robot. The robot then establishes a way to approach the object while giving due consideration to the dynamics of the environment including obstacles and other robots. As the robot moves close enough to the object with the workspace of the robotic arm encompassing the object, it will grasp the object. Finally the robot transports the object to its goal location. In this procedure, a sensor

network continuously assists the robot in object identification, mobile localization and navigation.

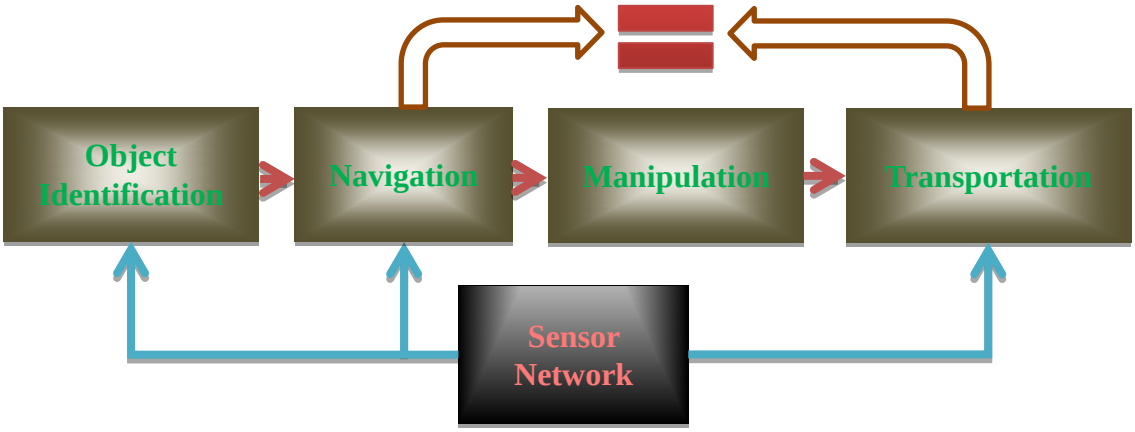


Figure 1.2: The flowchart of the mobile manipulation system.

There are three challenges in the development of the system, which are all key issues in realizing a working mobile manipulation system for application the targeted application scenarios. The first one is the object identification and tracking. In particular, the manipulation operation heavily relies on the results of identification and tracking because the approach of visual servoing utilizes camera information in the feedback loop of the robot control system. The failure of correct identification and tracking during manipulation will result in failure of the entire system.

The second challenge is localization and navigation, which concerns accurately determining the location and orientation of the robot at each instance of the robotic task in order to decide the appropriate next movement for achieving the goal. Because on-board sensors of the robot can only detect and scan a limited area of the work environment, it is impossible to monitor the entire robotic environment using these sensors. Moreover, without completely knowing the environment in advance, the robot cannot deploy the traditional path planning technologies for decision making related to navigation.

The third challenge concerns the manipulation process. In a mobile robot, the base frame is not fixed; and the base coordinate frame will change with time. A pre-defined approach to manipulation, as typically used in industry will not work in this case. A more robust approach needs to be developed that can function in a complex working environment.

In order to extend the capabilities of a mobile manipulation system to fit the entire global environment, rather complete information regarding the workspace is required, which cannot be provided by the robot's on-board sensing capability. How to select and cluster the available global sensors to generate the necessary information for robotic control, and to improve the accuracy of the information that is provided to the robot are important in this regard.

In the present thesis, several approaches are developed to address these issues and overcome the pertaining key challenges. This will enable the mobile manipulation system to work robustly and effectively in a dynamic and partially known environment.

1.4 Related Work

1.4.1 Machine Vision

Machine vision is a powerful sensory tool as it can mimic the human sense of vision and allow non-contact measurement of a working environment. Accordingly, much attention of the research community has gone to applying vision as a feedback sensor in industrial control applications. Among the projects of visual servoing, quite well known is the "DARPA Urban Challenge." This involves competing teams to build autonomous, driverless vehicles that are capable of driving in traffic while performing complex maneuvers such as merging, passing, parking and negotiating intersections in an urban environment. In these vehicles, camera is the main sensor for providing feedback from the vehicle environment to the vehicle control system.

The application of camera vision in computer-based machines is traditionally called machine vision (MV), which involves image processing and image interpretation (computer vision). It is a subfield of engineering that encompasses computer science, optics, mechanical engineering, and industrial automation. A common application of machine vision is the inspection of manufactured goods such as semiconductor chips, automobiles, food products, and pharmaceuticals. A machine vision system can carry out quality assessment tasks with good accuracy and repeatability. It requires digital input/output devices and computer networks, combined with image processing techniques, to control the manufacturing equipment. Other recent applications in this area involve vision-based object detection, tracking of UGVs (Unmanned Ground Vehicles),

following of mobile robots, and vision-based feedback control of robotic manipulator movements (visual servoing). This technology can also be implemented in applications of security and transportation such as video surveillance and traffic control.

Vision-based automated object detection has been playing a significant role in industrial and service applications. Associated studies have focused on detecting objects efficiently by using features such as color, shape, size, and texture. However, there are a number of problems that arise when using these methods to process real world images under different conditions and environments. Most recent machine vision algorithms may not necessarily possess adequate performance for common practical use.

Seelen et al. (2000) have used Symmetry Analysis and Model Matching to detect the rear, front and side views of a group of object types by measuring the inherent vertical symmetric structure. In their paper, the authors mention that the method has to be robust against changes in illumination and slight differences of the right and the left parts of an object. The symmetry-based method is challenged in this manner under real operating conditions.

As well known, color is a very useful feature in object detection. However, few existing applications of detection and tracking have used color for object recognition, because color-based recognition is complicated, and the existing color machine vision techniques have not been shown to be effective. Buluswar and Draper (1997) have presented a technique for achieving effective real-time color recognition in outdoor scenes. It is claimed that this method has been successfully tested in several domains, such as autonomous highway navigation, off-road navigation and target detection for unmanned military vehicles.

Bertozzi et al. (1997) have proposed a corner-based method to hypothesize vehicle locations. The system presented in their paper was composed of a pipeline of two different engines: a massively parallel architecture for efficient execution of low-level image processing tasks, improved by the integration of a specific feature for direct data I/O; and a traditional serial architecture running medium-level tasks aimed at the detection of the vehicle position in the sequence. A preliminary version of the system was reported, and it was demonstrated on the MOB-LAB land vehicle.

The use of constellations of vertical and horizontal edges has shown to be a strong cue for hypothesizing objects in some situations. In identifying pronounced vertical

structures in an image, Matthews et al. (1996) used edge detection to find strong vertical edges. To localize the left and right positions of a vehicle, they computed the vertical profile of the edge image followed by smoothing using a triangular filter. By finding the local maximum peaks of the vertical profile, they claimed that they could find the left and the right positions of a vehicle.

Template-based methods use a predefined pattern of the object class and perform correlation between the image and the template. Handmann et al. (2000) proposed a template based on the observation that the rear/frontal view of a vehicle has a “U” shape. During verification, they considered a vehicle to be present in the image if they could find the “U” shape. Ito et al. (1995) used a very loose template to recognize pronounced vertical/horizontal edges and existing symmetry. Due to the simplicity of the template, these two papers did not generate very accurate results.

Appearance-based methods learn the characteristics of object appearance from a set of training images which capture the variability in the object class. Compared to the previously discussed approaches, it is the most accurate and reliable one. In particular, Lowe (1999, 2004) proposed an algorithm for object recognition and tracking, called the Scale Invariant Feature Transform (SIFT), which uses a class of local image features. In his algorithm, the detected features are invariant to changes in illumination, noise, rotation and scaling; and it has been proven that this approach has high robustness and reliability. In the present thesis, the SIFT algorithm is utilized to enable a mobile robot track an object in the camera view, and feedback the environmental information, to control the robot to a goal location. To the best of our knowledge, this thesis is the first work to apply the SIFT algorithm for visual servoing in robust mobile robot tracking.

1.4.2 Robot Navigation

An important issue that is investigated within the community of mobile robot research is robot navigation. This is justified since this involves the capability of a robot to understand its workspace, determine efficient motion strategies to achieve the motion goal with consideration of its current pose, goal location in the workspace and possible obstacles in the path. There are many relevant research directions in this field.

A popular approach of mobile navigation involves path planning, which starts by acquiring a map of the workspace, and then uses an effective path planner to find a

suitable and available path to reach the goal. The field of path planning has been extensively studied with respect industrial manipulators (with fixed base), and is gradually being extended to mobile robots as well. As indicated in the review of Siegware and his colleagues (2004), the path-planning system usually converts the continuous environmental map into a discrete map, and then the planning is achieved by utilizing some general strategies to decompose the global problem into local or smaller problems, such as the roadmap method and the potential field method. Teng et al. (1993) proposed a navigation method for an initially-known, natural terrain which is assumed to be free of obstacles. It divides the environment into several sub-regions. In each time interval, the robot performs computations only in the current region according to the rule of navigating to the next sub-region which is nearest to the final goal. Bortoff (2000) and Gu et al. (2004, 2006) utilized a Voronoi diagram method to construct a graph based on the radar sites for flying navigation of an Unmanned Aerial Vehicle (UAV). The flying trajectory is generated by searching for the shortest path to the specified destination. This approach can also be used for on-land navigation of mobile robots in a free-space environment. However, this approach has a constraint in that it is hard to find a collision-free path in an unstructured environment. Birgesson et al. (2003) utilized the potential field algorithm in the path planning for robot navigation. Three forces are utilized: the repulsive force of the obstacles and the attractive forces of the goal and the waypoints. Each force is activated only if it meets certain criteria, and the goal force is disabled when a suitable waypoint is found. Therefore, the robot is pulled toward the nearest waypoint that lies within a predetermined range of angles between the robot and the goal. If no waypoints meet this criterion, the robot is pulled toward the goal. The repulsive force is activated only if the robot is within a threshold distance from the obstacles.

There is a drawback in this type of approach which can make navigation impossible when the robot is exploring an unknown territory: there has to be a global sensor that can monitor the entire workspace and generate a global map, which can then be utilized by the path planner. The poses and locations of the robots and the obstacles should be accurate because all decisions and actions are decided by assuming perfect data. Moreover, the potential filed method needs to take into account the local minimal problem. Therefore, this approach usually is used in initially-known and structured environments or with the availability of a powerful sensing system.

With the development of machine learning and artificial intelligence (AI), more researchers have focused on solving a global path planning problem by decomposing it into local problems. Bug1 and Bug2 algorithms (Lumelsky et al., 1987) are among the earliest, simplest schemes in this category. They are search approaches which are designed to move the robot towards the goal, going around obstacles and following the boundary of obstacles. Yufka et al. (2009) applied Bug1, Bug 2 and DistBug motion planning algorithms in a Pioneer mobile robot, and compared the resulting performance. The difficulty of these approaches is in finding the so-called leave point when the robot is following the boundary of the obstacles.

Another candidate for solving the mobile navigation problem in machine learning is reinforcement learning, especially Q-learning. Reinforcement learning has been studied by psychologists since the 1940's (Sutton and Barto, 1998). It involves learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. It is considered neither supervised learning nor unsupervised learning. The learner is a decision-making agent who takes actions in an environment and receives rewards for its actions in trying to solve a problem (Alpaydm, 2004). After a set of trial-and-error runs, it should learn the best policy, which is the sequence of actions that maximizes the total reward. Su and his colleagues (2004) applied reinforcement learning in a robot navigation application. Fuzzy rules were ultilzed in their work to reason the sensory data and provide them to the reinforcement learning module for appropriate selection of actions. Ying and de Silva (2010) proposed a sequential Q-learning algorithm with Kalman filtering (SQKF) in a multi-robot cooperation project. Their method solves the problem of behaviour conflicts by conditioning the credit assignment.

1.4.3 Visual Servoing

The earliest research of visual servoing was reported in 1979 (Agin). However, then the image processing procedure took seconds due to the limitation of computers and image sensing devices at that time, making real-time control virtually infeasible. Subsequently, Sanderson and Weiss (1980) introduced a taxonomy of visual servo systems through a control structure. It is called Dynamic Look-and-Move Structure where the control architecture is hierarchical and uses vision to provide set-point inputs to the joint-level controller. Then the sub-control system utilizes joint feedback to

internally stabilize the robot (Figure 1.3). In contrast, direct visual servoing utilizes a visual servo controller which directly relies on vision information to compute joint inputs, thereby stabilizing the robot (Figure 1.4).

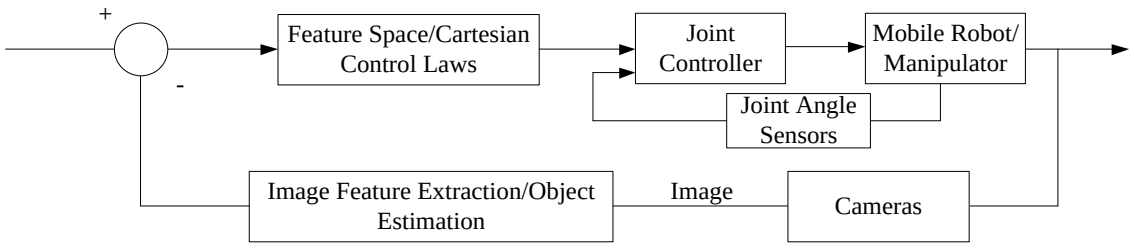


Figure 1.3: Block diagram of a dynamic look-and-move system.

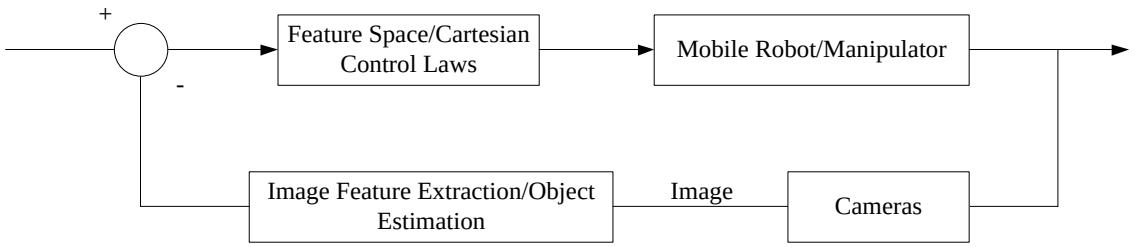


Figure 1.4: Block diagram of a typical (direct) visual servo system.

With the rapid advancement of computer technologies and image sensing hardware (CCD and CMOS), computer vision is much faster now than it did in 1980s. Direct visual servoing came to attention of researchers in the 1990s. Hutchinson has reviewed many of the related work (Hutchinson, et al., 1996). Since then the term “visual servo” has come to be accepted as a generic description for any type of visual “feedback control” of a robotic system. The subject has been under study in various forms for more than twenty years, in contexts ranging from simple pick-and-place tasks to today’s real-time, complex tasks involving multiple robots and objects, autonomous cooperation, and dynamic, unstructured and unknown environments.

Another major classification of vision-based servoing distinguishes between position-based approach and image-based approach (Chaumette and Hutchinson, 2006). Both approaches share a similar control block diagram with slight a difference in the control feedback loop and the reference input (Figure 1.5). In the former, features are extracted from images from one or more cameras, and used in conjunction with camera models and a geometric model of the target object to estimate the pose of the target with respect to

the cameras. The controller seeks to reduce the error between the current pose and the desired pose in a 3D (three-dimensional) workspace. In contrast, image-based visual servo control uses the 2D images (and their visible feature points) directly. Consequently, image-based visual servo control reduces the computational burden, omits unnecessary image interpretation, and eliminates the calibration errors in sensors and cameras.

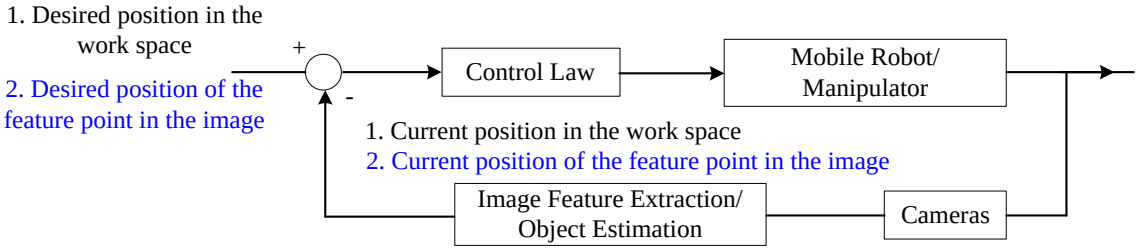


Figure 1.5: Block diagram of a typical visual servo system (1. Position-based; 2. Image-based).

Much of the work related to visual servoing and mobile robots to date has focused on the application of autonomous navigation control. Physical demonstrations of mobile manipulation and grasping using visual servoing have been somewhat limited. Some examples are indicated now. Ma et al. (1999) have developed a vision-guided navigation system where a non-holonomic mobile robot tracks an arbitrarily shaped continuous ground curve. They formulated this problem as one of controlling the shape of a curve on the image plane, and presented corresponding control laws. Dixon et al. (2006) presented an adaptive tracking controller of a wheeled mobile robot via an un-calibrated camera system. In their paper, the parameter uncertainty of the mechanical dynamics and the camera system was considered, and an adaptive controller was proposed to cope with the uncertainty. In order to implement robust vision-based autonomous navigation systems, various approaches have been explored, such as planning of image-trajectory or image-memory (Remazeilles et al., 2007; Dixon et al., 2001), embedded velocity fields (Kelly et al., 2006), specific geometry features (vanishing points and line orientations; Zhang et al., 1999), stereo cameras, and omnidirectional or catadioptric cameras (Mariottini and Prattichizzo, 2008; Chang and Hebert, 2000; Gaspar et al., 2000).

Epipole-based or homography-based techniques besides stereo cameras have been used for estimating 3D parameters or depth information of target objects in visual servo control of mobile robot navigation (Chen et al., 2006; Fang et al., 2005; Lopez-Nicolas et

al., 2007; Mariottini et al., 2007; Chesi et al., 2006). In these papers, the rotational and translational relationships between two camera frames were derived by relating the current image to the desired image of the same target object, which were then used in visual-servo control.

It has been noticed that robustness and response speed are two important issues in vision-based mobile navigation systems, and hybrid controllers have been developed to meet these challenges. For example, a vision-based hybrid control scheme has been developed (Amarasinghe et al., 2007) for autonomous parking of a mobile robot. Its hybrid controller includes a discrete event controller to change the direction of travel of the robot and a pixel-error-driven proportional controller to generate commands to achieve its continuous motion. A similar project is presented by Vassallo (2000), where a vision-based mobile robot attempts to autonomously navigate in a building. A visual-servo controller that uses vanishing point data is combined with an appearance-based navigation controller to provide extended autonomy even under modest computational resources.

It is clear that much of the research related to visual servoing and mobile robots concerns mobile robot navigation, and only a few physical implementations of vision-based mobile manipulation have been reported. The possible reason for this limited activity is that vision-based mobile manipulation requires accurate and robust positioning performance so that it is more challenging than vision-based mobile robot navigation. As an example of vision-based mobile manipulation, Luca et al. (2007) have developed an image-based visual-servo controller for non-holonomic mobile manipulators. In their paper, two well-known methods of redundancy resolution for fixed-base manipulators are extended for kinematic modeling of a specific non-holonomic mobile manipulator. Their approach is illustrated through computer simulation, not physical implementation and experimentation.

Recent work in vision-based mobile manipulation is presented in 2007 (Mansard et al.), where a visually-guided humanoid robot attempts to autonomously grasp an object while walking. They have proposed a high-level structure to sequence multiple control tasks so that the target object remains in the middle of the field of view of the camera when the robot walks along a planned path. When the robot is sufficiently close to the object, it grasps the object while continuing to walk.

Research of visual servoing has been challenged by an important requirement: how to keep the visual features within the field of view of the camera. To date, both image-based visual servoing (IBVS) and position-based visual servoing (PBVS) have used measurements of visual features to compute the controller outputs. If due to motion of the robot or some unknown disturbance, these visual features move outside the field of view of the camera the controller will completely fail. This issue even becomes more severe when visual servoing is applied to mobile robotic tasks, because mobile robots move over long distances than fixed-base robotic manipulators.

It is common for visual features to move out of the field of view due to camera/robot calibration errors or controller design in a visual-servo system. Since the visibility of the visual features directly affects the robustness of the system, a significant effort has gone into solving this problem. A popular solution is to employ the potential field approach to push visual features toward the center of the field of view when the features approach the edge of the image. A representative work in this area has been completed by Corke and Hutchinson (2001), where a potential function has been incorporated into an IBVS controller to repel visual feature points from the boundary of the image plane. Chesi and Hung (2007) employed a similar approach to solve the visibility constraint problem in their global path planner for optimal visual servoing. The approach of Navigation Function, which guarantees a unique minimum, has been introduced into image-space path planners of visual servoing to generate a desired image trajectory while keeping the visual features in the field of view (Chen et al., 2007; Cowan et al., 2002).

Another popular method of keeping visual features in the field of view is to employ a path-planning technique to plan a camera trajectory or a “virtual path” on the image plane while meeting the visibility constraint (Schramm et al., 2006; Zhang and Ostrowski, 2002; Chesi et al., 2007). It is common as well to employ a pan/tilt camera to enlarge the field of view so that the visual features will remain within it (Capparella et al., 2005; Nierobisch et al., 2006). Further approaches are available in the literature to solve the visibility problem. For example, Remazeilles and Chaumette (2007) have employed several specific visual features to ensure that the robot navigates within the visibility path. Cowan and Chang (2005) have developed a diffeomorphism-based IBVS controller to keep the features within the field of view and avoid self-occlusion.

There exists a common shortcoming in the approaches mentioned above, which are based on potential-field, navigation-function, or planning techniques: they represent static solutions designed in advance by a human. These approaches cannot improve their performance on line and also cannot autonomously adapt to changing control tasks.

Most visual-servo projects today primarily concern object modeling and the quality of the vision feature feedback while paying less attention to controller design. Notably, a simple P (Proportional) control law or a PID (Proportional, Integral, Derivative) control law is commonly used in the literature. However, a PID controller may not be adequate to handle the robustness and stability issues of real-life mobile robot applications. Spong and Hutchinson (2006) reviewed proportional control with Lyapunov stability, which is the controller that is most commonly used by researchers. Although this control law (Chaumette and Hutchinson, 2006, Spong et al. 2006) can guarantee system stability, its controller output is not optimal and it is unable to consider various constraints (robot location constraints, visibility constraints, velocity constraints, and so on) which are common in a mobile visual-servo system.

In order to get optimal controller outputs, Ginhoux et al. (2005) proposed to use GPC (generalized predictive controller) in visual servoing of a robotized surgery task. In particular, a 6 DOF (Degree of Freedom) robotic arm was developed to track the motion of a pig's heart based on the visual feedback. Since they employed the basic GPC scheme, no constraints were considered. In addition, they used the same idea in teleoperated laparoscopic surgery (Gangloff et al., 2006) and 3-D profile following (Gangloff and de Mathelin, 2002). Since the constraint issue is quite popular in visual-servo tasks, Sauvee et al. (2008) implemented a nonlinear model predictive controller for visual servoing of a robotic arm using vision feedback from ultrasound images. In this project, they considered various constraints and linearized the nonlinear model into a constant linear model at the equilibrium point.

Applying MPC (Model Predictive Control) to visual servoing is still in its juvenile stage. Only a limited number work has reported as above. Although some positive results were obtained in them, they shared a common shortcoming: in applying the existing (basic) MPC technique to visual-servo tasks, a constant linear model is assumed for the nonlinear plant. As a result, the implemented system can only work in a small neighborhood of the equilibrium point at which the model is linearized. If the current

operation point of the system is farther from the equilibrium point (which is common in a mobile manipulation task), the controller performance will deteriorate quickly due to the mismatch between the model and the plant.

1.4.4 Networked Sensing

It is known for a long time that sensor integration is fundamental to increasing the accuracy, versatility and the application domain of robots, but to date this has not proven cost effective for the bulk of robotic applications. Multi-sensor systems are designed to exploit several signature-generation phenomena and to gather different types of information about objects and scenes of interest. Design of a multi-sensor system involves optimization of sensors (including sensor location and orientation), data processing, and communication, and particularly the use of an appropriate fusion strategy for the sensory data; e.g., Bayesian and Dempster-Shafer inference; fuzzy logic; pattern recognition using signal processing algorithms, and artificial neural networks. The backbone of a multi-sensor system concerns how to utilize various streams of data by using an effective approach for sensor /data fusion.

Sensor fusion first appeared in the literature in the 1960s. Today, application of sensor fusion has expanded into a wide range of areas: machine health monitoring and diagnosis, maintenance engineering, robotics, pattern recognition, object tracking, and so on. Figure 1.6 shows a general architecture of a sensor fusion system which includes low-level data acquisition and processing and high-level data fusion processing. Low-level processing concerns the hardware level and the raw data processing; e.g., sensor selection and arrangement and data acquisition, while the high-level processing focuses on extracting the raw data of interest and transforming them into utilizable forms. A decision making system that uses the pre-processed information is also embedded in the high-level processing layer.

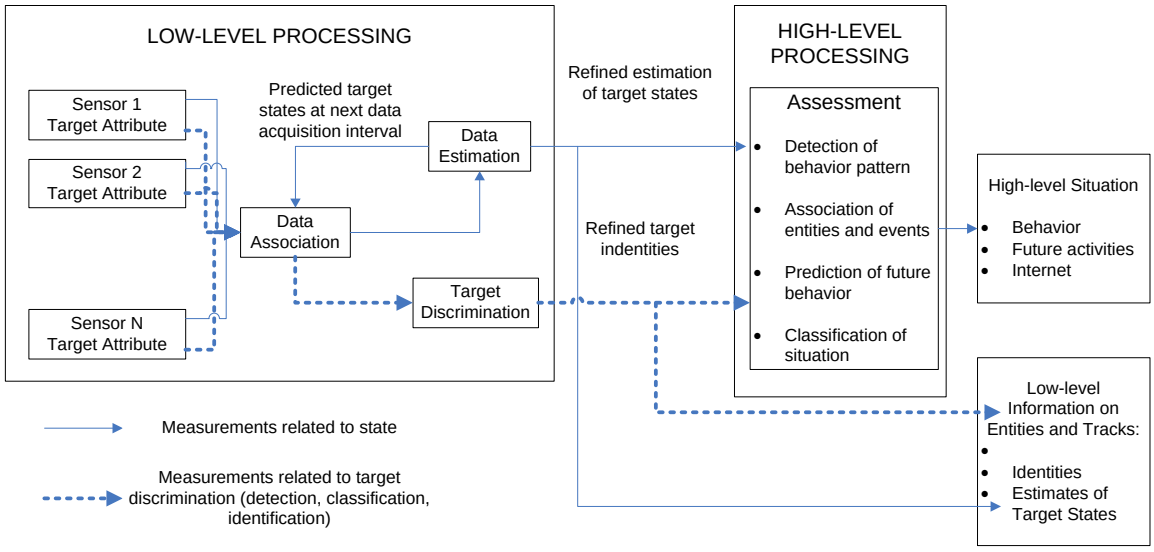


Figure 1.6: General architecture of data fusion processing.

Artificial neural networks (ANNs) is a commonly applied approach to solve the data fusion problem. It was introduced by Posner (1989) with the objective of understanding the functioning of the human brain. He built models of natural neural networks of the brain and carried out simulation studies. The general idea of ANNs is to make a nonlinear transformation from a *d*-dimensional input space to an *h*-dimensional output space through an appropriate number of hidden layers. Through appropriate training, the network weights are adjusted, establishing a reasonably accurate nonlinear relationship between the inputs and outputs.

Ghosh et al. (2007) proposed a neural network-based sensor fusion architecture for the estimation of tool wear of a computer numerical control (CNC) milling machine. Monitoring of tool wear is crucial in preventing the degradation of the machining quality. Unfortunately, there is no direct way of measuring the process variables related to tool wear. An ANN-based sensor fusion approach has been proposed by Ghosh et al. (2007) to fuse the data of cutting force, spindle vibration, spindle current, and sound level from different sensors. The approach had been validated by both laboratory and industrial implementation. Hong et al. (2005) developed a neural network-based sensor fusion system for real-time fault detection of reactive ion etching. The target of this project was to guarantee the system accuracy and real-time performance. Two in-situ sensors: optical emission spectroscopy (OES) and residual gas analysis (RGA) were used and the generated signals were sent to a time series neural networks (TSNNs) for fusing as well

as predicting the process parameters. Simulated fault processing data were used to train the NN and the authors claimed that this approach could potentially contribute to maintaining a consistent etching process by increasing the probability of identifying the incipient faults.

Considerable amount of work has been done in sensor fusion where fuzzy logic is implemented as the fusing method. Lotfi Zadeh (1978) developed the fuzzy set theory in 1965. Zadeh reasoned that the rigidity of the conventional set theory made it impossible to account for vagueness, imprecision, qualitative aspects, and shades of gray that are commonplace in real-world events. Consequently, fuzzy logic is valuable where the boundaries between sets of values are not sharply defined or there is partial possibility of occurrence of an event.

Although fuzzy logic and neural networks are structurally different, they share a rather complementary nature as far as strengths and weaknesses are concerned (Karray and de Silva, 2004). Applying fuzzy methods into the workings of neural networks constitutes a major thrust of neuro-fuzzy (NF) computing. Wang et al. (2004) proposed a neuro-fuzzy system to forecast damage propagation trend in rotary machinery and to provide an alarm before a fault reaches critical levels. After proper training, the performance of the NF was compared with the performance of a recurrent neural network (RNN). It showed that the NF was a reliable and robust machine health predictor which could capture the system dynamic behavior quickly and accurately. Palluat et al. (2006) designed an intelligent monitoring aid which used a neuro-fuzzy approach. The system contained a detection tool which used neural networks and a diagnosis tool which used a neuro-fuzzy approach. Four sensors were used in gathering information. After training, the NF demonstrated industrial usefulness in an application of monitoring a flexible production system.

The backbone of a multi-sensor system concerns how to utilize various data by using an effective approach for sensory data selection and fusion. Ray (2004) has proposed a novel concept of anomaly detection in complex systems by using Finite State Automata (FSA) and D-Markov machine. An application of sensor fusion has been reported by Chattopadhyay and his colleagues (2009). For sensing different types of objects, Chen (2004) proposed a dynamic architecture in an acoustic sensor network where the sensor nodes in a clustered sensor network adapts to different targets using Voronoi diagrams.

Yang and Sikdar (2003) utilized the Bayesian method to dynamically cluster and fuse the sensor information in a sensor network to achieve the task of tracking mobile targets.

1.5 Contributions and Organization of the Thesis

This Ph.D. thesis investigates and develops new techniques and expertise that will facilitate the implementation and effective operation of mobile robot manipulation systems in unknown, unstructured and dynamic environments. The four main contributions of the thesis are as follows:

- Efficient and robust machine vision algorithms are developed, which utilize color and feature of objects. With the integration of other sensors, they help to determine the pose of the robot in the workspace. They are incorporated into the feedback control loop of motion of the mobile robot platform and manipulation of the robotic arm by providing accurate and fast position information of the tracked object.
- A traditional reinforcement learning algorithm, specifically Q-leaning, is incorporated to enhance the operation of the mobile robot in an unknown, unstructured and dynamic workspace. With the help of a sensor network, it will guide the mobile robot to approach the object of interest with increasing effectiveness.
- An adaptive nonlinear model predictive controller is developed for accurate motion control of the robot when the object is within its local sensing area, and also for effective manipulation. The developed controller takes into account visibility constraints and physical constraints, and it is able to provide optimized controller outputs.
- A Probabilistic Finite State Automata (PFSA)-based, information-driven, and self-organized sensor network is proposed to dynamically cluster sensors in order to improve the decision making associated with the robot operation.

The organization of this thesis is as follows: The present chapter (Chapter 1) introduces existing activities in mobile robotics and highlights main research challenges of this field. Next it outlines the research objectives of the thesis and presents a literature survey to establish the related background work, mainly in the past decade. Chapter 2 introduces and discusses the machine vision techniques that are utilized in the identification and tracking of objects. The methods include color blob tracking, Scale

Invariant Feature Transform (SIFT) feature tracking and stereo vision. Their practical application in mobile manipulation is indicated. In Chapter 3, the conventional method of reinforcement learning (Q-learning) is customized and incorporated in mobile robot navigation. The detailed formulation of the states and actions of the algorithm is presented. This algorithm is then implemented in computer simulation and experimentation. Chapter 4 addresses several challenges in the area of mobile manipulation such as visibility constraint, physical constraints and optimal controller outputs. First, a hybrid controller, which combines a traditional proportional-integral-derivative (PID) controller and an intelligent Q-learning controller, is proposed. It mainly addresses the visibility constraint. Then a more advanced controller, termed Adaptive Nonlinear Model Predictive Controller (ANMPC), is proposed and developed for both mobile navigation and robotic manipulation. This approach is able to solve problems of visibility constraint and physical constraints, and also provide optimal controller outputs. Chapter 5 introduces a self-organized sensor network where Probabilistic Finite State Automata (PFSA) is utilized to organize and cluster suitable sensors and then fuse their data to make reliable and more accurate decisions. It has the ability to communicate with the robots in the workspace and provide information to assist the execution of the robotic tasks. Experimental investigation using the developed robotic system is presented in Chapter 6 along with discussions of the experimental results. Chapter 7 summarizes the primary contributions of the thesis, and indicates several relevant issues and possible future research directions in mobile manipulation.

CHAPTER 2 Machine Vision

Computer vision involves imaging of objects using cameras and high-level processing of those images to extract features and interpret objects. Machine vision (MV) is often considered the application of computer vision to industrial and manufacturing systems. Whereas computer vision is mainly focused on computer-based image processing, machine vision most often requires digital input-output devices and computer networks to control other manufacturing equipment such as robotic arms. Machine Vision is a subfield of engineering that encompasses computer science, image processing, mechanical engineering, and industrial automation. A common application of Machine Vision is the inspection of manufactured goods such as semiconductor chips, automobiles, food and pharmaceuticals; process control; and robot guidance in industrial applications (Steger et al., 2008; Graves and Batchelor, 2003).

In this thesis, machine vision is mainly applied for identifying and tracking objects as the robot navigates in its work environment. As presented in section 1.4.1, many methods are currently available for object identification and tracking. Among these methods, color tracking is one of the fastest and most straightforward one because the principles of the algorithm of color identification and tracking works at pixel level. In this chapter, a fast object tracking algorithm based on color is developed to provide position information of an object to the visual servo controller of a robot. However, the color tracking algorithm has its limitations. In particular, the object must have unique colors and the colors of the environment have to be different from the tracked colors. Because of these limitations, color tracking algorithms are not natural and adaptive to different scenarios. Next, a more robust and reliable feature-based object identification is introduced, which utilizes the SIFT features of objects to achieve object identification. Finally, a stereo vision system is presented for acquiring the depth information of a detected object.

2.1 Color Tracking

2.1.1 RGB and HSI Color Spaces

In the RGB (Red, Green, and Blue) color space, all colors are considered a combination of the three colors red, green and blue. It is based on a Cartesian coordinate system, which is shown in Figure 2.1. There are three axes representing Red, Green and Blue. The cube represents the overall RGB color space with eight corners: Red, Green, Blue, Cyan, Magenta, Yellow, Black and White. All values in this space range from 0 to 1.

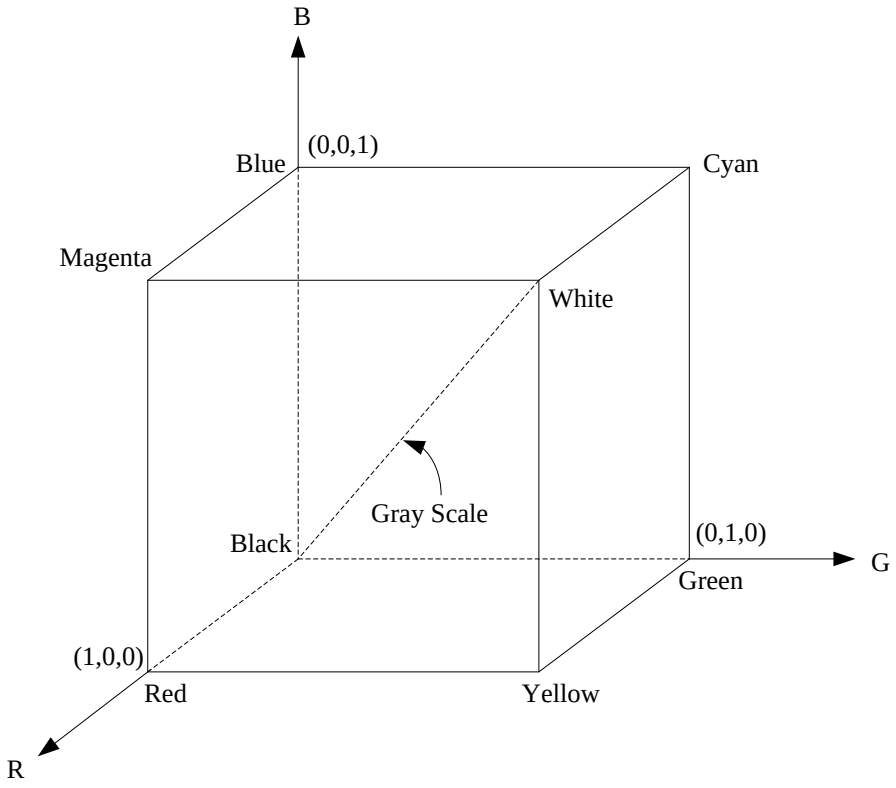


Figure 2.1: The RGB color model.

The HSI (Hue, Saturation, and Intensity) color space utilizes Hue, Saturation and Intensity to represent colors. Figure 2.2 illustrates the HSI color space model in a hexagon where its six corners represent Red, Yellow, Green, Cyan, Blue and Magenta. In this model, the color is decided only by the angle of Hue which ranges from 0 to 360 degrees. The saturation and intensity values do not contribute to the color, which means that the color cannot be changed by changing the color density and the lighting

conditions. This is the biggest advantage of the HSI color space over the RGB color space in color tracking because a change in any channel R, G or B will cause a change in the color.

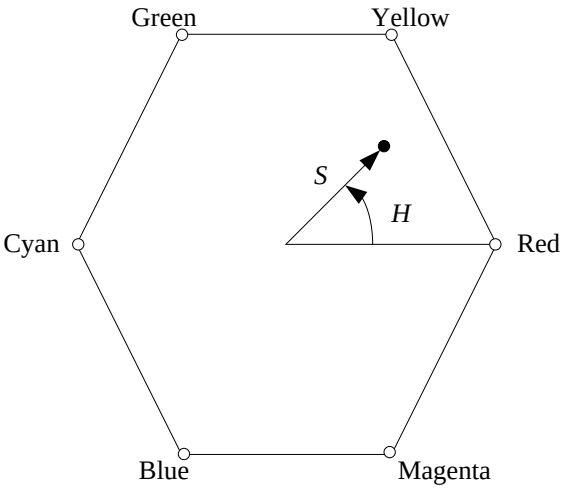


Figure 2.2: The HSI color model.

2.1.2 RGB to HSI Conversion

A common digital camera provides RGB signals. The first step of color tracking is to convert the RGB color space data into HSI color space data. Consider an RGB color point *P* that is acquired by a digital camera. The HSI value of the point *P* is given by:

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases} \quad (2.1)$$

where $\theta = cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}$. The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)}[min(R,G,B)] \quad (2.2)$$

The intensity is given by $I = \frac{1}{3}(R+G+B)$ (2.3)

2.1.3 HSI to RGB Conversion

The inverse conversion from the HSI color space to the RGB space is expressed by the following steps:

If $0^{\circ} \le H < 120^{\circ}$,

$$R = I\left[1 + \frac{ScosH}{cos(60^{\circ}-H)}\right](1-S) \quad (2.4)$$

$$G = 3I - (R + B) \quad (2.5)$$

$$B = I(1 - S) \quad (2.6)$$

If $120^{\circ} \leq H < 240^{\circ}$:

$$R = I(1 - S) \quad (2.7)$$

$$G = I\left[1 + \frac{ScosH}{cos(60^{\circ} - H)}\right](1 - S) \quad (2.8)$$

$$B = 3I - (R + G) \quad (2.9)$$

If $240^{\circ} \leq H \leq 360^{\circ}$:

$$R = 3I - (R + B) \quad (2.10)$$

$$G = I(1 - S) \quad (2.11)$$

$$B = I\left[1 + \frac{ScosH}{cos(60^{\circ} - H)}\right](1 - S) \quad (2.12)$$

2.1.4 Object Identification

The procedure of the color tracking algorithm is shown in Figure 2.3. As the camera grabs an image, first a Guassian filter is applied to the image in order to smooth it by removing high frequency noise. Next, the color image, which is in the RGB (red, green, blue) color space, is converted into the HSI (hue, saturation, intensity) color space. This is done because RGB colors are very sensitive to illumination and the RGB values will change due to slight changes of the lighting condition; while the values of hue and saturation will not change when the lighting varies. Therefore, tracking the colors of specific hue and saturation values is more robust and reliable. Next, a threshold is set according to a specific color, and the grabbed images are processed for specific color tracking (Wang and de Silva, 2006). Finally, the position of the center of gravity of the tracked object is computed using $r = \sum_{i=1}^{n} r_i \, /n$ and $c = \sum_{i=1}^{n} c_i \, /n$.

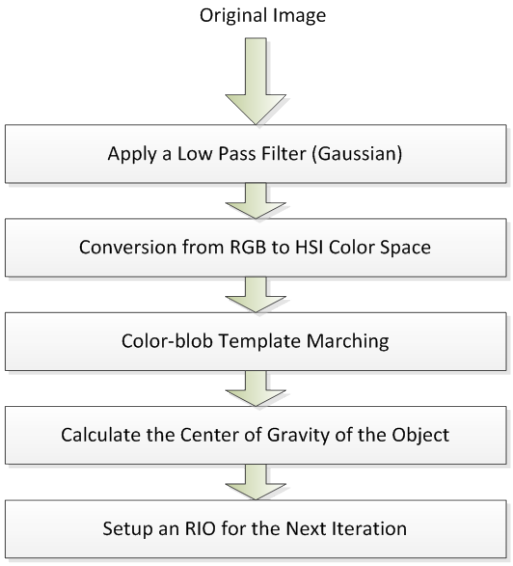


Figure 2.3: Procedure of the color tracking algorithm.

Figure 2.4 presents typical results from an application of color tracking. The objective of the application is to track an Amigo mobile robot while it is moving in the camera field of view. The left window in Figure 2.4 shows the raw data of the camera image, and the right window shows the result of identification. The white object in the image of the right window is the Amigo robot.

The main advantage of this color tracking algorithm is its high speed. It is able to provide the identification result almost instantaneously in real time. This will improve the performance of a visual servo control because time delay in the system can cause instability. However, it only works in a limited set of scenarios. In particular, if there are many objects with similar colors in the camera scene or if the background is noisy, which is a common situation, the efficiency of the color tracking algorithm will degrade and method may fail. Therefore, a more robust algorithm for object identification is desirable.

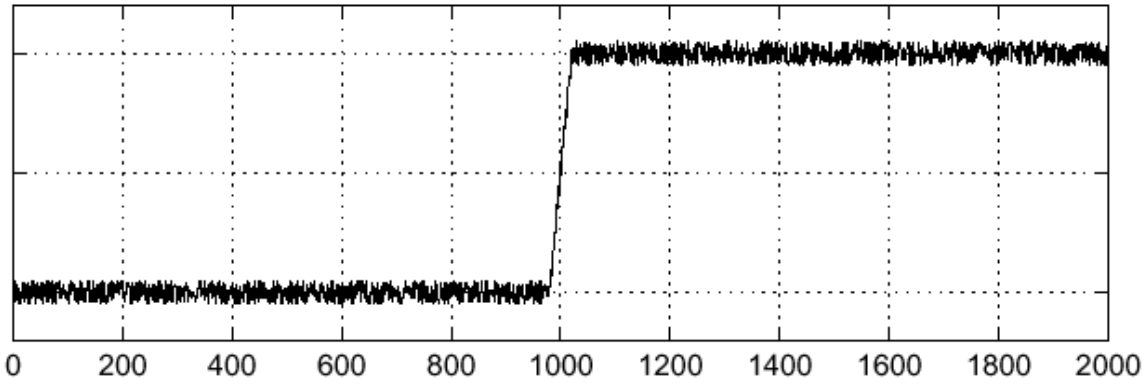
Figure 2.4: Implementation of the color tracking algorithm.

2.2 SIFT Feature Tracking

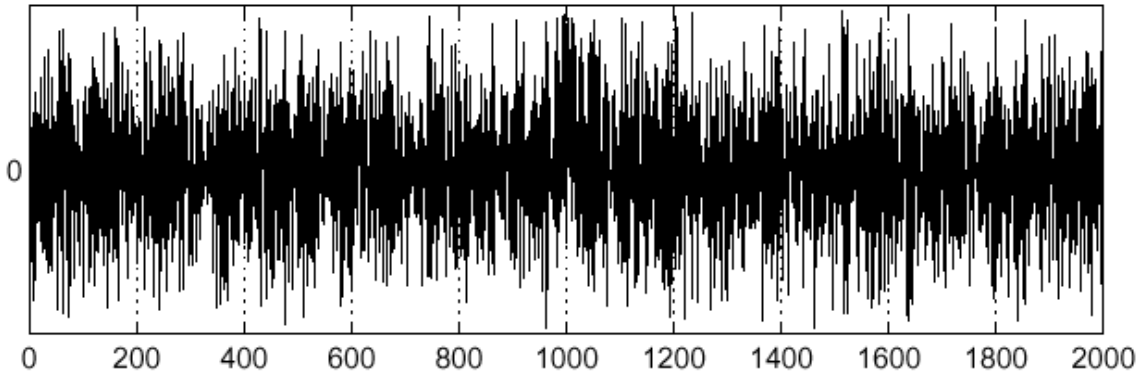
2.2.1 SIFT Feature Generation

The key to the approaches of feature-based object identification is to employ a robust feature detector with high repeatability with regard to rotation, illumination, and scaling. Repeatability of a feature detector, which evaluates the geometric stability under different transformations of images, is one of the most important criterions in choosing a good detector. It is given by the percentage of total detected features in the second image which is transformed from the first image. For example, if features detected in the first image of a video stream can also be detected in the second image by using the same detector, then the feature detector is said to have high repeatability. Edges are robust candidates for feature (Szeliski, 2011). In order to find the edges in an image, differentiation (or, gradient computation) is usually utilized. However, raw images usually contain high-frequency noise (Figure 2.5(a)). If a direct differentiation is applied on the raw data, the edge will not be found as shown in Figure 2.5(b). Application of a Gaussian convolution before taking the differentiation is known to solve the high frequency noise problem, as shown in Figure 2.6. Moreover, it can improve the efficiency by utilizing the Derivative of Gaussian (DoG) as shown in Figure 2.7. Mikolajczyk (2002) has found that the Laplacian of Gaussian (LoG) function, $\sigma^2\nabla^2 G$, generates most stable image features. However, Lowe proposed a Difference of Gaussian

(DoG) detector (Lowe, 1999, 2004) which provides an approximation to the LoG with a much lower computing effort.



(a) Raw data with noise.



(b) Processed data after taking derivative of the raw data.

Figure 2.5: An example of raw data and the results of its differentiation.

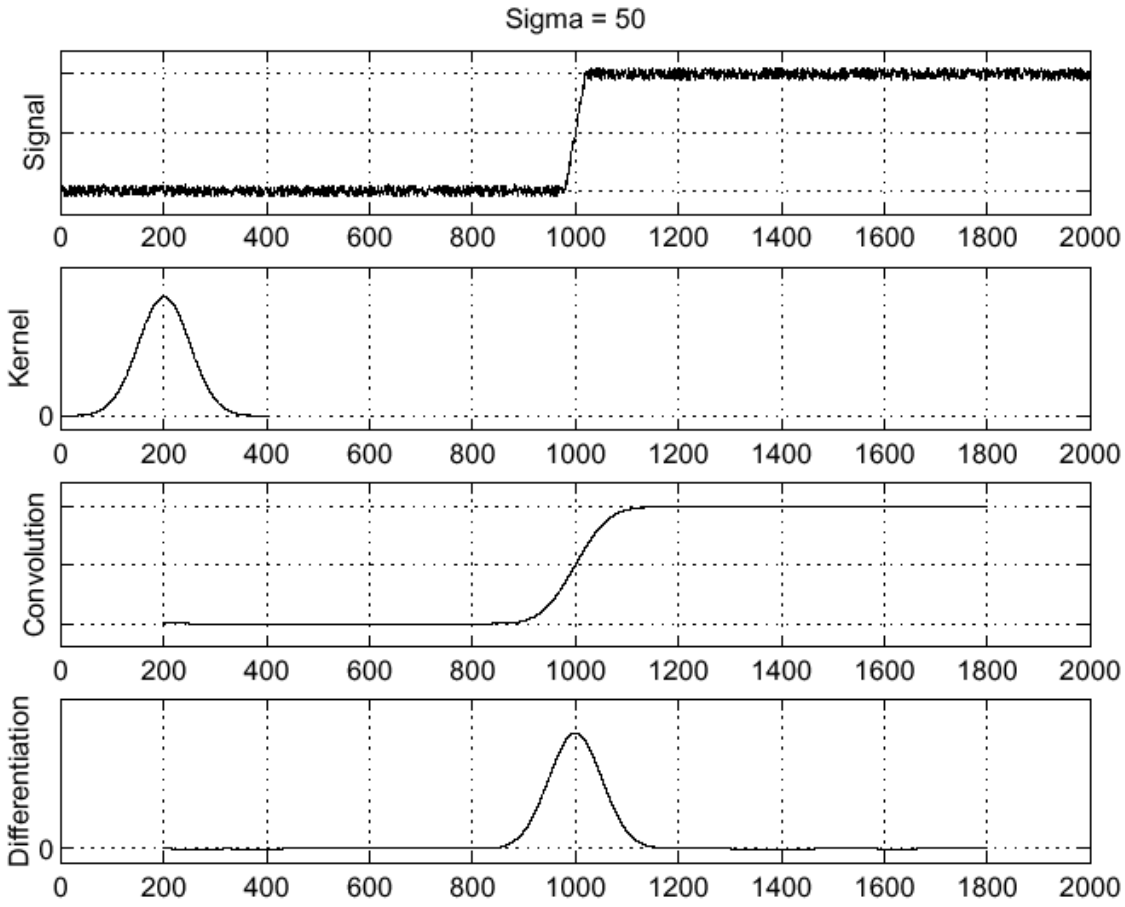


Figure 2.6: Guassian smoothing in edge detection.

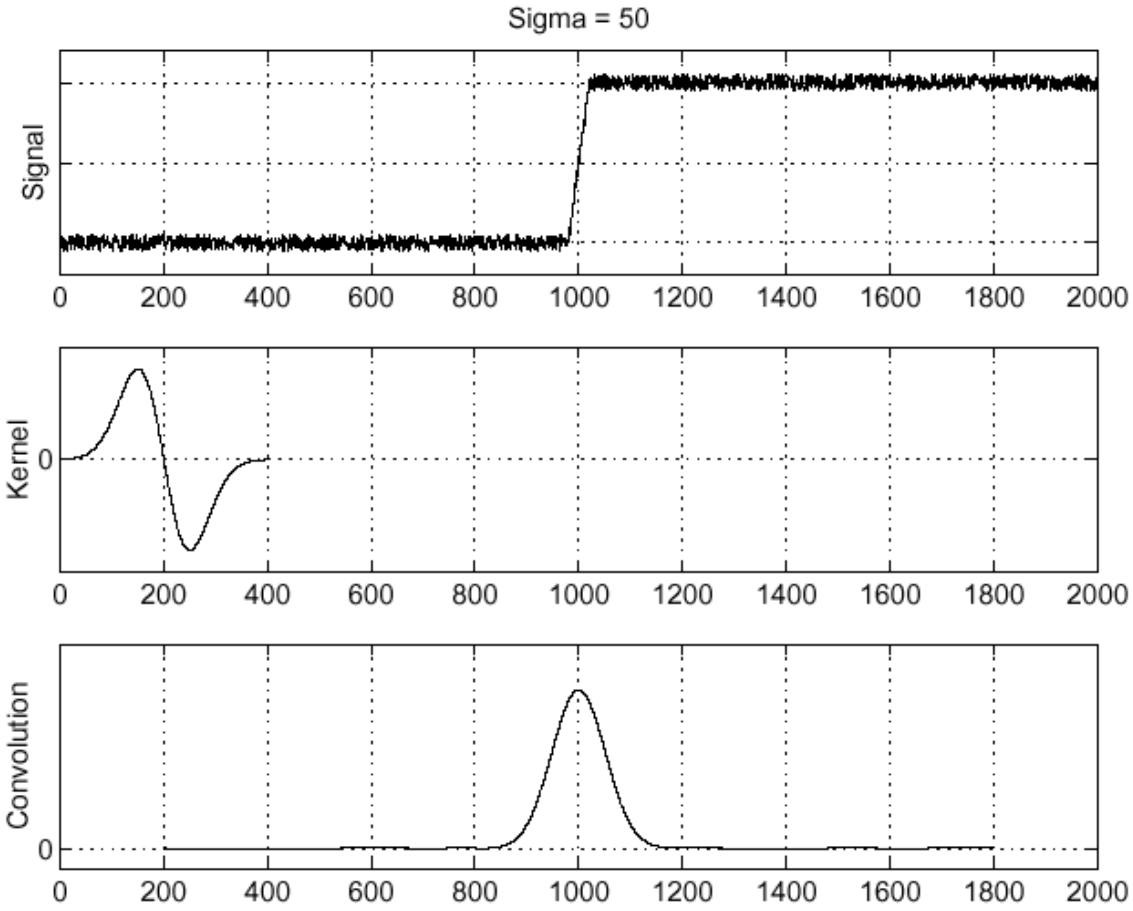


Figure 2.7: Derivative of Gaussian in edge detection.

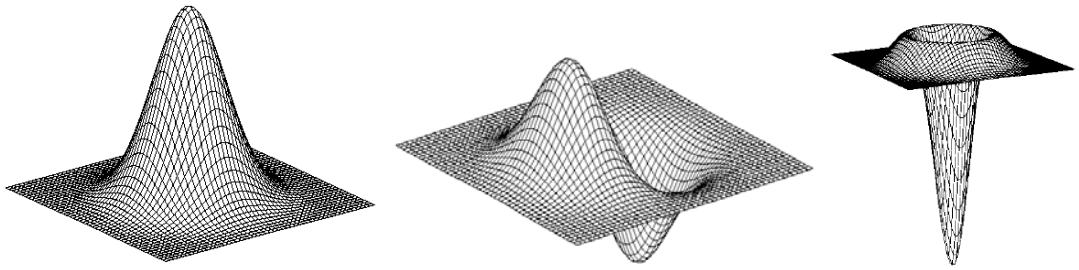


Figure 2.8: (1) Gaussian; (2) Derivative of Gaussian; (3) Laplacian of Gaussian.

The procedure of SIFT feature generation is presented in Table 2.1 where the raw images are first converted into grayscale images which are represented by a pixel matrix of unsigned 8 bit values. For convince of mathematical operation, each pixel is further converted into a double precision floating point ranging from 0 to 1.0 (0: black; 1.0: white).

Table 2.1: SIFT algorithm of local feature generator.

Input: A gray scale image.

Output: Local feature descriptors of the input image.

for the input image

Double the image size by using bilinear interpolation.

Build Gaussian DoG Pyramids.

Find local extremes as candidates of feature descriptor.

end

Eliminate unstable candidates.

Identify orientation of descriptor.

Generate local feature descriptors of the image.

return *descriptor* **vector**

In a natural scenario, the scales of objects in different images are different and unknown. Therefore, it is required that the features be stable in different scales. For this purpose, a DoG pyramid is generated for the image in different scales, as shown in Figure 2.9. The general idea of generating the DoG pyramid is presented now.

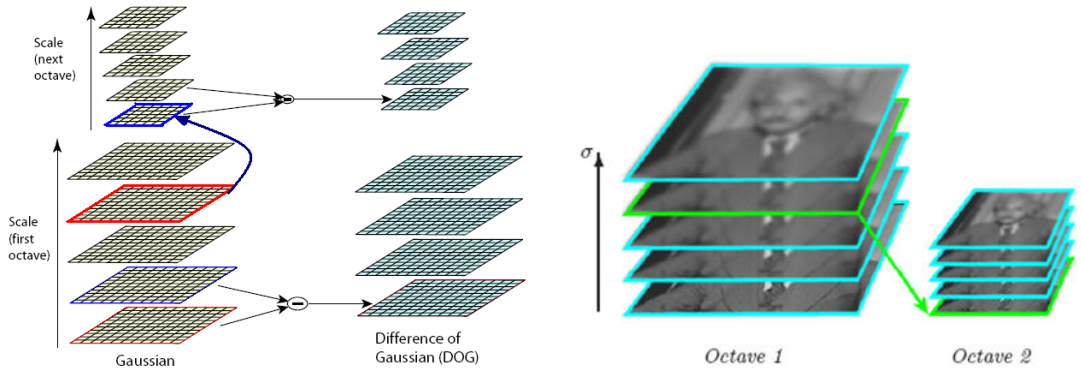


Figure 2.9: Difference of Gaussian (DoG) pyramid.

Suppose that the left-bottom of Figure 2.9 is the original image. A Gaussian blurred imaged is generated for the right second layer by using the convolution:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.13)$$

where $\sigma = 0.5$, $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right)$ (2.14)

The third, fourth and the fifth images are generated by using Gaussian blurring from the previous image; and the DoG images on the right of the Figure 2.9 are given by

$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,k\sigma)$ (2.15)

The feature points of an image are decided by finding the local extrema in different scales. Figure 2.10 shows three DoG images in the neighboring scales. Each pixel in the DoG images is compared with its 8 neighbor pixels in the same images and 9×2 corresponding neighboring scale. If the checked pixel has a maximum or a minimum value among these 27 pixels, it will be selected as a point of interest (Feature). For each point of interest, the gradient and orientation are calculated by using:

$g(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$ (2.13)

$\theta(x,y) = Atan2((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$ (2.14)

After eliminating key points with low contrast and along the edges, a set of feature points is generated. There are three parameters for each key point: location, gradient and orientation.

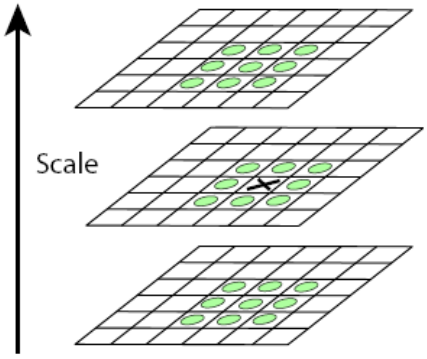


Figure 2.10: Local extrema.

Figure 2.11 shows the way in which the SIFT feature descriptors are generated (feature representation). The circle in the center represents a point of interest. First, an 8×8 pixel window is selected around the key point in the image. Second, the window is divided into 4 sub-windows in which each one contains 4×4 blocks. Third, the gradient histograms of the 8 directions are calculated. Accordingly, a SIFT feature vector contains

128 elements (4×4×8). Finally, the vector is normalized in order to be invariant to changes of illumination.

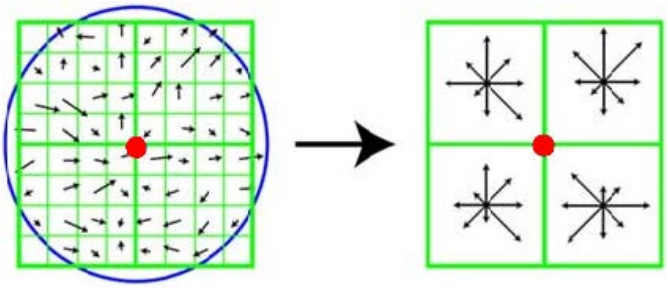


Figure 2.11: SIFT feature descriptor.

In order to find the object location in an image, the SIFT features of both the current image from the camera view and the template image are generated. The match points between the current image and the template are found by searching for the minimum of the Euclidean distance. The flowchart of the overall process of object identification is shown in Figure 2.12. First, the raw image is processed by utilizing digital image processing techniques. Then the SIFT feature of this image are generated using the steps discussed in the previous sections. The object of interest can be identified by comparing the SIFT features in the current image and the SIFT features in the template image.

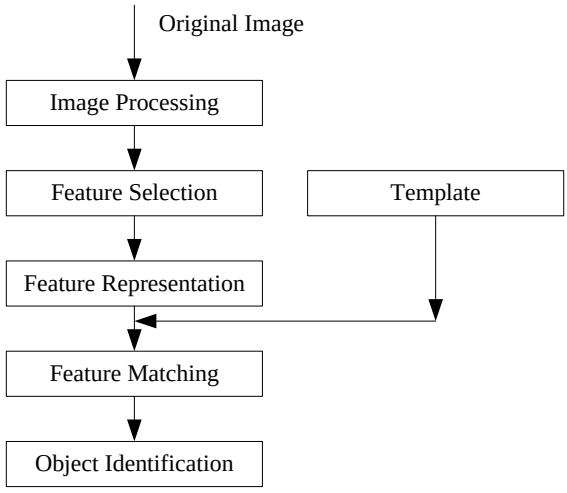


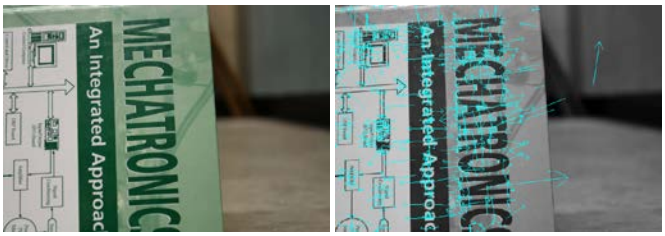
Figure 2.12: Flowchart of object identification.

2.2.2 Implementation of SIFT-based Object Identification

Figure 2.13 shows the results from an application of SIFT feature identification. The raw image is shown in Figure 2.13(a) which contains the object of interest: a book. Figures 2.13(b) and (c) show representations of the SIFT features of the template (the book) and the camera scene.

(a) SIFT keys of the object.

(b) SIFT keys of the camera view at the starting point.

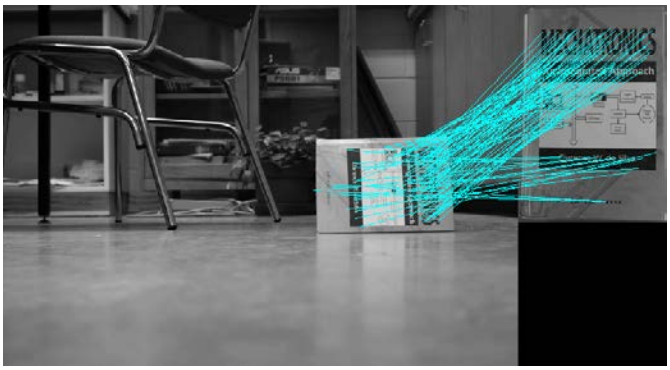


(c) SIFT keys of the camera view at goal location.

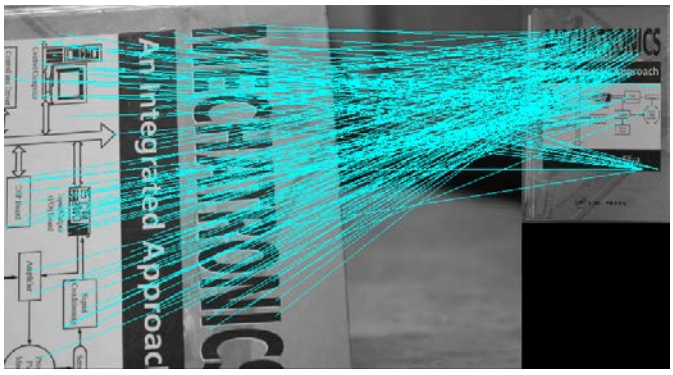
Figure 2.13: SIFT-based object identification and tracking.

The object is found by marching the SIFT features between the book template and the camera scene, as shown in Figure 2.14(a) and (b), which present two views from the on-board camera. Figure 2.14 (a) shows the camera view when the mobile robot is away

from the object, and Figure 1.14 (b) shows the camera view when the robot arrives at the goal location (grasping location). This experiment verifies good performance of SIFT feature-based object identification, especially when the object of interest and the environment have many features. However, the computational time of this method is rather excessive. Therefore, it is not suitable for visual servoing. However, it is a good candidate for object identification and tracking using global cameras in a sensor network where it has more time for decision making than in a visual servo application.



(a) Feature matching between current camera view and template.



(b) Feature matching between camera view of the goal location and template.

Figure 2.14: Examples of SIFT based feature matching.

2.3 Stereo Vision

Stereo vision is used for depth estimation of an image. A visual servo application requires the position of the object in the image plane as well as the depth information, which is the distance between the camera and the object with respect to the camera frame. A stereo camera can provide these two data values simultaneously. A stereo camera has two sets of cameras as shown in Figure 2.15. A point P_L in the world space

has projections on the two image frames with coordinates (u_R, v_R) and (u_L, v_L), as shown. The following equations can be derived from Figure 2.15:

$$(u_L, v_L) = \left(f\frac{X}{Z}, f\frac{Y}{Z}\right) \quad (2.15)$$

$$(u_R, v_R) = \left(f\frac{X-B}{Z}, f\frac{Y}{Z}\right) \quad (2.16)$$

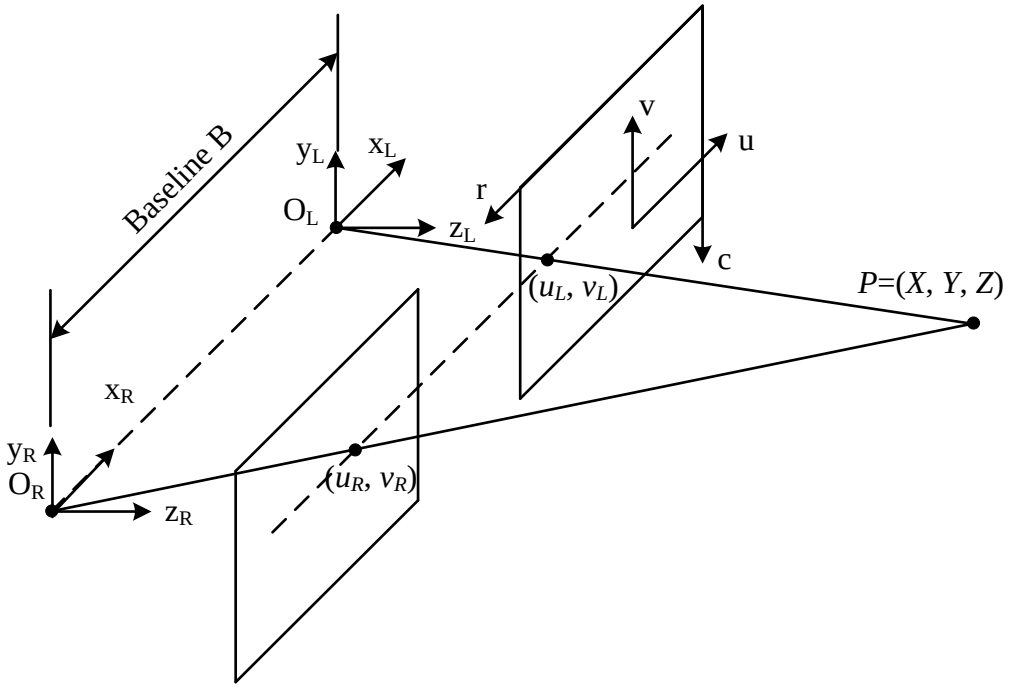


Figure 2.15: Model of a stereo camera.

Suppose that a point *P* (Figure 2.16) is observed by two cameras simultaneously. It projects to the image planes of these two cameras at *p* and *p'*, as shown. Note that *O* and *O'* are the optical center of these two image planes. The line *l'*, which is in image II, is called the epipolar line associated with point *p* of image *I*. Aso, *eL* and *eR* are called epipoles of the two cameras. The epipolar e_R is the projection of the optical center O_L in the right camera frame and so on.

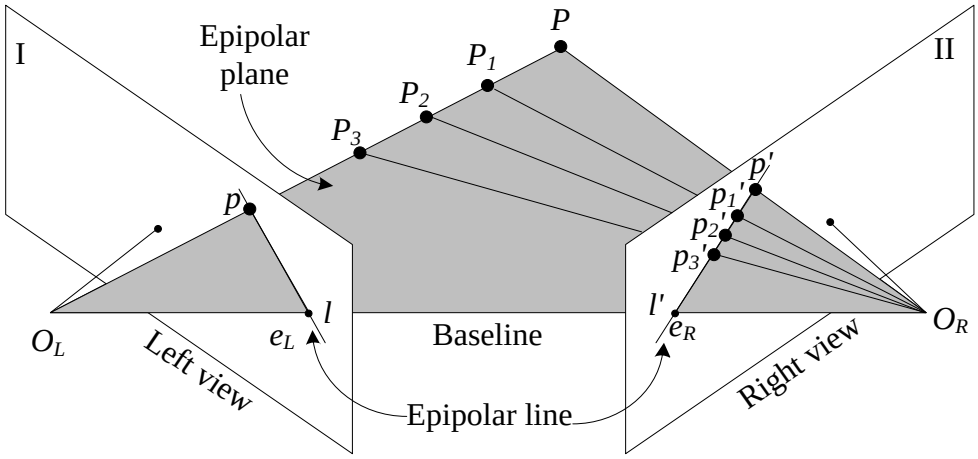


Figure 2.16: Epipolar geometry.

Epipolar constraint says that if *p* and *p'* are the projections of the same point *P* in different cameras, then *p'* must lie on the epipolar line associated with *p*. This result plays a fundamental role in stereo vision.

The epipolar geometry can be simplified if the two camera image planes coincide, as shown in Figure 2.17. In this case, the epipolar lines also coincide (*EL–PL = ER–PR*). Furthermore, the epipolar lines are parallel to the line *OL–OR* between the focal points, and can in practice be aligned with the horizontal axes of the two images. This means that for each point in one image, its corresponding point in the other image can be found by looking only along a horizontal line.

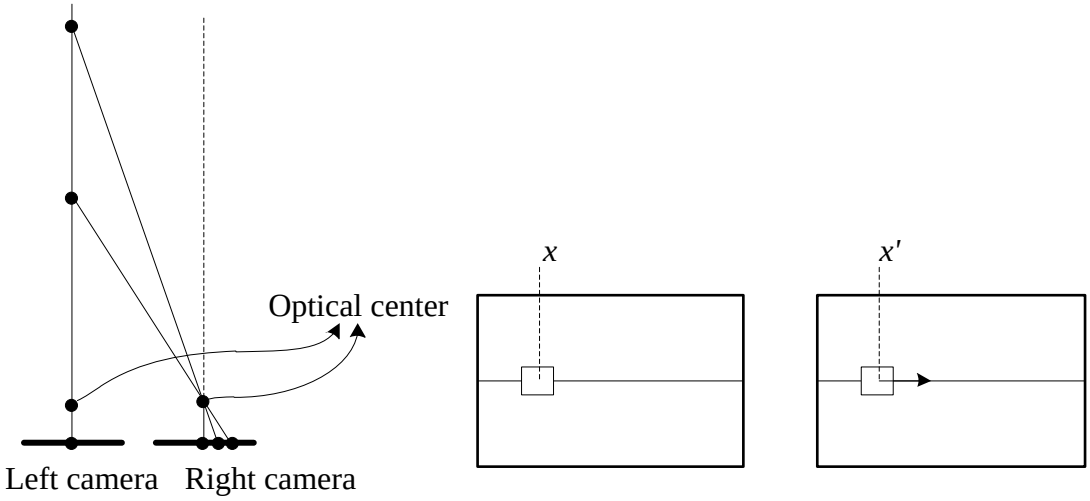


Figure 2.17: Simplified case of Epipolar geometry.

Therefore, the term “disparity” is defined as $d = u_L - u_R = f\frac{B}{Z} \rightarrow Z = f\frac{B}{d}$ (2.17) where Disparity = k/depth. Here k is a system parameter, which can be acquired by calibration (2.17). By introducing the Epipolar geometry and its constraints, the stereo vision problem is converted to a feature identification problem. In other words, the distance between the camera and the object in terms of the camera coordinates can be found by finding the position difference of the object in the right and left images (Figure 2.18). Since it is applied in visual servo control, the achieved efficiency is significant. A Normalized Cross Correlation (NCC) is utilized for searching most similar features between these two images according to:

$$NCC(A,B) = \frac{\sum_{(i,j)\in W} A_{ij}B_{ij}}{\sqrt{\sum_{(i,j)\in W} A_{ij}^2 \sum_{(i,j)\in W} B_{ij}^2}} \qquad (2.18)$$

where A, and B are the current position of the candidate features in the left and right images, respectively.

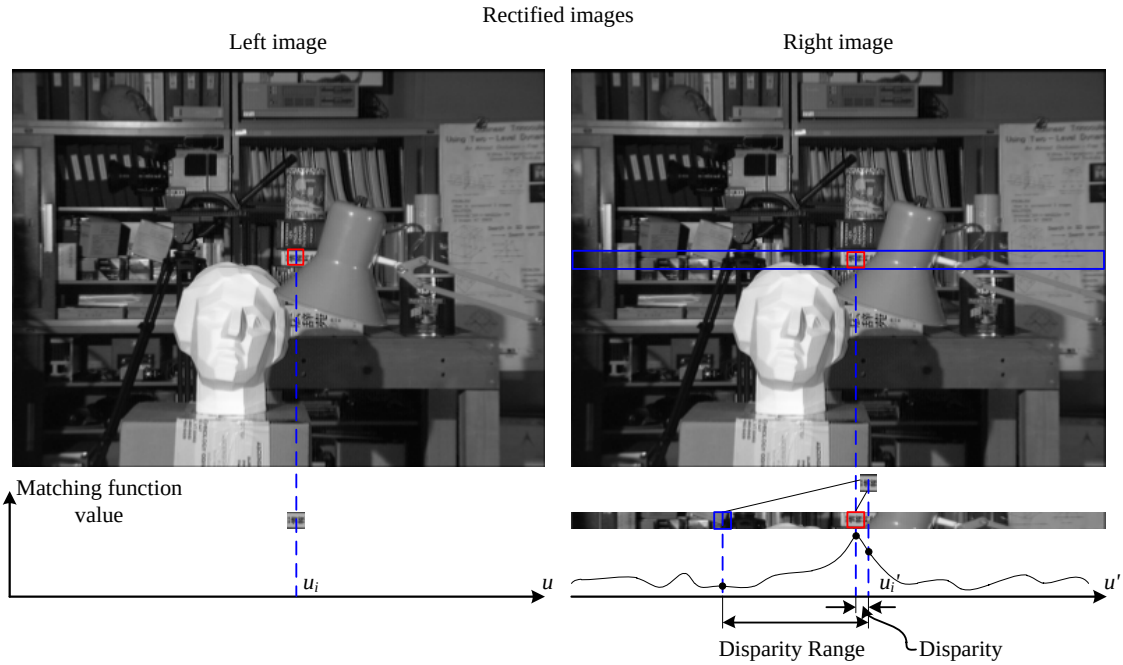


Figure 2.18: Disparity of features in the left and right camera scenes.

CHAPTER 3 Mobile Navigation

Navigation of mobile robots is an important research area among the research community of mobile robotics. This importance arises because understanding of the workspace and the availability of efficient motion strategies are necessary to effectively achieve a motion goal while giving proper consideration to the current pose, goal location in the workspace and possible obstacles on the path.

In this chapter, a method is developed for mobile localization and object post estimation in robotic navigation. Simulation experiments are carried out to show the performance of the method. Next a navigation system that uses Q-learning is developed for moving a mobile robot from its current position to a goal location in a dynamic and unstructured environment. A training software is developed to train and acquire a Q-table knowledge base, which is necessary in applying Q-learning. This methodology and the developed software are implemented in the physical mobile robot navigation system in our laboratory.

3.1 Mobile Localization and Object Pose Estimation

3.1.1 Sensors in Mobile Localization and Object Pose Estimation

Mobile robot localization and object pose (position and orientation) estimation in a work environment have been central research activities in mobile robotics. Significant research attention has been to these two issues in the past decade because of the successes of mobile robotic implementations such as vacuum cleaning robots, delivery robots, and elder care robots, which heavily rely on the capabilities of accurate robot localization and object detection. Solution of the problem of mobile robot localization requires addressing of two main problems (Siegwart and Nourbakhsh, 2004): the robot must have a representation of the environment; the robot must have a representation of its

understanding regarding its pose in this environment. Sensors are the basis of addressing both problems.

Many off-the-shelf sensors (e.g., GPS, compasses, gyroscopes, and ultrasonic sensors) that are available for mobile robots are introduced in (Siegwart and Nourbakhsh, 2004), giving their operating principles and performance limitations. Based on this introduction, ultrasonic sensors (Leonard, and Durrant-whyte, 1991), goniometers (Bonnifait and Garcia, 1998), laser range finders (Arsenio and Ribeiro, 1998), and CCD cameras (Yamamoto et al., 2005) are the commonly applied sensors in mobile robot localization projects for acquiring information on robotic pose estimation at high precision. Sonar is fast and inexpensive but is usually rather crude, whereas laser scanning is active, accurate and widely applied in mobile robotics. Vision systems are passive and have high resolution, and are the most promising sensors for future generations of mobile robots. In the work of Uğur et al. (2007), a 3-dimensional (3-D) laser scanner was applied to perceive the traversability affordance and used to wander in a room filled with different types of objects (spheres, cylinders and boxes). The results obtained through training showed that a mobile robot could wander around while avoiding collision with non-traversable objects, but traversable objects are handled by rolling them out of its way. A vision-based algorithm for mobile robot localization and mapping, which uses STIF (scale-invariant image feature) has been applied for mobile robot localization and map building (Borenstein et al., 1996). The associated experiment showed that visual landmarks were robustly matched and the pose of the robot was estimated in a 3D map. Previous work on robotic localization has indicated that laser scanners and CCD cameras are two promising sensors for this purpose.

The work presented in this session is a part of the mobile navigation system which we are developing, where a mobile robot detects obstacles, boxes in this case, in order to find suitable paths for navigation. Figure 3.1 presents the general scheme of mobile robot localization and object pose estimation in the present work. Successful localization and object detection involve the following three steps. Perception: the robot must interpret its sensor data to extract meaningful information; prediction: the robot should determine its global pose based on meaningful information; matching and pose update: the robot should detect objects and compute their poses with respect to a local coordinate system, and update their global poses.

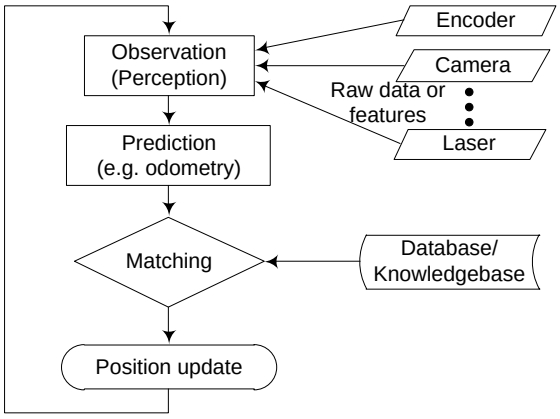


Figure 3.1: General scheme of mobile robot localization and object detection.

First, optical encoders and an odometry model are utilized to determine the pose of a mobile robot with respect to a global coordinate system. Next, a CCD camera, which is a passive sensor, is used to find objects (boxes in the present application) in the environment as well as the vertical surfaces of the objects (boxes). By identifying and tracking the color blobs that are attached to the center of each surface of a box, the robot rotates and adjusts its base pose to move the detected color blob into the center of the camera view. Finally, a laser range finder, which is mounted on the top of the mobile robot, is activated to measure the distance and the angle between the laser source and the laser contact surface on the box. Based on the information acquired in this manner, a homogeneous transformation matrix is applied to represent the global pose of the robot and the box. The developed approach is validated using the Microsoft Robotics Studio simulation environment.

3.1.2 Global Pose Estimation

Estimation of the pose of a mobile robot is a fundamental problem, which can be roughly divided into two classes (Borenstein et al., 1996): methods for keeping track of the robot's pose; and methods for global pose estimation. Much of the research carried out to date has concentrated on the first class, which assumes that the initial pose of the robot is known. A commonly used method for global pose estimation in this field is the odometry model, which determines the pose of a robot relative to a starting point during navigation of a wheeled vehicle.

Figure 3.2 shows a movement of a differential-drive robot, as used in the present work. Suppose that the initial pose of the robot is completely known. Then, real-time pose information of the mobile robot can be calculated using rotation measurements of the two wheels. The overall procedure of the estimation is outlined below. The location and orientation of the mobile robot in the global coordinate system are represented by the vector:

$$p = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \tag{3.1}$$

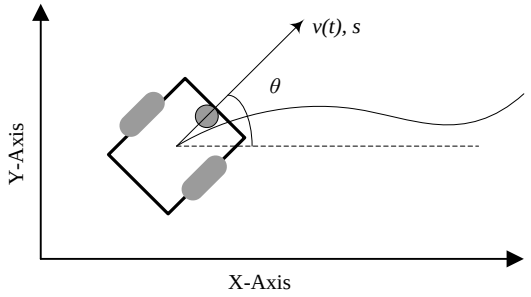


Figure 3.2: Motion of a differential-drive robot.

For a differential driving robot, the pose can be estimated from the starting pose by integrating the travel distance in each interval (during the fixed sampling interval Δt of sensors) using

$$\Delta x = \Delta s \cos(\theta + \Delta\theta / 2) \tag{3.2}$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta / 2) \tag{3.3}$$

$$\Delta\theta = (\Delta s_r - \Delta s_l) / d \tag{3.4}$$

$$\Delta s = \sqrt{\Delta x^2 + \Delta y^2} = (\Delta s_r + \Delta s_r) / 2 \tag{3.5}$$

Here, Δx and Δy are distances traveled in the last sampling interval along the x and y directions, respectively; $\Delta\theta$ is the travel angle with the last sampling interval; Δs_l and Δs_r are travel distances of the left and the right wheels, respectively; and d is the distance between the two wheels, which is a constant for a given robot. Therefore, the updated position p' for each interval can be found using

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta / 2) \\ \Delta s \sin(\theta + \Delta\theta / 2) \\ \Delta\theta \end{bmatrix} \tag{3.6}$$

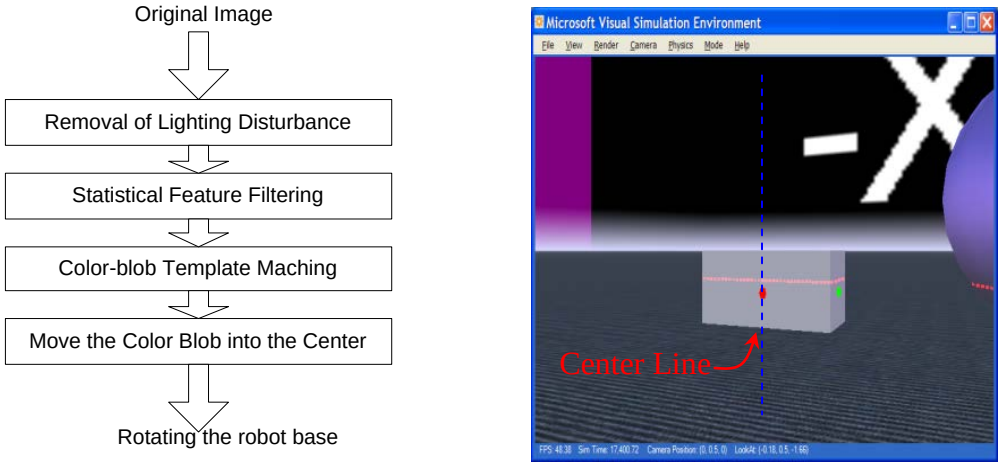
By using the equations (3.4), (3.5) and (3.6), p' can also be computed from the following equation:

$$p' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} ((\Delta s_l + \Delta s_r)/2)\cos(\theta + (\Delta s_l + \Delta s_r)/2d) \\ ((\Delta s_l + \Delta s_r)/2)\cos(\theta + (\Delta s_l + \Delta s_r)/2d) \\ (\Delta s_r - \Delta s_l)/2 \end{bmatrix} \tag{3.7}$$

In this situation, the global pose of the mobile robot can be found by computing equation (3.7) during each interval of sensor measurement.

3.1.3 Color Blob Tracking

The second step of the present work concerns accurately tracking or detecting the objects of interest (boxes) by using a color blob vision tracking system. In Chapter 2, a fast color-blob tracking algorithm has been developed for the present work, which can effectively detect and track different color blobs marked on each vertical surface of a box. The detailed process of color blob tracking in this thesis is shown in Figure 3.3(a). The original image as acquired by a CCD camera is first processed to remove the disturbances by transferring the image from the RGB (red, blue, and green) color space to the HIS (hue, saturation, and intensity) color space and removing its saturation and intensity components.



(a) (b)

Figure 3.3: (a) Color blob tracking procedure; (b) Camera view.

In the next step, a type of statistical feature filtering is employed to remove the color that is not related to the sample color blobs (5×5 pixel templates). For a 2-D image with

$i \times j$ pixels, the average hue value and the standard deviation σ of the corresponding color blob can be calculated by the following equations:

$$\bar{h} = \sum_{i=1}^{n}\sum_{j=1}^{n}(h(i,j))/n^2 \tag{3.8}$$

$$\sigma = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(h(i,j)-\bar{h})^2/(n^2-1)} \tag{3.9}$$

Here, $h(i,j)$ represents the hue value of the original pixel (i,j). By executing an "If…else" logic as shown below, the statistical feature filtering is completed.

For (each pixel in the original image)

If (its hue value is within the set of

$\{h \mid \bar{h}_1 - 1.2\sigma_1 \le h \le \bar{h}_1 + 1.2\sigma_1$ or

$\bar{h}_2 - 1.2\sigma_2 \le h \le \bar{h}_2 + 1.2\sigma_2$ or……

$\bar{h}_i - 1.2\sigma_i \le h \le \bar{h}_i + 1.2\sigma_i$ or ……

$\bar{h}_k - 1.2\sigma_k \le h \le \bar{h}_k + 1.2\sigma_k\}$

Then (it will not be changed)

Else (Its hue values will be replaced with 0)

Loop

After statistical feature filtering, the color-blob templates, which are 5×5 pixel matrices, are applied to search the entire image, using the algorithm given below:

Initialize min-distance=100000, blob_pos=(1,1);

For (each element $H(i,j)$ in the matrix of H)

distance= $\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(H(i+k-1, j+l-1)-T(k,l))^2/m^2}$ (3.10)

if (distance<min-distance) then

min-distance=distance;

update: blob_pos=(i,j)

output blob_pose;

Finally, the robot rotates its base to adjust the view of the image in order to make sure that the detected color blob is approximately located on the center line of the camera

view. By doing so, it guarantees that the robot is heading towards the color label and the box in the range of the laser scanner.

In the present work, 4 different color circle labels (red, green, blue, yellow) are placed on the four vertical surfaces of the box. By using different color labels on different surfaces, the specific surface toward which the robot is heading can be determined as well, at the same time.

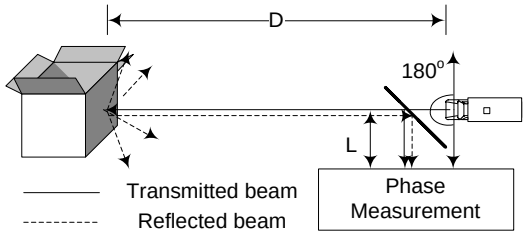
3.1.4 Estimation of Box Pose

Estimation of Relative Pose

The final step of the present work involves locating the box in the workspace, with respect to the global coordinate system. In order to achieve this objective, we first need to know the pose of the box with respect to the coordinate system attached to the mobile robot (local coordinate system). A laser range finder is used in this work for estimation of the relative pose. It is based on a SICK LMS 200 2D scanner, which has a horizontal range of 180^o with a maximum resolution of 0.5^o. This device produces a range estimation based on the time needed for the light to reach the target and return. Figure 3.4 (a) shows how it is used to measure range. The sensor transmits light at a known frequency and measures the phase shift between the transmitted and reflected signals. Then, the distance between the emission source and the target surface can be determined using:

$$D = \lambda\theta / 4\pi \qquad (3.11)$$

where θ is the electronically measured phase difference between the transmitted and reflected light beams.



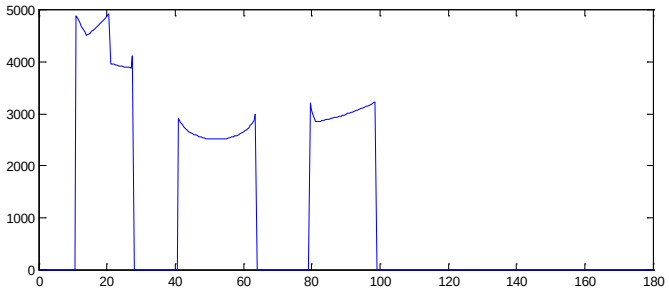
(a)

(b)

Figure 3.4: (a) Schematic drawing of laser range sensor; (b) a 180 degree laser range sensor.

Figure 3.5 shows the results of the laser range finder. There are four objects within the laser scanner according to these results. Because the box with a color blob label should be in the center of the laser range scan after applying the color blob tracking in the previous section, it is straightforward to establish that the box must be in the range between 79.5° and 99°.

(a)



(b)

Figure 3.5: (a) Visualized laser range finder results; (b) laser range finder results.

Figure 3.6 and Table 3.1 present the data that are acquired from the laser range finder for calculating the center location and orientation of the box with respect to the robot coordinate system. Based on this data, we can present the following group of equations, which represents the relationship among A, B, o' and o in the robot coordinate system.

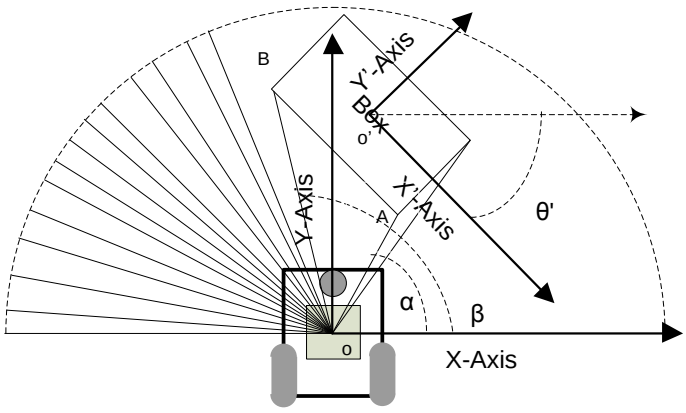


Figure 3.6: Laser range finder representation.

Table 3.1: Important laser range results.

Box Length	Box Width	α	β	OA (d1)	OB (d2)
1000mm	500mm	80.5°	99°	3079mm	3226mm

$$\begin{cases} x_A = d_1 \cos\alpha \\ y_A = d_1 \sin\alpha \\ x_B = d_2 \cos\beta \\ y_B = d_2 \sin\beta \\ (x_o^{'} - x_A)^2 + (y_o^{'} - y_A)^2 = 312500 \\ (x_o^{'} - x_B)^2 + (y_o^{'} - y_B)^2 = 312500 \end{cases} \tag{3.12}$$

$$\theta' = A\tan((y_B - y_A)/(x_B - x_A)) \tag{3.13}$$

By solving the equations (3.12) and (3.13), the pose of the box center (o') and the orientation of the coordinate frame that is attached to the box with respect to the robot coordinate system can be found and represented by the vector

$$o' = [o_x \quad o_y \quad \theta']^T \tag{3.14}$$

It describes the pose of the box with respect to the mobile robot, and is called the relative pose.

Estimation of Global Pose of Box

The homogeneous transformation (Spong et al., 2006) matrix, which represents a rigid body motion of translation and rotation, is applied in this work to represent the relationship among the three coordinate systems: the global coordinate system, the robot coordinate system, and the box coordinate system. It combines rotation and translation in a two-dimensional space, which can be used to perform coordinate transformations between frames that differ in orientation and translation.

By using the result obtained in the previous section, the homogeneous transformation matrix between the box coordinate system and the robot coordinate system can be written as

$$T' = \begin{bmatrix} \cos\theta' & -\sin\theta' & o_x \\ \sin\theta' & \cos\theta' & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

The pose of the robot coordinate frame expressed in the global coordinate system is known from equation (3.7). The homogeneous transformation matrix between the robot coordinate system and the global coordinate system can be found as

$$T'' = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

Therefore, the homogeneous transformation matrix T between the box coordinate system and the global coordinate system can be computed by using (3.15) and (3.16); specifically,

$$T = T''T' = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta' & -\sin\theta' & o_x \\ \sin\theta' & \cos\theta' & o_y \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos(\theta+\theta') & -\sin(\theta+\theta') & o_x\cos\theta - o_y\sin\theta + x \\ \sin(\theta+\theta') & \cos(\theta-\theta') & o_x\sin\theta + o_y\cos\theta + y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

In matrix (3.17), the origin and the orientation of the box with respect to the global coordinate system can be determined by

$$o'' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} o_x\cos\theta - o_y\sin\theta + x \\ o_x\sin\theta + o_y\cos\theta + y \\ A\tan(\sin(\theta+\theta')/\cos(\theta+\theta')) \end{bmatrix} \tag{3.18}$$

3.1.5 Simulation Environment

In this work, Microsoft Robotics Studio simulation environment is utilized to validate the developed method. The software and relevant algorithm are developed using Microsoft C#.

Figure 3.7 shows the simulation environment of Microsoft Robotics Studio. The Microsoft Robotics Studio is a Windows-based environment for robot control and simulation. Its features include: a visual programming tool, Microsoft Visual Programming Language for creating and debugging robot applications, web-based and windows-based interfaces, 3-D simulation (including hardware acceleration), a

lightweight service-oriented runtime facility, easy access to robot's sensors and actuators via a .NET-based concurrent library implementation, and support for a number of languages including C# and Visual Basic .NET, JScript, and IronPython.

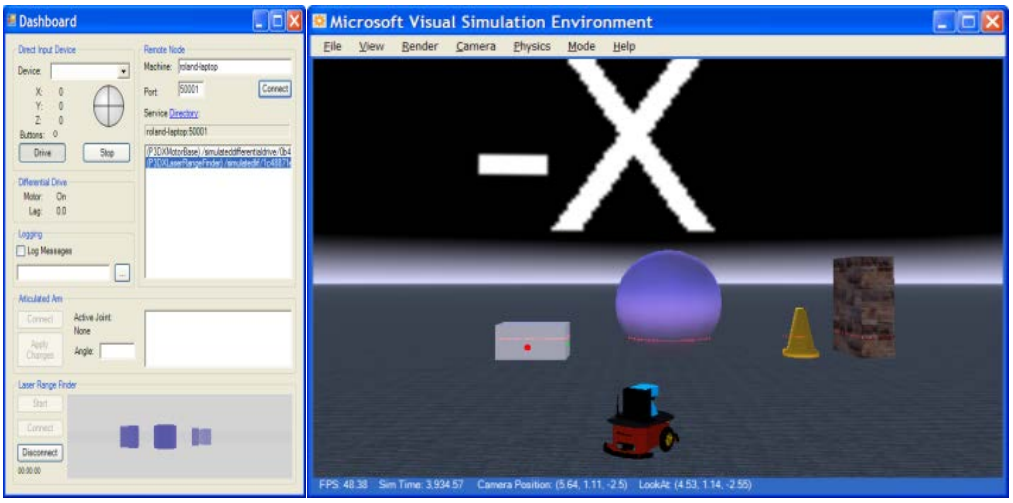


Figure 3.7: Simulation environment GUI.

3.1.6 Simulation Results

Figure 3.8 shows the experimental setup in this work. It contains a Pioneer AT mobile robot which carries a laser range finder and a CCD camera, and a gray object (a box that is 500 mm in width and 1000 mm in length) which is labeled using color blobs on its vertical surfaces. The objective of this experiment is to determine the box pose in the global coordinate system.

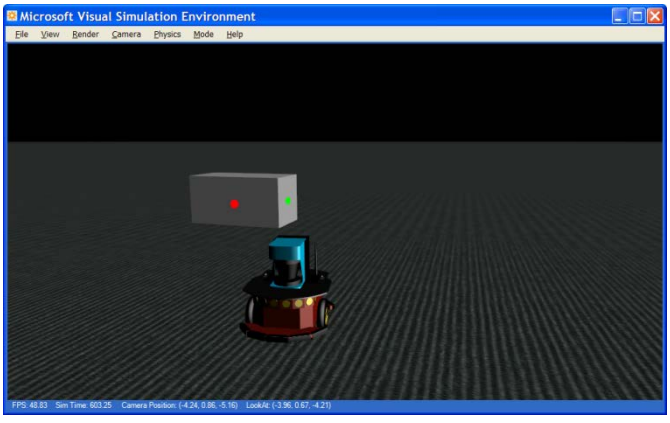
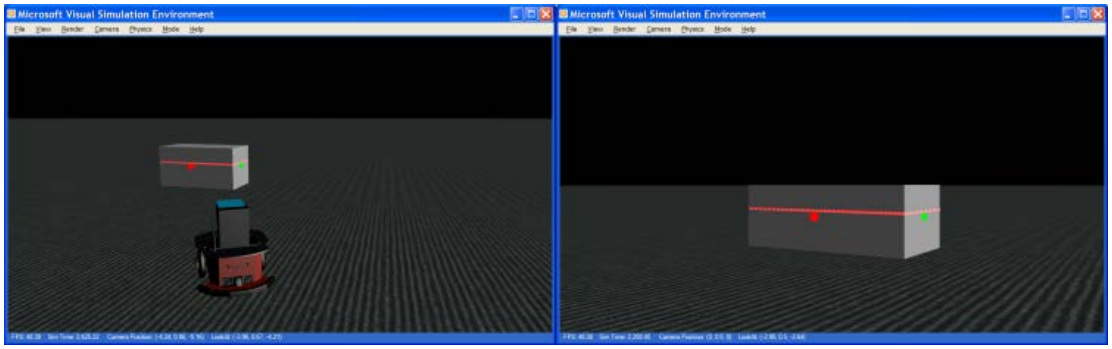


Figure 3.8: Experimental setup in the simulator.

First, the mobile robot rotates its base clockwise to search for the color blobs (i.e., the box) in the environment. Once the robot finds the color blobs, the color-blob tracking

algorithm is applied to identify candidates. Then, it keeps adjusting the base and moves the color blob into the middle of the vision frame in order to make sure that the box is within the range of the laser scanner. Meanwhile, the pose of the robot is recorded according to the odometry algorithm. Figure 3.9 shows the global camera view and the on-robot local camera view. The current robot pose in the global coordinate system in this experiment is $p' = \left[1000mm \quad 1500mm \quad -18°\right]^T$.



(a) (b)

Figure 3.9: (a) Global camera view; (b) Robot camera view.

The robot sits still while observing the box and the laser range finder is activated. Figure 3.10 shows the results acquired from the range scanner. The data for the pose calculation is given in Table 3.2.

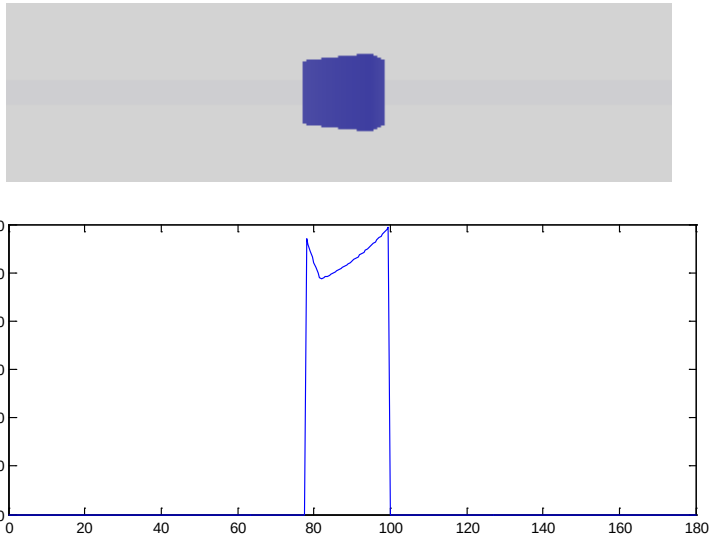


Figure 3.10: Laser range finder results.

Table 3.2: Data for pose calculation.

Box Length	Box Width	α	β	OA (d1)	OB (d2)
1000 mm	500 mm	82°	99.5°	2435 mm	2968 mm

By using the group of equations (3.12) and equation (3.13), the pose of the coordinate frame attached at the center of the box with respect to the robot coordinate system can be determined as: $o' = [2901\,\text{mm} \quad 69\,\text{mm} \quad -31.9°]^T$. The homogeneous transformation matrix between the box coordinate frame and the robot coordinate frame can be written as

$$T' = \begin{bmatrix} 0.8490 & 0.5284 & 2901 \\ -0.5289 & 0.8490 & 69 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

The homogeneous transformation matrix between the robot coordinate frame and the global coordinate frame can be generated as

$$T'' = \begin{bmatrix} 0.9511 & 0.3090 & 1000 \\ -0.3090 & 0.9511 & 1500 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

Therefore, the homogeneous transformation matrix between the box coordinate frame and the global coordinate frame is determined by using (3.19) and (3.20); as

$$T = T''T' = \begin{bmatrix} 0.6441 & 0.7649 & 3780 \\ -0.7653 & 0.6442 & 669 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

Finally, the origin and the orientation of the box in the global coordinate system can be determined according to (3.18) as

$$o'' = [x \quad y \quad \theta]^T = \left[3780\,\text{mm} \quad 669\,\text{mm} \quad 10.04°\right]^T \tag{3.22}$$

Figure shows the visualized experimental results, represented in the global coordinate system.

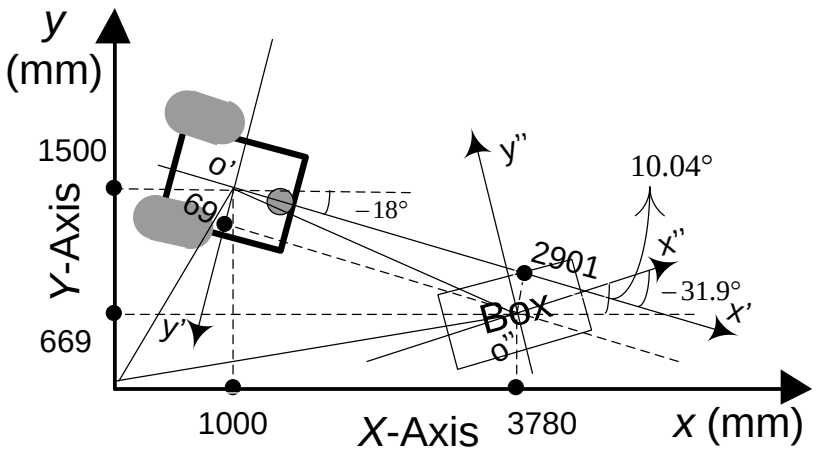


Figure 3.11: Visualized experimental result.

3.2 Autonomous Mobile Navigation

3.2.1 The Q-learning Algorithm

Reinforcement learning has been studied by psychologists since the 1940's (Sutton and Barto, 1998). It involves learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. It is neither supervised learning nor unsupervised learning. The learner is a decision-making agent who takes actions in an environment and receives rewards for its actions in trying to solve a problem (Alpaydm, 2004). After a set of trial-and error runs, it should learn the best policy, which is the sequence of actions that maximizes the total reward (Figure 3.12).

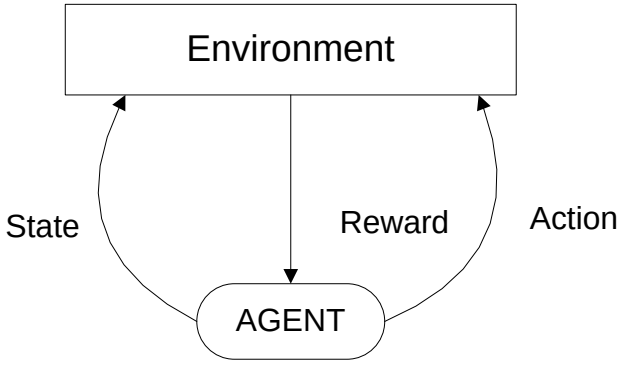


Figure 3.12: The agent interacts with an environment.

As shown in Figure 3.12, an intelligent agent knows its state in the current environment. By applying an action, which is chosen from a sequence of actions in this state, a feedback to this action is acquired from the environment, which will indicate whether the action is good (positive reward) or not (negative reward). After exploring actions in this sequence under different states, a table can be generated to represent the action probability of getting positive reward in certain situation. The greater the reward of an action in a particular state, the higher the probability of selecting that action.

Machine learning using the Q-learning algorithm incorporates Markov Decision Processes (MDP) (Russell and Norvig, 2003, Alpaydin, 2004), which can be defined by a 4-tuple $<S, A, T, R, \beta>$, where

$S = \{s_1, s_2, \ldots, s_n\}$, is a set of states of the work environment or the world

$A = \{a_1, a_2, \ldots, a_m\}$, is a set of actions available to the robot

$T : S \times A \to \prod(S)$, is a transition function, which decides the next environmental state s' when the robot selects an action a_i under the current state s. Also, $\prod(s)$ is the probability distribution over the states.

$R : S \times A \to \Re$, is a reward function, which determines the immediate reward when the robot takes an action a_i under the current state s.

In MDP, the core issue is to find the optimal action-selection policies so that the cumulative reward in a sequence of decision-making becomes a maximum. Since the transition function and the reward function are usually unknown in a real engineering application, the Q-learning algorithm (Sutton and Barto, 1998), which involves reinforcement learning, is employed to reach the optimal policies in an MDP problem. The steps of the Q-learning algorithm employed in the present work are given below:

For each state $s_i \in (s_1, s_2, \cdots, s_n)$ and action $a_j \in (a_1, a_2, \cdots, a_m)$, initialize the table entry $Q(s_i, a_j)$ to zero. Initialize τ to 0.9. Initialize the discount factor $0 < \beta \leq 1$ and the learning rate $0 < \eta \leq 1$.

Observe the current state *s*

Do repeatedly the following:

Probabilistically select an action a_k with probability

$P(a_k) = \frac{e^{Q(s,a_k)/\tau}}{\sum_{l=1}^{m} e^{Q(s,a_l)/\tau}}$, and execute it

Receive the immediate reward r

Observe the new state s'

Update the table entry for $Q(s,a_k)$ as follows:

$$Q(s,a_k) = (1-\eta)Q(s,a_k) + \eta(r + \beta \max_{a'} Q[s',a'])$$

$s \leftarrow s'$, $\tau \leftarrow \tau * 0.9$

It has been proved that the values of $Q(s_i,a_j)$ will converge in a static environment after a certain number of iterations of the Q-learning algorithm (Sutton and Barto, 1998). When the Q values converge, it implies that the agent or the robot has learned the correct action-selection policy; specifically, it always selects the optimal action for a specific world state so that the sum of the discounted rewards in the subsequent decision-making process will reach a maximum.

3.2.2 Problem Definition

The objective of the present mobile navigation project is to develop an autonomous mobile robot system that has the ability to autonomously navigate from an initial location to a goal location. As shown in Figure 3.13, in this workspace, a mobile robot attempts to navigate from its initial location to the goal location where a screw is located. In its workspace, obstacles may be present and even appear randomly during the navigation process.

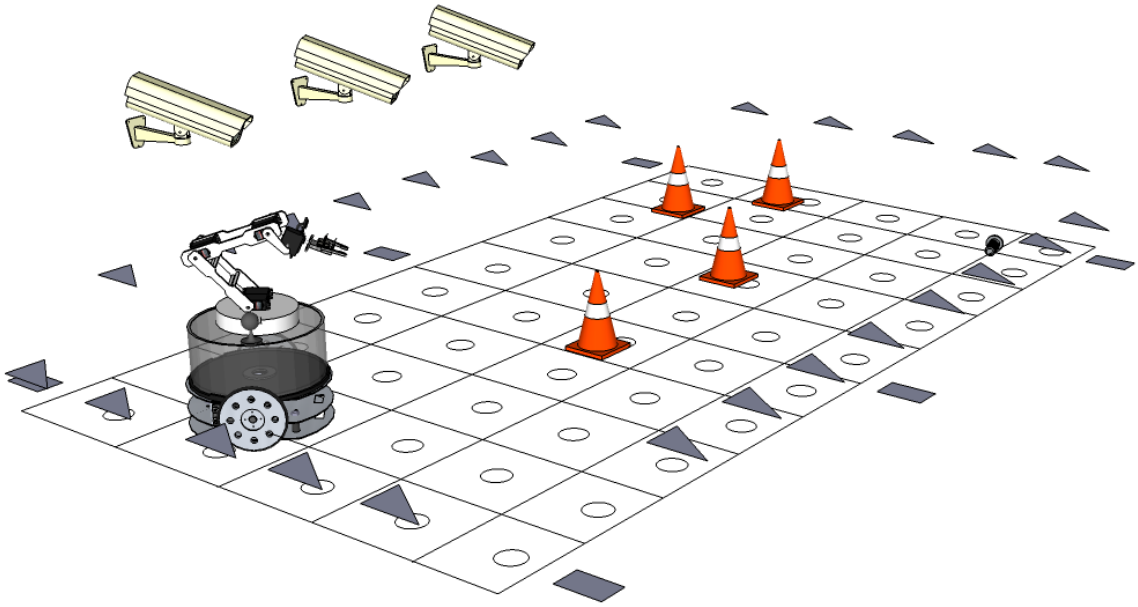
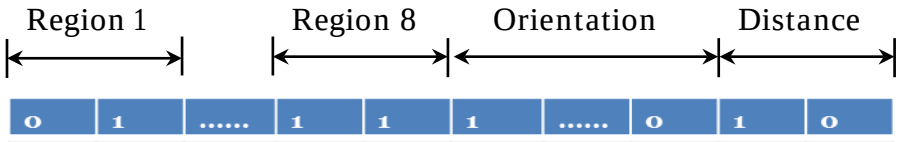


Figure 3.13: The mobile navigation system.

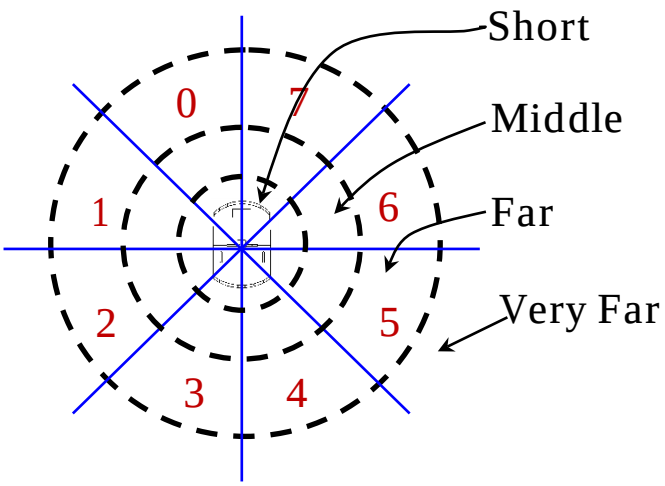
3.2.3 States, Actions and Rewards

Figure 3.14 presents the definition of the states of the robot in the environment. Suppose that the mobile robot is in the center of the circle in the right of Figure 3.14(b). It divides the surrounding area of the robot into eight regions as shown in Figure 3.14(b). For each region, a two bit binary data is utilized to represent the distance between the closest obstacle and the robot in the particular region. The binary data are lined up in a sequence ranging from region 1 to region 8 as shown in Figure 3.14 (a). The two digits of each group represent the distance between the obstacle and the robot. Therefore, a sequential 14-bit binary data represents the current state of the robot working environment which comes to 32 (8×4) possible states.



00: Short (0, 30] cm
01: Middle (30, 100] cm
10: Far (100, 200] cm
11: Very Far (200, ∞] cm

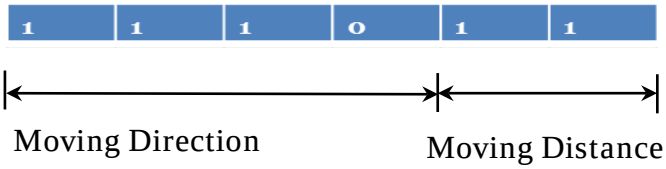
(a)



(b)

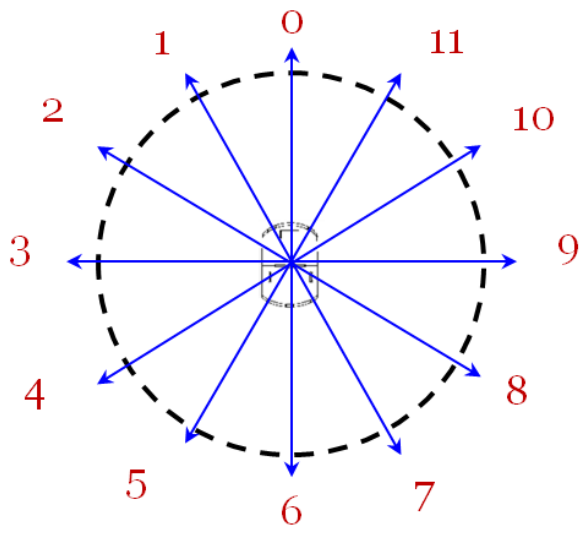
Figure 3.14: Definition of states of the mobile robot and environment.

Figure 3.15 presents the definition of possible actions of the robot. Based on the representation in Figure 3.15 (b), it divides the heading direction into 12 sectors and each sector covers 30 degrees. This means the robot can navigate in twelve directions as denoted by 0 to 11. There are six bits which are used to represent the motion strategy of the robot, as shown in Figure 3.15(a). The first four bits represent the moving direction and the last two bits give the distance of the motion. The combination of these 6 bits provides the action of the robot motion in terms of orientation and distance.



00: Short	Rand(0— 10)
01: Middle	Rand(10—20 cm)
10: Long	Rand(20—40 cm)
11: Extra long	Rand(40—60 cm)

(a)



(b)

Figure 3.15: Definition of Actions.

It follows that the robot will have 2^{21} states and 2^{6} actions in total, which comes to 2,097,152 states and 64 actions. In the Q-table, there will be one on one mapping between states and actions. It results in 134,217,728 entries in the table, which represent the mapping among states and actions. If each Q-value is a single floating-point data which will use 16 bits (2 bytes) of memory, the overall Q-table will use approximately 0.3 GB of memory space, which can be accommodated by a modern computer.

As mentioned before, the Q-Learning involves decision making of robotic motion based on the environmental states of the robot. Before the Q-table can be utilized in a robotic navigation task, it has to be trained. The training procedure is shown in Figure 3.16.

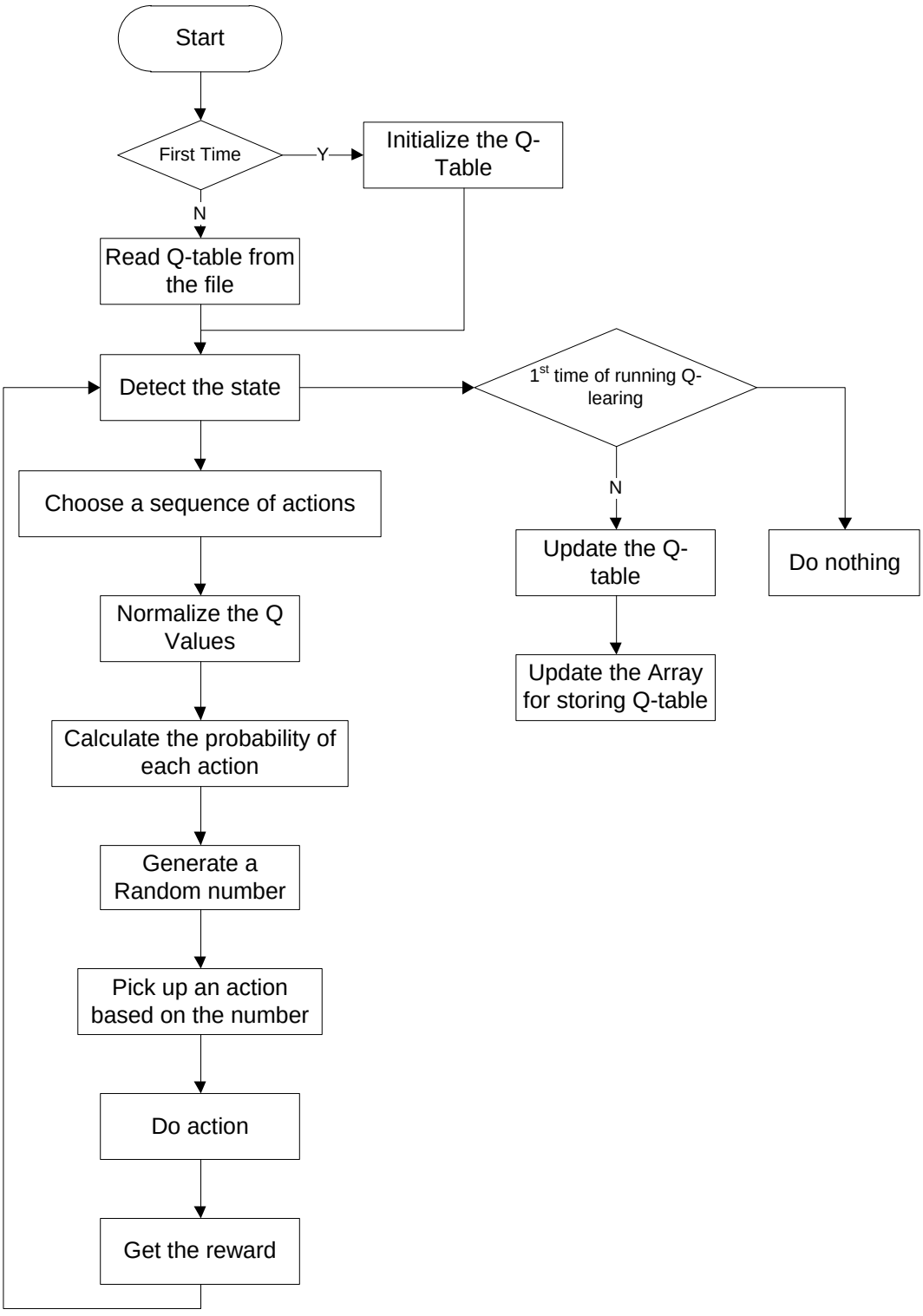


Figure 3.16: Flowchart of Q-Learning.

It will load the Q-table and initialize it if it, in the beginning of training. In each iteration, the robot will detect the state of the environment by using its local sensors

based on the state definition discussed above. Next, action selection is done based on the current state. The robot must select a good action from the action database in order to perform an efficient movement, one that can approach the goal while avoiding obstacles. The action is selected by applying the probabilistic function:

$$p_i = e^{\frac{Qnew_i}{\tau}} / \sum e^{\frac{Qnew_i}{\tau}} \quad (3.23)$$

where each action can get a probability to be selected in the range from 0 to 1 (Figure 3.17). The advantage of using the $e^{\frac{Qnew_i}{\tau}}$ term is to avoid the 0 probability, so that each action has a non-zero to be chosen. Also it solves the local maximum problem.

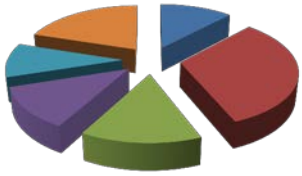


Figure 3.17: Probability of each action.

After the action is executed, the robot will re-evaluate the current state and compare it with the state before applying the action. Then it will generate a quantitative value, which is called the Q value, to represent the performance of the previous action by a cost function; specifically,

$$Q(s,a) = \mathrm{Q}(s,a) + \alpha[r + \gamma Max\big(Q(s',a')\big) - Q(s,a)] \quad (3.24)$$

where $Q(s,a)$ is the current value of a state-action pair, $Q(s',a')$ is the value of the state-action pair of the next step, α is the learning rate (0.8 in my case), γ is the discount factor (0.9 in our case)., and r is immediate reward. The new Q value will be utilized to update the existing Q value in the Q table. The Q values in the table will converge if sufficient time is provided for training.

3.2.4 Simulation Platform

Since the training procedure is complex and time-consuming, it is difficult to directly implement it in a physical system. A simulation platform is required to train the system. A simulation platform is developed in Visual Basic.net for acquiring the Q table (knowledge base) in mobile navigation system. The User Graphic Interface (GUI) of the

developed system is shown in Figure 3.18. The black dot represents the mobile robot; the red dots are obstacles; and the blue dot represents the goal location of the mobile robot. The goal of this simulation system is to train the mobile robot to navigate from the current location to the goal region without colliding with any obstacles.

Figure 3.18: Developed simulation platform for Q-learning training in mobile navigation.

As described in the previous session, the original Q-table is initialized so that all the actions have the same probability to be selected. This means the motion of the robot is purely random in the beginning of a training process. As the training process proceeds, the mobile robot will become smarter and choose the action more wisely according to the updated *Q* values which represent the efficiency of the motion at the particular state.

There are three stopping criteria for each training iteration: 1). Robot hits an obstacle (Figure 3.19); 2). Robot reaches the goal (Figure 3.20); 3). Robot reaches the allowed maximum numbers of moving steps (Figure 3.21).

Figure 3.19: Mobile robot collides with an obstacle.

Figure 3.20: Mobile robot reaches the goal.

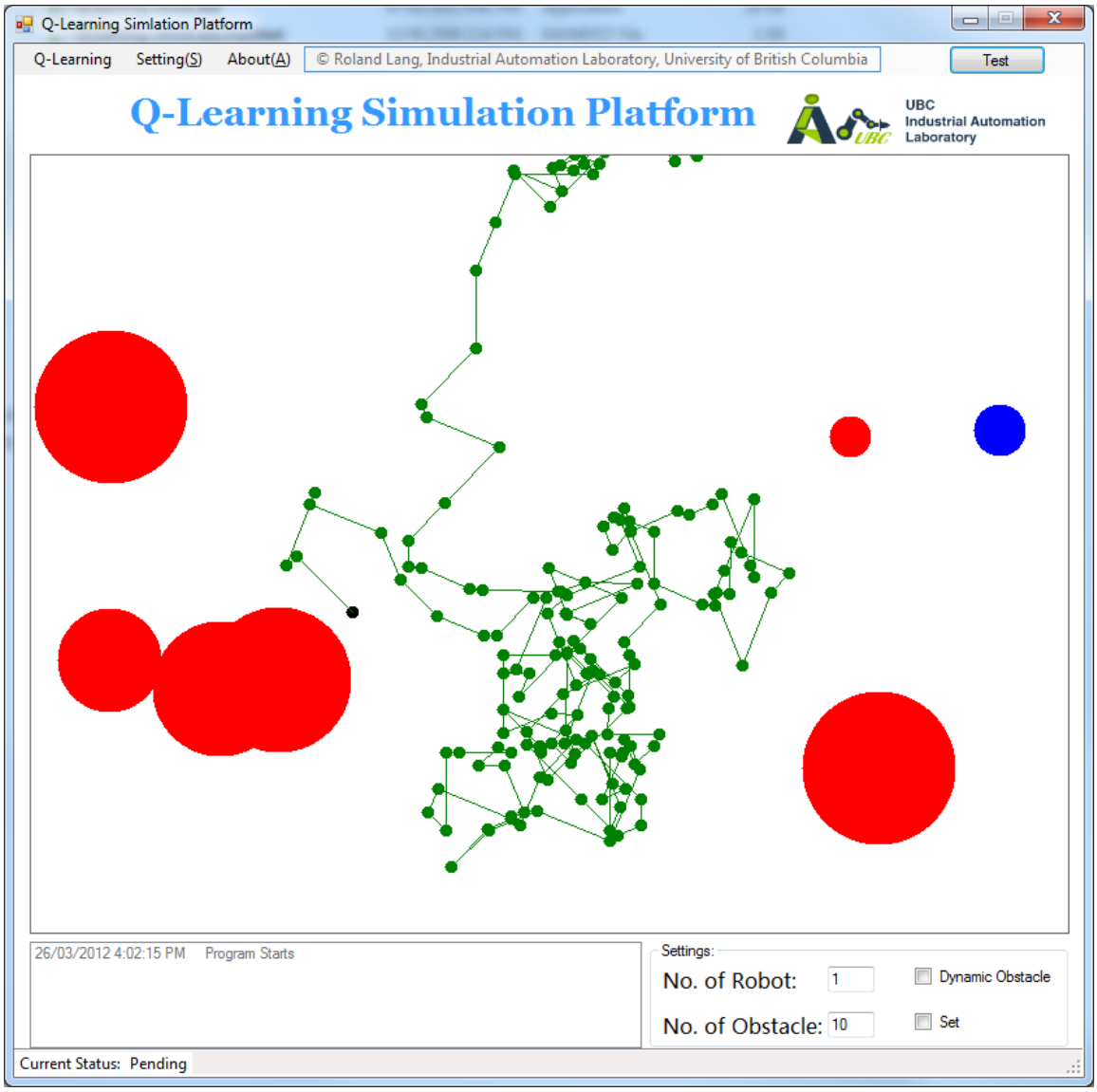
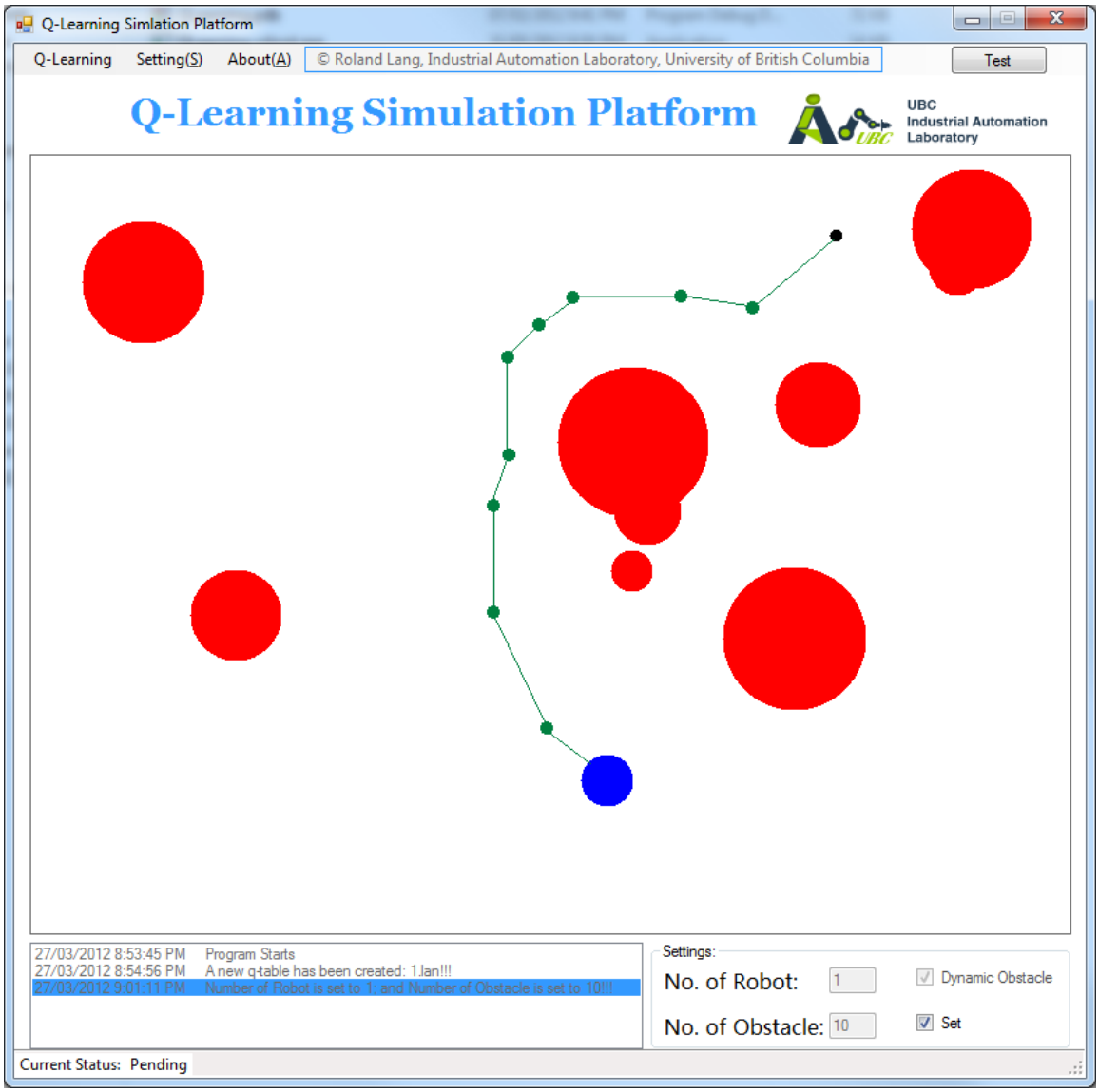


Figure 3.21: System exceeds the allowed maximum number of motion steps.

After training, a validating program in the develop system can be utilized to verify the training performance of the system. In the validating stage, the position of the robot, the goal region, the number of obstacles and their sizes are randomly generated in the workspace. In each motion step of the robot, it checks its states according to its local sensing capability. Then, it selects an action based on the probabilistic action selection function (3.32). Having selected an action in the Q-table, it executes the selected action and moves the mobile robot to the next states. It repeats this procedure until the robot reaches the goal region.


Q-Learning Simlation Platform
Q-Learning
Setting(S)
About(A)
© Roland Lang, Industrial Automation Laboratory, University of British Columbia
Test
New Q Table(N)
Load Q Table(L)
Training(T)
Validate(V)
Exit(E)
arning Simulation Platform
UBC
Industrial Automation
Laboratory
27/03/2012 8:53:45 PM Program Starts
27/03/2012 8:54:56 PM A new qtable has been created: 1Jan!!!
Settings:
No. of Robot: 1
Dynamic Obstacle
No. of Obstacle: 10
Set
Current Status: Pending

(a)



(b)

Figure 3.22: Validation of the training results.

CHAPTER 4 Visual Servo Control

A vision-based mobile manipulation system typically uses computer vision data as sensory feedback for controlling the motion of robots. This is also called a visual servoing system or a visual servo control system. It involves a fusion of many related research areas including machine vision, robot modeling, control theory, and real time computing. Specifically, in visual servo control, the vision system provides feedback information about the current state of the environment to the controller (Chaumette and Hutchinson, 2006). The robotic system usually consists of a mobile base on which a multiple-degree-of-freedom (multi-DOF) manipulator arm is mounted. This robot carries a group of sensors such as cameras, laser distance finders, bumper collision detectors, and sonars. If it has the ability to work in an unknown and unstructured environment on its own, without the external interaction of a human operator or another external system, it is known as autonomous manipulator.

A new trend is to integrate visual servoing into a mobile robot for carrying out grasping or manipulation activities autonomously, resulting in a vision-based autonomous mobile manipulation system (Spong, et al., 2006). Compared to traditional visual-servo control employed in fixed-base robotic manipulators, a vision-based mobile manipulation system has many advantages. The most important among them may be that a vision-based mobile manipulation system that integrates the capabilities of its vision subsystem, the on-board arm and its mobile base, can move about and work in an unstructured environment. Consequently, the robotic system becomes more flexible and has wider applications than a traditional fixed-base manipulator system. Since its arm (manipulator) and cameras are usually mounted on a mobile base, a mobile manipulation system possesses better maneuverability and terrain coverage capability than a fixed-base manipulator.

A multi-robot cooperative assembly system is being developed in the Industrial Automation Laboratory at The University of British Columbia. In this system, multiple mobile robots autonomously search for the parts of a target vehicle, which are scattered in an unstructured environment of complex terrain and obstacle distribution. Once an object of interest is encountered and identified, a robot will attempt to autonomously grasp it and pick it up with its on-board arm using visual servoing, and transport it to a designated site for further manipulation and assembly. When the robots determine that they have collected all the necessary parts, they proceed to assemble them to construct the target vehicle, by cooperatively manipulating the parts. The specific system is developed for application in robotic search and rescue.

This chapter addresses several challenges in the area of mobile manipulation with visual feedback control such as visibility constraint, physical constraints and optimal controller outputs. First, a hybrid controller, which combines a traditional proportional-integral-derivative (PID) controller and an intelligent Q-learning controller, is proposed for visual servo control. It mainly accommodates the visibility constraint. Then a more advanced controller, termed Adaptive Nonlinear Model Predictive Controller (ANMPC), is proposed and developed for both mobile navigation and robotic manipulation. This approach is able to solve problems of visibility constraint and physical constraints, and also provide optimal controller outputs.

4.1 Modeling

Models of robots and cameras are commonly used in schemes of visual servo control. In this section several relevant model formulations are presented.

4.1.1 Rigid Motions and Homogeneous Transformation

The rigid motions of a robot can be represented by the matrix

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, R \in SO(3), d \in \mathbb{R}^3 \qquad (4.1)$$

where R represents a 3×3 rotational matrix; and d is a distance vector. The matrix in Equation (4.1) is called a homogeneous transformation, which represents the rotational and displacement relationship between two coordinate frames. It is a 4×4 matrix. The last row consists of three zeros and a one; and remaining elements are composed of a

rotation matrix and a position vector. In robotic applications a commonly used convention for defining the coordinate frames of reference and generating the homogeneous transformation matrices is the Denavit-Hartenberg (DH) convention, which was introduced by Denavit and Hartenberg (Spong, et al., 2006) to simplify the modeling procedure in generating forward kinematics (i.e., expressing the end-effector movement in terms of the joint movements) of a robot. There are four parameters in this representation: θ_i, d_i, a_i and α_i. The detailed definitions of these four parameters are given in Table 4.1.

Table 4.1: Denavit-Hartenberg convention.

Parameter	Axis	Description
θ_i	z_{i-1}	Joint Angle: the angle from x_{i-1} to x_i measured about z_{i-1}. - Variable for revolute joints.
d_i	z_{i-1}	Link Offset: distance along z_{i-1} from o_{i-1} to the intersection of the axes x_i and z_{i-1}. - Variable for prismatic joints.
a_i	x_i	Link Length: distance along x_i from the intersection of the axes x_i and z_{i-1} axes to o_i . - Constant perpendicular distance between z_{i-1} and z_{i-1}.
α_i	x_i	Link Twist: the angle from z_{i-1} to z_i measured about x_i. - Constant angle between z_{i-1} and z_i.

Suppose that the coordinates are assigned based on the two DH rules:

(DH1): The axis x_i is perpendicular to the axis z_{i-1} ($x_i \perp z_{i-1}$).

(DH2): The axis x_i intersects the axis z_{i-1} ($x_i \cap z_{i-1} \neq \emptyset$).

Then, there exists a unique homogeneous transformation matrix that takes the coordinates from one frame to the base frame, as given by:

$$H = Rot_{z,\theta}Trans_{z,d}Trans_{x,a}Rot_{x,\alpha} =$$

$$\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

where $Rot_{z,\theta}$ is the rotational matrix about the z axis, $Tran_{x,a}$ is the translational matrix along the z axis, $Tran_{x,a}$ is the translational matrix along the x axis, and $Rot_{x,a}$ is the

rotational matrix. Figure 4.1 shows an example of coordinate assignment according to the DH rules where $x_i \perp z_{i-1}$ and $x_i \cap z_{i-1} \neq 0$.

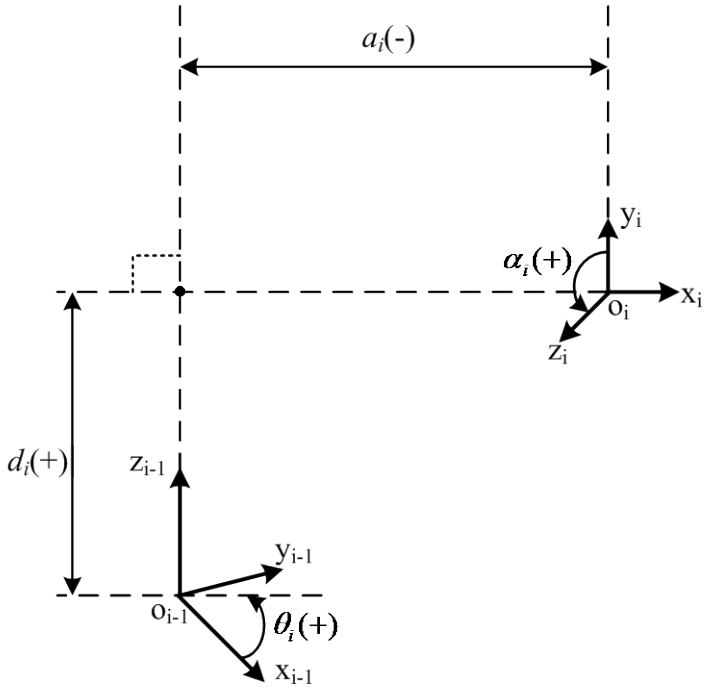


Figure 4.1: An example of coordinate frames satisfying DH convention.

4.1.2 Kinematic Modeling of the Robots

In order to mathematically modeling the robotics system, 6 coordinate frames (Figure 4.2) are introduced. They are camera coordinate frame, pixel coordinate frame, image coordinate frame, global coordinate frame, mobile base coordinate frame, and robot arm coordinate frame.

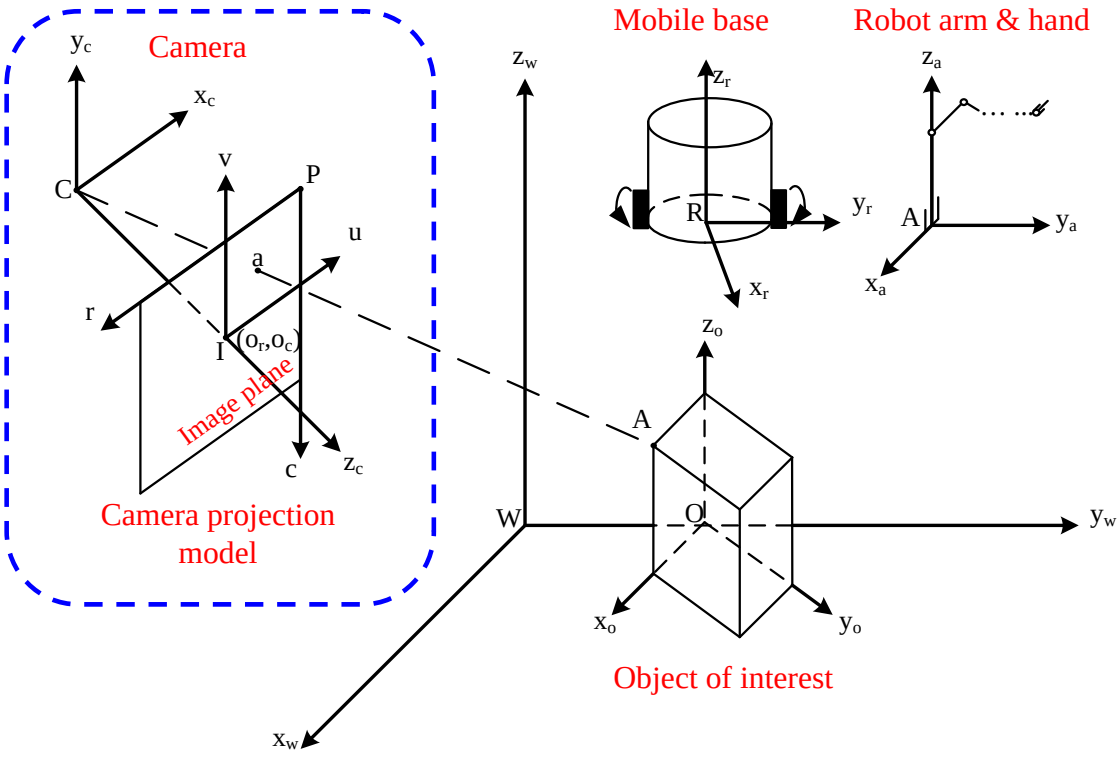


Figure 4.2: Definition of the system coordinates.

The purpose of forward kinematic modeling is to find the position and orientation of the robot end-effector in terms of the base frame by knowing the parameters of all joint variables. It is a critical problem in robot modeling. It is a usual case that the links of a robot are assumed to be rigid bodies; and they are connected together at joints. These joints determine the freedom of motion between the connected two joints. There are two typical types joints in general purpose industrial robots: 1) Revolute joints (or rotational joints) and 2) Prismatic joints (or translatory joints or telescopic joints or linear joints). Because of the rigid-body assumption, the forward kinematics problem can be solved by calculating the rigid body transformation among all robot links.

In order to derive the control law for a mobile robot with its manipulator, mathematical models that describe the relationship between the velocities of the robot joints and the velocity of a feature point in the camera scene are required.

Kinematic model of the mobile robot

Figure 4.3 shows the relationship between the mobile robot frame and the stereo camera frame, where the robot frame and the camera frame are rigidly attached to the

mobile robot and the stereo camera, respectively. The coordinate transformation between these two frames is given by

$$H_c^r = \begin{bmatrix} R_c^r & d_c^r \\ 0 & 1 \end{bmatrix} \tag{4.2}$$

where H_c^r represents the homogeneous transformation from the camera frame to the mobile robot frame, R_c^r is the rotational matrix, and $d_c^r = [d_x \quad d_y \quad d_z]^T$ is the position vector which represents the distance between the origin of the mobile robots and the origin of the camera frame with respect to the mobile robot frame.

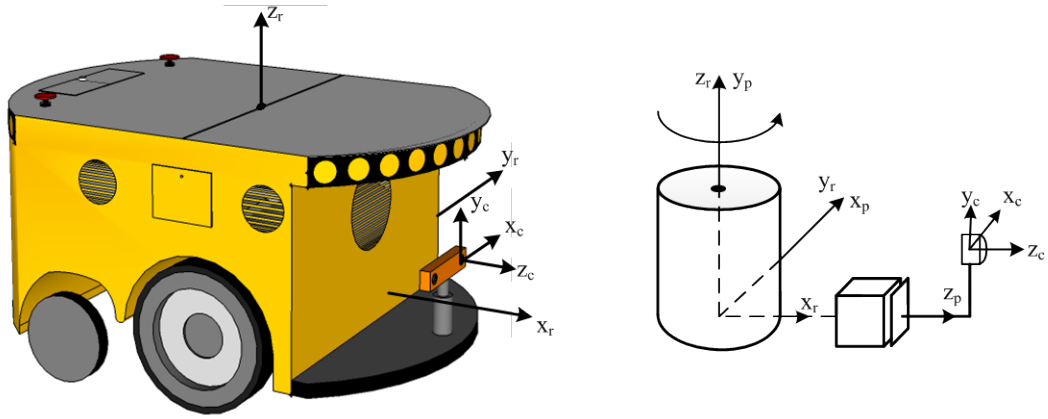


Figure 4.3: Abstraction of the mobile robot, stereo camera and their frames.

Kinematic model of the robotic manipulator

Figure 4.4 shows the kinematic chain of the manipulator. The Denavit-Hartenberg (DH) parameters of the RobuArm are given in Table 4.2. The homogeneous transformation of the RobuArm can be generated by using the DH convention, which represents the orientation of the tool frame and the distance of the origin of the tool frame from the origin of the robot base frame, with respect to the robot base frame.

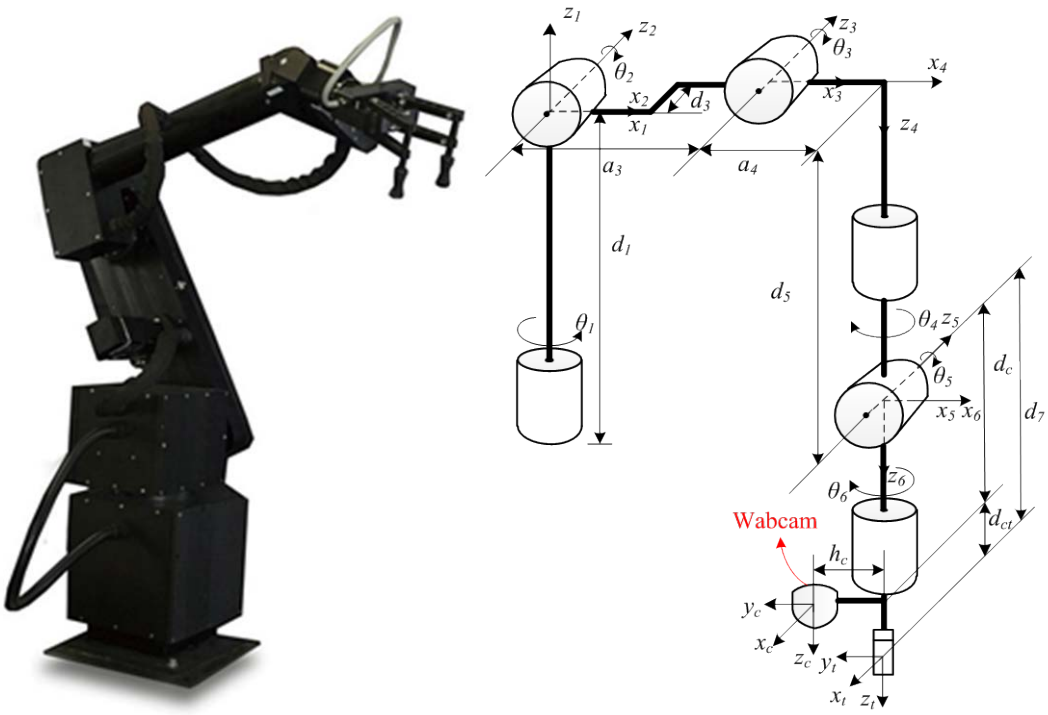


Figure 4.4: Kinematic chain representation of a robotic manipulator.

Table 4.2: DH table of the RobuArm.

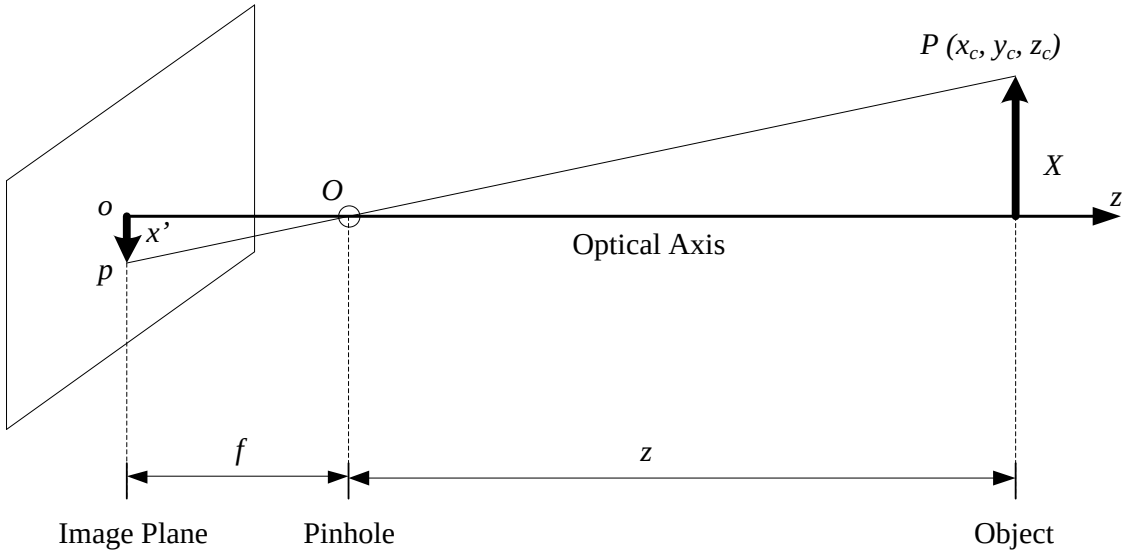
Link	α_i	a_i	d_i	θ_i
1	0	0	400	θ_1
2	$-\pi/2$	0	0	θ_2
3	0	418	12.8	θ_3
4	$-\pi/2$	115	0	θ_4
5	$\pi/2$	0	505	θ_5
6	$-\pi/2$	0	0	θ_6
End-effector	0	-68	301.45	0

4.1.3 Camera Modeling

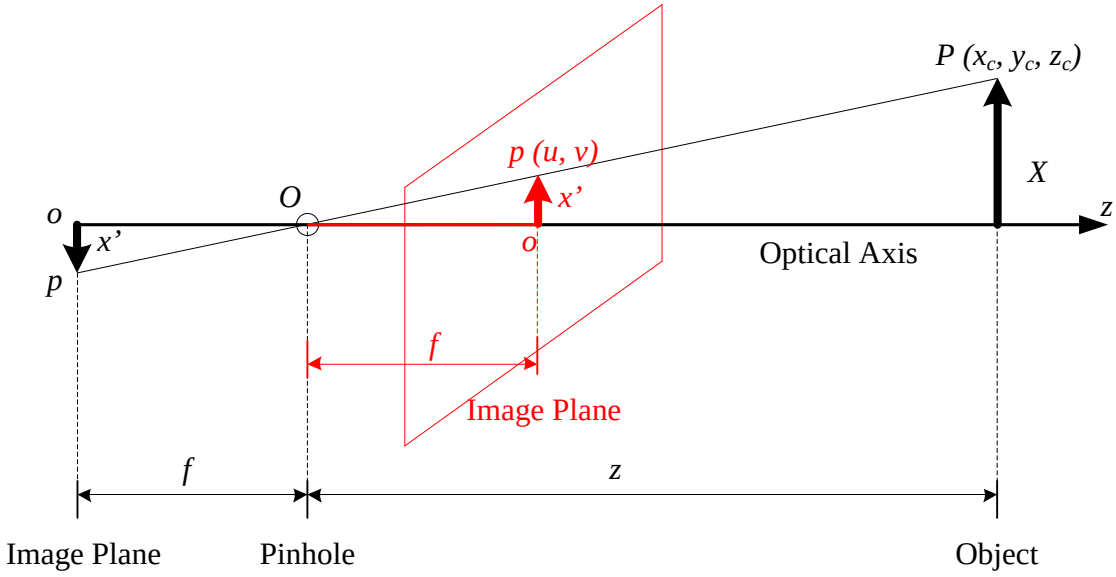
Perspective Projection

The simplest model of a camera is the pinhole camera model, which is shown in Figure 4.5(a). In this model, light from a point on the object (P) in the world spaces passes through a pinhole and projects onto the camera image surface called the image plane. For an ideal pinhole camera, the camera plane is located at a distance f (focal

length) behind the pinhole. The origin of the coordinate frame is called the center of projection in the image plane. The intersection of the z axis and the image plane is known as the principal point.



(a)



(b)

Figure 4.5: (a) Pinhole camera model; (b) Pinhole camera model with reflected image plane.

Figure 4.5(b) shows a pinhole camera model with the reflected image plane model. This model has the advantage that the projected object (x') in the image plane is reversed when compared with the object in the world space (X). Therefore, the modified pinhole

camera model simply shifts the image plane to the front of the principal point by $2f$. In this model, the image reversion problem is solved. Meanwhile, the size of the projected object does not change in view of the property of similar triangles. If camera focal length f is known, and the coordinates of the point object in terms of the camera frame is known as well, which is denoted by (x_c, y_c, z_c), we can determine the coordinates of the point in the real world in terms of the camera frame by sensing the position of the projected point in the image plane (u, v):

$$k\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} u \\ v \\ f \end{bmatrix} \tag{4.3}$$

where u,v are the coordinates of the projected point in the image plane. It is seen that these equations can help identify the 3D position of a point in the camera frame if u and v are measureable. However, it is not easy to measure these. The sensor of a digital camera is a two-dimensional array whose elements are called pixels, as shown in Figure 4.6. These pixels are not continuous, and there exist gaps among the pixels. Moreover, a pixel element is usually rectangular and not square, especially in inexpensive cameras. The dimensions of the pixel element are descried by s_x and s_y in Figure 4.6. Also, the origin of the image coordinate frame, which is the principal point, is usually not the center of the image/pixel plane. There exist offsets in real situations.

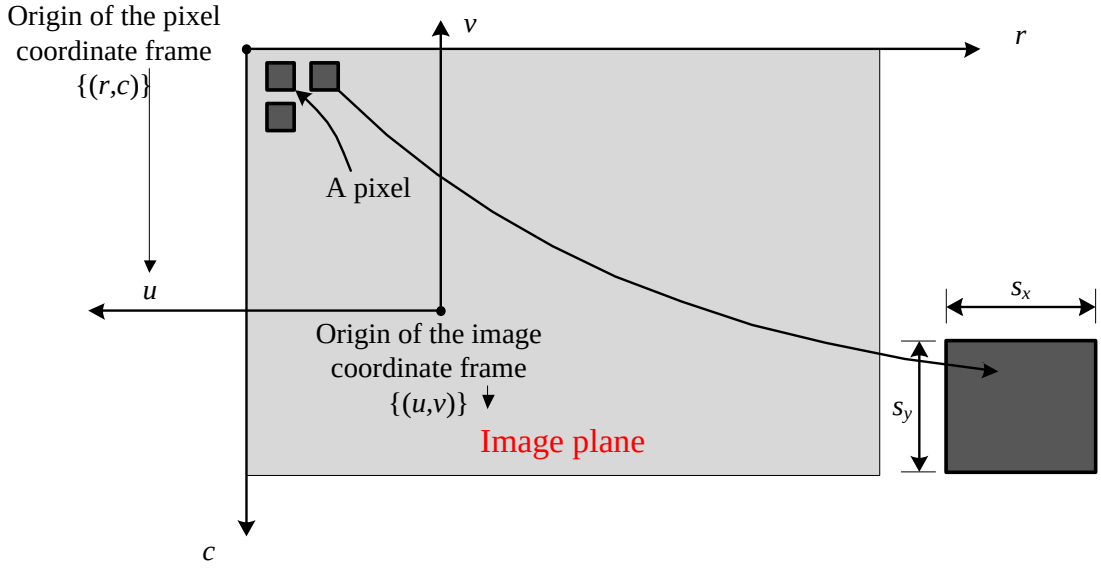


Figure 4.6: Image plane and pixel plane.

In Figure 4.5(b), P is a point in the work environment with coordinates $(x, y, z)^c$ relative to the camera frame, and p is the projection of P on the image plane with coordinates (u, v) relative to the image plane frame, and with coordinates (r, c) relative to the pixel coordinate frame. The distance between the origin of the camera frame and the image plane is denoted by λ, and the coordinates of the principal point are (o_r, o_c) with respect to the pixel coordinate frame. The coordinate transformation between the frames is given by:

$$\begin{cases} u = -s_x(r - o_r) \\ v = -s_y(c - o_c) \end{cases} \tag{4.4}$$

$$\begin{cases} r = -f_x \dfrac{x^c}{z^c} + o_r \\ c = -f_y \dfrac{y^c}{z^c} + o_c \end{cases} \tag{4.5}$$

Here, s_x and s_y are the horizontal and vertical dimensions, respectively, of a pixel as given by $f_x = \dfrac{\lambda}{s_x}$ and $f_y = \dfrac{\lambda}{s_y}$.

Let p be a feature point on the image plane with coordinates (u, v). The moving velocity of p can be expressed in terms of the camera velocity using the interaction matrix L as (Dean-Le´on, et al., 2006):

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = L\xi_c^c = \begin{bmatrix} -\dfrac{\lambda}{z^c} & 0 & \dfrac{u}{z^c} & \dfrac{uv}{\lambda} & -\dfrac{\lambda^2 + u^2}{\lambda} & v \\ 0 & -\dfrac{\lambda}{z^c} & \dfrac{v}{z^c} & \dfrac{\lambda^2 + v^2}{\lambda} & -\dfrac{uv}{\lambda} & -u \end{bmatrix} \xi_c^c \tag{4.6}$$

In Equation 4.6, there are several unknowns: the dimensions of the sensor (s_x and s_y), focal length of the camera (f_x and f_y), the location of the principal point (o_r and o_c), and the velocity of the camera with respect to the camera frame (ξ_c^c). These parameters can be determined through camera calibration, which involves finding two groups of parameters:

(1) Intrinsic camera parameters

(2) Extrinsic camera Parameters

Camera Parameters

Since cameras are usually rigidly attached to robots or mounted on tripods, the rotational matrix R_c^w and the translational vector T_c^w can be easily determined. Therefore, if we know the 3D position of a point P in the camera frame, the coordinates of this point in the world frame can be determined by using

$$P^w = R_c^w P^c + T_c^w \tag{4.7}$$

or conversely,

$$P^c = R_w^c(P^w - T_c^w) = R_w^c P^w - R_w^c T_c^w \tag{4.8}$$

where R_w^c and $R_w^c T_c^w$ are called the extrinsic camera parameters. In the section below, we denote them by R and T, respectively, for brevity. Consider the rotational matrix:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{4.9}$$

and the translational vector:

$$T = [T_x \quad T_Y \quad T_z]^T \tag{4.10}$$

with respect to the world coordinate frame. The coordinates of the point in terms of the camera frame can be found by

$$x^c = r_{11}x^w + r_{12}y^w + r_{13}z^w + T_x \tag{4.11}$$

$$y^c = r_{21}x^w + r_{22}y^w + r_{23}z^w + T_y \tag{4.12}$$

$$z^c = r_{31}x^w + r_{32}y^w + r_{33}z^w + T_z \tag{4.13}$$

Substituting (4.11)-(4.13) into (4.5), we obtain

$$r - o_r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x^w + r_{12}y^w + r_{13}z^w + T_x}{r_{31}x^w + r_{32}y^w + r_{33}z^w + T_z} \tag{4.14}$$

$$c - o_c = -f_y \frac{x^c}{z^c} = -f_y \frac{r_{21}x^w + r_{22}y^w + r_{23}z^w + T_y}{r_{31}x^w + r_{32}y^w + r_{33}z^w + T_z} \tag{4.15}$$

Combining (4.14) and (4.15), we obtain the equation

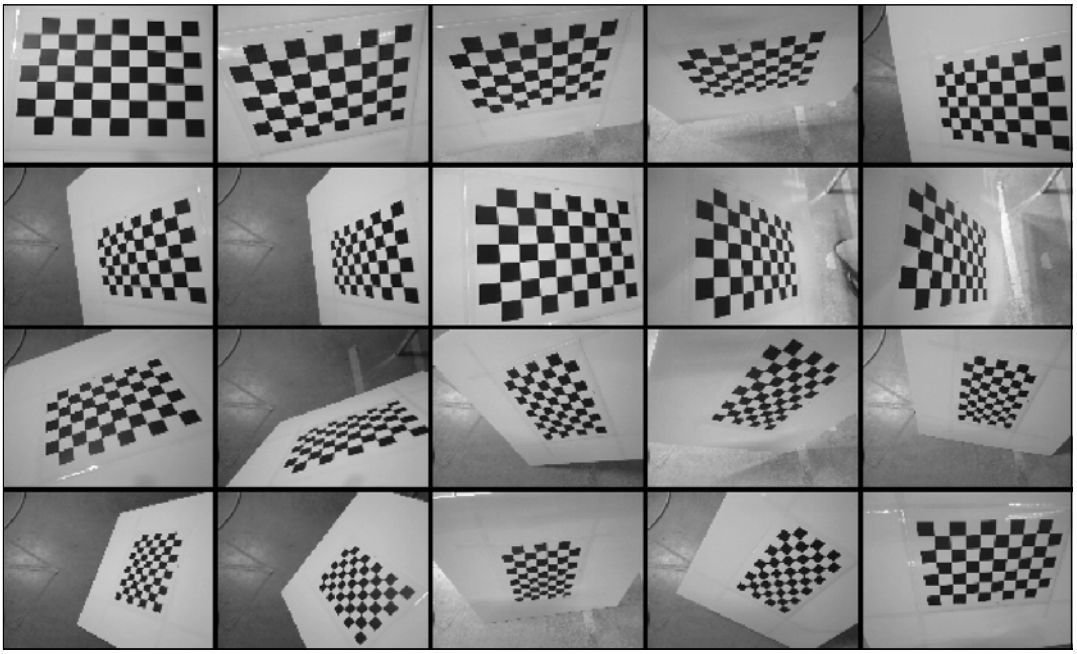
$$f_x(c - o_c)(r_{11}x^w + r_{12}y^w + r_{13}z^w + T_x) = f_y(r - o_r)\left(r_{21}x^w + r_{22}y^w + r_{23}z^w + T_y\right) \tag{4.16}$$

which has 9 unknowns ($f_x, f_y, r_{11}, r_{12,}r_{13}, r_{21}, r_{22}, r_{23}, T_x, T_y$). Here R and T are the extrinsic parameters, which denote the coordinate system transformations from the 3D world coordinates to the 3D camera coordinates. Equivalently, the extrinsic parameters define the position of the camera center and the camera heading in world coordinates, which may be determined by the homogeneous transformation as discussed

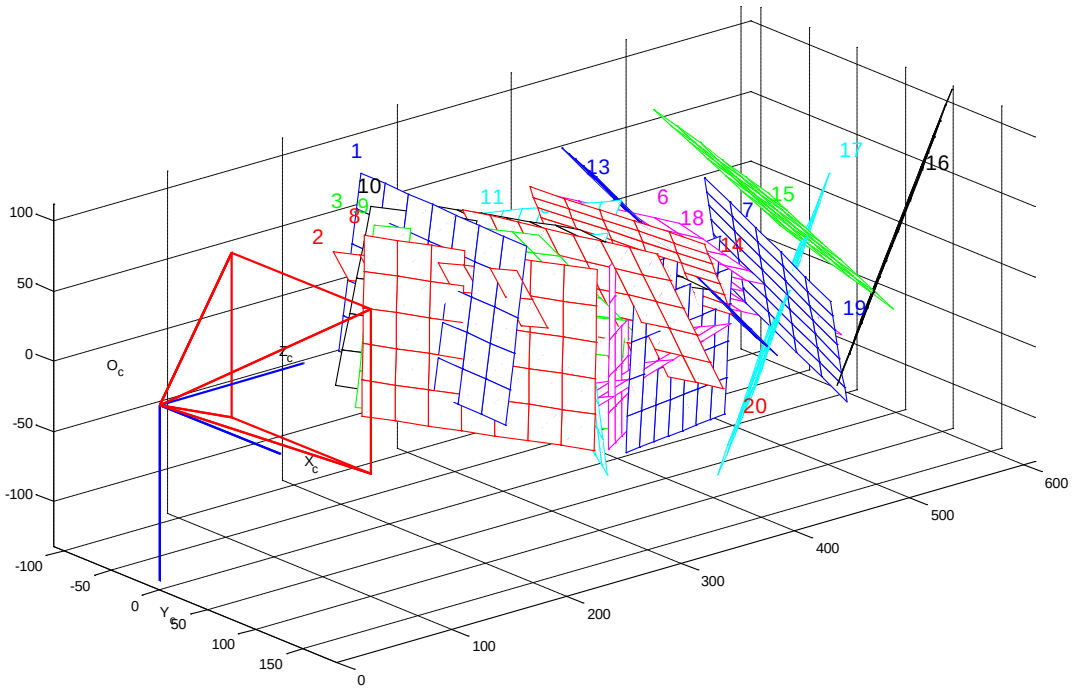
previously. The intrinsic parameters are determined by a camera calibration tool, as presented in (Zhang, 1999).

An Example of Intrinsic Parameter Determination

The intrinsic parameters determine the characteristics of a camera sensor, and they will remain constant for a camera. A chess board (Figure 4.7(a)) along with the Matlab camera calibration toolbox is utilized to determine the parameters of a Logitech web camera (Figure 4.7(b)). The results show that the Logitech camera has the focal length: f = [562.17889 522.49795] ± [1.38737 1.29051]; and the principal point: cc = [332.45990 236.88157] ± [2.13199 1.73026].



(a)



(b)

Figure 4.7: (a) Images for calibration; (b) camera reference and extrinsic parameters.

4.1.4 Models of Visual Servoing

The goal of image based visual servoing (IBVS) is to eliminate the position errors of the feature points in the image plane by controlling the velocities of the robot joints. To this end, the inputs and the outputs of the models are the linear/angular velocities of the robot joints and the positions of the feature points in the image plane, respectively. For modeling the manipulator, first, the relationship between the velocity of the end effector with respect to the robot base frame and the velocity of each joint is derived as given by:

$$\xi_t^r = J_1 \dot{q} \tag{4.17}$$

where J_1 is the Jacobian matrix (Spong, et al., 2006), and $\dot{q}$ is a vector which represents the linear/angular velocities of the joints of the robot. The velocity of the end effector with respect to its own frame is given by

$$\xi_t^t = G^{-1} \xi_t^r \tag{4.18}$$

where $= \begin{bmatrix} R_t^r & 0_{3\times3} \\ 0_{3\times3} & R_t^r \end{bmatrix}$. Since camera is rigidly attached to the end effector, the camera frame and the end effector frame have a constant relationship of homogeneous

transformation. Therefore, one can establish the relationship between the camera velocity and the velocity of the end effector as:

$$\xi_t^t = J_2 \xi_c^c \tag{4.19}$$

where $J_2 = \begin{bmatrix} R_c^t & s(d_c^t)R_c^t \\ 0_{3\times3} & R_c^t \end{bmatrix}$. Next, the relationship between the camera velocity and the velocities of the feature points in the image plane is written by using the interaction matrix as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = L\xi_c^c \tag{4.20}$$

Then, the relationship between the velocities of the feature points in the image plane and the velocities of the robot joints can be found through equations (4.17)-(4.20) as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = LJ_2^{-1}G^{-1}J_1\dot{q} \tag{4.21}$$

The model of the mobile robot can also be generated by using a similar procedure, and the result is given by

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = LG^{-1}J\dot{q} \tag{4.22}$$

Finally, the relationship between the positions of the feature points in the image plane and velocities of the joints can be derived through the integration of (4.21) and (4.22), which are the mathematic models of these two image-based visual servo systems.

4.2 Traditional Image-based Visual Servo (IBVS) Controller

A traditional image-based eye-in-hand visual-servo control law is developed for motion control of both Powerbot mobile platform and RobuArm robotic arm. The underlying concept of this IBVS is that the controller will continuously adjust the speed of the joints (rotational speed of the wheels for the Powerbot; and the speed of the revolute joints of the RobuArm) so that the coordinates (u, v) of the feature point are moved toward the desired position (u_d, v_d) on the image. In particular, the error vector of the image feature point is given by

$$e = \begin{bmatrix} u - u_d \\ v - v_d \end{bmatrix} = \begin{bmatrix} -s_x(r - r_d) \\ -s_x(c - c_d) \end{bmatrix} \tag{4.23}$$

where s_x and s_y are the dimensions of the pixel of the image sensor, which is a charge-coupled device (CCD) in this case.

The velocity of the feature point can be expressed as

$$\dot{e} = \begin{bmatrix} \frac{d(u-u_d)}{dt} \\ \frac{d(v-v_d)}{dt} \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \tag{4.24}$$

By substituting equations (4.21) and (4.22) into (4.24), the following two equations are obtained:

$$\dot{e} = LJ_2^{-1}G^{-1}J_1\dot{q} = M_1[\omega_1 \quad \omega_2 \quad \omega_3 \quad \omega_4 \quad \omega_5 \quad \omega_6]^T \tag{4.25}$$

$$\dot{e} = LG^{-1}J\dot{q} = M_2[v \quad \omega] \tag{4.26}$$

Here, M_1 and M_2 are time-varying matrices which become constant in each iteration after their time varying parameters are linearized through updating. In view of equations (4.25) and (4.26), assuming the error dynamics obeys $\dot{e} = -ke$, a proportional controller based on the Lyapunov method is designed according to:

$$[\omega_1 \quad \omega_2 \quad \omega_3 \quad \omega_4 \quad \omega_5 \quad \omega_6]^T = M_1^{-1}(-ke) \tag{4.27}$$

$$[v \quad \omega] = M_2^{-1}(-ke) \tag{4.28}$$

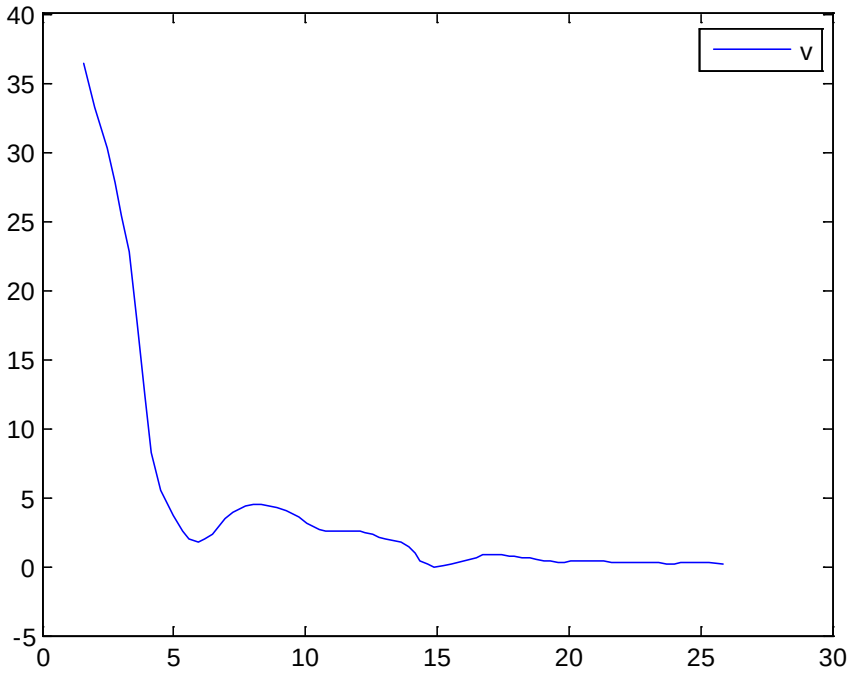
Here, k is the scalar proportional gain, with $k > 0$. The control law for both the mobile robot and the manipulator is obtained by substituting equation (4.23) into (4.27) and (4.28), as

$$\Omega = -kM^{-1}\begin{bmatrix} -s_x(r-r_d) \\ -s_y(c-c_d) \end{bmatrix} \tag{4.29}$$

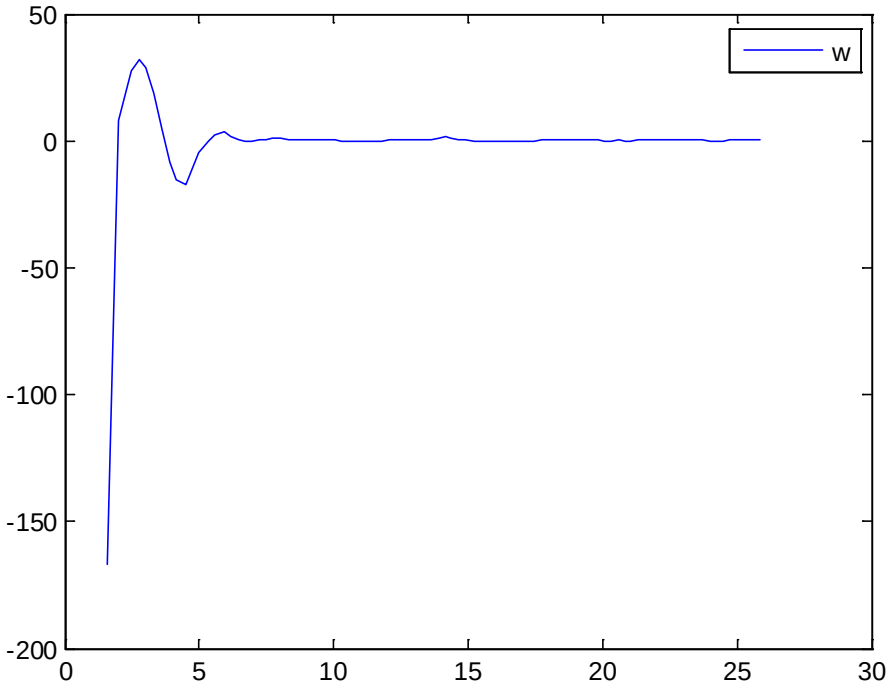
where Ω is the velocity vector of either the mobile robot or the RobuArm. In equation (4.29), the pixel coordinates r and c can be measured directly from the current image using the available image processing software. Therefore, according to equation (4.29), the desired angular velocities of each joint can be directly computed from the image measurements. Moreover, it is noted that the developed controller guarantees asymptotic stability of the closed-loop system, in view of the error equation $\dot{e} = -ke$.

Simulation Results

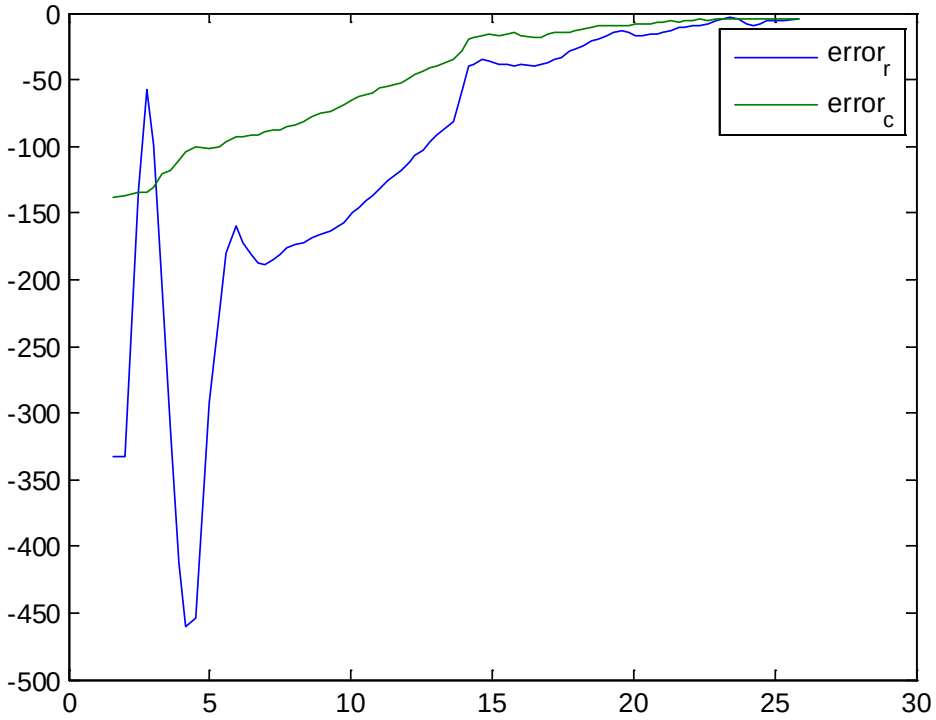
Figure 4.8 shows the simulation results of traditional visual servoing for the mobile base. Figure 4.8 (a) and (b) illustrate the linear velocities and angular velocities of the mobile robot, and Figure 4.8 (c) shows the position errors of the feature point on the image plane which converge to zero.


v
40
35
30
25
20
15
10
5
0
-5
0
5
10
15
20
25
30

(a)


w
50
0
-50
-100
-150
-200
0
5
10
15
20
25
30

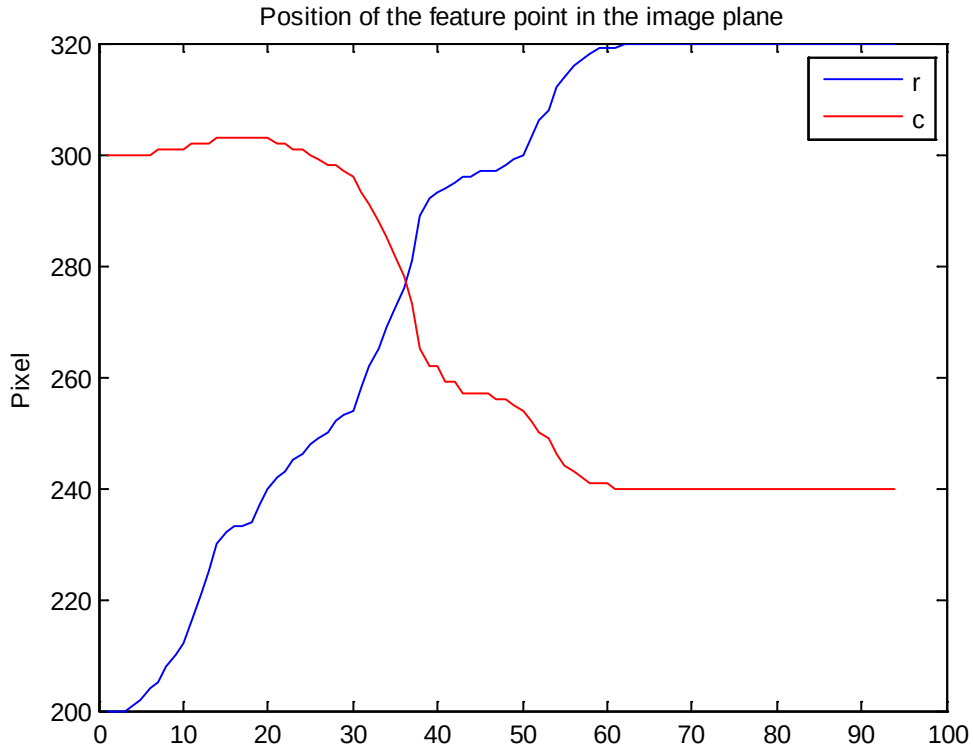
(b)



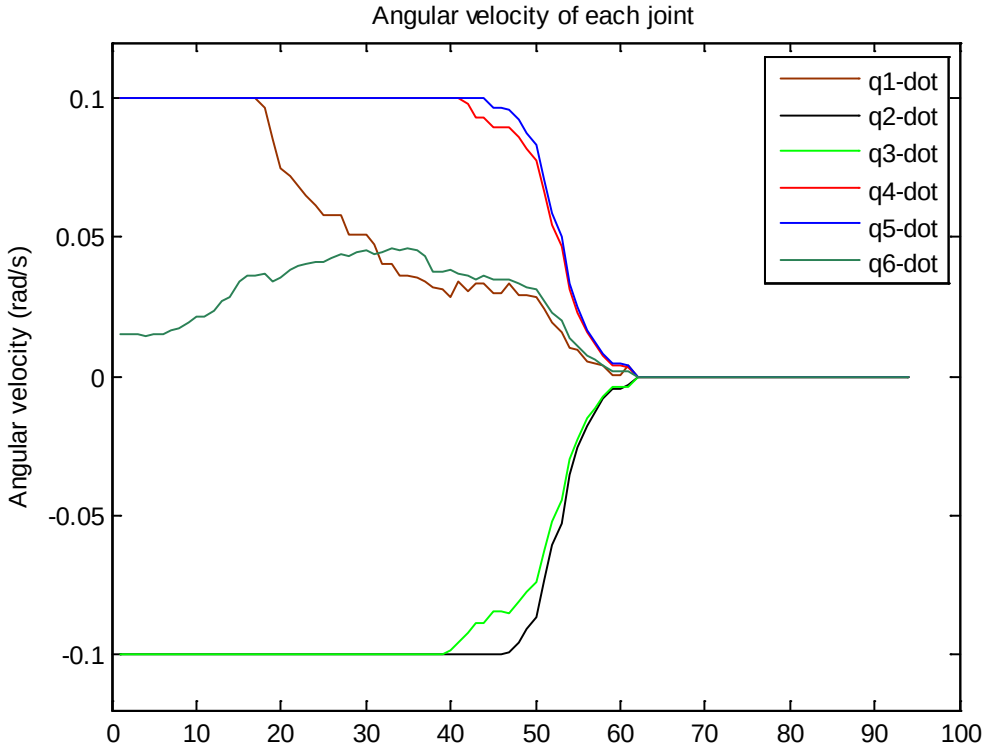
(c)

Figure 4.8: Simulation results of traditional visual servoing of mobile robot: (a) linear velocity trajectory; (b) angular velocity trajectory; (c) position errors of the feature on the image plane.

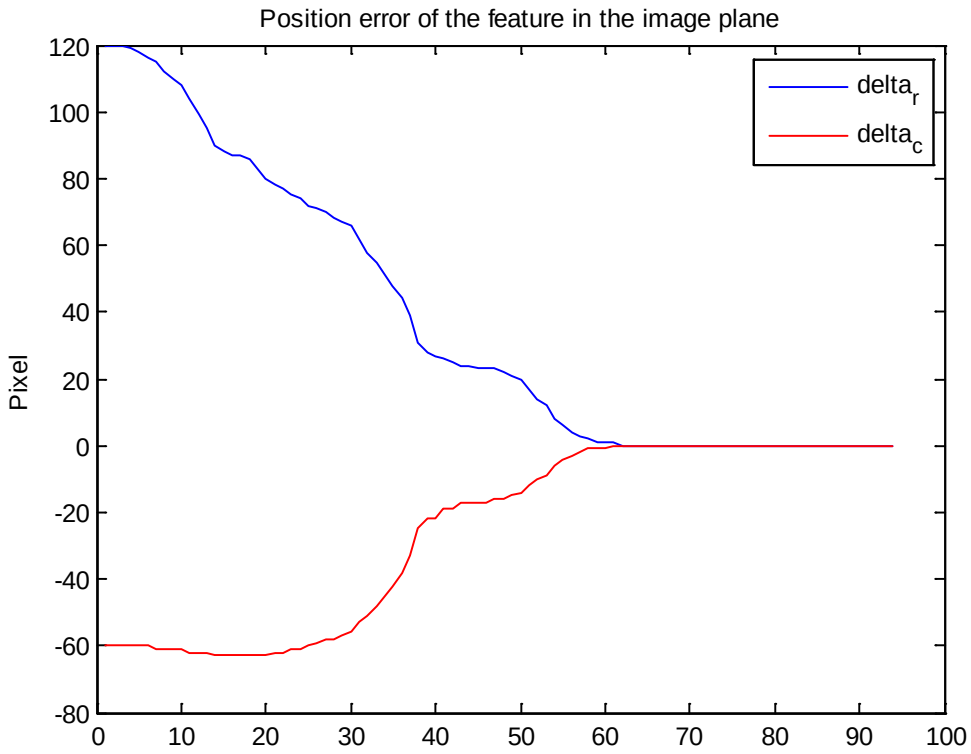
Figure 4.9 shows the simulation results of the traditional image based visual servoing (IBVS) of the robotic manipulator. Figure 4.9(b) shows the angular velocities of the six joints of the manipulator, and Figure 4.9(a) and (c) illustrate the trajectories of the feature point on the image plane along with the position errors.


Position of the feature point in the image plane
Pixel
r
c
320
300
280
260
240
220
200
0
10
20
30
40
50
60
70
80
90
100

(a)


Angular velocity of each joint
Angular velocity (rad/s)
q1-dot
q2-dot
q3-dot
q4-dot
q5-dot
q6-dot
0.1
0.05
0
-0.05
-0.1
0
10
20
30
40
50
60
70
80
90
100

(b)



(c)

Figure 4.9: Simulation results of traditional visual servoing of a robotic manipulator: (a) position trajectory of the feature point on the image plane; (b) angular velocities of the six joints; (c) position error trajectories of the feature on the image plane.

Experimental Results

In this section the vision-based mobile grasping system as developed in our laboratory is employed to validate the visual-servo controller. In the experiment, the mobile robot employs its on-board CCD stereo camera to continuously observe the position of the target object on the image plane with a color tracking algorithm which has been described in the previous chapter. It computes the visual error e on the image plane in each iteration. Next, the IBVS controller determines the desired wheel speeds ω_1 and ω_2 using equation (4.28), and accordingly commands the low-level controller of the mobile robot to gradually move the robot toward the target object. When the robot is sufficiently close to the object, it stops and the visual servo controller of the manipulator is activated for grasping. A Logitech camera is mounted on the end effector of the robotic

manipulator, acting as an eye-in-hand configuration. A laser distance finder is utilized to measure the depth information between the camera and the object of interest in the workspace.

Figures 4.10-4.13 show the experimental results from the traditional visual servo controller for the mobile robot. The trajectory of the visual feature point on the image plane is shown in Figure 4.10. The initial position of the visual feature is close to the top-right corner of the image. Then it moves directly toward the desired position at the bottom-left corner. The position and heading histories of the robot during the entire process are shown in Figure 4.11, the pixel errors on the image plane are shown in Figure 4.12, and the control inputs are given in Figure 4.13.

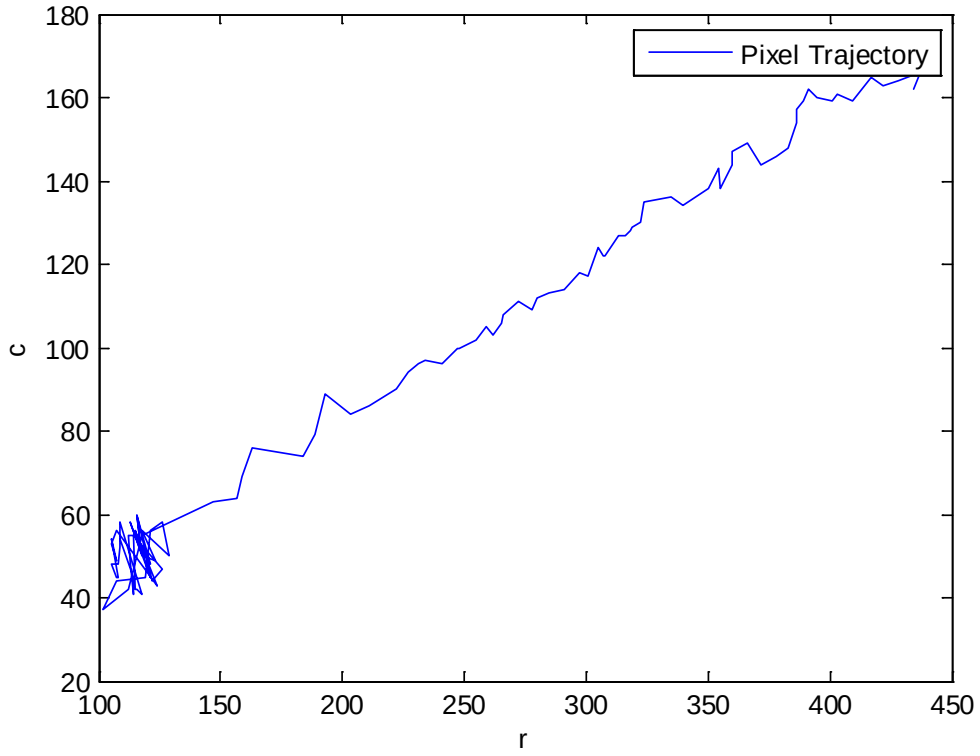


Figure 4.10: The trajectory of the visual feature point (object) on the image plane.

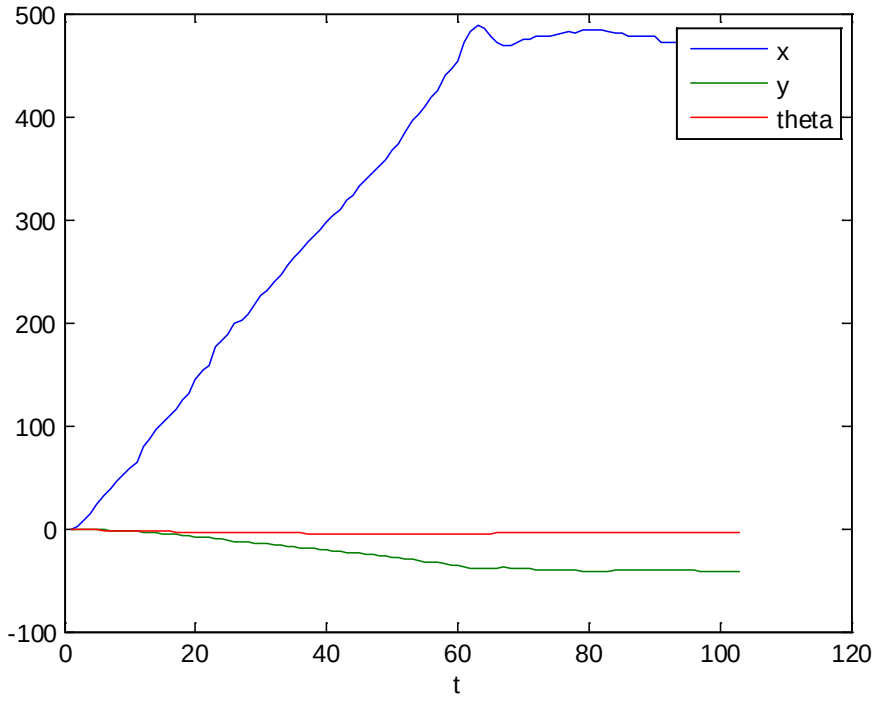


Figure 4.11: The trajectory (position and heading) of the mobile robot in the physical environment when it carries out the mobile manipulation task.

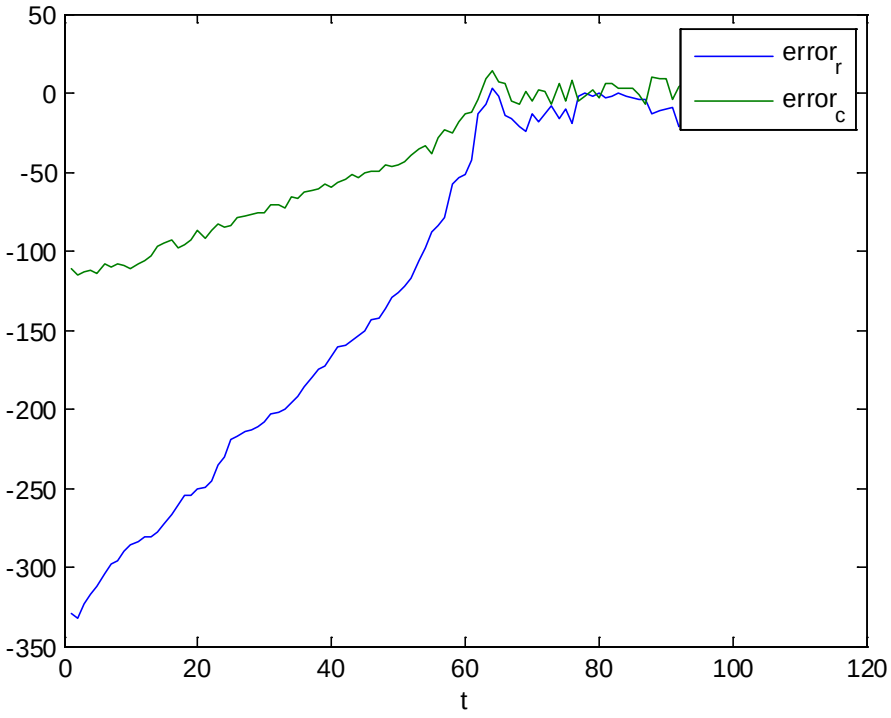


Figure 4 12: The visual errors on the image plane when the robot approaches the object and attempts to grasp it.

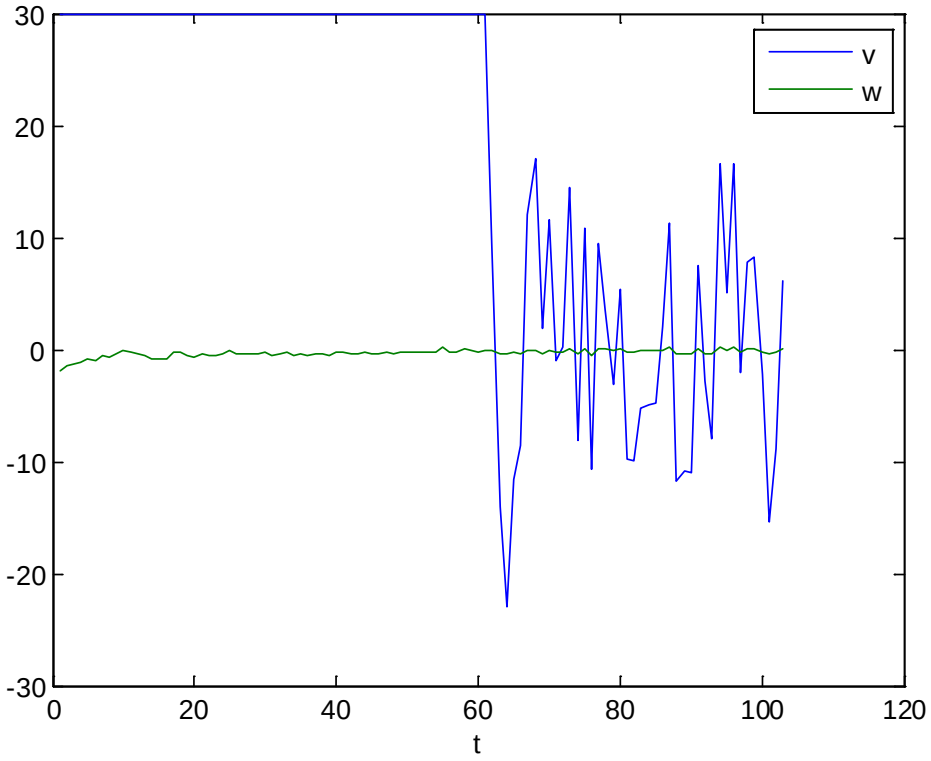


Figure 4.13: The control inputs of the plant when the new visual servo controller is operating.

From the results of both computer simulation and physical experimentation it is seen that the developed IBVS controller is able to effectively drive a wheeled mobile robot toward a target object and guide the robotic manipulator to grasp the object. However, the desired position of the feature point and the initial position of the feature have to be very carefully selected in order to make the system work. The reasons are as follows:

- It is easy to lose the view of the object of interest when the system is operating; especially, when the feature point is close to the margins of the image plane. On one hand, in order to increase the speed of response and to reduce steady state errors, the controller gain has to be sufficiently high. On the other hand, when the controller gain is too large, the control inputs (v and ω) increase correspondingly and as a result the visual features can quickly move out of the image plane, which leads to failure of the controller. This controller cannot guarantee the retention of visual features within the image plane.

- It does not consider the physical constraints of the robot. Therefore, the system can easily fail due to reaching a singular configuration or other physical constraints.
- The selection of the proportional gain is not straightforward. There has to be a trade-off between the performance and keeping the feature point inside the camera view.
- In the case of large displacement, the controller may enter a local minimum or an unstable region.

4.3 Hybrid Visual Servo Control

An important challenge of visual servoing, as discussed in the previous section, is attributed to the visibility constraint because if the feature point is moved out of the image plane, the entire system will fail. In view of such shortcomings, it is necessary to improve the robustness of the traditional visual servo controller. In the present section, a hybrid switching controller is developed to eliminate the main shortcomings of the previous controller. The new control scheme is schematically presented in Figure 4.16.

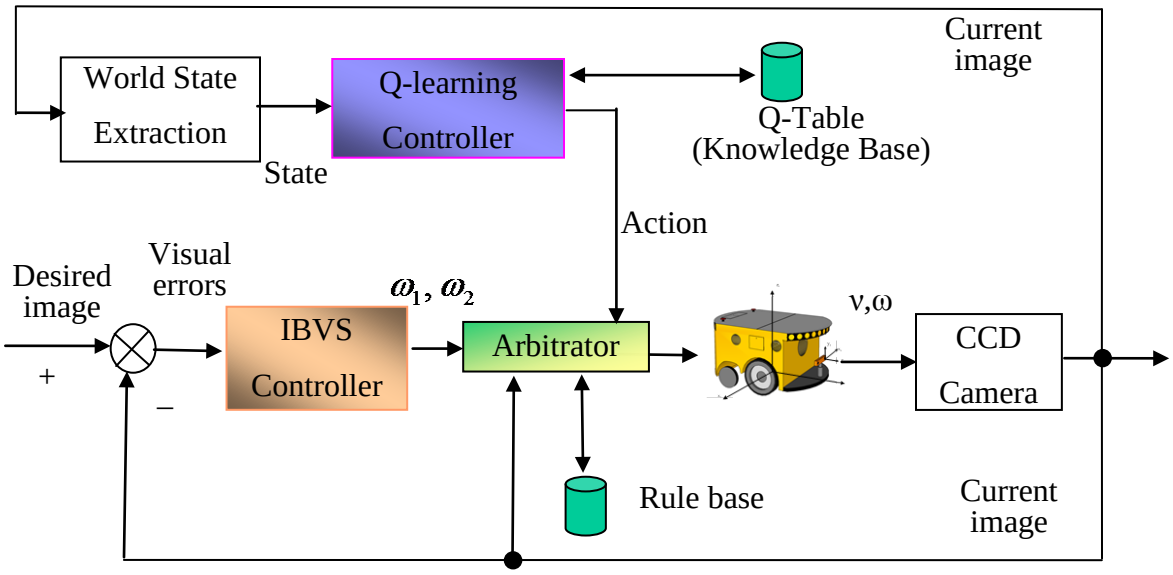


Figure 4.14: The hybrid control scheme for robust visual servoing.

In Figure 4.14, there are two control loops: the traditional IBVS controller and a new Q-learning controller. While the IBVS controller will drive the mobile robot toward the target object, the Q-learning controller will observe the visual features on the image

plane and select an optimal action (an appropriate rotational or translational movement) so that the visual features are pushed from the image edge to the center. In addition, the Q-learning controller is able to continuously learn and improve its action-selection policy on-line using its machine learning algorithm.

There is a rule-based arbitrator in the control system, as indicated in Figure 4.16, which autonomously switches between the outputs of the controllers so that the overall hybrid control system achieves a trade-off between its robustness and accuracy.

Finally, the Q-learning controller can be trained offline to improve its performance. Once it is trained, the Q-learning controller can quickly select the correct actions in a real-time manner (usually in less than 10 ms in our experiments).

Q-learning Controller

The Q-learning controller shown in Figure 4.14 is a customized controller, which will keep within the image plane the important visual features of an object of interest. It is based on the machine learning approach called Q-learning. The main advantage of the Q-learning controller is that it is able to autonomously learn the action-selection policy of the mobile robot and improve the controller performance continuously so that the visual features remain in the field of view of the CCD camera. Due to integration of the Q-learning controller with the IBVS controller developed in Section IV, the robustness of the resulting hybrid controller is improved.

The world states in the present Q-learning controller are defined by a discrete grid of the image plane, as shown in Figure 4.15.

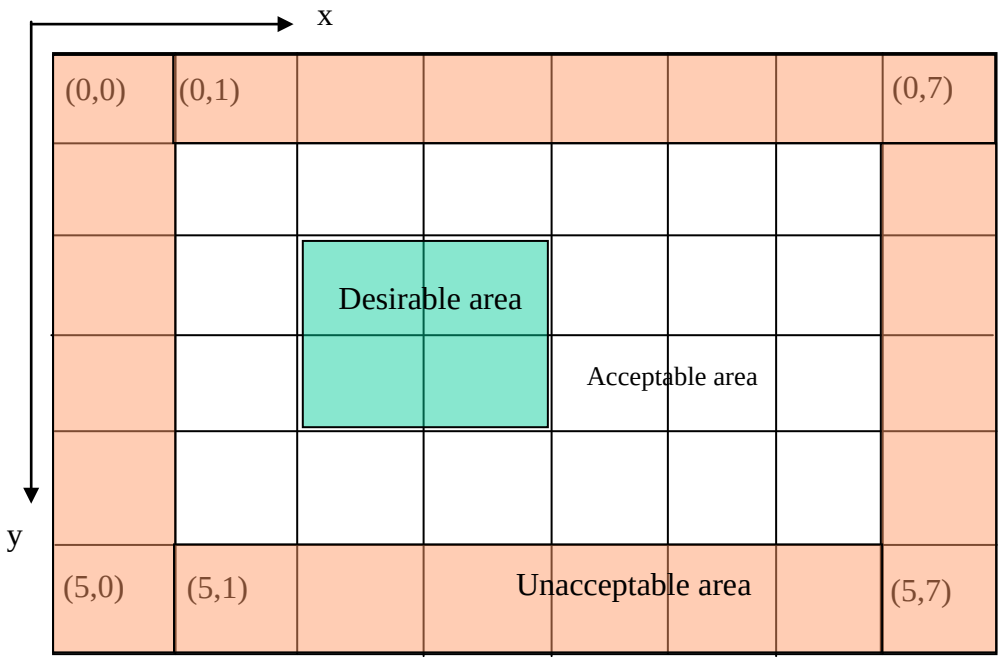


Figure 4.15: The discrete gird world defined on a 640×480 CCD image plane.

In Figure 4.15, the 640×480 CCD image plane is divided into an 8×6 discrete grid world where each cell of the grid has a length of 80 pixels. When an image is grabbed from the CCD camera, the position of the visual feature point on the image plane can be easily converted into the coordinates in the grid world as follows:

$$x = INT(r/80) \tag{4.30}$$

$$y = INT(c/80) \tag{4.31}$$

Here $r = 0,1,\cdots,639$ and $c = 0,1,\cdots,479$ are the pixel coordinates of the visual feature points on the image plane, $x = 0,1,\cdots,7$ and $y = 0,1,\cdots,5$ are the corresponding grid coordinates in the grid world, and $INT()$ is a function which converts a floating-point number into an integer by discarding its decimal portion.

The world state in the Q-learning controller is made up of the grid coordinates of the visual feature and the current depth from the robot to the target object, as given by:

$$s = (x, y, d) \tag{4.32}$$

Here, d is a discrete index value of the current depth, which is computed according to:

$$d = \begin{cases} 0, & if\ depth < 40cm \\ 1, & if\ 40cm < depth <= 90cm \\ 2, & if\ 90cm < depth <= 140cm \\ 3, & if\ depth > 140cm \end{cases} \tag{4.33}$$

Under each state, it is assumed that the mobile robot is able to select one of the following four actions:

Action #1: Move forward for 5 cm.

Action #2: Move backward for 5cm.

Action #3: Rotate for 5 degrees.

Action #4: Rotate for -5 degrees.

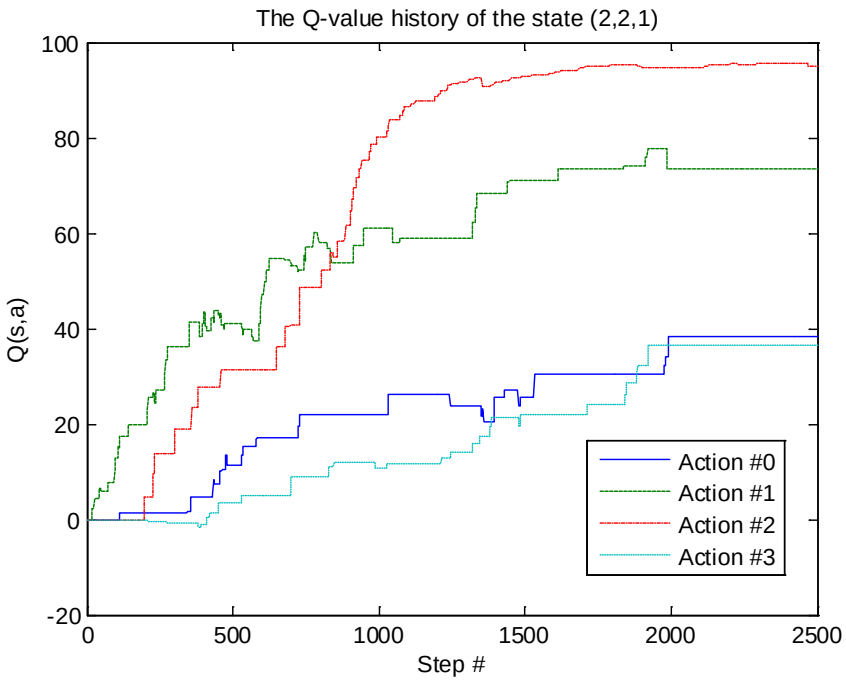
After the robot takes an action, it will receive a reward r from the environment. This reward is computed based on the new position of the visual feature on the image plane, according to:

$$r = \begin{cases} +20, & if\ (x, y) \in the\ desirable\ area \\ 0, & if\ (x, y) \in the\ safe\ area \\ -10, & if\ (x, y) \in the\ dangerous\ area \\ -20, & if\ (x, y)\ is\ out\ of\ the\ image \end{cases} \tag{4.34}$$

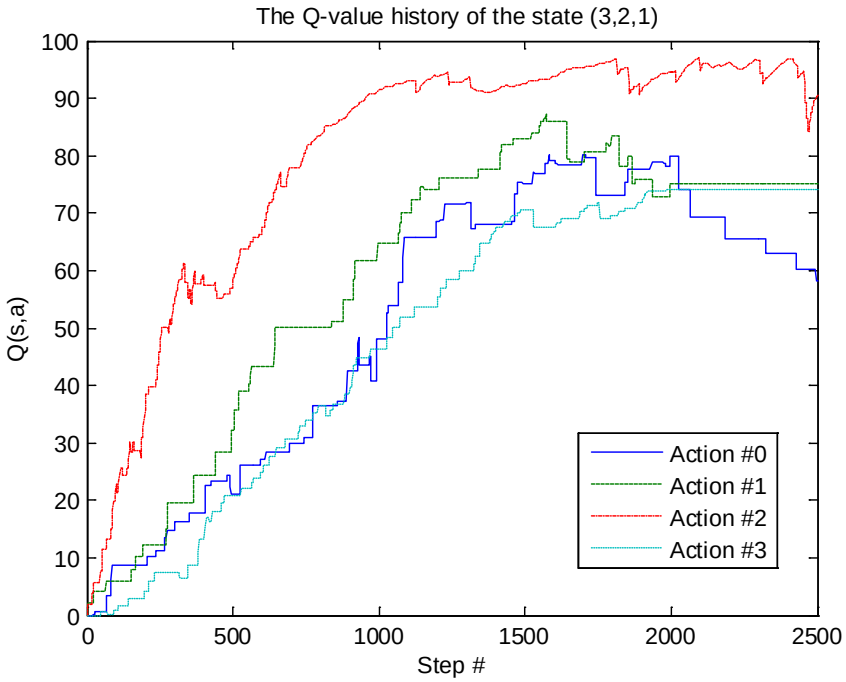
Here, (x, y) is the new position of the visual feature in the grid world after the robot takes an action. The definitions of the desirable, acceptable and unacceptable areas are given in Figure 4.15. From equation (4.34) it is clear that the robot is encouraged to select the correct actions, which will push the visual features toward the desirable area and away from the unacceptable area on the image plane.

Training Results

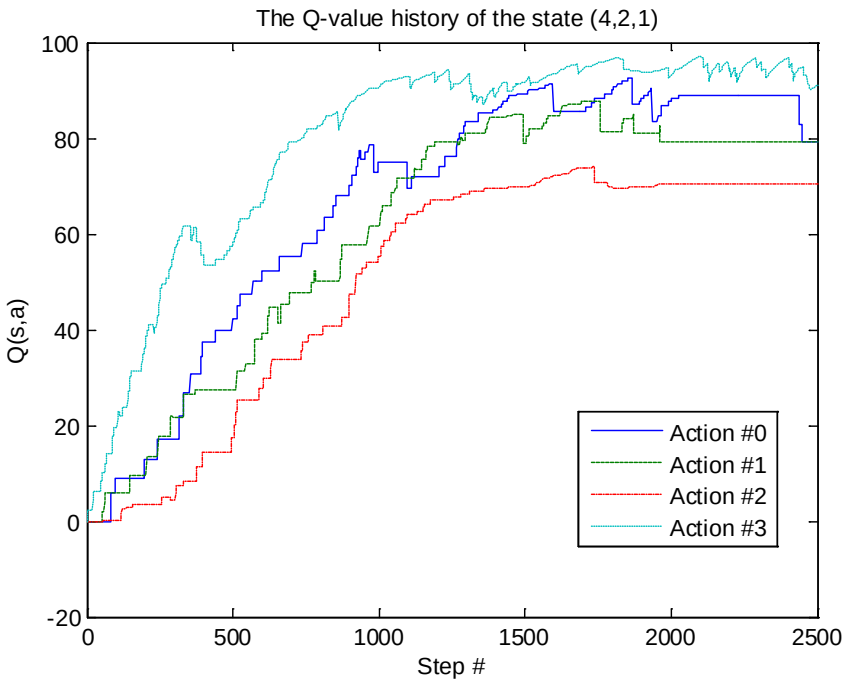
The Q-learning controller as presented here was trained with the physical robot-camera system described in Figure 4.14, to gain on-line experience and then learn/update its Q-values. In particular, the robot was required to track the position of target object with the on-board ACTS color-blob tracking software, and to take a series of actions based on the Q-learning algorithm so that the position of the target object remained in the desirable area of the image. In this experiment, the Q-learning algorithm was run for 2500 iterations (or steps), and the history of the Q values under four sample states was recorded, as presented in Figure 4.16.


The Q-value history of the state (2,2,1)
Q(s,a)
Step #
Action #0
Action #1
Action #2
Action #3
100
80
60
40
20
0
-20
0
500
1000
1500
2000
2500

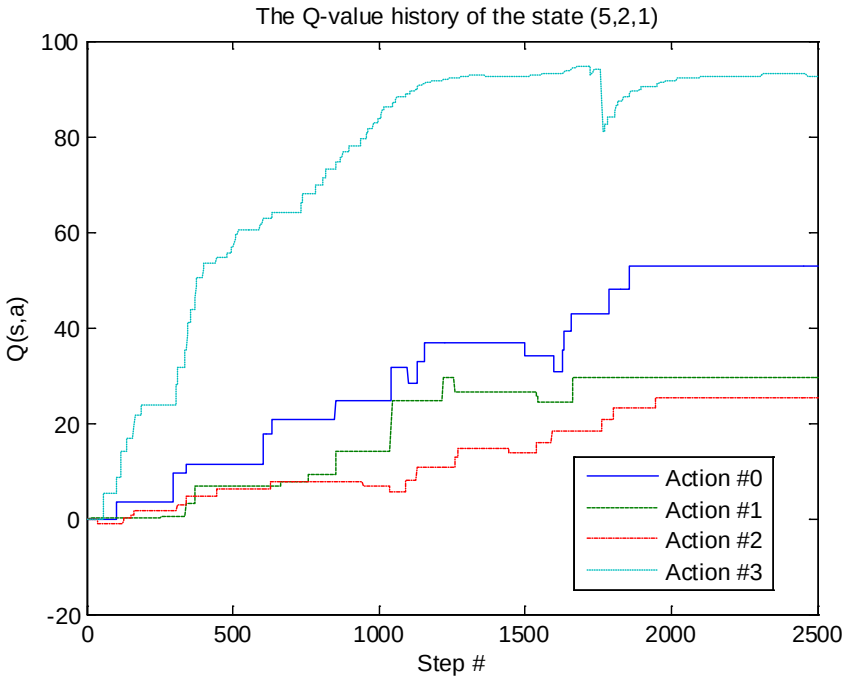
(a)


The Q-value history of the state (3,2,1)
Q(s,a)
Step #
Action #0
Action #1
Action #2
Action #3
100
90
80
70
60
50
40
30
20
10
0
0
500
1000
1500
2000
2500

(b)



(c)



(d)

Figure 4.16: The history of Q-values under different world states when the robot received off-line training: (a) State (2,2,1), (b) State (3,2,1), (c) State (4,2,1), (d) State (5,2,1).

From Figure 4.16 it is observed that the Q-values have converged after 2500 steps of decision-making. It means that the Q-learning controller has learned how to select the correct actions for each state. This will enable the mobile robot to move or rotate properly so as to keep the visual features in the desirable area on the image. In particular, for the state of (2,2,1), Figure 4.16(a) shows that the action #2 has a higher Q-value than the other three actions. Hence this action will have a higher probability to be selected by the Q-learning controller when the state (2,2,1) is observed by the robot. Similar conclusions can be drawn from Figure 4.16(b)-(d).

A trained Q-learning controller is in fact a behavior-based decision-making unit. It continuously observes world states and probabilistically selects actions for the robot based on the Q-values in the Q-table. Unlike the traditional behavior-based system (Arkin, 1998) where the behavior rule base is fixed and entirely designed by a human designer in advance, the rule base (i.e., the Q-table) of Q-learning is learned autonomously when the robot interacts with its work environment, and it can be improved online. Therefore, the Q-learning controller developed in this thesis is more desirable than the traditional behavior-based approaches. Furthermore, it is also more advantageous than the approaches of potential field and navigation function, and those based on path planning (Corke and Hutchinson, 2001; Chesi and Huang, 2007; Chen, et al., 2007; Cowan, et al., 2002; Schramm and Morel, 2006; Zhang and Ostrowski, 2002) because these approaches do not have a self-learning mechanisms and cannot adapt to a changing world and improve their decision-making abilities on-line.

The arbitrator of the hybrid control system

In the hybrid control scheme of Figure 4.14, there is an arbitrator which switches on the IBVS controller or the Q-learning controller depending on the position of the visual feature in the grid world in Figure 4.15. In particular, it is a rule-based arbitrator that switches the outputs of the controllers according to the following rules:

Rule #1: If the visual feature *P* is in the unacceptable area, the Q-learning controller is selected.

Rule #2: If *P* is in the desirable area, the IBVS controller is selected.

Rule #3: If *P* is in the acceptable area, no switching action is taken.

These rules, through the Q-learning controller, push a visual feature P back to the desirable region if it is located in the unacceptable area. Then, the IBVS controller will drive the robot toward the target object.

Experimental Results

Further experiments are carried out to validate the hybrid control scheme developed in this thesis. In the experiment, first, the mobile robot moves close to a target object for autonomous grasping. It utilizes the feedback information from its CCD camera and the hybrid controller developed in the previous session to guide the mobile robot to the grasping position.

Hybrid Visual Servoing with a Small Unacceptable Area

In this experiment, the robot carries out the mobile grasping task in the presence of a small unacceptable area, using the developed hybrid controller. The unacceptable area is given by:

$$r < 60 \text{ or } r > 550 \text{ or } c < 20 \text{ or } c > 450 \quad (4.35)$$

where, r and c are the pixel coordinates. The trajectory of the visual feature on the image plane is presented in Figure 4.17.

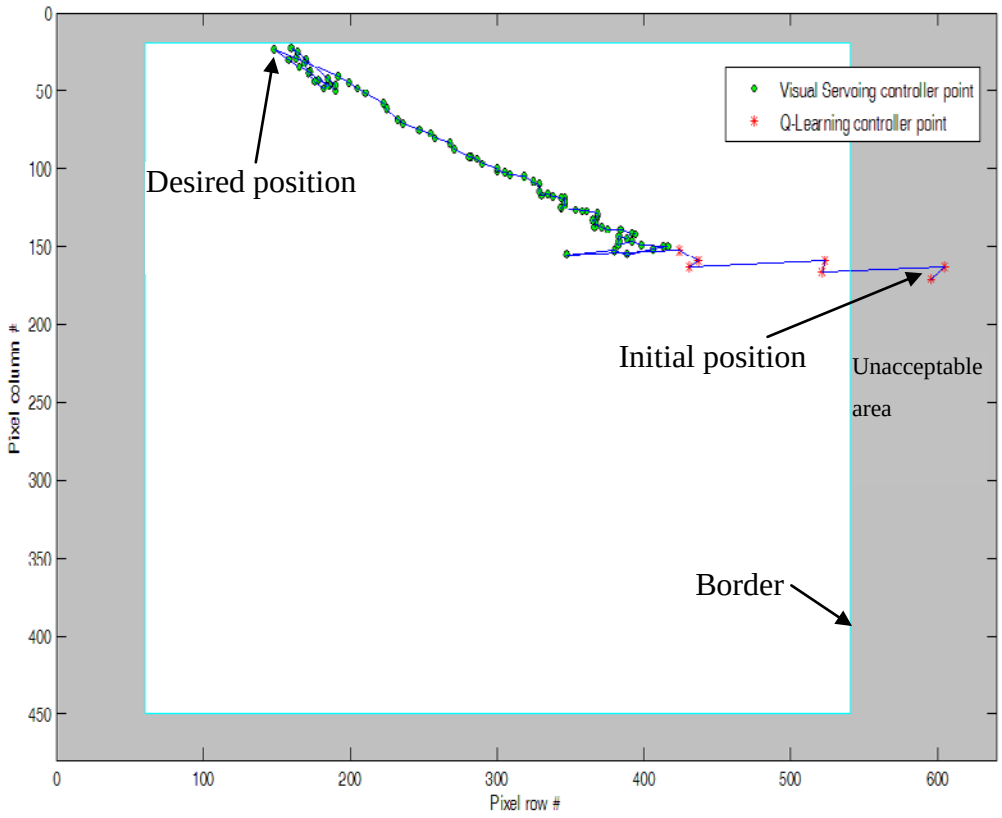


Figure 4.17: The trajectory of the visual feature on the 640×480 image plane when the hybrid controller operated in the presence of a small unacceptable area.

In Figure 4.17, the grey area represents the “unacceptable area” on the 640×480 image plane, which is defined by equation (4.46). Because initially the visual feature on the image is at (587, 165) which is in the unacceptable area, the arbitrator of the hybrid controller selects the Q-learning controller to determine the motion of the mobile robot. From Figure 4.17 it is seen that the Q-learning controller uses just 7 steps to bring the position of the visual feature into the “desirable area” on the image plane. After this quick action, the arbitrator selects the IBVS controller to take over the control of the mobile robot. In the remainder of the control process, the IBVS controller drives the robot to the goal location and orientation until the visual feature reaches its desired pixel position of (180, 40) on the image.

Figure 4.17 shows that the developed hybrid controller performs better than the IBVS controller of Section IV. In particular, when the visual feature is located in the “unacceptable area” of the image plane, the hybrid controller is able to quickly adjust the camera pose so that the visual feature will move through a large distance at a fast speed

of response so as to enter the “desirable area” on the image. It is usually difficult for the IBVS controller to move the visual feature through a large distance due to potential instability problems.

Figure 4.18 presents the history of the row and column pixel coordinates of the visual feature, and the history of controller switching for the experimental result in Figure 4.17.

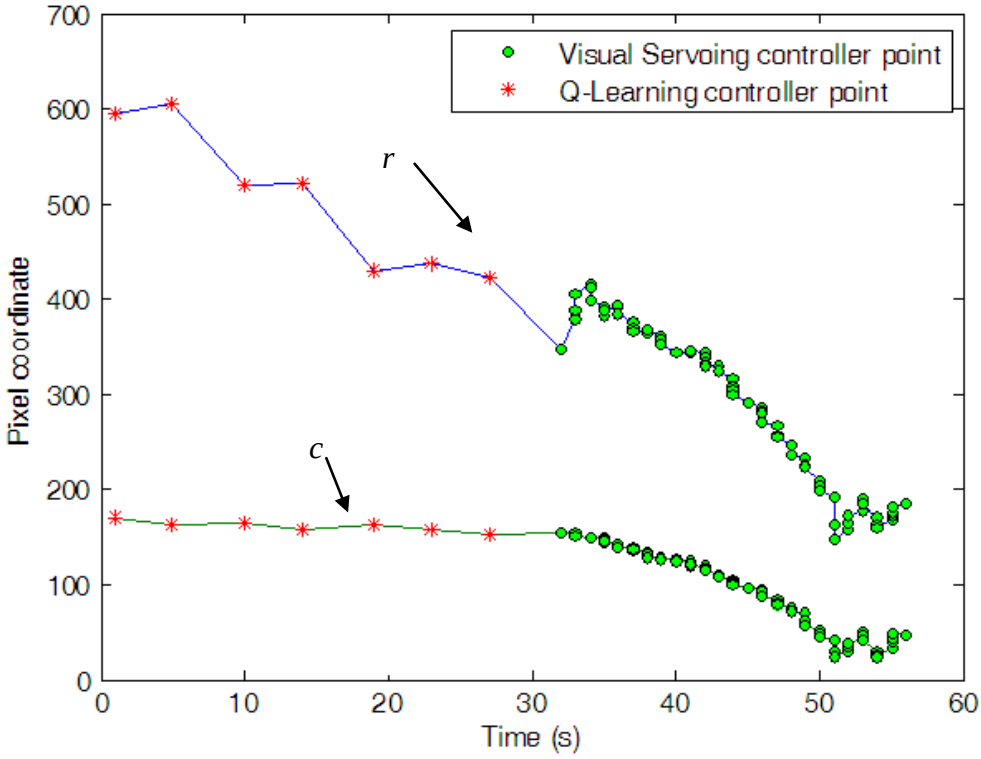
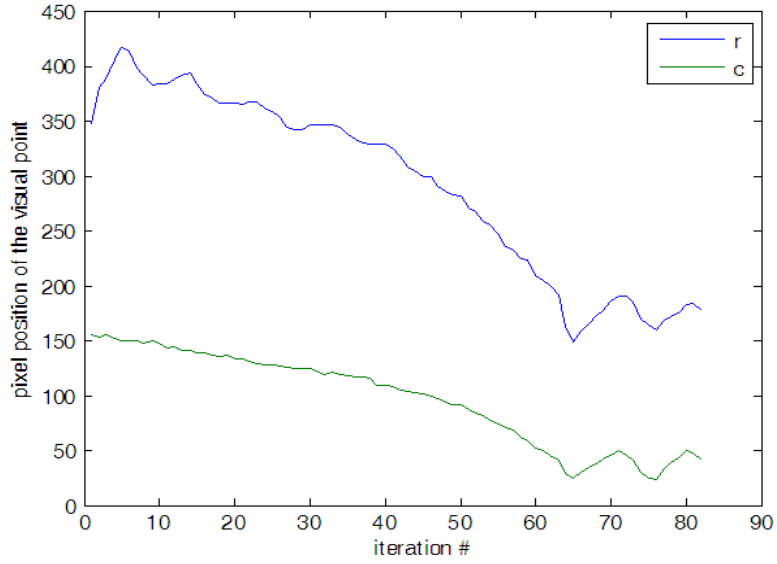
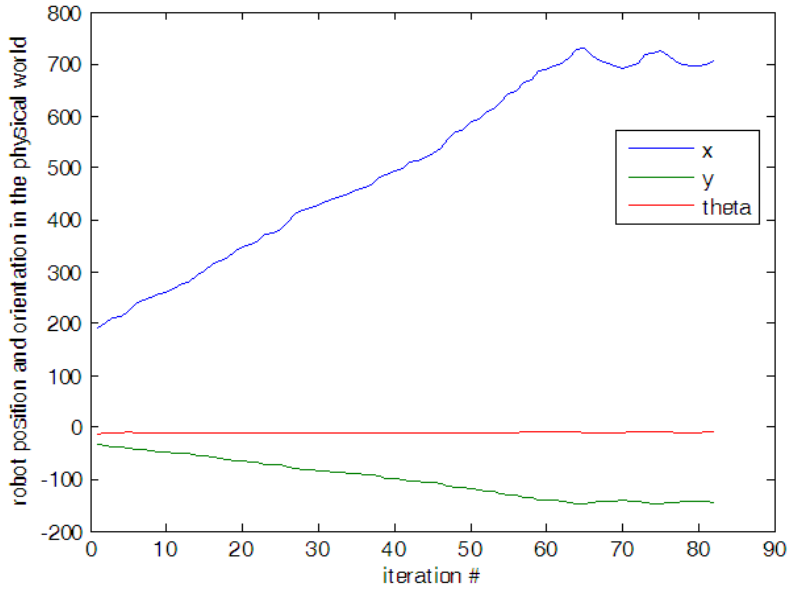


Figure 4.18: The history of the row and column pixel coordinates of the visual feature when the robot approached the object and grasped it.

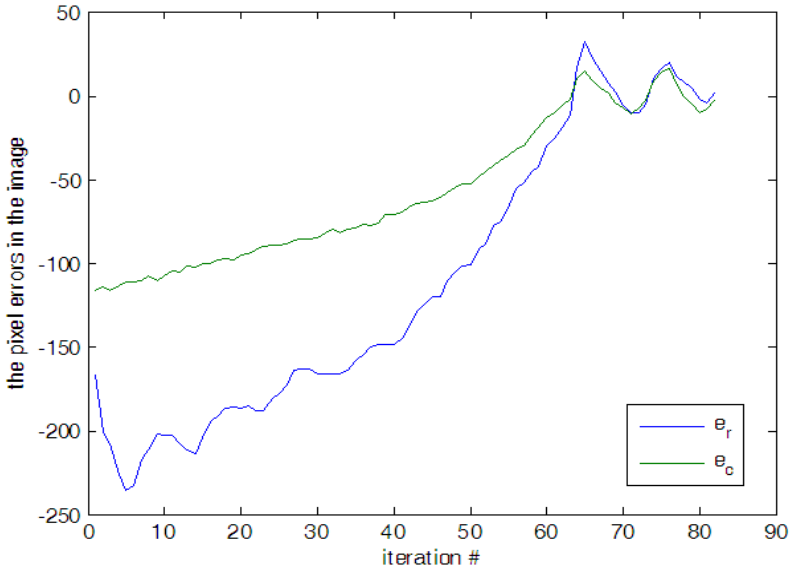
The performance of the IBVS controller within the hybrid controller is shown in Figures 4.19(a)-(c).


r
c
pixel position of the visual point
iteration #
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60
70
80
90

(a)


x
y
theta
robot position and orientation in the physical world
iteration #
-200
-100
0
100
200
300
400
500
600
700
800
0
10
20
30
40
50
60
70
80
90

(b)



(c)

Figure 4.19: The performance of the IBVS controller with a small unacceptable area when the robot carried out a mobile manipulation task: (a) history of the pixel coordinates of the visual feature on the image plane; (b) trajectory of the mobile robot in the physical environment; (c) visual errors on the image plane.

Figure 4.19(c) indicates that steady-state errors can result when the robot reaches its desired pose. These errors are caused due to the proportional controller given by equation (4.29). In the present experiments the steady state errors are quite small and do not result in a failed grasping.

Hybrid Visual Servoing with a Large Unacceptable Area

In this experiment, a large unacceptable area is defined on the image plane of the CCD camera, according to:

$$r < 60 \text{ or } r > 360 \text{ or } c < 20 \text{ or } c > 260 \quad (4.36)$$

It shows that the hybrid controller is able to drive the robot to its desired position and orientation in a robust manner, and make it successfully grasp the target object. When the unacceptable area is large, the hybrid visual-servo controller exhibits a very different behavior, as shown in Figure 4.20.

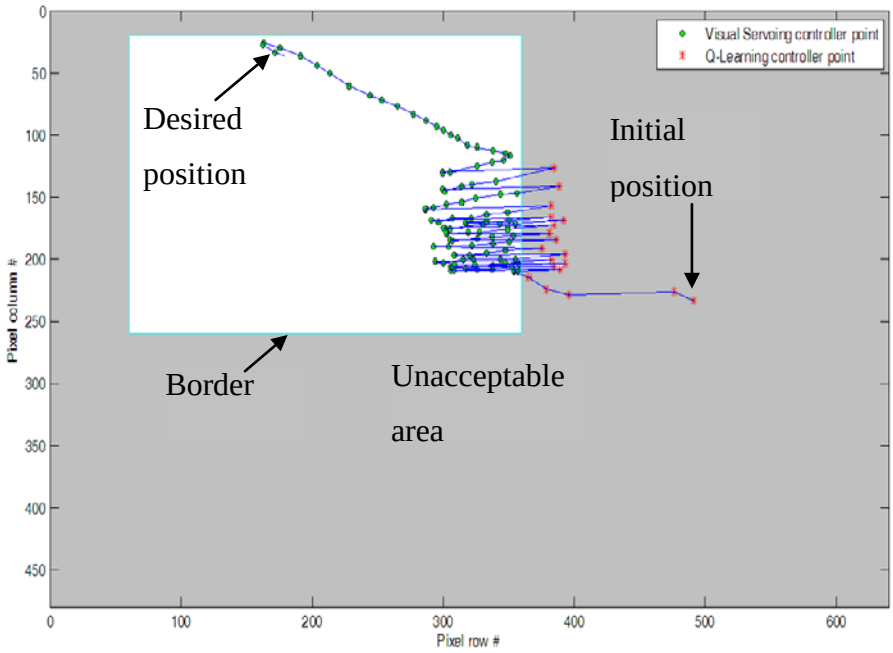


Figure 4.20: The trajectory of the visual feature on the 640×480 image plane (large unacceptable area).

As shown in Figure 4.20 and Figure 4.21, because the unacceptable area is large, under IBVS control a visual feature can easily enter the unacceptable area. Consequently, it is observed that the Q-learning controller is activated by the arbitrator very frequently to push back the visual feature into the acceptable area. From both figures, it is clear that the trained Q-learning controller has been quite successful in keeping the visual feature within the acceptable area. Usually, just one step is needed to bring the visual feature away from the unacceptable area and then transfer the robot control to the IBVS controller.

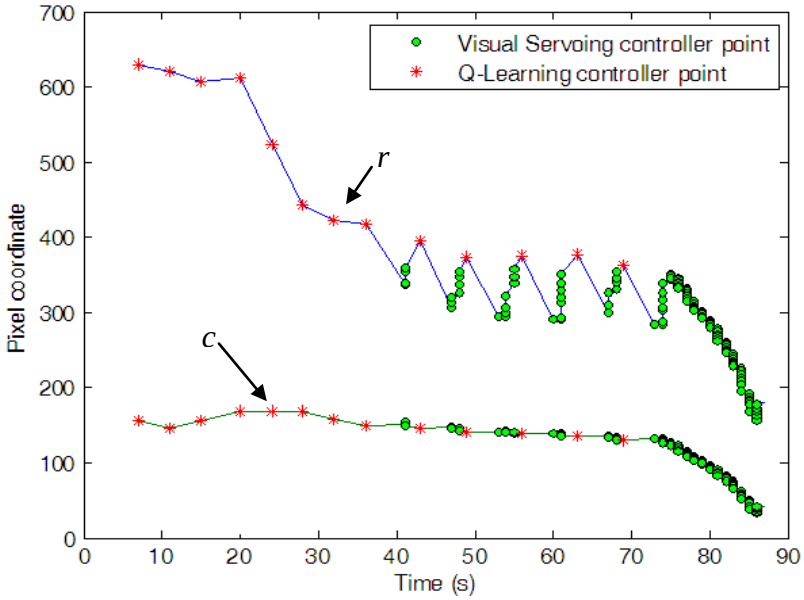
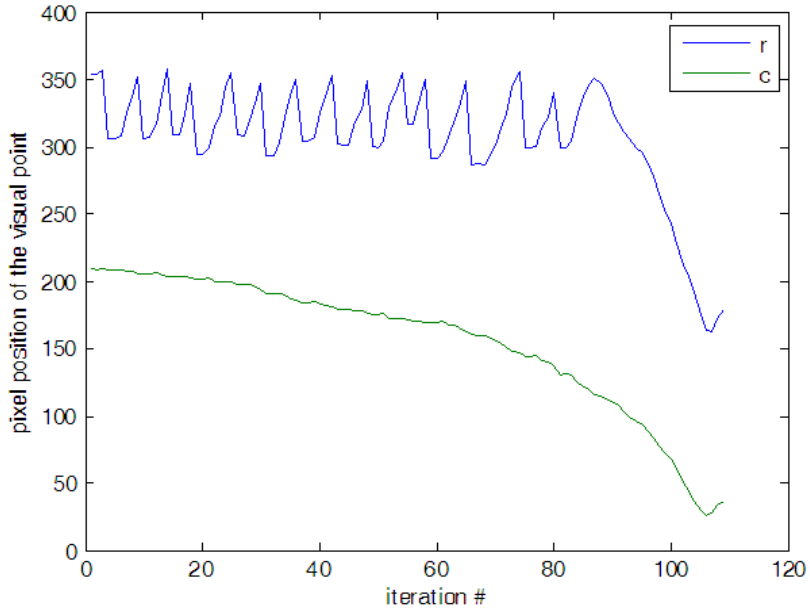
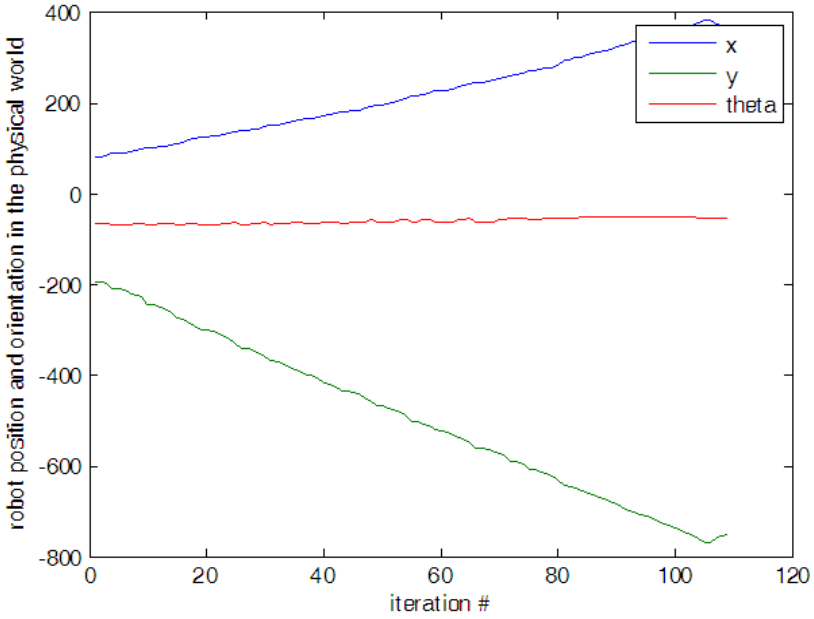


Figure 4.21: The history of the row and column pixel coordinates of the visual feature.

In the present work, the Q-learning controller makes the robotic system more robust (by keeping a visual feature within the acceptable area, under disturbances that drive the feature away from that area) and the IBVS controller guarantees accurate positioning performance. In this manner the hybrid controller developed in the present work provides a good trade-off between robustness and accuracy, as clear from Figure 4.20 and Figure 4.21. The performance of the IBVS controller is shown in Figure 4.22 (a)-(c).



(a)



(b)

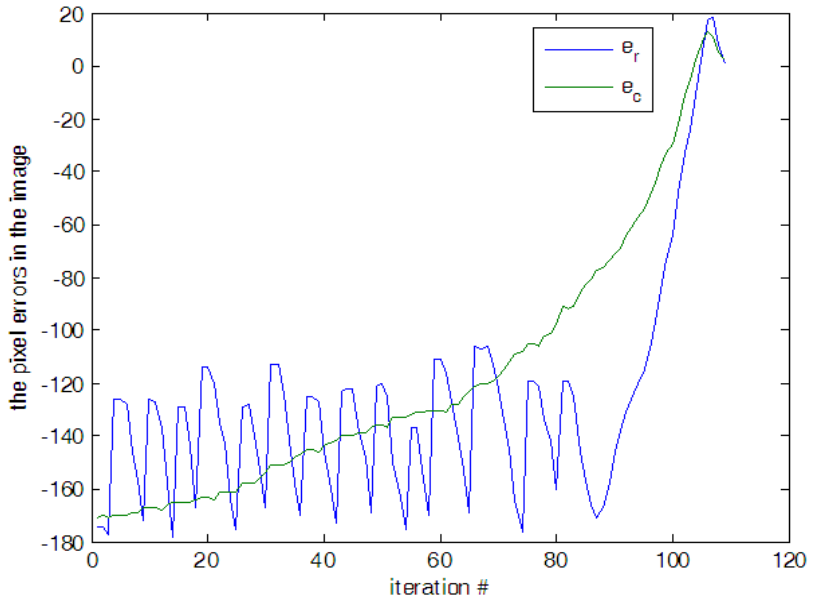


Figure 4.22: The performance of the IBVS controller when the unacceptable area is large: (a) history of the pixel coordinates of the visual feature; (b) trajectory of the mobile robot; (c) visual errors on the image plane.

In Figure 4.20, because initially the visual feature is located in the unacceptable area and since the unacceptable area is much larger than the one in Figure 4.17, the Q-learning controller is activated frequently in the first part of visual servoing to keep the visual feature within the acceptable area. As a result, it is observed from Figure 4.20 that the response curves oscillate significantly in the first stage (from iteration #0 to iteration #60). After the visual feature is moved away from the unacceptable area, the IBVS controller gradually takes control of the mobile robot and consequently the response curves become smoother until the visual feature reaches its desired position on the image plane. Again, Figure 4.20 shows that the hybrid visual-servo controller developed in this project provides a rather "balanced" performance in robustness and accuracy, as a result of the integrated IBVS controller and the Q-learning controller.

4.4 ANMPC Visual Servo Controller

The hybrid controller can solve the visibility constraints in mobile robot manipulation. However, it cannot be applied in robotic arms because a robotic arm has more degrees of freedom (DOFs) than the mobile robot does. Moreover, the controller outputs of the hybrid controller are not optimal. Therefore, an Adaptive Nonlinear Model Predictive Controller (ANMPC) is developed now to overcome the drawbacks of the previously-developed hybrid controller.

The principle of classical Model Predictive Control (MPC) is summarized in Figure 4.23. In each iteration *k*, an estimated model is used to predict the future states. Then, an optimal control law is computed based on the principle of forcing the predicted states to converge to a desired set-point while minimizing a cost function (Camacho and Bordons, 2007).

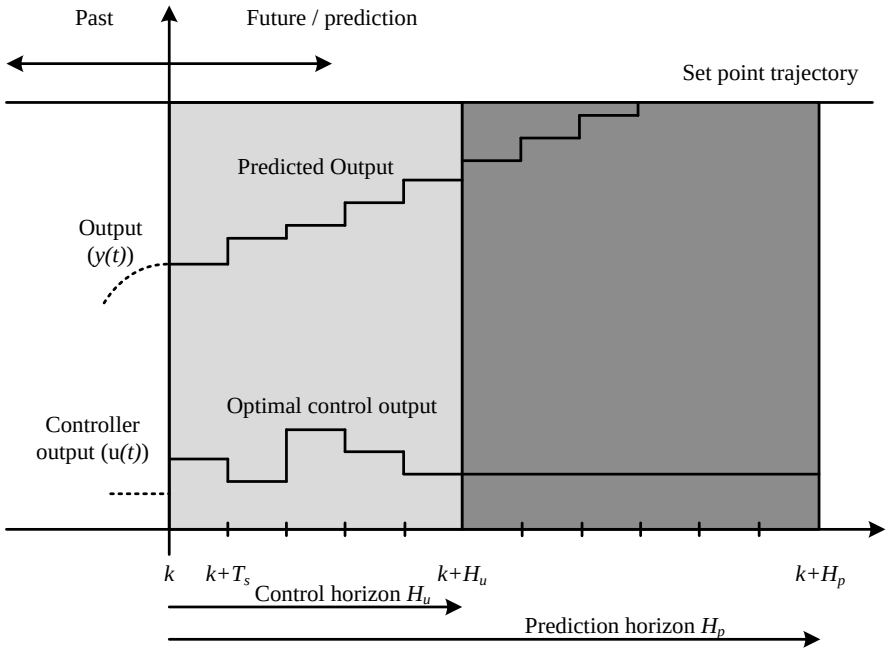


Figure 4.23: Strategy of model predictive control.

The classical predictive control law may present problems when applied in a visual servo system for mobile robots. In particular, because in image-based visual servoing, the current velocity of the visual feature points and depth information are involved in the interaction matrix (Spone and Hutchinson, 2006) of the camera model, the mathematical model of the plant (mobile base, camera and manipulator) is nonlinear and time-varying. In view of long and arbitrary movements of its base, the issues of nonlinearity and time variance will become more significant when visual servoing is applied to a mobile robot.

Due to possible large motions and nonholomic constraints in a mobile robotic system, the traditional visual servo-based velocity controller will usually show poor performance. A customized nonlinear model predictive controller is proposed here to meet these challenges and to improve the performance of the visual servo system. The proposed architecture for nonlinear time-varying model predictive control is shown in Figure 4.24. This architecture is applicable to both mobile robot systems (mobile platforms) and robotic manipulator systems (robotic arms) with slight differences in the plant model in the ANMPC controller, and the dimensions of the controller outputs (it has 2 outputs for the mobile robot and 6 outputs for the robotic manipulator arm).

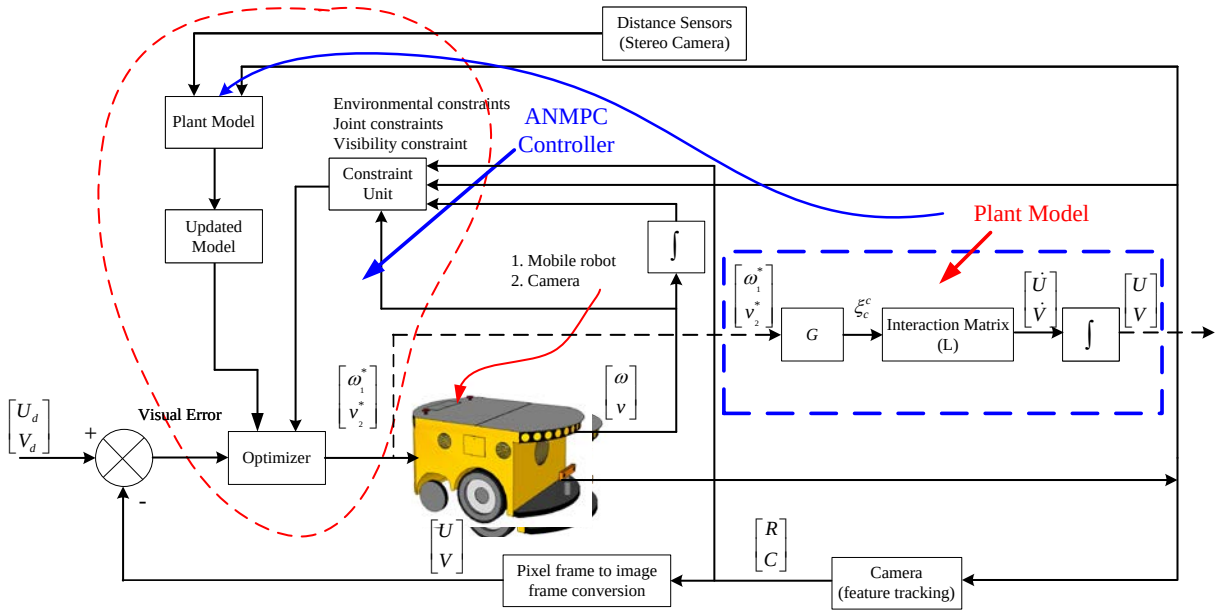


Figure 4.24: Block diagram of the mobile robot system with adaptive nonlinear model predictive control (ANMPC) for visual servoing.

In Figure 4.23, since MPC requires a linear model of the plant. Hence, the nonlinear plant is approximated to an adaptive linear model and updated in each iteration of the control loop using the current velocity and depth information. This nonlinear model predictive controller for visual servoing explicitly optimizes the positioning performance of the robot and simultaneously considers various constraints such as joint limits, singularities, and nonholonomic constraints. The cost function (Camacho and Bordons, 2007) is given by:

$$C(k)=\sum_{i=1}^{H_p}\left\|\begin{bmatrix}\hat{u}(k+i\,|\,k)-u_d\\ \hat{v}(k+i\,|\,k)-v_d\end{bmatrix}\right\|_{Q(i)}^2+\sum_{i=0}^{H_u-1}\left\|\begin{bmatrix}\Delta\hat{V}(k+i\,|\,k)\\ \Delta\hat{\omega}(k+i\,|\,k)\end{bmatrix}\right\|_{R(i)}^2 \tag{4.37}$$

where H_p and H_u are the prediction horizon and the control horizon, respectively. Also, (u, v) is the desired system output, which is the position of the feature point in the image plane, and *Q(i)* and *R(i)* are the weighting matrices (Maciejowski, 2000). Moreover, the visibility and joint constraints are taken into account in the ANMPC controller.

In Figure 4.24, the ANMPC controller includes four units: Model Updating Unit, Linearized Model, Constraint Unit, and Optimizer Unit. In particular, the current position of the target object in the image (i.e., $[u,v]^T$) and the current depth z are continuously measured with the CCD camera and the laser distance finder, and are sent into the model

updating unit. Then the linearized model of the plant is updated by the model updating unit with the latest u, v and z so that the model can always track the nonlinear plant. In addition, three kinds of constraints are considered and set up by the constraint unit which will constrain $[v, \omega]^T$ (the translational and rotational velocities of the robot), $[x, y, \theta]^T$ (the location and heading of the mobile robot in the environment), and $[u, v]^T$ (the position of the target object in the image). Finally, based on the latest linearized model, the constraint requirements and the current outputs of the plant, the optimizer unit will calculate an optimal control input sequence using the quadratic programming algorithm (QP), as described in (Camacho and Bordons, 2007).

Constraints

The ANMPC controller, as shown in Figure 4.24, considers three types of constraints, as follows:

$$u_{\min} \le u(t) \le u_{\max} \text{ and } v_{\min} \le v(t) \le v_{\max}$$

(Visibility Constraints) (4.38)

$$x(t) \le x_{\max},\ y(t) \le y_{\max},\ \theta_{\min} \le \theta(t) \le \theta_{\max}$$

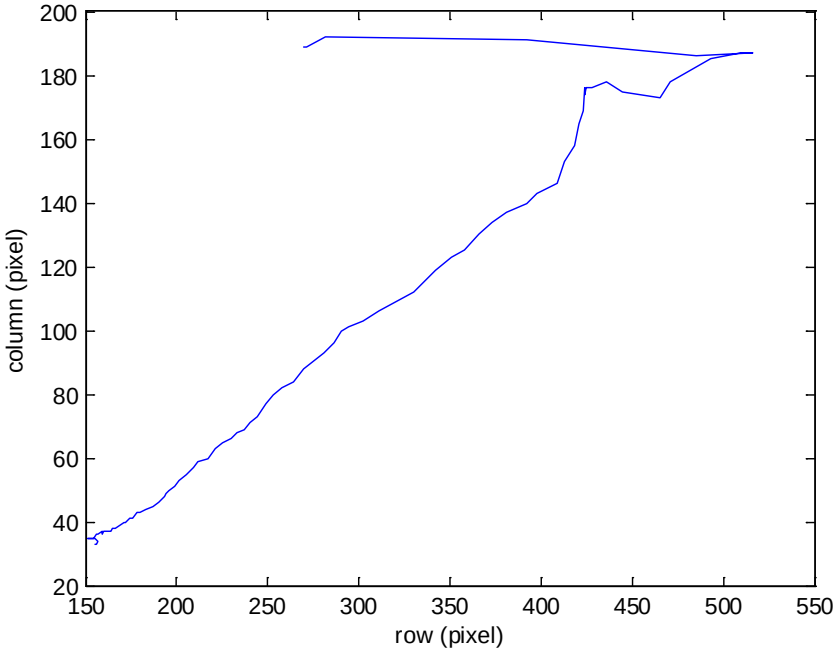
(Robot Location Constraints) (4.39)

$$v_l(t) \le v_{l\max} \text{ and } \omega(t) \le \omega_{\max}$$

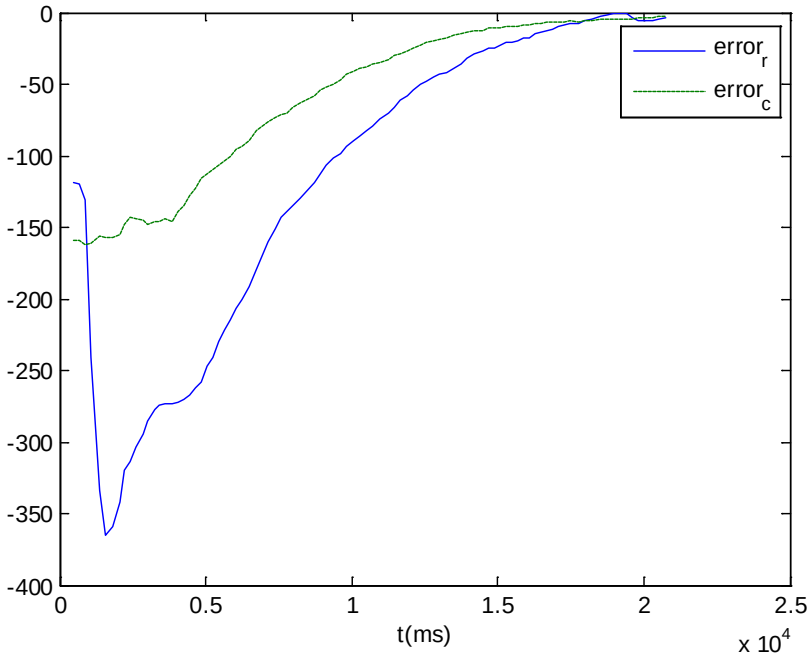
(Robot Velocity Constraints) (4.40)

Experimental Results

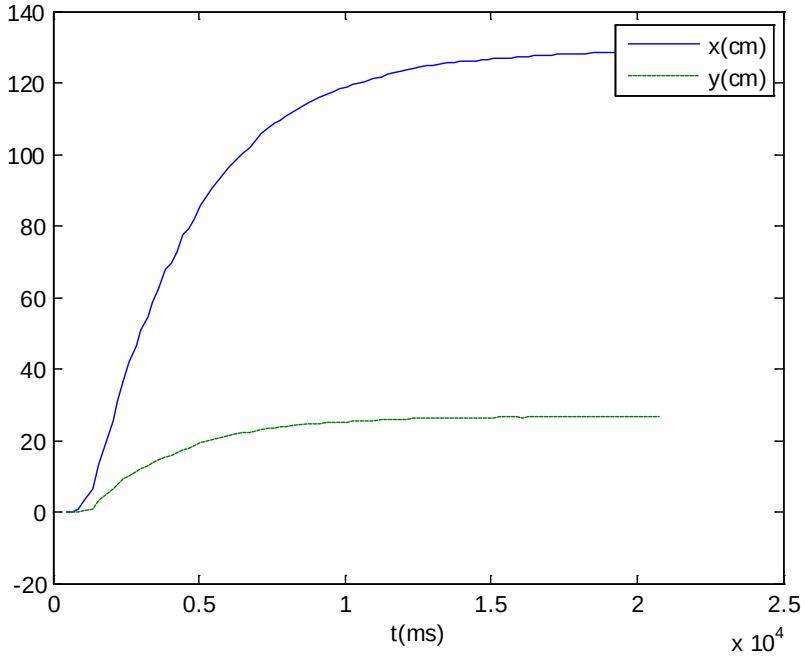
In the first experiment, an ANMPC controller was tested without considering any constraints. The controller parameters are: Sampling time $t_s = 100ms$, $H_p = 10$, $H_u = 3$, $Q(i) = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$ and $R(i) = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. The experimental results are presented in Figure 4.25.


column (pixel)
row (pixel)
200
180
160
140
120
100
80
60
40
20
150
200
250
300
350
400
450
500
550

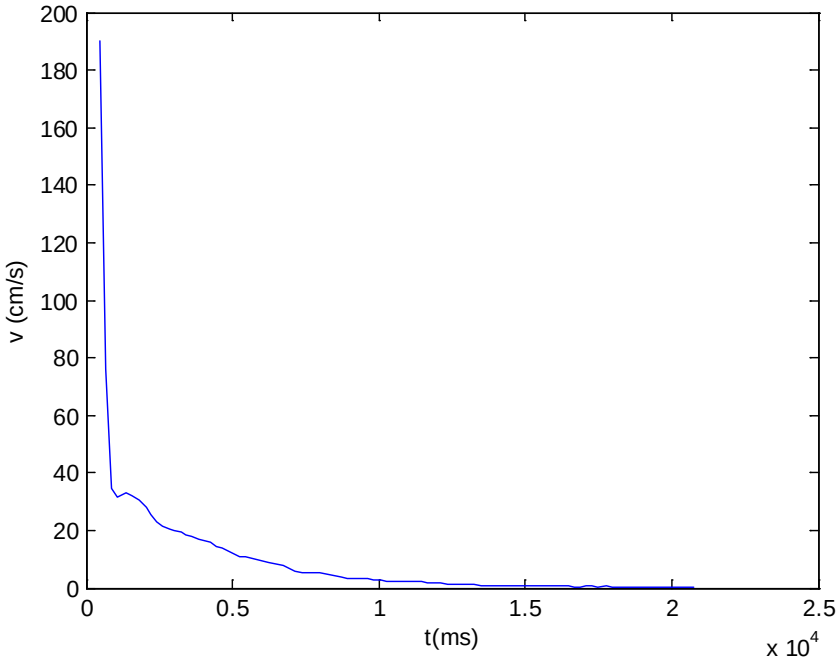
(a)


error$_r$
error$_c$
t(ms)
x 10^4
0
-50
-100
-150
-200
-250
-300
-350
-400
0
0.5
1
1.5
2
2.5

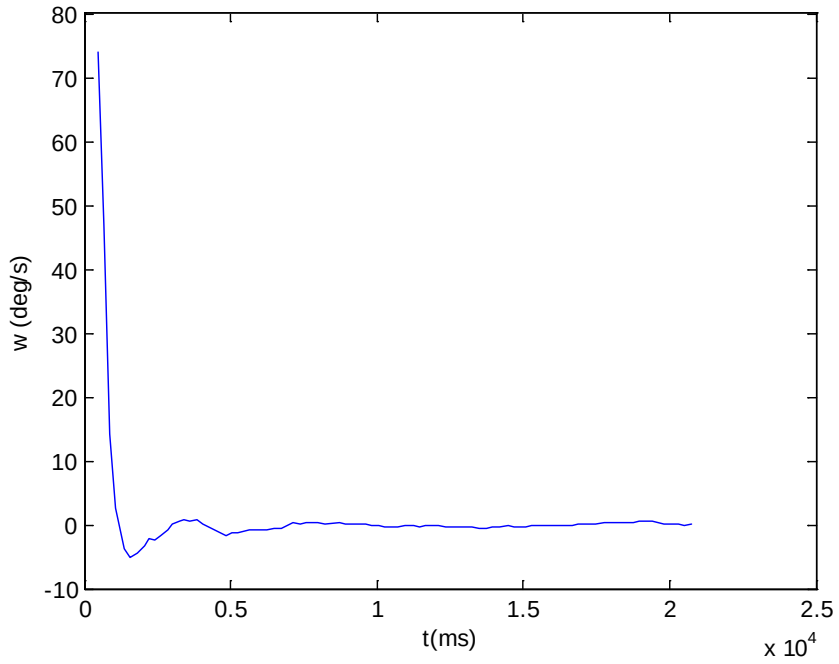
(b)


x(cm)
y(cm)
140
120
100
80
60
40
20
0
-20
0
0.5
1
1.5
2
2.5
t(ms)
x 10^4

(c)


200
180
160
140
120
100
80
60
40
20
0
v (cm/s)
0
0.5
1
1.5
2
2.5
t(ms)
x 10^4

(d)



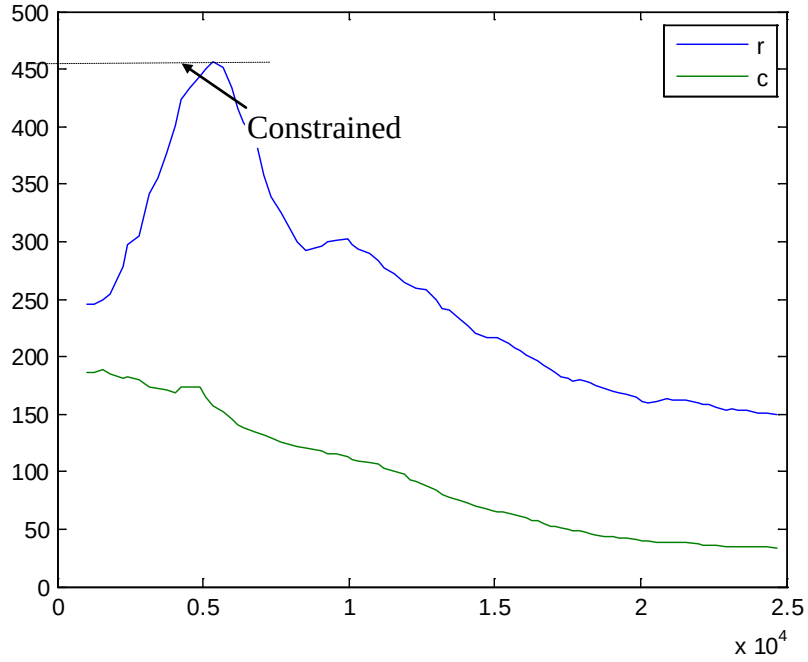
(e)

Figure 4.25: The experimental results of a mobile visual servo system using un-constrained ANMPC: (a) trajectory of the target object in the image; (b) pixel errors on the image plane; (c) history of the mobile robot location; (d) history of the robot translational velocity (control input); (e) history of the robot rotational velocity (control input).

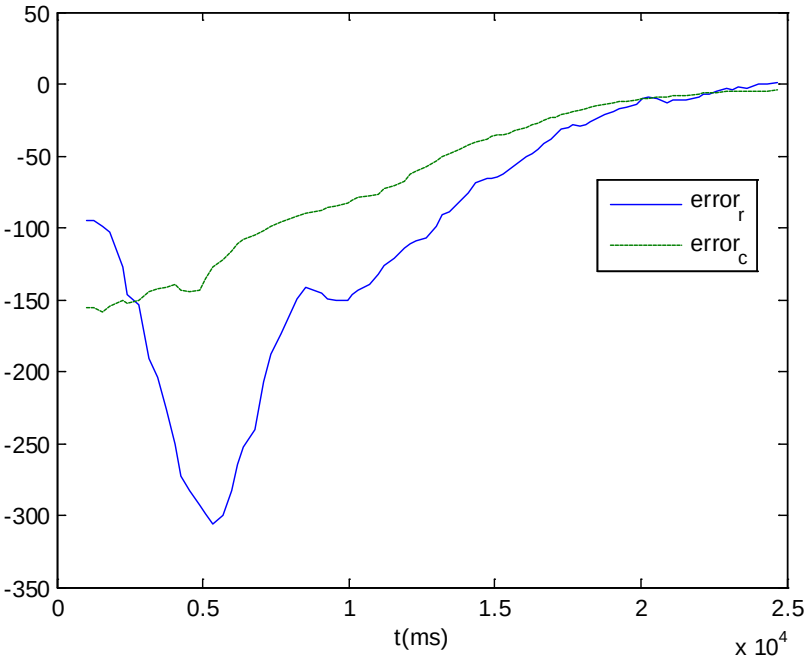
Constrained ANMPC Controller

In this experiment, the ANMPC has the following constraint parameters: $20 \le r \le 450$, $20 \le c \le 300$, -20 cm/s $\le v_l \le 20$ cm/s, -6 deg/s $\le \omega \le 6$ deg/s, $y \ge -15$ cm.

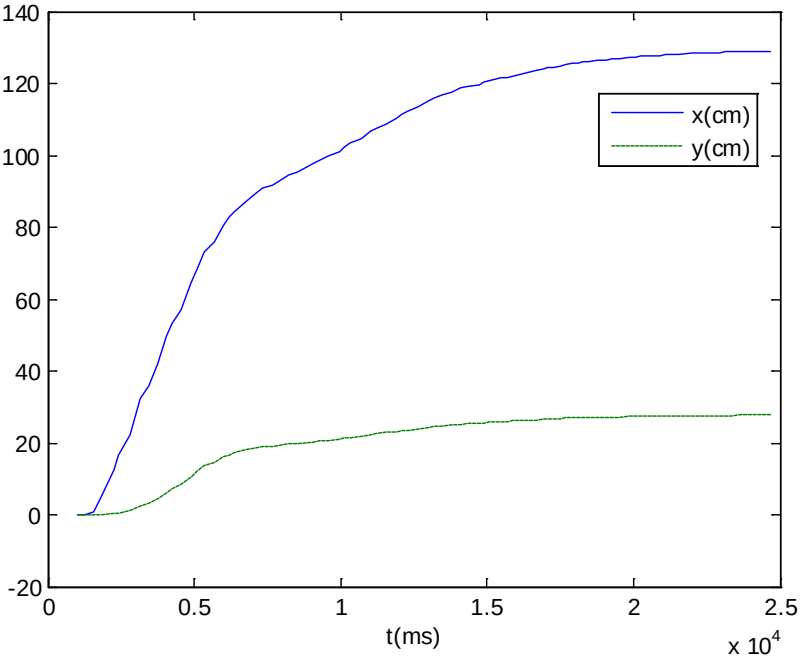
In the process of visual servoing, the system outputs and the control inputs are presented in Figure 4.26.



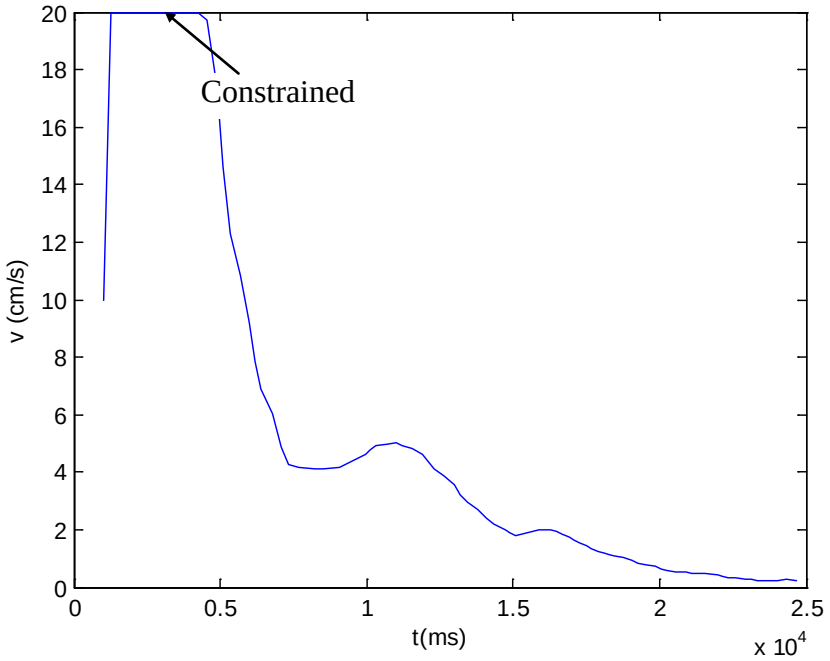
(a) System outputs (pixel position of the target object in the image).



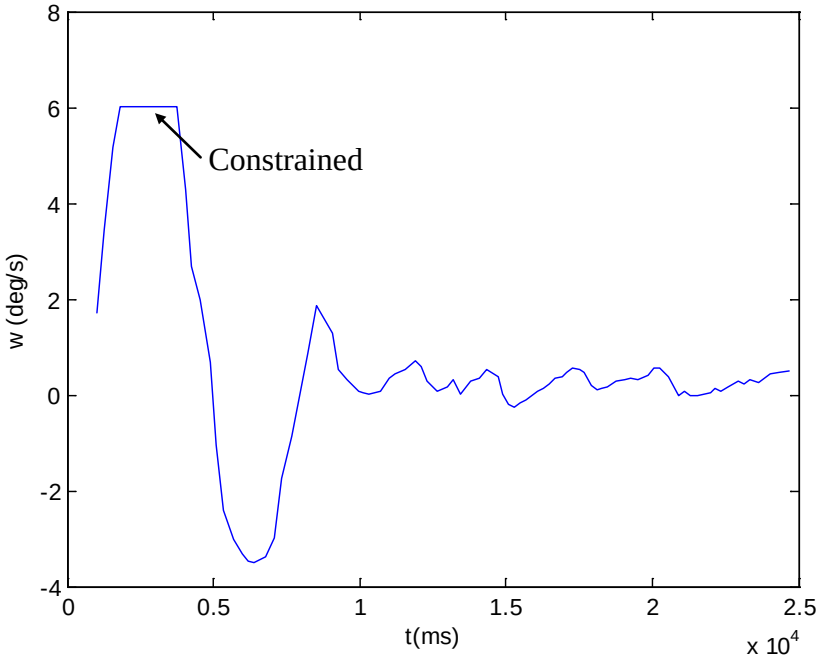
(b) Pixel errors on the image plane.



(c) History of the mobile robot location.



(d) History of the robot translational velocity (control input).

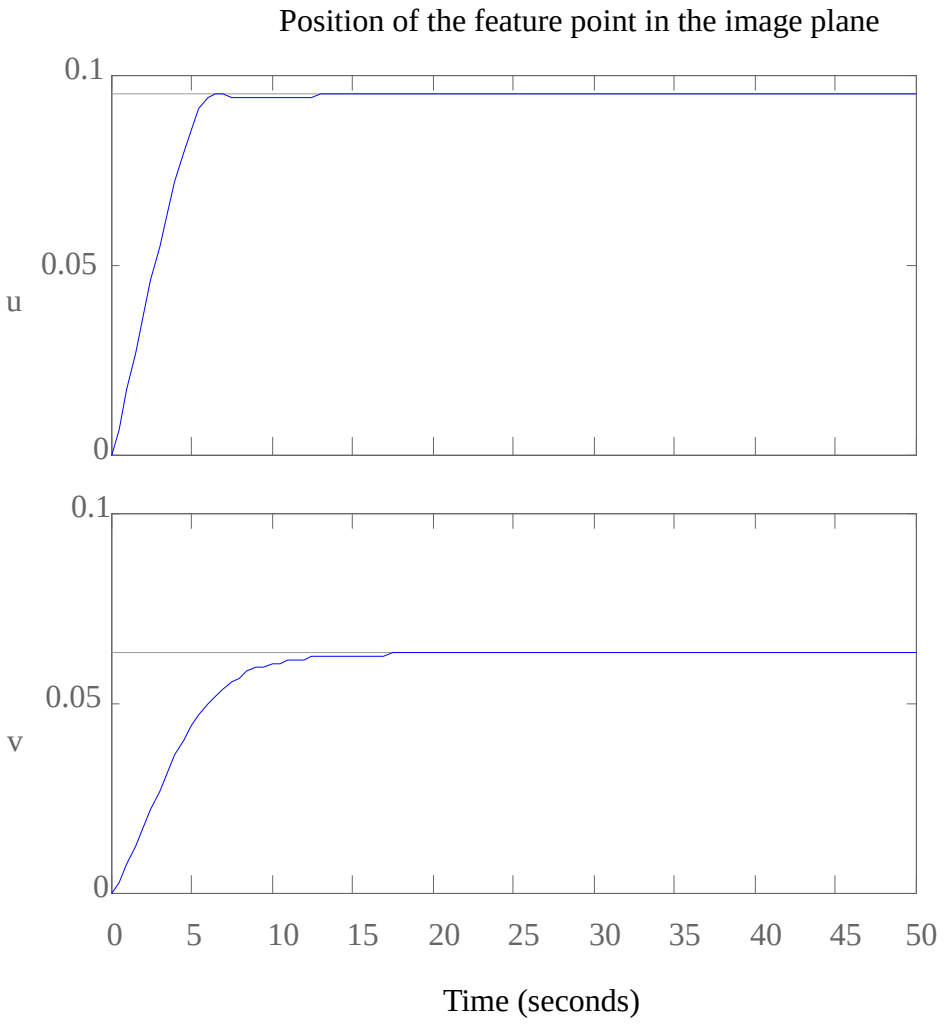


(e) History of the robot rotational velocity (control input).

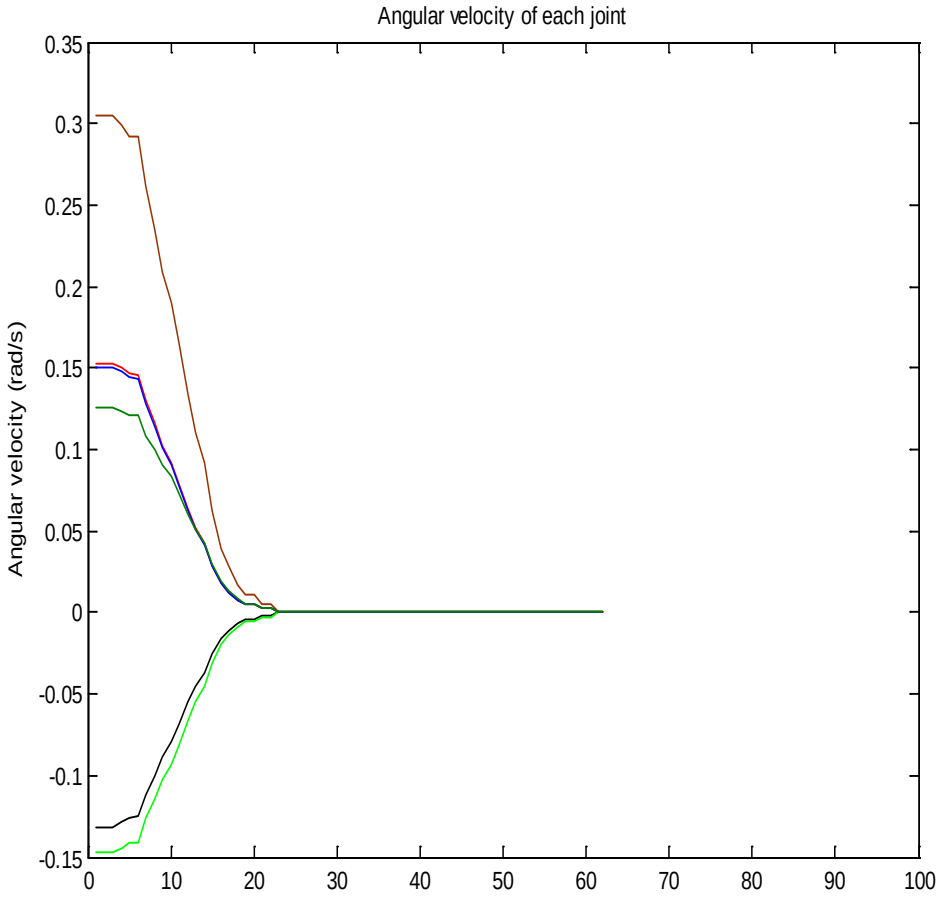
Figure 4.26: Mobile visual servoing using constrained ANMPC.

From Figure 4.26(a), 4.26(d) and 4.26(e), it is clear that the system outputs ($r(t)$ and $r(t)$) and the control inputs ($v_l(t)$ and $\omega(t)$) were constrained within the desired limits while Figure 4.26(b) showed that the errors quickly converged to zero. Therefore, from Figure 4.26, it can be concluded that the ANMPC visual servo controller developed in this thesis is quite successful in maintaining optimal control performance and simultaneously it respects various constraints (visibility constraints, velocity constraints, and so on).

Figure 4.27 shows the results of the ANMPC controller of the manipulator. Figure 4.27(a) shows the trajectory of the feature point in the image plane, and Figure 4.27(b) shows the controller outputs which are angular velocities of the six joints of the manipulator. According to the figures, it can be concluded that the ANMPC visual servo controller developed in this thesis performs satisfactorily for a robotic manipulator.



(a) System outputs (pixel position of the target object in the image).



(a) Angular velocities of the six joints.

Figure 4.27: Visual servoing of a robotic arm using ANMPC.

CHAPTER 5 Networked Sensing and Sensor Fusion

As mentioned in the previous chapter, a critical challenge in mobile manipulation is the accommodation of visibility constraints because if the camera does not see the object of interest, there would be no feedback signal in the manipulation control loop. This will lead to the failure of the entire system. In Chapter 5, a hybrid controller and an adaptive nonlinear model predictive controller (ANMPC) were proposed with the objective of keeping the object of interest inside the camera view, which can guarantee functioning of the manipulation system. However, in these methods the operational workspace is limited by the camera scene and the flexibility of the robot. Since the workspace within a camera scene is much smaller than the actual physical workspace of a robot, if the size of camera scene can be expanded to the entire robotic workspace, the flexibility and capability of the robotic manipulation system can be dramatically improved. This idea brings us into the areas of networked sensing and sensor fusion.

A multi-sensor system may be treated as a sensor network. It usually consists of different types of spatially distributed autonomous sensors such as those for temperature, acoustics, magnetic field, vision, and pressure. Each sensor is considered an individual node in the sensor network. Most of them are fixed and cannot move. Some sensors have mobility and can be moved to different locations according to the specific requirements of a sensing scenario. They are called dynamic sensors, and have the ability to collect data from the distributed work environment and forward them to a data processing center. Two possible applications of networked sensing are shown in Figure 5.1. There are two challenges in such a large-scale and complex system: 1) what information from the entire sensor data is useful; and 2) how can the system utilize the sensor data to make decisions accurately and reliably.

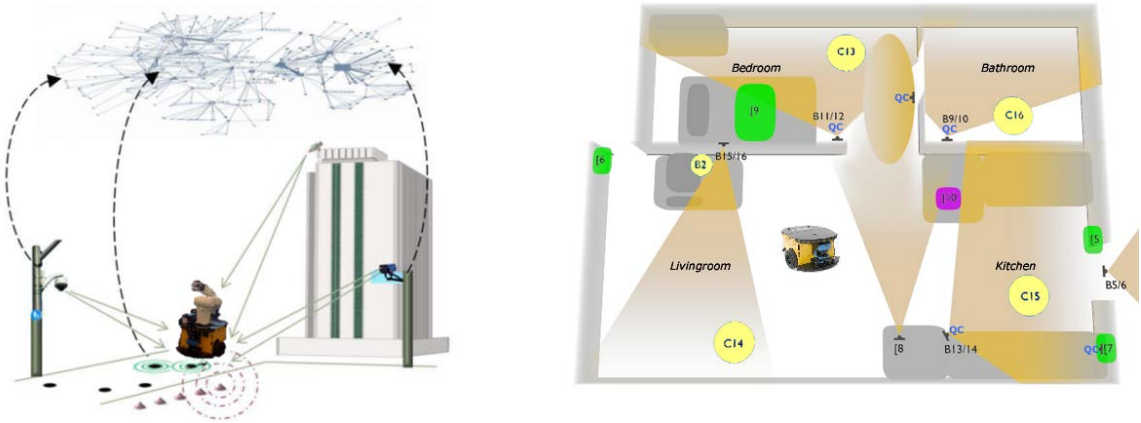


Figure 5.1: Possible application of a sensor network: (a) a future city; (b) a future home environment.

These two questions bring us to the problem of sensor fusion. There, multi-sensor systems are designed to exploit several signature-generation phenomena, gather different types of information about objects and scenes of interest, and “fuse” them to obtain more accurate and reliable information. Design of a multi-sensor system involves optimization of sensors, data processing, communication, and particularly the use of an appropriate fusion strategy. As described in the literature review on sensor fusion in Chapter 1, many methods such as neural networks and fuzzy logic have been applied in sensor fusion. However, they are only suitable for simple applications because appropriate training is difficult in complex problems.

In this chapter, Probabilistic Finite State Automata (PFAS) is proposed for application in sensor feature representation and fusion. It introduces a self-organized sensor network where PFSA is utilized to organize and cluster suitable sensors and then fuse their data to make reliable and more accurate decisions. It has the ability to communicate with the robots in the workspace and provide information to assist the execution of the robotic tasks.

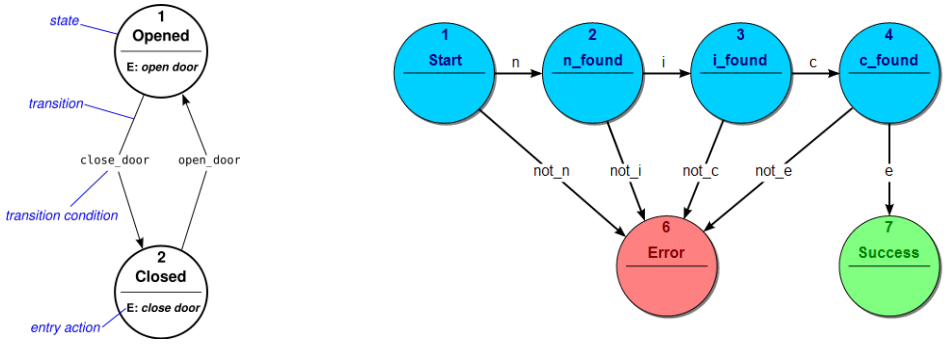
5.1 Definitions

5.1.1 Formal Languages

Formal language is a technical term used in mathematics and computer science. A formal language (L) over an alphabet Σ is a set of words, which is denoted by Σ^*; and an empty word is denoted by λ ($L = \Sigma^*; \lambda = \emptyset$).

5.1.2 Finite State Machine

A Finite State Machine (FSM) or Finite State Automaton (FSA) is an abstract model of a machine that composes a finite number of states, transitions between those states, and actions. Figure 5.2 (a) shows an example of an FSM which has 2 states, 2 transitions, and 2 actions. Figure 5.2 (b) shows a more complex example. The transition indicates a state change; and is described by a condition that would need to be fulfilled to enable the transition. The action is a description of an activity that is to be performed at a given moment. Since there is only one possible transition for any input, it is called a Deterministic Finite State Machine (DFSM) or Deterministic Finite State Automata (DFSA).



a. Elevator b. Parsing the word "Nice"

Figure 5.2: An example of finite state machine.

The mathematical description of a DFSA is composed of $\Sigma, S, s_0, \delta, F$, where: Σ is the input alphabet; S is a finite, non-empty set of states; s_0 is an initial state; δ is the state transition function; and F is the set of final state.

5.1.3 Probabilistic Finite State Automata

Non-deterministic Finite State Automata (NDFSA) or Probabilistic Finite State Automata (PFSA) is different from DFSA in the transition where PFSA has several possible next steps for an input. PFSA is generalized by probability automata (PA) which assigns a probability to each state transition.

A PFSA can be defined by $Q, \Sigma, \delta, q_0, F$, where Q is a finite set of states; Σ is a finite set of input symbols; δ is a transition function; and F is a set of final states. It has two with two probabilities: the probability P that a particular state transition is triggered; and the probability that the initial state q_0 is replaced by a stochastic vector.

In the formal language theory (Hopcrof, et al., 2005), an alphabet Σ is a (non-empty finite) set of symbols. A string x over Σ is a finite-length sequence of symbols in Σ. The length of a string x, denoted by $|x|$, represents the number of symbols in x. The Kleene closure of Σ, denoted by Σ^*, is the set of all finite-length strings of events including the null string ϵ, where $|\epsilon| = 0$. A string of length $d \in \mathbb{N}$ is denoted by $\Sigma^d \subset \Sigma^*$.

A probabilistic finite state automaton is a tuple $G = (Q, \Sigma, \delta, q^0, \tilde{\pi})$, where

- Q is a (nonempty) finite set, called set of states;
- Σ is a (nonempty) finite set, called input alphabet;
- $\delta: Q \times \Sigma \to Q$ is the state transition function;
- $q^0 \in Q$ is the start state;
- $\tilde{\pi}: Q \times \Sigma \to [0, 1]$ is an output mapping which is known as the probability morph function and satisfies the condition $\sum_{\tau \in \Sigma} \tilde{\pi}(q_j, \tau) = 1$ for all $q_i \in Q$. The morph function π has a matrix representation $\tilde{\Pi}$, called the morph (probability) matrix $\tilde{\Pi}_{ij} = \pi(q_i, \sigma_j), \forall q_i \in Q$ and $\forall \sigma_j \in \Sigma$. Note that $\tilde{\Pi}$ is a ($|Q| \times |\Sigma|$) stochastic matrix; i.e., each element of $\tilde{\Pi}$ is non-negative and each row sum of $\tilde{\Pi}$ is equal to 1.

The transition map δ naturally induces an extended transition function $\delta^*: Q \times \Sigma^* \to Q$ such that $\delta^*(q, \epsilon) = q$ and $\delta^*(q, \omega\tau) = \delta(\delta^*(q, \omega), \tau)$ for $q \in Q$, $\omega \in \Sigma^*$ and $\tau \in \Sigma$. In this thesis, it is assumed that all states in a PFSA are reachable from the start state. Otherwise, the non-reachable states should be removed from Q.

5.1.4 Cross Machine

A cross machine is a 6-tuple $(Q, \Sigma_i, \Sigma_o, \delta, q_0, \Psi)$, where

- Q is a (nonempty) finite set, called set of states;
- Σ_i is a (nonempty) finite set, called input alphabet;
- Σ_o is a (nonempty) finite set, called output alphabet;
- δ: $Q \times \Sigma_i \to Q$ is the state transition function;
- $q_0 \in Q$ is the start state;
- ψ: $Q \times \Sigma_o \to [0,1]$ is an output mapping, which is known as the output morph function and satisfies the condition $\Sigma_{\sigma \in \Sigma_o} \psi(q_j, \sigma) = 1$ for all $q_j \in Q$. The output morph function ψ has a matrix representation Ψ, called the output morph (probability) matrix.

Cross machine can be considered as a transfer function (Figure 5.3), which takes PFSA as inputs and provides stochastic languages as outputs. Figure 5.4 shows an example of an input-output system. The cross machine models the binary symmetric channel, and the input machine is a D-Markov machine of order 1.

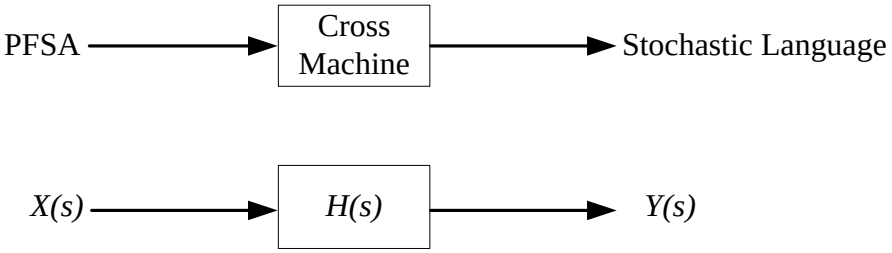


Figure 5.3: Analogy between a cross machine and a transfer function.

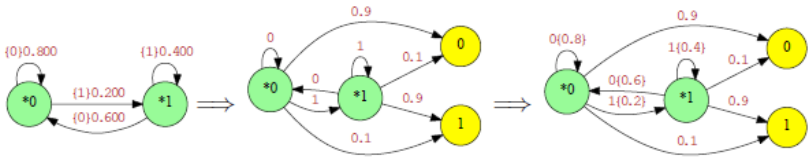


Figure 5.4: An example of a system.

5.2 Fusion-driven Sensor Network

Since the local sensors of a mobile robot may not be adequate to effectively carry out the necessary robotic tasks, a dynamic sensor network is proposed in this thesis. This will facilitate detection, identification, and tracking of objects of interest in the robotic workspace and subsequent robotic grasping, manipulation, and processing as well. The network will incorporate heterogeneous sensing elements. Both static and dynamic

sensors will be present. The sensor network will be adaptive in the sense that the structure in which the sensors are connected, the sensor locations, and data pathways all may be adjustable during robotic operation. This adaptation will require feedback of requests from the robot depending on the nature and requirements of the task and the necessary information to effectively carry out the task. In order to meet these needs, a new network structure is proposed, as shown in Figure 5.5. During operation, a robot may need to search for an object of interest that is not within its local sensing range. Then, the robot will send a "help" message to the sensor network, requesting the location of the object. As the sensor network receives the request from the robot, it will adapt to the identification and tracking requirements to reconfigure the nodes of the sensors dynamically concerning different types of targets.

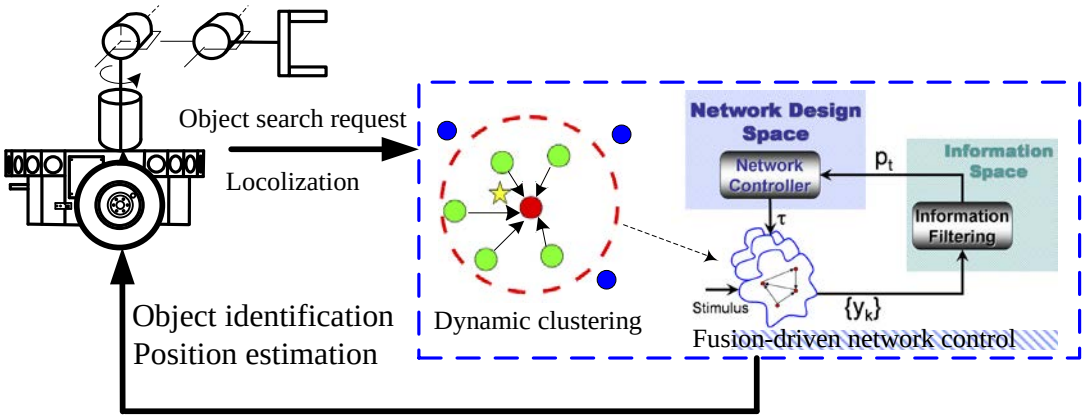


Figure 5.5: Overall network architecture.

The dynamic clustering and fusion are done in response to from the robot with regard to some objects of interest in the environment. This will cause statistical changes and changes to the spatial-temporal representation of the sensor data. Finally the system will dynamically reallocate sensing and network resources as appropriate. In summary, the system has the following features:

- Heterogeneous dynamic space-time clustering (DSTC) protocol
- Sensor node mobility
- Object detection and classification of composite patterns
- Feedback of information requests from the robot

5.3 Design and Implementation of a Fusion-Driven Sensor Network

5.3.1 Mobile Target Tracking Application

Figure 5.6 shows the architecture of a sensor network designed for mobile target tracking. A sensor is defined as a mapping $S: \mathbb{P} \times \mathbb{D} \times \mathbb{T} \rightarrow \mathbb{V}$ where $\mathbb{P}$ is the set of sensing modalities (e.g., acoustic energy); $\mathbb{D}$ is the network design parameter space consisting of design and operational parameters for sensors (e.g., sensing and communication range, sampling frequency, communication bandwidth); $\mathbb{T}$ represents the time domain; and $\mathbb{V}$ is a vector space over the real field $\mathbb{R}$. Let $\mathbb{S}$ be the set of all sensors in the network. A space time neighborhood η centered on the spatial-temporal point (x_0, t_0) is defined as $\eta(x_0, t_0; \delta_x, \delta_t) \triangleq \{(x, t): \|x - x_0\| < \delta_x \; and \; |t - t_0| < \delta_t\}$, where the spatial radius $\delta_x > 0$ and the time interval $\delta_t > 0$ may vary with (x_0, t_0). A hotspot $\mathcal{H}_s$ for a sensor $S \epsilon \mathbb{S}$ at time $t_0 \epsilon \mathbb{T}$, associated with a stimulus e, is defined as the space time neighborhood: $\mathcal{H}_s(e, t_0) \triangleq \eta_e(x_S, t_0; \delta_x, \delta_t)$ of the sensor where δ_x and δ_t are the sensing radius and sensing time interval of the sensor S at location x_S and time t_0, respectively. The hotspot for the sensor set $\mathbb{S}$ at time epoch $t_0 \epsilon \mathbb{T}$, associated with a stimulus e, is defined to be the set of sensors that have non-empty space-time neighborhoods: $\mathcal{H}_s(e, t_0) \triangleq \{S \epsilon \mathbb{S}: \mathcal{H}_s(e, t_0 \neq \emptyset)$. A collection of clusters is called a cluster bank. The Network Design Space is represented by $\mathbb{D} \times \mathbb{C}$, where $\mathbb{C}$ is the set of connectivity vectors of all sensor nodes, and $\mathbb{D}$ is the network design parameter space as described before. The topology of the sensor network is defined to be the set $\Im(t) \triangleq \{x, d, C_t(S): \exists S \epsilon \mathbb{S}$, with $d \; \epsilon \; \mathbb{D}$ and connectivity vector $C_t(S))$ at location x at time $t \epsilon \mathbb{T}\}$. Therefore, $\Im(t)$ contains all variables that a network controller may need to manipulate for adaptation to changes in the information space.

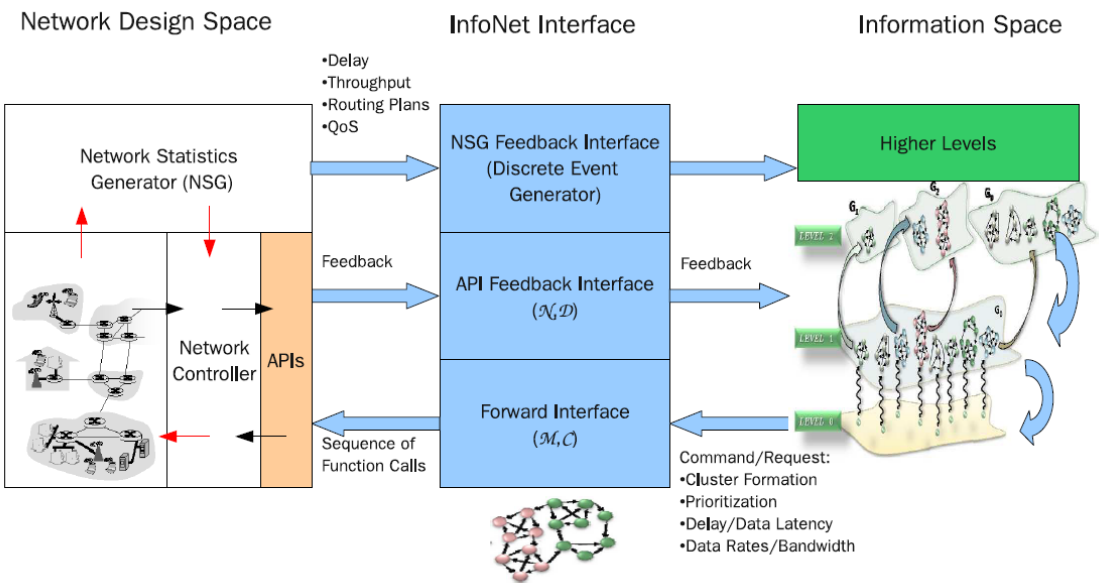


Figure 5.6: Design architecture of a sensor network.

The Network Design Space is configured to adapt to the needs of the Information Space. Reconfiguration of a large sensor network may be achieved through various means such as adaptive sampling of an individual sensor, sensor mobility, turning on/off the existing sensors, bandwidth reallocation, protocol modification, and resource reallocation. Adaptive parameters of the Network Design Space include sensor position, resource assignments, and network connectivity.

The Information Space is mainly used to derive the spatial-temporal statistics of sensor data. Since the system is distributed, each sensor addresses the data compression and communication constraints by autonomously generating the PFSA interpretation of data. The constructed PFSAs are compared against a library of pre-defined patterns of interest, allowing the sensor network to selectively track interesting targets among many candidates. The PFSAs constructed by individual sensors are fused at the cluster head. The cluster is dynamically formed by grouping the nodes (sensors) that observe the same pattern.

The sensor set is denoted by $\mathfrak{F}$ which represents the set of all sensors; G is defined as a pattern of a PFSA; and $\mathcal{H}$ is defined to map each sensor $s\epsilon\mathfrak{F}$ to the PFSA from the sensor data stream. Among all identified patterns, a set G of K patterns of interest, $\mathbb{G} = \{G^i : i = 1, \cdots, K\}, K\epsilon\mathbb{N}$ is defined. The set of patterns of interest $\mathbb{G}$ is completely

ordered via the pattern characteristic function $x_{\mathbb{G}}: \mathbb{G} \rightarrow [0,1]$. A sensor data segment is mapped into the pattern of interest G.

The InforNet Interface functions the interaction between the Information Space and the Network Design Space. It consists of forward and feedback interfaces as shown in Figure 5.6. The forward interface translates the Information Space requirements to enable the network to act as an actuator to best meet the fusion requirements such as dynamically changing the sensor clusters and network topology. Once the network finishes executing the command from the Information Space, feedback information will be sent back to the Information Space through the InfoNet Interface to show the results from the reconfiguration of the network. It is seen that the Information Space can selectively cluster the available sensors in order to track different objects. This mechanism has high scalability and robustness for different objectives.

5.3.2 An Experiment of a Pressure Sensor Field

In this experiment, a pressure sensitive floor (Figure 5.7) is simulated, which consists of an array of piezoelectric wires that serve as distributed pressure sensors. A coil of piezoelectric wire is placed under the square floor tiles of size 0.65 m $\times$ 0.65 m such that each sensor generates an analog voltage due to the pressure applied on it. The output voltage is in the range of 0 to 1023. A total of 144 sensors are placed in a 9×16 equidistant grid to cover the entire workspace. The objective of this experiment is to identify two different moving objects which are simulated by two different types of mobile robots: 1). Pioneer mobile robot, and 2) Segway RMP robot (Figure 5.8).

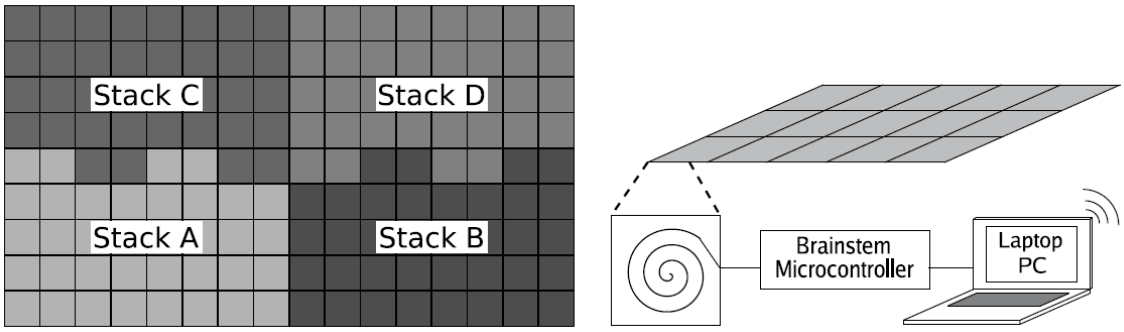


Figure 5.7: Pressure sensor field.

(a) (b)

Figure 5.8: The experimental setup: (a) Pioneer mobile robot; (b) Segway RMP robot.

In this work (Figure 5.8(b)), a Segway RMP robot is made to move in the sensor field. The sensor network (pressure sensor field) detects and tracks spatial-temporal events of the behavior patterns of a Segway RMP. A Segway RMP moving in different types of motion trajectories is considered for illustration of detecting and tracking of various behavior patterns.

Each sensor addresses the issues of data compression and communication constraints by autonomously aggregating the data through symbolization and semantic construction of probabilistic finite state automata (PFSA). Figure 5.9 shows the procedure of PFAS feature generation. As the sensors receive time series data, the data will be divided into four sub-regions. Next, a feature vector $V = (V_1, V_2, V_3, V_4)^T$ is generated, which represents the signature of the acquired signal. Finally, the feature vector is utilized to generate a PFAS pattern.

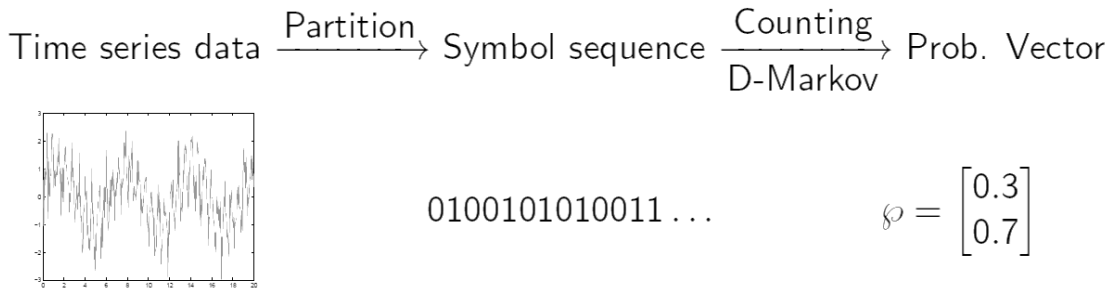


Figure 5.9: Basic modeling procedure of sensory data.

The constructed PFSA are compared against a library of pre-determined patterns of interest with an appropriate metric, thus allowing the sensor network to selectively track interesting targets amongst many candidate targets. The observed PFSA which are matched the PFSA from the library are clustered dynamically, with additional network and physical requirements. Figure 5.10 shows the logic of sensor clustering.

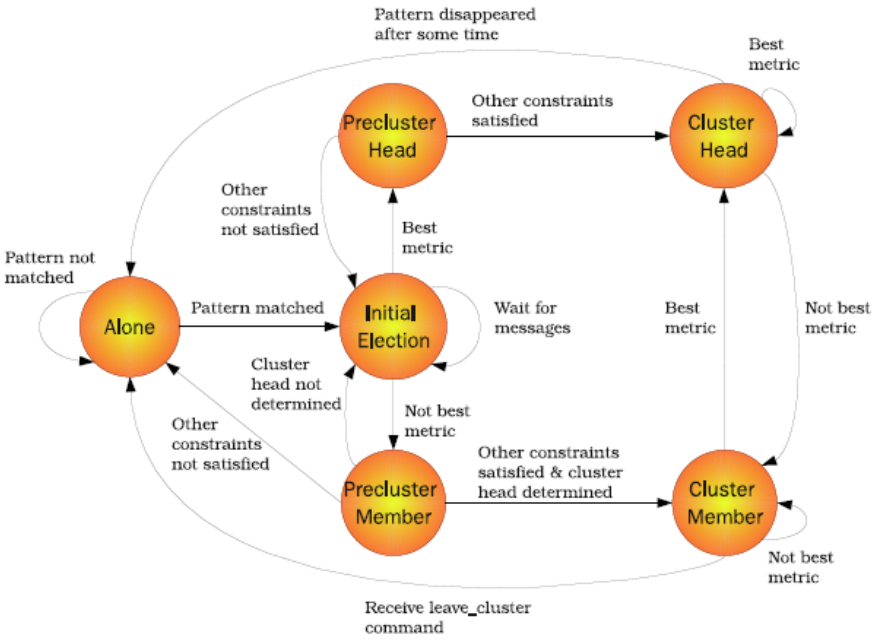


Figure 5.10: Dynamic space-time clustering.

Suppose that d is the distance of the sensor s from the target location, G is the observed pattern of interest at its origination point, and $H_t(s)$ is constructed by PFSA from the data of the sensors. It yields $d = D\big(\theta(H_t(s), G)\big)$, where θ is a set of PFSA which measure the deterioration of signal from its origination to the location of the sensor. Also, $P[s, G]$ is defined the probability that the sensor has observed the pattern of interest G.

Therefore, if G^k is a pattern of interest and ϵ_i is the detection region, the probability of using sensor i in the work environment is given by

$$P[i, G^k; \epsilon_i] = \begin{cases} P[i, G^k] & if\ D\Big(\theta(H_t(s), G^k)\Big) \le \epsilon_i \\ 0 & otherwise \end{cases}. \qquad (5.1)$$

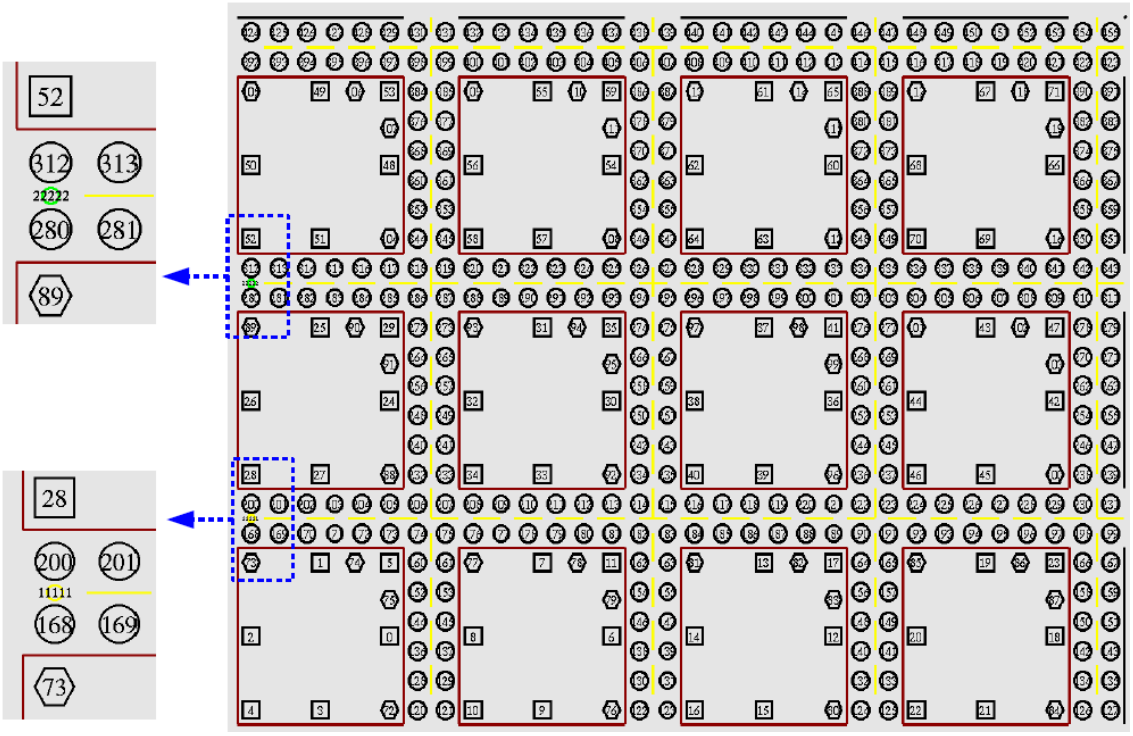
The estimated physical location of a sensed pattern of interest G^k is estimated by the cluster as follows:

$$\rho^{*}_{k,C_t^l(G^k)}(t) = \sum_{i \in C_t^l(G^k)} \Gamma(i) \cdot P[i, G^k; \epsilon_i] \quad (5.2)$$

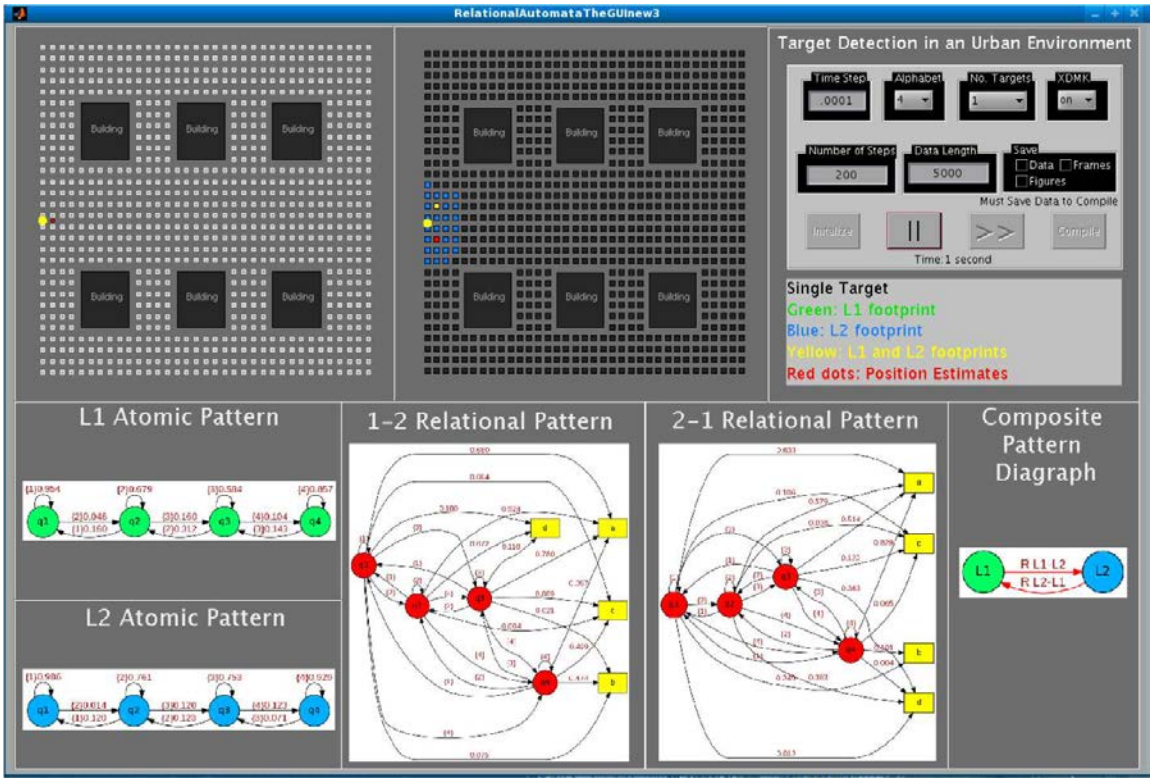
where $\Gamma(i)$ is the physical location of sensor i.

5.3.3 Heterogeneous Sensor Field

In order to extend the work from a network of single sensor type to a heterogeneous sensor field, we now develop a simulation environment, which is shown in Figure 5.11(a). In this simulation environment, different shapes represent different types of sensors. The purpose of this simulation is to identify a moving object in the scene. It is able to track the motion of the object while it is moving. Figure 5.11 (b) shows the simulation platform which has been developed in Matlab. The top left window shows the real trajectory of the moving object in the workspace. The top middle window shows the active sensors in the sensor network which are indicated in color. The red one is the head of the sensor cluster, and the blue dot is the moving object. The four windows in the bottom show the detected PFSA pattern of the object.



(a)



(b)

Figure 5.11: (a) Simulation of a heterogeneous sensor field; (b) Developed Matlab simulation environment.

In the simulation, the moving objects are made to move along a horizontal line. Figure 5.12 shows the performance of the system when it tracks one target and two targets.

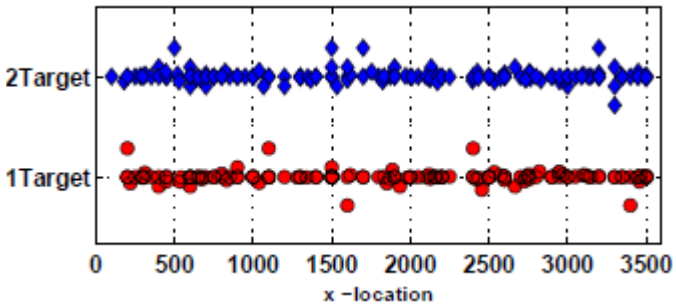
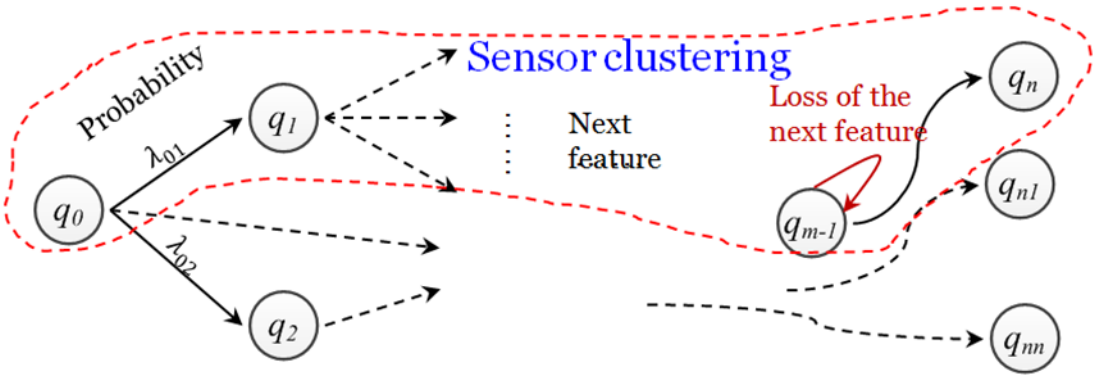


Figure 5.12: Simulation results.

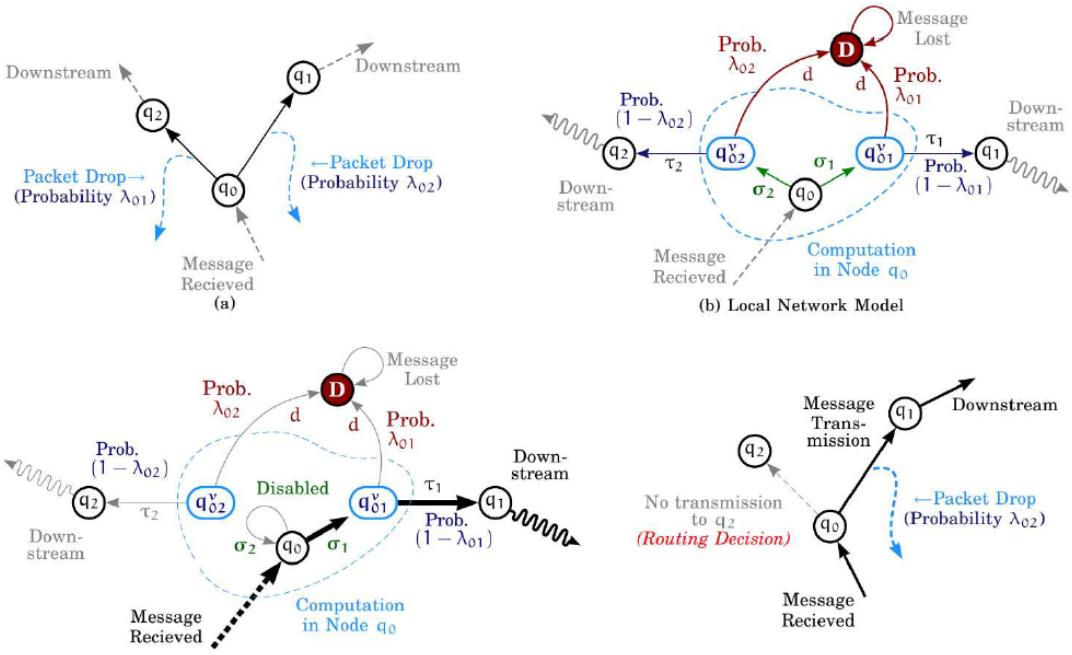
5.3.4 Modeling the Sensor Network as PFAS

A sensor network operates in an infrastructure of sensing, computation, and communication, through which it perceives the evolution of a physical dynamic process in its environment. It is data-centric (Tubaishat, M. and Madria, 2003) because data from the sensor nodes are of primary importance for this application. A common characteristic of information fusion in sensor networks is that data streamed from different sources are brought together through a communication network. Therefore, the following factors should be taken into account. Typically, there are thousands of sensor nodes in a wireless sensor network. It is almost impossible to have a global centralized fusion center. In other words, the data fusion scheme has to be designed and implemented in a distributed fashion for wireless sensor networks. It is impossible to broadcast all sensor data in the network because of the limitation of sensor power and bandwidth of the network. Therefore, selected use of sensors for different sensing purpose is necessary.

Figure 5.13(a) and (b) show the idea of dynamic clustering and organizing of sensors as a sensor network for a specific sensing purpose, which is represented by PFSA. In Figure 5.13(a), each node represents a type of sensor, which can also be considered as a representation of PFSA features of a type of sensor. As one type of sensor (q_0) is activated to reason the possible object that is tracked, it will keep searching for the corresponding PFSA features of other types of sensors(q_1, q_2). As the searching proceeds all the way from q_0 to q_n, it gathers all possible PFSA features of the tracked object. This mechanism gathers the neighboring sensors to be clustered as a sensor network for the purpose of identifying and tracking a detected object of interest in the workspace, as shown by the red dash line in Figure 5.13(a).



(a)



(b)

Figure 5.13: (a) Dynamic clustering mechanism; (b) network clustering.

CHAPTER 6 Physical Implementation and Experimentation

In this chapter, the techniques developed in the previous chapters such as machine vision, mobile robot navigation with Q-learning, robot motion control, and networked sensing are integrated and implemented in a physical mobile manipulation system in laboratory. This system contains a mobile robot platform and a robotic manipulator arm mounted on it, which has been developed and tested in the Industrial Automation Laboratory (IAL) of the University of British Columbia. In Section 6.1, the overview of the overall robotic system and the objective of the present experimentation are introduced. The test environment is demonstrated in order to simulate the search and retrieval scenario. Section 6.2 focuses on the hardware and software configuration of the system and development of the test bed, which consists of the mobile manipulation system. The sensors that are utilized and mounted on the robot, are discussed as well. Section 6.3 discusses the experimentation with the mobile manipulation system, which demonstrates that the developed system has the ability of identifying objects, navigation, and grasping an object of interest.

6.1 Overview

Physical experimentation is carried out in a laboratory environment to evaluate the performance of the methodologies and the robotic system that have been in the present thesis. Figure 6.1 gives an overview of the overall experimental procedure. The test environment contains a mobile robot platform and a manipulator arm mounted on it, several obstacles, and an object of interest. In this experiment, the robot navigates and finds the object of interest in the workspace. Then it moves close to the object. Finally, it grasp the object by using the on-board manipulator.

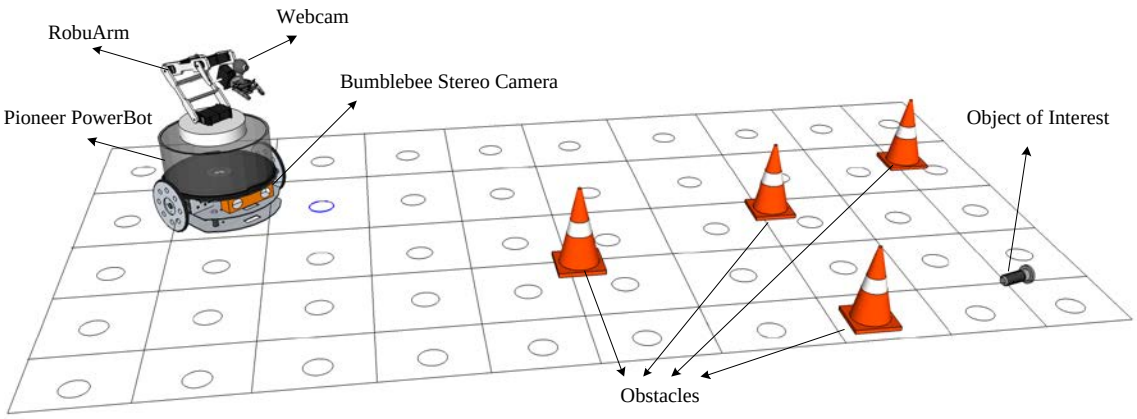


Figure 6.1: Overview of the experiment.

6.2 Test Bed

Figure 6.2 shows the physical configuration of the mobile manipulation system. It contains a Powerbot which is the mobile base, a RubuArm (the manipulator), and different types of sensors (web camera, laser, sonar, stereo camera).

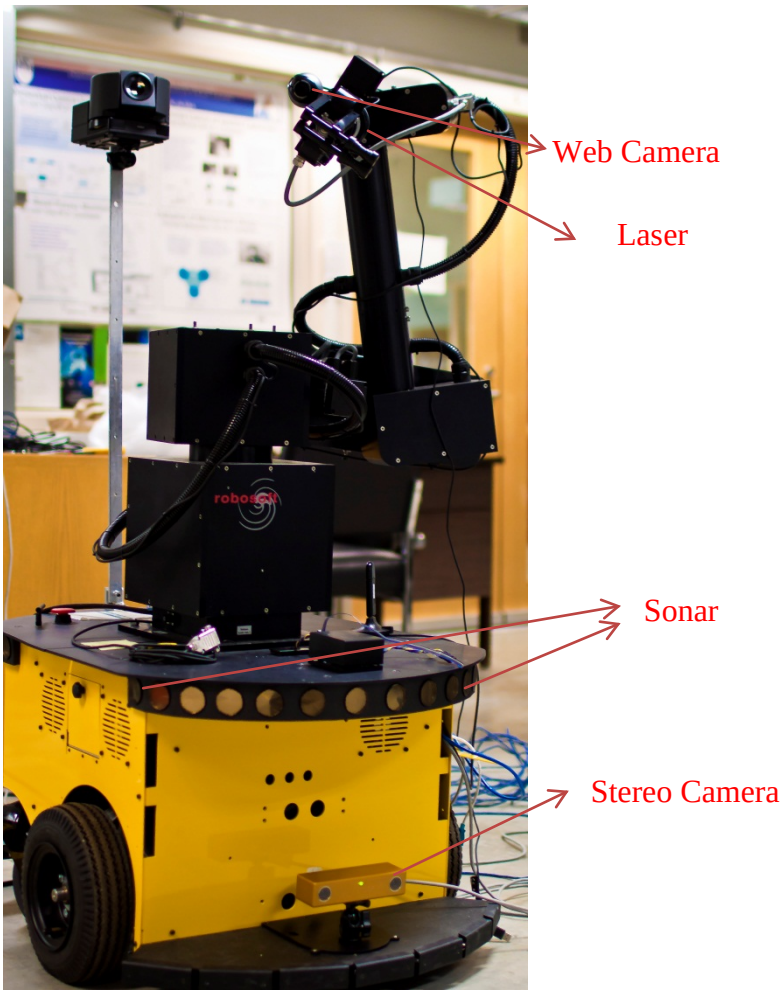


Figure 6.2: Physical configuration of the test bed.

6.2.1 Pioneer Powerbot

Figure 6.3 shows the original Powerbot, which is a high-payload differential-drive robot developed by Adept Mobile Robots Company. It is able to move at a speed of up to 1.6 m/s with a payload of up to 100 kg. Hence it is an ideal platform for laboratory and research tasks involving delivery, navigation, and manipulation. The Powerbot is equipped with a gyro sensor and 2 groups of sonars. The combination of readings from the gyro and the encoders is able to provide the position and orientation of the mobile robot. There are 32 sonar cones embedded in the mobile base and they can measure the distance between the robot and the neighborhood obstacles.

Figure 6.3: Poineer Powerbot—the mobile base.

The standard Service Information Packet (SIP), which is utilized to control the robot, is shown in Figure 6.4. It is sent on a constant cycle, and the reception of this SIP triggers a new iteration of ArRobot's *synchronized task processing cycle*. This cycle consists of a series of synchronized tasks, including SIP packet handling, invocation of sensor interpretation tasks, action handling and resolution, state reflection, and invocation of user tasks, in that order. The task cycle is (normally) triggered by the reception of a SIP (unless the robot platform fails to send SIPs or the task cycle is explicitly disassociated from the robot connection—see below). Each task will be invoked in a predictable order, and will have the most recent data to act upon. Hence, no task will miss the opportunity to use a SIP, and as long as the tasks do not take too much time to execute, each SIP is handled as soon as possible once the robot sends it.

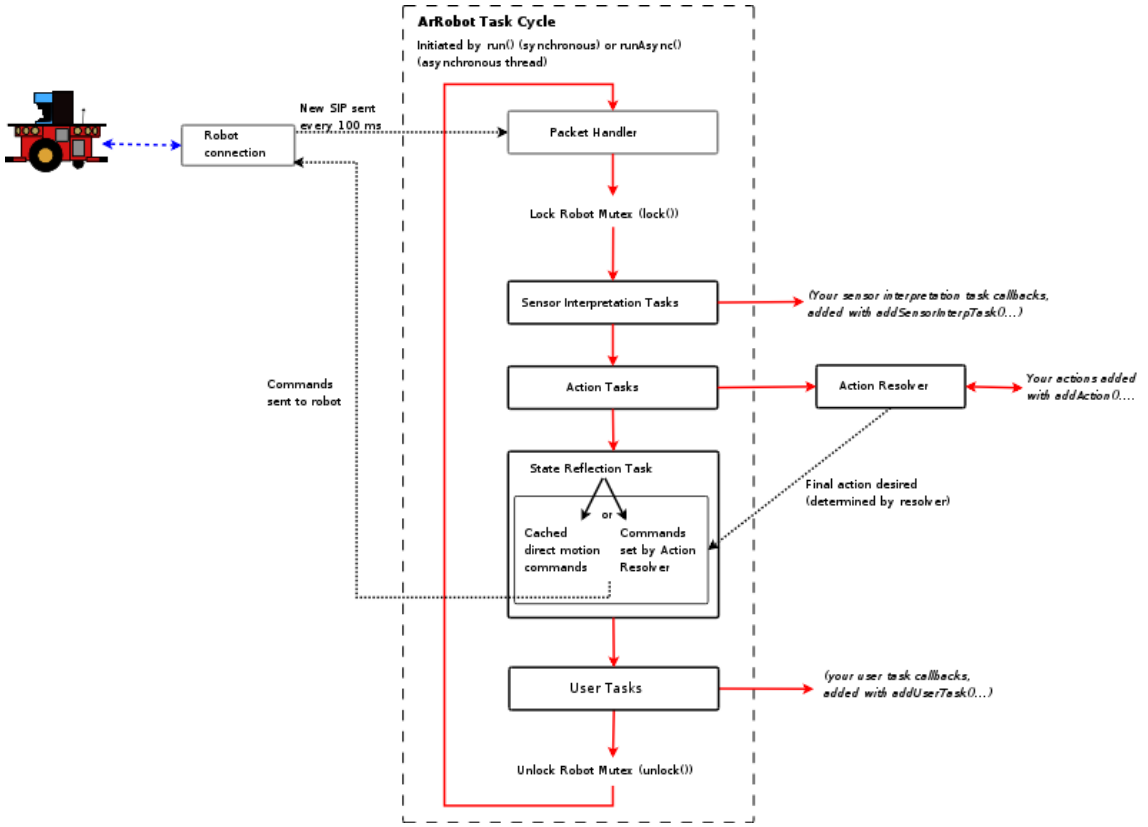


Figure 6.4: Overview of the task cycle.

6.2.2 RobuArm

Figure 6.4 shows the robotic manipulator called RobuArm, which is utilized in the experimental system. It is a six-DOF robotic manipulator which is developed by Robosoft Company in France. It has six revolute joints with a 10 kg payload, and is very powerful. The main advantage of this arm is the controllability of the acceleration, velocity, and position. The detailed specifications of this arm are given in Table 6.1.

Figure 6.5: RobuArm—the manipulator.

Table 6.1: Technical specifications RobuArm.

Specification	Value	Notes
Usage	Outdoor	
Kinematics	6 independent axes	The 3 first axis have mechanical brakes when power is OFF
Weight	About 42 kg	
Maximum payload	10 kg	
Maximum speed per axis	1 rad/sec	
Dimensions (*L* x *w* x *h*)	See pictures below for its main dimensions and the working envelope	
Main power supply	48 VDC @ 30 A	
Protection index	IP65	IP65: Protected against water splashes from all directions by hose (NF EN60529)
Humidity	0-90% without condensation	
Temperature	Functioning : 0°C +50°C Storage : 0°C +50°C	

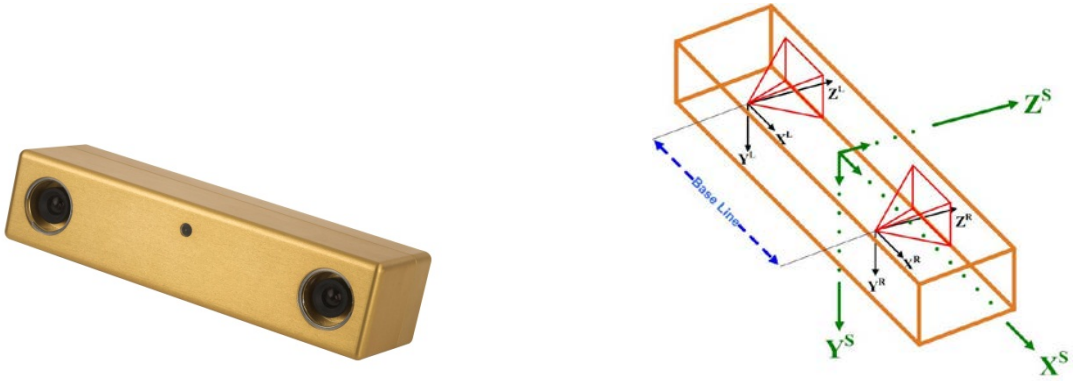
Specification	Value	Notes
Embedded controller	EMTRION HiCO.SH7780-SBC	
OS	Windows-CE®	C++
Programming software	robuBOX	C#, Microsoft Robotic Developer Studio
Integrated sensors	Optical encoder on each axis Magnetic position limit switch	
Interface of supervision	Ethernet	On-board UDP server
User interface	- Power connector - Communication connector - Brake release switches	
Main Colour	Black	

6.2.3 Sensors

There are two types of cameras that are utilized in the present work: a stereo camera and a webcam.

Stereo Camera

A BumbleBee®2 (Figure 6.6(a)) by Point Grey Research is used as the stereo-vision eye-to-object camera. The stereo camera is directly mounted on the mobile base. Each camera is calibrated using the same camera calibration method that was described in Chapter 3. The two cameras are ultimately linked to each other by a common coordinate frame, as shown in Figure 6.6 (b). The calibration results of these two cameras are given in Table 6.2.



(a) (b)

Figure 6.6: (a) BumbleBee®2 stereo camera; (b) Coordinate frames of the camera.

Table 6.2: Intrinsic parameters of the stereo camera.

		Left	Right
Focal length	f (mm)	2.5	
Focal Length to Pixel Size Ratio	f_x	1321.63149	1314.97694
	f_y	1355.18201	1344.62115
Principal Point	O_r	466	470
	O_c	406	392
Resolution	R	768	
	C	1024	

WebCam

A Logitech QuickCam® Communicate STX™ CCD webcam is used as the monocular eye-in-hand camera. The ball-socket base of the webcam is removed and replaced by a custom-made fixture. Finally, the entire assembly is rigidly mounted on the manipulator's end-effector using a band clamp (see Figure 6.2). The calibration result has been presented in Chapter 3.

Figure 6.7: The Logitech webcam.

Laser distance finder

Figure 6.7 shows the Hokuyo URG-04LX sensor. Scanning laser range finders can be thought of as little sonars using light instead of sound to create 2-D maps of the proximity to nearby objects. Because lasers use light instead of sound, they can make measurements very fast and with an extremely narrow field of view (FOV). The scanner that is utilized in this work has an angular scanning range of 240 degrees, and the angular resolution is ~0.36 degrees with a scanning refresh rate of up to 10 Hz. Distances are reported from 20 mm to 4 m. Power is very reasonable at 500 mA and 5 V. The detailed

specifications are listed in Table 6.3. Figure 6.9 demonstrates an example of the laser readings.

Figure 6.8: The Hokuyo URG-04LX sensor.

Table 6.3: Specifications of the laser distance finder.

Item	Specifications
Power source	5 V +/-5%
Current consumption	0.5 A (Rush current 0.8 A)
Detection range	0.02 m to approximately 4 m
Laser wavelength	785 nm, Class 1
Scan angle	240°
Scan time	100 ms/scan (10.0 Hz)
Resolution	1 mm
Angular Resolution	0.36°
Interface	USB 2.0, RS232
Weight	5.0 oz (141 gm)

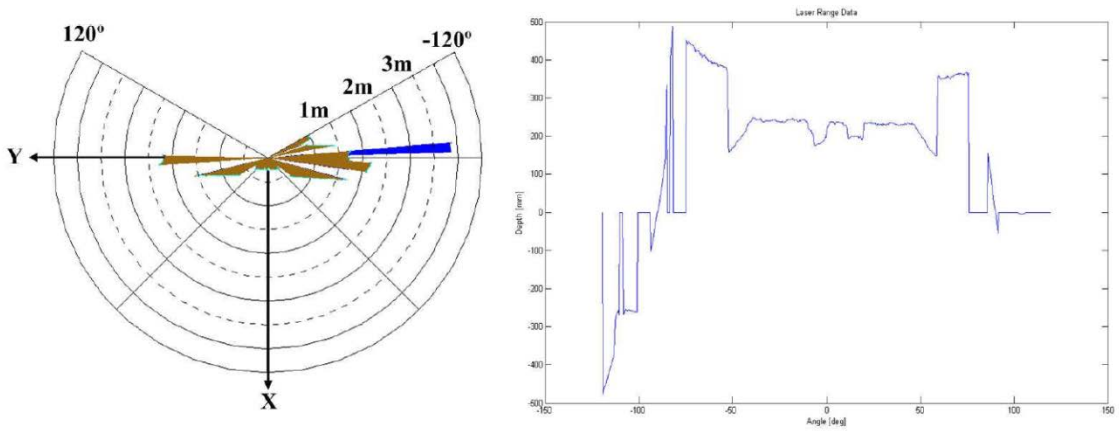


Figure 6.9: An example of laser reading results.

6.3 Software Development

Development, application, and testing of computer software constitute a significant part of the present development. Two types of computer programming language, Visual Basic and C++, have been used to develop software for implementing the functions in the project, particularly related to the robot control subsystem and the mobile navigation subsystem.

The software tool that is used in developing most of the system in this project is Visual Studio. It is a Microsoft's flagship software development tool for computer programmers. It provides a friendly programming environment for Basic and C/C++. Moreover, it has strong compatibility with any Microsoft operating system (Windows 2000, Windows 7, etc.) and is well supported by Windows API for hardware-level programming.

Several other computer technologies are applied in this thesis to enhance the system performance. Windows API (Application Programming Interface), abbreviated as WinAPI, is one of them. It is the core set of application programming interfaces available in Microsoft Windows operating systems, and it provides low-level hardware access to a Windows system from third-party software (sound card access, etc.). Matlab, which includes control toolboxes such as control toolbox, MPC toolbox, optimization toolbox and computer toolbox, is utilized in this work to assist the purposes of analysis and development.

6.4 Mobile Manipulation System

A physical experiment is carried out in the Industrial Automation Laboratory of the University of British Columbia to evaluate the performance of the developed system. Figure 6.10 shows the overall procedure of the experiment. First, an overhead camera with a fisheye lens grabs pictures of the entire workspace. Images are rectified and processed as gray scale images. Second, the SIFT-based object identification algorithm is utilized to identify the mobile robot and the goal location by identifying and tracking the Starbucks logo which is stuck on the robot and the book which is placed in the goal location, respectively. Meanwhile, the camera can provide localization information of the mobile robot and the location of the goal to the mobile robot. Then, the robot checks the current state with the help of the global camera and its own sensing capability. Third, an action is selected by the Q-learning algorithm from the Q-table, which will move the robot to approach the goal. As the object (a bottle with a red cap) moves into the local sensing area of the robot, the ANMPC visual servoing is activated, guiding the mobile robot to move closer to the object. Finally, anther ANMPC visual servoing for the manipulator will be utilized to grasp the object when the object is inside the workspace of the manipulator. The corresponding experimental results are shown in the video clip # 3. Some snapshots of the video clip are shown in Figure 6.10.

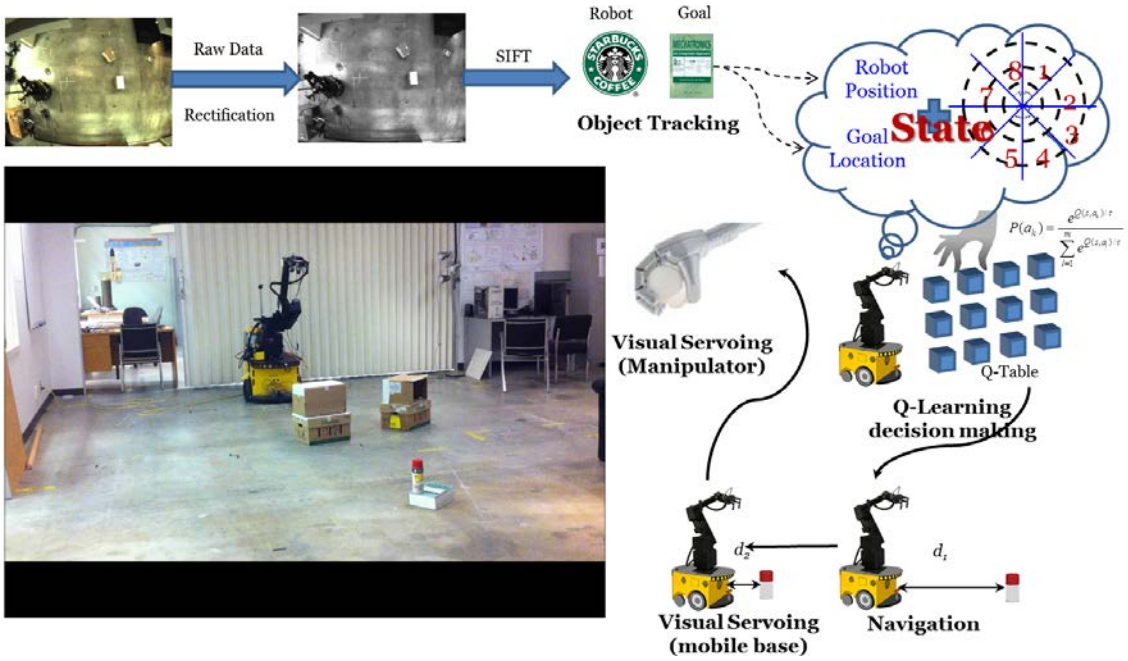
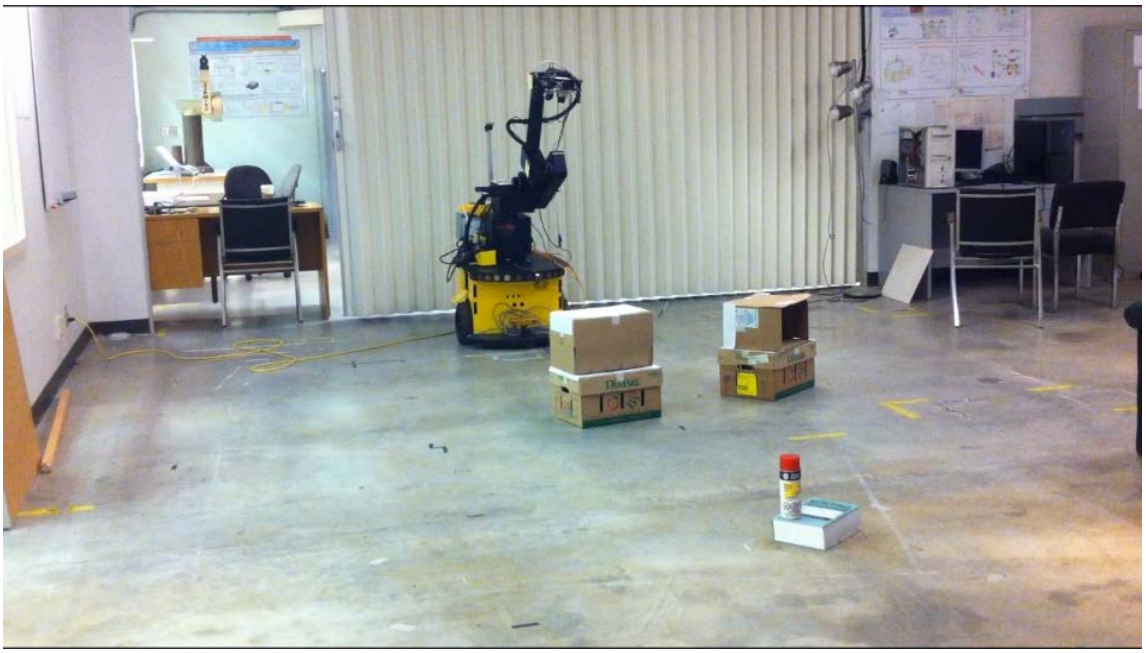
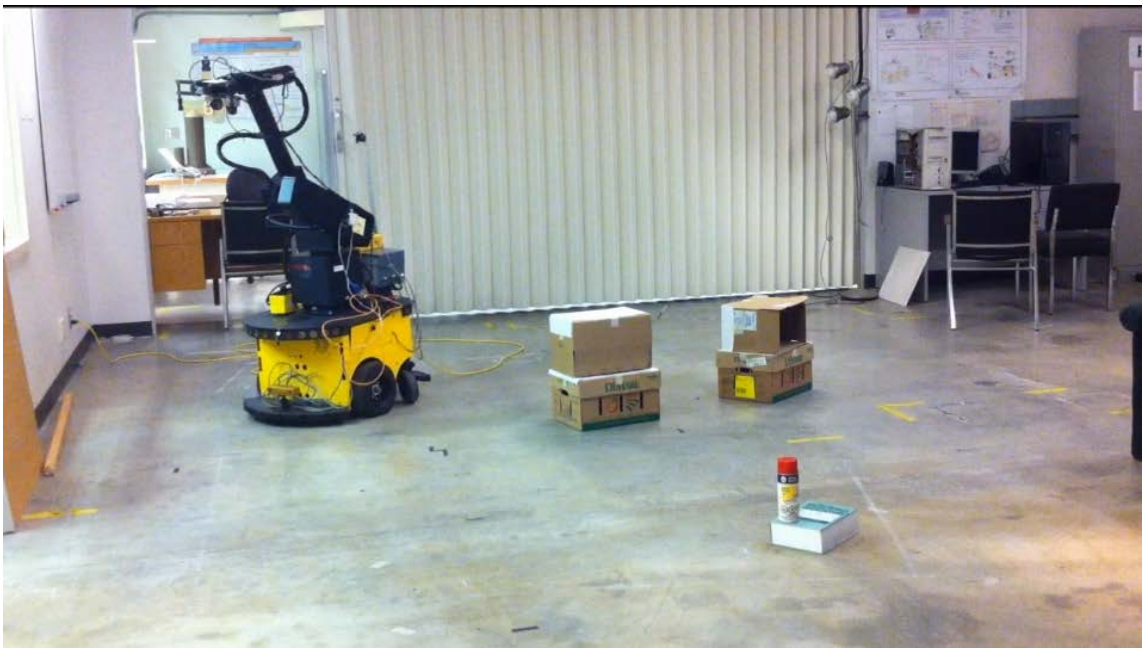


Figure 6.10: The overall procedure of the mobile manipulation experiment.

Several screen shots of the video clip of the experiment are shown in Figure 6.11. Figure 6.11(a) shows the workspace and the initial position of the robot, obstacles, and the object of interest. First, the robot navigates from the current location to a location that is close to the object of interest by utilizing a navigation system that uses pre-trained Q-learning (Figure 6.11(b) and (c)). Next, visual servoing using ANMPC is activated to move the mobile robot closer to the object of interest while making sure that the object is inside the workspace of the manipulator (Figure 6.11(d)). Finally, the visual servo control with ANMPC is used for the manipulator to grasp the object of interest.



(a)



(b)

(c)

(d)

(e)

(f)

Figure 6.11: Screen shots of the experiment.

CHAPTER 7 Conclusions and Suggestions

This thesis designed, analyzed, and implemented a mobile manipulation system which is developed for robotic search-and-rescue and homecare applications, where the robot workspace is unknown and unstructured. The conclusions are summarized in the next section. The subsequent sections discuss primary research contributions made and limitations of the current work, and indicate possible improvements that can be made in future research.

7.1 Conclusions

This thesis addressed the manipulation control of a mobile robot with the support of a sensor network for carrying out dynamically challenging tasks. Two object tracking algorithms were developed in object identification and tracking. Specifically, a color-based tracking algorithm was developed to identify the object of interest and provide the position of the object in the image plane for visual servoing, and a SIFT-based object identification was developed for localization. The depth measurement using stereo cameras was introduced. A system that used Q-learning was developed for mobile robot navigation. The experimental results showed that the robot learned and operated effectively in an unknown and unstructured dynamic environment.

Kinematic models of the mobile robotic platform and the manipulator and a model of the camera were developed and utilized to represent the physical system, where the joint velocities of the robot are the inputs and the position of the feature point in the image is the output. A traditional approach of image-based visual servoing was developed and demonstrated. The scheme of ANMPC was developed, which incorporated a multi-input multi-output (MIMO) control system that could accommodate constraints, including environmental constraints (e.g., obstacles, boundaries, visibility) and physical constraints of the robots (e.g., limits on joint movement, velocity, and torque). In implementing the ANMPC scheme, the nonlinear and time-variant model was linearized on line with

respect to the current position of the feature point and robot joints, using an adaptive approach. The corresponding control architecture predicts the system outputs and generates optimized control actions according to a cost function. This approach was demonstrated to be rather effective in robotic navigation and manipulation.

In order to extend the mobile manipulation system to a wider workspace such as that found in cities and home scenarios, a sensor network was designed and developed employing PFSA (Probabilistic Finite State Automata). The developed PFSA was utilized in both modeling of the sensor data and organizing and representing the sensor network. An application of object identification and tracking was presented; and a heterogeneous sensor network was developed along with a simulation platform in MATLAB. A self-organized and clustered sensor network, which is based on PFSA, was demonstrated.

An integrated robotic system was developed and experimental studies were carried out. The experimentation showed good performance of the developed approaches and of the overall system.

7.2 Primary Contributions

The primary contribution of this thesis was to develop an Adaptive Nonlinear Model Predictive Controller (ANMPC) for motion control of both a mobile robot platform and a robotic manipulator arm. The proposed controller took into account the visibility constraints and physical constraints, and it was able to provide optimized controller outputs. It was shown that the proposed controller was able to perform properly when the object was within the robot's local sensing area.

Efficient and robust algorithms of machine vision were developed, which utilized color and features of objects. With the integration of other sensors, these algorithms helped to determine the pose of the robot in the workspace. Also, they were implemented in the motion control feedback loops of the mobile robot and the robotic manipulator, providing accurate and fast position information of a tracked object.

A traditional algorithm of reinforcement learning, specifically Q-leaning, was developed to properly execute the motion of a mobile robot in an unknown, unstructured and dynamic workspace. With the help of a sensor network, the algorithm guided the mobile robot to move close to the object of interest.

A Probabilistic Finite State Automata (PFSA)-based, information driven, self-organized sensor network was proposed to model the sensor data, and dynamically cluster the sensors for organizing the sensor network to identify and track different objects of interest.

Physical experimentation was developed and presented, which showed the effectiveness of the developed methodologies and systems in applications of mobile robot manipulation.

7.3 Limitations and Suggested Future Research

Although the mobile manipulation system that was developed in the present thesis has shown quite good performance in both simulation and experimentation, there are potential improvements which may constitute a possible future direction of research.

The primary criterion of a successful image-based visual servo system is that the feature point must remain in the camera frame of view, which greatly limits the workspace of the robot. In order to overcome this limitation, 1/2D visual servoing or trajectory planning plus visual servoing may be used, thereby expanding the available workspace.

Linear and fixed constraints are considered in the thesis, which may not represent a real work scenario. Moreover, it also limits the workspace of the robot. In future, more complex constraints may be considered to meet the real world scenarios and increase the robustness of the system.

A more complex experiment may be designed and developed to evaluate the performance and effectiveness of a self-organized and clustered sensor network in identifying and tracking more objects.

In order to perform properly in a real-world scenario such as unstructured and uneven terrains, dark rooms and in the night, a more enhanced robotic system (e.g., a mechanical design that has the ability to autonomously recover from a fall) and sensors (e.g., night vision cameras) will be required.

BIBLIOGRAPHY

Agin, G. J., "Real Time Control of a Robot with a Mobile Camera," *Technical Note 179, SRI International*, Menlo Park, CA, Feb. 1979.

Arkin, R.C., *Behavior-Based Robotics*, The MIT Press, Cambridge, MA, 1998.

Alpaydm, E., *Introduction to Machine Learning*, MIT Press, Cambridge, MA, 2004.

Amarasinghe, D., Mann, G. K. I., and Gosine, R. G., "Vision-based hybrid control scheme for autonomous parking of a mobile robot," *Advanced Robotics*, Vol. 21, No. 8, pp. 905-930, 2007.

Arsenio, A. and Ribeiro, M. I., "Active range sensing for mobile robot localization," *Proceedings of IEEE/RSJ international conference on intelligent robotics and system (IROS' 98)*, Canada, 1998.

Bertozzi, M., Broggi, A., and Castelluccio, S., "A real-time oriented system for vehicle detection," *J. System Architecture*, pp. 317-325, 1997.

Birgesson, E., Howard, A., and Sukhatme, G., "Towards stealthy behaviors," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, Nevada, pp. 1703-1708, October 2003.

Bonnifait, P. and Garcia, G., "Design and experimental validation of an Odometric and Goniometric localization system for outdoor robot vehicles," *IEEE transactions on robotics and automation*, Vol. 14, No. 4, pp. 541-548, 1998.

Borenstein, J., Everett, B., and Feng, L., *Navigating mobile robot: system and techniques*, A. K. Peters, Ltd., Wellesly, MA, 1996.

Bortoff, S., “Path planning for UAVs,” *American Control Conference*, pp.364-368, Chicago, USA, 2000.

Buluswar, S. D. and Draper, B. A., “Color machine vision for autonomous vehicles,” *J. System Architecture*, pp. 317-325, 1997.

Capparella, F., Freda, L., Malagnino, M., and Oriolo, G., “Visual servoing of a wheeled mobile robot for intercepting a moving object,” *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, August, 2005.

Camacho, E. F. and Bordons, C., *Model Predictive Control*, 2nd Edition, Springer, 2007.

Chang, P., and Hebert, M., “First results in omni-directional visual servoing,” *Advanced Robotics*, Vol. 14, No. 3, pp. 205-220, 2000.

Chattopadhyay, I., Mallapragada, G., and Ray, A., “V*: a robot path planning algorithm based on renormalized measure of probabilistic regular languages,” *International Journal of Control*, Vol. 82, No. 5, pp. 849-867, 2009.

Chaumette, F. and Hutchinson, S., “Visual servo control part I: basic approaches,” *IEEE Robotics & Automation Magazine*, Vol. 13, No. 4, pp. 82-90, December 2006.

Chen, J., Dawson, D.M., Dixon, W.E., and Chitrakaran, V.K., “Navigation function-based visual servo control,” *Automatica*, Vol. 43, No. 7, pp. 1165-1177, 2007.

Chen, J., Dixon, W. E., Dawson, D. M., and Mcintyre, M., “Homography-based visual servo tracking control of a wheeled mobile robot,” *IEEE Transactions on Robotics*, Vol. 22, No. 2, pp. 407-416, 2006.

Chen, W. P., Hou, J., and Sha, L., "Dynamic clustering for acoustic target tracking in wireless sensor networks," *IEEE Transaction on Mobile Computing*, Vol. 3, pp. 258-271, July 2004.

Chesi, G. and Hung, Y.S., "Global path-planning for constrained and optimal visual servoing," *IEEE Transactions on Robotics*, Vol. 23, No. 5, pp. 1050-1060, 2007.

Chesi, G., Mariottini, G. L., Prattichizzo, D., and Vicino, A., "Epipole-based visual servoing for mobile robots," *Advanced Robotics*, Vol. 20, No. 2, pp. 255-280, 2006.

Chesi, G., Prattichizzo, D., and Vicino, A., "Straight line path-planning in visual servoing," *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 129, n4, pp. 541-543, 2007.

Corke, P. I. and Hutchinson, S. A. "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 4, pp. 507-515, 2001.

Cowan, N. J., Weingarten, J. D., and Koditschek, D. E., "Visual servoing via navigation functions," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 4, pp. 521-532, 2002.

Cowan, N. J. and Chang, D.E., "Geometric visual servoing," *IEEE Transactions on Robotics*, Vol. 21, No. 6, No. 1128-1138, 2005.

Dean-Le´on, E. C., Parra-Vega, V., and Espinosa-Romero, A., "Visual servoing for constrained planar robots subject to complex friction," *IEEE/ASME Transactions on Mechatronics*, Vol. 11, No. 4, pp. 389-400, 2006.

Dixon, W. E., Dawson, D. M., Zergeroglu, E., and Behal, A. “Adaptive tracking control of a wheeled mobile robot via an uncalibrated camera system,” *IEEE Transactions on Systems, Man and Cybernetics* – Part B, Vol. 31, No. 3, pp. 341-352, 2001.

Fang, Y., Dixon, W. E., Dawson, D. M., and Chawda, P. “Homography-based visual servo regulation of mobile robots,” *IEEE Transactions on Systems*, Man and Cybernetics – Part B, Vol. 35, No. 5, pp. 1041-1050, 2005.

Gangloff, J., Ginhoux, R., and de Mathelin, M., “Model predictive control for compensation of cyclic organ motions in teleoperated laparoscopic surgery,” *IEEE Transactions on Control Systems Technology*, Vol. 14, No. 2, pp. 235-245, 2006.

Gangloff, J. and de Mathelin, M., “Visual servoing of a 6-DOF manipulator for unknown 3-D profile following,” *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 4, pp. 511-519, 2002.

Gaspar, J., Winters, N., and Santos-Victor, J., “Vision-based navigation and environmental representations with an omnidirectional camera,” *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 6, pp. 890-898, 2000.

Ghosh, N., Ravi, Y. B., Patra, A., Mukhopadhyay, S., Paul, S., Mohanty, A. R., and Chattopadhyay, A. B., “Estimation of tool wear during CNC milling using neural network-based sensor fusion ,” *Mechanical Systems and Signal Processing*, Vol. 21, No. 1, pp. 466-479, 2007.

Ginhoux, R., Gangloff, J., and de Mathelin, M., “Active filtering of physiological motion in robotized surgery using predictive control,” *IEEE Transactions on Robotics*, Vol. 21, No. 1, pp. 67-79, 2005.

Goedeme, T., Nuttin, M., Tuytelarrs, T., and Gool, L.V., "Vision based intelligent wheel chair control: the role of vision and inertia sensing in topological navigation," *Journal of Robotic Systems*, Vol. 21, No. 2, pp. 85-94, 2004.

Graves, M., and Batchelor, B., *Machine Vision for the Inspection of Natural Products*, Springer, 2003.

Gu, D., Kamal, W., and Postlethwaite, I., "A UAV waypoint generator," *AIAA 1st Intelligent Systems Technical Conference*, September 2004.

Gu, D., Postlethwaite, I., and Kim, Y., "A comprehensive study on flight path selection algorithms," *IEE Seminar on Target Tracking: Algorithms and Applications*, Birmingham, March, 2006.

Handmann, U., Kalinke, T., Tzomakas, C., Werner, M., and Seelen, W.V., "An image processing system for driver assistance," *Image and Vision Computing*, Vol.18, pp. 367-376, 2000.

Hong, J. S. and May, S. G., "Neural-network-based sensor fusion of optical emission and mass spectroscopy data for real-time fault detection in reactive ion etching," *IEEE Transactions on Industrial Electronics*, Vol. 52, No. 4, pp. 1063-1072, 2005.

Hopcrof, J. E., Motwani, R., and Ullman, J. D., *Introduction to Automata Theory, Languages, and Computation*, 2nd edition. Addison-Wesley, 2001.

Hutchinson, S., Hager, G. D. and Corke, P. I., "A tutorial on visual servo control," *IEEE Transaction On Robotics and Automation*, Vol. 12, No. 5, pp. 651-670, October, 1996.

Ito, T., Yamada, K., and Nishioka, K., "Understanding driving situation using a network model," *Intelligent Vehicle*, pp. 48-53, 1995.

Karray, F. and de Silva, C.W., *Soft Computing and Intelligent Systems Design*, Addison Wesley, New York, 2004.

Kelly, R., Bugarin, E., and Sanchez, V., "Image-based visual control of nonholonomic mobile robots via velocity fields: case of partially calibrated inclined camera," *Proceedings of the 45th IEEE Conference on Decision & Control*, San Diego, CA, December, 2006.

Leonard, J. and Durrant-Whyte, H., "Mobile robot localization by tracking geometric beacons," *IEEE transactions on robotics and automation*, Vol. 7, pp. 89-97, 1991.

Lopez-Nicolas, G., Sagues, C., and Guerrero, J.J., "Homography-based visual control of nonholonomic vehicles," *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, April, 2007.

Lowe, D. G., "Object recognition from local scale-invariant features," *International Conference on Computer Vision*, Corfu, Greece, pp. 1150-1157, September 1999.

Lowe, D. G., "Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*," Vol.60, No. 2, pp. 91-110, 2004.

Luca, A. D., Oriolo, G., and Giordano, P. R., "Image-based visual servoing schemes for nonholonomic mobile manipulators," *Robotica*, Vol. 25, No. 2, pp. 131-145, 2007.

Lumelsky, V. and Stepanov, A., "Path planning strategies for point mobile automation moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, Vol. 2, pp. 402-430, 1987.

Lind, D. and Marcus,M. *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, Cambridge, UK, 1995.

Ma, Y., Kosecka, J., and Sastry, S. S., "Vision guided navigation for a nonholonomic mobile robot," *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 3, pp. 521-536, 1999.

Maciejowski, J.M., *Predictive Control with Constraints*, Prentice Hall, Upper Saddle River, NJ, 2000.

Mansard, N., Stasse, O., Chaumette, F., and Yokoi, K., "Visually-guided grasping while walking on a humanoid robot," *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, April, 2007.

Mariottini, G. L. and Prattichizzo, D. "Image-based visual servoing with central servoing with central catadioptric cameras," *The International Journal of Robotics Research*, Vol. 27, No. 1, pp. 41-56, 2008.

Mariottini, G. L., Oriolo, G., and Prattichizzo, D., "Image-based visual servoing for nonholonomic mobile robots using epipolar geometry," *IEEE Transactions on Robotics*, Vol. 23, No. 1, pp. 87-100, 2007.

Matthews, N., An, P., Charnley D., and Harris, C., "Vehicle detection and recognition in greyscale imagery," *Control Engineering Practice*, Vol. 4, pp. 473-479, 1996.

Mikolajczyk, K. and Schmid, C., "An affine invariant interest point detector," *European Conference on Computer Vision*, Copenhagen, Denmark, pp. 128-142, 2002.

Nierobisch, T., Fischer, W., and Hoffman, F., "Large view visual servoing of a mobile robot with a pan-tilt camera," *Proceedings of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October, 2006.

Palluat, N., Racoceanu, D., and Zerhouni, N., “A neuro-fuzzy monitoring system Application to flexible production systems,” *Computers in Industry*, Vol. 57, No. 6, pp. 528-538, 2006.

Ray, A., “Symbolic dynamic analysis of complex systems for anomaly detection,” *Signal Processing*, Vol. 84, pp. 1115-1130, 2004.

Remazeilles, A. and Chaumette, F. “Image-based robot navigation from an image memory,” *Robotics and Autonomous Systems*, Vol. 55, No. 4, pp. 345-356, 2007.

Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach, Second Edition*, Pearson Education, Inc., Upper Saddle River, NJ, 2003.

Sanderson, A. C. and Weiss, L. E., “Image-based visual servo control using relational graph error signals,” *in Proc. IEEE*, pp. 1074-1077,1980.

Sauvee, M., Poignet, P., and Dombre, E. “Ultrasound image-based visual servoing of a surgical instrument through nonlinear model predictive control,” *The International Journal of Robotics Research*, Vol. 27, No. 1, pp. 25-39, 2008.

Seelen, W.V., Curio, C., Gayko, J., Handmann, U., and Kalinke, T., “Scene analysis organization of behavior in driver assistance system,” *Proceeding of IEEE International Conference on Image Processing*, pp. 524-527, 2000.

Schramm, F. and Morel, G., “Ensuring visibility in calibration-free path planning for image-based visual servoing,” *IEEE Transactions on Robotics*, Vol. 22, No. 4, 848-854, 2006.

Spong, M. W., Hutchinson, S., and Vidyasagar, M., *Robot Modeling and Control*, John Wiley & sons, Inc, 2006.

Steger, C., Ulrich, M., and Wiedemann C., *Machine vision algorithms and applications*, Wiley-VCH, 2008

Siegware, R. and Nourbakhsh, I. R., *Introduction to Autonomous Mobile Robots*, The MIT Press, London, English, 2004.

Su, M. C., Huang, D. Y., Chou, C. H., and Hsieh, C. C., "A reinforcement-learning approach to robot navigation," *IEEE International Conference on Networking, Sensing and Control*, Taipei, Taiwan, March 2004.

Sutton, R. and Barto, A. G., *Reinforcement Learning*, MIT Press, Cambridge, MA, 1998.

Szeliski, R., *Conputer Vision: Algorithms and Applications*, Springer, Ithaca, NA, 2011.

Teng, Y., DeMenthon, D., and Davis, L., "Stealth terrain navigation," *IEEE Transactions on Systems*, Man, and Cybernetics, Vol. 23, No. 1, pp.96-110, 1993.

Tubaishat, M. and Madria, S., "Sensor networks: an overview," *IEEE Potential*, Vol. 22, 2003.

Uğur, E., Doğar, M. R., Çakmak, M., and Şahin, E., "The learning and use of traversability affordance using range images on a mobile robot," *International conference on robotics and automation*, Roma, Italy, pp. 1721-1726, 2007.

Vassallo, R. F., Schneebeli, H. J., and Santos-Victor, J., "Visual servoing and appearance for navigation," *Robotics and Autonomous Systems*, Vol. 31, No. 1-2, pp. 87-97, 2000.

Wang, Y. and de Silva, C. W. , "A fast and robust algorithm for color-blob tracking in multi-robot coordinated tasks," *International journal of information acquisition*, Vol. 3, No. 3, pp. 191-200, 2006.

Wang, Y. and de Silva, C.W., "The sequential Q-learning algorithm with Kalman filtering (SQKF) for multi-robot cooperative transportation tasks," *IEEE/ASME Transactions on Mechatronics*, Vol. 15, No. 2, pp. 261-268, 2010.

Wang, W. Q., Golnaraghi, F. M., and Ismail, F., "Prognosis of machine health condition using neuro-fuzzy systems," *Mechanical Systems and Signal Processing*, Volume 18, No. 4, pp.813-831, 2004.

Yamamoto, Y., Pirjanlan, P., Munich, M., Dibernardo, E., Goncalves, L., Ostrowski, J., and Larlsson, N., "Optical sensing for robot perception and localization," *IEEE workshop on advanced robotics and its social impacts*, Nagoya, Japan, pp. 14-17, 2005

Yang, H. and Sikdar, B., "A protocol for tracking mobile targets using sensor networks," *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications*, Anchorage, Alasha, USA, May 2003.

Yufka, A. and Parlaktuna, O., "Performance comparison of bug algorithms for mobile robots," *5th International Advantaged Technologies Symposium (IATS'09)*, Karabyj, Turkey, May 12-15, 2009.

Zhang, H. and Ostrowski, J. P., "Visual motion planning for mobile robots," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 2, pp. 199-208, 2002.

Zhang, Z., Weiss, R., and Hanson, A. R., "Visual servoing control of autonomous robot calibration and navigation," *Journal of Robotic Systems*, Vol. 16, No. 6, pp. 313-328, 1999.

Zhang, Z. Y., "Flexible camera calibration by viewing a plane from unknown orientations," The Proceeding of the Seventh IEEE International Conference on Computer Vision, Vol. 1, pp. 666-673, September 1999.