Optimal Adaptive Visual Servoing of Robot Manipulators

A Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Electronic Instrumentation and Control

Submitted by

Sikander Hans
(Roll No.800951021)

Under the Guidance of

Souvik Ganguli
Assistant Professor
Department of Electrical and Instrumentation Engineering

Thapar University
(Established under the section 3 of UGC act, 1956)

Patiala, 147004, Punjab, India
July 2011

DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"Optimal Adaptive Visual Servoing Of Robot Manipulators"** in partial fulfillment of award of degree of **Master of Engineering** in **Electronics Instrumentation and Control** submitted in Electrical and Instrumentation Engineering department, Thapar University, Patiala is an authentic record of my own work carried under the supervision of **Souvik Ganguli**, Associate Professor, Department of Electrical and Instrumentation Engineering, Thapar University, Patiala, Punjab.

Date: 14-07-11

Sikander Hans
Roll No: 800951021

I certify that the above statement made by the student is correct to the best of my knowledge and belief.

Date:

Souvik Ganguli
Assistant Professor
Department of Electrical and Instrumentation Engineering
Thapar University, Patiala
Punjab

Countersigned By

S. Ghosh 14/7/11

Dr. Smarajit Ghosh
Head of Department
Department of Electrical and Instrumentation Engineering
Thapar University, Patiala
Punjab

Dr. S K Mohapatra
Dean of Academic Affairs
Thapar University, Patiala
Punjab

ABSTRACT

The system proposed in this thesis is a part of object tracking system. The work done here is designed to perform two tasks. Firstly the direction of motion of the object is detected and given to the user in visual form. Secondly the speed with which the object is moving is calculated in terms of pixels per frame. In order to achieve this first a set of photographs of the moving object in various positions at regular time intervals is taken. Then the motion is detected by finding the difference of the consecutive images. In order to calculate speed, first the segmentation is done of the images so that the different objects can be separated. Then on the basis of the area the desired region in the image (i.e. the object to be tracked) is separated from the others. In this case a ball has been used as the object. After separating the region the centroid of the region of interest is calculated. This is done for all the images collected. Once the centroid position of the region is found in all the images the coordinates are subtracted to find out the number of pixels the object has moved in the next frame. In this manner it can be calculated that how much pixels the object has traveled in the complete set of images. Therefore the speed of the image is obtained in terms of pixels per frame.

ACKNOWLEDGEMENT

I express my deep sense of gratitude and respects to my guide **Mr. Souvik Ganguli**, Department of Electrical & instrumentation Engineering, Thapar University, Patiala for his keen interest and valuable guidance, strong motivation and constant encouragement for the Seminar work. I thank him for his great patience, constructive criticism and myriad useful suggestions apart from invaluable guidance to me. I am sure that the knowledge gained through my association with my supervisor shall go a long way in helping me to realize my goals in life.

I owe my thanks to **Dr. Smarajit Ghosh**, Head of Department of Electrical & Ins - trumentation engineering for his kind support.

Finally, I would like to express my deepest gratitude to my parents and family, without whom I am nothing, to provide me great opportunities, everlasting support, big encouragement and lots of love.

SIKANDER HANS
Roll No: 800951021

CONTENTS

LIST OF FIGURES

Chapter 1

VISUAL SERVOING SYSTEM

1.1 Introduction: The objective of this work is to use of computer vision data to control the motion of a robot. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or on a mobile robot, in which case motion of the robot induces camera motion, or the camera can be fixed in the workspace so that it can observe the robot motion from a stationary configuration. The accuracy of the resulting operation then depends directly on the accuracy of the visual sensor and the robot manipulator. An alternative to increase the accuracy of these subsystems is to use a 'visual-feedback' control loop that will improve the overall accuracy of the system. The use of computer vision techniques to control robotic systems has received great popularity in recent times. In addition to robotics, visual servoing algorithms also find interesting applications for interactive vision systems such as video conferencing, tracking, active vision, augmented reality etc.

1.2. Visual Servoing

Visual servo control refers to the use of computer vision data to control the motion of a robot. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or on a mobile robot, in which case motion of the robot induces camera motion, or the camera can be fixed in the workspace so that it can observe the robot motion from a stationary configuration. Other configurations can be considered such as, for instance, several cameras mounted on pan-tilt heads observing the robot motion. The mathematical development of all these cases is similar, and in this tutorial we will focus primarily on the former, so-called eye-in-hand, case. Visual servo control relies on techniques from image processing, computer vision, and control theory. Since it is not possible to cover all of these

in depth in a single article, we will focus here primarily on issues related to control, and to those specific geometric aspects of computer vision that are uniquely relevant to the study of visual servo control. We will not specifically address issues related to feature tracking or three-dimensional (3-D) pose estimation.

1.2.1. Visual Servoing in Robotics

Robot Motion and Robotic Vision are the challenging problems in Robotics research. The task in Robotic Vision is to analyze the images of the scene taken by the cameras attached to the robot and obtain a physical interpretation of the world around it. The motion problem is more complex. Given the position of an object, how does the robot navigate to reach the object? Traditionally, the visual sensing and the navigation problems have been combined in an open-loop fashion i.e., 'looking' and then 'moving' (See Fig. 1.1). The accuracy of the resulting operation then depends directly on the accuracy of the visual sensor and the robot manipulator. An alternative to increase the accuracy of these subsystems is to use a 'visual-feedback' control loop that will improve the overall accuracy of the system. The use of computer vision techniques to control robotic systems has received great popularity in recent times.

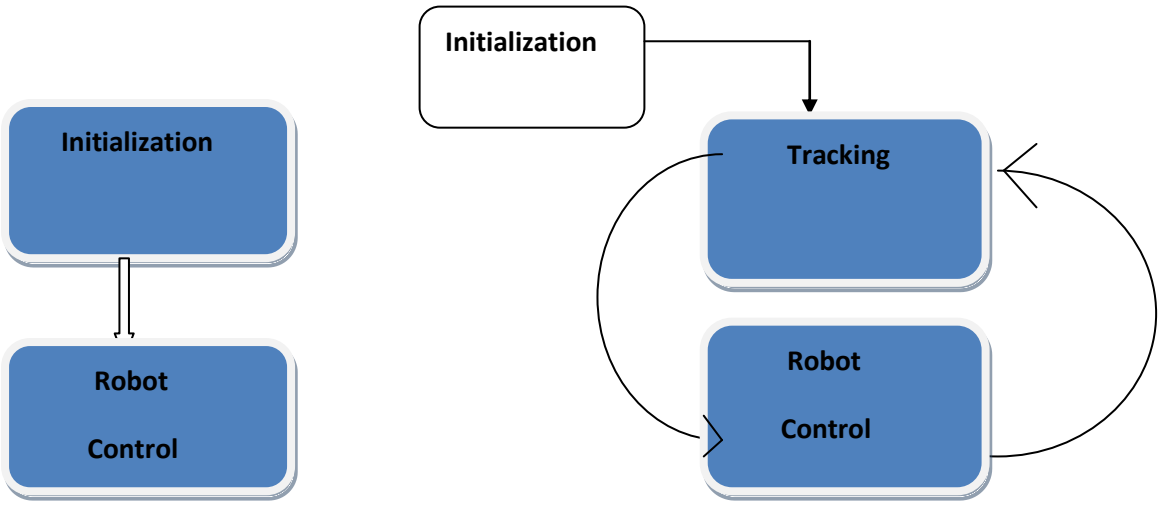
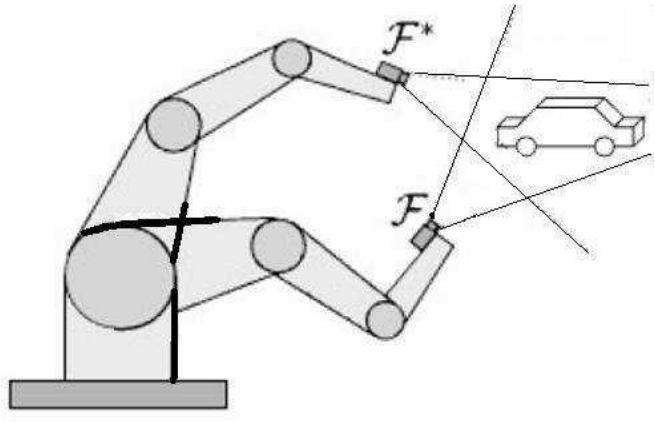


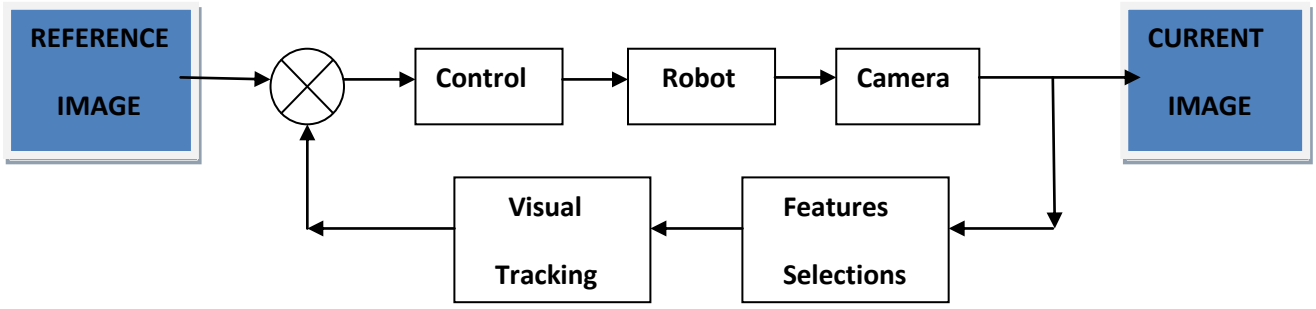
Fig 1.1 Visual Servoing – Feedback-based Systems

1.2.2. Problem of Visual Servoing

The aim of visual servoing is to control a robot using the information provided by a vision system. More precisely, the basic problem to position the robot at a desired position relative to a target object or a set of target features (See Fig. 1.2). It involves the use of one or more cameras and a computer vision system to control the position of the robot's end-effecter as required by the task.



(a)



(b)

Fig 1.2 Task in visual serving is to move the robot from its current to a desired position

Many serving techniques have been proposed and extensively studied in literature. They can be, in general classified into two categories. Algorithms that use optical flow along with Jacobin-based control to control the camera position constitute the class of Image-based Visual Servoing algorithms [1] while techniques that employ 3D information to regulate the error in camera pose pertain to Position-based Visual Servoing [2]. For the relative merits and demerits of the above techniques, the reader may refer to [3].

Recently, a new group of algorithms have been proposed in [4, 5, 6] that exploit a combination of the above methods to estimate the camera displacement between the desired and the current pose. They combine the traditional Jacobin-based control with other techniques to form the class of Hybrid Visual Servoing algorithms. These methods yield a decoupled, optimal camera trajectory and possess a large singularity-free task space.

1.2.3. The Basic Components of Visual Servoing

The aim of all vision-based control schemes is to minimize an error **e**(t), which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^{*} \qquad (1)$$

This formulation is quite general, and it encompasses a wide variety of approaches, as we will see below. The parameters in (1) are defined as follows. The vector **m** (t) is a set of image measurements (e.g., the image coordinates of interest points or the image coordinates of the centroid of an object). These image measurements are used to compute a vector of k visual features, **s** (**m** (t), **a**), in which **a** is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3-D models of objects). The vector $\mathbf{s}^{*}$ contains the desired values of the features. In Part I of the tutorial (this article), we consider the case of a fixed goal pose and a motionless target, i.e., $\mathbf{s}^{*}$ is constant, and changes in **s** depend only on camera motion. Further, we consider here the case of controlling the motion of a camera with six degrees of freedom (6 DOF); e.g., a camera attached to the end effecter of a six degree-of-freedom arm. We will treat more

general cases in Part II of the tutorial. Visual servoing schemes mainly differ in the way that **s** is designed. In this article, we will see two very different approaches. First, we describe image-based visual servo control (IBVS), in which **s** consists of a set of features that are immediately available in the image data. Then, we describe position-based visual servo control (PBVS), in which **s** consists of a set of 3-D parameters, which must be estimated from image measurements. Once **s** is selected, the design of the control scheme can be quite simple. Perhaps the most straightforward approach is to design a velocity controller. To do this, we require the relationship between the time variation of **s** and the camera velocity. Let the spatial velocity of the camera be denoted by $v_c = (v_c, \omega_c)$, with v_c the instantaneous linear velocity of the origin of the camera frame and ω_c the instantaneous angular velocity of the camera frame the relationship between $\dot{\mathbf{s}}$ and v_c is given by

$$\dot{s} = L_s \, v_c, \tag{2}$$

In which $L_s \in Rk\times6$ is named the interaction matrix related to **s**. The term feature Jacobin is also used somewhat interchangeably in the visual servo literature. Using (1) and (2), we immediately obtain the relationship between camera velocity and the time variation of the error:

$$\dot{e} = L_e \, v_c \tag{3}$$

Where $L_e = L_s$.

Considering v_c as the input to the robot controller, and if we would like for instance to try to ensure an exponential decoupled decrease of the error (i.e., $\dot{e} = -\lambda\mathbf{e}$), we obtain using (3):

$$v_c = -\lambda L_e + e \tag{4}$$

Where $L_e + \in R^{6\times k}$ is chosen as the Moore-Penrose pseudo inverse of L_e, that is

$$L_e + = \left(L_e t L_e\right)^{-1} L_e t$$

When L_e is of full rank 6. This choice allows $\|\dot{e} - \lambda L_e L_e + e\|$ and $\|v_c\|$ to be minimal. When k = 6, if det $L_e \neq 0$ it it possible to invert L_e, giving the control $v_c = -\lambda L_e - 1e$. In real visual servo systems, it is impossible to know perfectly in practice either L_e or $L_e +$. So an approximation or an estimation of one of these two matrices must be realized. In the sequel, we denote both the pseudo inverse of the approximation of the interaction matrix and the approximation of the pseudo inverse of the interaction matrix by the symbol $\widehat{L}_e +$.

Using this notation, the control law is in fact:

$$v_c = -\lambda \widehat{L}_e + e \qquad (5)$$

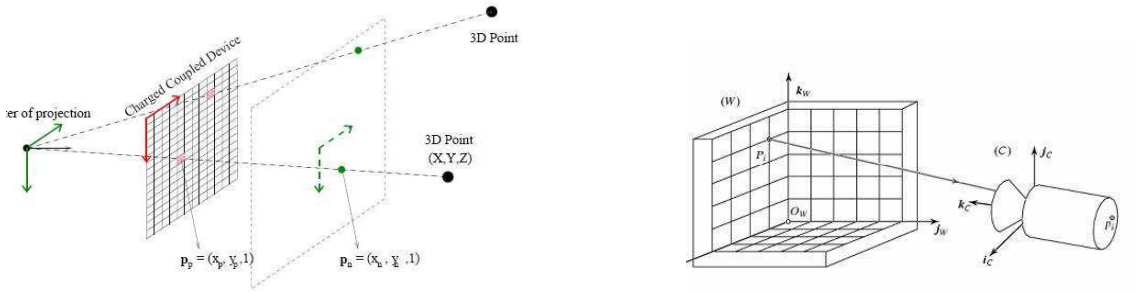
This is the basic design implemented by most visual servo controllers.

1.3. Background and Related Work

1.3.1 Computer Vision fundamentals

To avoid the intrusiveness of mechanical and magnetic sensors, it is desirable to use 'touch free' computer vision-based systems. In context of visual systems, the primary concept that needs to be understood is the geometry of the scene being analyzed by the camera. Geometry is used to explain the relation between robot motion in the world and related 'image' motion. In general, three coordinate frames are used to describe the scene viz., task space (on robot), sensor space (on camera) and the 3D world. The transformations across frames can be computed by estimating the displacement between the frames. The position and orientation of the camera in the 3D world is given by its 'pose', which is calculated during the 'calibration' step (See Fig. 1.3). It is given by a 11-parameter model consisting of the camera

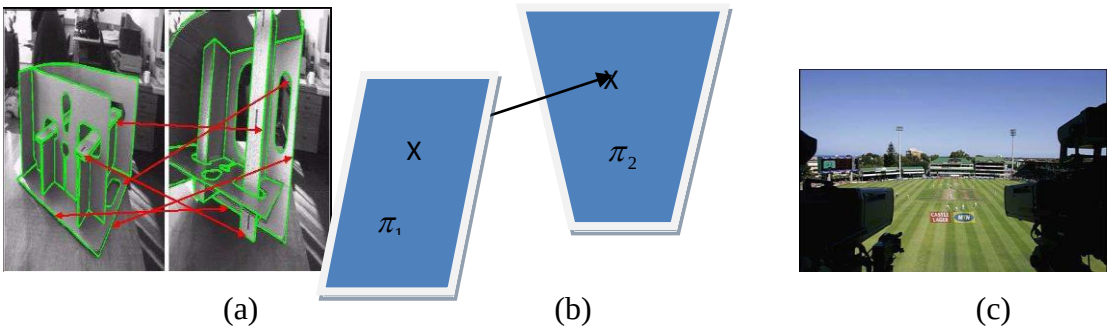
intrinsic and extrinsic parameters. Given the camera pose, the projection of the 3D features on to the 2D image plane is given by the projection equation as p = MP. The parameters extracted from the image features give the 'feature vector'. Most vision-based control approaches generally use points as visual features. Other features (straight lines, ellipses, contours, etc.) can be also 6extracted from the images. A fundamental yet unsolved problem in computer vision is to estimate correspondences between features across images taken by a camera in different views.



(a) (b)

Fig 1.3 Camera projection and calibration

Fig. 1.4(a) shows the corresponding features of two scene views. This problem is particularly difficult when the displacement of the camera between the two images is big and when lighting conditions change.



(a) (b) (c)

Fig 1.4 Correspondence and Homography estimation

Tracking features is similar to the matching problem, however in this case the displacement between the two images is generally smaller. If all the 3D points lie on a plane, then the transformation relating two corresponding images is defined by a homography (Fig. 1.4(b)). A good example illustrating the concept of homography is the appearance of advertisements on playgrounds (Fig. 1.4(c)).

1.4. Types of Visual Control

Visual servoing systems are generally classified depending on the number of cameras used (monocular, stereo etc) or on the positions of the camera with respect to the robot (eye-in-hand or eye-to-hand etc) or based on the use of the target knowledge or on the design of the error function to minimize in order to reposition the robot. The general paradigm consists a combination of off-line and on-line steps. The offline consists of camera/robot calibration, computation of the target features (image set-point), matching target features with observed features while the on-line steps include tracking of image features over time, computation of the error function between current and target location and the computation of the kinematic screw for moving the robot.

Major classification of the controls is based on the use of features in defining the task function. Based on the visual information being utilized, the serving algorithms can be classified into three categories: position-based (3D), image-based (2D), or hybrid (21 2D) visual servo control.

1.4.1. Image-based Visual Servo Control

Classical Image-Based Visual Servo

Traditional image-based control schemes [7], [8] use the image-plane coordinates of a set of points (other choices are possible, but we defer discussion of these for Part II of the tutorial) to define the set **s**. The image measurements **m** are usually the pixel coordinates of the set of

image points (but this is not the only possible choice), and the parameters **a** in the definition of **s** = **s** (**m**, **a**) in (1) are nothing but the camera intrinsic parameters to go from image measurements expressed in pixels to the features.

The Interaction Matrix

More precisely, for a 3-D point with coordinates **X** = (X, Y, and Z) in the camera frame, which projects in the image as a 2-D point with coordinates **x** = (x, y), we have:

$$x = X/Z = (u - c_u)/f\alpha$$
$$y = Y/Z = (v - c_v)/f, \qquad (6)$$

Where **m** = (u, v) gives the coordinates of the image point expressed in pixel units, and **a** = (c u, c v, f, α) is the set of camera intrinsic parameters: c u and c v are the coordinates of the principal point, f is the focal length, and α is the ratio of the pixel dimensions. In this case, we take **s** = **x** = (x, y), the image plane coordinates of the point. The details of imaging geometry and perspective projection can be found in many computer vision texts, including [9], [10]. Taking the time derivative of the projection equations (6), we obtain

$$\begin{cases} \dot{x} = \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\ \dot{y} = \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z. \end{cases} \qquad (7)$$

We can relate the velocity of the 3-D point to the camera spatial velocity using the well-known equation

$$\dot{X} = -v_c - \omega_c \times X \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \end{cases} \qquad (8)$$

Injecting (8) in (7) and grouping terms we obtain

$$\begin{cases} \dot{x} = -v_x / Z + xv_z / Z + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y / Z + yv_z / Z + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{cases} \tag{9}$$

Which can be written:

$$\dot{x} = L_x \, v_c \,, \tag{10}$$

Where the interaction matrix L_x related to **x** is

$$L_x = \begin{bmatrix} \frac{-1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{z} & \frac{y}{z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{11}$$

In the matrix L_x, the value Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of Z. Similarly, the camera intrinsic parameters are involved in the computation of x and y. Thus, L_x cannot be directly used in (4), and estimation or an approximation $\widehat{L}_x$ must be used. We discuss this in more detail below. To control the 6 DOF, at least three points are necessary (i.e., we require k ≥ 6). If we use the feature vector **x** = (**x1**, **x2, x3**), by merely stacking interaction matrices for three points we obtain

$$L_x = \begin{bmatrix} L_{x_1} \\ L_{x_2} \\ L_{x_3} \end{bmatrix}$$

In this case, there will exist some configurations for which L_x is singular [11]. Furthermore, there exist four distinct camera poses for which **e** = 0, i.e., four global minima exist, and it is impossible to differentiate them [12]. For these reasons, more than three points are usually considered.

Approximating the Interaction Matrix

There are several choices available for constructing the estimate $\widehat{L}_e+$ to be used in the control law. One popular scheme is, of course, to choose $\widehat{L}_e+ = L_e+$ if $L_e = L_x$ is known; that is, if the current depth Z of each point is available [13]. In practice, these parameters must be estimated at each iteration of the control scheme. The basic methods presented in this article use classical pose estimation methods that will be briefly presented in the next section.

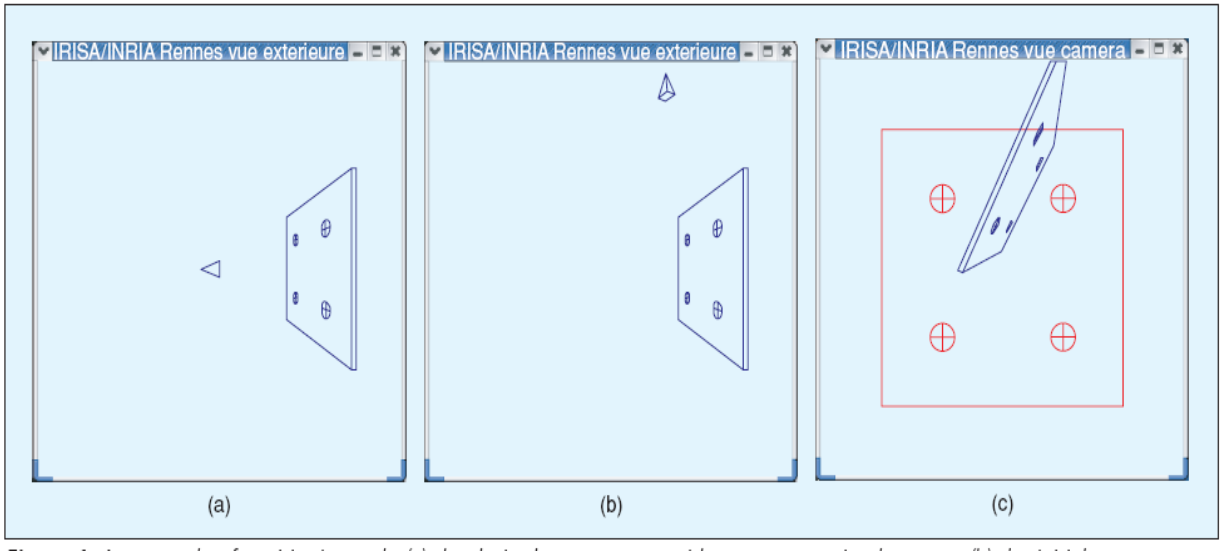


Fig 1.5 An example of positioning task: (a) the desired camera pose with respect to a simple target, (b) the initial camera pose, and (c) the corresponding initial and desired image of the target

Another popular approach is to choose $\widehat{L}_e + = L_{e^*} +$, where L_{e^*} is the value of L_e for the desired position **e** = **e**∗ = 0 [14]. In this case, $\widehat{L}_e +$ is constant, and only the desired depth of each point has to be set, which means no varying 3-D parameters have to be estimated during the visual servo. Finally, the choice $\widehat{L}_e + = ½(L_e + L_{e^*}) +$ has recently been proposed in [15]. Since L_e is involved in this method, the current depth of each point must also be available. We illustrate the behavior of these control schemes with an example. The goal is to position the camera so that it observes square as a centered square in the image (see Figure 1.5).

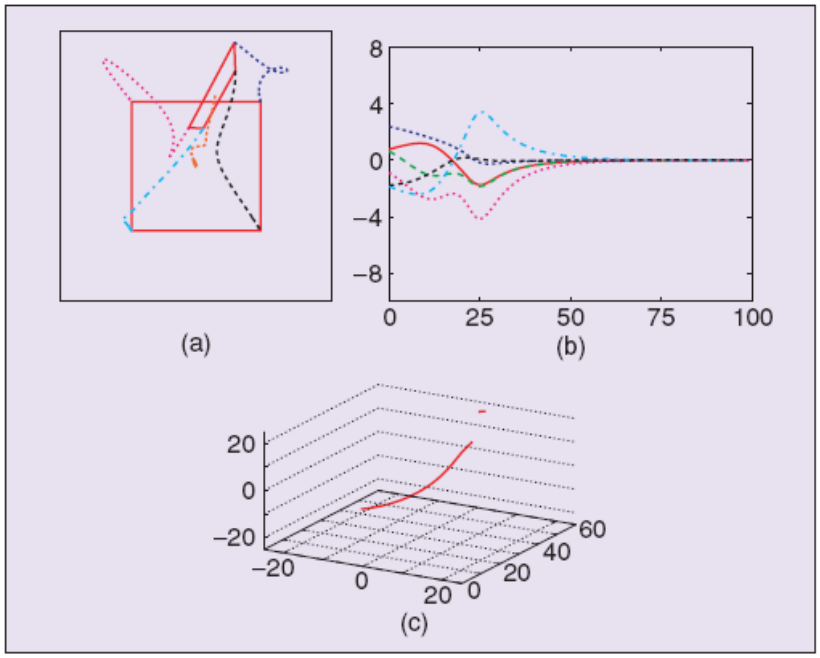


Fig 1.6 The system behavior using **s** = (x1, y1, . . . , x4, y4) and $\widehat{L}_e + = L_{e^*} +$: (a) image points trajectories including the trajectory of the center of the square, which is not used in the control scheme, (b) v_c components (cm/s and dg/s) computed at each iteration of the control scheme, and (c) 3-D trajectory of the camera optical center expressed in Rc∗ (cm).

Define **s** to include the x and y coordinates of the four points forming the square. Note that the initial camera pose has been selected far away from the desired pose, particularly with

regard to the rotational motions, which are known to be the most problematic for IBVS. In the simulations presented in the following, no noise or modeling errors have been introduced in order to allow comparison of different behaviors in perfect conditions.

The results obtained by using $\widehat{L}_e + = L_{e^*} +$ are given in Figure 1.6. Note that despite the large displacement that is required, the system converges. However, neither the behavior in the image nor the computed camera velocity components nor the 3-D trajectory of the camera present desirable properties far from the convergence (i.e., for the first 30 or so iterations). The results obtained using $\widehat{L}_e + = L_e +$ are given in Figure 1.7. In this case, the trajectories of the points in the image are almost straight lines, but the behavior induced in 3-D is even less satisfactory than for the case of $\widehat{L}_e + = L_{e^*} +$ The large camera velocities at the beginning of the servo indicate that the condition number of $\widehat{L}_e +$ is high at the start of the trajectory, and the camera trajectory is far from a straight line.

The choice $\widehat{L}_e + = ½(L_e + L_{e^*}) +$ provides good performance in practice. Indeed, as can be seen in Figure 1.8, the camera velocity components do not include large oscillations and provide a smooth trajectory both in the image and in 3-D.

A Geometrical Interpretation of IBVS

It is quite easy to provide a geometric interpretation of the behavior of the control schemes defined above. The example illustrated in Figure 1.9 corresponds to a pure rotation around the optical axis from the initial configuration (shown in blue) to the desired configuration of four coplanar points parallel to the image plane (shown in red). As explained above, using $L_e +$ in the control scheme attempts to ensure an exponential decrease of the error **e**. It means that when x and y image point coordinates compose this error, the points' trajectories in the image follow straight lines from their initial to their desired positions, when it is possible. This leads to the image motion plotted in green in the figure. The camera motion to realize this image motion can be easily deduced and is indeed composed of a rotational motion around the optical axis, but is combined with a retreating translational motion along

the optical axis [16]. This unexpected motion is due to the choice of the features and to the coupling between the third and sixth columns in the interaction matrix. If the rotation between the initial and desired configurations is very large, this phenomenon is amplified and leads to a particular case for a rotation of π rad where no rotational motion at all will be induced by the control scheme [17]. On the other hand, when the rotation is small, this phenomenon almost disappears. To conclude, the behavior is locally satisfactory (i.e., when the error is small), but it can be unsatisfactory when the error is large. As we will see in the last part of this article, these results are consistent with the local asymptotic stability results that can be obtained for IBVS.

If instead we use $L_{e^*}+$ in the control scheme, the image motion generated can easily be shown to be the blue one plotted in Figure 1.9. Indeed, if we consider the same control scheme as before but starting from $\mathbf{s}*$ to reach $\mathbf{s}$, we

$$v_c = -\lambda\ L_{e^*}+(s^* - s),$$

Which again induces straight-line trajectories from the red points to the blue ones, causing image motion plotted in brown? Going back to our problem, the control scheme computes a camera velocity that is exactly the opposite once

$$v_c = -\lambda\ L_{e^*}+(s - s^*)$$

And thus generates the image motion plotted in red at the red points. Transformed at the blue points, the camera velocity generates the blue image motion and corresponds once again to a rotational motion around the optical axis, combined now with an unexpected forward motion along the optical axis. The same analysis as before can be done, as for large or small errors. We can add that, as soon as the error decreases significantly, both control schemes get closer and tend to the same one (since $L_e = L_{e^*}$ when $e = e^*$) with a nice behavior characterized

with the image motion plotted in black and a camera motion composed of only a rotation around the optical axis when the error tends towards zero.

If we instead use $\widehat{L}_e + = ½(L_e + L_{e^*})+$, it is intuitively clear that considering the mean of L_e and L_{e^*} generates the image motion plotted in black, even when the error is large. In all cases but the rotation around π rad, the camera motion is now a pure rotation around the optical axis, without any unexpected translational motion.

IBVS with a Stereovision System

It is straightforward to extend the IBVS approach to a multicamera system. If a stereovision system is used, and a 3-D point is visible in both left and right images (see Figure 1.10), it is possible to use as visual features.

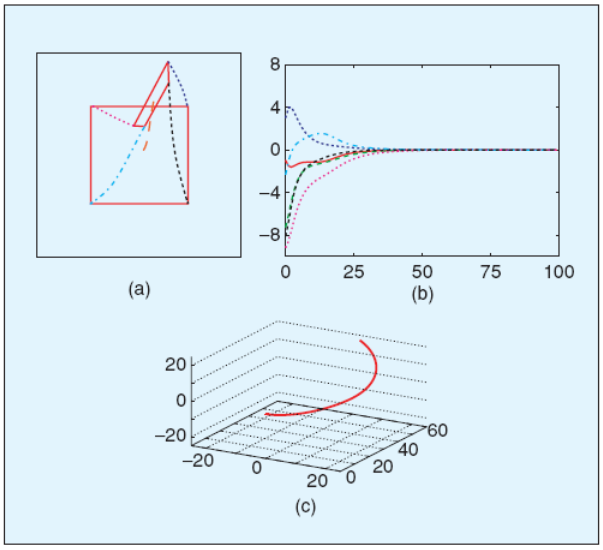


Fig 1.7 The system behavior using **s** = (x1, y1. . . x4, y4) and $\widehat{L}_e + = L_e +$.

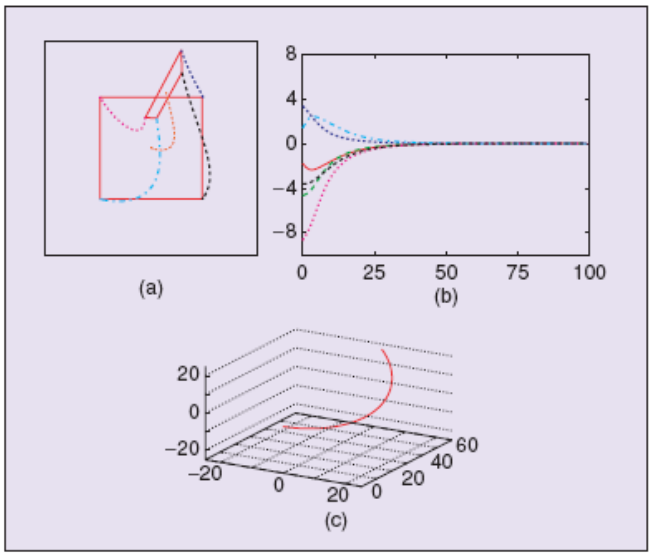


Fig.1.8. The system behavior using **s** = (x1, y1. . . X4, y4) and $\widehat{L}_e + = \frac{1}{2}\ (L_e + L_{e^*})+$.

$$\mathbf{S} = x_s = (x_l, x_r) = (x_l, y_l, x_r, y_r,),$$

i.e., to represent the point by just stacking in **s** the x and y coordinates of the observed point in the left and right images [18]. However, care must be taken when constructing the corresponding interaction matrix since the form given in (10) is expressed in either the left or right camera frame. More precisely, we have:

$$\dot{x}_l = L_{x_l} V_l$$

$$\dot{x}_r = L_{x_r} V_r ,$$

Where v_l and v_r are the spatial velocity of the left and right camera respectively and where the analytical form of L_{x_l} and L_{x_r} are given by (11). By choosing a sensor frame rigidly linked to the stereovision system, we obtain:

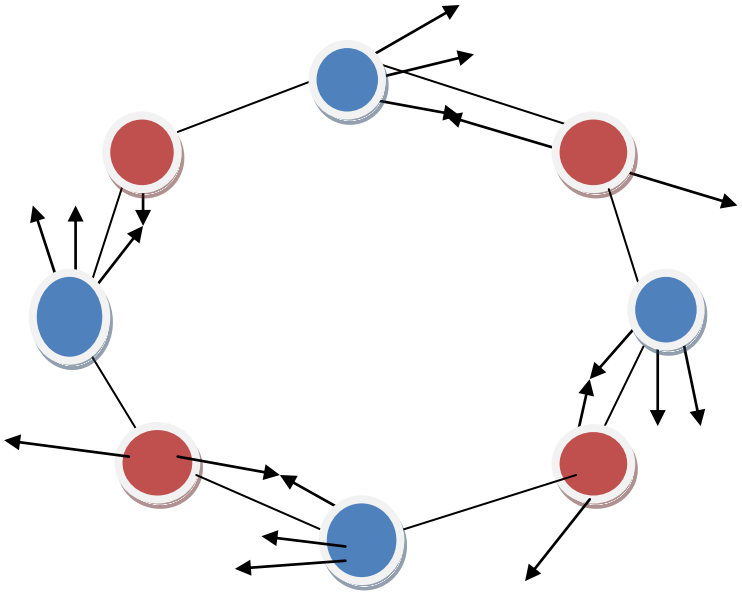


Fig 1.9. A geometrical interpretation of IBVS

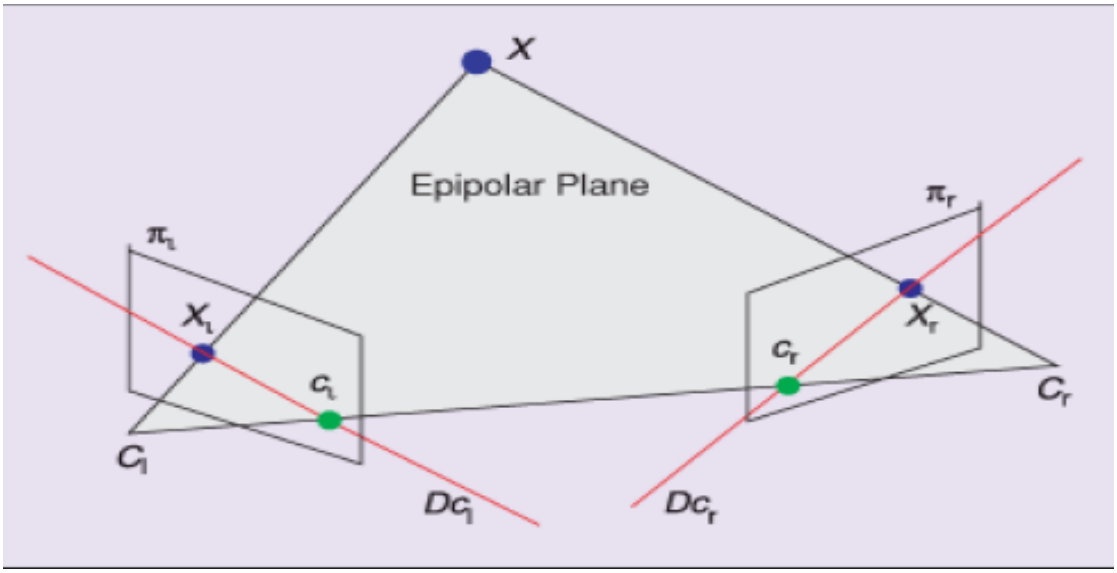


Fig 1.10. A stereovision system

$$\dot{x} = \begin{bmatrix} \dot{x}_l \\ \dot{x}_r \end{bmatrix} = L_{x_s} \; v_s \, ,$$

Where the interaction matrix related to **x**s can be determined using the spatial motion transform matrix **V** to transform velocities expressed in the left or right cameras frames to the sensor frame. **V** is given by [19]

$$\mathbf{V} = \begin{bmatrix} R & [t] \times R \\ 0 & R \end{bmatrix} \qquad \textbf{(12)}$$

Where [**t**] × is the skew symmetric matrix associated to the vector **t** and where (**R**, **t**) ∈ SE (3) is the rigid body transformation from camera to sensor frame. The numerical values for these Position matrices are directly obtained from the calibration step of the stereovision system. Using this equation, we obtain

$$L_{x_s} = \begin{bmatrix} L_{x_l} & {}^{l}V_s \\ L_{x_r} & {}^{r}V_s \end{bmatrix}$$

Note that $L_{x_s} \in R^{4\times6}$ is always of rank 3 because of the epipolar constraint that links the perspective projection of a 3-D point in a stereovision system (see Figure 1.10). Another simple interpretation is that a 3-D point is represented by three independent parameters, making it impossible to find more than three independent parameters using any sensor observing that point.

To control the 6 DOF of the system, it is necessary to consider at least three points, the rank of the interaction matrix by considering only two points being equal to 5. Using a stereovision system, since the 3-D coordinates of any point observed in both images can be easily estimated by a simple triangulation process it is possible and quite natural to use these

3-D coordinates in the features set **s**. Such an approach would be, strictly speaking, a position-based approach, since it would require 3-D parameters in **s**.

1.4.2. Position-based Visual Servo Control

The 3D visual serving is also called position-based visual-servoing [20] since the control law uses directly the error on the position of the camera. Depending on the number of visual features available in the image one can compute the position error using or not the model of the target. The main advantage of this approach is that it directly controls the camera trajectory in Cartesian space. However, since there is no control in the image, the image features used in the pose estimation may leave the image (especially if the robot or the camera are coarsely calibrated), which thus leads to servoing failure. Also note that, if the camera is coarse calibrated, or if errors exist in the 3D model of the target, the current and desired camera poses will not be accurately estimated. Position based visual servoing is usually referred to as a 3D servoing control since image measurements are used to determine the pose of the target with respect to the camera or some common world frame. The error between the current and the desired pose of the target is defined in the task (Cartesian) space of the robot. Hence, the error is a function of pose parameters, **e**(**X**). Two examples of position based servoing are presented in Figure 1.11. The figure on the left shows an example where the camera is controlled from its current pose, ${}^{C}X_{O}$, so to achieve the desired pose with respect to the object, ${}^{C}X^{*}_{O}$. In this example, the camera is attached to the last link of a manipulator and observes a static or a moving target, and the model of the object is used to estimate its pose. The figure on the right shows an example of a static camera and a moving object. It is assumed here that the object is held by a manipulator which is then controlled to, again, achieve the desired pose between the object and the camera. Since the pose of the object is estimated relative to the camera, the transformation between the robot and the camera has to be known to generate the required motion of the manipulator.

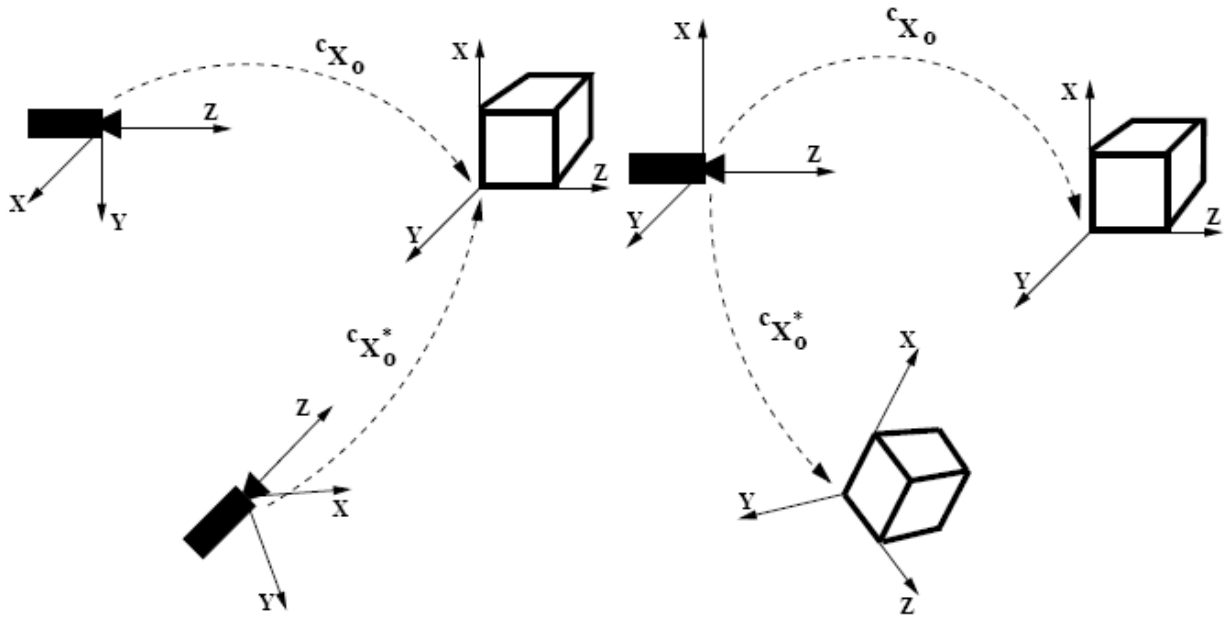


Fig 1.11 Two examples of position based visual servoing control: left) an example of an eye–in–hand camera configuration where the camera/robot is 20erved from the ${}^{C}X_{O}$ (current pose) to the ${}^{C}X^{*}_{O}$ (desired pose), and right) a monocular, stand– alone camera system used to servo a robot held object from its current to the desired pose.

These examples demonstrate two main reasons why the position based visual servoing is usually not adopted for servoing tasks: i) it requires the estimation of the pose of the target or which requires some form of a model, and ii) to estimate the desired velocity screw of the robot and in order to achieve accurate positioning, it requires precise system calibration (camera, camera/robot). Here, the difference in pose between the desired and the current pose represents an error which is then used to estimate the velocity screw for the robot, $\dot{q} = [V;\Omega]^{T}$, so to minimize the error.

1.4.3. Hybrid Visual Servo Control

Recently, Mails et al. [21] proposed a hybrid control scheme (called 21 2D visual servoing) that avoids the respective drawbacks of 2D and 3D controls. Contrary to position-based visual control, the hybrid control does not need any geometric 3D model of the object.

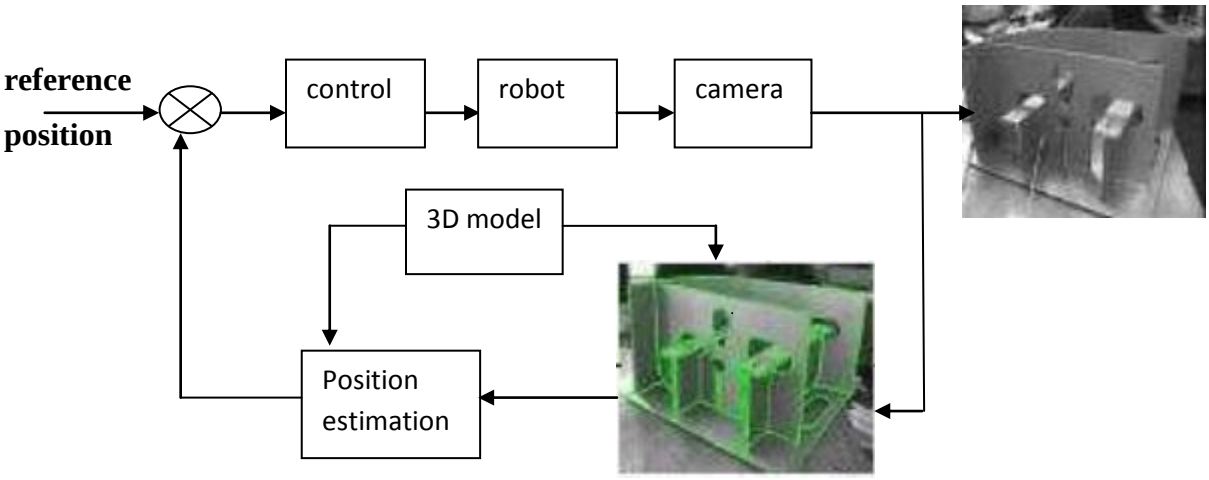


Fig 1.12 Position-based Visual Servo Control

Furthermore and contrary to image-based visual control, the approach ensures the convergence of the control law in the whole task space. This control is based on the estimation of the partial camera displacement from the current to the desired camera poses at each iteration of the control law. Visual features and data extracted from the partial displacement are used to design a decoupled control law controlling the six camera dof [21].

1.5 Conclusion:

Accuracy of the visual servoing system is a challenging problem in the field of robotics. The accuracy can be increased by increasing the accuracy of the subparts. This can be achieved by using a feedback control loop that improves the accuracy of the system as a whole.

1.6 Report Outline

This thesis report is divided into 5 chapters. The first chapter gives the introduction about the visual servoing system. It throws light on the basic components of the visual servoing and the types of visual control. In the second chapter the literature review has been given. This tells us about the background research work that has already been done. The third chapter talks about the adaptive visual servoing. It discusses the classification of the adaptive control techniques and the experimental setup used for it. In the fourth chapter the construction and working of the robot manipulators has been illustrated. Finally in the fifth chapter the system design is given along with the experimental results that were obtained.

Chapter 2

Literature Review

2.1 Introduction:

Visual servoing essentially comprises of two steps: sensing and tracking. Visual Servoing is known as vision based robot control and abbreviated VS is a technique which uses feedback information extracted from a vision sensor to control the motion of the robot. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or a mobile robot in which case motion of the robot induce camera motion, or the camera can be fixed in the work space so that it can observe the robot motion from a stationary configuration.

2.2 Literature Review

XIAO-JING SHEN, JUN-MIN PAN et al .Traditionally, Visual servoing system is divided in position-based visual servoing system(F'BVS) and image-based visual servoing system(IBVS). PBVS needs to estimate the pose of target in 3-D space, and BVS often control the feature in 2-D image plane. The performances of both systems depend on the selection of the state vector. So, it is important to select a desirable state vector to design a visual servoing system. Of image motion, we find when camera motion consists of only translation motion and the target is one rigid planar surface, the projected point position of the target centroid in image plane-$\left[x_{c,}y_c\right]^T$, and the radio of two depths-s, can explicitly describe the target pose with respect to the camera. In addition, x_c, y_c, s can be easily calculated .from image moments which are quickly computed from a binary image. So, in this paper, the vector $\left[x_c, y_c, s\right]^T$ is used as the state vector. Then a simple image Jacobean matrix is gotten from the state equation and leads to a simple adaptive controller which can drive a camera to the ideal position [47].

Chris Gaskett, Luke Fletcher and Alexander Zelinsky et al.A novel reinforcement learning algorithm is applied to a visual servoing task on a real mobile robot. There is no requirement for camera calibration, an actuator model or a knowledgeable teacher. The controller learns from a critic which gives a scalar reward. The learning algorithm handles continuously valued states and actions and can learn from good and bad experiences including data gathered while performing unrelated behaviours and from historical data. Experimental results are presented. [46]

Hutchinson, S. Hager, G.D. Corke, P.I. et al. This article provides a tutorial introduction to visual servo control of robotic manipulators. Since the topic spans many disciplines our goal is limited to providing a basic conceptual framework. We begin by reviewing the prerequisite topics from robotics and computer vision, including a brief review of coordinate transformations, velocity representation, and a description of the geometric aspects of the image formation process. We then present taxonomy of visual servo control systems. The two major classes of systems, position-based and image-based systems, are then discussed in detail. Since any visual servo system must be capable of tracking image features in a sequence of images, we also include an overview of feature-based and correlation-based methods for tracking. We conclude the tutorial with a number of observations on the current directions of the research field of visual servo control. [3]

Koh Hosoda and Minoru Asada et al. This paper propose an adaptive visual servoing method consisting of an on-line estimator of the robot/image Jacobean matrix and a feedback/ feed forward controller for uncaliberated camera-manipulator systems. The estimator does not need a priori knowledge on the kinematic structure nor on parameters of the camera-manipulator system. The controller consists of feed forward and feedback terms to make the image features converge to the desired trajectories using the estimated results. Some experimental results are given to show the validity of the proposed method. [45]

Junaed Sattar, Philippe Giguere, Gregory Dudek and Chris Prahacs et al. This paper describes a visual servoing system for an underwater legged robotic system named AQUA and initial experiments with the system performed in the open sea. A large class of significant applications can be leveraged by allowing such a robot to follow a diver or some other moving target. The robot uses a suite of sensing technologies, primarily based on computer vision, to allow it to navigate in shallow-water environments. The visual servoing system described here allows the robot to track and follow a given target underwater. The servo package is made up of two distinct parts: a tracker and a feedback controller. The system has been evaluated in the sea water and under natural lighting conditions. The servo system has been tested underwater, and with minor modifications the system can be used while the robot is walking on the ground. [44]

Ezio Malis, Fran¸cois Chaumette, and Sylvie Boudet et al. In this paper, we propose a new approach to vision based robot control, called 2-1/2-D visual servoing, which avoids the respective drawbacks of classical position-based and image based visual servoing. Contrary to the position-based visual servoing, our scheme does not need any geometric three-dimensional (3-D) model of the object. Furthermore and contrary to image based visual servoing, our approach ensures the convergence of the control law in the whole task space. 2-1/2-D visual servoing is based on the estimation of the partial camera displacement from the current to the desired camera poses at each iteration of the control law. Visual features and data extracted from the partial displacement allow us to design a decoupled control law controlling the six camera d.o.f. The robustness of our visual servoing scheme with respect to camera calibration errors is also analyzed: the necessary and sufficient conditions for local asymptotic stability are easily obtained. Then, due to the simple structure of the system, sufficient conditions for global asymptotic stability are established. Finally, experimental results with an eye-in-hand robotic system confirm the improvement in the stability and convergence domain of the 2-1/2-D visual servoing with respect to classical position-based and image-based visual servoing. [21]

W. J. Wilson, C. C. W. Hulls, and G. S. Bell et al . This paper presents a complete design methodology for Cartesian position based visual servo control for robots with a single camera mounted at the end-effector. Position based visual servo control requires the explicit calculation of the relative position and orientation (POSE) of the workpiece object with respect to the camera. This is accomplished using image plane measurements of a number of known feature points on the object, and then applying an extended Kalman filter to obtain a recursive solution of the photogrammetric equations, and to properly combine redundant measurements. The control is then designed by specifying the desired trajectories with respect to the object and forming the control error in the end-effector frame. The implementation using a distributed computer architecture is described. An experimental system has been built and used to evaluate the performance of the POSE estimation and the position based visual servo control. Several results for relative trajectory control and target tracking are presented. Results of the experiments showing the effect of loss of some of the redundant features are also presented. [20]

E. Malis, F. Chaumette, and S. Boudet et al. In this paper, we propose a new approach to vision based robot control, called 2-1/2-D visual servoing, which avoids the respective drawbacks of classical position-based and image based visual servoing. Contrary to the position-based visual servoing, our scheme does not need any geometric three-dimensional (3-D) model of the object. Furthermore and contrary to image based visual servoing, our approach ensures the convergence of the control law in the whole task space. 2-1/2-D visual servoing is based on the estimation of the partial camera displacement from the current to the desired camera poses at each iteration of the control law. Visual features and data extracted from the partial displacement allow us to design a decoupled control law controlling the six camera d.o.f. The robustness of our visual servoing scheme with respect to camera calibration errors is also analyzed: the necessary and sufficient conditions for local asymptotic stability are easily obtained. Then, due to the simple structure of the system, sufficient conditions for global asymptotic stability are established. Finally, experimental results with an eye-in-hand robotic system confirm the improvement in the stability and

convergence domain of the 2-1/2-D visual servoing with respect to classical position-based and image-based visual servoing. [21]

2.2 Conclusion:

The visual servo control relies on techniques from image processing, computer vision and control theory. The two major components of the system are a feedback controller and tracker. The accuracy of the resulting operation directly depends on the accuracy of the visual sensor and robot manipulators. This technique has been successfully tested upon the land as well as the underwater robots. The results in case of the land robots is quiet satisfactory while the underwater system is still not that popular.

Chapter 3

Adaptive Visual Servoing for Various Kinds of Robot Systems

3.1. Introduction

Visual information plays an important role for a robot to accomplish given tasks in an unknown/dynamic environment. Many vision researchers have been adopting deliberative approaches to the problem of reconstructing 3-D scene structure from visual information, which are not only very time consuming but also brittle to noise, therefore it seems hard to apply these methods to real robot tasks. Recently, there have been several studies on visual serving, using visual information in the dynamic feedback loop to increase robustness of the closed loop system [22]. For vision-based robots, image features on the image planes are primitive descriptions of the environments. In this sense, feature based visual serving control is the most fundamental one for the vision-based robots in which image features are controlled to converge to the desired ones, and therefore has been focused by a number of researchers [23–34]. In most of the previous work on visual servoing, they assumed that the system structure and parameters were known [23–28], or that the parameters could be identified in an off-line process [35]. Such a controller, however, is not robust for disturbances and changes of the parameters. To overcome this problem, some on-line parameter identification schemes have been proposed [29–34]. Weiss et al. [29] assumed that the system could be modeled by linear SISO (Single Input Single Output) equations, and applied an MRAC (Model Reference Adaptive Control) controller. In [30], the structure and parameters of a camera-manipulator system were assumed to be known, and an ARMAX (auto-regressive with external inputs) model was used to estimate disturbances and positions of the target points. Papanikolopoulos et al. [31, 32] modeled the system using an ARMAX model and estimated the coefficients of the model. They also estimated the depth related

parameters in [33]. Yoshimi and Allen [34] used a special camera-manipulator setup to realize an uncelebrated camera system. Thus, in these approaches, there were restrictions and assumptions on the system that the camera-manipulator system was described as SISO equations [29], that a priori knowledge on the system structure was required [30, 33, 34], or that the depth was assumed to be constant [31, 32]. On the other hand, most of the previous works have paid their attentions only to the feedback serving. They sensed positions of targets and made feedback inputs by subtracting the sensed positions from the desired ones. Using their controllers, the manipulator does not move until the error is observed, which can be considered to be reactive. To increase the ability of trajectory tracking, there have been several researches on feed forward, in which the dynamic motion of the target is predicted [36–38], but no one has mentioned to feed forward control to predict the motion of the robot itself to the best of our knowledge. If the estimator can obtain a kinematic model of the robot system appropriately, the servoing controller can feed forward the obtained kinematics to realize smooth trajectory tracking motion along the desired trajectories designed to accomplish a certain task. For example, a trajectory generator for obstacle avoidance is proposed by authors utilizing the adaptive visual servoing method [39]. In this paper, we propose an adaptive visual serving method consisting of an on-line estimator and a feedback/feed forward controller for uncelebrated camera-manipulator systems.

It has the following features:

1. The estimator does not need a priori knowledge on the system parameters nor on the kinematic structure of the system. That is, we need not to devote ourselves to tedious calibration processes, or to separate the unknown parameters from the system equations, which depend on the detailed knowledge on the kinematic structure of the system.
2. There is no restriction on a camera-manipulator system: the number of cameras, kinds of images features, structure of the system (camera-in manipulator or camera-and-manipulator), the numbers of inputs and outputs (SISO or MIMO). The proposed method is applicable to any kinds of systems.

3. The aim of the estimator is not to obtain the true parameters but to ensure asymptotical convergence of the image features to the desired values under the proposed controller. Therefore, the estimated parameters do not necessarily converge to the true values. In [29–33], they tried to estimate the true parameters, and therefore they need their restrictions and assumptions.

4. The proposed controller can realize smooth tracking motions along the desired trajectories because not only the feedback terms but also feed forward terms are utilized based on the estimated results.

This paper is organized as follows. First, we propose an estimator for an image Jacobean that represents the relation between the image features and the variables that describe the state of the system.

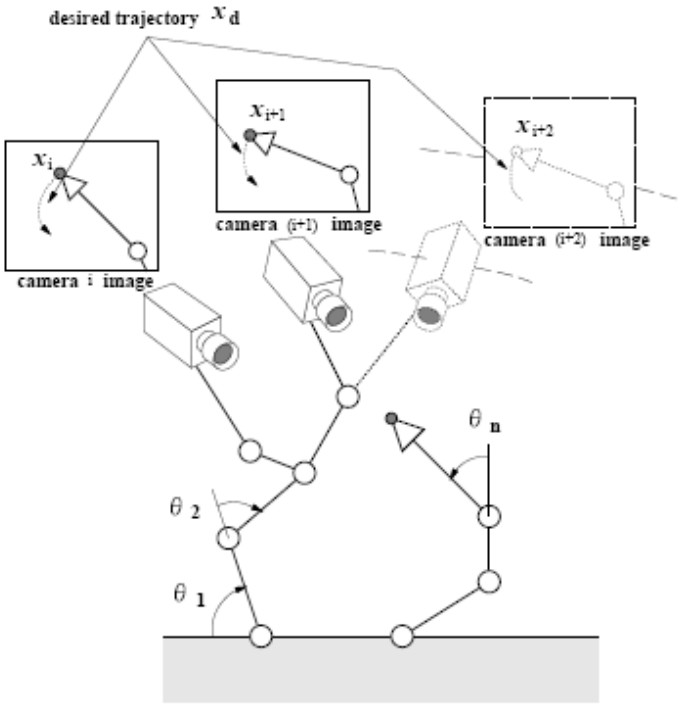


Fig 3.1. Camera-manipulator system

Then, a visual feed forward/feedback servoing controller is proposed based on the estimated Jacobean. Finally, experimental results show the validity of the proposed method

3.2 Adaptive visual servoing

3.2.1. Estimation of the relation between image features and system describing variables

A camera-manipulator system consists of manipulators and cameras, as shown in Figure 3.1. From cameras, one can observe quantities of image features such as position, line length, contour length, and/or area of certain image patterns. The task of the system is to make the quantities of image features converge to the given desired values. The image features are assumed to be on the tip/body of the manipulators or fixed to the ground. That is, a set of measurable variables (which we call system describing variables in the rest of this paper) can describe the state of the camera-manipulator system. Changes of the image features caused by an independently moving object with unknown velocity are not dealt with here.

Let $\theta \in \Re^m$ and $x \in \Re^m$ denote the vectors of the system describing variables and the image features obtained from visual sensors, respectively. A relation between θ and x is

$$x=x(\theta), \tag{1}$$

Because we assume that the system describing variables can describe the state of the system. Differentiating eq.(1), we obtain a velocity relation,

$$\dot{x} = J(\theta)\dot{\theta}, \tag{2}$$

where $J(\theta) = \partial x / \partial \theta^T \in \Re^{m \times n}$ is a Jacobean matrix of time-derivatives of the quantities of image features with respect to those of system describing variables This Jacobean matrix depends on the kinematic structure of the system, the internal camera parameters such as focal length, aspect ratio, distortion coefficients, and the kinematic parameters such as the length of links and the relative position and orientation of cameras with respect to the tip of

the manipulator. Assuming that movement of the camera-manipulator system is slow enough to consider the Jacobean matrix J to be constant during the sampling time, we obtain

$$x(k+1) = x(k) + J(k)u(k), \tag{3}$$

As a discrete model of the system, where J (k) and $u(k)(=\dot{\theta}\Delta T)$ denote the constant Jacobean matrix and a control input vector in k-th step during sampling rate ΔT, respectively. From eq.(3), i-th row vector of the matrix J, $j_i T$, satisfies

$$\left\{ j_i(k+1)^T - j_i(k)^T \right\} u(k+1) = \left\{ x(k+2) - x(k+1) - J(k)u(k+1) \right\}_i. \tag{4}$$

Among an infinite number of solutions of eq. (4), we pick up one to make the norm of weighted time-derivatives of $\hat{j}_i$ as small as possible by the iteration

$$\hat{j}_i(k+1) - \hat{j}_i(k) = \frac{\left\{ x(k+1) - x(k) - \hat{J}(k)u(k) \right\}_i}{u(k)^T W_i(k) u(k)} W_i(k)u(k) \tag{5}$$

Theoretically, the right-hand side of eq.(5) does not tend to infinity when $\|u\|$ tends to 0, because $\left| \left\{ x(k+1) - x(k) - \hat{J}(k)u(k) \right\}_i \right|$ also tends to 0 at the same or faster speed. In real situations, however, the right-hand side is prone to be unstable because of disturbances. To increase the stability of estimation (5), particularly when $\|u\|$ tends to 0, the estimating law is modified as

$$\hat{j}_i(k+1) - \hat{j}_i(k) = \frac{\left\{ x(k+1) - x(k) - \hat{J}(k)u(k) \right\}_i}{\rho_i + u(k)^T W_i(k) u(k)} W_i(k) u(k), \qquad (6)$$

Where ρ_i is an appropriate positive constant that makes the iteration (6) stable When $\|u\|$ tends to 0, the denominator tends to ρ_i and the stability is ensured even if the numerator does not tend to 0 because of disturbances. The positive constant ρ_i is determined so small that ρ_i can be neglected with respect to $\|u\|$ when $\|u\|$ is large. Note that when ρ is in the range $0 < \rho \leq 1$ and the matrix W_i is a covariance matrix, the proposed estimator coincides with the least-mean-square method [40]. The proposed estimator is intended not to obtain the true Jacobean matrix/parameters, but to estimate a matrix that satisfies eq. (3). This is the main difference from [30], [31], and [32], in which they tried to estimate the true parameters. To estimate the true parameters, one has to make restrictions and assumptions on the camera-manipulator system. The proposed estimator, however, is not intended to estimate the true parameters, but to make the closed loop system consisting of this estimator and a controller stable. Therefore, there is neither a restriction nor assumptions on the camera-manipulator system.

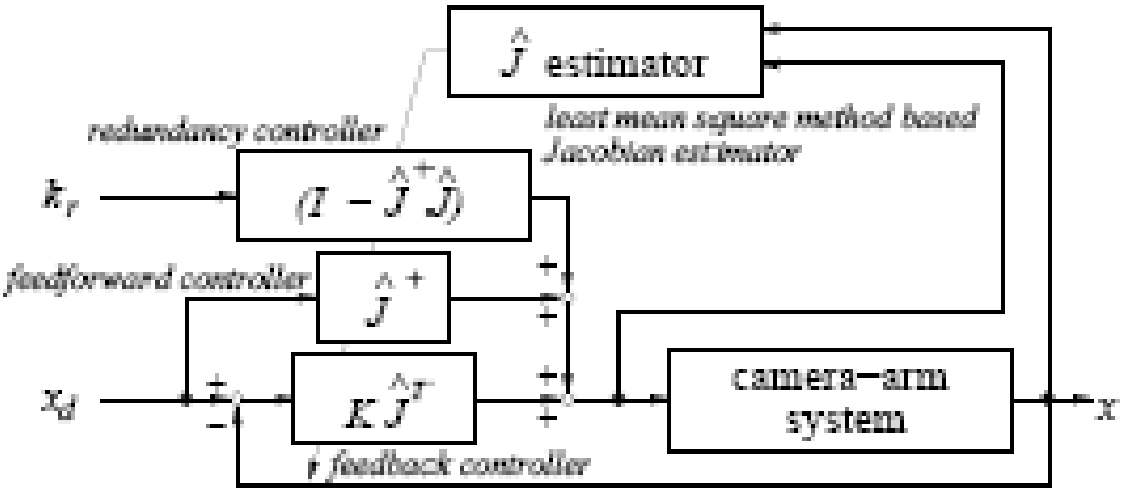


Fig 3.2. Block diagram of the proposed method

3.2.2. Classification of adaptive control techniques

- **Feedforward Adaptive Control**
- **Feedback Adaptive Control**

Feedforward adaptive control: Feed-forward is a term describing an element or pathway within a control system which passes a controlling signal from a source in the control system's external environment, often a command signal from an external operator, to a load elsewhere in its external environment. The main purpose of adaptive control is to handle situations where loads, inertias, and other forces acting on the system change drastically. Some system changes can be unpredictable, and ordinary closed-loop systems may not respond properly when the system transfer function varies. Sometimes, these effects can be handled by conventional linear-control techniques such as gain scheduling (feed-forward control). Conservative design practices may also enable some systems to remain stable even when subjected to parameter changes or unanticipated disturbances.

The price paid for such stability is suboptimal performance, however. Response to changes may be sluggish. Errors may fail to stay within satisfactory limits, or designs must compensate for loose error tolerances in other ways. Adaptive control can help deliver both stability and good response. The approach changes the control algorithm coefficients in real time to compensate for variations in the environment or in the system itself. In general, the controller periodically monitors the system transfer function and then modifies the control algorithm. It does so by simultaneously learning about the process while controlling its behavior. The goal is to make the controller robust to a point where the performance of the complete system is as insensitive as possible to modeling errors and to changes in the environment.

Even ordinary feedback-control systems are adaptive in a limited sense, in that they can compensate for changes at their input that are within the system bandwidth. But these changes are comparatively small. Such systems can become unstable for large input swings, or may simply be unable to compensate for sufficiently large input changes.

There are two main approaches to adaptive feedback-control design: model reference adaptive control (MRAC) and self-tuning regulators (STRs). In MRAC, a reference model describes system performance. The adaptive controller is then designed to force the system or plant to behave like the reference model. Model output is compared to the actual output, and the difference is used to adjust feedback controller parameters.

Most work on MRAC has focused on the design of the adaptation mechanism. This mechanism must note the output error and determine how to adjust the controller coefficients. It must also remain stable under all conditions. One problem with the approach is that there is no general theoretical method of designing an adapter. Thus, most adapter functions are specially keyed to some kind of end application.

An advantage of MRAC is that it provides quick adaptations for defined inputs. A disadvantage is that it has trouble adapting to unknown processes or arbitrary disturbances.

Model-reference controllers have an adaptation mechanism. The comparable component in self-tuning regulators is a tuning algorithm. A self-tuning regulator assumes a linear model for the process being controlled (which is generally nonlinear). It uses a feedback-control law that contains adjustable coefficients. Self-tuning algorithms change the coefficients.

These controllers typically contain an inner and an outer loop. The inner loop consists of an ordinary feedback loop and the plant. This inner loop acts on the plant output in conventional ways. The outer loop adjusts the controller parameters in the inner feedback loop. The outer loop consists of a recursive parameter estimator combined with a control design algorithm.

The recursive estimator monitors plant output and estimates plant dynamics by providing parameter values in a model of the plant. These parameter estimates go to a control-law

design algorithm that sends new coefficients to the conventional feedback controller in the inner loop.

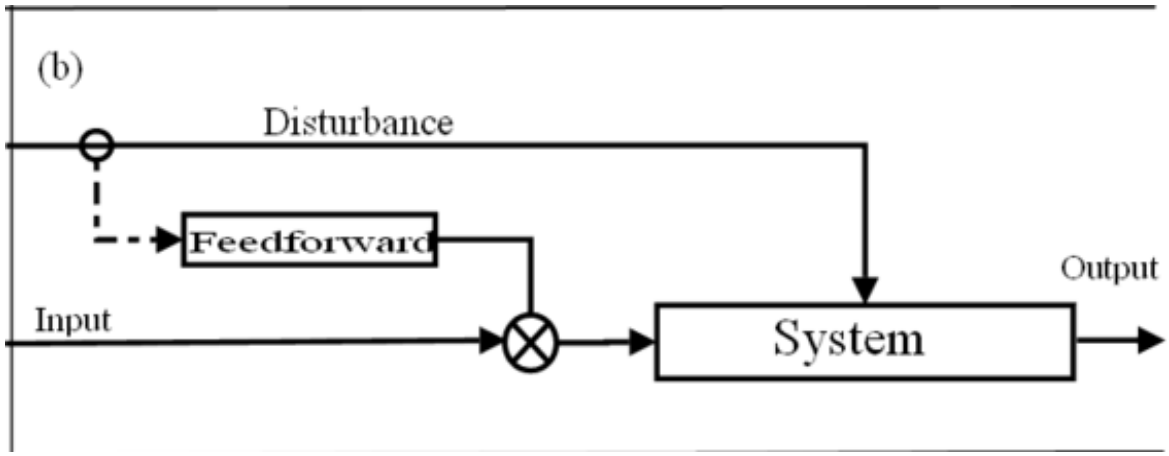


Fig 3.3. feedforward adaptive control

Feedback adaptive control: In case of the feedback adaptive control technique the information about the initial event that is the basis for subsequent modification of the event.

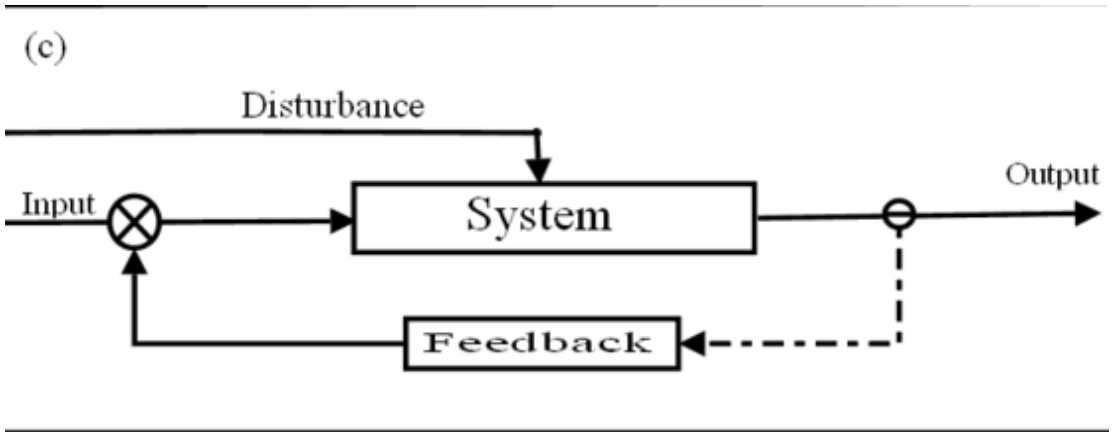


Fig 3.4. feedback adaptive control

Feed forward/feedback visual controller

In this section, a feed forward/feedback visual controller is proposed, based on the estimated Jacobean matrix Ĵ. The aim of the controller is to ensure convergence of the image feature vector x (k) to the desired vector x_d (k). From eq. (3), we can derive a feed forward/feedback controller,

$$u(k) = \hat{J}(k)^{+}\left\{x_d(k+1) - x_d(k)\right\} + \left\{I_n - \hat{J}(k)^{+}\hat{J}(k)\right\}k_r + K\hat{J}(k)^{T}\left\{x_d(k+1) - x(k)\right\} \quad (7)$$

Where Ĵ (k) +, I_n and K denote a pseudo-inverse matrix of Ĵ (k), an n × n identity matrix, and a positive-definite gain matrix, respectively. Let k_r be an arbitrary vector.

The first and second terms on the right-hand side are feed forward terms. The second term on the right-hand side denotes the redundancy of the camera manipulator system. The third term on the right-hand side is a feedback term that ensures stability of the closed loop system. Note that one can use Ĵ (k) + instead of Ĵ (k) T to ensure the closed loop stability [24].

We propose an adaptive visual servoing method consisting of the proposed estimator and controller shown in Figure 3.2 .

3.3. Experiments

To show the validity of the proposed method, some experimental results are given in this section. First step response results of two kinds of camera manipulator systems are given to show how the estimator and the feedback terms of the controller can realize reactive tasks well and how the proposed method can be applied to various kinds of system structures. Then, a result of trajectory tracking is given to show how the feed forward terms work.

3.3.1. Experimental equipment

In Figure 3.5, a camera-manipulator system used for the experiments is shown. The video signals from two cameras (UN401, ELMO) are sent to an image processing board MV200 (Data Cube, image size: 512[pixel] × 480[pixel]) and compressed into the half along the horizontal axis (256[pixel] × 480[pixel]). Two images are pasted onto one image (512[pixel] × 480[pixel], which is sent to a tracking module equipped with a high-speed correlation processor utilizing a SAD (Sum of Absolute Difference) measure by Fujitsu [41].

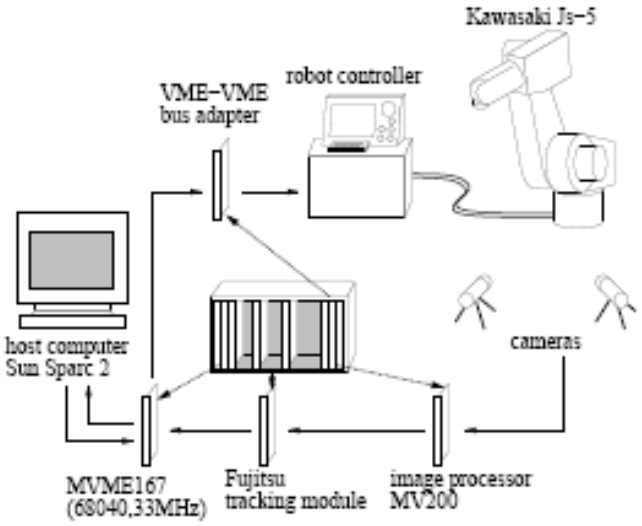


Fig 3.5. Experimental equipment

Before starting an experiment, we specify target images to be tracked by the module. During the experiment, the module tracks the target images, and it feeds coordinates of the images in the image plane to the main control board MVME167 (CPU: 68040, 33MHz, Motorola). The board calculates a desired posture of the manipulator by the proposed method and sends it to the manipulator controller through a VME-VME bus adapter. We use a 6 d. o. f. manipulator Js-5 (Kawasaki Heavy Industry Co.) as a 3 d. o. f. manipulator, maintaining fixed desired orientation of the tip of the manipulator. Therefore, the system describing variable vector Θ is a tip position vector of the manipulator in these experiments. Using this experimental

equipment and writing programs using C language on VxWorks (Wind River), the sampling ratio is 30[Hz].

3.3.2. Step responses of two kinds of systems

To show that the estimator and the feedback terms of the controller can realize reactive tasks well and that the estimator and controller can be applied to different system structures, step responses of two kinds of systems are given in this subsection. At t = 0, the desired image feature coordinates which are taught by showing are fed to the controller. In these experiments, we show step responses, and therefore the feed forward terms in eq. (7) equal zeros. The positive constants ρ_i = 0:8 (i = 1; . . . ; 4) are selected as small as possible in the trial and error manner. We set the weighting matrices W_i(k) = I_3 (i = 1; . . . ; 4). The feedback gain matrix K [m/pixel] in eq. (7) is also selected in the trial and error manner,

$$K = diag\left[1.5\times10^{-4}1.5\times10^{-4}1.5\times10^{-4}1.5\times10^{-4}\right].$$

An initial Jacobean matrix, which is needed at the beginning of the control phase, can be given arbitrarily as far as its rank is full. In the experiment therefore, we give the initial Jacobean matrix as

$$\hat{j}(0) = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \\ 0 & 0 & 0.1 \end{bmatrix}$$

First, we fix the two cameras on the ground. The reference images are windows of a pattern (a cross) which is fixed at the tip of the manipulator. We can find the initial posture of the manipulator, the initial positions of the reference images, and the desired positions of the

reference images. Responses of two cases, (a) applying the proposed controller with the proposed estimator, and (b) applying the proposed controller without on-line estimation, are shown. Because the initial Jacobean matrix is given arbitrary, the controller cannot eliminate error without on-line estimation. On the other hand, using the proposed method, the manipulator can be controlled to make the image features converge to the desired ones. Second, in order to show that it can be applied to various kinds of system structures without a priori knowledge on the systems, we apply the proposed method to a different system in which two cameras are mounted on the tip of the manipulator. The reference images are the windows of a pattern (a cross) fixed on the ground. The positive constants ρ_i, the gain matrix K and the weighting matrices W_i are the same as the previous case. That is, the estimator and the controller are all the same as in the previous experiment. The reference images, their initial positions, and the desired positions are shown.

From these experimental results, we can conclude that the proposed online estimator and the feedback term of the proposed controller are effective to realize a reactive task, and that the proposed method is applicable to different kinds of systems.

3.3.3. Trajectory tracking task

In this subsection, a result on trajectory tracking task is given to show the validity of the proposed purposive visual control. We fix two cameras on the ground. The reference images are windows of a pattern (a cross) fixed at the tip of the manipulator. The desired trajectories must be realizable. Therefore, the desired trajectories are taught by showing to satisfy this constraint. The squares in the figure indicate the location along the desired trajectory every 0:2[s]. The target moves along each trajectory in 12[s] in each image. The initial Jacobean matrix is the same one given in the previous experiments. The tracked image feature vector is $x \in \Re^4$, and the controlled tip position vector of the manipulator is $\theta \in \Re^3$, therefore the second feed forward term on the right-hand side of eq. (7) equals zero. Because our experimental system has a time-delay problem, we cannot stabilize the closed loop system

with 100% feed forward terms in eq. (7). Therefore we apply 30% feed forward terms in the following experiment. One of experimental results is given, where desired trajectories and realized ones are indicated. From this figure, we can see how the proposed method can track the desired trajectories better than the controller without the feed forward terms.

3.4. Conclusion and Discussion

In this paper, we have proposed an adaptive visual servoing method consisting of an on-line estimator and a feedback/feed forward controller for uncelebrated camera-manipulator systems. We have proposed an estimator for the Jacobean matrix that describes the relation between the image features and the system describing variables. Then, a feed forward/feedback controller has been proposed making use of the estimated relation. Finally experimental results are given to show that the proposed method is validity to various kinds of robot systems.

We have to mention to the redundancy of the system. If the camera manipulator system is redundant to accomplish given tasks, the redundancy can be utilized to realize other sub-tasks such as obstacle avoidance. In the proposed controller, the redundancy is denoted as the second term on the right-hand side of eq. (7), but we have not mentioned how to utilize the redundancy in this paper. A study on redundancy in the uncelebrated system is one of our future major works.

One alternative to deal with such redundancy is to introduce another serving method, and to build a hybrid servoing controller. The authors have shown some theoretical and experimental results: (1) hybrid adaptive visual serving/force serving control [42], and (2) adaptive visual servoing control for legged robots [43]. When we build a fast/robust robot system, such kinds of hybrid serving controllers would be powerful and essential.

CHAPTER-4

ROBOT MANIPULATORS

4.1 INTRODUCTION

The word robot finds its origins in robota which means work in Czech

- Robot manipulators
- Mobile robots
  - Ground robots
    - Wheeled robots
    - Legged robots
  - Submarine robots
  - Aerial robots

Both, mobile robots and manipulators are key pieces of the mosaic that constitutes robotics nowadays. Robotics – a term coined by the science fiction writer Isaac Asimov – is as such a rather recent field in modern technology. The good understanding and development of robotics applications are conditioned to the good knowledge of different disciplines. Among these, electrical engineering, mechanical engineering, industrial engineering, computer science and applied mathematics.Hence, robotics incorporates a variety of fields among which is automatic control of robot manipulators.

The movement of each joint may be prismatic, revolute or a combination of both. We consider only joints which are either revolute or prismatic. Under reasonable considerations, the number of joints of a manipulator determines also its number of degrees of freedom (DOF). Typically, a in the Cartesian space and 3 more specify its orientation.

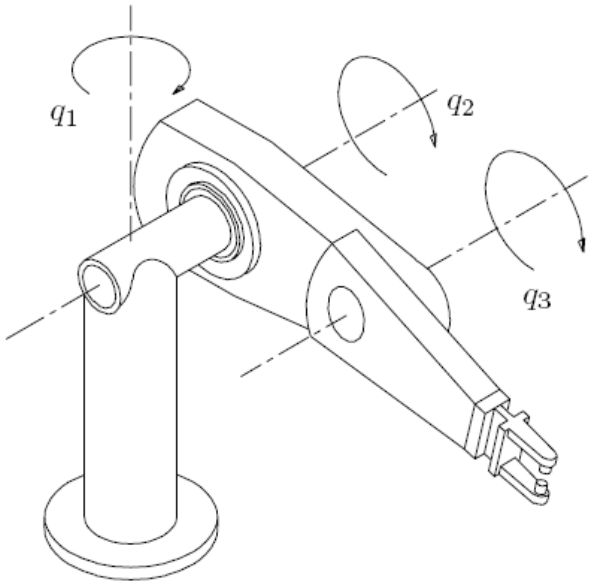


Fig 4.1 Robot manipulator

Figure 4.1 illustrates a robot manipulator. The variables q1, q2 and q3 are referred to as the joint positions of the robot. Consequently, these positions denote under the definition of an adequate reference frame, the positions (displacements) of the robot's joints which may be linear or angular. Considering an n-DOF robot manipulator, the joint positions are collected in the vector **q**

$$q := \begin{bmatrix} q_1 \\ q_2 \\ . \\ q_n \end{bmatrix}$$

the joint positions **q** are measured by sensors conveniently located on the robot[48]. The corresponding joint velocities $\dot{q} := \frac{d}{dt} q$ may also be measured or estimated from joint position evolution.

To each joint corresponds an actuator which may be electromechanical, pneumatic or hydraulic. The actuators have as objective to generate the forces or torques which produce the movement of the links and consequently, the movement of the robot as a whole. For analytical purposes these torques and forces are collected in the vector $\boldsymbol{\tau}$, i.e.

$$\tau := \begin{bmatrix} \tau 1 \\ \tau 2 \\ . \\ . \\ \tau n \end{bmatrix}$$

In its industrial application, robot manipulators are commonly employed in repetitive tasks of precision and others, which may be hazardous for human beings. The main arguments in favor of the use of manipulators in industry is the reduction of production costs, enhancement of precision, quality and productivity while having greater flexibility than specialized machines. In addition to this, there exist applications which are monopolized by robot manipulators, as is the case of tasks in hazardous conditions such as in radioactive, toxic zones or where a risk of explosion exists, as well as spatial and submarine applications. Nonetheless, short-term projections show that assembly tasks will continue to be the main applications of robot manipulators.

APPLICATION

The main arguments in favor of the use of manipulators in industry is the reduction of production costs, enhancement of precision, quality and productivity while having greater flexibility than specialized machines short-term projections show that assembly tasks will continue to be the main applications of robot manipulators

4.2 TYPES OF DESIGN ROBOT

As a general rule, control design may be divided roughly into the following steps:

- familiarization with the physical system under consideration;
- modeling;
- control specifications

4.2.1 Familiarization with the Physical System under Consideration

In the particular case of robot manipulators, there is a wide variety of outputs – temporarily denoted by **y** – whose behavior one may wish to control. For robots moving freely in their workspace, i.e. without interacting with their environment (cf. Figure 4.2) as for instance robots used for painting, "pick and place", laser cutting, etc., the output **y** to be controlled, may correspond to the joint positions **q** and joint velocities ˙**q** or alternatively, to the position and orientation of the end-effector (also called end-tool).

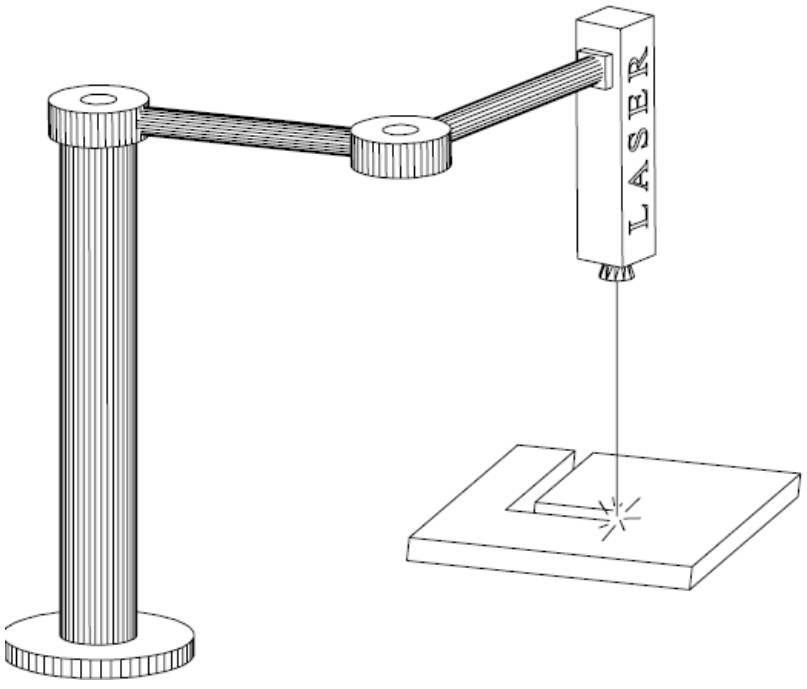


Fig 4.2. Freely moving robot

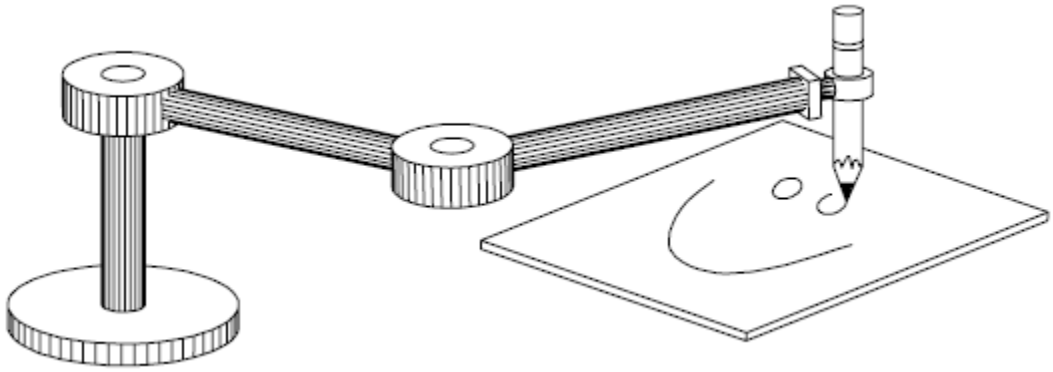


Fig 4.3. Robot interacting with its environment

For robots such as the one depicted in Figure 4.3 that have physical contact with their environment, e.g. to perform tasks involving polishing, deburring of materials, high quality assembling, etc., the output **y** may include the torques and forces **f** exerted by the end-tool over its environment.

Figure 4.4 shows a manipulator holding a marked tray, and a camera which provides an image of the tray with marks. The output **y** in this system may correspond to the coordinates associated to each of the marks with reference to a screen on a monitor[49]. Figure 4.5 depicts a manipulator whose end-effectors has a camera attached to capture the scenery of its environment. In this case, the output **y** may correspond to the coordinates of the dots representing the marks on the screen and which represent visible objects from the environment of the robot From these examples we conclude that the corresponding output **y** of a robot system – involved in a specific class of tasks – may in general, be of the form

$$\mathbf{y} = \mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{f}) \ldots$$

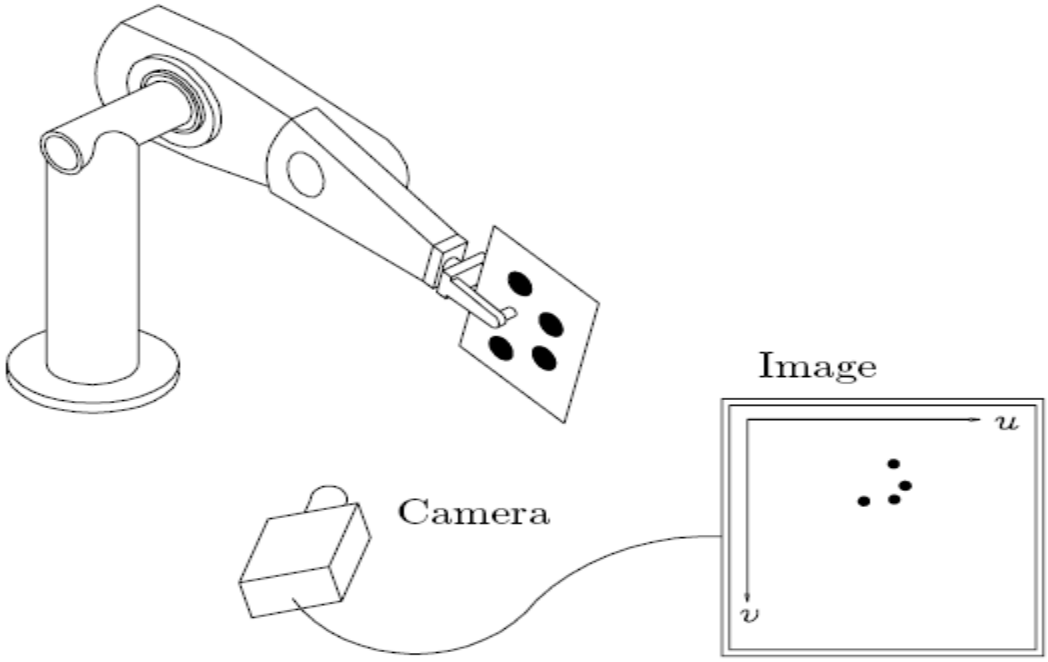


Fig 4.4. Robotic system: fixed camera

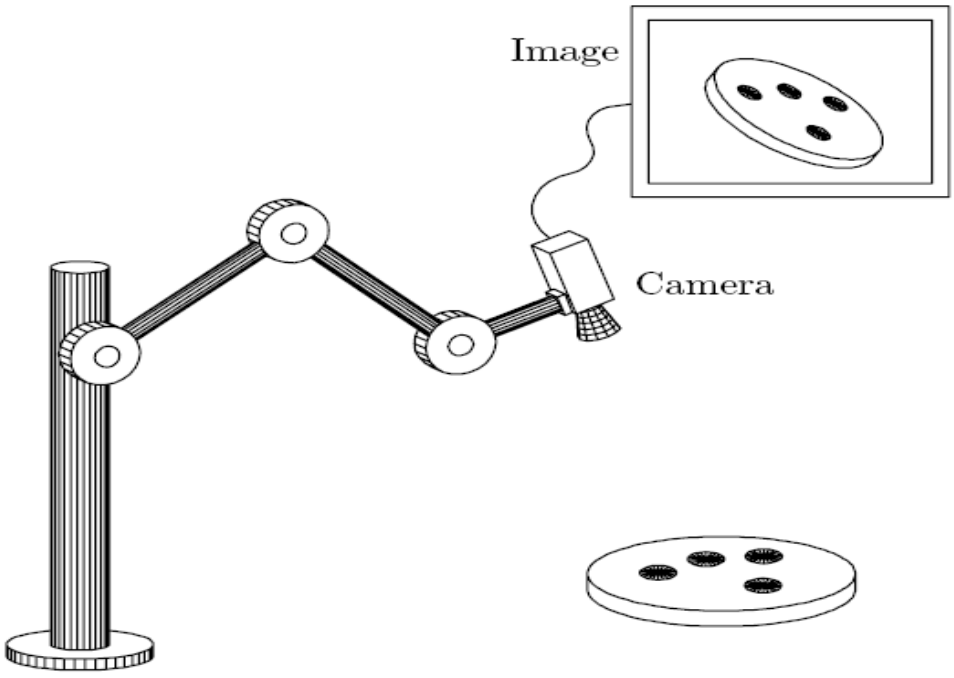


Fig 4.5. Robotic system: camera in hand

The block-diagram corresponding to the case when the outputs are the joint positions and velocities, that is,

$$y = y(q, \dot{q}, f) = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

while $\boldsymbol{\tau}$ is the input. In this case notice that for robots with n joints one has, in general, 2n outputs and n inputs

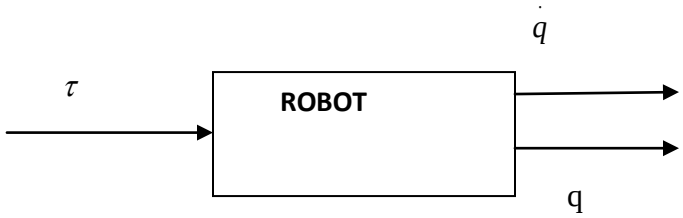


Fig 4.6. Input–output representation of a robot

4.2.2. Dynamic Model

At this stage, one determines the mathematical model which relates the input variables to the output variables. In general, such mathematical representation of the system is realized by ordinary differential equations. The system's mathematical model is obtained typically via one of the two following techniques.

• Analytical: this procedure is based on physical laws of the system's motion. This methodology has the advantage of yielding a mathematical model as precise as is wanted.
• Experimental: this procedure requires a certain amount of experimental data collected from the system itself. Typically one examines the system's behavior under specific input signals. The model so obtained is in general more imprecise than the analytic model since it largely depends on the inputs and the operating point1. However, in many cases it has the advantage of being much easier and quicker to obtain. On certain occasions, at this stage one proceeds

to a simplification of the system model to be controlled in order to design a relatively simple controller. Nevertheless, depending on the degree of simplification, this may yield malfunctioning of the overall controlled system due to potentially neglected physical phenomena. The ability of a control system to cope with errors due to neglected dynamics is commonly referred to as robustness. Thus, one typically is interested in designing robust controllers.

In other situations, after the modeling stage one performs the parametric identification. The objective of this task is to obtain the numerical values of different physical parameters or quantities involved in the dynamic model. The identification may be performed via techniques that require the measurement of inputs and outputs to the controlled system.

The dynamic model of robot manipulators is typically derived in the analytic form, that is, using the laws of physics. Due to the mechanical nature of robot manipulators, the laws of physics involved are basically the laws of mechanics. On the other hand, from a dynamical systems viewpoint, an n-DOF system may be considered as a multivariable nonlinear system. The term "multivariable" denotes the fact that the system has multiple (e.g. n) inputs (the forces and torques $\boldsymbol{\tau}$ applied to the joints by the electromechanical, hydraulic or pneumatic actuators) and, multiple (2n) state variables typically associated to the n positions $\mathbf{q}$, and n joint velocities $\dot{\mathbf{q}}$. In Figure 4.6 we depict the corresponding block-diagram assuming that the state variables also correspond to the outputs. we provide the specific dynamic model of a two-DOF prototype of a robot manipulator that we use to illustrate through examples, the performance of the controllers studied in the succeeding chapters. As was mentioned earlier, the dynamic models of robot manipulators are in general characterized by ordinary nonlinear and no autonomous differential equations. This fact limits considerably the use of control techniques tailored for linear systems, in robot control. In view of this and the present requirements of precision and rapidity of robot motion it has become necessary to use increasingly sophisticated control techniques. This class of control systems may include nonlinear and adaptive controllers.

4.2.3 Control Specifications

During this last stage one proceeds to dictate the desired characteristics for the control system through the definition of control objectives such as:

- stability;
- regulation (position control);
- trajectory tracking (motion control);
- Optimization.

The most important property in a control system, in general, is stability. This fundamental concept from control theory basically consists in the property of a system to go on working at a regime or closely to it for ever. Two techniques of analysis are typically used in the analytical study of the stability of controlled robots. The first is based on the so-called Lyapunov stability theory. The second is the so-called input–output stability theory. Both techniques are complementary in the sense that the interest in Lyapunov theory is the study of stability of the system using a state variables description, while in the second one; we are interested in the stability of the system from an input–output perspective. In this text we concentrate our attention on Lyapunov stability in the development and analysis of controllers. In accordance with the adopted definition of a robot manipulator's output **y**, the control objectives related to regulation and trajectory tracking receive special names. In particular, in the case when the output **y** corresponds to the joint position **q** and velocity $\dot{\mathbf{q}}$, we refer to the control objectives as "position control in joint coordinates" and "motion control in joint coordinates" respectively. Or we may simply say "position" and "motion" control respectively. The relevance of these problems motivates a more detailed discussion which is presented next.

4.3. Motion Control of Robot Manipulators

The simplest way to specify the movement of a manipulator is the so-called "point-to-point" method. This methodology consists in determining a series of points in the manipulator's workspace, which the end-effectors is required to pass through (cf. Figure 4.7). Thus, the position control problem consists in making the end-effectors go to a specified point regardless of the trajectory followed from its initial configuration.

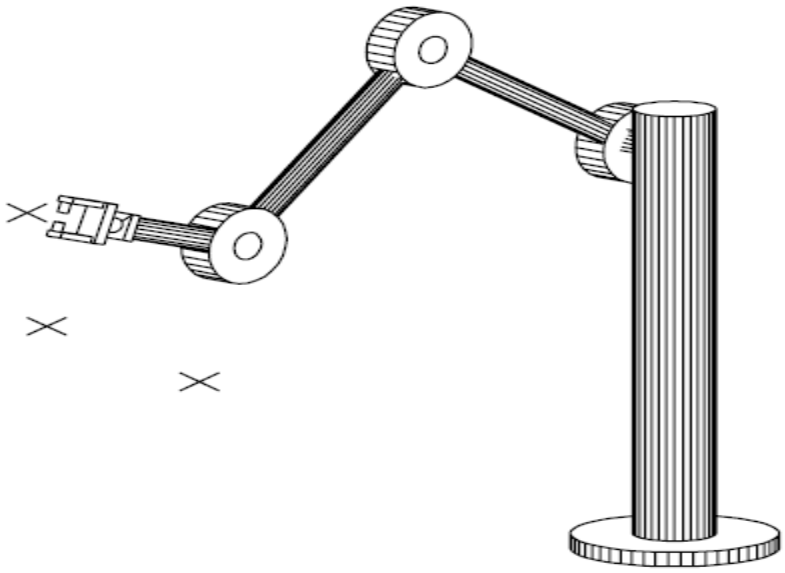


Fig 4.7. Point-to-point motion specification

A more general way to specify a robot's motion is via the so-called (continuous) trajectory. In this case, a (continuous) curve, or path in the state space and parameterized in time, is available to achieve a desired task. Then, the motion control problem consists in making the end-effector follow this trajectory as closely as possible (cf. Figure 4.8). This control problem, whose study is our central objective, is also referred to as trajectory tracking control. Let us briefly recapitulate a simple formulation of robot control which, as a matter of fact, is a particular case of motion control; that is, the position control problem. In this formulation the specified trajectory is simply a point in the workspace (which may be

translated under appropriate conditions into a point in the joint space). The position control problem consists in driving the manipulator's end-effector (resp. the joint variables) to the desired position, regardless of the initial posture. The topic of motion control may in its turn, be fitted in the more general framework of the so-called robot navigation. The robot navigation problem consists in solving, in one single step, the following sub problems:.

- path planning;
- trajectory generation;
- control design

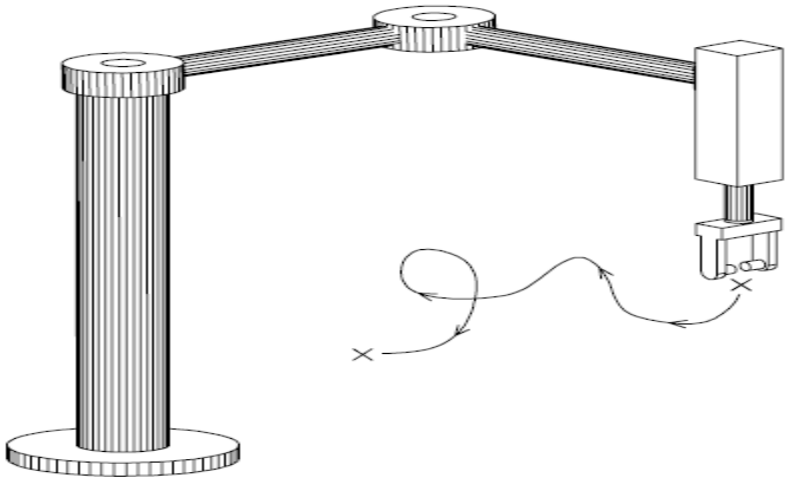


Fig 4.8. Trajectory motion specification

Path planning consists in determining a curve in the state space, connecting the initial and final desired posture of the end-effectors, while avoiding any obstacle. Trajectory generation consists in parameterizing in time the so obtained curve during the path planning. The resulting time-parameterized trajectory which is commonly called the reference trajectory, is obtained primarily in terms of the coordinates in the workspace. Then, following the so called method of inverse kinematics one may obtain a time-parameterized trajectory for the joint coordinates. The control design consists in solving the control problem mentioned

above. The main interest of this is the study of motion controllers and more particularly, the analysis of their inherent stability in the sense of Lyapunov. Therefore, we assume that the problems of path planning and trajectory generation are previously solved The dynamic models of robot manipulators possess parameters which depend on physical quantities such as the mass of the objects possibly held by the end-effector. This mass is typically unknown, which means that the values of these parameters are unknown. The problem of controlling systems with unknown parameters is the main objective of the adaptive controllers. These owe their name to the addition of an adaptation law which updates on-line, an estimate of the unknown parameters to be used in the control law. This motivates the study of adaptive control techniques applied to robot control. In the past two decades a large body of literature has been devoted to the adaptive control of manipulators.

4.4 Conclusion

Robot manipulators are created from a sequence of link and joint combinations. The links are the rigid members connecting the joints, or axes. The axes are the movable components of the robotic manipulator that cause relative motion between adjoining links. The mechanical joints used to construct the manipulator consist of five principal types. Two of the joints are linear, in which the relative motion between adjacent links is non-rotational, and three are rotary types, in which the relative motion involves rotation between links. The arm-and-body section of robotic manipulators is based on one of four configurations. Each of these anatomies provides a different work envelope and is suited for different applications.

- **Gantry** - These robots have linear joints and are mounted overhead.
- **Cylindrical -** Named for the shape of its work envelope, cylindrical anatomy robots are fashioned from linear joints that connect to a rotary base joint.
- **Polar -** The base joint of a polar robot allows for twisting and the joints are a combination of rotary and linear types.
- **Jointed-Arm -** This is the most popular industrial robotic configuration. The arm connects with a twisting joint, and the links within it are connected with rotary joints. It is also called an articulated robot.

CHAPTER 5

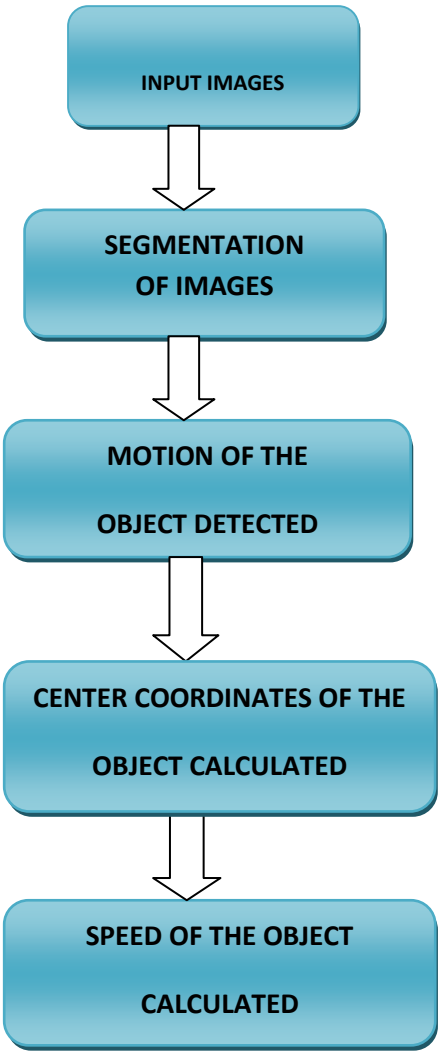
SYSTEM DESIGN

The proposed system is designed basically to measure the speed of a moving object in terms of pixels per frame using multiple images clicked by the camera.

5.1. Introduction to System Algorithm

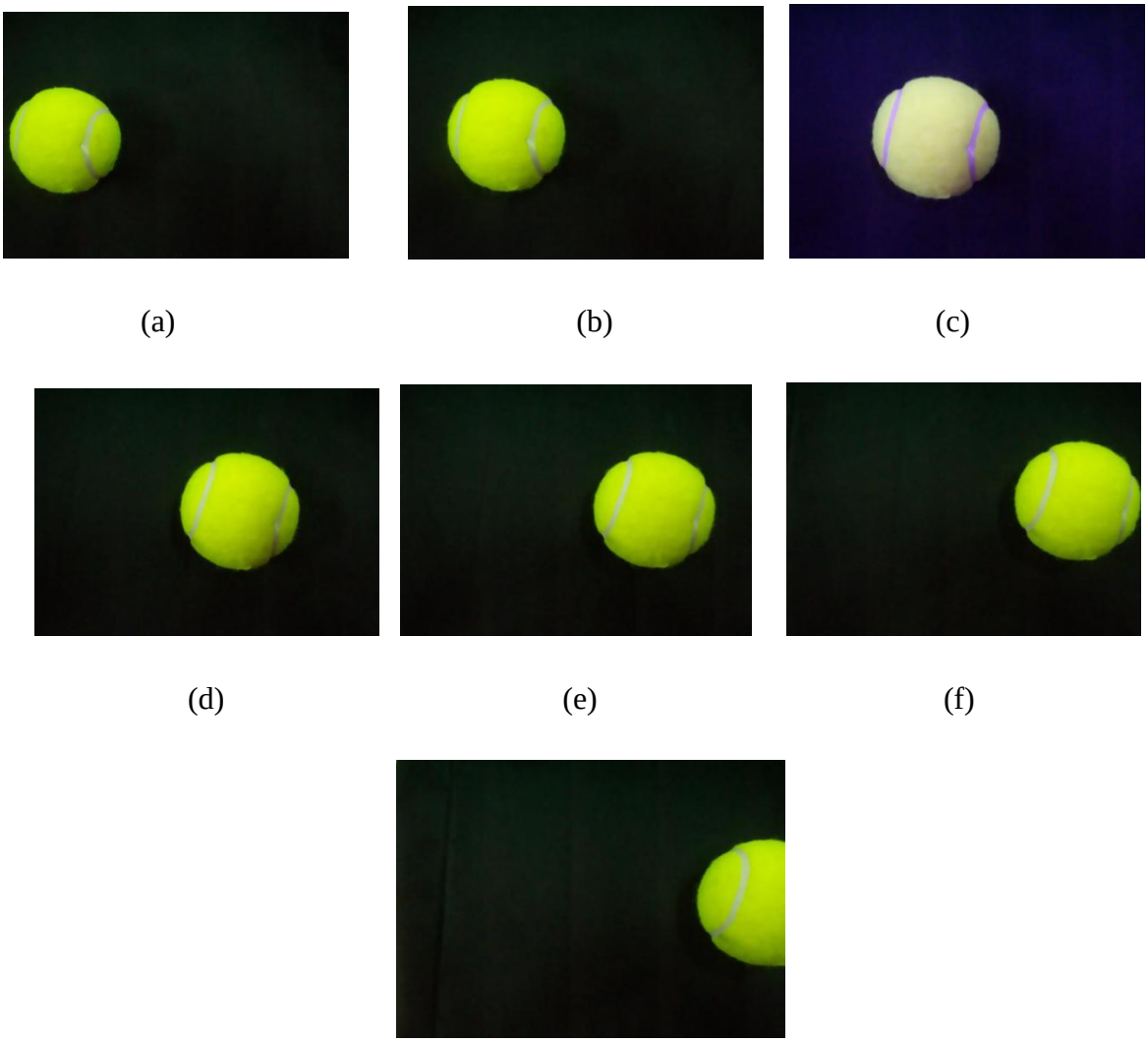
The working of the system basically comprises of 5 steps:

1. Collecting the input images
2. Segmentation of images
3. Motion direction of the object detected
4. Center coordinates of the object calculated
5. Using these coordinates the speed of the object calculated



5.1.1 Collecting the input images

The input given to the system is a collection of images taken by the camera of the moving object. The camera is positioned such that it captures the images of the object at regular intervals when it is moving. This set of images is fed to the system. A sample of the input images is shown below.



(a) (b) (c)

(d) (e) (f)

(g)

Fig 5.1 Images of a ball in different positions

5.1.2 Segmentation of images

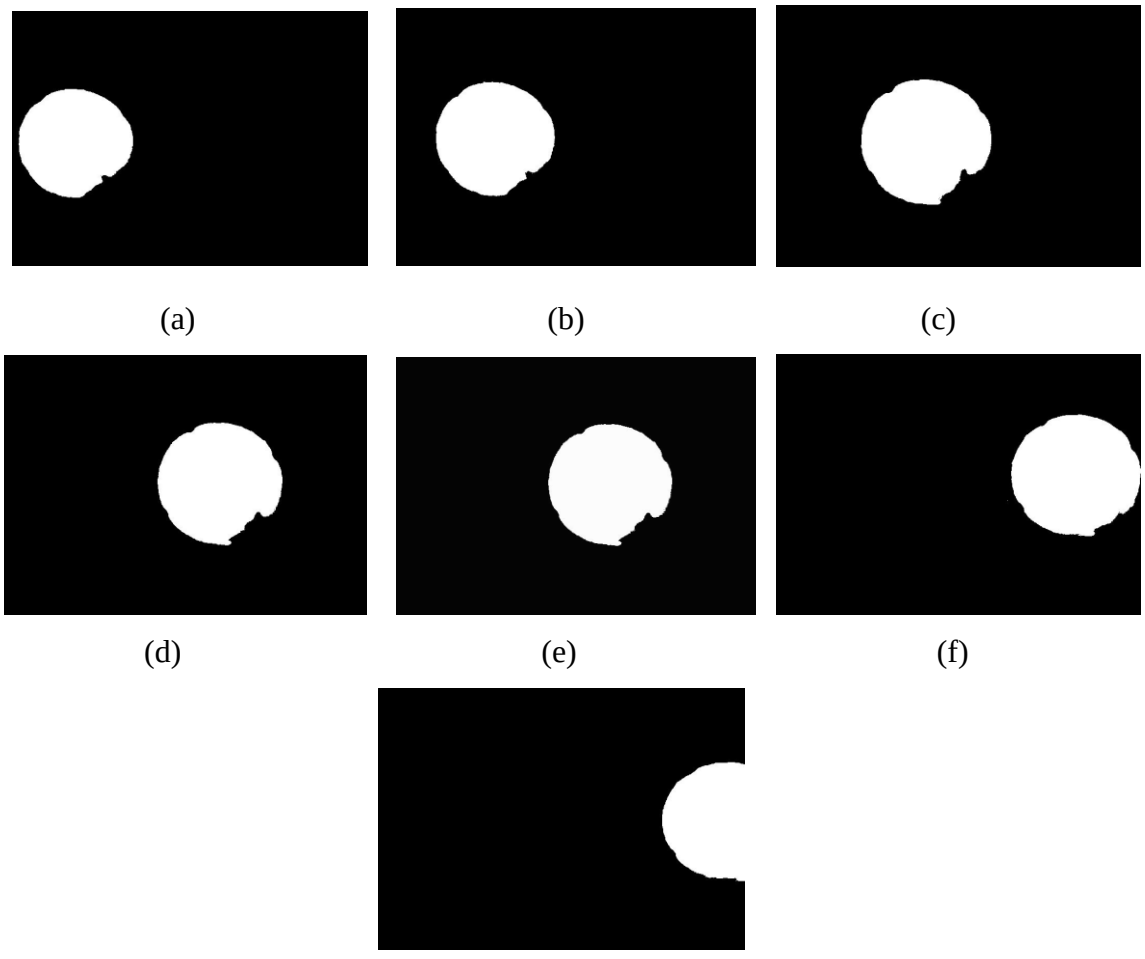
Segmentation refers to the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

After acquiring the set of input images segmentation was done on them in order to identify the desired object. The following code was used for the segmentation of the input images.

```
img=imread('C:\Users\Siki\Desktop\images\s7.jpg');
img=rgb2gray(img);
img=double(img);
thres_new=120;
thres=0;
[r c]=size(img);
while(thres~=thres_new)
    m1=0;
    m2=0;
    thres=thres_new;
    for i=1:r
        for j=1:c
            if(img(i,j)>=thres)
                m1=m1+img(i,j);

            else
                m2=m2+img(i,j);
            end
        end
    end
    [x y]=size(find(img>=thres));
```

```
    m1_len=x;
    [x y]=size(find(img<thres));
    m2_len=x;
    avg=((m1/m1_len)+(m2/m2_len))/2;
    thres_new=avg;
end
img_thresh=img>thres_new;
subplot(2,1,1)
imshow(uint8(img))
subplot(2,1,2)
imshow(img_thresh)
imwrite(img_thresh,'C:\Users\Siki\Desktop\images\ss7.jpg')
```

The above mentioned code was applied on **e**ach of the input images shown in figure 5.1 and the corresponding outputs are shown below.



(a) (b) (c)

(d) (e) (f)

(g)

Fig 5.2. Segmented images

5.1.3 Motion direction of the object detected

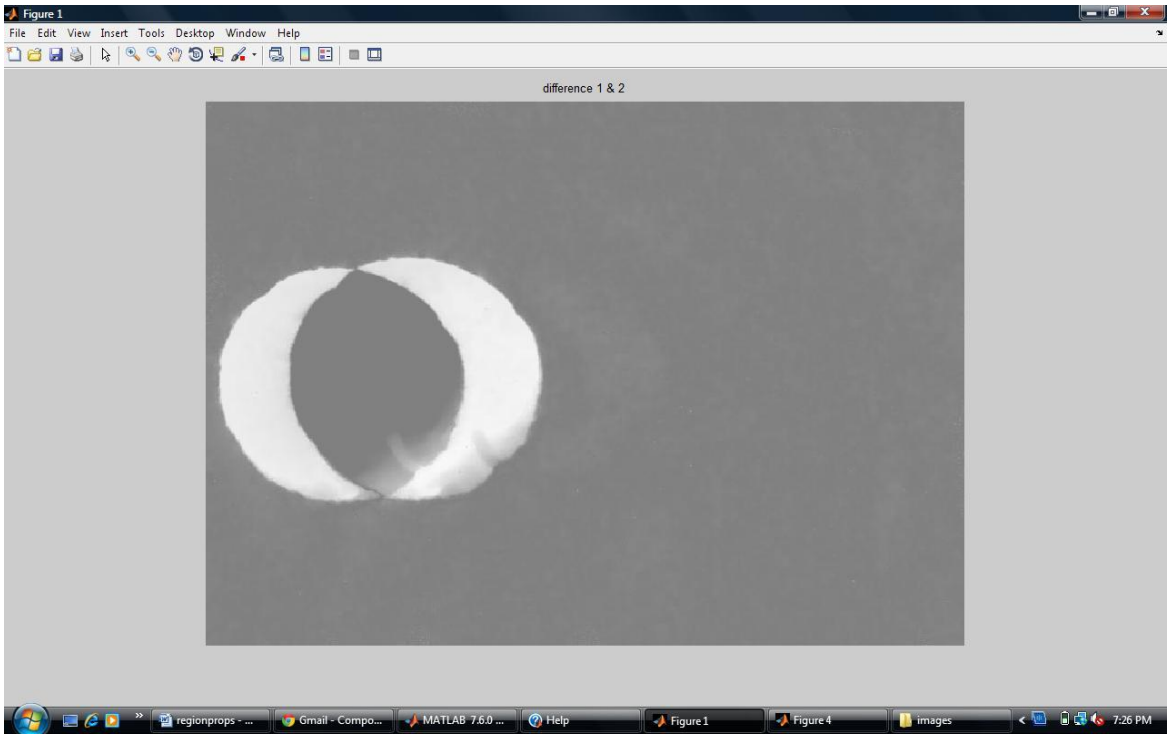
Once the input images are segmented the direction of motion is found and depicted in pictorial form. All the segmented images are combined to generate a combined image showing the motion of the object. The following code was used to perform this task.

```
tic;            %Counter start
t=29;           % threshold value

% n=13;
% j=n;
% img1=imread(strcat('C:\Users\Siki\Desktop\b1.jpg',num2str(j),'.jpg'));
img1=imread('C:\Users\Siki\Desktop\images\input images\s1.jpg');
img1=rgb2gray(img1);
%imshow(img1);


% img2=imread(strcat('C:\Users\Siki\Desktop\b2.jpg',num2str(j+1),'.jpg'));
img2=imread('C:\Users\Siki\Desktop\images\input images\s2.jpg');
img2=rgb2gray(img2);
%figure,imshow(img2);

%diff1=img2-img1;
diff21=abs(int8(img2)-int8(img1));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
                      %threshold.
figure,imshow(diff21);title('difference 1 & 2');
%imhist(diff1);
[r1,c1]=size(diff21);
for x1=1:r1
    for y1=1:c1
        if diff21(x1,y1)>t
            diff21(x1,y1)=255;
        else diff21(x1,y1)=0;
        end
    end
end

%  j=n+1;
%  img3=img2;
%
img3=imread(strcat('C:\Users\Siki\Desktop\b3.jpg',num2str(j),'.jpg'));
img3=imread('C:\Users\Siki\Desktop\images\input images\s3.jpg');
img3=rgb2gray(img3);

 diff31=abs(int8(img3)-int8(diff21));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
```

```
                    %threshold.
figure,imshow(diff31);title('difference 21 & 3');
%imhist(diff1);
[r2,c2]=size(diff31);
for x1=1:r2
    for y1=1:c2
        if diff31(x1,y1)>t
            diff31(x1,y1)=255;
        else diff31(x1,y1)=0;
        end
    end
end
% figure,imshow(diff31);title('difference 21 & 3');
% img4=imread(strcat('C:\Users\Siki\Desktop\b4.jpg',num2str(j+1),'.jpg'));
img4=imread('C:\Users\Siki\Desktop\images\input images\s4.jpg');
img4=rgb2gray(img4);

%diff2=img4-img3;
diff41=abs(int8(img4)-int8(diff31));
figure,imshow(diff41);title('difference 4 & 31');
%figure, imshow(diff2)

[r3,c3]=size(diff41);
for x2=1:r3
    for y2=1:c3
        if diff41(x2,y2)>t
            diff41(x1,y1)=255;
        else diff41(x1,y1)=0;
        end
    end
end
 %
img4=imread(strcat('C:\Users\Siki\Desktop\b4.jpg',num2str(j+1),'.jpg'));
img5=imread('C:\Users\Siki\Desktop\images\input images\s5.jpg');
img5=rgb2gray(img5);
diff51=abs(int8(img5)-int8(diff41));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
                      %threshold.
figure,imshow(diff51);title('difference 5 & 41');
%imhist(diff1);
[r4,c4]=size(diff51);
for x1=1:r4
    for y1=1:c4
        if diff51(x1,y1)>t
            diff51(x1,y1)=255;
        else diff51(x1,y1)=0;
        end
    end
end
img6=imread('C:\Users\Siki\Desktop\images\input images\s6.jpg');
img6=rgb2gray(img6);
```

```
%diff2=img4-img3;
diff61=abs(int8(img6)-int8(diff51));
 %figure, imshow(diff61);
[r5,c5]=size(diff61);
for x2=1:r5
    for y2=1:c5

          if diff61(x2,y2)>t
            diff61(x1,y1)=255;
        else diff61(x1,y1)=0;
        end
    end
end
```

All the segmented images are used in the above code to generate the images showing the direction of the motion of the moving object. The figures given below show the result of the above mentioned code.



(a)

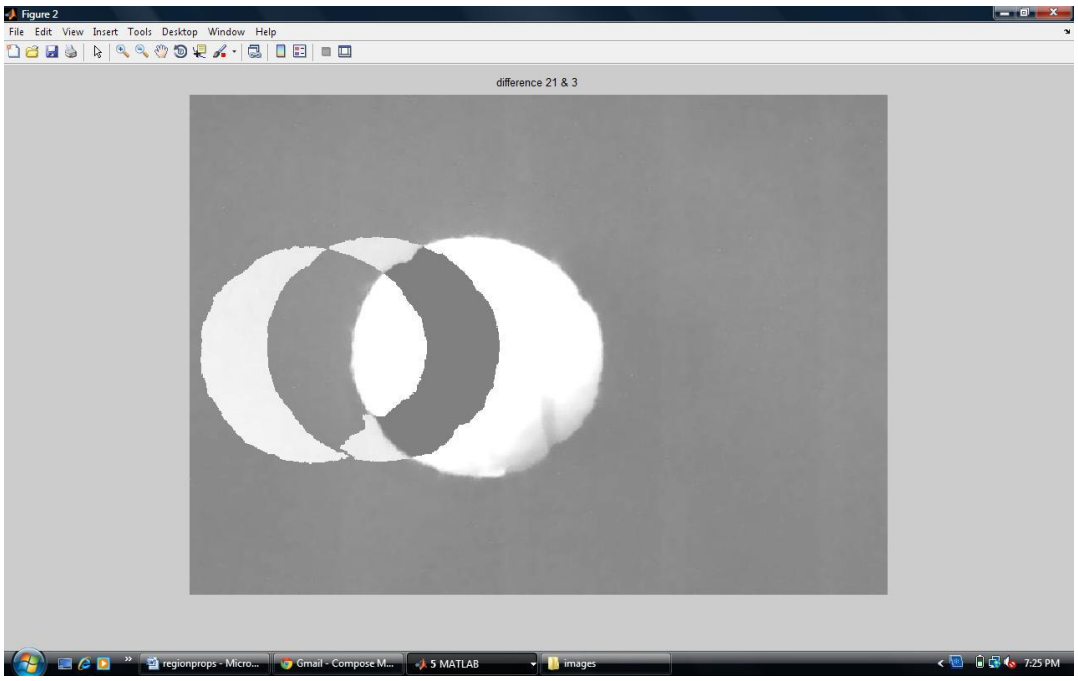


Figure 2
File Edit View Insert Tools Desktop Window Help
difference 21 & 3
regionprops - Micro...
Gmail - Compose M...
5 MATLAB
images
7:25 PM

(b)

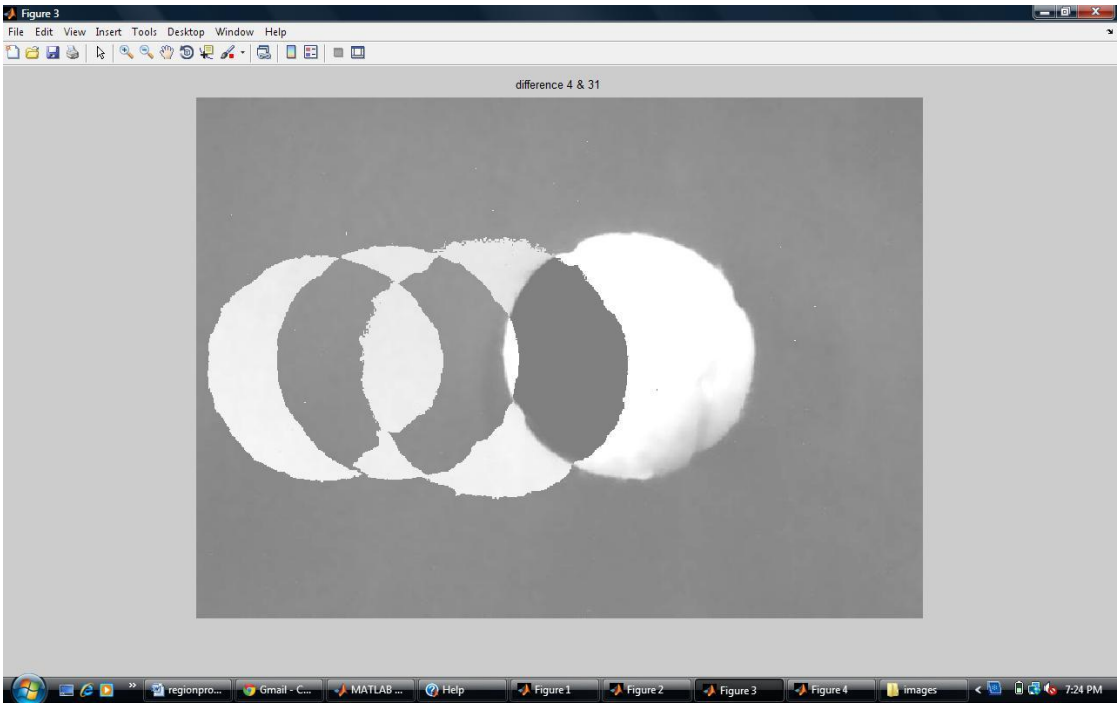
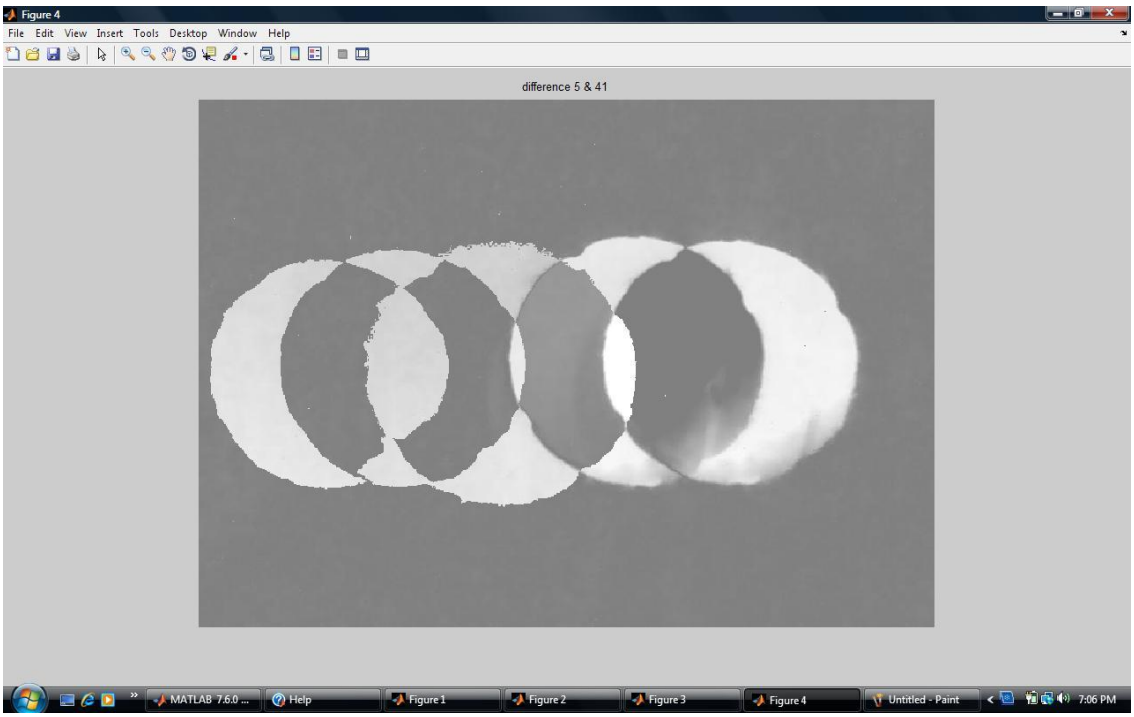


Figure 3
File Edit View Insert Tools Desktop Window Help
difference 4 & 31
regionpro...
Gmail - C...
MATLAB ...
Help
Figure 1
Figure 2
Figure 3
Figure 4
images
7:24 PM

(c)



(d)

Fig 5.3 Images depicting the motion of the object stepwise

This program output helps the user to identify the direction of motion of the ball.

5.1.4 Calculating the center coordinates of the object

In the next step the segmented images are used to calculate the center coordinates of the object. The position of the centre in all the input images is used to calculate the speed of the object. In order to calculate the center coordinates we need to know the area of the region of interest in the image which is the object in this case. Therefore first the area of the object should be known to the system. This is done by using an image of the required object beforehand. The code used to calculate the area of the object (ball in this case) is given below.

```
b=imread('C:\Users\Siki\Desktop\images\segmented
images\ss1.jpg');
%b=rgb2gray(b);
```

```
figure,imshow(b)
%b=im2bw(b);
%ibw=b<60;
%figure,imshow(ibw)
l=bwlabel(b);
%figure,imshow(ibw)
ball_area=regionprops(l,'area')
```

Once the area is known to the system it can identify the region of interest in the complete image and processing can be done on the required region only. This reduces our calculations and makes it more precise and close to accurate. Now the next task is to calculate the center of this region in all the different images. All the 7 images are used in the function code that calculates the horizontal and the vertical coordinates of the centers of the object in all the different images. The code used to calculate these is given below.

```
% ball image ss1

b1=imread('C:\Users\Siki\Desktop\images\segmented
images\ss1.jpg');
%b=rgb2gray(b);
b1=im2bw(b1);
b1=double(b1);
figure,imshow(b1);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r1 c1]=size(b1);
pixval on
ball_area1=regionprops(b1,'area');
area_values1=[ball_area1.Area];
filterball1=find(((area_values1)>=37547)|((area_values1)<=3754
7));
ibw_filtered1=ismember(b1,filterball1);
l1=bwlabel(ibw_filtered1);
cen_value1=regionprops(l1,'Centroid');
ball_center1=[cen_value1.Centroid];
g1=ball_center1(1);

% ball image  ss2
```

```
b2=imread('C:\Users\Siki\Desktop\images\segmented
images\ss2.jpg');
%b=rgb2gray(b);
b2=im2bw(b2);
b2=double(b2);
figure,imshow(b2);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r2 c2]=size(b2);
pixval on
ball_area2=regionprops(b2,'area');
area_values2=[ball_area2.Area];
filterball2=find(((area_values2)>=40083)|((area_values2)<=4008
3));
ibw_filtered2=ismember(b2,filterball2);
l2=bwlabel(ibw_filtered2);
cen_value2=regionprops(l2,'Centroid');
ball_center2=[cen_value2.Centroid];
g2=ball_center2(1);

% ball image ss3

b3=imread('C:\Users\Siki\Desktop\images\segmented
images\ss3.jpg');
%b=rgb2gray(b);
b3=im2bw(b3);
b3=double(b3);
figure,imshow(b3);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r3 c3]=size(b3);
pixval on
ball_area3=regionprops(b3,'area');
area_values3=[ball_area3.Area];
filterball3=find(((area_values3)>=37547)|((area_values3)<=3754
7));
ibw_filtered3=ismember(b3,filterball3);
l3=bwlabel(ibw_filtered3);
cen_value3=regionprops(l3,'Centroid');
ball_center3=[cen_value3.Centroid];
g3=ball_center3(1);

%ball image ss4
```

```
b4=imread('C:\Users\Siki\Desktop\images\segmented
images\ss4.jpg');
%b=rgb2gray(b);
b4=im2bw(b4);
b4=double(b4);
figure,imshow(b4);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r4 c4]=size(b4);
pixval on
ball_area4=regionprops(b4,'area');
area_values4=[ball_area4.Area];
filterball4=find(((area_values4)>=43692)|((area_values4)<=4369
2));
ibw_filtered4=ismember(b4,filterball4);
l4=bwlabel(ibw_filtered4);
cen_value4=regionprops(l4,'Centroid');
ball_center4=[cen_value4.Centroid];
g4=ball_center4(1);

% ball image ss5

b5=imread('C:\Users\Siki\Desktop\images\segmented
images\ss5.jpg');
%b=rgb2gray(b);
b5=im2bw(b5);
b5=double(b5);
figure,imshow(b5);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r5 c5]=size(b5);
pixval on
ball_area5=regionprops(b5,'area');
area_values5=[ball_area5.Area];
filterball5=find(((area_values5)>=43491)|((area_values5)<=4349
1));
ibw_filtered5=ismember(b5,filterball5);
l5=bwlabel(ibw_filtered5);
cen_value5=regionprops(l5,'Centroid');
ball_center5=[cen_value5.Centroid];
g5=ball_center5(1);
```

```
%ball image ss6

b6=imread('C:\Users\Siki\Desktop\images\segmented
images\ss6.jpg');
%b=rgb2gray(b);
b6=im2bw(b6);
b6=double(b6);
figure,imshow(b6);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r6 c6]=size(b6);
pixval on
ball_area6=regionprops(b6,'area');
area_values6=[ball_area6.Area];
filterball6=find(((area_values6)>=46092)|((area_values6)<=4609
2));
ibw_filtered6=ismember(b6,filterball6);
l6=bwlabel(ibw_filtered6);
cen_value6=regionprops(l6,'Centroid');
ball_center6=[cen_value6.Centroid];
g6=ball_center6(3);

%ball image ss7

b7=imread('C:\Users\Siki\Desktop\images\segmented
images\ss7.jpg');
%b=rgb2gray(b);
b7=im2bw(b7);
b7=double(b7);
figure,imshow(b7);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r7 c7]=size(b7);
pixval on
ball_area7=regionprops(b7,'area');
area_values7=[ball_area7.Area];
filterball7=find(((area_values7)>=29999)|((area_values7)<=2999
9));
ibw_filtered7=ismember(b7,filterball7);
l7=bwlabel(ibw_filtered7);
cen_value7=regionprops(l7,'Centroid');
ball_center7=[cen_value7.Centroid];
g7=ball_center7(3);
```

The above program gives us a set of 14 values i.e. x-y coordinates of the center point of each image. These values will further be used in the speed calculation of the object.

5.1.5. Speed of the object calculated

In this step we extract 7 out of the 14 values calculated in the previous step which correspond to the direction of motion of the object. In this case as we can see the ball is moving in the horizontal direction therefore we extract the coordinates which correspond to the position of the point in horizontal axis. These 7 points are marked as g1, g2, g3,g4, g5, g6 and g7. Further the difference between the positions of the ball in consecutive frames is calculated. These differences are named as df1, df2, df3, df4, df5 and df6. These values are defined as:

df1=g2-g1

df2=g3-g2

df3=g4-g3

df4=g5-g4

df5=g6-g5

df6=g7-g6

Further the average of these differences is calculated. This gives us the average speed of the object in pixels per frame. The code used for this is given below:

```
%  speed calculation
df1=g2-g1;
df2=g3-g2;
df3=g4-g3;
df4=g5-g4;
df5=g6-g5;
df6=g7-g6;
avgspeed=(df1+df2+df3+df4+df5+df6)/6
```

Conclusion

The algorithm proposed in this thesis has given good results. We have used a tennis ball as an object under test. A set of seven images was taken of the ball in different positions. When the motion detection code was run a clear image depicting the direction of motion of the ball was obtained. Further the area code was executed followed by code for speed calculation. The area of the ball was found to be 43692 pixels. Also the coordinate of the centre of the ball, responsible for the movement in the horizontal direction was found. The values obtained were 122.7074, 190.9012, 285.4424, 414.0403, 504.5932, 575.0129 and 631.9534. For the given set of images the speed was calculated as 84.8743 pixels per frame.

Future Work

This algorithm when refined a bit can become a part of the object tracking system. When used with a very high precision camera the time interval between the click of consecutive images can be reduced and thus accuracy increased. Further a feed back system can be designed which can give a control signal to the camera attached to the end manipulator and guide it appropriately to follow the object. Both the direction and speed of the manipulator can be controlled for the efficient functioning.

References

[1] F. Chaumette and B. Espiau. A new approach to visual servoing in robotics. IEEE Transactions on Robotics and Automation, 8(3):313–327, June 1992.

[2] W. J. Wilson, C. C. W. Hulls, and G. S. Bell. Relative end effectors control using Cartesian position based visual servoing. IEEE Transactions on Robotics and Automation, 12(5):684–696, October 1996.

[3] S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. IEEE Transactions on Robotics and Automation, 12(5):651–670, October 1996

[4] E. Malis and F. Chaumette. 2 1/2d visual serving with respect to unknown objects through a new estimation scheme of camera displacement. International Journal of Computer Vision, 37(1):79–97, June 2000.

[5] P. Rives. Visual servoing based on epipolar geometry. IEEE/RSJ International Conference on Intelligent Robots and Systems, 1:602–607, November 2000.

[6] C. J. Taylor, J. P. Ostrowski, and S.-H. Jung. Robust visual servoing based on relative orientation. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2:574–580, 1999.

[7] J. Feddema and O. Mitchell, "Vision-guided serving with feature based trajectory generation," IEEE Trans. Robot. Automat. vol. 5, pp. 691–700, Oct. 1989.

[8] L. Weiss, A. Sanderson, and C. Neumann, "Dynamic sensor-based control of robots with visual feedback," IEEE J. Robot. Automat. vol. 3, pp. 404–417, Oct. 1987.

[9] D. Forsyth and J. Ponce, Computer Vision: A Modern Approach. Upper Saddle River, NJ: Prentice Hall, 2003. [10] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, An Invitation to 3-D Vision: From Images to Geometric Models. New York: Springer-Verlag, 2003.

[11] H. Michel and P. Rives, "Singularities in the determination of the situation of a robot effectors from the perspective view of three points," INRIA Research Report, Tech. Rep. 1850, Feb. 1993.

[12] M. Fischler and R. Boles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” Communicate. ACM, vol. 24, pp. 381–395, June 1981

[13] S. Hutchinson, G. Hager, and P. Corke, “A tutorial on visual servo control,” IEEE Trans. Robot. Automat. vol. 12, pp. 651–670, Oct. 1996.

[14] B. Espiau, F. Chaumette, and P. Rives, “A new approach to visual servoing in robotics,” IEEE Trans. Robotics and Automation, vol. 8, pp. 313–326, June 1992.

[15] E. Malis, “Improving vision-based control using efficient second-order minimization techniques,” in Proc. IEEE Int. Conf. Robot. Automat., pp. 1843–1848, Apr. 2004.

[16] P. Corke and S. Hutchinson, “A new partitioned approach to image based visual servo control,” IEEE Trans. Robot. Automat, vol. 17, no. 4, pp. 507–515, 2001.

[17] F. Chaumette, “Potential problems of stability and convergence in image-based and position-based visual servoing,” in The Confluence of Vision and Control, vol. 237, Lecture Notes in Control and Information Sciences, D. Kriegman, G. Hager, and S. Morse, Eds. New York: Springer- Verlag, 1998, pp. 66–78.

[18] G. Hager, W. Chang, and A. Morse, “Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination,” IEEE Control Syst. Mag., vol. 15, pp. 30–39, Feb. 1995.

[19] R. Paul, Robot Manipulators: Mathematics, Programming and Control. Cambridge, MA: MIT Press, 1982.

[20] W. J. Wilson, C. C. W. Hulls, and G. S. Bell. Relative end effectors control using cartesian position based visual servoing. IEEE Transactions on Robotics and Automation, 12(5):684–696, October 1996.

[21] E. Malis, F. Chaumette, and S. Boudet. 2 1/2d visual servoing. IEEE Transactions on Robotics and Automation, 15(2):238–250, April 1999.

[22] Junaed Sattar, Philippe Giguere, Gregory Dudek and Chris Prahacs Centre for Intelligent Machines McGill University

[23] P. I. Corke. Visual control of robot manipulators – a review. In Visual Servoing, pages 1–31. World Scientific, 1993.

[24] W. Jang and Z. Bien. Feature-based visual serving of an eye-in-hand robot with improved tracking performance. In Proc. of IEEE Int. Conf. on Robotics and Automation, pages 2254–2260, 1991.

[25] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura. Manipulator control with image-based visual servo. In Proc. of IEEE Int. Conf. on Robotics and Automation, pages 2267–2272, 1991.

[26] N. Maru, H. Kase, et al. Manipulator control by visual servoing with the stereo vision. In Proc. of the 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 1865–1870, 1993.

[27] P. Allen, A. Timcenko, B. Yoshimi, and P. Michel man. Automated tracking and grasping of a moving object with a robotic hand-eye system. IEEE Trans. On Robotics and Automation, RA–9(2):152–165, 1993.

[28] A. Castano and S. Hutchinson. Visual compliance: Task-derected visual servo control. IEEE Trans. on Robotics and Automation, 10(3):334–342, 1994.

[29] G. D. Hager, W.-C. Cang and A. S. Morse. Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination. In Proc. of IEEE Int. Conf. on Robotics and Automation, pages 2850–2856, 1994.

[30] L. E. Weiss, A. C. Sanderson, and C. P. Neumann. Dynamic sensor-based control of robots with visual feedback. IEEE J. of Robotics and Automation, RA- 3(5):404–417, 1987.

[31] J. T. Feddema and C. S. G. Lee. Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera. IEEE Trans. on System, Man, and Cybernetics, 20(5):1172–1183, 1990.

[32] N. P. Papanikolopoulos and P. K. Khosla. Adaptive robotic visual tracking: Theory and experiments. IEEE Trans. on Automatic Control, 38(3):429–445, 1993.

[33] B. Nelson, N. P. Papanikolopoulos, and P. K. Khosla. Visual servoing for robotic assembly. In Visual Servoing, pages 139–164. World Scientific, 1993.

[34] N. P. Papanikolopoulos, B. Nelson, and P. K. Khosla. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. In Proc. of IEEE Int. Conf. on Robotics and Automation, pages 174–179, 1994. [34] B. H. Yoshimi and P. K. Allen. Alignment using an

uncelebrated camera system. IEEE Trans. on Robotics and Automation, 11(4):516–521, 1995.

[35] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. IEEE Trans. on Robotics and Automation, 5(3):345–358, 1989.

[36] C. Brown. Gaze controls with interactions and delays. IEEE Trans. on System, Man, and Cybernetics, 20(1):518–527, 1990.

[37] E. D. Dickmanns, B. Mysliwetz, and T. Christians. An integrated spatiotemporal approach to automatic visual guidance of autonomous vehicles. IEEE Trans. on System, Man, and Cybernetics, 20(6):1273–1284, 1990.

[38] W. J. Wilson. Visual servo control of robots using kalman filter estimates of robot pose relative to work-pieces. In Visual Servoing, pages 71–104. World Scientific, 1993. [39] K. Hosoda, K. Sakamoto, and M. Asada. Trajectory generation for obstacle avoidance of uncelebrated stereo visual servoing without 3d reconstruction. In Proc. of the 1995 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 29–34, 1995.

[40] P. Eykhoff. System Identification, chapter 7. John Wiley & Sons Ltd., 1974.

[41] M. Inaba, T. Kamata, and H. Inoue. Rope handling by mobile hand-eye robots. In Proc. of Int. Conf. on Advanced Robotics, pages 121–126, 1993.

[42] Koh Hosoda, Katsuji Igarashi, and Minoru Asada. Adaptive hybrid visual servoing/ force control in unknown environment. In Proc. of the 1996 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 1097–1103, 1996.

[43] K. Hosoda, M. Kamado, and M. Asada. Vision-based servoing control for legged robots. In Proc. of IEEE Int. Conf. on Robotics and Automation, pages 3154– 3159, 1997.

[44] Junaed Sattar, Philippe Giguere, Gregory Dudek and Chris Prahacs Centre for Intelligent Machines. A Visual Servoing System for an Aquatic Swimming Robot,2005 McGill University 3480 University Street, Montreal, QC, H3A 2A7, Canada

[45] Koh Hosoda and Minoru Asada Dept. of Adaptive Machine Systems, Osaka University Yamadaoka 2–1, Suita 565, Japan, Page(s): 778 - 784 vol.21997

[46] Chris Gaskett, Luke Fletcher and Alexander Zelinsky Robotic Systems Laboratory Department of Systems Engineering

[47] XIAO-JING SHEN, JUN-MIN PAN Department of Automation, Shanghai JiaoTong University

[48] Ogata K., "Modern control engineering", Prentice-Hall, 1970.

[49] Spong M., Vidyasagar M., "Robot dynamics and control", John Wiley and Sons, Inc. Various nonlinear models of friction for DC motors are presented in 1989.

Motion detection code

```
function motion_detection
                %program for motion detection using three image difference
                %method. the image difference of Nth and N+1th is ANDed
                %with image image difference N+1th and N+2nd. Thresholding
                %is used. Refer IEEE paper "Research on Three Image
                %Difference Algorithm" for more details. %
                %last modified on 17/04/2011


tic;            %Counter start
t=29;           % threshold value

% n=13;
% j=n;
% img1=imread(strcat('C:\Users\Siki\Desktop\b1.jpg',num2str(j),'.jpg'));
img1=imread('C:\Users\Siki\Desktop\images\input images\s1.jpg');
img1=rgb2gray(img1);
%imshow(img1);


% img2=imread(strcat('C:\Users\Siki\Desktop\b2.jpg',num2str(j+1),'.jpg'));
img2=imread('C:\Users\Siki\Desktop\images\input images\s2.jpg');
img2=rgb2gray(img2);
%figure,imshow(img2);

%diff1=img2-img1;
diff21=abs(int8(img2)-int8(img1));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
                      %threshold.
figure,imshow(diff21);title('difference 1 & 2');
%imhist(diff1);
[r1,c1]=size(diff21);
for x1=1:r1
    for y1=1:c1
        if diff21(x1,y1)>t
            diff21(x1,y1)=255;
        else diff21(x1,y1)=0;
        end
    end
end
```

```
%  j=n+1;
%  img3=img2;
%
img3=imread(strcat('C:\Users\Siki\Desktop\b3.jpg',num2str(j),'.jpg'));
img3=imread('C:\Users\Siki\Desktop\images\input images\s3.jpg');
img3=rgb2gray(img3);

 diff31=abs(int8(img3)-int8(diff21));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
                      %threshold.
figure,imshow(diff31);title('difference 21 & 3');
%imhist(diff1);
[r2,c2]=size(diff31);
for x1=1:r2
    for y1=1:c2
        if diff31(x1,y1)>t
            diff31(x1,y1)=255;
        else diff31(x1,y1)=0;
        end
    end
end
% figure,imshow(diff31);title('difference 21 & 3');
% img4=imread(strcat('C:\Users\Siki\Desktop\b4.jpg',num2str(j+1),'.jpg'));
img4=imread('C:\Users\Siki\Desktop\images\input images\s4.jpg');
img4=rgb2gray(img4);

%diff2=img4-img3;
diff41=abs(int8(img4)-int8(diff31));
figure,imshow(diff41);title('difference 4 & 31');
%figure, imshow(diff2)

[r3,c3]=size(diff41);
for x2=1:r3
    for y2=1:c3
        if diff41(x2,y2)>t
            diff41(x1,y1)=255;
        else diff41(x1,y1)=0;
        end
    end
end
 %
img4=imread(strcat('C:\Users\Siki\Desktop\b4.jpg',num2str(j+1),'.jpg'));
img5=imread('C:\Users\Siki\Desktop\images\input images\s5.jpg');
img5=rgb2gray(img5);
diff51=abs(int8(img5)-int8(diff41));
%maxx=max(max(diff1)); %maximum val of the difference. not used in
                      %the algorithm. could be used for selecting
                      %threshold.
figure,imshow(diff51);title('difference 5 & 41');
%imhist(diff1);
[r4,c4]=size(diff51);
for x1=1:r4
```

```
    for y1=1:c4
        if diff51(x1,y1)>t
            diff51(x1,y1)=255;
        else diff51(x1,y1)=0;
        end
    end
end
img6=imread('C:\Users\Siki\Desktop\images\input images\s6.jpg');
img6=rgb2gray(img6);

%diff2=img4-img3;
diff61=abs(int8(img6)-int8(diff51));
 %figure, imshow(diff61);
[r5,c5]=size(diff61);
for x2=1:r5
    for y2=1:c5
        if diff61(x2,y2)>t
            diff61(x1,y1)=255;
        else diff61(x1,y1)=0;
        end
    end
end
%figure, imshow(diff61);
%title('last image');

%e1=and(e,diff3);
%histogram(2,2,1),
%figure, imshow(img1);
%title('(N-1)th Image');
% histogram(2,2,2),
%figure, imshow(img2);
%title('Nth Image');
%histogram(2,2,3),
%figure, imshow(img4);
%title(('(N+1)th Image'));
%figure,imshow(img2);title('Nth Image');
%figure, imshow(e);
%figure,imshow(e1);
%title('Moving Object Detected');
```

Area of the object code

```
b=imread('C:\Users\Siki\Desktop\images\segmented images\ss1.jpg');
%b=rgb2gray(b);
figure,imshow(b)
%b=im2bw(b);
%ibw=b<60;
%figure,imshow(ibw)
l=bwlabel(b);
%figure,imshow(ibw)
ball_area=regionprops(l,'area')
```

Image segmentation code

```
img=imread('C:\Users\Siki\Desktop\images\s7.jpg');
img=rgb2gray(img);
img=double(img);
thres_new=120;
thres=0;
[r c]=size(img);
while(thres~=thres_new)
    m1=0;
    m2=0;
    thres=thres_new;
    for i=1:r
        for j=1:c
            if(img(i,j)>=thres)
                m1=m1+img(i,j);

            else
                m2=m2+img(i,j);
            end
        end
    end
    [x y]=size(find(img>=thres));
    m1_len=x;
    [x y]=size(find(img<thres));
    m2_len=x;
    avg=((m1/m1_len)+(m2/m2_len))/2;
    thres_new=avg;
end
img_thresh=img>thres_new;
subplot(2,1,1)
imshow(uint8(img))
subplot(2,1,2)
imshow(img_thresh)
imwrite(img_thresh,'C:\Users\Siki\Desktop\images\ss7.jpg')
```

Speed calculation of the object code

```
% ball image ss1

b1=imread('C:\Users\Siki\Desktop\images\segmented images\ss1.jpg');
%b=rgb2gray(b);
b1=im2bw(b1);
b1=double(b1);
figure,imshow(b1);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r1 c1]=size(b1);
```

```
pixval on
ball_area1=regionprops(b1,'area');
area_values1=[ball_area1.Area];
filterball1=find(((area_values1)>=37547)|((area_values1)<=37547));
ibw_filtered1=ismember(b1,filterball1);
l1=bwlabel(ibw_filtered1);
cen_value1=regionprops(l1,'Centroid');
ball_center1=[cen_value1.Centroid];
g1=ball_center1(1);

% ball image  ss2

b2=imread('C:\Users\Siki\Desktop\images\segmented images\ss2.jpg');
%b=rgb2gray(b);
b2=im2bw(b2);
b2=double(b2);
figure,imshow(b2);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r2 c2]=size(b2);
pixval on
ball_area2=regionprops(b2,'area');
area_values2=[ball_area2.Area];
filterball2=find(((area_values2)>=40083)|((area_values2)<=40083));
ibw_filtered2=ismember(b2,filterball2);
l2=bwlabel(ibw_filtered2);
cen_value2=regionprops(l2,'Centroid');
ball_center2=[cen_value2.Centroid];
g2=ball_center2(1);

% ball image ss3

b3=imread('C:\Users\Siki\Desktop\images\segmented images\ss3.jpg');
%b=rgb2gray(b);
b3=im2bw(b3);
b3=double(b3);
figure,imshow(b3);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r3 c3]=size(b3);
pixval on
ball_area3=regionprops(b3,'area');
area_values3=[ball_area3.Area];
filterball3=find(((area_values3)>=37547)|((area_values3)<=37547));
ibw_filtered3=ismember(b3,filterball3);
l3=bwlabel(ibw_filtered3);
cen_value3=regionprops(l3,'Centroid');
ball_center3=[cen_value3.Centroid];
g3=ball_center3(1);

%ball image ss4
```

```
b4=imread('C:\Users\Siki\Desktop\images\segmented images\ss4.jpg');
%b=rgb2gray(b);
b4=im2bw(b4);
b4=double(b4);
figure,imshow(b4);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r4 c4]=size(b4);
pixval on
ball_area4=regionprops(b4,'area');
area_values4=[ball_area4.Area];
filterball4=find((area_values4)>=43692)|((area_values4)<=43692));
ibw_filtered4=ismember(b4,filterball4);
l4=bwlabel(ibw_filtered4);
cen_value4=regionprops(l4,'Centroid');
ball_center4=[cen_value4.Centroid];
g4=ball_center4(1);

% ball image ss5

b5=imread('C:\Users\Siki\Desktop\images\segmented images\ss5.jpg');
%b=rgb2gray(b);
b5=im2bw(b5);
b5=double(b5);
figure,imshow(b5);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r5 c5]=size(b5);
pixval on
ball_area5=regionprops(b5,'area');
area_values5=[ball_area5.Area];
filterball5=find((area_values5)>=43491)|((area_values5)<=43491));
ibw_filtered5=ismember(b5,filterball5);
l5=bwlabel(ibw_filtered5);
cen_value5=regionprops(l5,'Centroid');
ball_center5=[cen_value5.Centroid];
g5=ball_center5(1);

%ball image ss6

b6=imread('C:\Users\Siki\Desktop\images\segmented images\ss6.jpg');
%b=rgb2gray(b);
b6=im2bw(b6);
b6=double(b6);
figure,imshow(b6);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r6 c6]=size(b6);
pixval on
```

```
ball_area6=regionprops(b6,'area');
area_values6=[ball_area6.Area];
filterball6=find(((area_values6)>=46092)|((area_values6)<=46092));
ibw_filtered6=ismember(b6,filterball6);
l6=bwlabel(ibw_filtered6);
cen_value6=regionprops(l6,'Centroid');
ball_center6=[cen_value6.Centroid];
g6=ball_center6(3);

%ball image ss7

b7=imread('C:\Users\Siki\Desktop\images\segmented images\ss7.jpg');
%b=rgb2gray(b);
b7=im2bw(b7);
b7=double(b7);
figure,imshow(b7);
%ibw=b<40;
%l=bwlabel(ibw);
%figure,imshow(ibw)
[r7 c7]=size(b7);
pixval on
ball_area7=regionprops(b7,'area');
area_values7=[ball_area7.Area];
filterball7=find(((area_values7)>=29999)|((area_values7)<=29999));
ibw_filtered7=ismember(b7,filterball7);
l7=bwlabel(ibw_filtered7);
cen_value7=regionprops(l7,'Centroid');
ball_center7=[cen_value7.Centroid];
g7=ball_center7(3);

%  speed calculation
df1=g2-g1;
df2=g3-g2;
df3=g4-g3;
df4=g5-g4;
df5=g6-g5;
df6=g7-g6;
avgspeed=(df1+df2+df3+df4+df5+df6)/6
```