VISUAL GUIDANCE OF ROBOT MOTION

This thesis is
presented to the
Department of Computer Science
for the degree of
Master of Science
of the
University of Western Australia

By
Li Fang Gu
October 1996

Abstract

Future robots are expected to cooperate with humans in daily activities. Efficient cooperation requires new techniques for transferring human skills to robots. This thesis presents an approach on how a robot can extract and replicate a motion by observing how a human instructor conducts it. In this way, the robot can be taught without any explicit instructions and the human instructor does not need any expertise in robot programming.

A system has been implemented which consists of two main parts. The first part is data acquisition and motion extraction. Vision is the most important sensor with which a human can interact with the surrounding world. Therefore two cameras are used to capture the image sequences of a moving rigid object. In order to compress the incoming images from the cameras and extract 3D motion information of the rigid object, feature detection and tracking are applied to the images. Corners are chosen as the main features because they are more stable under perspective projection and during motion. A reliable corner detector is implemented and a new corner tracking algorithm is proposed based on smooth motion constraints. With both spatial and temporal constraints, 3D trajectories of a set of points on the object can be obtained and the 3D motion parameters of the object can be reliably calculated by the algorithm proposed in this thesis.

Once the 3D motion parameters are available through the vision system, the robot should be programmed to replicate this motion. Since we are interested in smooth motion and the similarity between two motions, the task of the second part of our system is therefore to extract motion characteristics and to transfer these to the robot. It can be proven that the characteristics of a parametric cubic B-spline curve are completely determined by its control points, which can be obtained by

the least-squares fitting method, given some data points on the curve. Therefore a parametric cubic B-spline curve is fitted to the motion data and its control points are calculated. Given the robot configuration, the obtained control points can be scaled, translated, and rotated so that a motion trajectory can be generated for the robot to replicate the given motion in its own workspace with the required smoothness and similarity, although the absolute motion trajectories of the robot and the instructor can be different.

All the above modules have been integrated and results of an experiment with the whole system show that the approach proposed in this thesis can extract motion characteristics and transfer these to a robot. A robot arm has successfully replicated a human arm movement with similar shape characteristics by our approach.

In conclusion, such a system collects human skills and intelligence through vision and transfers them to the robot. Therefore, a robot with such a system can interact with its environment and learn by observation.

Acknowledgements

This thesis would not be possible without various forms of support from many people. First of all, I would like to thank my supervisor, Associate Professor, Robyn Owens, for her inspiration, guidance, constant support and encouragement throughout the course of this research work. Her useful suggestions on improving the contents and presentation of this thesis are also gratefully appreciated.

I am also grateful to many members of the Robvis research group for their suggestions and help. In particular, I would like to thank Dr. D. Huynh for allowing me to use her interactive program for picking up camera calibration points and many useful suggestions she offered me on depth calculation from two stereo images. Rameri Salama's help with taking image sequences of arm movement and controlling the robot arm along a given trajectory is very much appreciated. With E. J. Holden I had many stimulating discussions on feature tracking and she also offered me lots of moral support. I should also mention Peter Kovesi, who offered me assistance in various areas. Special thanks also go to Dr. N. Spadaccini, Ben Robbins, and Rajiv Ellepola for their careful reading of this thesis and useful comments.

Finally, I would like to thank my husband, Yong, and my daughter, Fan, for their love, understanding, and patience. I thank my parents for all the support they have given me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The long-term goal of research in artificial intelligence is to build intelligent robots that can see and learn, have goal-directed behavior, communicate with humans in natural language, and operate in real-time. This goal has been partly achieved by the progress made in several research fields, such as neural networks, computer vision, and robotics.

Human beings have five senses, of which vision is no doubt the most important form of obtaining information. Progress made in optics and electronics offers techniques of providing robots with visual sensors. Video cameras are the most common sensors used to obtain images of the environment of a robot, and they play a similar role to human eyes.

The human brain processes the images on the retinas and extracts the relevant information for performing a given task. From this information, the brain interprets the scene and makes a plan for performing that task. Subsequently the brain sends signals to the corresponding body parts to perform the task. To give a robot some degree of such intelligence, researchers in the field of computer vision are developing algorithms to extract useful information from images, understand and interpret the scene, plan the path for the robot and control the robot to perform a task [5,7,8,9].

However scene understanding and interpretation require much intelligence and even with the most advanced algorithms they might be too slow and too complex to be practically feasible. Fortunately scene understanding and interpretation are not always necessary to perform a task. For example, when children learn to swim,

they simply watch the swimming teacher's arm and leg movements and copy these movements without understanding the physics of floatation. In the following, we detail the advantages of the *teaching by showing and learning by watching* method for teaching a robot to perform a task.

1.1 Teaching by Showing

There are currently three types of robot systems which can perform a task in a certain environment. The first type of system uses a set of preprogrammed paths to accomplish some simple repetitive tasks. Spot welding in car assembly plants is one application of this type of robot system. The parts to be welded must be positioned accurately in front of the robot because the robot has no sensors to detect misalignment and is unable to adjust its path. This type of system cannot, therefore, consider a variety of situations and is not flexible in a noisy environment.

The second type of system has a master control unit attached to the robot, which instructs the robot how to perform a task. The master control unit normally receives such information from a human operator. For example, the human operator can hold the control unit and guide it along a certain movement path or trajectory. The control unit records this configuration and sends it to the robot later. In this way, the human expert, or the teacher, does not need to know the exact coordinate values of the desired position. Telemanipulation is one application of this kind of robot system. However it can be difficult to give the control unit instructions for graceful optimal paths since the human operator is often physically constrained by the control unit. Consequently the human operator needs to develop special skills for giving good instructions.

The third type of robot system obtains its movement path or trajectory by its own reasoning or exploration. In order to find the optimal path, the robot must understand the environment and the task. Finding this optimal path by reasoning may require a level of intelligence that is too complex to be practically feasible. Finding this path by exploration may be too time consuming, and the risk of hitting an object in the workspace during a search is high.

When we teach somebody to perform a task, we normally demonstrate that task rather than describe it in words. This is especially so for tasks involving movement. For example, when an experienced nurse teaches a trainee how to feed a disabled person, the whole feeding process is simply demonstrated. The trainee watches the demonstration and tries to replicate the whole action. While humans are not good at estimating the exact coordinate values of a position, they can manipulate their body parts to reach that position precisely. The person who watches the demonstration can replicate that action without the need to understand the environment and the task.

Research on 'teaching by showing and learning by watching' has become very active in recent years. Kuniyoshi et al. [33,35,34] developed a system that generates assembly programs by analysing human pick and place action sequences. Their action recognizer consists of visual feature detectors, an action/environment model, and an attention stack. Kang and Ikeuchi [28,27,29] described a robot system that observes and replicates grasping tasks by identifying different grasp phases. Kosuge et al. [32] proposed a method for representing human skills in assembly tasks so that they can be easily transferred to a robot.

Most of these systems emphasize the segmentation and recognition of motion action sequences, which are important in assembly tasks, while the actual motion path is not recognized and followed. However the motion path and its characteristics during one particular operation are very important for applications such as automatic arc welding, visual arts, and sports. The instructor and the robot may have different configurations and workspaces and therefore representation and extraction of motion characteristics need to be studied in order to transfer a motion path between different agents.

Stoica [55] presented a motion learning approach using fuzzy neural networks. A robot arm with the same configuration as that of the master arm successfully copied the master movement for the 2D case. However the low-level visual input, the positions of robot joints in the 2D images, were directly used to derive the instructions through the fuzzy neural network. Therefore, the viewing angle of the camera and the distance between the camera and the instructor adversely affect the

replication performance of the slave. Furthermore, if the configuration of the slave is different from that of the master, the neural network needs to be re-trained.

This thesis proposes a robot system that obtains the motion instruction through a vision system, since such a system has the advantages that the instructor needs not have any special sensor attached and does not need special training in teaching the robot. Our method of motion extraction does not make any assumptions about the scene or the motion type and is therefore quite general and reliable compared with existing approaches. Features from 2D images are reliably extracted and tracked in the captured image sequence. The 3D positions of these features are calculated from the image pair. Motion parameters, which are represented by the explicit high-level 3D parameters, i.e., the rotation matrix and translation vector, are obtained by a reliable least-squares method. Our system therefore does not suffer from the problems of Stoica's approach above. The trajectory of the moving object from these motion parameters is represented by a parametric cubic B-spline curve. Due to the property that the control points of the B-spline curve totally determine the characteristics of the curve [2], the motion characteristics of the trajectory are inherent in these control points. As the shape of a parametric cubic B-spline curve is invariant under translation, scale, and rotation of its control points, trajectories for robots with different configurations can be easily generated by manipulating the control points, while retaining the motion characteristics of the instructor.

1.2 Overview of Our System

Figure 1 shows the block diagram of different modules in our system. Two CCD cameras are used to capture images containing scenes of the instructor's movement. Since corner points are stable in images under perspective projection and during 3D motion, they are chosen as our features. Corner detection is applied to the first frame of each sequence. From the second frame onward, corners are tracked in each sequence. The tracked corner points are then matched in two stereo images and their 3D coordinates are calculated from the matched image coordinates. The motion parameters are obtained from these 3D positions using the algorithm described in

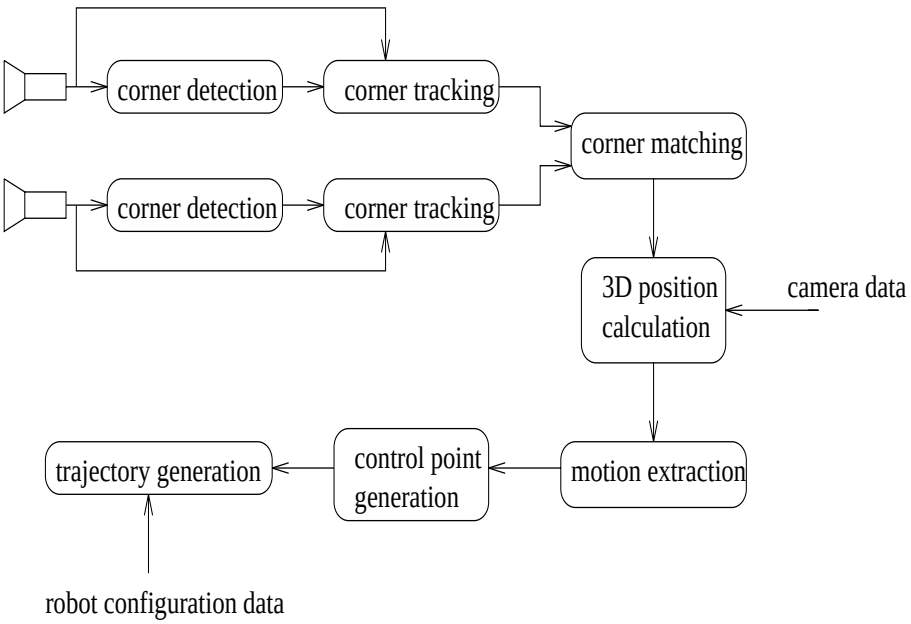


Figure 1: Block diagram of modules in our system.

Chapter 4. Following the motion extraction, a parametric cubic B-spline curve is then fitted to the obtained motion trajectory data and its control points are calculated by the least-squares method. Depending on the robot's configuration, a corresponding trajectory can be generated for the robot to replicate the extracted motion with similar shape and smoothness characteristics.

1.3 Organization of the Thesis

This thesis is organized mainly according to the order of information processing.

Given a sequence of images, it is difficult to get any useful information without any processing. Chapter 2 describes feature detection and tracking, and in particular corner detection and tracking. The current existing corner detection algorithms are reviewed first and the corner detector proposed by Harris and Stephens [20] is then implemented for our application. Since the subsequent motion extraction algorithm requires the feature positions in each image of the sequence, a new fast corner tracking algorithm based on smooth motion constraints is developed and its experimental results are compared with those of a corner tracker based on the Kalman filter technique.

Camera calibration is necessary for calculating the depth from the positions of

any 2D features. Chapter 3 discusses camera calibration and depth calculation from two stereo images. Problems with the existing camera calibration algorithms are stated and solutions to overcome them are suggested. An existing algorithm [25] to estimate the depth value from feature positions in a pair of stereo images is presented for our special case.

Chapter 4 starts with a review on algorithms of extracting motion from a pair of images taken at two different time instances. Most of these algorithms have made some assumptions about the scene and the motion type and therefore only work well in some special cases. A more robust and general algorithm modified from an existing model fitting method [39] is proposed and implemented. The results show that this algorithm is robust against noise and errors in feature locations.

Chapter 5 is devoted to motion representation and curve fitting. Motion representation is critical in transferring a trajectory from one agent to another. By representing the motion trajectory as a parametric cubic B-spline curve we can easily generate a trajectory with the same characteristics for a different agent by manipulating the control points. A least-squares algorithm for finding the control points from data points is implemented.

Chapter 6 describes an experiment to verify our 'teaching by showing and learning by watching' approach. All modules are integrated to form a whole system and experimental results show that such a system successfully replicates the motion demonstrated by a human instructor with similar shape characteristics.

Chapter 7 concludes the thesis with a summary of the contributions this research has made and some suggestions for future research work.

Chapter 2

Feature Extraction and Tracking

Images acquired by most vision sensors are represented as two-dimensional arrays of numbers. An image of dimensions 256×256 with 256 grey levels corresponds to 65536 bytes of information. It is difficult to recover any useful information from such a large amount of low level data without some form of data compression. For example, an edge detection process can reduce the amount of data in an image by more than 50 percent for most natural scenes. For this reason, significant research in machine vision is devoted to feature detection.

The difference between static vision and dynamic vision is that the former tries to get as much information as possible from a single image, while the latter is more interested in features relevant to the dynamic tasks in a sequence of images. Dynamic vision methods are more successful in most applications [12]. For this project, corners are chosen as the main features and the methodology of dynamic vision is applied when tracking features.

The contents of this chapter are organized as follows: Section 2.1 reviews some existing corner detection techniques and describes the implementation of one corner detector used in our project. Corner tracking for dynamic vision is described in Section 2.2. Some experimental results obtained by the corner detection and tracking process on real images are then shown in Section 2.3. Section 2.4 concludes this chapter with a summary.

2.1 Corner Detection

There are many different types of features in natural images, such as edges, corners and regions, and there are many techniques available for their detection. The selection of feature types depends on the particular applications. For applications dealing with moving scenes, corners are more useful and more favorable than step edges because they are very stable under a small change in lighting conditions and viewing directions. They also don't suffer from the so-called aperture problem [42] in computing optical flow and stereo matching. Therefore corner detection is usually the first step to take in applications such as stereo vision, target motion detection, and target tracking.

2.1.1 Review of Corner Detection

A corner is a point where the grey-level intensity changes in at least two directions. In relatively simple scenes it can be modelled as the intersection point or the junction point of two or more straight line edges. A corner can also be defined as a point in an image where the edge contour curvature is high. Figure 2 shows a test image of different corner types including L-junctions, T-junctions, line end points and other 2D features.

Corner detection algorithms can be broadly categorized as either using the information contained in the local grey-level values, or involving the analysis of boundary features of the object. The schemes belonging to the second category are not considered here because object boundary detection is itself a very broad research area. Within the first category, several corner detectors have been developed during the last twenty years.

The earliest ones are Beaudet's rotationally invariant operator [3] and Moravec's interest operator [41]. Beaudet's operator, called DET, is derived from the Taylor expansion of the intensity function $I(x, y)$:

$$DET = I_{xx} I_{yy} - I_{xy}^2, \tag{1}$$

where I_{xx}, I_{yy} and I_{xy} are the second order partial derivatives of $I(x, y)$. The DET measure is related to the measure of the Gaussian curvature. In practice, this corner

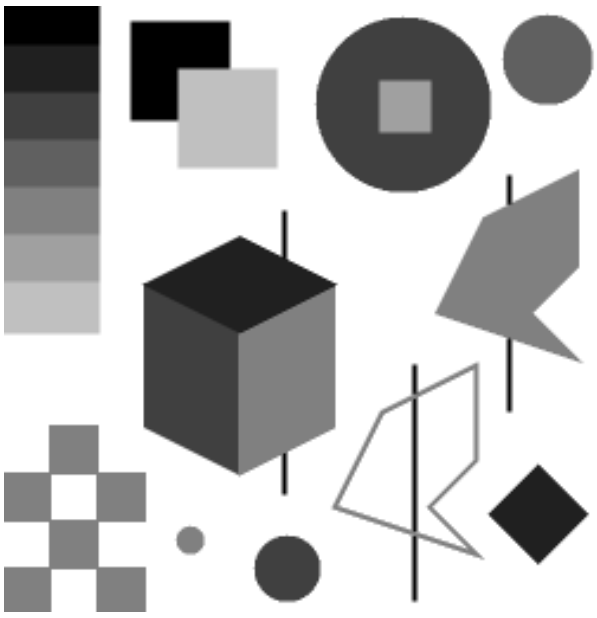


Figure 2: A test image showing different types of corner points including conventional L-junctions, T-junctions, line end points etc.

detector is reported to perform poorly at sharp corners and the second order partial derivatives are sensitive to noise [10]. Moravec's interest operator looks for large intensity variations in the horizontal, vertical and two diagonal directions in a small rectangular window. The operator suffers from a number of problems because of its discrete set of shifts in orientation at every 45 degrees and its rectangular window with only the convolution mask coefficients of 0 and 1.

Dreschler and Nagel [13] proposed a corner detector that first computes the Gaussian curvature for each pixel in the image and then finds the extrema of the Gaussian curvature several years later. Pixels lying between these extrema are declared as corner candidates. Kitchen and Rosenfeld's corner detector [30] measures "cornerness" by the product of the gradient of intensity and the gradient of gradient direction. Zuniga and Haralick [64] proposed a method for computing cornerness based on the facet model for edge detection. As reported in Nagel [42] and Mehrotra et al. [40], these three corner detectors are very similar to each other.

Harris and Stephens [20] modified Moravec's interest operator and calculated cornerness using only the first derivatives of the intensity. A circular Gaussian smoothing window is used to overcome the problem caused by the rectangular binary window. This corner detector is proven to be quite robust against noise.

Mehrotra and Nichani [40] proposed a different corner detector based on half-edge detection algorithms. Half-edge detection algorithms perform better than most edge detection algorithms at corner points because assumptions incorporated into edge detectors (for example, the infinite extent of an edge and the image intensity being an analytic function) are mostly violated at corner points. In addition, this corner detector provides not only the corner strength, but also some corner attributes, such as corner orientation and corner angles. However it assumes that a corner is formed at 90° of two edges, and is thus biased more towards some corner angles than others.

A new corner detector, which was originally designed for detecting road junctions for an autonomous vehicle [16], has been developed, based on detecting edge termination points. This corner detector has similar advantages to those of Mehrotra and Nichani's corner detector, but unlike the half-edge detection based schemes, this corner detector is not biased to corners with particular angles. This is because it only detects edge termination points and these termination points are also corner points, in most cases. It can also provide corner attributes and has been used in real-time applications, such as navigating a mobile robot in laboratory corridors.

More recently, new corner detectors have emerged, including Noble's differential geometry approach [44], Rosenthaler et al.'s keypoint detector [51], Deriche and Giraudon's scale-space based approach [11], Robbins and Owens' 2D local energy model [49], and Smith and Brady's SUSAN corner detector [54]. Most of these recent corner detectors try to improve the corner localization accuracy, to lower the false detection rate, and to classify corners according to their attributes.

2.1.2 Corner Detection and Localization

Mehrotra and Nichani [40] characterise a good corner detector by precise detection, insensitivity to noise, good localization, a low false response rate, and the ability to give corner attributes, such as corner angles and the number of edges intersected. From the above review, it can be seen that most corner detectors assume that the intensity is an analytic function, which is normally not the case in many applications. Some of these corner detectors obtain the strength of a corner by calculating

the second derivatives of the intensity values in a neighborhood of the corner point, and are thus very sensitive to noise. The corner detector developed by Harris and Stephens [20] is used in this project because it meets most of the criteria for a good corner detector, can detect most 2D features, and performs reliably in our application.

A corner is defined by Harris and Stephens as the point where the eigenvalues α and β of a given matrix G are both large. The matrix G is given by the following equation:

$$G = \begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix}, \tag{2}$$

where I_x and I_y are the first derivatives of the intensity function in the x and y directions respectively, and $<>$ is the Gaussian smoothing operator. The spread σ of the Gaussian smoothing function and therefore the size of this operator have some effects on the localization of the detected corner points and noise reduction. A wide smoothing function can reduce noise effectively, but the location of the detected corner using such a function may be offset from the true location. A trade-off has therefore to be found between noise reduction and good localization.

The matrix G describes the shape of the local autocorrelation function at the origin, and thus the values of α and β are proportional to the principal curvatures of the local autocorrelation function. Therefore a corner corresponds to the point where both curvatures of the local autocorrelation function are high. The measure of corner response C_S can be given by using the trace $Tr(G)$ and determinant $Det(G)$ of the matrix G as in the following equation, since this avoids the explicit eigenvalue decomposition of G:

$$C_S = Det - kTr^2, \tag{3}$$

where $Det = \alpha\beta =< I_x^2 >< I_y^2 > - < I_x I_y >^2$, $Tr = \alpha + \beta =< I_x^2 > + < I_y^2 >$, and k is a coefficient. The corner response C_S is positive in the corner point regions, negative in the edge regions, and small in non-feature areas. Once the corner response values are available, a non-maximum suppression process is applied to remove multiple candidates of a corner point. The 8-way local maximum is finally chosen as the detected corner point.

In summary, the corner detection process involves the following steps:

- Calculation of the gradients I_x and I_y in the x and y directions.
- Convolution of the gradients and their product with a smooth circular Gaussian window function.
- Calculation of the corner responses from the smoothed gradients.
- Thresholding the corner responses and suppressing non-maxima to eliminate multiple responses.

The corner points detected by the above method have a location resolution of one pixel. If high precision of corner localization is required, a local quadratic function can be fitted to the response values and the corner points can be located to sub-pixel accuracy. Since the repeatability of corner points is most important for our subsequent corner tracking process, sub-pixel localization is not implemented in this project.

2.2 Corner Tracking

2.2.1 Introduction

In order to estimate structure and 3D object motion from a sequence of time-varying images, it is necessary to find the correspondences of feature points (e.g., corners) on the object between successive frames once feature points in each image have been detected. Conventional matching methods find feature correspondences in two images by the relaxation technique [1,47]. This technique first estimates an initial probability of each possible matching pair. Then this probability is updated iteratively based on some criteria, such as the consistency property used by Barnard and Thompson [4], or the minimum distance. Lee and Deng [37] proposed a slightly different relaxation process by introducing a velocity consistency property. The computational cost of such a matching process is roughly proportional to the fourth power of the number of feature points present. To avoid this time-consuming matching process, some spatial and temporal constraints must be exploited [17].

It is well-known that for stereo images, the search for a correspondence in the second image can be limited to within a certain region because of the geometric constraint imposed by the relative positions of the two cameras, the so-called epipolar line. For images containing an object that undergoes an arbitrary motion, such a geometric constraint is generally not available. However, if we assume that the object motion is smooth, features in the current frame will not move far from the locations predicted from the information of the previous frames because of the temporal continuity of the motion. Such a temporal constraint plays a similar role to that of the epipolar constraint in stereo matching and therefore limits the search for a correspondence to a small area around the predicted location. By using some feature attributes introduced in the next section, spatial constraints can be added to help resolve multiple correspondences within the search area. The tracking process can be applied to each corner independently, and multiple corner tracking can be implemented in parallel if hardware is available.

2.2.2 Corner Attributes

In order to resolve multiple matching of a corner point, some attributes should be introduced for each corner point. A good attribute of a corner point must characterize it and remain stable in the image during motion. A very direct attribute is the intensity value of a point. Under certain conditions, e.g., for matte surfaces illuminated by extended light sources, the intensity value of a scene is quite stable under motion. Therefore, only points with similar intensity values should be matched, and this is called the *intensity similarity criterion* here.

However, there may be many corners with the same or similar intensity in an image or region and ambiguous matching still cannot be resolved with this criterion alone. Certain relationships among intensity values of nearby points may be relatively stable under motion, even when the actual intensity varies. These relationships provide local structural information of a point. A candidate for such an attribute is the pair of local gradients (I_x, I_y) of the intensity value in the x and y directions. The gradient is less sensitive to motion since a sharp intensity transition is likely to remain a sharp transition after a limited motion, even though

the intensity values have changed. Therefore, a correspondence can be established between points with similar gradients, and we refer to this as the *gradient similarity criterion.*

More high-level or global feature attributes, such as the number of intersected edges or region shapes computed from larger neighborhoods, are in general more stable under motion. However, their use requires time-consuming processing and effects such as perspective projection and occlusion must be taken into account. Therefore only local attributes are used here, although additional attributes can be easily included into our algorithm.

Since real images are inevitably corrupted by noise, smoothed values of attributes, i.e., the smoothed intensity value and smoothed gradient calculated using a Gaussian window, are used here. As different attributes may have different ranges, they are all normalized so that they contribute equally to the overall match value.

2.2.3 Corner Tracking Algorithm

As mentioned in section 2.2.1, we can avoid the time-consuming matching process by exploiting the temporal continuity of object motion between frames. Tomasi and Kanade [57] use the constraint of small inter-frame motion and present a point feature tracker using the correlation method. Reid and Murray [48] relax the small inter-frame motion constraint to one of the constant image-velocity between frames, and implement a corner tracker using the Kalman filter. Both trackers resolve matching ambiguities by selecting the winning candidate on the basis of the best correlation above a threshold.

Considering our application of extracting 3D smooth motion, we also assume that the inter-frame image motion is smooth. By smooth we mean that the velocity vector of the motion, its magnitude and direction, changes slowly in successive frames. Therefore the possible location of a corner in the following frame can be predicted from its velocity in the current frame, and the search for a possible correspondence can be limited to a small area around that predicted location. Suppose that a corner is located at (x_{k-1}, y_{k-1}) in frame $k-1$ and at (x_k, y_k) in frame k and the frame interval Δt is uniform. The image velocity of this corner in frame k can

be approximated by $\dot{x}_k = (x_k - x_{k-1})/\Delta t$ and $\dot{y}_k = (y_k - y_{k-1})/\Delta t$. Therefore its estimated location (x_{k+1}, y_{k+1}) in frame $k+1$ will be

$$\hat{x}_{k+1} = x_k + (x_k - x_{k-1}) \quad (4)$$

$$\hat{y}_{k+1} = y_k + (y_k - y_{k-1}). \quad (5)$$

To account for any deviation from the smooth motion assumption and measurement errors, the search for a correct match in frame $k+1$ is carried out in a window centered at the predicted location. The size of this search window depends on the largest amount of the allowed assumption and measurement errors. Corner detection in frame $k+1$ is carried out within this window and for each detected corner point, the mismatch values between these points and the corners in frame k can be calculated from their attribute values in the following way.

If there are N attributes for a corner point, an attribute vector $\mathbf{a}$ can be formed by using them as the vector elements, i.e., $\mathbf{a} = [a_1, a_2, \cdots, a_N]^T$. Suppose the attribute vector of the corner in frame k is $\mathbf{a_k}$ and that of a corner point in frame $k+1$ is $\mathbf{a_{k+1}}$. One way to compute the mismatch value between these two points is

$$m_{k,k+1} = \frac{\|\mathbf{a_k} - \mathbf{a_{k+1}}\|}{\sqrt{\|\mathbf{a_k}\| \cdot \|\mathbf{a_{k+1}}\|}}, \quad (6)$$

where $\|\cdot\|$ is the norm of a vector and $\|\mathbf{a_k}\| \cdot \|\mathbf{a_{k+1}}\|$ is the product of two vector norms. The norm (length) of the attribute vector here is always greater than zero because, according to the corner definition, a corner point will not have an attribute vector with all its components equal to zero. A perfectly matched pair will have a zero mismatch value and therefore the lower bound of $m_{k,k+1}$ is 0. Of course, the effect of some attributes can be emphasised by being given a large weight. For a successful match, the mismatch value must be lower than a preset threshold. In our work this threshold value is set to 0.2.

Once corner points within the search window, whose mismatch values are lower than the threshold value, become available, selection of the best match corner point in frame $k+1$ for the tracked corner in frame k is made according to the mismatch values. The one with the lowest mismatch value is chosen as the winning candidate. The location of this candidate is taken as the best estimate of the corner location in frame $k+1$.

2.2.4 Comparison with a Tracker Based on Kalman Filter

The Kalman filter is good at predicting some variables given a set of measurements and updating them after more measurements become available. To compare our prediction and update method with the Kalman filter, we repeat here the Kalman filter equations used by Reid and Murray [48]. The state variable is a 4-vector:

$$\mathbf{x}(k) = \; [x(k), \;\; y(k), \;\; \dot{x}(k), \;\; \dot{y}(k)]^T \, . \tag{7}$$

The state transition equation is:

$$\mathbf{x}(k+1) = F\mathbf{x}(k) + \mathbf{v}(k) \, , \tag{8}$$

where $\mathbf{v}(k)$ is zero-mean and temporally uncorrelated Gaussian noise with covariance Q, and

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{9}$$

The measurement equation, which relates the measurement variables $\mathbf{z}(k)$ with the state variables $\mathbf{x}(k)$, is given by

$$\mathbf{z}(k) = H\mathbf{x}(k) + \mathbf{w}(k) \, , \tag{10}$$

where $\mathbf{w}(k)$ is zero-mean and temporally uncorrelated Gaussian noise with covariance R, and

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} . \tag{11}$$

The purpose of using the Kalman filter here is to predict the state variable $\hat{\mathbf{x}}(k+1/k)$ and thus also the measurement vector $\hat{\mathbf{z}}(k+1/k)$ given the estimated state variable $\hat{\mathbf{x}}(k/k)$, and to update $\hat{\mathbf{x}}(k+1/k+1)$ when the new measurement $\mathbf{z}(k+1)$ becomes available. Here the terms $\mathbf{x}$ and $\mathbf{z}$ are the actual unknown/measured values, and $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ are their estimated values. The terms $\hat{\mathbf{x}}(k+1/k)$ and $\hat{\mathbf{z}}(k+1/k)$ represent the predicted values of the state variable vector and the measurement vector at time

$k+1$ given their values at time k. The Kalman filter update equation, which is not given by Reid and Murray, has the following form:

$$\hat{\mathbf{x}}(k+1/k+1) = \hat{\mathbf{x}}(k+1/k) + G(k+1)(\mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1/k)), \qquad (12)$$

where $G(k+1)$ is the Kalman filter gain matrix. It can be computed by the following equation:

$$G(k+1) = P(k+1/k+1)H^T S(k+1)^{-1} \qquad (13)$$

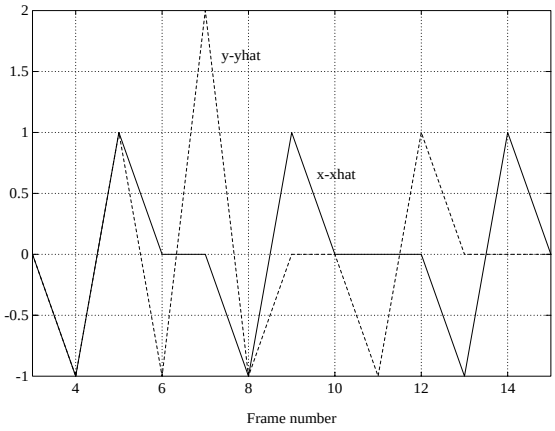
where P and S are the prediction error covariances of the state variables and measurement variables respectively.

The performance of the Kalman filter depends largely on its gain matrix. The larger the weight, the more effect on the estimate the measurement values have and the less effect on the estimate the prediction values have. When the gain matrix is the identity matrix, the estimated variables are totally dependent on the measurement variables. If the state variable estimate is initialized to the following values [48],

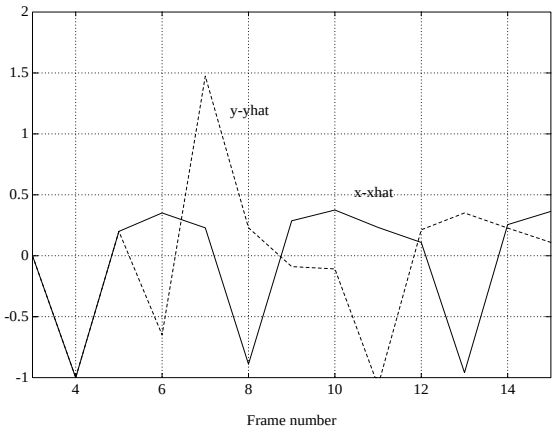
$$\hat{\mathbf{x}}(1/1) = \quad [x_1, \quad y_1, \quad x_1 - x_0, \quad y_1 - y_0]^T, \qquad (14)$$

our prediction equations are equivalent to the Kalman filter with an identity gain matrix.

Figure 3 compares the prediction errors of the measurement variables (x,y), the location of a corner point in an image in our case, of our method with those of the Kalman filter with the noise covariances $\sigma_Q = 1$ and $\sigma_R = 1$ for an image sequence containing corner points moving about five pixels in the x direction per frame. From the figure, it can be seen that the prediction error of the Kalman filter is smoother and smaller than that of our method since the corner movement matches the assumed model, and with enough frames, it will eventually converge to a value within the noise limit. However, since the resolution of our corner detection is currently ± 1 pixel, the extra accuracy gained by the Kalman filter cannot be fully utilized. In addition, if the real behavior of the state vector is not consistent with the assumed model used in the Kalman filter, the prediction error of the Kalman filter will accumulate, while our method can update the prediction values accordingly. For example, if the model assumes a constant image velocity, i.e., straight line trajectory,



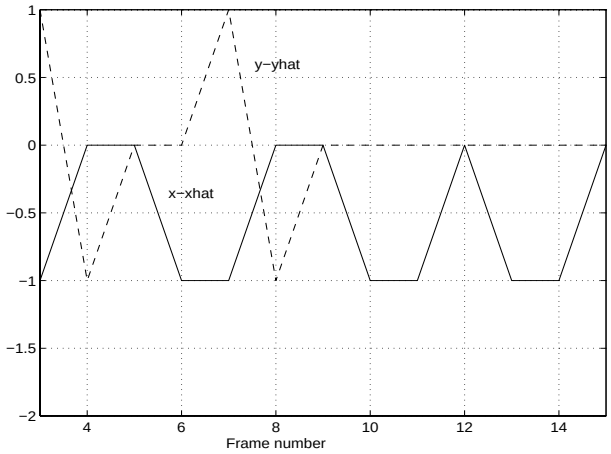
(a)



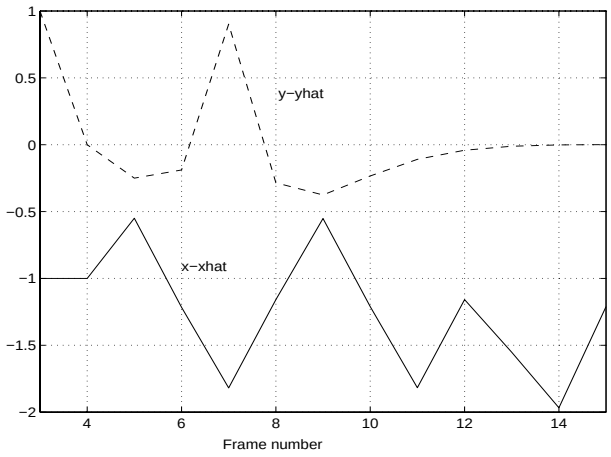
(b)

Figure 3: The prediction errors of the corner positions. (a) Errors (in pixels) of our method. (b) Errors of the Kalman filter.

the Kalman filter will predict the state/measurement variables according to this model. Quite large prediction errors will arise if the actual image velocity changes during the whole image sequence, e.g., a curved trajectory. Our method predicts state/measurement variables according to the current measurement values and the velocity changes are quickly incorporated into the updating equation. Figure 4 shows the prediction errors for another image sequence containing corner points moving along a curve. The prediction error of our method in this case is smaller



(a)



(b)

Figure 4: The prediction errors of the corner positions along a curved trajectory. (a) Errors of our method. (b) Errors of the Kalman filter.

than that of the Kalman filter method. Another advantage of our method is its low computation cost, because it involves only the corner locations in three frames while the Kalman filter uses all frames to optimize its estimates.

2.3 Experimental Results

Both the corner detection and tracking algorithms described above have been implemented on a Sun SPARC machine. For corner detection, the first derivatives of the intensity function I_x and I_y in the x and y directions are calculated by convolving the intensity values with the following two 5×5 masks:

$$M_x = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix} \tag{15}$$

$$M_y = \begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{16}$$

The Gaussian window used in smoothing the first derivatives of the intensity function has a size of 9×9, which corresponds to a spread of $\sigma = 2$ of the Gaussian smoothing function, as does the smoothing window for computing the corner attributes. The non-maxima suppression process is carried out within a 3×3 window., i.e., a corner point has to be an 8-way local maximum over a threshold. The selection of the corner response threshold value, which is set to 1000 in our implementation, is not very critical and it is mainly used to reduce noise. All the images used in these experiments are real and of dimensions 256×256 with 256 grey levels.

Figure 5 shows images of a cube and several other blocks with the detected corner points marked as white dots. It can be seen that various corners with different contrasts and angles have been correctly detected.

To test the performance of the corner tracking algorithm, two image sequences have been generated. The first sequence contains 20 frames of a cube moving from

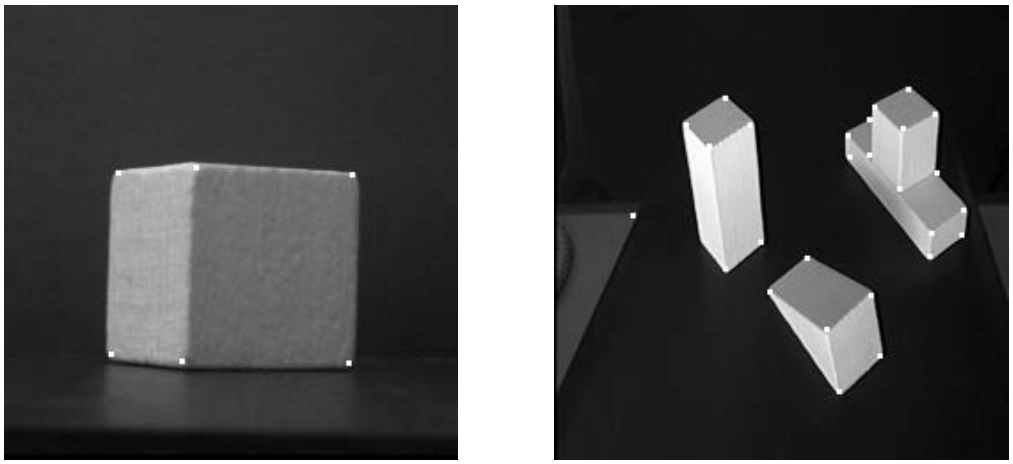
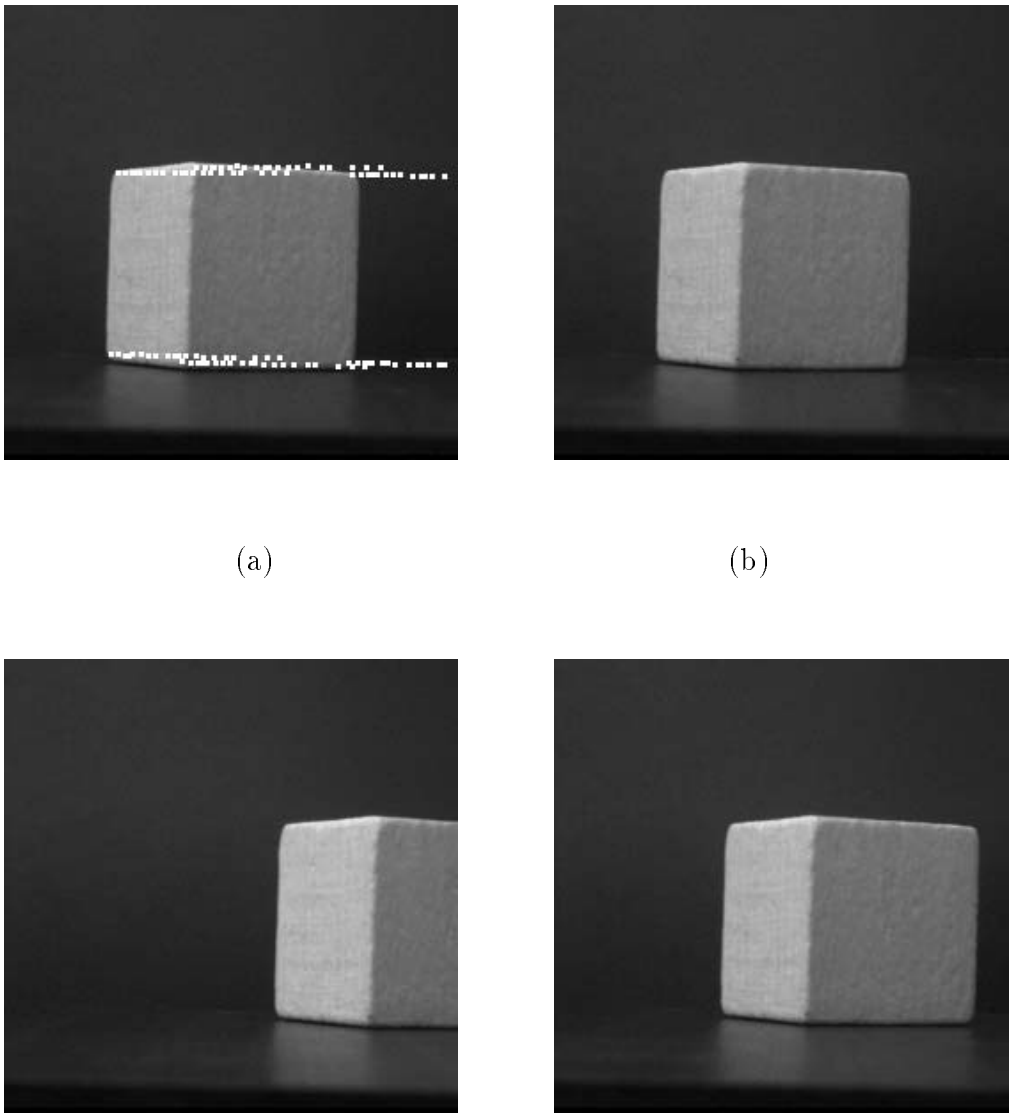


Figure 5: The detected corner points in two images.

left to right in the x direction. The movement is controlled manually and thus the trajectory is not very uniform. The original images of frame 1, 10, and 20 of this sequence are displayed in Figure 6b, 6c, and 6d.

The second sequence is a set of 15 images of a robot arm gripper grasping a planar sheet and undergoing a rotation about the vertical axis. Figure 7b, 7c, and 7d show the images of frame 1, 10, and 15.

Corner detection is carried out in the whole image for the first frame and the corner attributes are calculated for each corner point. Then a 33×33 search window, the maximal allowed displacement of a corner point between two successive frames, is defined at each corner in the second frame. The size of this search window and that of the following 11×11 tracking search window are determined by the maximal object motion amount allowed between two successive frames and the maximal prediction error of our corner tracker respectively. Within each window in the second frame, corners are searched and their attributes and mismatch values, as defined by Equation 6, are computed. If there is a candidate whose mismatch value is lower than the preset threshold 0.2, it is accepted as the correct match of that corner. In the case of multiple candidates, the one with the lowest mismatch value is selected as the winning candidate. If no corners or no matched corners are found in the search window, the corner is reported as lost and no further tracking is attempted in the following frames. For the matched corners, their locations in the following frame will be predicted according to Equations 4 and 5. As the image



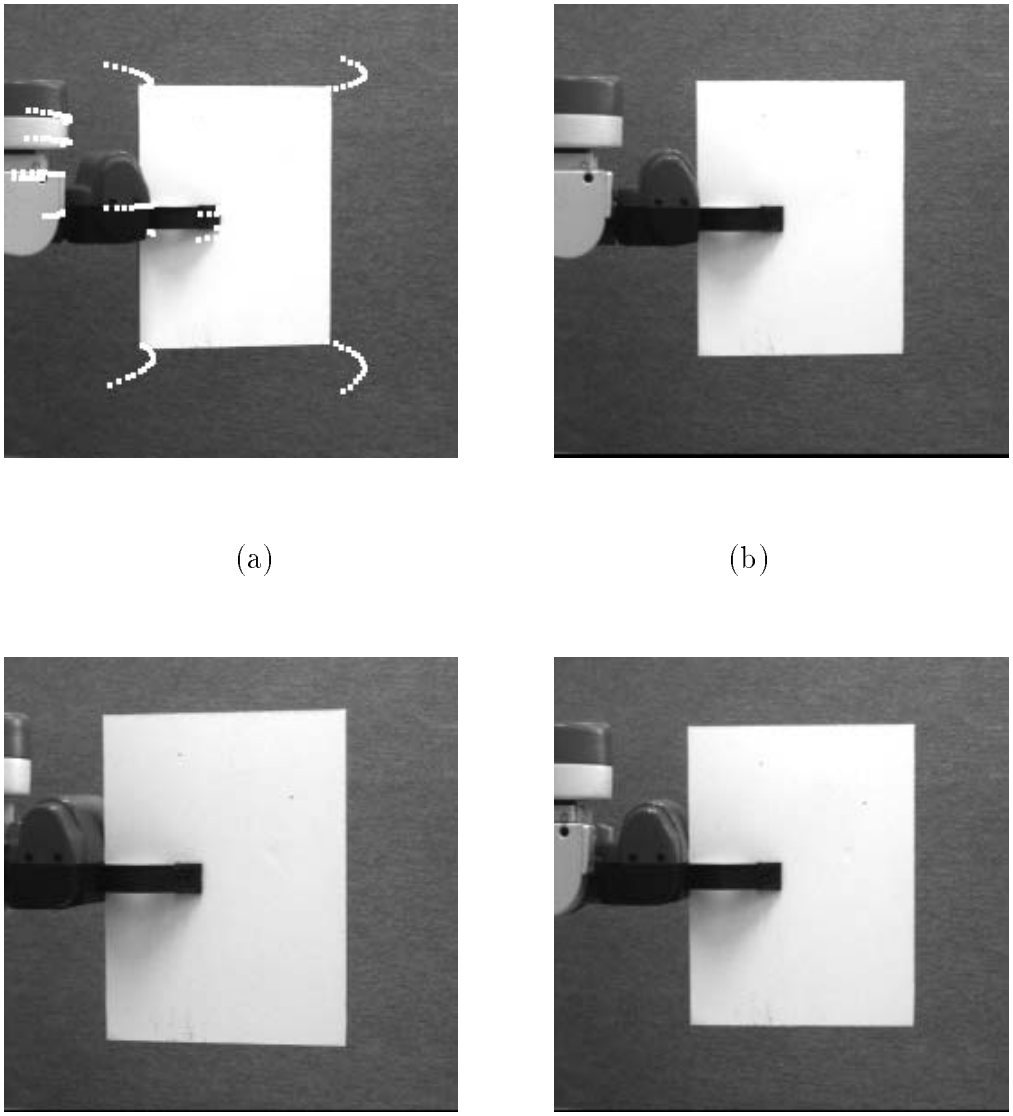
(a) (b)

(d) (c)

Figure 6: Several images of a sequence and tracked corner points. (a) The image of frame 1 with the trajectories of corner points tracked until frame 20. (b), (c), and (d) The original grey-level images of frame 1, 10, and 20.

of the following frame becomes available, an 11×11 search window, the maximal prediction error, is set around the predicted location of each corner. The search for a correspondence candidate within this window is carried out in a similar way as in frame 2. The tracking goes on until all frames are processed.

Figure 6a shows the trajectories of tracked corner points of the cube, superimposed on the image of the first frame, in a sequence of 20 frames. The image



(a) (b)

(d) (c)

Figure 7: Several images of another sequence and their tracked corner points. (a) The image of frame 1 with the trajectories of corner points tracked until frame 15. (b), (c), and (d) the images of frame 5, 10, and 15.

displacement of most corners is roughly 5 pixels in the x direction between successive frames. Two corner points are lost at frame 15 because their corresponding physical points go beyond the camera's field of view. It can be seen that all corners have been robustly tracked although the magnitude of the velocity between frame 17 and frame 18 is larger than the others.

Figure 7a shows trajectories of tracked corner points of the robot gripper, superimposed on the image of the first frame, in a sequence of 15 frames. Different corner points have different image displacements in this sequence because of the perspective projection effect. Some corner points, such as points next to the gripper, are lost during tracking due to either occlusion or some unstable factors such as shadows. These problems can be solved using more robust corner detectors and methods dealing with occlusion.

2.4 Conclusions

In this chapter some corner detection techniques have been reviewed, and a corner detector based on the first derivatives of the intensity function has been implemented. A dynamic corner tracking algorithm has been developed and implemented. Experimental results show that both the corner detector and tracker perform reliably and can be used as the first two important steps toward the ultimate goal of motion extraction.

Chapter 3

Camera Calibration and Depth Estimation

Given the position of a point in the pixel coordinates in an image plane, one must generally determine the camera characteristics and its relative geometric position and orientation with respect to a global world coordinate system in order to calculate the 3D coordinates of that point in that global world coordinate system [6,59,19]. In this chapter, we will first review some existing camera calibration techniques in Section 3.1, and then describe methods of recovering the perspective transformation matrix and the camera parameters in Section 3.2. Subsequently depth estimation from a pair of stereo images will be discussed in Section 3.3. Finally some calibration results will be presented in Section 3.4 and problems that remain will be pointed out.

3.1 Introduction

A camera is a device that acts like the eye in biological systems. It transforms the 3D coordinates of a point in some coordinate system into the pixel units on the image plane. Camera calibration is the process of finding this transformation function. This function contains the geometric and optical characteristics of the camera, the *intrinsic parameters*, and the relative 3D position and orientation of the camera coordinate system with respect to a certain world coordinate system,

the *extrinsic parameters*. For a point on the object, camera calibration provides a way of determining a ray in 3D space that the point must lie on, given its image coordinates in pixel units.

By assuming a pinhole camera model for the camera and a perfect perspective projection, the transformation from the 3D world points to the image points can be represented by a 3×4 matrix M, which implicitly contains the camera intrinsic and extrinsic parameters mentioned above. There are methods which take the various lens distortions into account and express the camera parameters explicitly. The problem of camera calibration then becomes one of solving a set of nonlinear equations and finding a solution that minimizes the sum-squared residuals of some objective function [19]. However most of these non-linear methods require a good initial guess to start the search, intensive computation, and convergence of the algorithm.

There are also techniques which simplify the nonlinear equations by using special calibration frames, or special geometric alignments of the camera and object frame, or by assuming that some of the camera parameters, e.g., the focal length and camera sensor geometric characteristics, are known [59]. They normally work under special conditions and are difficult to generalize.

The most direct method is one that combines the unknown camera parameters into a 3×4 matrix so that the equations become linear in terms of the elements of this matrix [6]. The matrix can be easily computed using a least squares method. However, the camera intrinsic and extrinsic parameters are not directly available from this matrix and nonlinear equations still need to be solved to obtain the camera parameters for applications where the world coordinate system is different from that of the calibration frame. Ganapathy [15] derived a non-iterative technique for computing camera parameters from the given perspective transformation matrix. This method is used in our project because of its simplicity, but some improvements to this algorithm will be described in the next section.

Once the camera parameters are available and the feature points in two stereo images have been detected and matched using the algorithms described in Chapter 2, 3D coordinates of these feature points can be calculated using the algorithm

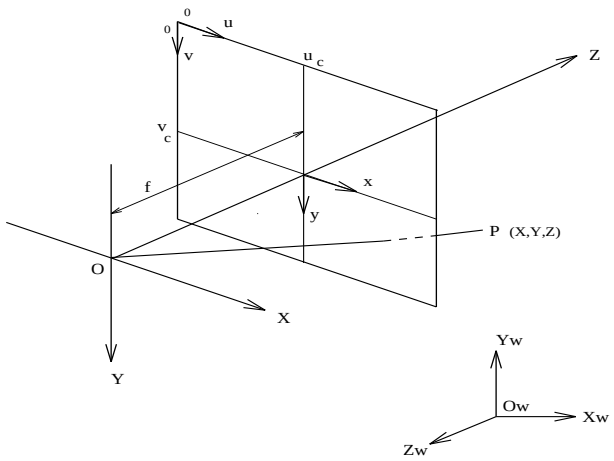


Figure 8: The camera model and geometry.

described in Section 3.3.

3.2 Camera Calibration

3.2.1 Camera Model

We assume that the camera performs a perfect perspective transformation. A coordinate system $(OXYZ)$ is attached to the camera. Its origin is located at the projection center O, the optical axis is taken as its Z axis, and its X and Y axes are chosen according to the right hand rule. The image plane is perpendicular to the Z axis and located at $Z = f$ with its x and y axes parallel to the X and Y axes.

A point P in space has the coordinate (X, Y, Z) in the camera coordinate system and is projected onto the image plane at (x, y), as shown Figure 8. By using similar triangles, we have the following relationship:

$$x = Xf/Z \tag{17}$$

$$y = Yf/Z. \tag{18}$$

Since the location (u, v) of a point in an image is expressed in pixel units and the origin of the pixel coordinate system is normally located at the top left corner of the image, the transformation from (x, y) to (u, v) takes the following form:

$$u = u_c + k_x x \tag{19}$$

$$v = v_c + k_y y \tag{20}$$

where u_c and v_c are pixel coordinates of the image plane center, and k_x and k_y are the pixel numbers within the unit length in the x and y directions. By substituting Equations 17 and 18 into Equations 19 and 20, we have

$$Zu = Zu_c + k_x f X \tag{21}$$

$$Zv = Zv_c + k_y f Y. \tag{22}$$

Letting $f_u = k_x f$ and $f_v = k_y f$, i.e., focal lengths measured in units of the pixel size in the x and y directions respectively, the transformation from 3D world coordinates to image pixel coordinates can be expressed as a set of linear equations (a 3×4 matrix) in the homogeneous coordinate system:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c & 0 \\ 0 & f_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{23}$$

where s is the scaling factor, which is equal to the value of Z. The above 3×4 matrix, written as $[INT]$ is usually referred to as the *camera intrinsic matrix* because it depends completely on the optical and geometric characteristics of the camera.

In practice, it is difficult to measure the 3D coordinates of a point relative to the camera coordinate system and often there is another global coordinate system $(O_w X_w Y_w Z_w)$ to which the object can be easily referred. The coordinates (X_w, Y_w, Z_w) of a point in this world coordinate system can be related to those in the camera coordinate system by the following matrix equation:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{24}$$

where R is the 3×3 rotation matrix relating the orientations of the two coordinate systems and $\mathbf{T}$ is the translation vector relating the locations of the two origins. The above 4×4 matrix, written as $[EXT]$, is often referred to as the *camera extrinsic*

matrix because it totally depends on the relative position and orientation of the camera with respect to the world coordinate system.

Combining Equations 23 and 24, we have the following equation:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = M_{3\times4} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad (25)$$

where $M_{3\times4} = [INT][EXT]$ and is often referred to as the *perspective transformation matrix.*

3.2.2 Computing the Perspective Transformation Matrix

Equation 25 can be easily expanded to:

$$\begin{cases} m_{11}X_w + m_{12}Y_w + m_{13}Z_w + m_{14} = su \\ m_{21}X_w + m_{22}Y_w + m_{23}Z_w + m_{24} = sv \\ m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34} = s \end{cases}$$

As s is a scaling factor, m_{34} can be set to 1. Substituting the last equation into the first two to eliminate the variable s, we have:

$$\begin{cases} m_{11}X_w + m_{12}Y_w + m_{13}Z_w + m_{14} - m_{31}X_w u - m_{32}Y_w u - m_{33}Z_w u = u \\ m_{21}X_w + m_{22}Y_w + m_{23}Z_w + m_{24} - m_{31}X_w v - m_{32}Y_w v - m_{33}Z_w v = v \end{cases}$$

Each point, with its known 3D coordinates (X_w, Y_w, Z_w) in the world coordinate system and its corresponding 2D image coordinates (u, v), gives two linear equations in 11 unknown variables m_{ij}. Given $N(\geq 6)$ calibration points on the calibration object with known coordinates (X_w, Y_w, Z_w) and their corresponding 2D image coordinates (u, v), we have $2N$ linear equations, which can be solved using a least

squares method. That is, we have

$$A \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ \vdots \\ m_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \vdots \\ u_i \\ v_i \\ \vdots \\ u_N \\ v_N \end{bmatrix}, \tag{26}$$

where the matrix A is a $2N \times 11$ matrix and is given by

$$A = \begin{bmatrix} X_{w1} & Y_{w1} & Z_{w1} & 1 & 0 & 0 & 0 & 0 & -X_{w1}u_1 & -Y_{w1}u_1 & -Z_{w1}u_1 \\ 0 & 0 & 0 & 0 & X_{w1} & Y_{w1} & Z_{w1} & 1 & -X_{w1}v_1 & -Y_{w1}v_1 & -Z_{w1}v_1 \\ \vdots & & & & & & & & & & \\ X_{wi} & Y_{wi} & Z_{wi} & 1 & 0 & 0 & 0 & 0 & -X_{wi}u_i & -Y_{wi}u_i & -Z_{wi}u_i \\ 0 & 0 & 0 & 0 & X_{wi} & Y_{wi} & Z_{wi} & 1 & -X_{wi}v_i & -Y_{wi}v_i & -Z_{wi}v_i \\ \vdots & & & & & & & & & & \\ X_{wN} & Y_{wN} & Z_{wN} & 1 & 0 & 0 & 0 & 0 & -X_{wN}u_N & -Y_{wN}u_N & -Z_{wN}u_N \\ 0 & 0 & 0 & 0 & X_{wN} & Y_{wN} & Z_{wN} & 1 & -X_{wN}v_N & -Y_{wN}v_N & -Z_{wN}v_N \end{bmatrix} \tag{27}$$

Equation 26 has the form of the classical matrix equation $AX = B$, and therefore the elements of the matrix M can be obtained using any least-squares method or by the singular value decomposition method.

3.2.3 Recovery of Camera Parameters

Since the perspective transformation matrix M is the multiplication of the intrinsic and extrinsic matrices, non-linear equations have to be solved in order to obtain the camera parameters once the matrix M is available. Ganapathy [15], Faugeras et al. [14], and Puget and Skordas [45] derived a non-iterative method to solve the non-linear equations using some properties of the rotation matrix. In fact, the

matrix M can be expanded to:

$$M = \begin{bmatrix} f_u\mathbf{r_1} + u_c\mathbf{r_3} & f_uT_x + u_cT_z \\ f_v\mathbf{r_2} + v_c\mathbf{r_3} & f_vT_y + v_cT_z \\ \mathbf{r_3} & T_z \end{bmatrix}, \tag{28}$$

where $\mathbf{r_1}$, $\mathbf{r_2}$, and $\mathbf{r_3}$ are the row vectors of the rotation matrix R, and T_x, T_y, and T_z are the components of the translation vector T in the X, Y and Z directions. If the matrix M is written in the following form,

$$M = \begin{bmatrix} \mathbf{m_1} & m_{14} \\ \mathbf{m_2} & m_{24} \\ \mathbf{m_3} & m_{34} \end{bmatrix}, \tag{29}$$

we have the following equations:

$$\begin{aligned} \mathbf{m_1} &= f_u\mathbf{r_1} + u_c\mathbf{r_3} \\ \mathbf{m_2} &= f_v\mathbf{r_2} + v_c\mathbf{r_3} \\ \mathbf{m_3} &= \mathbf{r_3} \\ m_{14} &= f_uT_x + u_cT_z \\ m_{24} &= f_vT_y + v_cT_z \\ m_{34} &= T_z \end{aligned} \tag{30}$$

Since the value of m_{34} is set to 1 when computing M, all of the variable values on the left hand side of Equation 30 are scaled by T_z, and therefore there are eleven equations with ten unknowns, six for camera extrinsic parameters and four for intrinsic parameters, as observed by Ganapathy.

Since the rotation matrix R is an orthonormal matrix, it has the following properties:

$$\begin{cases} \mathbf{r_i} \cdot \mathbf{r_i} = 1 \\ \mathbf{r_i} \cdot \mathbf{r_j} = 0 \\ \mathbf{r_1} \times \mathbf{r_2} = \mathbf{r_3} \\ \mathbf{r_2} \times \mathbf{r_3} = \mathbf{r_1} \end{cases} \tag{31}$$

where $i, j = 1, 2, 3$, $i \neq j$, and $\times$ and $\cdot$ denote the cross and dot products of two vectors respectively. Combining the above properties with Equation 30, and letting

$C^2 = \mathbf{m_3} \cdot \mathbf{m_3}$, we can calculate the camera intrinsic and extrinsic parameters by the following equations:

$$\begin{cases} u_c = \frac{\mathbf{m_1} \cdot \mathbf{m_3}}{C^2} \\ v_c = \frac{\mathbf{m_2} \cdot \mathbf{m_3}}{C^2} \\ f_u^2 = \frac{\mathbf{m_1} \cdot \mathbf{m_1}}{C^2} - u_c^2 \\ f_v^2 = \frac{\mathbf{m_2} \cdot \mathbf{m_2}}{C^2} - v_c^2 \\ \mathbf{r_1} = \frac{\mathbf{m_1} - u_c \mathbf{m_3}}{f_u C} \\ \mathbf{r_2} = \frac{\mathbf{m_2} - v_c \mathbf{m_3}}{f_v C} \\ \mathbf{r_3} = \frac{\mathbf{m_3}}{C} \\ T_x = \frac{m_{14} - u_c}{f_u C} \\ T_y = \frac{m_{24} - v_c}{f_v C} \\ T_z = \frac{1}{C} \end{cases} \tag{32}$$

As pointed out by Puget and Skordas [45], the solution obtained by the above equations is very sensitive to even a very small change in the elements of the matrix M, especially the vector $\mathbf{m_3}$. One way to improve the above algorithm is to calculate the matrix M and the intrinsic camera parameters in several calibration positions and use the mean values of these parameters as the optimal solution, since intrinsic parameters should remain constant regardless of changes in the relative position and orientation of the camera with respect to the calibration object. The extrinsic parameters can then be calculated using these optimal intrinsic parameters. Another improvement can be made to the process of calculating the matrix M. Instead of setting the element m_{34} to 1, a check on the condition number of the matrix A in Equation 27 should be made and another element of the matrix M can be set to 1 instead of the element m_{34}, if the condition number is low. This will improve the accuracy of inverting the matrix A and therefore also the accuracy of the matrix M.

3.3 Depth Estimation

To calculate the 3D positions of the matched feature points, a pair of cameras has to be used since the algorithm of structure from motion can only compute the 3D

information up to a scaling factor if only one camera is used. For a pair of cameras, the above calibration process can be applied to each of them separately. If the 3D coordinates of matched points are represented in a world coordinate system which is the same as the calibration object coordinate system, they can be calculated from the two sets of Equation 25, which become 4 independent linear equations in 3 unknowns and can be solved using a least-squares method, once the two perspective transformation matrices are available.

However the calibration object coordinate system is quite arbitrary and may not be suitable for representing objects in the scene. In this case, the 3D positions of the feature points have to be represented in a world coordinate system different from that of the calibration object, and therefore the intrinsic and extrinsic parameters of both cameras are needed to calculate the 3D information of these points. The selection of the world coordinate system depends on the individual application and the coordinate system used in this project is the same as that of the left camera when the two optical axes are parallel (see Figure 9a). Since a large overlapping field of view is favorable in most applications [25] and also makes the subsequent motion extraction algorithm more reliable, our two cameras are aligned so that their optical axes have different rotation angles around the Y_w-axis and meet at a point called the *fixation point*.

Figure 9 shows the world coordinate system and the camera alignment used in our project. When the optic axes of the two cameras are parallel, they are referred to as the vectors $\mathbf{l}$ and $\mathbf{l}'$ and are perpendicular to the vector along the baseline. When the cameras are verging, θ_1 is taken to be the offset angle of the left camera axis from the vector $\mathbf{l}$ and θ_2 is the offset angle of the right camera axis from the vector $\mathbf{l}'$. They are also the offset angles of the X-axes of the cameras with the baseline vector $\mathbf{b}$. The relative positions and orientations of the two cameras can be specified with the offset angles θ_1, θ_2 and the baseline b, which can be calculated from their extrinsic parameters using the following equations:

$$b = ||\mathbf{T}' - \mathbf{T}||, \tag{33}$$

$$\theta_1 = \cos^{-1}(\mathbf{r_1} \cdot \mathbf{b}), \tag{34}$$

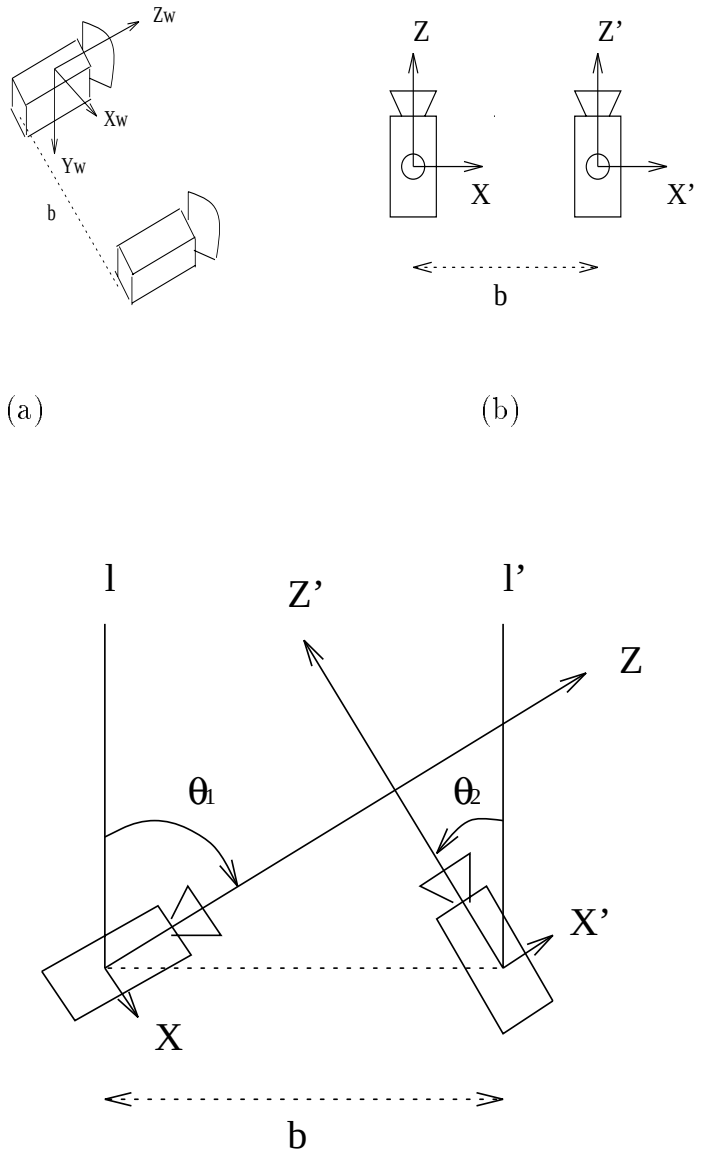


Figure 9: The coordinate system and camera alignment. (a) The perspective view with the world coordinate system attached to the left camera. (b) The top view of the camera pair when their optical axes are parallel. The Y-axes of both cameras are pointing into the page. (c) The top view of a verging camera pair with a verging angle $\theta = \theta_1 + \theta_2$ where θ_1 and θ_2 are the offset angles of the camera optical axes with respect to $\mathbf{l}$ and $\mathbf{l}'$.

$$\theta_2 = \cos^{-1}(\mathbf{r}'_1 \cdot \mathbf{b}), \tag{35}$$

where $\mathbf{b} = (\mathbf{T}' - \mathbf{T})/||\mathbf{T}' - \mathbf{T}||$ is a unit vector and $\mathbf{r_1}$ and $\mathbf{r}'_1$ are the first row vectors of the rotation matrices of the two cameras respectively.

For a 3D world point P with its image projection coordinates (u, v) and (u', v') in two images, the depth value Z_w in this world coordinate system can be calculated as follows:

$$\frac{b}{Z_w} = \tan(\theta_1 + \alpha) + \tan(\theta_2 + \beta), \tag{36}$$

where $\alpha = \tan^{-1}((u - u_c)/f_u)$ and $\beta = -\tan^{-1}((u' - u'_c)/f'_u)$. The X-value and Y-value of the point P can be calculated using the following equations:

$$Y_w = \frac{(v - v_c)Z_w \cos(\alpha)}{f_v \cos(\theta_1 + \alpha)} = \frac{(v' - v'_c)Z_w \cos(\beta)}{f'_v \cos(\theta_2 + \beta)}. \tag{37}$$

$$X_w = \begin{cases} Z_w \tan(\theta_1 + \alpha) & \text{if } \theta_1 + \alpha \text{ is not a multiple of } \pi/2 \\ b - Z_w \tan(\theta_2 + \beta) & \text{otherwise.} \end{cases} \tag{38}$$

The derivation of all the above equations are given in [25]. The equations are useful if the camera system has been pre-calibrated and its extrinsic parameters remain unchanged or are changed by some known values during the image acquisition process. In our application, the position and orientation of the camera system remain unchanged when the two cameras capture images of a scene containing moving rigid objects. The 3D positions of the feature points on the rigid object can be calculated using the above equations and motion parameters of the object can be computed using the algorithm described in Chapter 4 when two pairs of stereo images are available.

3.4 Experimental Results

A two plane calibration frame is used for calibrating the cameras in our experiment. The two planes are perpendicular to each other and contain 12 accurately marked black dots (see Figure 10), the coordinates of which are known in the calibration frame coordinate system. The coordinate resolution of these dots in this calibration frame coordinate system is 1 mm. This calibration frame is placed in front of a

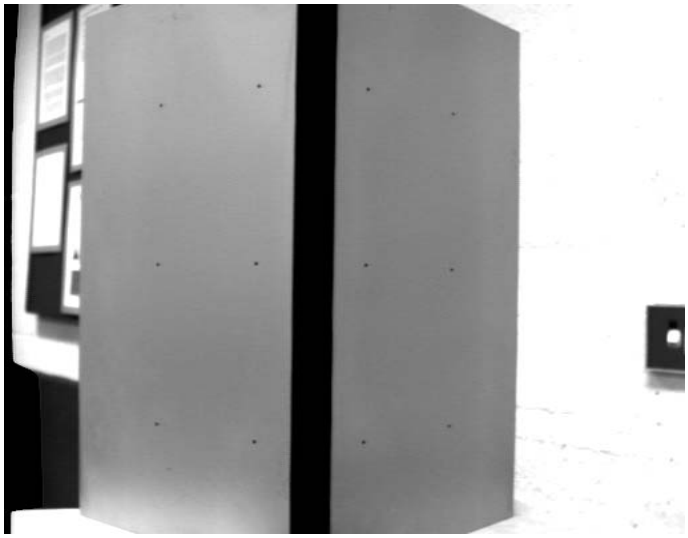


Figure 10: The frame used for camera calibration

camera so that all the dots are in the camera's visual field. In order to improve the accuracy of the intrinsic and extrinsic parameters obtained from Ganapathy's decomposition method, a sequence of 10 images is captured while the calibration frame moves to a series of slightly different locations.

The black dots are either detected by the corner detector described in Chapter 2 or picked up manually from the screen using an interactive program. The resolution of both detection methods is one pixel. For each image, the perspective transformation matrix M is first calculated and the decomposition algorithm is then used to recover the intrinsic and extrinsic parameters. One of the experimental results is given below.

The captured images have a dimension of 512×512 with 256 grey-levels. The 3D coordinates and the corresponding image coordinates of these 12 calibration points in one of the 10 images are given here:

$$
\begin{aligned}
P_1 &= (0.0, 150.145, 275.180), \quad p_1 = (81.00, 69.00) \\
P_2 &= (0.0, 150.110, 175.090), \quad p_2 = (77.00, 262.00) \\
P_3 &= (0.0, 150.105, 75.020), \quad p_3 = (77.00, 457.00) \\
P_4 &= (0.0, 50.065, 275.185), \quad p_4 = (146.00, 27.00)
\end{aligned}
$$

$$
\begin{aligned}
P_5 &= (0.0, 50.055, 174.980), & p_5 &= (143.00, 254.00) \\
P_6 &= (0.0, 50.010, 74.940), & p_6 &= (144.00, 484.00) \\
P_7 &= (49.925, 0.0, 275.350), & p_7 &= (255.00, 21.00) \\
P_8 &= (49.940, 0.0, 175.300), & p_8 &= (254.00, 252.00) \\
P_9 &= (50.010, 0.0, 75.090), & p_9 &= (254.00, 488.00) \\
P_{10} &= (150.040, 0.0, 275.310), & p_{10} &= (362.00, 58.00) \\
P_{11} &= (150.060, 0.0, 175.230), & p_{11} &= (363.00, 260.00) \\
P_{12} &= (150.095, 0.0, 75.200), & p_{12} &= (362.00, 465.00)
\end{aligned}
$$

The recovered perspective transformation matrix M is

$$
M = \begin{bmatrix}
1759.36437 & -580.94101 & 29.19952 & 185.17994 \\
495.69844 & 617.65019 & -2527.80899 & 694.62202 \\
1.58965 & 2.00829 & 0.08636 & 1.0000
\end{bmatrix}.
$$

Applying the decomposition technique described above to the matrix M, we have the extrinsic and intrinsic parameters as follows:

$$
[EXT] = \begin{bmatrix}
0.78399 & -0.62075 & 0.00444 & -0.0364 \\
-0.02255 & -0.02513 & 0.99943 & 0.1641 \\
0.62029 & 0.783647 & 0.03369 & 0.390206 \\
0.00000 & 0.000000 & 0.00000 & 1.00000
\end{bmatrix}
$$

$$
\begin{cases}
u_c = 248.58, & v_c = 275.61 \\
f_u = 678.9885, & f_v = 996.2206 \\
k = f_u / f_v = 0.6816
\end{cases}
$$

where k is the camera's aspect ratio.

Having recovered the camera intrinsic parameters for all ten images, we calculate their mean values and have the following results:

$$
\begin{cases}
\overline{u}_c = 240.12, & \sigma_{u_c} = 3.23 \\
\overline{v}_c = 276.40, & \sigma_{v_c} = 1.38 \\
\overline{f}_u = 667.67, & \sigma_{f_u} = 3.33 \\
\overline{f}_v = 978.50, & \sigma_{f_v} = 4.514 \\
\overline{k} = 0.6823, & \sigma_k = 0.00076
\end{cases}
$$

where σ is the standard deviation of each corresponding variable. It can be easily seen from the above results that the aspect ratio k is relatively constant although f_u and f_v vary marginally when the relative location of the calibration frame to the camera changes. The above standard deviations are small compared with those reported by Robert [50]. The sensitivity of these values to the position of the calibration frame is due to the nature of the linear method and the assumption of the perfect perspective transformation. If high calibration accuracy is required, more accurate and complicated models have to be used. From these mean values of the intrinsic parameters, i.e., the optimal estimation, the extrinsic parameters are calculated using Equation 32. Once the two cameras have been calibrated, their baseline and verging angles θ_1 and θ_2 are calculated using the equations given in Section 3.3.

3.5 Conclusions

The experimental results have shown that the camera calibration method described above can provide relatively reliable camera parameters. Lens distortion is not considered here mainly because only the center part of an image is used. The accuracy of these parameters is high enough for our application since the cameras will not move once they have been calibrated. The results from the algorithm for calculating the 3D coordinates of feature points from a pair of stereo images provide the input for the subsequent motion extraction process.

Chapter 4

Motion from Two Image Sequences

Given a sequence of images taken by one or two cameras at different time instances containing moving objects, how can we extract and interpret motion? The importance of motion extraction from a sequence of images is indicated in many areas, such as in characterizing and understanding human motion in dancing and athletics.

In this chapter we first investigate existing methods of calculating motion parameters from feature correspondences in image sequences and then present an algorithm to obtain the general 3D motion parameters from two pairs of images.

Section 4.1 introduces the problem of motion extraction. Most existing motion extraction algorithms make assumptions about the scene and the motion types. In Section 4.2 we establish a 3D motion model based on a perspective projection camera model, which is general enough to be suitable for most applications. Section 4.3 describes some existing algorithms of extracting motion parameters of pure translation and pure rotation from a pair of monocular images. A linear algorithm for general 3D motion is then reviewed in Section 4.4 and some of the non-linear methods to overcome the problem of the sensitivity of the solution to the error in locating feature points are also briefly mentioned. The existing algorithms for extracting general 3D motion from a pair of images suffer from the problem that the results are sensitive to the errors of feature locations or they require extensive

computation while the convergence is still not guaranteed. Section 4.5 proposes a robust linear method to extract the motion from two pairs of stereo images. This method is modified from an existing model fitting algorithm and the 3D information from the stereo vision provides better-conditioned equations. The experimental results in Section 4.6 show that this algorithm provides satisfactory results.

4.1 Introduction

The relative motion between objects in a scene and a camera gives rise to the apparent motion of objects in a sequence of images. This motion may be characterized by observing the apparent motion of a discrete set of features or brightness patterns in the images. The objective of the analysis of a sequence of images is the derivation of the motion of the objects in the scene through the analysis of the motion of features or brightness patterns associated with objects in the sequence of images.

There are two main methods developed for the computation of motion from image sequences. The first of these is based on extracting a set of relatively sparse, but highly discriminatory, two-dimensional object features in the images corresponding to three-dimensional object features in the scene, such as corners, or occluding boundaries of surfaces. Such points or lines are extracted from each image and the inter-frame correspondence is then established between these features (see Chapter 2). Constraints are formulated on assumptions such as rigid body motion, i.e., the 3-D distance between two features on a rigid body remains the same after object/camera motion. Such constraints usually result in a system of non-linear equations in terms of 3D motion parameters. The observed displacement of the 2-D image features is used to solve these equations leading ultimately to the computation of motion parameters of objects in the scene.

The other method is based on computing the optical flow or the two-dimensional field of instantaneous velocities of brightness values in the image plane [26,56,21]. In this case, a relatively dense flow field is estimated, usually at every pixel in the image. The optic flow is then used in conjunction with added constraints or information regarding the scene to compute the actual three-dimensional relative

velocities between scene objects and the camera. Computing the optic flow involves calculating the first and second derivatives of the image intensity function [22,31,53], which are normally very sensitive to noise.

In the following sections, only the feature-based approach is considered. We first present the perspective projection camera model and its geometric relation with the object considered. The problem to be solved is then presented in terms of mathematical equations. Once the model is established, algorithms for extracting certain special types of motions are described. Robust algorithms for general cases are then proposed. We mainly consider the case of extracting motion from images taken at two different time instances though multiple frames can be easily added into these algorithms using techniques such as Kalman filtering. Corners, or points, are used as the main features and it is assumed that they have been extracted from each frame and their inter-frame correspondences have already been established. Only rigid motion is considered here and no distinction is made between the situations where a) the camera is moving and the imaged scene is stationary, b) the camera is stationary while the imaged object is in motion, or c) both camera and objects are in motion. What is computed is the relative position and motion between the camera and the imaged object although most of the time it is assumed that the object is in motion.

4.2 Camera and Motion Model

A camera projects a 3D world point $\mathbf{X} = (X_w, Y_w, Z_w)^T$ onto a 2D image point $\mathbf{x} = (x, y)^T$. As described in Chapter 3, the general mapping from the point $\mathbf{X}$ to the point $\mathbf{x}$ can be written as Equation 25 in terms of a 3×4 perspective transformation matrix M. Different camera models give different properties of the matrix M and Shapiro et al. [52] described the various properties of the matrix M for the affine camera model. The affine camera model simplifies the mapping and therefore also makes the subsequent motion estimation easier. However, since objects are moving during motion estimation or extraction, the condition of the *weak perspective* for the affine camera model cannot always be met. The weak

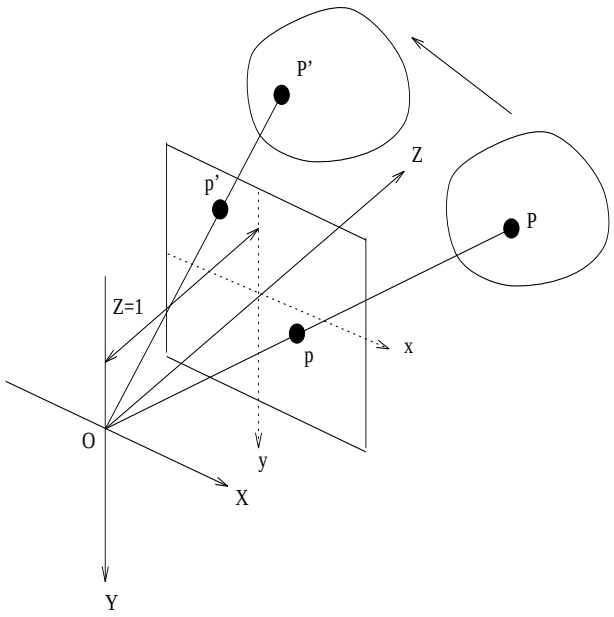


Figure 11: The camera model and a moving scene.

perspective condition requires the ratio of the depth variation of the object to the average distance of the object from the camera to be small, usually 1:10.

We use the more general perspective projection camera model here. The calibration of this type of camera has been discussed in Chapter 3. Figure 11 shows the basic geometric layout of the camera and a moving object. A camera coordinate system $(OXYZ)$ is attached to the projection center O of the camera and the camera's optic axis is taken as its Z axis. The image plane x-y is located at $Z = f$ (f is the focal length) and parallel to the X-Y plane. Using the perspective projection model, a point $P(= (X, Y, Z))$ on the surface of an object is projected to a point $p(= (x, y))$ on the image plane and we have

$$\begin{aligned} x &= Xf/Z \\ y &= Yf/Z. \end{aligned} \tag{39}$$

For simplicity, we assume that $f = 1$. Its actual value can be incorporated into the camera intrinsic parameters. The location (x, y) of the point p, can be calculated using the intrinsic parameters of the camera once its location (u, v) in the image pixel unit becomes available.

Let the two images be taken at time t and t'. The point P at time t will move to P' at time t' in the world coordinate system, and its corresponding projection

positions on the image plane move from p to p'.

It is well known from kinematics that the coordinates of a point on a rigid object at time instances t and t' are related by the following equation:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{T} \tag{40}$$

where $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and $\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$.

The matrix R here represents a rotation and the vector $\mathbf{T}$ represents the translation. Rotation can be represented in a number of other equivalent ways. For example, rotation can be around an axis passing through the origin of a coordinate system. Let $\hat{\mathbf{n}} = (n_1, n_2, n_3)^T$ be a unit vector along the axis of rotation and let θ be the angle of rotation from time t to time t'. Then the rotation matrix R can be expressed as

$$R = S + K, \tag{41}$$

where

$$\begin{aligned} S &= (\cos\theta)I + (1 - \cos\theta)\hat{\mathbf{n}}\hat{\mathbf{n}}^T \\ K &= (\sin\theta)N, \end{aligned} \tag{42}$$

where I is the 3×3 identity matrix and

$$N = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \tag{43}$$

It is obvious that $\hat{\mathbf{n}}$ and θ can be obtained easily from R. In fact, we have the following relations:

$$\begin{aligned} Tr(R) &= r_{11} + r_{22} + r_{33} = 1 + 2\cos\theta \\ \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} &= 2(\sin\theta)\hat{\mathbf{n}} \end{aligned}$$

Alternatively, R can be specified as three successive rotations around the X, Y, and Z axes, by angles α, β, and γ, respectively, and can be written as a product of these three rotations

$$R = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{44}$$

Finally, a 3D rotation can also be represented as a unit quaternion. A quaternion $\mathbf{q}$ generally consists of a component of a three-dimensional vector $\mathbf{Q} = (q_1, q_2, q_3)^T$ and a scale component q_0:

$$\mathbf{q} = \mathbf{Q} + q_0. \tag{45}$$

Some notations and definitions of quaternion operations are given in Appendix A. For the unit quaternion describing a 3D rotation, $\mathbf{q}$ has the following form:

$$\mathbf{q} = (\sin(\theta/2)\hat{\mathbf{n}}, \sin(\theta/2)), \tag{46}$$

where $\hat{\mathbf{n}}$ is the unit vector along the rotation axis and θ is the rotation angle. We will use this representation to extract motion.

The problem to be solved here can be stated generally as follows: Given the positions, $(x_i, y_i)|i = 1, .., N$ and $(x_i', y_i')|i = 1, .., N$, of a set of N feature points in images at time t and t' and their correspondences, we would like to calculate the rotation matrix R and the translation vector $\mathbf{T}$ using some reasonable constraints. Obviously the camera model and motion type will affect the resulting relationship between the 2D position of a point and motion parameters since the camera model affects the mapping from (x, y) to (X, Y, Z) and the motion type affects the motion Equation 40.

Different assumptions can be made on the camera model, usually depending on the applications. Orthographic projection cancels the unknown depth variable and thus makes the subsequent motion extraction easy [60,26,24,58,61]. However this model is only valid when small objects are viewed by a distant camera and motion parameters in certain directions, e.g., along the optical axis, cannot be determined using this camera model. Weak perspective and para-perspective projections assume an unknown average depth and thus simplify the resulting equations

for motion extraction [36,52]. This type of camera model is only valid when the field of view is small, the variation in depth of the scene along the line of sight is small compared to its average distance from the camera, and the motion is small along the line of sight.

In the following, the more general perspective projection camera model is used. The linear algorithms for the two special types of motions are first given and then an existing method of extracting the general 3D motion is described. Problems with this method are pointed out and a more reliable algorithm is proposed.

4.3 Special Cases

4.3.1 Pure Translation

If an object undergoes a motion such that the rotation component can be ignored, the motion can be approximated as a pure translation. In this case, Equation 40 becomes

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}. \tag{47}$$

Combining the above equation with Equation 39, we obtain the following basic equation for one point [23], which is linear in terms of unknown variables T_1, T_2, and T_3:

$$\Delta y T_1 - \Delta x T_2 + (y\Delta x - x\Delta y)T_3 = 0, \tag{48}$$

where $\Delta x = x' - x$ and $\Delta y = y' - y$ are the motion disparities in the x and y directions respectively. For a set of N feature points, we have the following matrix equation:

$$A \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = 0, \tag{49}$$

where A is an $N \times 3$ matrix

$$A = \begin{bmatrix} \Delta y_1 & -\Delta x_1 & (y_1 \Delta x_1 - x_1 \Delta y_1) \\ \Delta y_2 & -\Delta x_2 & (y_2 \Delta x_2 - x_2 \Delta y_2) \\ \vdots & & \\ \Delta y_i & -\Delta x_i & (y_i \Delta x_i - x_i \Delta y_i) \\ \vdots & & \\ \Delta y_N & -\Delta x_N & (y_N \Delta x_N - x_N \Delta y_N) \end{bmatrix}. \tag{50}$$

Since a single camera has the so-called scale ambiguity problem, i.e., an object closer to the camera and moving slowly, and one further away from the camera but moving fast will result in the same relative motion, a solution can only be found to within a scale factor. With two corresponding feature points the above equation can be solved. However there are generally errors in locating the feature points and therefore more than just two points are needed. An optimal solution can be found by minimizing the norm of the matrix $A\mathbf{T}$. By setting one of the elements of $\mathbf{T}$ to one, normally the most dominant one, the other two elements can be found by applying the singular value decomposition to the matrix A. The solution is also the unit eigenvector $\hat{\mathbf{T}}$ of the matrix $A^T A$, associated with its smallest eigenvalue.

4.3.2 Pure Rotation

If an object undergoes a pure rotation or the translation amount is negligible, Equation 40 becomes

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{51}$$

From Equations 39 and 51, we have

$$\begin{aligned} x' &= \frac{r_{11}x + r_{12}y + r_{13}}{r_{31}x + r_{32}y + r_{33}} \\ y' &= \frac{r_{21}x + r_{22}y + r_{23}}{r_{31}x + r_{32}y + r_{33}}. \end{aligned} \tag{52}$$

The above equations are linear in terms of the unknown variables r_{ij}. A similar linear method to the pure translation case can be used to obtain the rotation matrix

elements if more than four corresponding points are available. However the solution obtained may not form an orthonormal matrix because the rotation matrix has only three degrees of freedom, thus not all its elements are independent and their relations are not considered in the above least-squares method. If constraints on the rotation matrix need to be considered, the resulting equations become non-linear. A unit quaternion representation of rotation avoids non-linear equations and the problem can be solved using the techniques similar to the least-squares method. Since pure rotation is rare in most applications, we do not go into further detail here.

4.4 General Three-Dimensional Motion

If an object undergoes a general motion, we can write the position of a feature point on the image plane as a three-dimensional column image vector $\mathbf{x} = (x, y, 1)^T$. Equation 40 can be rewritten using the image vectors $\mathbf{x}$ and $\mathbf{x}'$ of point P at time t and t' as follows:

$$Z'\mathbf{x}' = ZR\mathbf{x} + \mathbf{T}. \tag{53}$$

Weng et al. [63] developed a linear method to extract motion from a pair of monocular images. In the following we briefly present their results and discuss several problems with this algorithm.

If the translation component is not zero, i.e., $\|\mathbf{T}\| \neq 0$, the above equation can be written as

$$\frac{Z'}{\|\mathbf{T}\|}\mathbf{x}' = R\frac{Z}{\|\mathbf{T}\|}\mathbf{x} + \hat{\mathbf{T}}, \tag{54}$$

where

$$\hat{\mathbf{T}} = \frac{\mathbf{T}}{\|\mathbf{T}\|}. \tag{55}$$

Let $\mathbf{T}_s$ be a unit vector that is aligned with the translation vector $\mathbf{T}$, i.e.,

$$\mathbf{T}_s \times \mathbf{T} = 0. \tag{56}$$

By precrossing both sides of Equation 54 by $\mathbf{T}_s$ we get

$$\frac{Z'}{\|\mathbf{T}\|}(\mathbf{T}_s \times \mathbf{x}') = \frac{Z}{\|\mathbf{T}\|}[\mathbf{T}_s]_\times R\mathbf{x}, \tag{57}$$

where $[\mathbf{T}_s]_\times$ is the mapped 3×3 skew symmetric matrix from the three-dimensional vector according to the following mapping:

$$[(t_1, t_2, t_3)^T]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{58}$$

Premultiplying both sides of Equation 57 by $\mathbf{x'}^T$, we have

$$\mathbf{x'}^T [\mathbf{T}_s]_\times R\mathbf{x} = 0 \tag{59}$$

since $\mathbf{x'}^T(\mathbf{T}_s \times \mathbf{x'}) = 0$ and $Z \neq 0$. Let matrix E be

$$E = [\mathbf{T}_s]_\times R = \begin{bmatrix} \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{bmatrix}; \tag{60}$$

E is usually referred to as the *Essential Matrix* in the research field of stereo vision. Using this definition, we can rewrite Equation 59 as

$$\mathbf{x'}^T E\mathbf{x} = 0. \tag{61}$$

This equation was originally derived by Longuet-Higgins [38] using the epipolar constraint. The objective here is to find the essential matrix E first from the image vectors $\mathbf{x}$ and $\mathbf{x'}$. Each point correspondence gives one equation which is linear and homogeneous in the elements of E and can be written as

$$\begin{pmatrix} xx' & xy' & x & yx' & yy' & y & x' & y' & 1 \end{pmatrix} \mathbf{E} = 0, \tag{62}$$

where the elements of the vector $\mathbf{E} = (e_1, e_2, \cdots, e_9)^T$ correspond to the elements of the matrix E, i.e.,

$$E = \begin{bmatrix} e_1 & e_4 & e_7 \\ e_2 & e_5 & e_8 \\ e_3 & e_6 & e_9 \end{bmatrix}. \tag{63}$$

If there are N ($N \geq 8$) pairs of point correspondences in two images, the equation we have to solve becomes $A\mathbf{E} = 0$, where A is an $N \times 9$ matrix. Taking possible errors in locating feature points into account, the matrix E can be solved using the least-squares method. Once the matrix E is available, the rotation matrix R and the unit translation vector $\hat{\mathbf{T}}$ can be easily solved using the relation between the

matrix E, R, and $\hat{\mathbf{T}}$ (Equation 60). The paper by Weng et al. [63] gives the details to calculate motion parameters from the matrix E and some error analysis.

One problem with this linear method is that the matrix E is not necessarily the essential matrix due to the errors in locating feature points, and thus the final solution (R and $\hat{\mathbf{T}}$) might be totally different from the correct one. The ideal essential matrix ($E = [\hat{\mathbf{T}}_s]_\times R$) has to meet the following two conditions:

- Two of the eigenvalues of the matrix $E^T E$ must be equal.
- The other eigenvalue of the matrix $E^T E$ must be equal to zero.

These two conditions will result in two polynomial equations of elements of the matrix E. Incorporating these two constraints directly into the matrix equation seems difficult and also results in a set of non-linear equations. Weng et al. [62] suggested a non-linear method of motion extraction by indirectly incorporating the E matrix constraints into the matrix equation, e.g. by minimizing the image vector errors. Although errors can be reduced, non-linear methods are generally computationally expensive and do not always converge. Poor initial guess values can also lead to the wrong solution.

4.5 Motion Extraction from Two Pairs of Stereo Images

As described above, most methods of motion extraction by a single camera suffer from the problems of an ambiguous scaling factor and also the sensitivity of the solution to the measurement error. There are many algorithms available to recover the 3D scene structure from a pair of stereo images. The main problem to be solved here is to establish the correspondence of features in the two images (left and right). Much research has been carried out in this field and we assume here that feature points have been detected and matched in both images and their 3D positions are calculated. Given two pairs of such stereo images and 3D positions of a set of feature points on a moving object at time t and t', the following linear method, modified

from the algorithm of model estimation [39], calculates the motion parameters of the moving object from time t to t'.

Given 3D positions of a set of points $\mathbf{P}_i(X,Y,Z)$ and $\mathbf{P}'_i(X,Y,Z)$ where $i = 1,2,\cdots,N$ on a moving object at time t and t', how can we extract the motion parameters? The relation between a point $\mathbf{P}$ of a rigid body at time t and that same point moved to $\mathbf{P}'$ at time t' is represented by Equation 40. The objective here is to find the motion parameters that minimize

$$\Sigma_i ||\mathbf{X}'_i - (R\mathbf{X}_i + \mathbf{T})||^2. \tag{64}$$

Since all points on the object are translated by the same amount, we can construct a set of N vectors in $3D$ space from N points by vector subtraction

$$\mathbf{V}_i = \mathbf{X}_i - \mathbf{X}_m, \tag{65}$$

where $i = 1,2,\cdots,N$ and $\mathbf{X}_m$ is the middle point (centroid) of these N points. Substituting the above equation into Equation 40 leads to the following equation:

$$\mathbf{V}'_i = R\mathbf{V}_i. \tag{66}$$

The minimization problem can be reformulated as one of finding the rotation that minimizes the following:

$$\Sigma_i \|\mathbf{V}'_i - R\mathbf{V}_i\|^2. \tag{67}$$

As mentioned earlier, a rotation can also be represented by a unit quaternion $\mathbf{q} = (\sin(\theta/2)\hat{\mathbf{n}}, \sin(\theta/2))$. By using the definitions and notations described in Appendix A, we have

$$R\mathbf{V} = \mathbf{q} * \mathbf{V} * \tilde{\mathbf{q}} \tag{68}$$

for any three-dimensional vector $\mathbf{V}$ [18], where $\tilde{\mathbf{q}}$ is the conjugate of $\mathbf{q}$ and $*$ denotes quaternion multiplication. The quantity to be minimized in Expression 67 can be therefore rewritten as

$$\Sigma_i \|\mathbf{V}'_i - \mathbf{q} * \mathbf{V}_i * \tilde{\mathbf{q}}\|^2. \tag{69}$$

After some matrix manipulations, Expression 69 becomes

$$\Sigma_i \|\mathbf{q}A_i\|^2 = \Sigma_i \mathbf{q}A_iA_i^T\mathbf{q}^T = \mathbf{q}B\mathbf{q}^T, \tag{70}$$

where $B = \Sigma_i A_i A_i^T$ is a 4×4 symmetric matrix. The matrix A_i can be obtained from the vectors $\mathbf{V}_i$ and $\mathbf{V}'_i$ by the following equation:

$$A_i = \begin{bmatrix} [\mathbf{V}'_i + \mathbf{V}_i]_\times & -(\mathbf{V}'_i - \mathbf{V}_i) \\ (\mathbf{V}_i - \mathbf{V}'_i)^T & 0 \end{bmatrix}, \tag{71}$$

where $[\mathbf{V}'_i + \mathbf{V}_i]_\times$ is the mapped 3×3 matrix from the vector $(\mathbf{V}'_i + \mathbf{V}_i)$ by Equation 58. Minimization of Expression 70 represents a classical problem where the solution $\mathbf{q}_{min}$ is the eigenvector corresponding to the smallest eigenvalue of the matrix B. The smallest eigenvalue also gives a measure of the error in the estimation of the rotation parameters. The rotation matrix R is easily recovered once the eigenvector $\mathbf{q}_{min} = (\mathbf{w}, s)$ of the matrix B has been found by computing

$$R = (\mathbf{I}_{3\times3} + (1 - \cos\gamma)(\mathbf{W}_\times)^2 + \sin\gamma[\mathbf{W}]_\times)^T, \tag{72}$$

where $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix, $\gamma = 2\cos^{-1} s$, and $[\mathbf{W}]_\times$ is the skew symmetric matrix associated with the vector $\mathbf{w}/\sin(\gamma/2)$.

Once the rotation matrix R is determined, the translation vector can be easily obtained by Equation 40 and the best estimation for $\mathbf{T}$ is the mean value of the following vectors:

$$\mathbf{T}_i = \mathbf{X}'_i - R\mathbf{X}_i, \tag{73}$$

where $i = 1, 2, \cdots, N$.

4.6 Experimental Results

All the images used in the experiments are real and of dimensions 256×256 with 256 grey levels unless it is indicated otherwise. Corner points are detected and tracked using the algorithms described in Chapter 2. Camera parameters are obtained using the calibration process described in Chapter 3.

4.6.1 Pure Translation

For a pure translation motion, the least-squares method described above is used to calculate the translation vector to within a scale factor. A sequence of 15 images

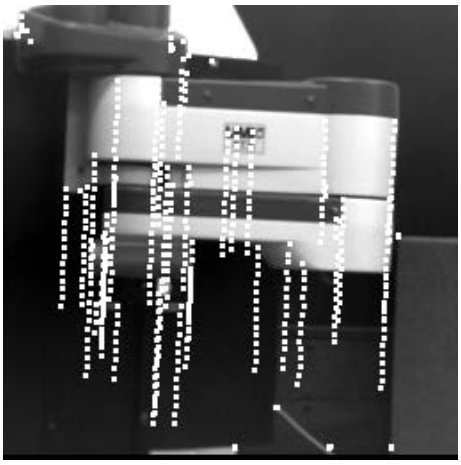


Figure 12: The image of the first frame of a robot arm with the marked trajectories of corner points in the sequence.

frame pair	$\overline{\Delta x}$, $\overline{\Delta y}$ in pixel unit	$\hat{\mathbf{T}}_s$ (estimated)	$\mathbf{T}_s$ (true)
(1,2)	(0,5)	(0.0100, 0.1577, 0.9874)	(0.0, 1.0, 0.0)
(1,3)	(0,10)	(-0.0087, 0.5595, 0.8280)	(0.0, 1.0, 0.0)
(1,4)	(0,15)	(-0.0638, 0.9978, -0.0196)	(0.0, 1.0, 0.0)

Table 1: Results of estimated motion parameters

containing a robot arm moving downwards (i.e. in the Y direction) are captured by a single camera. Figure 12 shows the original image of the first frame in the sequence with the marked trajectories of all detected corner points in the subsequent frames.

Figure 13 shows the grey-level images of frame 1 to 4. The estimated motion parameters (translation vectors) of the robot arm from frame 1 to frame 2, 3 and 4 are shown in Table 1. The average pixel displacements in the x and y directions are also given in the table. From the table, it can be clearly seen that the errors in calculating the motion parameters using the linear least-squares method reduce when the pixel displacements are over 10 pixels, while the solution is totally useless when the pixel displacements are less than 10 pixels. This indicates that this algorithm works better when the resulting interframe motion in the image plane is large. However, this in turn will make feature matching difficult.

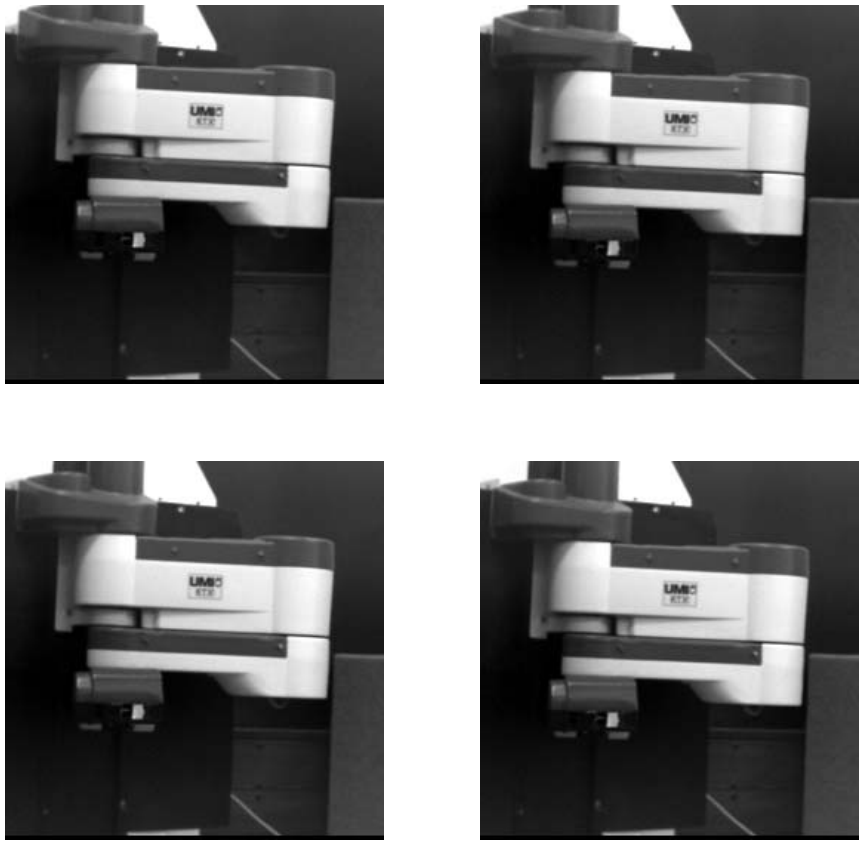


Figure 13: Grey-level images of frame 1 to 4.

4.6.2 General Motion

Figure 14 shows a pair of images used for testing the linear least-squares method. The images are captured using a single camera at two different locations, translated by a small amount in the X direction. This is equivalent to the case when the object moves in the opposite direction. This motion results in a maximal displacement of about 7 pixels in the image plane. A total of 12 points have been detected and used for calculating the matrix E. If the essential matrix E is calculated from these 12 sets of matching points without considering the epipolar constraints, the resulting motion parameters for this pair of images are as follows:

$$\hat{R} = \begin{bmatrix} 0.9999 & -0.0009 & -0.0101 \\ 0.0009 & 1.0000 & -0.0002 \\ 0.0101 & 0.0002 & 0.9999 \end{bmatrix} \tag{74}$$

and

$$\hat{\mathbf{T}}_s = [0.0004, 0.1203, 0.9927]^T. \tag{75}$$

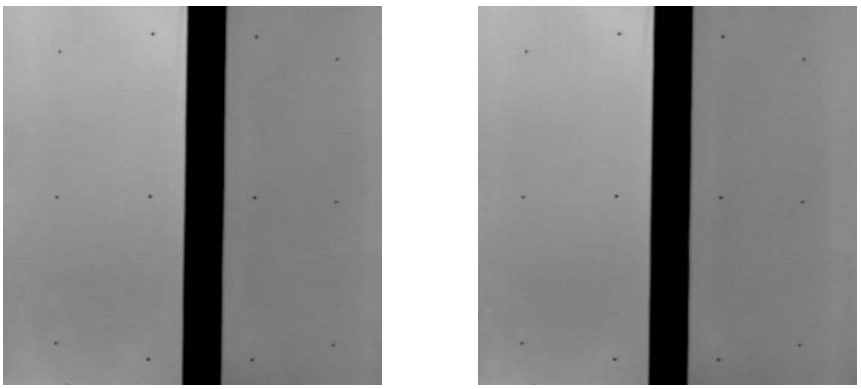


Figure 14: A pair of test images (the camera calibration frame)

This is totally different from the real motion, which has no rotation and the translation vector is $\mathbf{T}_s = [-1.000, 0.0, 0.0]^T$. The values of $\mathbf{x}'^T E \mathbf{x}$, which should theoretically be equal to zero, have been calculated and they are actually quite small. The reason behind this is that the matrix E that satisfies Equation 61 is not unique and the one obtained by the least-squares method without considering the epipolar constraints will not necessarily give the right motion parameters.

4.6.3 Motion from Two Pairs of Stereo Images

To test the algorithm of motion extraction from two pairs of stereo images as described in Section 4.5, a set of 3D simulation points are generated. These are four corner points of a cube with a size of $1000 \times 1000 \times 1000$ (see Figure 15). The cube is rotated by 30 degrees around the Y-axis and translated 100 along the X-axis. The coordinates of these four points are corrupted by a noise level of ten percent. Using the method described in Section 4.5, a set of four vectors are formed from these four points and their centroid to calculate the rotation parameters first. The translation vector $\mathbf{T}$ is then calculated using Equation 73 once the rotation parameters are available. The simulation has run 100 times with random noise, and the mean values of the calculated motion parameters are shown in Table 2 together with their true values. From the table it can be seen that the estimated motion parameters are very close to their true values although the 3D point coordinates are corrupted with noise. It can be concluded that this method is quite robust against noise.

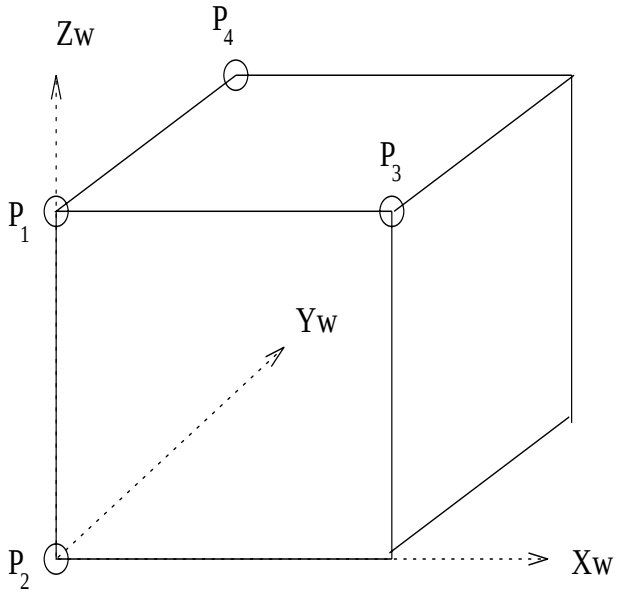


Figure 15: A set of corner test points of a cube.

	translation vector (T_x, T_y, T_z)	rotation axis (n_x, n_y, n_z)	rotation angle θ
true values	(100.0000,0.0000,0.0000)	(0.0000,1.0000,0.0000)	30.00
estimated values	(99.5754,0.0595,0.3989)	(0.0105, 0.9947,0.0015)	30.01
errors	(0.4246,-0.0595,-0.3989)	(-0.0105,0.0053,-0.0015)	-0.01

Table 2: Results of estimated motion parameters from stereo images

4.7 Conclusions

Several algorithms for motion extraction have been described in this chapter. For pure translation and pure rotation, linear methods are still favourable and can give reasonable results. For general cases, the availability of two pairs of stereo images leads to a linear method which can give much more reliable results and which does not suffer from the ambiguous scale problem existing for any method using a pair of monocular images.

Chapter 5

Motion Representation and Curve Fitting

This chapter presents a novel motion representation scheme using parametric cubic B-spline curves. In this representation, motion characteristics are completely determined by the control points of a trajectory curve. Therefore, a similar motion trajectory for another agent (a robot) in its own workspace can be easily generated by manipulating the control points obtained from the trajectory of the primary agent (an instructor). Together with the vision system described in the previous chapters, this representation scheme gives a robot the ability to conduct a similar motion to the one it perceives.

Section 5.1 explains the importance of a robot's imitation ability in performing a given task and introduces the concept of motion characteristics, as well as features of parametric curve representation. Section 5.2 and 5.3 briefly state some basic concepts of cubic B-splines and describe parametric curve representation using them as basis functions. Section 5.4 describes a method for finding control points from a set of given measurement data. Section 5.5 discusses some useful properties of the cubic B-spline curve representation for transferring a motion trajectory between different workspaces. Section 5.6 covers curve fitting to the orientation data. Section 5.7 presents some experimental results of the curve fitting algorithm and discusses the effect of the number of control points on the fitting accuracy. Section 5.8 concludes the chapter with some remarks on the potential applications of this representation

scheme.

5.1 Introduction

As mentioned in Chapter 1, most current robot systems have a master control unit, which tells the robot how to move to get to its goal. While the master control unit may get visual instructions from a human operator, it is often difficult to give instructions for a graceful movement path. The human operator also needs special training to be a good instructor.

Our vision system can obtain the motion parameters of any moving object, in particular a human instructor in this special case. The motion trajectories of the instructor, the sequential origin positions and orientations of its coordinate system, are therefore available. As we know, the easiest way to control the robot to follow a trajectory is to connect the successive points on the trajectory by straight lines and to get a piecewise linear curve. However, the trajectory described by the piecewise linear curve is generally not smooth, especially when the object movement is large between two successive frames. Besides, the robot arm can only be controlled to follow the absolute trajectory of the instructor's motion. Normally different agents have different configurations and therefore different workspaces. For example, when a young girl learns to dance from an adult instructor, she will conduct a similar motion relative to her coordinate system by observing how the instructor moves and extracting the characteristics of the motion, e.g., whether it is a straight line or a curve and in which direction it goes. The absolute trajectories here are different, but we, human beings, perceive and interpret them as being similar. We are more interested in the property that makes two motions similar, i.e. the characteristics of a motion. For example, if the human instructor conducts a circular motion with his finger, the robot should also be controlled to move along a circular path although the radius of that circle may be larger or smaller depending on its configuration.

Given a set of points on the trajectory in 3D space, the trajectory of the origin position of the coordinate system in our case, how can we represent this motion

so that its characteristics can be extracted and transferred to the robot if its configuration is known? Since we are interested in a smooth motion, fitting a cubic curve to the trajectory points and its parametric representation are chosen. The parametric curve representation has the advantage that its variables vary within a certain range, normally $[0, 1]$. If B-spline or Bezier basis functions are used to represent a curve, the curve shape or its characteristics, are completely determined by its control points. A similar trajectory for the robot to replicate the perceived motion within its own workspace, can be generated by translating, rotating, and scaling the control points according to its relative position, orientation, and configuration with respect to those of the instructor. In the following sections, some basic concepts of cubic B-splines and curve representation using cubic B-splines as basis functions are first given, and then the method to find the control points of the trajectory curve given a set of origin positions of the object coordinate system is described. Some invariance properties of parametric curve representation are derived and used to transfer the motion trajectory from one workspace to another. A method for fitting a curve to a set of rotation matrices will be briefly discussed.

5.2 Cubic B-Splines

A three-dimensional curve can be represented parametrically as

$$\mathbf{Q}(\bar{u}) = (X(\bar{u}), Y(\bar{u}), Z(\bar{u})), \tag{76}$$

where $X(\bar{u})$, $Y(\bar{u})$, and $Z(\bar{u})$ are each single-valued functions of the parameter $\bar{u}$ and usually polynomials [2]. They yield the X-, Y-, and Z-coordinates of a point on the curve. Figure 16 shows a parametrically defined two-dimensional curve.

Usually it is not possible to define a satisfactory curve using single polynomials for $X(\bar{u})$, $Y(\bar{u})$, and $Z(\bar{u})$. Instead we can break the curve into some number of pieces called segments, each defined by separate polynomials, and join the segments together to form a piecewise polynomial curve. Certain distinguished values of $\bar{u}$, which correspond to the parametric values at the joints between the polynomial segments and called knots, will be encountered as the parameter $\bar{u}$ varies between

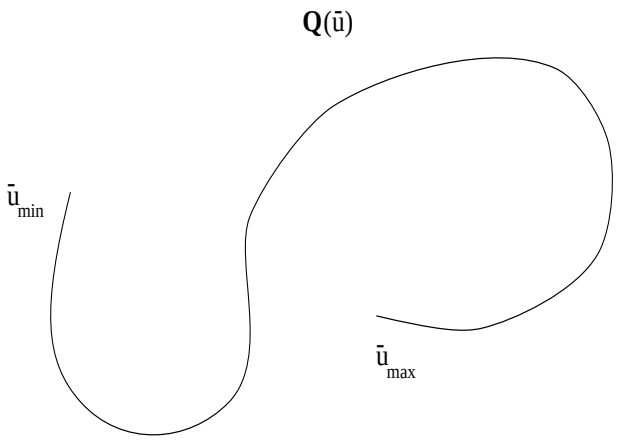


Figure 16: A parametrically defined curve.

the initial minimum value $\bar{u}_{min}$ and the final maximum value $\bar{u}_{max}$. The sequence of knot values is required to be nondecreasing such that

$$\bar{u}_0 \leq \cdots \leq \bar{u}_{min} \leq \cdots \leq \bar{u}_l = \bar{u}_{max} \leq \cdots \leq \bar{u}_{last}.$$

The interval $[\bar{u}_i, \bar{u}_{i+1})$ is called a span.

A cubic B-spline is a spline over an indefinite number of spans, but departs from zero only over precisely four of these spans. One particular property of the B-spline curves is *local control*, by which we mean that altering the position of a single point causes only a part of the curve to change. These points will be called *control vertices* since they are usually connected by straight lines to form the vertices and edges of a control polygon. Figure 17 shows an example of a curve defined by a sequence of control vertices. In section 5.4, we will discuss how to calculate these control points from the given points on the trajectory.

5.3 Representation of a Cubic B-Spline Curve

A cubic B-spline curve can be represented by the following form

$$\mathbf{Q}(\bar{u}) = \Sigma_i \mathbf{V}_i B_i(\bar{u}) = \Sigma_i (X_i B_i(\bar{u}), Y_i B_i(\bar{u}), Z_i B_i(\bar{u})), \tag{77}$$

where $\mathbf{V}_i$ are the control points, and $B_i(\bar{u})$ is the shifted cubic B-spline function. Each function $B_i(\bar{u})$ is composed of four cubic segments, $b_{-0}(u)$, $b_{-1}(u)$, $b_{-2}(u)$, and

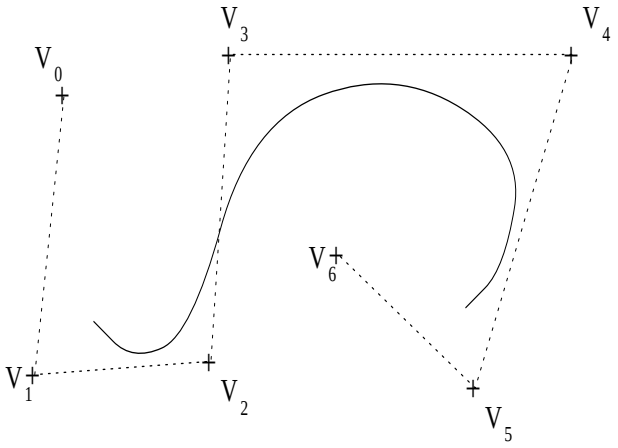


Figure 17: An example of a curve defined by a sequence of *control vertices*, represented here by "+" signs, near which the curve passes. The dotted line connecting the control vertices forms the control polygon.

$b_{-3}(u)$. They can be written as the following equations:

$$\begin{cases} b_{-0}(u) = u^3/6 \\ b_{-1}(u) = (1 + 3u + 3u^2 - 3u^3)/6 \\ b_{-2}(u) = (4 - 6u^2 + 3u^3)/6 \\ b_{-3}(u) = (1 - 3u + 3u^2 - u^3)/6 \end{cases} \tag{78}$$

where the variable u is within the range of $[0, 1)$ and is the local parametrization of each span or interval, $[\bar{u}_i, \bar{u}_{i+1})$, given by

$$u = (\bar{u} - \bar{u}_i)/(\bar{u}_{i+1} - \bar{u}_i). \tag{79}$$

Some examples of cubic B-spline functions are shown in Figure 18 along with the four cubic segments. It can be seen clearly from the figure that the function $B_i(\bar{u})$ is simply a copy of B, shifted so that its support extends from $\bar{u}_i$ to $\bar{u}_{i+4}$. Because the basis functions are nonzero on only four successive spans, if $\bar{u}_i \leq \bar{u} \leq \bar{u}_{i+1}$ then the curve in this span can be represented by

$$\begin{aligned} \mathbf{Q}_i(\bar{u}) &= \Sigma_{r=-3}^{r=0} \mathbf{V}_{i+r} B_{i+r}(\bar{u}) \\ &= \mathbf{V}_{i-3} B_{i-3}(\bar{u}) + \mathbf{V}_{i-2} B_{i-2}(\bar{u}) + \mathbf{V}_{i-1} B_{i-1}(\bar{u}) + \mathbf{V}_{i-0} B_{i-0}(\bar{u}). \end{aligned} \tag{80}$$

If there are $m + 1$ control points indexed from 0 through m, there will be $m - 2$ curve segments bounded by $m - 1$ knots since each curve segment is defined by four

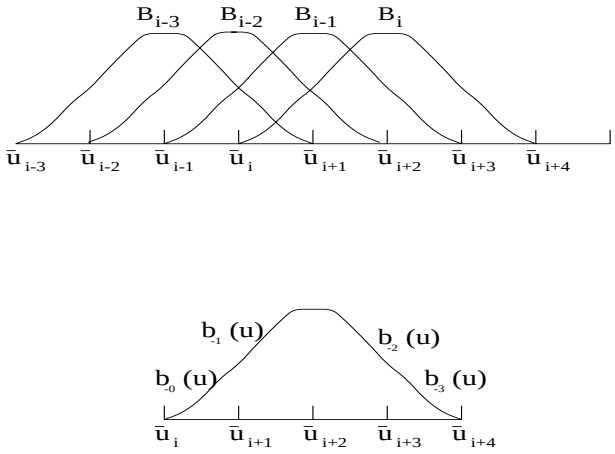


Figure 18: Curves of some cubic B-spline functions.

basis functions. There are $m-1+3+3=m+5$ knots altogether since the leftmost basis function extends three additional intervals to the left of the curve and the rightmost basis function extends three additional intervals to the right.

If we replace each basis function $B_i(\bar{u})$ by the particular segment that pertains to the interval $[\bar{u}_i, \bar{u}_{i+1})$, the above equation can be written as

$$\begin{aligned} \mathbf{Q}_i(u) &= \Sigma_{r=-3}^{r=0} \mathbf{V}_{i+r} b_r \\ &= \mathbf{V}_{i-3} b_{-3}(u) + \mathbf{V}_{i-2} b_{-2}(u) + \mathbf{V}_{i-1} b_{-1}(u) + \mathbf{V}_{i-0} b_{-0}(u). \end{aligned} \tag{81}$$

The curve segment in each span is therefore defined by four control points and the non-zero cubic segments of the basis functions.

5.4 Finding the Control Points

In practice, we are normally given a set of measurement points on the trajectory, so how can we find suitable control points or vertices so that the measurement data points can be fitted with a cubic B-spline curve with a reasonably small error? The approach we take here is to perform a least squares fit of the data by a cubic B-spline curve.

Suppose that we are given $M+1$ measurement data points $\mathbf{P}_i = (x_i, y_i, z_i)$. We want to find a set of $N+1$ control points $\mathbf{V}_j$ that minimize the distance between the cubic B-spline curve they define and the given data points. The number of control

points needed for a specific application depends on the complexity of the curve and whether one wishes to interpolate or approximate the data with the curve. We assume that N is given here. Recall that

$$\mathbf{Q}(\bar{u}) = \Sigma_{j=0}^{N}(X_j B_j(\bar{u}), Y_j B_j(\bar{u}), Z_j B_j(\bar{u}))$$

where the position of the j^{th} control point is represented by (X_j, Y_j, Z_j) so as to distinguish it from the data points (x_i, y_i, z_i) we are fitting. What we will actually minimize is the expression

$$\begin{aligned} Re &= \Sigma_{i=0}^{M}|\mathbf{Q}(\bar{U}_i) - \mathbf{P}_i| \\ &= \Sigma_{i=0}^{M}[(X(\bar{U}_i) - x_i)^2 + (Y(\bar{U}_i) - y_i)^2 + (Z(\bar{U}_i) - z_i)^2] \end{aligned} \tag{82}$$

where $\bar{U}_i$ is some parameter value associated with the i^{th} data point. It can be calculated from the following formula:

$$\bar{U}_{i+1} = \bar{U}_i + (N-2)\frac{|\mathbf{P}_{i+1} - \mathbf{P}_i|}{S}, \tag{83}$$

where $\bar{U}_0 = 0$ and $S = \Sigma_{i=0}^{M-1}|\mathbf{P}_i - \mathbf{P}_{i+1}|$ is the total length of the line segments connecting the data points. As a result, the spacing between the parametric values $\bar{U}_i$ is proportional to the Euclidean distance between their associated data points. Since Expression 82 is quadratic its minimum occurs for those values of X_k, Y_k, and Z_k such that

$$\begin{aligned} \frac{\partial Re}{\partial X_k} &= 0 \\ \frac{\partial Re}{\partial Y_k} &= 0 \\ \frac{\partial Re}{\partial Z_k} &= 0 \end{aligned} \tag{84}$$

where k ranges between 0 and N.

We will consider only the case of X_j, since Y_j and Z_j can be treated analogously. By computing the partial derivative of the above expressions, we obtain

$$\frac{\partial Re}{\partial X_k} = \Sigma_{j=0}^{N}[\Sigma_{i=0}^{M} B_j(\bar{U}_i)B_k(\bar{U}_i)]X_j - \Sigma_{i=0}^{M} x_i B_k(\bar{U}_i) = 0$$

where the basis functions can be calculated once the value $\bar{U}_i$ is known. This will give us $N+1$ such linear equations for $N+1$ unknowns X_k. For most applications it

is important that the first and last data points be interpolated so that the user has explicit control over the endpoints of the curve when manipulating control vertices. By giving the initial and final knots, $\bar{u}_0$ and $\bar{u}_{N+4}$, multiplicity four [2], i.e.,

$$\bar{u}_0 = \bar{u}_1 = \bar{u}_2 = \bar{u}_3 \tag{85}$$

$$\bar{u}_{N+1} = \bar{u}_{N+2} = \bar{u}_{N+3} = \bar{u}_{N+4} \tag{86}$$

the curve will interpolate the first and last data points and we have the end conditions $X_0 = x_0$ and $X_N = x_M$. This will leave us $N - 1$ linear equations. The above equations can be easily written in the form of the matrix equation $AX = B$. A solution can be easily obtained by the conventional singular value decomposition method or any least squares method. The Y and Z coordinates of the control point $\mathbf{V}_j$ can be calculated similarly.

5.5 Properties of Parametric Curve Representation

Once the control points are available, the curve representing the trajectory of the origin point of a moving object's coordinate system can be obtained from Equation 80. As mentioned earlier, one of the advantages of the parametric curve representation is that its shape is completely controlled by the control points. Another property of the parametric curve representation is that the curve shape remains the same if its control points are rotated, scaled, or translated.

Suppose that we rotate the control points by some angle θ. Let R be the matrix accomplishing this rotation. If $Q(\bar{u})$ is the curve defined by the control points $\mathbf{V}_i$, and $Q_r(\bar{u})$ is the curve defined by the control points $R\mathbf{V}_i$, the following can be obtained from Equation 77:

$$Q_r(\bar{u}) = \Sigma_i (R\mathbf{V}_i) B_i(\bar{u}). \tag{87}$$

Since for any matrix M and vectors $\mathbf{a}$ and $\mathbf{b}$, $M\mathbf{a} + M\mathbf{b} = M(\mathbf{a} + \mathbf{b})$, we have

$$Q_r(\bar{u}) = R\Sigma_i \mathbf{V}_i B_i(\bar{u}) = RQ(\bar{u}). \tag{88}$$

Thus we may either rotate the control points and then compute the curve they define, or compute the curve first and then rotate the points lying on it. The two operations result in the same curve.

Since scaling and translation can also be represented as a matrix operation, a similar property holds so that the shape of a cubic B-spline curve is not affected by either scaling or translating the control points. The property of rotation-, scale-, and translation-invariance makes it possible to transfer a motion from one agent (instructor) to another agent (robot) without changing the shape of its trajectory. If the initial position and orientation of a moving object's coordinate system are represented by the vector $\mathbf{T}_0$ and rotation matrix R_0 in the robot coordinate system, we can rotate and translate the control points of the object trajectory by R_0 and $\mathbf{T}_0$ and get the corresponding trajectory for the robot to replicate in its own coordinate system. If the workspaces of the two agents are known, a scale factor can be applied to the control points so that a similar motion can be conducted in a different workspace, e.g., two circles with different radii.

5.6 Fitting a Curve to Rotation Matrices

For a motion involving both translation and rotation, there is also a trajectory for the object's orientation in addition to that of the object's origin. Orientation is usually represented by a rotation matrix. To fit a curve directly to a set of rotation matrices seems difficult since a rotation matrix is orthonormal and its elements are not independent. This is an area in which research is being actively undertaken by mathematicians [43].

However if the rotation matrix R is specified as three successive rotations around the X, Y, and Z axis, and the three corresponding parameters α, β, and γ are calculated, a curve can be fitted to the data points of these three parameters using the above method. Once the curve is defined in terms of these three parameters, the corresponding rotation matrix at any given time instance can be easily calculated from the curve coordinates. Thus any subsequent operation using the rotation matrix can proceed in the usual way.

5.7 Experimental Results

The above curve fitting algorithm has been implemented, but only two dimensional data are used here for the purpose of easy graphical illustration.

Figure 19 shows some two-dimensional data points generated from a quadratic form ($y = x^2$) and the fitted cubic B-spline curves using $4, 5, 6$, and 7 control points. As mentioned above, the number of control points chosen, and thus the number of curve segments, depends on the complexity of the measurement data to be fitted and this will affect the quality of the curve fit. From the figure it can be clearly seen that the quality of the fit is improved as the number of control points increase. Of course, it can also be improved by intelligently selecting where the joints between successive segments occur, and by adjusting the knot values $\bar{U}_i$ associated with data points $\mathbf{P}_i$. Doing so is more expensive and therefore this option is not implemented here. The curves generated here have uniform knot intervals, i.e., the knots are equally spaced, except at the two end points. The squared sum of the residuals, $R\epsilon$, is used here for measuring the quality of the curve fit. The number of control points can be interactively increased if a certain fitting quality needs to be reached. In our implementation, the maximal number of control points is 8, which gives an acceptable fitting accuracy for most of measurement data.

Figure 20(a) shows the pixel coordinates of one tracked corner in the two-dimensional image plane and the fitted cubic curve using 7 control points. It can be seen that a relative good quality has been achieved. The curve shown in Figure 20(b) is generated by scaling the control points by a factor of 0.5. It can be seen that the shape of the curve remains the same while the absolute path is different. This would be the motion trajectory for an agent with a smaller workspace.

Three dimensional data will give similar results, and therefore the motion trajectory of a moving object can be represented by the control points and cubic B-spline basis functions. By manipulating the control points according to the relative configuration of different agents, similar motion trajectories can be generated for an agent to replicate the motion conducted by the primary agent.

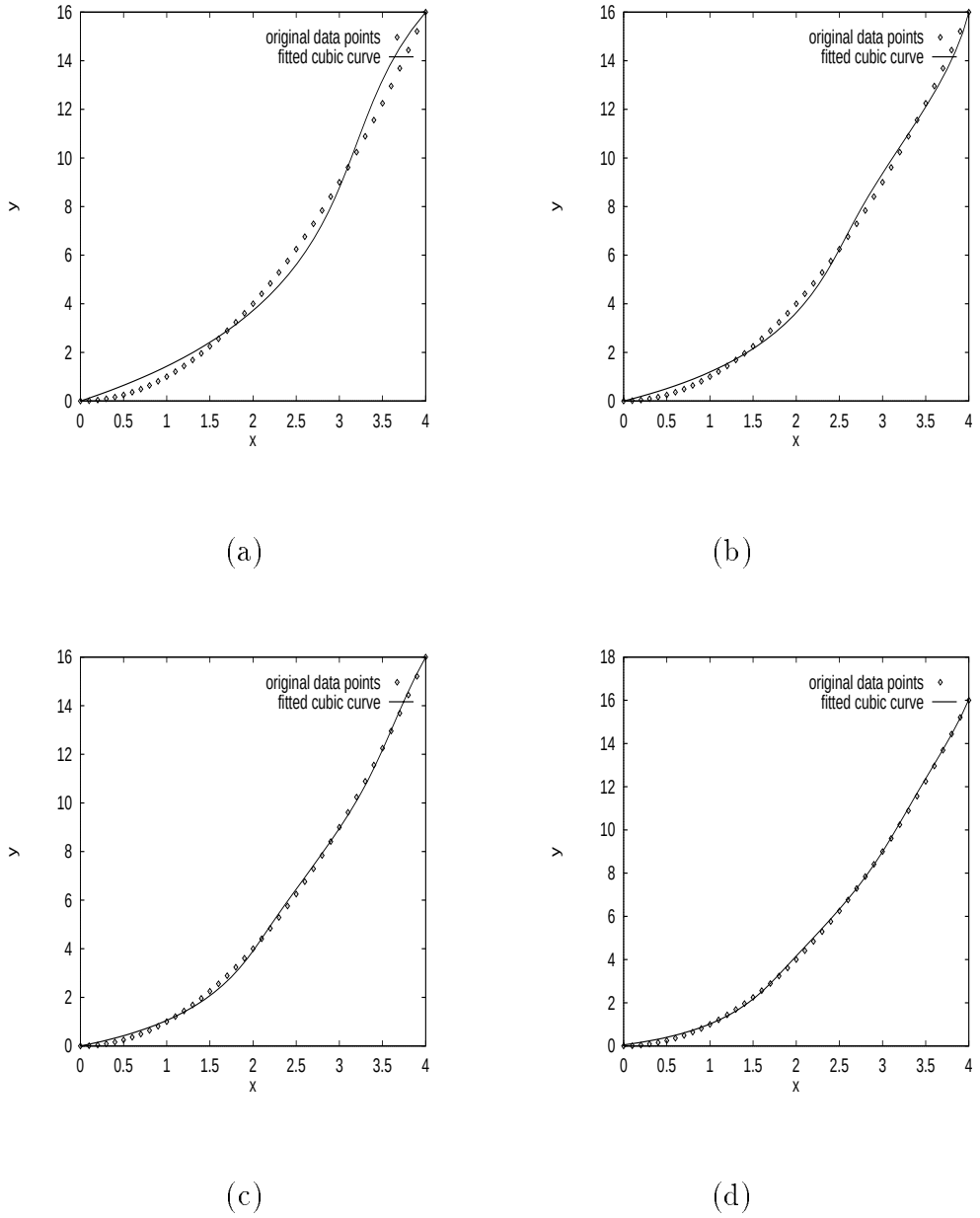
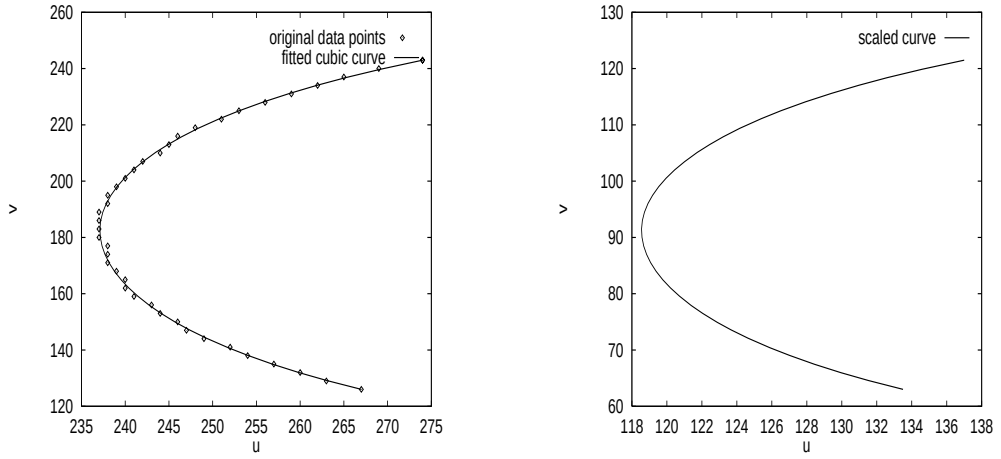


Figure 19: Measurement data points and their fitted cubic curves using 4 (a), 5 (b), 6 (c), and 7 (d) control points.

5.8 Conclusions

In this chapter, we introduced some characteristics of B-spline functions. Data points have been fitted with the parametric curves, which are represented by cubic B-spline basis functions and control points. Once the basis functions are chosen, the control points for a set of measurement data points can be obtained by the least squares method. The number of control points needed depends on the complexity



(a) (b)

Figure 20: (a) Positions of a tracked corner in a 2D image and its fitted cubic curve with 7 control points. (b) A curve generated by the same control points scaled by a factor of 0.5.

of the data set and the required fit quality.

Because the control points completely determine the characteristics of a cubic parametric curve, motion trajectories with the same characteristics for differently configured robots can be easily generated from the instructor's motion trajectory by manipulating, i.e., rotating, scaling, and translating, the control points if it is represented as a parametric cubic B-spline curve.

Chapter 6

Module Integration

This chapter describes an experiment with the robot system proposed in the previous chapters. Putting all the modules together, we demonstrate that a robot arm with such a system can replicate the observed motion with similar shape characteristics.

6.1 Experimental Setup

Figure 21 is a photo of a UMI RTX robot arm and its workspace. A video camera is mounted on the top area and its position and orientation are fixed during the experiment. The work bench is painted black for easy image processing. A human demonstrates a movement by moving his or her arm on the black work bench in front of the video camera. The robot watches this movement through the video camera.

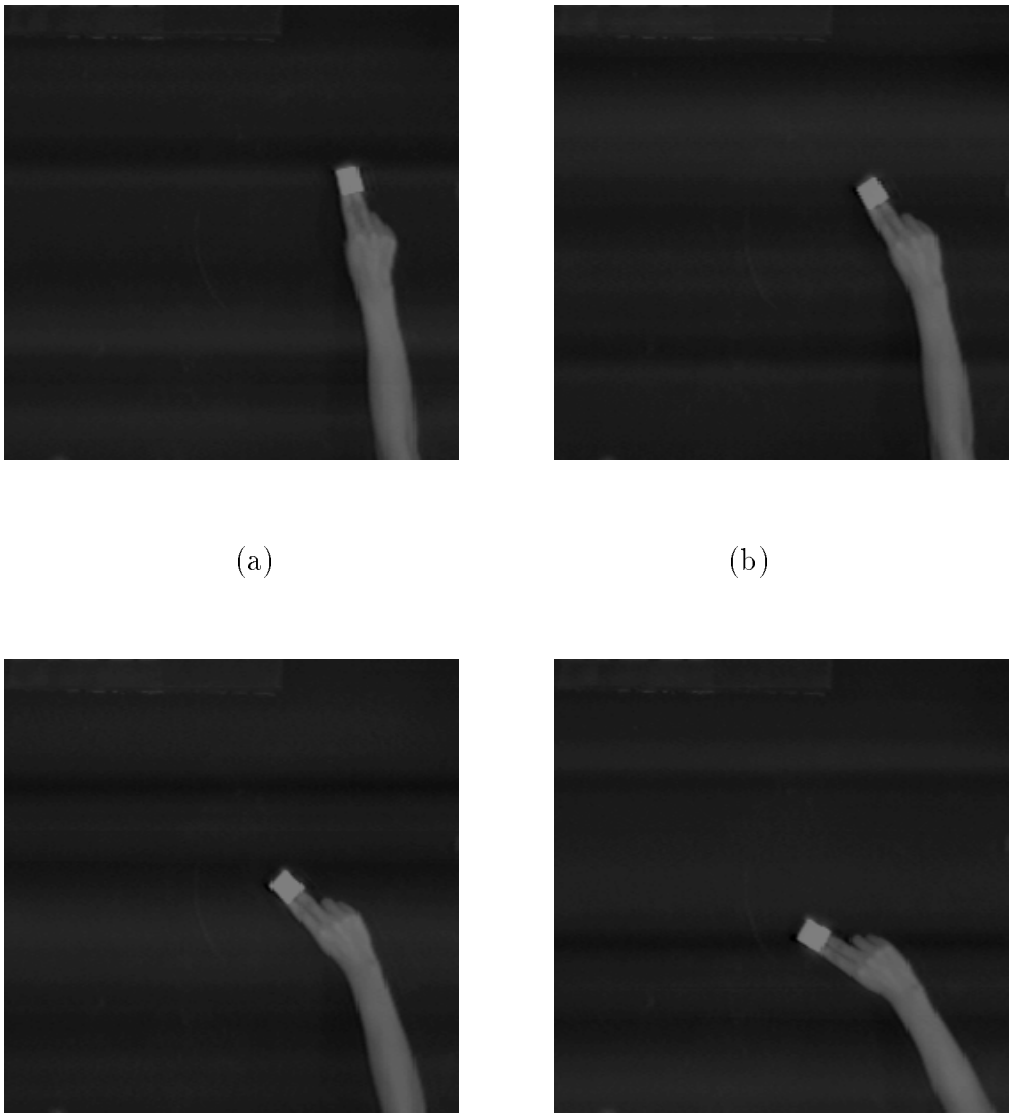
The video camera captures a sequence of images containing the moving arm. The images will be used as input to the corner detection and tracking module. Since the motion is planar and the camera is calibrated in the robot's workspace, i.e., the transformation from a point in the image coordinate system to the corresponding point in the robot's world coordinate system is known, a trajectory in the image coordinate system can be directly used as the input to the program controlling the robot arm movement. The positions in pixel units of the moving arm in the image sequence from the corner tracking module are fitted with a parametric cubic

Figure 21: The robot arm used in our experiment.

B-spline curve. According to the relative position and orientation of the human arm to the robot arm, a trajectory for the robot arm to replicate the human arm movement can be generated by transforming the control points of the curve into the robot's workspace.

6.2 Results of Corner Detection and Tracking

Figure 22 shows several frames from a sequence of 10 images of a moving arm. The results of corner detection and tracking are shown in Figure 23. There are four corners detected, marked as white dots in Figure 23a. The trajectory of one of these four corners is shown in Figure 23b. The corners and the trajectory points are both superimposed on the grey-level image of the first frame in the figure.



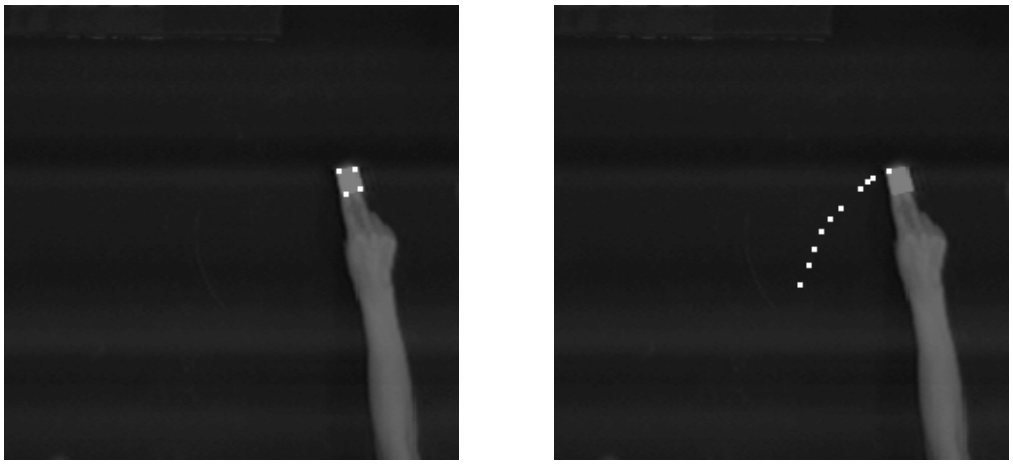
(a) (b)

(c) (d)

Figure 22: Several images of a moving arm. (a) Frame 1. (b) Frame 4. (c) Frame 7. (d) Frame 10

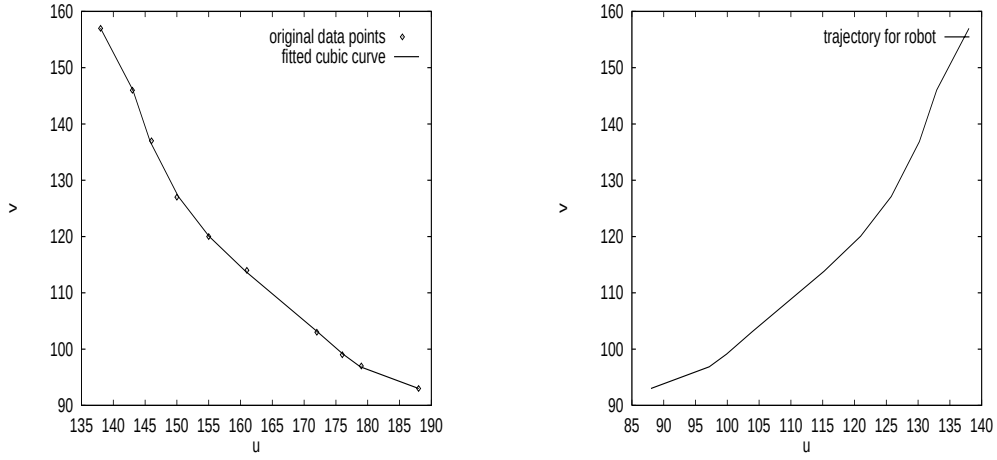
6.3 Motion Representation and Trajectory Generation

The trajectory of one of the four corners on the arm is used as that of the human arm. The position data of this corner are therefore input to the curve fitting module described in Chapter 5. The resulting cubic B-spline curve using 7 control points is shown in Figure 24a. This curve can be transformed to a curve with any scale and



(a) (b)

Figure 23: Results of corner detection and tracking.



(a) (b)

Figure 24: Resulting trajectory curves. (a) The trajectory of the human arm in pixel units. (b) The trajectory generated for the robot in pixel units.

orientation in a different coordinate system. Considering our robot workspace, we rotate the control points by 180 degrees around the Y-axis (the vertical direction in the image), and a mirror curve around the Y-axis can be generated. Figure 24b shows a trajectory curve generated in this fashion for the robot to replicate the corresponding motion.

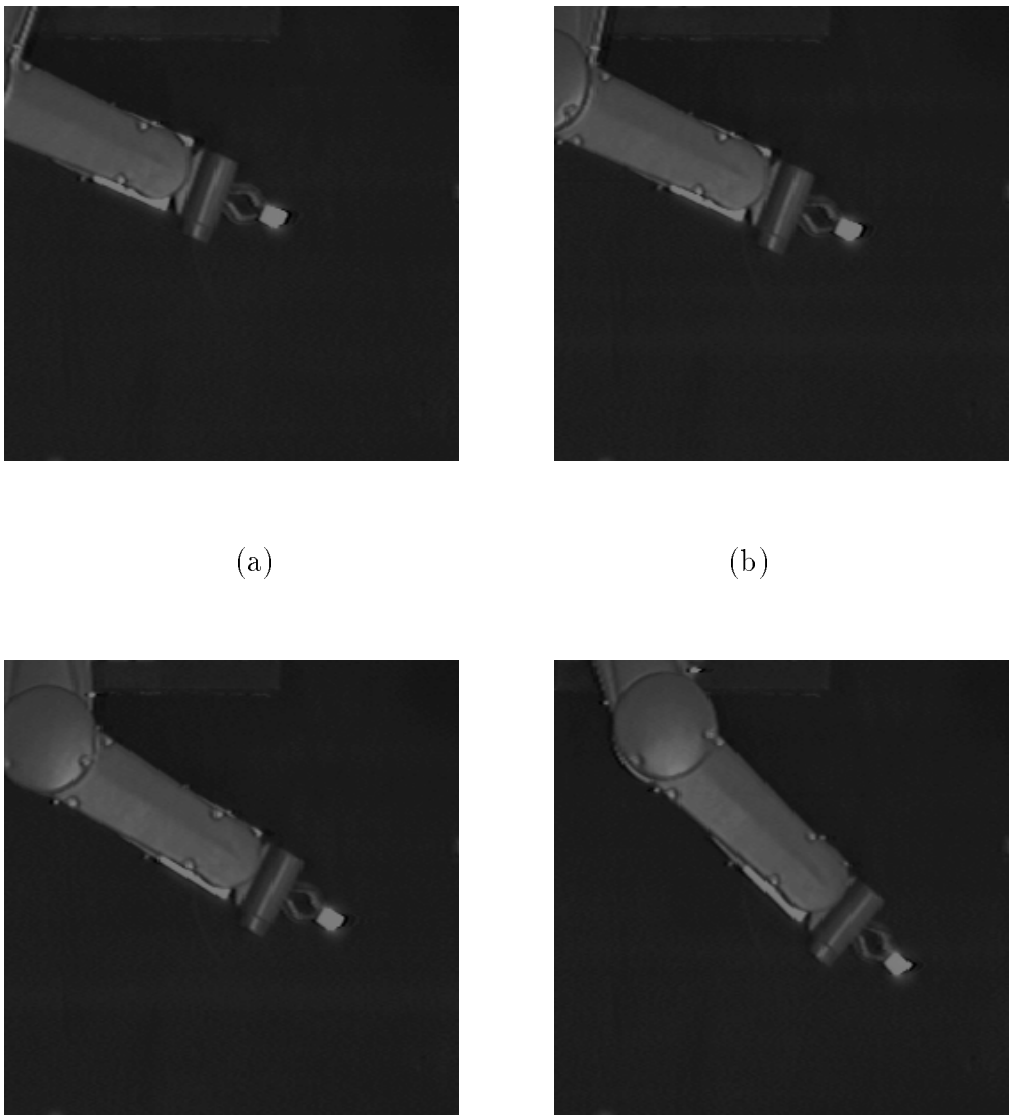
6.4 Motion Replication by a Robot Arm

Points on the above-generated trajectory are input to a program, which controls the robot arm movement. The mapping function from a point in pixel units in the image to a point in robot's world coordinates is first obtained by this program by driving the robot arm to several positions and then calibrating the camera. This mapping function is then used to convert the trajectory data in the image coordinate system to those in the robot world coordinate system. The program finally drives the robot arm to the desired positions according to the trajectory data. Figure 25 shows several frames from the sequence of 10 images of the robot arm replicating the movement of the human arm.

Figure 26 shows the trajectory of a corner point on the robot arm. There is another point on the robot arm marked as a white dot in the image since it is detected by the corner detector also as a strong corner point. It can be clearly seen that this trajectory is a mirror curve of the original trajectory of the human arm movement and it has similar shape characteristics to those of the human arm. Our emphasis is the similarity between end effector trajectories of the robot and human arms. Their actual configurations are not of interest here.

6.5 Conclusions

The various modules described in this thesis have been integrated to demonstrate that the proposed approach is feasible. A robot arm has successfully replicated the motion given by a human instructor. More importantly, the replicated motion has similar shape characteristics to those of the original one due to our reliable motion extraction algorithms and novel motion representation scheme.



(a) (b)

(c) (d)

Figure 25: Several images of of the robot arm replicating the movement of the human arm. (a) Frame 1. (b) Frame 4. (c) Frame 7. (d) Frame 10

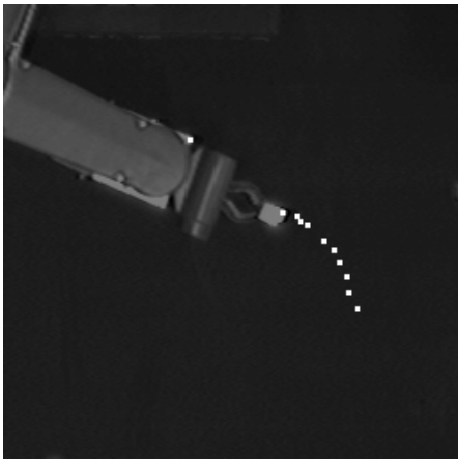


Figure 26: Trajectory of a corner point on the robot arm.

Chapter 7

Conclusion

7.1 Contributions

The goal of the research presented in this thesis has been to develop a robot system that has the ability to see and observe, to extract motion, and to replicate the extracted motion in the sense of similarity. The previous chapters have described such a system from corner detection and tracking to generation of robot trajectory with the same *characteristics* as those of the instructor's trajectory.

The following is a summary of contributions made in this thesis.

- The selection of corner points as features has been proven to be appropriate since they are stable under perspective projection and easy to track in the image sequence.
- A reliable corner detector has been implemented and the results have shown that most types of corners can be correctly detected.
- A novel corner tracking algorithm, which is designed based on the characteristics of dynamic vision, has been proposed and implemented. It is fast and provides better results than the corner tracker based on the Kalman filter in some applications.
- Existing camera calibration algorithms have been reviewed and methods to improve the calibration accuracy have been suggested.

- A reliable algorithm for extracting general 3D motion has been developed and experimental results have shown that this algorithm is robust against errors that occurred during corner detection.

- Parametric curves have been used to represent the motion trajectory. The characteristics of the motion trajectory are then completely determined by the curve's control points. This has the advantage that for robots with different configurations, different absolute trajectories can be generated by manipulating the control points while the characteristics of the motion trajectories remain unchanged.

- All the modules have been integrated to produce a prototype of the 'teaching by showing and learning by watching' method for robot control.

7.2 Future Research Directions

The focus of this thesis was on extracting the motion of a *rigid* object. Human bodies consist of several rigid objects joined together at joints. Before the approach proposed in this thesis can be used, the moving agent has to be segmented into several rigid objects. If motions of all parts of a human body are required, methods of segmenting it into several rigid objects need to be explored [46].

Currently, corner detection has been implemented with an accuracy of ± 1 pixels. If the motion between two frames is small and the fine details of the motion trajectory are required, high accuracy corner detectors need to be developed. One easy way to achieve this is to interpolate the corner response function with a second or third order polynomial function in the neighborhood of corner points detected by the corner detector described in this thesis so that the corner coordinates can be expressed in subpixel units.

For a scene containing several moving rigid objects, corner tracking for each object can proceed on different parallel processors if suitable hardware is available. Even for one object, different corners can be tracked in parallel to achieve the real-time processing speed since the tracking algorithm is parallel in nature.

The motion extraction method proposed in Chapter 4 uses the 3D information obtained from stereo vision. Since the algorithm is least-squares based, more points should be used to improve the accuracy if they are available in the scene.

The parametric curve representation has been successfully used in extracting the characteristics of the motion trajectory. It has a great potential in many other applications. For example, character recognition systems can use it to represent characters so that the recognition method is scale- and orientation-invariant. To take full advantage of this representation, its application in other areas needs to be explored.

The simple experiment described in Chapter 6 has shown that our 'teaching by showing and learning by watching' approach has been successfully used for a robot to replicate a motion demonstrated by a human. Experiments with more complicated motion need to be conducted to demonstrate and explore additional advantages and features of our motion extraction algorithm and representation scheme though this can be easily derived from existing experimental results principally.

Appendix A

Quaternions

In this appendix, we introduce some of the quaternion notations and definitions. A quaternion $\mathbf{q}$ consists of a scale component q_0 and a component of a three dimensional vector $\mathbf{Q} = (q_1, q_2, q_3)^T$;

$$\mathbf{q} = q_0 + \mathbf{Q}. \tag{89}$$

Two quaternions are equal only if the corresponding components are equal. A vector quaternion is a quaternion with zero scalar component. A scalar quaternion is a quaternion with zero vector component. The conjugate of a quaternion $\mathbf{q}$, denoted by $\tilde{\mathbf{q}}$, is defined by

$$\tilde{\mathbf{q}} = q_0 - \mathbf{Q}. \tag{90}$$

The addition of two quaternions is defined by

$$(q_0 + \mathbf{Q}) + (p_0 + \mathbf{P}) = (q_0 + p_0) + (\mathbf{Q} + \mathbf{P}). \tag{91}$$

The multiplication of two quaternions, denoted by "$*$", is defined by

$$(q_0 + \mathbf{Q}) * (p_0 + \mathbf{P}) = (q_0 p_0 - \mathbf{Q} \cdot \mathbf{P}) + (q_0 \mathbf{P} + p_0 \mathbf{Q} + \mathbf{Q} \times \mathbf{P}) \tag{92}$$

where "$\cdot$" and "$\times$" are vector dot and cross products, respectively. It is easy to prove that quaternion multiplication is not commutative (unless a quaternion is scalar), but is associative, and is distributive over addition. The norm of a quaternion, $||\cdot||$, is defined to be a non-negative value given by $||\mathbf{q}||^2 = \mathbf{q} * \tilde{\mathbf{q}}$. It can be proven that $\mathbf{q} * \tilde{\mathbf{q}} = \sum_{i=0}^{4} q_i^2$. Therefore, the norm of a quaternion is equal to the Euclidean

norm of the corresponding four-dimensional vector. A quaternion with a unit norm is called a unit quaternion. So, for a unit quaternion $\mathbf{q}$ we have $\mathbf{q} * \tilde{\mathbf{q}} = 1$.

Bibliography

[1] J. K. Aggarwal, L. S. Davis, and W. N. Martin. Correspondence Processes in Dynamic Scene Analysis. *Proc. of IEEE*, 69:562–572, 1981.

[2] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, 1987.

[3] P. R. Beaudet. Rotational Invariant Image Operators. In *Proc. 4th Int. Joint Conf. on Pattern Recognition*, pages 579–583, 1978.

[4] S. T. Bernard and W. B. Thompson. Disparity Analysis of Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-2:333–340, 1980.

[5] T. O. Binford. Image Understanding: Intelligent Systems. In *Proc. Image Understanding Workshop*, pages 17–28. Morgan Kaufmann Publishers, 1988.

[6] R. C. Bolles, J. H. Kremers, and R. A. Cain. A Simple Sensor to Gather Three-Dimensional Data. Technical Report 249, Stanford Research Institute International, July 1981.

[7] M. Brady. Computational Approaches to Image Understanding. *Computing Surveys*, 14:3–71, 1982.

[8] R. A. Brooks. Symbolic Reasoning among 3-D Models and 2-D Images. *Artificial Intelligence*, 17:285–348, 1981.

[9] R. A. Brooks. Model-Based Three-Dimensional Interpretations of Three-Dimensional Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-15:140–150, 1983.

[10] J. Cooper. *Real-Time Task-Directed Robot Vision.* PhD Thesis, Department of Computer Science, University of Western Australia, 1992.

[11] R. Deriche and G. Giraudon. A Computational Approach for Corner and Vertex Detection. *Int. Journal of Computer Vision*, 10:101–124, 1993.

[12] E. D. Dickmanns and V. Graefe. Dynamic Monocular Machine Vision. Technical Report UniBwM/LRT/WE13/FB/88-3, Fakultaet fuer Luft- und Raumfahrttechnik, Universitaet der Bundeswehr Muenchen, Germany, July 1988.

[13] L. Dreschler and H. Nagel. Volumetric Model and 3D Trajectory of a Moving Car Derived from Monocular TV-Frame Sequences of a Street Scene. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 692–697, 1981.

[14] O. D. Faugeras and G. Toscani. The Calibration Problem for Stereo. In *Proc. Computer Vision and Pattern Recognition*, pages 15–20, Miami Beach, Florida, USA, 1986.

[15] S. Ganapathy. Decomposition Of Transformation Matrices For Robot Vision. In *Proc. Int. Conf. on Robotics and Automation*, pages 130–139, 1984.

[16] L. Gu. A Corner Detector Based on Detecting Edge Termination Points. In *Proc. Australian and New Zealand Conference on Intelligent Information Systems*, pages 427–431, Perth, Australia, December 1993.

[17] L. Gu and R. Owens. Corner Tracking for Dynamic Vision. In *Proc. of WA Computer Science Symposium*, pages 427–431, Perth, Australia, September 1994.

[18] W. R. Hamilton. *Elements of Quaternions.* New York: Chelsea, 1969.

[19] C. Harris. Camera Calibration. Technical report, Roker Manor Research Ltd, 1991.

[20] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proc. Alvey Vision Conference*, pages 147–151, UK, December 1988.

[21] D. J. Heeger. Subspace Methods for Recovering Rigid Motion I: Algorithm and Implementation. *Int. Journal of Computer Vision*, 7:95–117, 1992.

[22] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[23] T. S. Huang. *Handbook of Pattern Recognition and Image Processing.* Academic Press, 1986.

[24] T. S. Huang and C. H. Lee. Motion and Structure from Orthographic Projections. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-11:536–540, 1989.

[25] D. Q. Huynh. *Feature-Based Stereo Vision on a Mobile Platform.* PhD Thesis, Department of Computer Science, University of Western Australia, 1994.

[26] K. I. Kanatani. Structure and Motion from Optical Flow under Orthographic Projection. *Computer Vision, Graphics, and Image Processing*, 35:181–199, 1986.

[27] S. B. Kang and K. Ikeuchi. A Grasp Abstract Hierarchy for Recognition of Grasping Tasks from Observation. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 194–201, 1993.

[28] S. B. Kang and K. Ikeuchi. Toward Automatic Robot Instruction from Perception - Recognizing a Grasp from Observation. *IEEE Trans. on Robotics and Automation*, 9:432–443, 1993.

[29] S. B. Kang and K. Ikeuchi. Robot Task Programming by Human Demonstration: Mapping Human Grasps to Manipulator Grasps. In *Proc. IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems*, pages 97–104, 1994.

[30] L. Kitchen and A. Rosenfeld. Grey-Level Corner Detection. *Pattern Recognition Letters*, 1:95–102, 1982.

[31] J. J. Koenderink. Optic Flow. *Vision Research*, 26:161–180, 1986.

[32] K. Kosuge, T. Fukuda, and H. Asada. Acquisition of Human Skills for Robotic Systems. In *Proc. IEEE Int. Symposium on Intelligent Control*, pages 469–474, 1991.

[33] Y. Kuniyoshi, M. Inaba, and H. Inoue. Teaching by Showing: Generating Robot Programs by Visual Observation of Human Performance. In *Proc. 20th Int. Symposium on Industrial Robots*, pages 119–126, 1989.

[34] Y. Kuniyoshi, M. Inaba, and H. Inoue. Seeing, Understanding and Doing Human Tasks. In *IEEE International Conference on Robotics and Automation*, pages 2–9, 1992.

[35] Y. Kuniyoshi, H. Inoue, and M. Inaba. Design and Implementation of a System That Generates Assembly Programs from Visual Recognition of Human Action Sequences. In *IEEE International Workshop on Intelligent Robots and System*, pages 567–574, 1990.

[36] C. Lee and T. S. Huang. Finding Point Correspondences and Determining Motion of a Rigid Object from Two Weak Perspective Views. *Computer Vision, Graphics, and Image Processing*, 52:309–327, 1990.

[37] H-J. Lee and H-C. Deng. Three Frame Corner Matching and Moving Object Extraction in a Sequence of Images. *Computer Vision, Graphics, and Image Processing*, 52:210–238, 1990.

[38] H. C. Longuet-Higgins. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293:133–135, 1981.

[39] A. D. Marshall and R. R. Martin. *Computer Vision, Models, and Inspection.* World Scientific, 1992.

[40] R. Mehrotra and S. Nichani. Corner Detection. *Pattern Recognition*, 23(11):1223–1233, 1990.

[41] H. P. Moravec. Towards Automatic Visual Obstacle Avoidance. In *Proc. Int. Joint Conf. on Artificial Intelligence*, page 584, 1977.

[42] H. H. Nagel. Displacement Vector Derived from Second-Order Intensity Variations in Image Sequences. *Computer Vision, Graphics, and Image Processing*, 21:85–117, 1983.

[43] L. Noakes. Fitting a Parametric Curve to Data of Rotation Matrices. *Private Communication*, 1994.

[44] J. A. Noble. Finding Corners. *Image and Vision Computing*, 6(2):121–128, 1988.

[45] P. Puget and T. Skordas. Calibrating a Mobile Camera. *Image and Vision Computing*, 8(4):341–348, 1990.

[46] R. J. Qian and T. S. Huang. Motion Analysis of Human Ambulatory Patterns. In *Proc. 11th IAPR Int. Conf. on Pattern Recognition*, pages 220–223, 1992.

[47] S. Ranade and A. Rosenfeld. Point Pattern Matching by Relaxation. *Pattern Recognition*, 12:269–275, 1980.

[48] I. D. Reid and D. W. Murray. Tracking Foveated Corner Clusters Using Affine Structure. In *Proc. 4th International Conf. on Computer Vision*, pages 76–83, Berlin, Germany, 1993.

[49] B. Robbins and R. Owens. The 2D Local Energy Model. Technical Report 94/5, Department of Computer Science, The University of Western Australia, August 1994.

[50] L. Robert. Camera Calibration without Feature Extraction. Technical Report 2204, INRIA, February 1994.

[51] L. Rosenthaler, F. Heitger, O. Kuebler, and R. von der Heydt. Detection of General Edges and Keypoints. In *Proc. 2nd European Conf. on Computer Vision*, pages 78–86, 1992.

[52] L. S. Shapiro, A. Zisserman, and M. Brady. 3D Motion Recovery via Affine Epipolar Geometry. *Int. Journal of Computer Vision*, 16:147–182, 1995.

[53] A. Singh. *Optic Flow Computation: A Unified Perspective.* IEEE Computer Society Press, 1993.

[54] S. M. Smith and J. M. Brady. SUSAN - A New Approach to Low Level Image Processing. Technical Report TR95SMS1b, DRA Chertsey, Surrey, UK, 1995.

[55] A. Stoica. *Motion Learning by Robot Apprentice - A Fuzzy Neural Approach.* PhD Thesis, Department of Electrical and Electronic Engineering, Victoria University of Technology, 1995.

[56] M. Subbarao. Closed Form Solutions to Image Flow Equations for Planar Surfaces in Motion. *Computer Vision, Graphics, and Image Processing*, 36:208–228, 1986.

[57] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, PA, April 1991.

[58] C. Tomasi and T. Kanade. Shape and Motion from Image Streams under Orthography: A Factorization Method. *Int. Journal of Computer Vision*, 9(2):137–154, 1992.

[59] T. R. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proc. of Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[60] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, 1979.

[61] D. Weinshall and C. Tomasi. Linear and Incremental Acquisition of Invariant Shape Models from Image Sequences. In *Proc. Int. Conf. on Computer Vision*, pages 675–682, 1993.

[62] J. Weng, N. Ahuja, and T. S. Huang. Optimal Motion and Structure Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:864–884, 1993.

[63] J. Weng, T. S. Huang, and N. Ahuja. Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11:451–476, 1989.

[64] O. A. Zuniga and R. M. Haralick. Corner Detection Using the Facet Model. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 30–37, 1983.