〔非公開〕

ＴＲ－Ｃ－００９９

Perceptual Kinematics: Vision-based Control of Robot Manipulators

ピーター　クッカ　　大谷　淳
Peter　CUCKA　　Jun OHYA

１９９４　４．１８

ＡＴＲ通信システム研究所

Perceptual Kinematics:
Vision-based Control of Robot Manipulators

Peter CUCKA[*], Jun OHYA
ATR Communication Systems Research Lab

(* Computer Vision Lab, Center for Automation Research,
University of Maryland, College Park,
Maryland 20742-3411, U.S.A.)

Chapter 1

Introduction

Problems involving the control of robot manipulators ("robot arms") typically require knowledge of the relationship between the position and orientation ("pose") of the hand in space and the settings of the joints of the arm. In general, it is easy to determine the pose of the hand given the settings of the joints, but it is difficult to determine how to set the joints to achieve a desired pose of the hand. The former relationship is given by the *kinematic map* and the latter by the inverse of the kinematic map, but in some cases inversion is not possible, and the inversion computations can be unstable. Approximate computations can relieve some of these problems, but they introduce errors of their own. However, by introducing a sensor (for example, a camera) that can provide data that implicitly captures information about the pose of the hand, it is possible to compensate for these errors.

In earlier work [4, 9] we presented a new method for manipulator control in which sensory data (specifically, visual data) are incorporated into a *perceptual kinematic map* (PKM), which expresses the relationship between the settings of the arm joints and quantities measured by the sensor. The method assumes virtually no prior information about the robot's kinematic behavior and therefore avoids the costly and time-consuming calibration required by many existing methods. Instead, it takes advantage of geometric properties of the graph of the PKM, a hypersurface that we call the *control surface*. In this way, manipulator pose control can be achieved using measurements made on images without the need to explicitly recover the pose itself.

This document is divided into two sections. In the first, we review the concepts behind the PKM and PKM-based control (this section also appeared in [4]). In the second, we describe in detail software developed to both simulate a robot manipulator with a PKM controller and to control a real manipulator.

Chapter 2

Perceptual Kinematics: Theory

2.1 Kinematics and Perceptual Kinematics

We begin with formal definitions of the forward and inverse kinematic maps and of the perceptual kinematic map. More comprehensive information about the former can be found in numerous sources, including [6] and [8]. For additional details about the latter, see [9].

2.1.1 Manipulator kinematics

A robot manipulator can be modeled as a sequence of rigid links connected by either revolute (turning) joints or prismatic (sliding) joints. Each link of the manipulator, apart from the last, maintains a fixed relationship between the two joints to which it is attached. The relationship is described by two parameters, l_i and α_i, where l_i is the distance between joints i and $i+1$ measured along the common normal of their axes of rotation or translation (the "length" of the link), and α_i is the angle between the axes in a plane perpendicular to the normal (the "twist" between the two joints). Each joint may also be described by two parameters, d_i and β_i, where d_i is the distance along the joint axis between the two links it connects, and β_i is the angle between the two links in a plane perpendicular to the joint axis. Thus, for a revolute joint, β_i is the only parameter that varies, and for a prismatic joint, only d_i varies [6]. We will let q_i denote the joint variable of the ith joint.

Choosing an appropriate convention, one can systematically associate with each link a coordinate system. The relationship between successive links can then be described by a coordinate transformation, and the configuration of the end effector with respect to the base can be described by a sequence of coordinate system transformations. In practice, this involves the product of a sequence of rotation and translation matrices that are

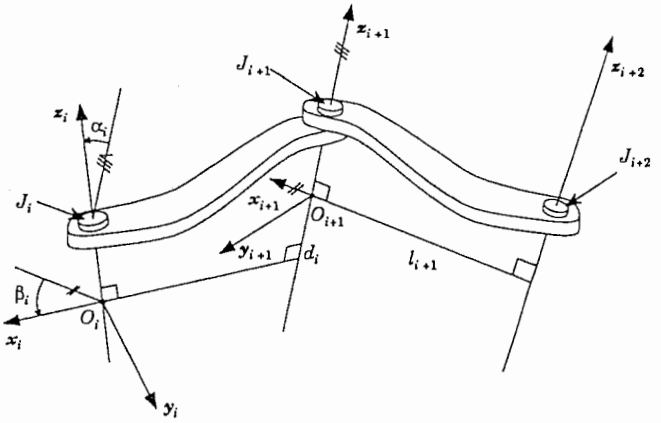

Figure 2.1: Coordinate systems and parameters for a generic link

determined from the parameters l_i, α_i, d_i, and β_i. A common and convenient method for the selection of coordinate systems is the one proposed by Denavit and Hartenberg [8]. Denoting the links as S_0 (the fixed base) through S_n (the end effector) and the joint connecting S_{i-1} and S_i as J_i, we associate a coordinate system $\{O_i, X_i, Y_i, Z_i\}$ with each link S_i such that the Z_{i-1} axis coincides with the axis of joint J_i, the X_i axis is normal to the X_{i-1} axis (with X_0 chosen arbitrarily) and points away from it, and the Y_i axis is chosen in accordance with a right-handed coordinate system (Figure 2.1).

The transformation from one coordinate system to another is described in terms of the parameters l_i, α_i, d_i, and β_i, where β_i is now the angle between X_{i-1} and X_i about Z_{i-1}, d_i is the projection onto Z_{i-1} of the distance from O_{i-1} to the intersection of Z_{i-1} with X_i, l_i is the projection onto X_i of the distance from O_i to the intersection of Z_{i-1} with X_i, and α_i is the angle between Z_{i-1} and Z_i about X_i.

The homogeneous transformation between the coordinate systems of adjacent links S_{i-1} and S_i is given in terms of the Denavit-Hartenberg parameters by the matrix

$$\mathbf{A}_{i-1,i} = \begin{bmatrix} \cos\beta_i & -\cos\alpha_i \sin\beta_i & \sin\alpha_i \sin\beta_i & l_i \cos\beta_i \\ \sin\beta_i & \cos\alpha_i \cos\beta_i & -\sin\alpha_i \cos\beta_i & l_i \sin\beta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The coordinates of a point $\mathbf{M}_n$ on the end effector with respect to the base coordinate system can now be obtained using a series of transformations along the kinematic chain $S_0, S_1, \ldots, S_n$:

$$\mathbf{M}_0 = \mathbf{A}_{0,n}\mathbf{M}_n = \mathbf{A}_{0,1}\mathbf{A}_{1,2}\cdots\mathbf{A}_{n-1,n}\mathbf{M}_n.$$

The signals sent to a manipulator affect the configuration of its joints and thereby the position and/or orientation of its end effector in the work space. The forward

kinematic map of the manipulator is given by a mapping between the *joint space* $\mathcal{J}$ of the manipulator and the *task space* $\mathcal{K}$ of the end effector, the former being the space of all joint configurations $\mathbf{q} = (q_1, q_2, \ldots, q_n)^T$ (recall that $q_i = \beta_i$ for a revolute joint, and $q_i = d_i$ for a prismatic joint), and the latter being the space of possible positions and orientations of the end effector, i.e. the set of all pairs $(\mathbf{p}, \omega)$, where $\mathbf{p} \in \mathbb{R}^3$ is a translation vector relative to the origin of the base coordinate frame and ω is a three-dimensional rotation relative to the axes of the base coordinate frame. The (forward) *kinematic map* is then the following mapping:

$$\begin{aligned} \kappa : \mathcal{J} &\longrightarrow \mathcal{K} \\ \mathbf{q} &\longmapsto (\mathbf{p}, \omega). \end{aligned}$$

2.1.2 The perceptual kinematic map

In the sensorless approach to manipulator control, the first stage is the establishment of the kinematic map for the manipulator, as described above. This problem has been studied extensively, and numerous models and methods have been proposed for its solution. Moving the end effector to a given position and orientation then involves the inversion of the kinematic map. However, because this inversion can be prohibitively difficult to compute, involving complex, nonlinear functions of the joint parameters, inverse kinematics algorithms generally provide only a fast approximation to the solution. The inversion is even more complex, if not impossible, at singular points of the kinematic map (though it generally suffices to determine the locations of these singularities and to avoid them).

With the introduction of a sensor, we need to compose the kinematic map with an additional mapping that describes the manner in which the configuration of the manipulator affects the sensory data. For a visual sensor, this mapping can be determined using the classical pinhole model of a camera, by which the scene is mapped into a viewer-based coordinate system defined as follows: the origin O is the optical center of the camera; the z axis, the optical axis of the camera, intersects the image plane orthogonally at $z = f$, where f is the focal length of the camera; and the x and y axes are parallel to the axes of the image plane and are chosen in accordance with a right-handed coordinate system (Figure 2.2).

Since the kinematic map as defined above specifies the positions of points on the end effector in terms of the base coordinate system, the transformation from the base frame to the viewer frame is required. This is given conveniently by the homogenous

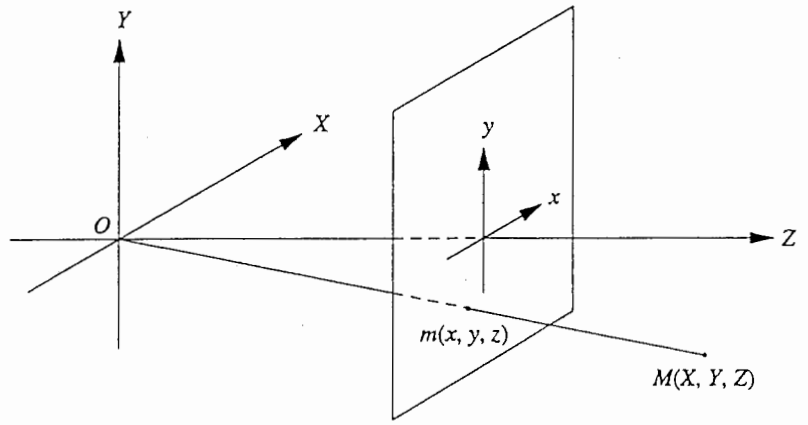

Figure 2.2: Perspective projection of a point onto the image plane

transformation matrix

$$\mathrm{C} = \begin{bmatrix} \mathrm{c}\phi\mathrm{c}\psi - \mathrm{s}\phi\mathrm{c}\theta\mathrm{s}\psi & -\mathrm{c}\phi\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\theta\mathrm{c}\psi & \mathrm{s}\phi\mathrm{s}\theta & x_c \\ \mathrm{s}\phi\mathrm{c}\psi + \mathrm{c}\phi\mathrm{c}\theta\mathrm{s}\psi & -\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{c}\theta\mathrm{c}\psi & -\mathrm{c}\phi\mathrm{s}\theta & y_c \\ \mathrm{s}\theta\mathrm{s}\psi & \mathrm{s}\theta\mathrm{c}\psi & \mathrm{c}\theta & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where s and c denote sine and cosine, and the position of the camera relative to the manipulator's base is given by (x_c, y_c, z_c) and its orientation is given in terms of Euler angles by (ϕ, θ, ψ). The coordinates of a point $\mathbf{M}_n$ on the end effector with respect to the viewer coordinate system are then given by

$$\mathbf{M} = \mathrm{C}\mathbf{A}_{0,n}\mathbf{M}_n.$$

Finally, to determine the point m in the image plane that corresponds to a point M in the scene, let $\mathbf{M} = (X, Y, Z)^T$ be the vector from O to M and $\mathbf{m} = (x, y, f)^T$ be the vector from O to m; then

$$\mathbf{m} = \frac{f}{Z}\mathbf{M} = \frac{f}{\mathbf{z} \cdot \mathbf{M}}\mathbf{M},$$

where $\mathbf{z}$ is a unit vector in the direction of the z axis.

Now consider an array of measurable image features $\mathbf{s} = (s_1, s_2, \ldots, s_m)^T$, where each s_i might, for example, be the x or y coordinate of an image point corresponding to a point on the end effector. (If the goal is to control the pose of the end effector, it is preferable to derive the image features from portions of the image of the end effector: this assumes that the end effector remains in view of the camera at all times.) Evidently

s is a function of the joint parameters. If $\mathcal{S}$ is the set of such arrays, we call the mapping $\pi : \mathcal{J} \rightarrow \mathcal{S}$ a *perceptual kinematic map* (PKM).

In the class of PKMs defined above, we assumed that the camera is fixed relative to the base of the manipulator, and that the camera is able to view the end effector. An equally important possibility is the *eye-in-hand* configuration, in which the camera is attached to the end effector and views a set of fixed landmarks in the scene. (Here the end effector may still be used for manipulation, or the "end effector" may be the camera itself, and the goal may be to position the camera for inspection of an object or area in the scene). In this situation, a different sequence of transformations is needed. A landmark point, given in the coordinate system of the manipulator's base, is first transformed to the coordinate system of the hand:

$$\mathrm{M}_n = \mathrm{A}_{n,0}\mathrm{M} = \mathrm{A}_{n,n-1}\mathrm{A}_{n-1,n-2}\cdots\mathrm{A}_{1,0}\mathrm{M}_0,$$

where the $\mathrm{A}_{i,i-1}$ are the inverses of the corresponding $\mathrm{A}_{i-1,i}$ defined earlier. Then, an additional, constant transformation expressing the pose of the camera with respect to the hand gives the position of the point in the camera coordinate system:

$$\mathrm{M}_c = \mathrm{C}'\mathrm{M}_n.$$

Finally, perspective transformation again gives the image coordinates of the point:

$$\mathrm{m} = \frac{f}{Z}\mathrm{M}_c.$$

In this configuration, the measurable image features might be coordinates of landmark points, or any features derivable from the image of these points.

2.1.3 Differential perceptual kinematics

The PKM involves nonlinear functions of the joint parameters (trigonometric functions of the joint angles) and may also involve nonlinear functions of the image data (used to compute the image features s_i). However, for a sufficiently small change in q (the vector of joint parameters), we have approximately

$$\Delta\mathrm{s} = \mathrm{J}\cdot\Delta\mathrm{q},$$

where J is the Jacobian matrix, which expresses the differential relation between image features and joint parameters:

$$\mathrm{J} = \begin{bmatrix} \frac{\partial s_1}{\partial q_1} & \frac{\partial s_1}{\partial q_2} & \cdots & \frac{\partial s_1}{\partial q_n} \\ \frac{\partial s_2}{\partial q_1} & \frac{\partial s_2}{\partial q_2} & \cdots & \frac{\partial s_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_m}{\partial q_1} & \frac{\partial s_m}{\partial q_2} & \cdots & \frac{\partial s_m}{\partial q_n} \end{bmatrix}.$$

Once again, to achieve a desired pose of the end effector, we require the inverse relation,

$$\Delta \mathbf{q} = \mathbf{J}^{-1} \cdot \Delta \mathbf{s},$$

which expresses the joint displacement required to achieve a desired displacement of the image features.

The Jacobian matrix is invertible only if it is a square matrix of full rank, so the number of image features should equal the number of joints to be controlled. When $\mathbf{J}$ is of less than full rank, its determinant, called the *Jacobian*, is zero and indicates a singular point of the PKM. (Later, we will make use of an alternative, geometric interpretation of the Jacobian matrix as the orientation of the local tangent hyperplane to the hypersurface defined by the PKM.)

2.2 PKM-Based Control

As mentioned in Section 1, there is little point in attempting to invert the PKM exactly, since the computations are likely to be even more complex than those for the original kinematic map, and even small discretization and measurement errors would be likely to render the results unreliable. On the other hand, with the availability of sensory data, simplifying approximations to the inverse PKM can be made more safely, as sensory feedback provides a means to correct the resulting errors. Several methods have been used to simplify computation of the inverse PKM under these conditions, including differential methods, model reference methods, neurally-inspired methods, and a new geometric method that we introduced in [9]. These classes of methods are described in the following subsections.

2.2.1 Related work

Differential feedback schemes, characterized by [5], employ the inverse Jacobian transformation to convert desired image feature perturbations into required joint parameter perturbations. Ideally, the inverse Jacobian matrix must be computed at each cycle of the visual feedback control loop, but the computation is prohibitively time-consuming for real-time control. In [5], this problem is avoided by using a precomputed, constant Jacobian matrix corresponding to the goal configuration of the manipulator; this matrix is determined off-line from models of the camera and manipulator. The method fails in the neighborhood of a singular point, where the Jacobian matrix is ill-conditioned, so the initial configuration must be sufficiently close to the goal configuration.

Model reference methods generally attempt to approximate the complex, nonlinear inverse kinematics by a simplified, linear model with unknown parameters. In the Model

Reference Adaptive Control (MRAC) described in [14], control signals to the manipulator's joints are supplied by the model, and an adaptive controller adjusts the unknown parameters to force to zero the error between the actual response of the system (as determined by the sensory data) and the predicted response (as determined by the model). Similarly, in [13] the unknown parameters of the camera pose are estimated by comparing image measurements with predictions derived from a model of the manipulator's kinematics, which must be determined beforehand. Both of these approaches have been demonstrated only for low-degree-of-freedom ($n \leq 3$) or planar manipulators; parameter estimation for more sophisticated systems may be considerably more difficult.

Biologically-inspired approaches dispense with explicit inverse kinematics computations altogether. The Cerebellar Model Articulation Controller (CMAC) described in [1] combines sensory data into an input vector that is used to address a hash table memory where the corresponding joint parameters are stored, and in [10] a neural network learns both the PKM and its inverse differential map. Both approaches require extensive training and large amounts of memory.

2.2.2 The geometric approach

In earlier work [9] we examined geometric properties of the graph of the PKM, a hypersurface that we call the "control surface." For example, if the manipulator is made to rotate joint J_1 while holding its other joints stationary, we expect that a given point on its end effector will trace an ellipse in the image plane. If we change the settings of the remaining joints, the elliptical image trajectory of the point due to rotation of joint J_1 will also change. Experiments in which we tracked a point on a moving hand show that the elliptical trajectory varies slowly and smoothly as the settings of the other joints vary. For example, Figure 2.3 shows how the image trajectory due to wrist rotation (joint J_5) varies for different settings of the shoulder (joint J_2). Note that the slopes vary little from curve to curve, a property we exploit in our control strategy.

Exploration of the control surface

Each possible configuration of the manipulator corresponds to a point on the control surface; hence in its simplest form, the positioning problem is one of computing a trajectory along the control surface between the two points corresponding to the initial and desired configurations of the manipulator. The trajectory is constrained by limits on the travel of the joints and the presence of singular points on the control surface. In the neighborhood of a nonsingular point, the control surface can be approximated by its tangent hyperplane, facilitating a particularly simple gradient-descent trajectory generation strategy, using a cost function that measures the "distance" between the current and

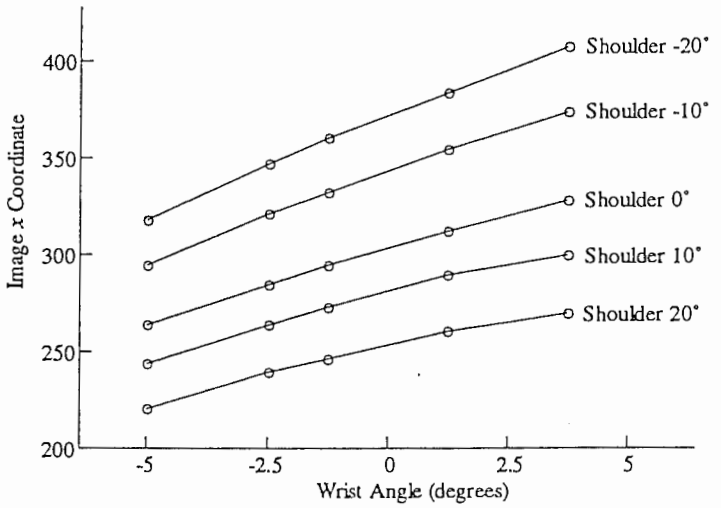

Figure 2.3: Experimental results: x coordinate *vs.* wrist

goal points. (In the neighborhood of a singular point, the surface can have a complex topological structure, and the tangent plane approximation is generally unacceptable.)

At the beginning of each new positioning task, the robot must explore its neighborhood to estimate the local tangents to the control surface, that is, to estimate the Jacobian matrix. The exploration proceeds by perturbing each joint individually by a known Δq_i and measuring the resulting perturbation Δs_j of each image feature. This initial estimate of the Jacobian matrix remains valid, due to the smoothness of the PKM, provided that the robot subsequently avoids singular configurations. In practice, either because the trajectory generator is ignorant of the global structure of the control surface or because the initial and goal configurations lie on opposite sides of a singularity, singular configurations may be unavoidable. Ideally, such configurations are identified by zeros of the Jacobian, but since we are dealing with discrete images and manipulator displacements, we can rely only on qualitative information. As such, the robot detects that it has crossed a singularity by noting a change in the *sign* of the Jacobian.

The list of possible types of singularities of a generic $\mathbb{R}^6 \rightarrow \mathbb{R}^6$ mapping is well known [7]. Singularities of the control surface will typically be folds, which are degenerate along one direction of the control surface. The direction of a fold can be directly determined from the measurements made, and it qualitatively identifies the singularity. After deciding on which side of the singularity the goal lies, the robot explores its neighborhood to recompute the Jacobian matrix and returns to the regular control mode.

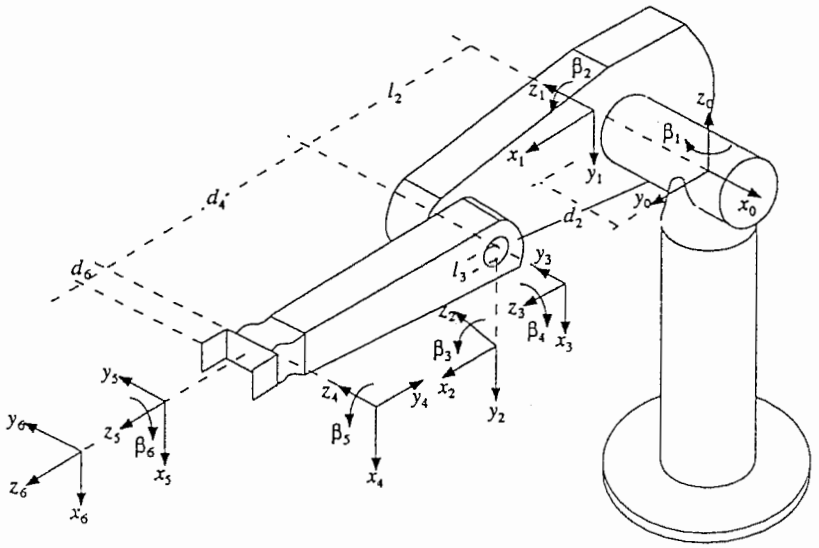

Figure 2.4: Diagram of a PUMA manipulator (redrawn from [6])

J_i	l_i	α_i	d_i	β_i
1	0 mm	$-\pi/2$	0 mm	q_1
2	431.8 mm	0	149.09 mm	q_2
3	-20.32 mm	$\pi/2$	0 mm	$\pi/2$
4	0 mm	$-\pi/2$	433.07 mm	0
5	0 mm	$\pi/2$	0 mm	0
6	0 mm	0	56.25 mm	0

Table 2.1: Denavit-Hartenberg parameters for a PUMA with joints $J_3, \ldots, J_6$ fixed (from [6])

2.2.3 Examples of PKMs

In this section we describe the graph of the PKM for two simple hand/eye systems, using simple choices of geometric features. Both of our examples involve a two-degree-of-freedom manipulator that we obtain by fixing four of the six joints in a six-degree-of-freedom manipulator. Specifically, we use a PUMA manipulator and describe its configuration using the Denavit-Hartenberg parameters, which can be found in [6]. The parameters are listed in Table 2.1 and a diagram of the PUMA robot is given in Figure 2.4.

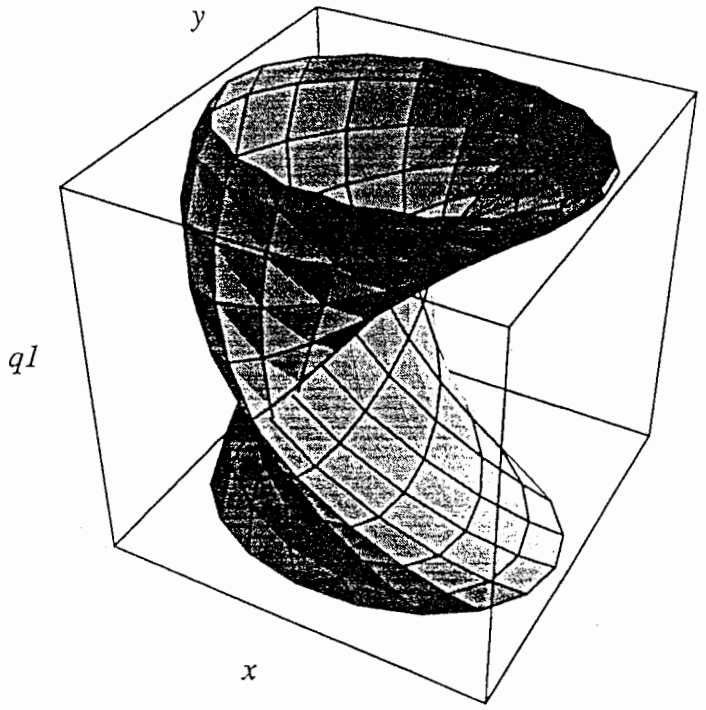

Figure 2.5: Graph of a PKM for a two-degree-of-freedom manipulator viewed by a fixed camera

A fixed camera example

We first assume that the manipulator is viewed by a camera whose position relative to the base of the manipulator is fixed. The camera parameters $(x_c, y_c, z_c, \phi, \theta, \psi)$ are chosen so as to place the camera slightly above and looking down on the robot, and far enough away so that the hand cannot intersect the image plane.

Two image features are required to control the two movable joints of the manipulator (see Section 2.1.3); we choose the two coordinates of the image point p corresponding to the origin of the coordinate system of link S_6 (the point in the "palm" of the hand shown in Figure 2.4). As indicated in Section 2.1.2, we must assume that this point is visible in the image at all times.

With joints J_3 through J_6 fixed in the positions shown in Figure 2.4, the reachable positions of the hand lie on the surface of an ellipsoid whose axes are the axes of rotation of joints J_1 and J_2. For each fixed setting of J_1, the curve traced in the image by the point p through one full rotation of J_2 is an ellipse, as indicated in Section 2.2.2. Thus, in the PKM, the curves of intersection of horizontal planes with the surface are ellipses, where each horizontal plane corresponds to a fixed setting of J_1. Figures 2.5 and 2.6 show two

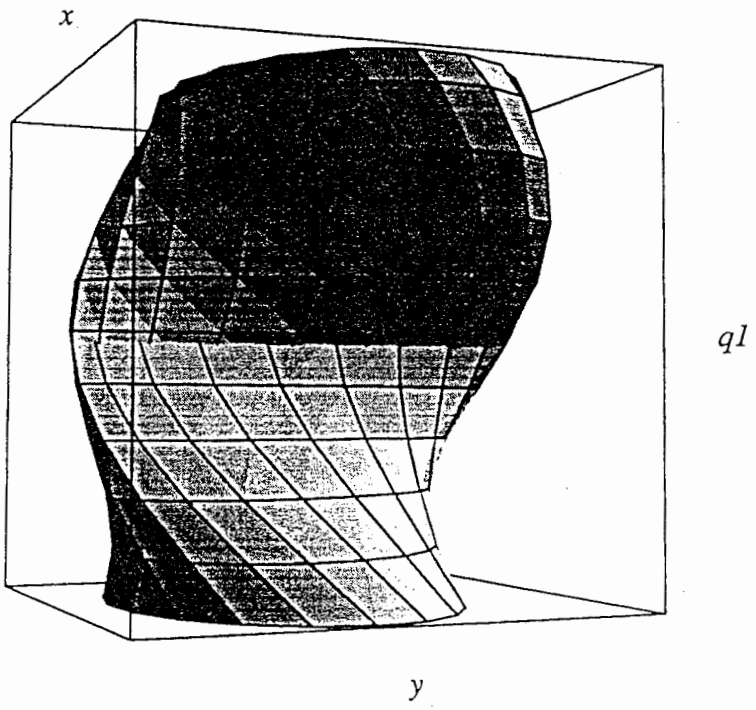

Figure 2.6: Another view of the same PKM

views of this surface for the manipulator defined by the parameters in Table 2.1. Note that the surface exhibits a crease-like singularity corresponding to a degenerate view of the manipulator. That is, for one particular setting of J_1, the point p rotates about the axis of J_2 in a plane perpendicular to the image plane; the curve thus traced in the image is a line segment. (This singularity is introduced by the perspective transformation and is independent of the kinematics of the manipulator.) If the features had been derived from noncoplanar points, this singularity would not have arisen; but for a two- or even three-degree-of-freedom manipulator, when we use the coordinates of two or three image points as features, the singularity will always arise, since three points in space are always coplanar.

An eye-in-hand example

We next consider an eye-in-hand configuration, in which the camera is attached to the end effector. In this situation, changing the settings of the manipulator's joints controls the pose (and hence the field of view) of the camera.

Control for the eye-in-hand configuration is achieved by positioning and orienting the

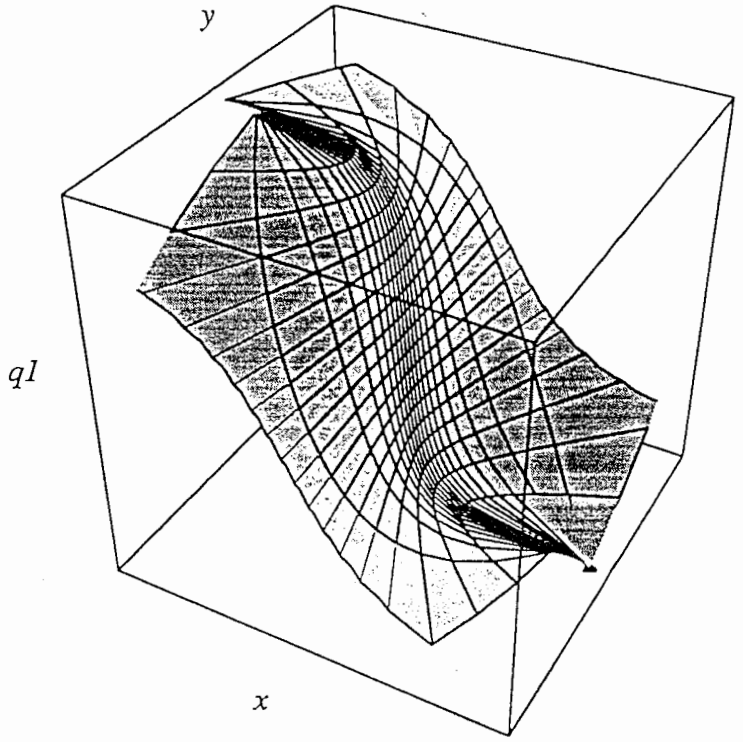

Figure 2.7: Graph of a PKM for a two-degree-of-freedom manipulator in the eye-in-hand configuration

camera in space relative to one or more fixed landmarks. The image features used for control can be functions of points in the image that correspond to landmark points—for example, coordinates of these image points. As before, it is necessary to restrict the motion of the manipulator so that there is a landmark within the field of view of the camera at all times.

To illustrate the PKM in this situation, we used the same PUMA manipulator with the same joints fixed. We used as features the two coordinates of the image point corresponding to a landmark point located approximately ten meters from the base of the manipulator along the y_0 axis (see Figure 2.4). To insure that this point remained within the field of view at all times, J_1 and J_2 were allowed to rotate only a quarter of a revolution. (Figure 2.4 shows the configuration of the manipulator at the midpoint of its trajectory). Figures 2.7 and 2.8 show two views of the graph of the resulting PKM. In this example, the cross sections of the surface are hyperbolas; the surface is not closed as it was in the fixed camera case. This is because the image coordinates of the landmark point lie at infinity when the camera is oriented so that the landmark point lies in the x-y plane of the camera coordinate system (see Figure 2.2). Note that here too the surface exhibits crease-like features.

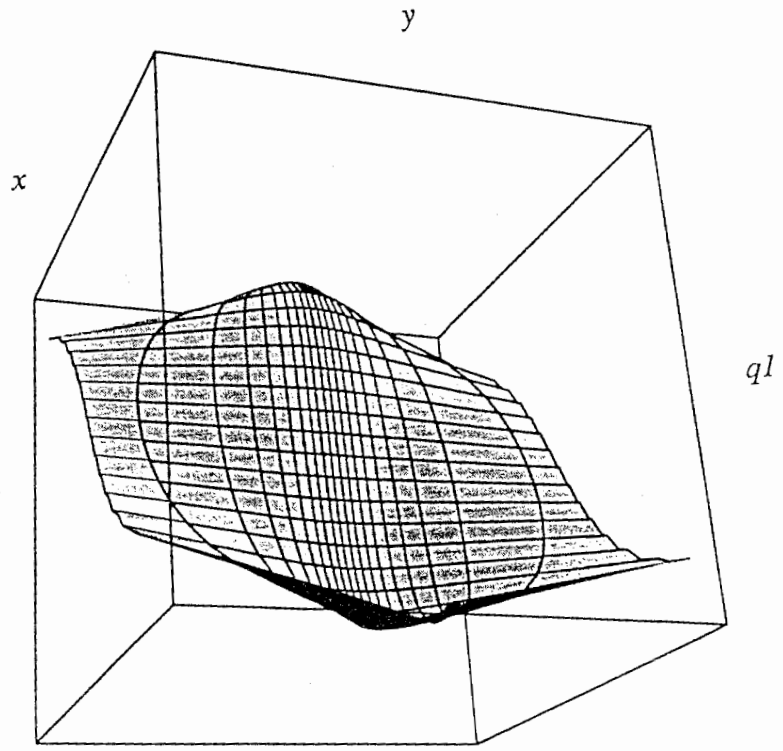

Figure 2.8: Another view of the same PKM

Chapter 3

Perceptual Kinematics: Implementation

We have implemented a software environment in which to experiment with PKM controllers for both real and simulated manipulators. (The simulator currently supports six-degree-of-freedom PUMA 560 and Kawasaki Js-10 manipulators, though others can be easily added.) The software package comprises approximately 10,000 lines of ANSI standard C code and includes facilities for collecting experimental data and a graphical interface based on the X Window System. The program code is divided into a number of source and header files, which are summarized in Table 3.1. In addition, the X Window application resource file `HandEye.ad` and the system preferences file, usually called `handeyerc`, are supplied.

We begin this chapter with an overview of the system and proceed to describe each of its principal components in detail. Finally, we present initial results.

3.1 Overview

Figure 3.1 provides an overview of the structure and the flow of control of the experimental environment. Solid arrows indicate the flow of control, and dashed arrows indicate the flow of data. Unshaded boxes represent modules of the main control loop, and shaded boxes represent auxiliary modules whose behavior depends on whether the system is being used to control a simulated or a real manipulator.

The system is supplied with an image feature vector $\mathbf{s}_f$ corresponding to the goal configuration of the manipulator. At each iteration of the control loop, the "cost" of the current configuration is computed, for example by measuring the Euclidean distance $|\mathbf{s}_f - \mathbf{s}|$ between the current and goal image feature vectors. Using gradient descent, the

Filename	Function
HEMain.c	Main event loop (initializes the system and handles user interaction)
HEControl.c	Main control loop (also allocates memory for data structures)
HEXWindow.c	X Window-specific user interface code (also, interface code for the Nexus image processor and the Kawasaki Js-10 robot)
HEGraphics.c	Non-X-specific graphics and user interface code
HERobot.c	Generic robot interface code
HEPrefs.c	Code to read the "preferences" file
HEOutput.c	Code to output experimental data from the system
kinematics.c	Kinematic simulation of a multi-degree-of-freedom manipulator
perception.c	Simulation of camera geometry and computation of the PKM
displacement.c	Cost functions used for trajectory generation
kalman.c	Kalman filter code
ananu_lin_alg.c	Linear algebra routines (mostly taken from [12])
ananu_util.c	Additional matrix and vector routines
HEDefs.h	System-wide constants and type definitions
HEProtos.h	ANSI function prototypes
Q_variance.h	System noise variance matrix for Kalman filter
js-10.h	Parameters of the Kawasaki Js-10 robot, for use in the kinematics routines
puma_560.h	Parameters of the PUMA 560 robot, for use in the kinematics routines.

Table 3.1: Component files of the software package

joint displacement required to minimize this cost is given by the relation

$$\Delta \mathbf{q} = \mathbf{J}^T \cdot (\mathbf{s}_f - \mathbf{s})$$

(see [9] for details).

An initial estimate of the Jacobian matrix is computed by perturbing each of the manipulator's joints individually, as described in Section 2.2.2. Then, taking advantage of the smoothness of the PKM, a Kalman filter is used to track both the image feature

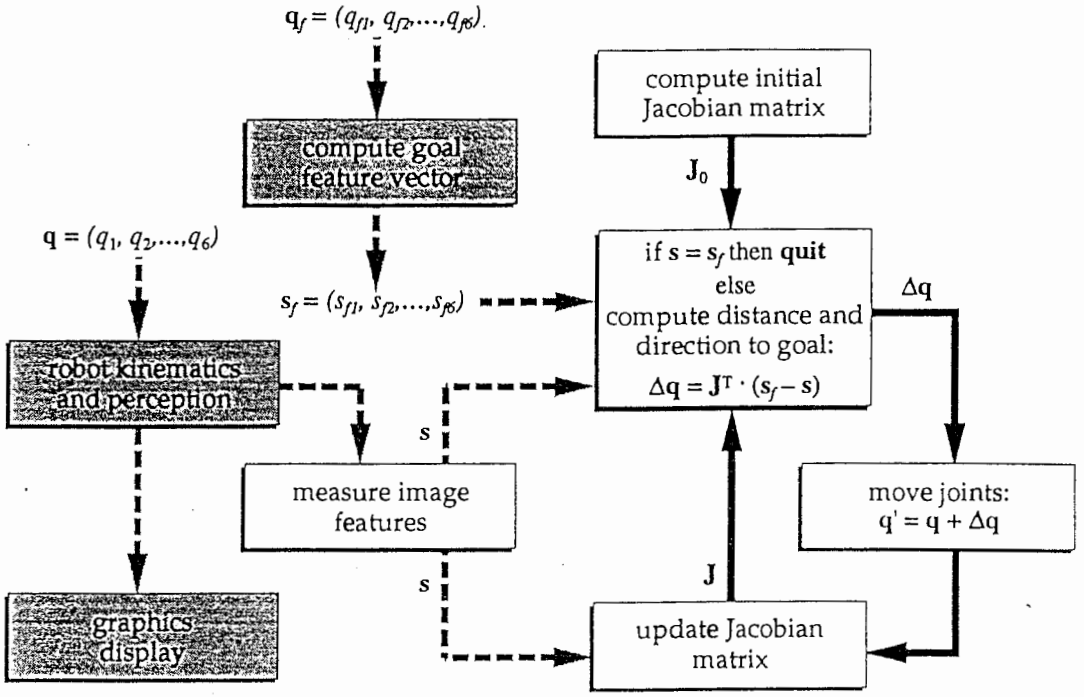

Figure 3.1: Overview of the experimental environment

vector and the measured Jacobian matrix. The estimate of the Jacobian matrix is refined using derivatives measured in the direction of the joint displacement as the robot maneuvers on the control surface.

3.2 System Components

3.2.1 Cost functions

One way to obtain a trajectory in the image space $\mathcal{S}$ between the current and final points s and s_f is to define a *cost function* $\mathcal{C}(\mathrm{s}) = d(\mathrm{s}, \mathrm{s}_f)$ on $\mathcal{S}$ that measures some "distance" to the goal. We want to minimize this image distance, but we can do so only indirectly, by moving in the joint space $\mathcal{J}$. We therefore define a "global" cost function $\mathcal{G} = \mathcal{C} \circ \pi$, which measures distances in $\mathcal{J}$, and apply a gradient descent strategy:

$$\mathrm{q}(k+1) = \mathrm{q}(k) - \gamma \mathbf{u},$$

where $\gamma \geq 0$ is an arbitrary gain and $\mathbf{u}$ is a unit vector in the direction of the gradient of $\mathcal{G}$:

$$\overrightarrow{\text{grad}}(\mathcal{G}) = \left(\frac{\partial \mathcal{G}}{\partial \mathbf{q}}\right)^{\mathcal{T}} = \left(\frac{\partial \pi}{\partial \mathbf{q}}\right)^{\mathcal{T}} \cdot \left(\frac{\partial \mathcal{C}}{\partial \mathbf{s}}\right)^{\mathcal{T}}.$$

For positioning tasks, a simple cost function that measures the Euclidean distance $|\mathbf{s}_f - \mathbf{s}|$ between the current and goal image feature vectors performs well. (Figure 3.2 illustrates a trajectory on the surface defined by such a cost function for an actual two-degree-of-freedom positioning task.) When orientation must be controlled as well, a somewhat more complicated strategy is needed. In the sensorless approach, a pose control task is typically decomposed into an initial three-degree-of-freedom positioning step followed by a six-degree-of-freedom fine pose control step. We cannot directly apply this scheme, however, since, in order for our Kalman filter (see the following section) to maintain a correct estimate of the Jacobian matrix, the directional derivatives of the PKM must not be zero over long sequences of joint motions. Instead, we decompose the joint displacement vector into "arm" (position) and "wrist" (orientation) motions:

$$\Delta\mathbf{q} = \Delta\mathbf{q}_a + \Delta\mathbf{q}_w = \gamma_a \mathbf{u}_a + \gamma_w \mathbf{u}_w.$$

The arm motion is determined as above by the gradient of the cost function. Then, using the current estimate of the Jacobian matrix, a vector of predicted image measurements $\mathbf{s}_a$ is computed by applying the arm motion alone:

$$\mathbf{s}_a(\mathbf{q} + \Delta\mathbf{q}) = \mathbf{s}(\mathbf{q}) + \mathbf{J} \cdot \Delta\mathbf{q}_a.$$

Finally, a wrist motion is chosen so as to minimize the difference between the current and desired orientations of the manipulator's hand. This is done by introducing a measure of "parallelness" $\mathcal{P}$ that is zero when the two orientations are identical and reaches a maximum when they are orthogonal. (Exactly how $\mathcal{P}$ is defined depends on the nature of the image measurements.) Then $\mathbf{u}_w$ is chosen so as to minimize

$$\mathcal{P}[\mathbf{s}(\mathbf{q} + \Delta\mathbf{q})] = \mathcal{P}[\mathbf{s}(\mathbf{q}) + \mathbf{J} \cdot \Delta\mathbf{q}_a + \gamma_w \mathbf{J} \cdot \mathbf{u}_w].$$

As in any minimization problem, the global cost function may have local minima. In order to guarantee that the goal is reached, the point reached during the positioning step must lie within the attraction domain of the global minimum. In the event that a local minimum is reached, any subsequent joint perturbation will result in a higher-cost configuration. Having observed that this is the case, the system can respond by temporarily increasing the gain so as to escape the attraction domain of the local minimum. While this approach does not guarantee success (and has not yet been implemented), we expect it to give reasonable results in most cases.

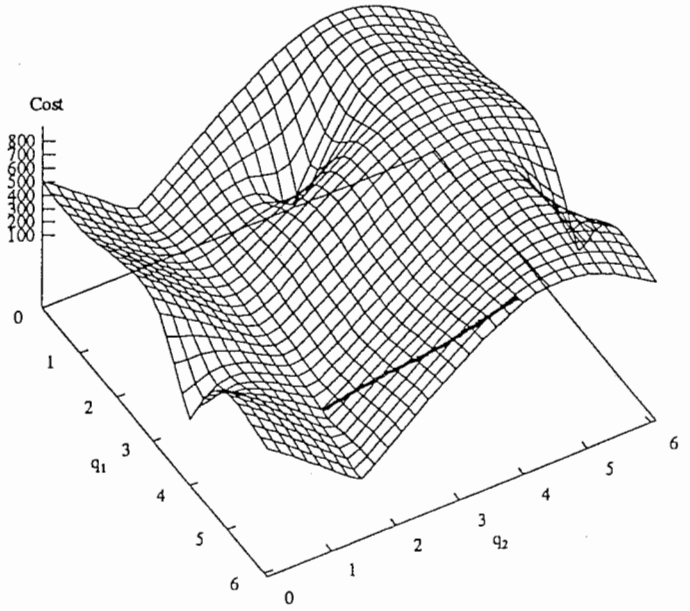

Figure 3.2: Trajectory on the surface defined by a cost function

3.2.2 Kalman filter

In choosing a sensory feedback approach to the manipulator control problem, we are trading the static calibration problems faced by sensorless methods, as well as some of the methods described in Section 2.2.1, for dynamic visual tracking problems. At each iteration of the control loop we need to have both the current values of the image measurements and a reasonable estimate of the Jacobian matrix of the PKM at $\mathbf{q}$. We can obtain an acceptable initial estimate of $\mathbf{J}$ by computing directional derivatives in the neighborhood of the initial configuration, as described in Section 2.2.2. However, as the robot begins to move, we are able to compute the derivative only in the direction of joint displacement $\mathbf{u}$:

$$D_u \pi(\mathbf{q}) = \mathbf{J}(\mathbf{q}) \cdot \mathbf{u}.$$

Using a Kalman filter we can simultaneously track both the image feature vector and the Jacobian matrix (at least, given the partial information that is available). For an n-degree-of-freedom manipulator, the state vector $\mathbf{x}(k)$ of the Kalman filter at time kT is an $n(n+1) \times 1$ column vector comprising the n image measurements and the $n \times n$ elements of the Jacobian matrix:

$$\mathbf{x}(k) = \left[\mathbf{s}^T, \frac{\partial s_1}{\partial \mathbf{q}}, \frac{\partial s_2}{\partial \mathbf{q}}, \ldots, \frac{\partial s_n}{\partial \mathbf{q}} \right]^T.$$

As noted in Section 2.2.2, we observe that the Jacobian matrix varies little in the neighborhood of a regular point $\mathbf{q}$ of the joint space (though this is not the case in the neighborhood of a singular point). If we therefore assume that $\mathbf{J}(k+1) \approx \mathbf{J}(k)$, then, using a first order Taylor expansion of the PKM:

$$\mathbf{s}(k+1) = \mathbf{s}(k) + \gamma \mathbf{J}(k) \cdot \mathbf{u} + \text{h.o.t.},$$

where $\mathbf{w} = \gamma \mathbf{u}$ is the joint displacement at time kT, we can use the following state equation:

$$\begin{aligned} \mathbf{x}(k+1) &= \begin{matrix} \\ n\{ \\ n^2\{ \end{matrix} \overset{\begin{matrix} n & n^2 \end{matrix}}{\begin{pmatrix} \mathbf{I}_n & \mathbf{W} \\ 0 & \mathbf{I}_{n^2} \end{pmatrix}} \cdot \mathbf{x}(k) + \mathbf{v}_k \\ &= \mathbf{A}_k \cdot \mathbf{x}(k) + \mathbf{v}_k, \end{aligned}$$

where $\mathbf{v}_k$ is the $n(n+1) \times 1$ disturbance vector, and $\mathbf{W}$ is an $n \times n^2$ matrix defined by the $n \times 1$ joint displacement vector $\mathbf{w}$:

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}^T & & & 0 \\ & \mathbf{w}^T & & \\ & & \ddots & \\ 0 & & & \mathbf{w}^T \end{pmatrix}.$$

The n image measurements made at each iteration define the $n \times 1$ measurement vector $\mathbf{y}(k)$ (as noted above, we cannot directly measure the remaining elements of the state vector). This is expressed in terms of the state vector as follows:

$$\begin{aligned} \mathbf{y}(k) &= \overset{\begin{matrix} n \times n & n \times n^2 \end{matrix}}{\begin{pmatrix} \mathbf{I}_n & 0 \end{pmatrix}} \cdot \mathbf{x}(k) + \mathbf{r}_k \\ &= \mathbf{C}_k \cdot \mathbf{x}(k) + \mathbf{r}_k, \end{aligned}$$

where $\mathbf{C}_k$ is the $n \times n(n+1)$ measurement matrix and $\mathbf{r}_k$ is the $n \times 1$ measurement error vector.

Given the initial mean and variance of the state vector, we seek an unbiased linear estimate $\hat{\mathbf{x}}(k)$ of $\mathbf{x}(k)$ that minimizes the variance estimate

$$\mathbf{P}(k) = \mathrm{E}[(\mathbf{x}(k) - \hat{\mathbf{x}}(k))(\mathbf{x}(k) - \hat{\mathbf{x}}(k))^T].$$

If we assume that $\mathbf{v}_k$ and $\mathbf{r}_k$ are white Gaussian noise processes uncorrelated with either the initial state or with each other then $\hat{\mathbf{x}}(k)$ can be found by the Kalman-Bucy algorithm, described in [2] and elsewhere.

One problem that commonly arises in Kalman filtering is the determination of the variance matrices of the noise sources. Here, it is reasonable to assume that the n image measurements are independent, so that we can conveniently choose $\mathrm{Var}(\mathbf{r}_k) = \alpha \mathbf{I}_n$, with $\alpha \in [0, 1]$. The variance of the system noise is harder to determine, however. One or more of the higher order terms in the Taylor expansion of the state equation can be used to approximate the system noise, but this requires an analytical expression of the PKM. A reasonable alternative is to acquire these data over the course of a number of manipulation tasks. For now, though, we settle for $\mathrm{Var}(\mathbf{v}_k) = \beta \mathbf{I}_{n(n+1)}$, $\beta \in [0, 1]$.

3.2.3 Kinematics and perception

The experimental system incorporates a complete kinematic model of a generic six-degree-of-freedom manipulator, allowing a wide range of existing manipulators to be simulated (Denavit-Hartenberg parameters for PUMA 560 and Kawasaki Js-10 manipulators are included in the package). A pinhole camera, whose desired pose relative to the manipulator can be specified in the system preferences file, is also modeled, completing the perceptual kinematic map. While these facilities are not needed for controlling a real manipulator, they supply the "image measurements" that are made while controlling a simulated manipulator (and allow for an animated, graphical display of the manipulator). The details of the kinematic and perceptual kinematic map calculations have already been discussed in Section 2.1. Here, we describe the computation of the PKM Jacobian matrix, for a fixed camera configuration.

Extending the notation of Section 2.1.2, let $\mathbf{M}_{0_i}$, $i = 1, \ldots, n/2$, be the coordinates of the ith feature point on the end effector with respect to the coordinate system of the manipulator's base, and let $\mathbf{M}_{n_i} = [X_i, Y_i, Z_i]$ be the (fixed) coordinates of the same point with respect to the coordinate system of the end effector. Then

$$\mathbf{M}_{0_i} = \mathbf{p} + \omega \times \mathbf{M}_{n_i}$$

for some translation $\mathbf{p} = [p_x, p_y, p_z]$ and rotation $\omega = [\omega_\phi, \omega_\theta, \omega_\psi]$ determined by the current values of the joint parameters via the forward kinematic map. We first compute the derivatives $d\mathbf{M}_{0_i}/d\mathbf{q}$ of the kinematic map and then of the full PKM:

$$\begin{aligned} \frac{d\mathbf{M}_{0_i}}{d\mathbf{q}} &= \frac{d\mathbf{p}}{d\mathbf{q}} + \frac{d}{d\mathbf{q}}(\omega \times \mathbf{M}_{n_i}) \\ &= \dot{\mathbf{p}} + \dot{\omega} \times \mathbf{M}_{n_i} \end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} \dot{p}_x + Z_i\dot{\omega}_\theta - Y_i\dot{\omega}_\psi \\ \dot{p}_y + X_i\dot{\omega}_\psi - Z_i\dot{\omega}_\phi \\ \dot{p}_z + Y_i\dot{\omega}_\phi - X_i\dot{\omega}_\theta \end{bmatrix} \\
&= \begin{bmatrix} 0 & Z_i & -Y_i & 1 & 0 & 0 \\ -Z_i & 0 & X_i & 0 & 1 & 0 \\ Y_i & -X_i & 0 & 0 & 0 & 1 \end{bmatrix} [\dot{\omega} \quad \dot{\mathrm{p}}]^T \\
&= [\mathbf{H}_i \quad \mathbf{I}_3][\dot{\omega} \quad \dot{\mathrm{p}}]^T,
\end{aligned}
$$

where the antisymmetric matrix $\mathbf{H}_i$ corresponds to the linear application $\dot{\omega}\times$.

Next, we transform base system coordinates into camera system coordinates by application of the appropriate rotation (the Euler matrix $\mathbf{E}$ corresponding to the upper left 3×3 submatrix of the camera system transformation matrix C from Section 2.1.2):

$$\dot{\mathrm{M}}_i = \mathbf{E} \cdot \dot{\mathrm{M}}_{0_i}.$$

Finally, in terms of image coordinates,

$$\dot{\mathrm{m}}_i = \frac{1}{Z_i} \begin{bmatrix} f & 0 & -x_i \\ 0 & f & -y_i \end{bmatrix} \dot{\mathrm{M}}_i,$$

and the PKM Jacobian matrix is the $n \times n$ matrix given by

$$\mathbf{J} = [\dot{\mathrm{m}}_1, \dot{\mathrm{m}}_2, \ldots, \dot{\mathrm{m}}_{n/2}]^T.$$

(Note that if n is odd, one of the image coordinates of one of the feature points should be discarded.)

3.3 Preliminary Results

At this point, the system, running in its simulation mode, is producing promising results, some of which are reproduced below. We hope to be able to present similar results with a real manipulator in the near future. One of the issues that we would like to address is camera calibration, so that we can compare the Jacobian matrix measured with the real camera with theoretical data produced by the simulator using an accurate perceptual kinematic model.

The six-degree-of-freedom simulated robot performs pose control tasks between randomly or user-selected initial and goal joint configurations. The manipulator is viewed by a simulated fixed camera whose pose relative to the manipulator, specified by the camera parameters $(f, x_c, y_c, z_c, \phi, \theta, \psi)$, can be selected by the user or chosen at random

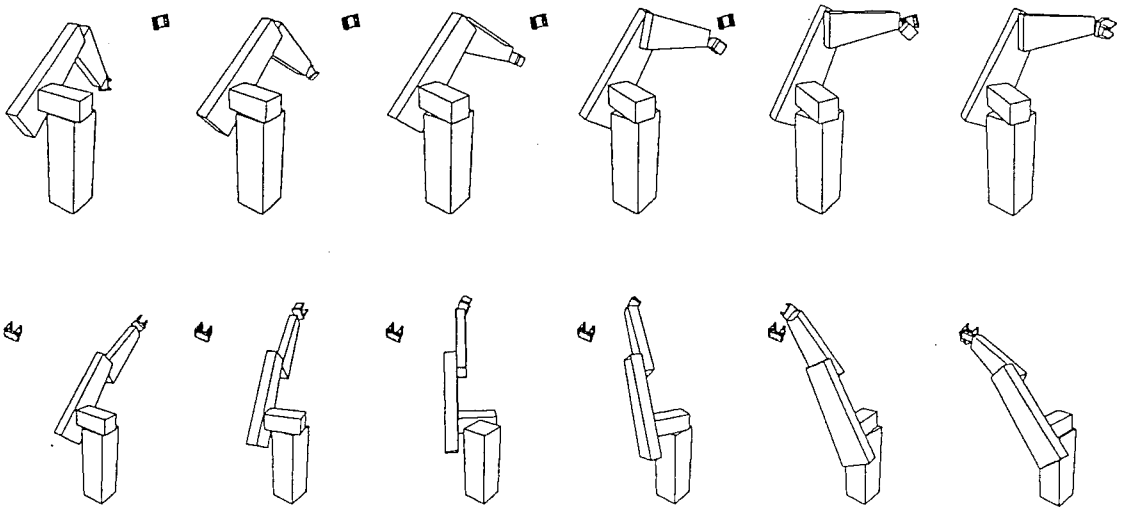

Figure 3.3: Sequences from the simulation of two positioning tasks by a PUMA 560 manipulator

(but so as to ensure that the manipulator lies within the camera's field of view). Two sequences from the simulation are shown in Figure 3.3 (the gray object in each frame is an image of the hand as it should appear in the goal configuration). The two sequences were generated using the same camera parameters but different, randomly-selected initial and goal configurations of the manipulator.

As the Kalman filter tracks the measured Jacobian matrix, the kinematic and perceptual modules simultaneously compute the exact Jacobian matrix. This allows us to compare the two and determine the accuracy of the Kalman filter's estimates. In Figure 3.4 we compare estimated and actual local tangents in the direction of joint displacement $\mathbf{u}$ along the trajectory shown in the first sequence. The plots illustrate the exact and measured values of the derivatives $dx_1/d\mathbf{u}$, $dy_1/d\mathbf{u}$, and $dx_2/d\mathbf{u}$ at each of the fifty-five iterations of the control loop, where x_1, y_1, and x_2 are three of the six measured image features. (Plots for the remaining three features are very similar and are therefore omitted here.) Note that, while the estimated values do not always agree with the true values, the two curves are qualitatively very similar, indicating that the PKM varies slowly as expected.

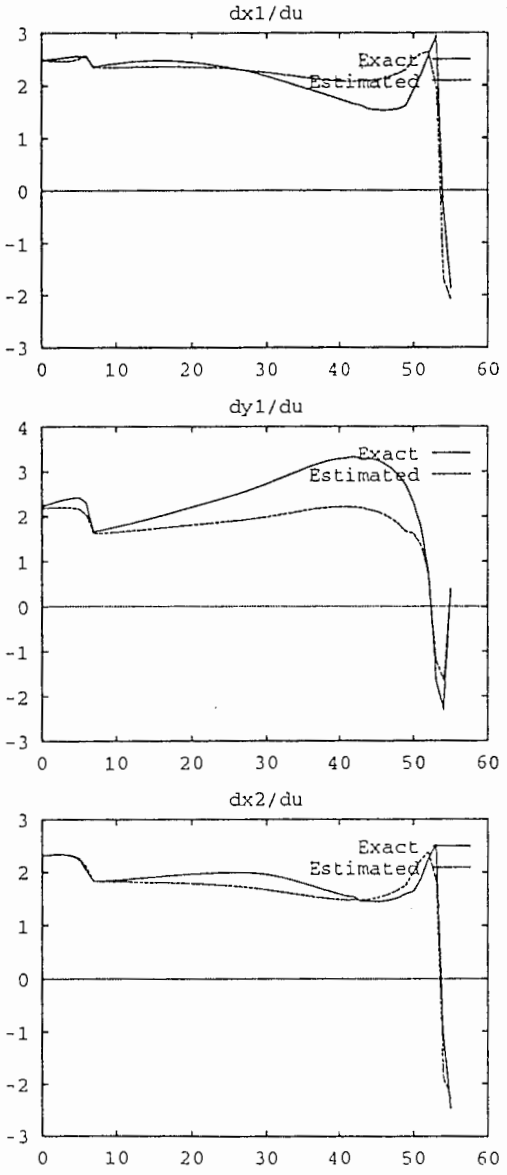

Figure 3.4: Comparison of estimated and actual local tangents (slope *vs.* iteration) for the first sequence in Figure 3.3

3.4 Summary and Future Work

We have demonstrated a new method for sensor-based control of manipulators based on the analysis of perceptual kinematic maps and their corresponding control surfaces, and we have shown that pose control is possible without costly inverse kinematics computations or time-consuming calibration.

Our approach can potentially benefit from a learning strategy, in that it is possible to sample and record the PKM while "exploring" the control surface by performing many manipulation tasks. This knowledge can be used in subsequent tasks to simplify trajectory generation. In contrast to connectionist methods, however, trajectory generation can be accomplished without any training whatsoever, simply by relying on the tangent plane approximation and on local estimates of directional derivatives. In future, we plan to investigate appropriate learning strategies.

Bibliography

[1] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control* **97**(9), 1975, pp. 220-227.

[2] C. K. Chui and G. Chen, *Kalman Filtering with Real-time Applications*, Springer-Verlag, New York, 1987.

[3] P. Cucka, R. Sharma and J-Y Hervé, "Grasping goals for visual coordination," in *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville, Virginia, 1991, pp. 51–56.

[4] P. Cucka, J. Ohya and F. Kishino, "Perceptual kinematics: vision-based control of robot manipulators," Technical Report IE93-77 of the Institute of Electronics, Information and Communication Engineers (IEICE), Tokyo, Japan, 1993.

[5] J. T. Feddema and O. R. Mitchell, "Vision guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation* **5**(5), 1989, pp. 691-699.

[6] K. S. Fu, R. C. Gonzalez and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill, New York, 1987.

[7] M. Golubitsky and V. Guillemin, *Stable Mappings and their Singularities*, Springer-Verlag, New York, 1973.

[8] R. S. Hartenberg and J. Denavit, *Kinematic Synthesis of Linkages*, McGraw-Hill, New York, 1964.

[9] J-Y Hervé, P. Cucka, and R. Sharma, "Qualitative visual control of a robot manipulator," in *Proceedings of the 1990 DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, 1990, pp. 895–908.

[10] B. W. Mel, *Connectionist Robot Motion Planning: A Neurally-Inspired Approach to Visually-Guided Reaching*, Academic Press, San Diego, California, 1990.

[11] W. T. Miller, "Sensor-based control of robotic manipulators using a general learning algorithm," *IEEE Journal of Robotics and Automation* **3**(2), 1987, pp. 157–165.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, New York, 1988.

[13] S. B. Skaar, W. H. Brockman and W. S. Jang, "Three-dimensional camera space manipulation," *International Journal of Robotics Research* **9**(4), 1990, pp. 22–39.

[14] L. E. Weiss, A. C. Sanderson and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation* **3**(5), 1987, pp. 404–411.