

Sensor-Based Autonomous Control of Dynamic Robotic Manipulation

by

Mei Zhang

Department of Mechanical Engineering
McGill University, Montréal

A Thesis submitted to the Faculty of Graduate Studies and Research in partial
fulfillment of the requirements of the degree of Master of Engineering

May 1995

©Mei Zhang, 1995

Abstract

Autonomous grasping in dynamic environments where the object, the robot or both are moving is an increasingly common task in robotic applications. This thesis describes an online trajectory planner – a geometric controller – which evaluates a nonlinear memoryless function to map the current object position and velocity into a desired robot pose. If the robot tracks these set points, it is guaranteed to match the object’s velocity and acceleration on a specified grasp surface. A planar simulation demonstrates that this paradigm performs favorably when compared with the traditional planning approach. Since it does not depend on future state predictions, no object model is required. Without trajectory prediction, the computational effort is drastically reduced, allowing for higher controller speed and tracking feedback gains. The geometric controller has been successfully implemented on the 7 dof Sarcos Dextrous Arm.

In order to provide the critical local sensing just before robot-environment contact, this thesis reports on work in progress towards the development of a proximity sensing network, located in a robot’s multi-fingered gripper. This network will form an integral part of a multistage sensing system with vision and tactile sensor pads. Sensing information is passed on to geometric controller which provides a framework for general sensor-based control of robotic tasks.

Résumé

La prise autonome d'un objet est une tâche de plus en plus commune dans les environnements dynamiques où l'objet et/ou le robot sont en mouvement. Cette thèse décrit un planificateur de trajectoire en temps réel – un contrôleur géométrique – qui évalue une fonction non-linéaire sans mémoire pour faire correspondre la pose désirée du robot à la position et à la vitesse instantanées de l'objet. Si le robot suit la trajectoire formée par cet ensemble de poses, il sera synchronisé avec la vitesse et l'accélération de l'objet sur une surface de prise donnée. Une simulation bidimensionnelle démontre que ce paradigme fonctionne mieux que les méthodes traditionnelles. Puisque notre méthode ne dépend pas de l'état futur, l'objet n'a pas besoin d'être modélisé. Sans la prédiction de la trajectoire, le calcul numérique est considérablement allégé, permettant un contrôleur plus rapide et des gains plus élevés. Nous avons implémenté le contrôleur géométrique sur le bras agile Sarcos (Sarcos Dextrous Arm) avec succès.

Cette thèse présente aussi la recherche en cours sur le réseau de capteurs de proximité qui est situé sur les doigts du robot pour capter suffisamment d'information. Le réseau sera une partie intégrale d'un système sensoriel à plusieurs étapes, comprenant des capteurs visuels et tactiles. L'information des capteurs est transmise au contrôleur géométrique et le tout constitue une structure générale de contrôle de tâches robotiques.

Acknowledgements

I would first like to thank Professor Martin Buehler for the opportunity, supervision and especially the creative ideas provided throughout this thesis work. Without him, I would not have been able to work on this project. I recalled the first time I had a chance to talk with him, I was impressed by his unique idea about dynamic grasping via the prototype of our later “geometric controller”. At that time, I had just been to this country for three months, it was a great challenge for me to develop the control algorithm and finally implement it. Throughout this thesis work, I have learned a great deal from Professor Buehler, his solid background in robotics and practical experiences in both hardware and software developments. The opportunities to expand my knowledge and resources provided by Professor John Hollerbach are also gratefully acknowledged. Dr. Yangming Xu, Donghai Ma, Lydia Giugovaz and Nadim El-fata are also thanked for the assistance to my experimental work.

I would also like to thank my husband Yongsheng Liu who has steadfastly provided support. His generosity and understanding have been truly appreciated. Finally I thank my family for the encouragement and faith they have given me.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Background | 1 |
| 1.2 | Survey of Dynamic Grasping and IR Sensing | 3 |
| 1.3 | Contributions | 5 |
| 2 | Geometric Controller | 7 |
| 2.1 | Problem Formulation | 7 |
| 2.2 | Traditional Replanning Approach | 9 |
| 2.3 | Background of Geometric Controller: Robot Juggling | 10 |
| 2.4 | Development of Geometric Controller | 12 |
| 2.5 | Theory of Geometric Controller | 14 |
| 2.5.1 | Vertical Components | 14 |
| 2.5.2 | Horizontal components | 15 |
| 2.5.3 | Spherical coordinates | 17 |
| 2.5.4 | Orientation | 18 |
| 2.6 | Dynamic simulation and comparisons | 20 |
| 3 | Real-Time Implementation | 22 |
| 3.1 | Kinematics | 22 |
| 3.1.1 | Forward kinematics | 22 |
| 3.1.2 | Inverse kinematics | 28 |
| 3.2 | Open Loop Kinematic Calibration | 36 |

| | | |
|----------|--|-----------|
| 3.2.1 | The Kinematic Model | 36 |
| 3.2.2 | Iterative identification | 37 |
| 3.2.3 | Parameter Scaling | 40 |
| 3.2.4 | Calibration on Sarcos Dextrous Arm | 41 |
| 3.3 | Experiment Setup and Data Analysis | 42 |
| 3.4 | Gravity Compensation | 47 |
| 4 | Infrared Proximity Sensing | 49 |
| 4.1 | Local Sensor Requirements | 49 |
| 4.2 | Local Sensor Options | 51 |
| 4.3 | Infrared Sensor Model | 53 |
| 4.4 | IR sensor Calibration and Characterization | 55 |
| 4.5 | Object Localization Strategy | 58 |
| 4.6 | Object Localization Experiment | 60 |
| 4.7 | Future Work | 61 |
| 5 | Conclusions | 62 |

List of Figures

| | | |
|------|--|----|
| 2.1 | 1 dof illustration | 11 |
| 2.2 | Juggle followed by a catch (1), from [4]. | 11 |
| 2.3 | End Effector Orientation Control Variables | 18 |
| 2.4 | Orientation Control for Parabolic Object Trajectory, 1: object, 2: robot | 19 |
| 2.5 | Parabolic Object Trajectories (1: object, 2: geometric controller, 3: replanning, const. acceleration, 4: replanning, const. velocity) | 20 |
| 2.6 | Sinusoidal Object Trajectories (1: object, 2: geometric controller, 3: replanning, const. acceleration, 4: replanning, const. velocity) | 21 |
| 3.1 | DH frame assignment | 24 |
| 3.2 | The Sarcos Dextrous Arm Geometry Structure | 26 |
| 3.3 | The redundancy corresponds to rotating the elbow about the shoul- der/wrist line | 28 |
| 3.4 | Reduction to a planar manipulator for θ_4 by referring positions to link 1 coordinates | 29 |
| 3.5 | Elbow position correspond to rotational angle | 31 |
| 3.6 | Find θ_3 | 33 |
| 3.7 | Denavit-Hartenberg coordinates and end effector vector \mathbf{b}_j^i | 36 |
| 3.8 | Experiment Setup | 42 |
| 3.9 | Experiment Block Diagram | 43 |
| 3.10 | Dynamic Grasping Result in Z Dimension | 45 |
| 3.11 | Dynamic Grasping Result in X and Y Dimension | 46 |

| | | |
|------|---|----|
| 3.12 | Vector Definition for Gravity Compensation | 47 |
| 4.1 | Global and Local Sensing for robotic manipulation | 50 |
| 4.2 | Triangulation Based local sensing system | 51 |
| 4.3 | Phase Based local sensing system | 52 |
| 4.4 | Amplitude Based local sensing system | 53 |
| 4.5 | Albedo Parameter Calibration via Sensor Fusion | 55 |
| 4.6 | Calibration Experimental Setup | 56 |
| 4.7 | Output Voltage Curve versus Distance or Angle | 57 |
| 4.8 | Surface fitting with measurement data (circles) | 57 |
| 4.9 | Albedo Parameter Calibration | 58 |
| 4.10 | Sensing a Planar, Rectangular and Circular Object | 59 |

Chapter 1

Introduction

1.1 Background

Robotics is advancing rapidly into applications which require increasing dexterity and dynamic response. This is heightening the demand for tightly integrated sensing and control, “sensor-based control,” to support autonomous or supervisory operation. In teleoperation sensor-based autonomous control of subtasks can greatly enhance system performance especially where accurate and/or fast robot movement is required, or where time delays in the signal paths would compromise bandwidth, for example when operating in remote areas or in space. Under certain circumstances, full master-slave operation is too slow, inaccurate or impossible due to time-delays between master and slave. In manufacturing setups, much or all of the environment is known, and in this scenario, the critical pre-contact proximity information can be acquired by a number of small Infra-Red (IR) sensor pairs, embedded in the robot fingers. Such a sensor is inexpensive, robust and, when integrated with machine vision and tactile sensing, provides a suitable solution particularly for dynamic grasping tasks, where a smooth robot-object contact is required.

A smooth and dynamic grasp of a moving object is defined as a match between the position, velocity and acceleration of the object and the robot’s end effector at the

grasp position. In order to facilitate tracking, velocity and acceleration at initial time t_0 should match as well. In addition, we consider avoiding pre-grasp collisions between robot and object. In theory, we generate a desired robot trajectory to eliminate the errors between the object and the desired robot trajectory. In practices, we optimize the controller to reduce the tracking errors between the desired and the actual robot trajectories.

Dynamic grasping is a fundamental task that precedes many manipulation operations and is by itself the main constituent of a large class of robotic tasks. It has various applications in industrial and commercial fields. In manufacturing, robots pick up objects from conveyor belts and J-hooks, and will in the near future have to grasp objects with a priori unknown trajectories handed to them from humans or other robots. In space, this problem occurs when capturing satellites or when containing and docking space assemblies. In other tasks, for example robotic live line maintenance of live power lines, both the robot on its boom as well as the power lines are moving.

These tasks belong to a larger class of tasks which require **dynamic dexterity** and which are one of the key challenges in robotics. Dynamic manipulation, catching, juggling, walking, running, brachiating are all confronting this challenge. Despite the apparent ease with which humans accomplish them, they still represent a myriad of difficult research challenges. We must integrate advanced sensing, filtering, sensor fusion, planning, modelling, controlling, and under hard real time constraints, to succeed.

Dynamic grasping requires more accurate sensing information when the robot end effector approaches to the moving object. The sensor fusion becomes necessary to provide global sensing information of the moving object with respect to the robot base coordinate, as well as the local proximity sensing information with respect to the robot end effector. The electro-optical sensors are at present the most appropriate for short range proximity sensing in robotic application. We choose the infrared light

(IR) sensor based on amplitude measurement due to its small size, low cost, high bandwidth, ruggedness, and ease of use.

In this thesis, developing the path planning and control algorithm and applying to the complicated industrial robot arm to achieve dynamic grasping in 3D space is one of the major tasks. At the same time, local proximity sensing is also developed in order to optimise the grasping performance. In next section, I'd like to start from the review of the history work in this area.

1.2 Survey of Dynamic Grasping and IR Sensing

One of the first account of dynamic grasping in 2D was given by Andersson [2]. He described and implemented an approach where rolling ping pong balls on a slanted surface were caught. A straight line was fitted to the ball trajectory which was intersected with an "intersect line" to determine the catch point. This catch point was selected to be reachable by the robot in advance of the ball. Velocity or acceleration matching were not attempted.

Lin, Zeman and Patel [22] proposed a heuristic procedure which contains coarse tuning and fine tuning parts. A coarse tuning algorithm first drives the end effector into the neighborhood of the object in near-minimum time and then a fine tuning algorithm provides precise matching of the object's trajectory to ensure zero relative velocity and acceleration at the time of grasping. A quintic polynomial of time is constructed in joint space for each joint of the robot.

Houshangi [17] addresses explicitly the time delay introduced by image processing in visual feedback controlled robots by predicting the object trajectory via an autoregressive discrete-model. A planar adapted online to changes in the target position. Allen et al. [1] grasp a moving train with a PUMA under vision guidance based on straight line planning in Cartesian space between the robot's end effector and the target.

Hove and Slotine [18] implemented spatial robot catching based on a trajectory

matching algorithm that combines an observer with a varying-strength filter, an error estimator, and an initial motion algorithm. They applied this approach successfully to catching balls with the four degree-of-freedom WAM cable drive arm.

Gosselin's work on global inverse robot kinematics [12, 13] includes a path planning scheme to catch moving objects: the last object state measurements are employed to predict its path as a parabola, assuming free falling object. The closest point along this trajectory to the robot's base is then selected as the catching point. Quintic polynomials of time are then planned in joint space between the current and catch configuration.

The most recent version of the classical sense-plan-replan paradigm is proposed by Buttazo, Allota and Fanizza [10], where an object moving in the plane is caught based on a predicted path assuming constant acceleration. The problem of blending trajectory segments in real time subject to sensory inputs and tracking constraints is solved by Lloyd and Hayward [23].

The use of electro-optical proximity sensing has a long history, beginning with the work of Ernst at MIT [14] who used them as inputs to his event based programming language. Johnston at PL [19] carried out a mapping experiment where the proximity sensor was used to follow the profile of a surface and to guide an end effector into position for grasping. Bejczy [5] incorporated the IR sensors with a vision system to provide acoustic feedback in telerobotic operations. In [11] IR sensors aid telerobotic grasping when the manipulator is within the region of the object. Balek and Kelley used gripper mounted proximity sensors for robot control [4] via a hierarchical control scheme. In [24], Marszalec describes optical *fibres* based proximity sensor characteristics and their incorporation on a robot gripper. A fibre optic proximity sensor that measures object-sensor distance via a simple inverse square relationship of the magnitude of the received signal is discussed by Li [21]. Masuda [25] uses the phase shift of the received signal to measure distance, angle or orientation depending on the mode of operation. The sensor consists of six LED's in a cross shaped pattern with the

photo transistor in the center. Goldenberg et al. [26] show how this sensor's design parameters affect its performance and propose an optimal sensor design.

1.3 Contributions

In this thesis, "Geometric Controller," a new approach to online trajectory planning and dynamic grasping is presented. It simply evaluates a nonlinear function which maps the current measured position of the object to be grasped into a current desired robot pose. If the robot tracks these set points, it is guaranteed to match the object's velocity and acceleration on a specified grasp surface. Since it does not depend on future object measurements, no object model is needed for trajectory prediction. Without trajectory prediction, the computational effort is drastically reduced, allowing for higher controller speed and tracking feedback gains. At the same time this approach, reported in [28], provides a framework for general sensor-based autonomous control of robotics tasks.

The sensor-based autonomous control of dynamic grasping via geometric controller has been successfully implemented on the Sarcos Dextrous Arm, an industrial robot arm with seven joints and a three-fingered hand. The object trajectory is measured via an infrared OPTOTRAK motion tracking system. In order to track the set points in Cartesian space, forward and inverse kinematics (7 dof) on Sarcos Dextrous Arm have been fulfilled. The redundancy provided multi-solutions in joint space for specific Cartesian space pose. The switches between different solution branches may cause jumps of the robot arm, thus we keep the shoulder down and elbow up selection and fix the ψ which is defined as the forearm-upper arm plane rotation angle. Then the on-line sensor-based tracking became the task where calibration issue arose, where the joint angle offsets, gain and offset of position sensor for each joint, position offset and orientation matrix from the robot base coordinate with respect to the OPTOTRAK sensory coordinate, etc need to be calibrated. To deal with it, the open-loop kinematics calibration method is examined. Then we achieve the on-line tracking and with the

tracking errors need to be eliminated. Feedback controller gains are optimized, as well as the gravity compensation is discussed. The software module for implementing the geometric controller on the Sarcos Dextrous arm to grasp moving object in 3D space and interface with OPTOTRAK system are created in a C40 environment, which fulfills various tasks. The dynamic grasping have been successfully implemented with experimental results analysis. Furthermore, in order to optimize the grasping performance, model based local infrared proximity sensing, IR sensor model and calibration have been developed to provide more local information of object position with respect to the end effector. The object location strategy is also presented with experimental results, reported in [29], in the context of dynamic grasping.

In this thesis, details of the background, derivation and simulation of the Geometric Controller are given in Chapter 2. Real time implementation of the algorithm on Sarcos Dextrous Arm is described in Chapter 3, where kinematics, calibration, tracking performance optimization and result discussions are discussed. In Chapter 4 grasping performance optimization by using proximity sensing is discussed. Chapter 5 contains conclusions that can be made about the success and efficiency of the geometric controller algorithm and IR proximity sensing, as well as future work.

Chapter 2

Geometric Controller

In this chapter, a controller strategy is presented for dynamic grasping and trajectory generating. Based on proper task-oriented sensing, the “Geometric Controller” translates relative Cartesian position and velocity information between robot and object into an online reference trajectory, which, if tracked, will result in a smooth contact with few assumptions on the object trajectory. In the context of dynamic grasping, “proper task-oriented sensing” means, for example, an increasing sensing accuracy and rate as the object approaches the robot end effector. The relevant IR sensing technology and object pose estimations will be presented in Chapter 4.

2.1 Problem Formulation

Let

$$\mathbf{r}_d = \begin{bmatrix} r_{d,x} \\ r_{d,y} \\ r_{d,z} \end{bmatrix}, \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \in \mathbb{R}^3$$

represent the desired and actual robot end effector position and the actual object position, respectively. The end point position error vector is

$$\mathbf{e}_b = \mathbf{r}_d - \mathbf{b}$$

and the tracking error is

$$\mathbf{e}_r = \mathbf{r}_d - \mathbf{r}.$$

We translate the temporal specification for making a grasp contact into a geometric one by assuming that the contact time t_c is the time when the object traverses the horizontal “grasp surface”

$$b_z(t_c) = 0. \quad (2.1)$$

Depending on the task and the robot’s workspace, other surfaces can be selected, for example, spherical surface.

A **smooth grasp** of a moving object is defined as a match between the position, velocity and acceleration of the object and the robot’s end effector on the grasp surface, i.e.

Condition 1:

$$\mathbf{e}_b(t_c) = \dot{\mathbf{e}}_b(t_c) = \ddot{\mathbf{e}}_b(t_c) = \mathbf{0}. \quad (2.2)$$

In addition, in order to avoid pre-grasp collisions between robot and object, we require that

Condition 2:

$$r_{d,z}(t) < b_z(t) \quad \forall t < t_c. \quad (2.3)$$

The tracking problem is not directly addressed here. To eliminate the tracking error is one of the major task during the implementation and will be discussed in detail in Chapter 3. However, in order to facilitate tracking, the initial desired and actual robot positions, velocities and accelerations at t_0 should match,

Condition 3:

$$\mathbf{e}_r(t_0) = \dot{\mathbf{e}}_r(t_0) = \ddot{\mathbf{e}}_r(t_0) = \mathbf{0}. \quad (2.4)$$

Since we are focussing on the online trajectory generation problem here, we also make the following assumptions:

1. The inverse kinematics, possible singularities and interference with the workspace boundaries and other obstacles are dealt with in Chapter 3. There are several

solutions available, for example [15].

2. An integrated sensing and processing system provides the object's Cartesian position and velocity relative to the robot's end effector will be presented in Chapter 3 and 4. A system based on real time stereo vision has been successfully implemented for robot juggling ([27]).
3. The robot end effector can be closed when the smooth grasp conditions are satisfied.

2.2 Traditional Replanning Approach

There has been much study devoted to the trajectory generation for dynamic tracking and grasping. Traditionally a quintic polynomial of time is constructed which satisfies initial and final position, velocity and acceleration conditions in task or usually joint space in order to achieve smooth grasp. In case the object's trajectory is known exactly à priori, its intersection with the desired contact surface, at the time of contact, t_c , can be computed.

The problem with this approach is that in practice, the exact future object trajectory is never available accurately enough for smooth grasping purposes. Therefore the robot trajectory has to be continuously replanned based on the latest sensory information. This in effect leads to a highly inefficient implementation, since only the initial small portion of the planned trajectory is used. This problem is exacerbated if no information about the object's dynamics is available to the robot trajectory planner. In this case typically a straight line (constant velocity) or parabola (constant acceleration) is assumed. That is, the traditional replanning approach requires more computation time and information about the object dynamics. Compared to it, the new geometric controller shows its outstanding advantages, which will be presented in the next section.

2.3 Background of Geometric Controller: Robot Juggling

The basic ideas for simple geometric feedback controllers came from a seeming complex task domain - robot juggling. The motivation for this thesis originates from a successful prototype described in [8] where it was applied to juggling one and two balls in the plane [7]. This concept was extended further by Rizzi [27] to spatial juggling of two balls with a three degree-of-freedom direct drive robot, guided by a real time stereo vision system.

It turns out that a simplified version of the geometric controller for juggling can accommodate a velocity and acceleration matching contact with a moving object at a desired position. To define the contact position instead of the contact time is a major difference between the geometric controller and traditional replanning approach. It presents a new concept for dynamic grasping provided the moving object approaches and traverses the grasp surface no matter what kind of trajectory, velocity or acceleration it will have. The grasp surface has practical values in the industrial fields where dynamic grasping tasks are required. Under certain circumstance, where to grasp the workpiece is more considered than when to grasp it.

To illustrate the basic idea, consider the trivial 1 dof prismatic robot in Fig. 2.1 with vertical coordinate r and an object moving along a line, with height b . A geometric controller for the desired robot height r_d to smoothly grasp the object would be

$$r_d = \kappa \cdot \text{atan}(b/\kappa), \quad (2.5)$$

where $\kappa > 0$ is a user selectable gain.

For purposes of improved tracking, the desired robot velocity is available to the PD controller as well:

$$\dot{r}_d = \kappa \frac{\dot{b}/\kappa}{1 + (b/\kappa)^2}. \quad (2.6)$$

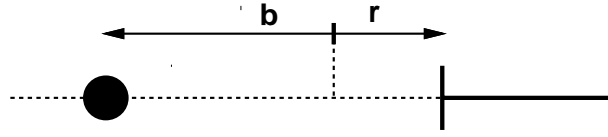


Figure 2.1: 1 dof illustration

Assuming perfect tracking, $r(t) = r_d(t)$ robot-object contacts can only occur when $r = b = 0$, and at that point a velocity matching contact is guaranteed,

$$\dot{r}_c = \dot{r}(r = b = 0) = \kappa \frac{\dot{b}/\kappa}{1 + (b/\kappa)^2} = \dot{b}_c.$$

The object's acceleration is matched as well:

$$\ddot{r}_c = \ddot{r}(r = b = 0) = \frac{\ddot{b}}{1 + (b/\kappa)^2} - \frac{2(b/\kappa^2)\dot{b}^2}{(1 + (b/\kappa)^2)^2} = \ddot{b}_c.$$

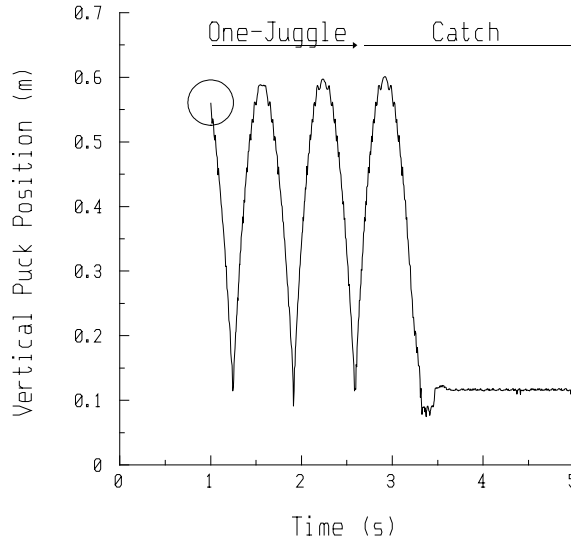


Figure 2.2: Juggle followed by a catch (1), from [4].

A successful experimental implementation of this geometric catch controller, taken from [20] is shown in Fig. 2.2 where it is preceded by a vertical one-juggle, controlled by a similar algorithm.

The reason why the algorithm is called geometric controller since it contains no dynamics itself, it is simply a map from current object phase space to a desired

current robot state. All the planning to ensure smooth grasping is done off line, when designing the controller. Notice that this approach completely eliminates the online trajectory planning in the traditional sense. The task is encoded in (2.5) and is achieved by tracking the set-points computed by the same equation at every instant. No assumptions were made about the object's trajectory, and the algorithm will succeed as long as tracking is achieved. In the next section we will generalize this basic controller (2.5) for dynamic grasping in six degrees-of-freedom.

2.4 Development of Geometric Controller

There are two major shortcomings in the simple geometric catch controller (2.5). First, it needs to be extended for spatial grasping. This is non-trivial, since a straightforward addition of similar controllers for the additional degrees of freedom does not guarantee time synchronization such that all degrees of freedom satisfy the smooth grasp conditions at the same time. Second, at $t = 0$ there might be a large initial error \mathbf{e}_r between the commanded and the actual robot position, which leads to large transient torque requirements and might ultimately prevent a sufficiently small asymptotic tracking error at time of contact t_c , which would violate Condition 3 (2.4).

It turns out that there exist many functions that would result in a velocity and acceleration matching contact, in a similar fashion to (2.5), and a few can even satisfy the additional conditions (2.4) and (2.3). Below is the list of the functions we had examined where many of them have the same shortcomings as the basic atan function.

$$r_d = \kappa_1 (e^{\kappa_2 b} - e^{-\kappa_2 b}) \quad (2.7)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \kappa_1 (e^{\kappa_2 b} - 1) \quad (2.8)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \kappa_1 \frac{(e^{\kappa_2 b} - e^{-\kappa_2 b})}{(e^{\kappa_2 b} + e^{-\kappa_2 b})} \quad (2.9)$$

where $\kappa_1 \cdot \kappa_2 = 2$

$$r_d = \kappa_1 \ln(\kappa_2 b + 1) \quad (2.10)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \kappa_1 \ln \frac{1 - \kappa_2 b}{1 + \kappa_2 b} \quad (2.11)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \kappa_1 \ln(\sqrt{1 + (\kappa_2 b)^2} + \kappa_2 b) \quad (2.12)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \kappa_1 (\kappa_2^b - 1) \quad (2.13)$$

where $\kappa_1 \cdot \ln \kappa_2 = 1$

$$r_d = \kappa_1 \sin(\kappa_2 b) \quad (2.14)$$

where $\kappa_1 \cdot \kappa_2 = 1$

$$r_d = \frac{\kappa_1 b^2 + \kappa_2 b}{b + \kappa_3} \quad (2.15)$$

where $\frac{\kappa_2}{\kappa_3} = 1$

$$r_d = \frac{b}{\kappa_1 b^2 + 1} \quad (2.16)$$

where κ_1 is selectable gain.

However, the construction of polynomials is the simplest solution and allows a procedure equivalent to spline generation. In the sequel we will derive a spatial

geometric controller based on polynomials of the object position for the vertical and the horizontal components.

2.5 Theory of Geometric Controller

2.5.1 Vertical Components

A quintic function **of the vertical object position** satisfies all the grasping conditions:

$$r_{d,z}(b_z) = \sum_{n=0}^5 \kappa_{z,n} b_z^n = q_z(b_z). \quad (2.17)$$

The three gains $\kappa_{z,0}, \kappa_{z,1}, \kappa_{z,2}$ can be obtained from Condition 1 (2.2). For the vertical robot position at contact we obtain from (2.17)

$$r_{d,z}(t_c) = q_z(b_z(t_c)) = q_z(0) = \kappa_{z,0}$$

and to satisfy the grasping condition (2.2)

$$r_{d,z}(t_c) = b_z(t_c) = 0$$

requires $\kappa_{z,0} = 0$. Similarly, the velocity and acceleration conditions result in $\kappa_{z,1} = 1$ and $\kappa_{z,2} = 0$. The remaining three gains $\kappa_{z,3}, \kappa_{z,4}, \kappa_{z,5}$ are derived from the three initial conditions (2.4):

$$r_{d,z}(t_0) = q_z(b_{z,0}) = r_{z,0}$$

with

$$r_{d,z,0} = r_{d,z}(t_0), \quad r_{z,0} = r_z(t_0).$$

The velocity and acceleration initial conditions result in

$$\dot{q}_z(b_{z,0}) = \dot{r}_{z,0}, \quad \ddot{q}_z(b_{z,0}) = \ddot{r}_{z,0}.$$

The solution for the last three gains can be expressed as

$$\begin{bmatrix} \kappa_{z,3} \\ \kappa_{z,4} \\ \kappa_{z,5} \end{bmatrix} = \mathbf{B}_{z,0}^{-1} \cdot \begin{bmatrix} r_{z,0} - b_{z,0} \\ \dot{r}_{z,0} - \dot{b}_{z,0} \\ \ddot{r}_{z,0} - \ddot{b}_{z,0} \end{bmatrix}$$

where

$$\mathbf{B}_{z,0} = \begin{bmatrix} b_{z,0}^3 & b_{z,0}^4 & b_{z,0}^5 \\ 3b_{z,0}^2 \dot{b}_{z,0} & 4b_{z,0}^3 \dot{b}_{z,0} & 5b_{z,0}^4 \dot{b}_{z,0} \\ 6b_{z,0} \dot{b}_{z,0}^2 + 3b_{z,0}^2 \ddot{b}_{z,0} & 12b_{z,0}^2 \dot{b}_{z,0}^2 + 4b_{z,0}^3 \ddot{b}_{z,0} & 20b_{z,0}^3 \dot{b}_{z,0}^2 + 5b_{z,0}^4 \ddot{b}_{z,0} \end{bmatrix}. \quad (2.18)$$

Subscript 0 refers to the initial condition. We have now obtained the desired robot trajectory which satisfies the grasping conditions (2.2) and (2.4). Notice that this computation is performed only once at $t = 0$, and subsequently only the evaluation of the polynomial (2.17) is required (in addition to its derivative for tracking purposes).

If the initial object velocities and accelerations are not available, a third order polynomial is sufficient to match the initial position only, as well as match the position, velocity and acceleration at contact time. In this case, $\kappa_{z,3} = (r_{z,0} - b_{z,0})/b_{z,0}^3$ and the geometric controller for the vertical component takes on the extremely simple form

$$r_{d,z} = b_z + \kappa_{z,3} b_z^3. \quad (2.19)$$

Note that Condition 2 (2.3) is satisfied by (2.19) as long as it is satisfied at the initial time,

$$\kappa_{z,3} = (r_{z,0} - b_{z,0})/b_{z,0}^3 < 0.$$

Note in this case, the pre-grasp collision is avoided as long as it is satisfied at the initial time for $\kappa_{z,3}$.

2.5.2 Horizontal components

In this section, the algorithm for online generation of a robot reference trajectory is developed which achieves velocity and acceleration matching for the r_x and r_y components. The critical task here is that this matching must be accomplished at the object's spatial position and **at the same time** t_c when the vertical component is contacting the object. This is an illustrative example how purely geometric controllers are used to achieve time synchronization. Time synchronization between two processes has been achieved once before with purely geometric controllers, however in an entirely

different fashion. In [9] it was critical for juggling two balls which required the control of a certain phase relationship between them. For brevity, only the r_x component is developed. The algorithm is identical for the r_y component.

In order to satisfy the grasping conditions at t_c and the initial condition at t_0 in the x coordinate, a quintic polynomial of the **combined** ball positions in x and z coordinates must be used.

$$r_x = b_x + \sum_{n=3}^5 \kappa_{x,n} b_z^n \quad (2.20)$$

satisfies all the requirements, since on the contact surface $b_{z,c} = 0$, and thus

$$\dot{r}_{x,c} = \dot{b}_{x,c} + \dot{b}_{z,c} \sum_{n=3}^5 n \kappa_{x,n} b_{z,c}^{n-1} = \dot{b}_{x,c} \quad (2.21)$$

$$\begin{aligned} \ddot{r}_{x,c} &= \ddot{b}_{x,c} + \dot{b}_{z,c}^2 \sum_{n=3}^5 n(n-1) \kappa_{x,n} b_{z,c}^{n-2} + \ddot{b}_{z,c} \sum_{n=3}^5 n \kappa_{x,n} b_{z,c}^{n-1} \\ &= \ddot{b}_{x,c}. \end{aligned} \quad (2.22)$$

The solution for the last three gains can be expressed from (2.20), (2.21), (2.22) and Condition 3 (2.4) as

$$\begin{bmatrix} \kappa_{x,3} \\ \kappa_{x,4} \\ \kappa_{x,5} \end{bmatrix} = \mathbf{B}_{z,0}^{-1} \cdot \begin{bmatrix} r_{x,0} - b_{x,0} \\ \dot{r}_{x,0} - \dot{b}_{x,0} \\ \ddot{r}_{x,0} - \ddot{b}_{x,0} \end{bmatrix}$$

with $\mathbf{B}_{z,0}$ as defined in (2.18).

Again, if only initial position matching is required, we can use the third order polynomial version of (2.20). The constant $\kappa_{x,3}$ is determined as above, $\kappa_{x,3} = (r_{x,0} - b_{x,0})/b_{z,0}^3$ and

$$r_{d,x} = b_x + \kappa_{x,3} b_z^3. \quad (2.23)$$

Notice that there is no interference problem since the vertical controller assures Condition 2, namely that the robot's end effector is always below the object. So there are no spatial restrictions on the horizontal components.

2.5.3 Spherical coordinates

For common non-Cartesian manipulators, it would be advantageous to express the desired robot end point and object position in spherical coordinates.

The position of the end effector and the object are now defined with respect to the spherical coordinate (r, ϕ, θ) . The algorithms are issued as below:

$$r_{d,r}(b_r) = \sum_{n=0}^5 \kappa_{r,n} b_r^n = q_r(b_r). \quad (2.24)$$

where the gains are calculated from the grasping conditions (2.2) and initial conditions (2.4), $\kappa_{r,0} = 0$, $\kappa_{r,1} = 1$ and $\kappa_{r,2} = 0$, as well as

$$\begin{bmatrix} \kappa_{r,3} \\ \kappa_{r,4} \\ \kappa_{r,5} \end{bmatrix} = \mathbf{B}_{r,0}^{-1} \cdot \begin{bmatrix} r_{r,0} - b_{r,0} \\ \dot{r}_{r,0} - \dot{b}_{r,0} \\ \ddot{r}_{r,0} - \ddot{b}_{r,0} \end{bmatrix}$$

where

$$\mathbf{B}_{r,0} = \begin{bmatrix} b_{r,0}^3 & b_{r,0}^4 & b_{r,0}^5 \\ 3b_{r,0}^2 \dot{b}_{r,0} & 4b_{r,0}^3 \dot{b}_{r,0} & 5b_{r,0}^4 \dot{b}_{r,0} \\ 6b_{r,0} \dot{b}_{r,0}^2 + 3b_{r,0}^2 \ddot{b}_{r,0} & 12b_{r,0}^2 \dot{b}_{r,0}^2 + 4b_{r,0}^3 \ddot{b}_{r,0} & 20b_{r,0}^3 \dot{b}_{r,0}^2 + 5b_{r,0}^4 \ddot{b}_{r,0} \end{bmatrix}. \quad (2.25)$$

If only the initial position matching is required, we use the third order polynomial version of (2.24) for r_r , r_ϕ , r_θ , and r_χ .

$$r_r = b_r + \kappa_{r,3} b_r^3 \quad (2.26)$$

where $\kappa_{r,3}$ is determined by the initial condition

$$\kappa_{r,3} = \frac{r_{r,0} - b_{r,0}}{b_{r,0}^3} \quad (2.27)$$

$$r_\phi = b_\phi + \kappa_{\phi,3} b_r^3 \quad (2.28)$$

where $\kappa_{\phi,3}$ is determined by the initial condition

$$\kappa_{\phi,3} = \frac{r_{\phi,0} - b_{\phi,0}}{b_{r,0}^3} \quad (2.29)$$

$$r_\theta = b_\theta + \kappa_{\theta,3} b_r^3 \quad (2.30)$$

where $\kappa_{\theta,3}$ is determined by the initial condition

$$\kappa_{\theta,3} = \frac{r_{\theta,0} - b_{\theta,0}}{b_{r,0}^3} \quad (2.31)$$

2.5.4 Orientation

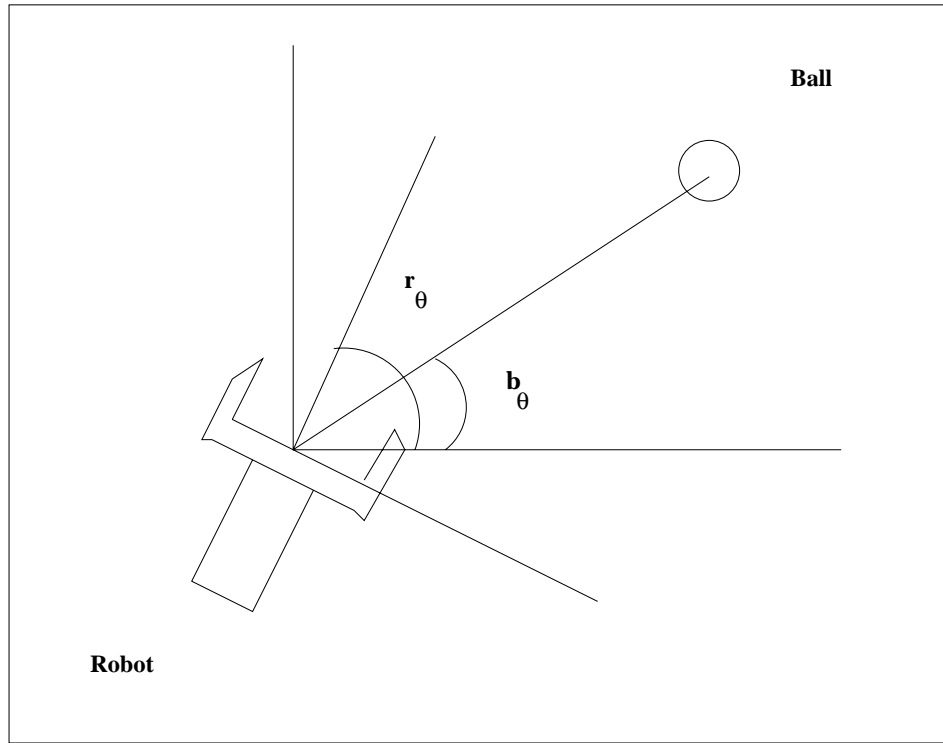


Figure 2.3: End Effector Orientation Control Variables

For any practical implementation, the end effector orientation, angular velocity and acceleration should be matched to the object's at time of contact as well. In addition, the initial desired end effector orientation should match the actual orientation. In the Cartesian case this is implemented in the geometric controller framework by specifying the Euler angles of end effector $(r_\phi, r_\theta, r_\chi)$ with respect to the base coordinates as a polynomial of the object's Euler angles with respect to the end effector frame (Fig. 2.3)

$(b_\phi, b_\theta, b_\chi)$ expressed in the base coordinates and the vertical object component b_z

$$r_i = b_i + \kappa_{i,3} b_z^3; \quad \kappa_{i,3} = \frac{r_{i,0} - b_{i,0}}{b_{z,0}^3} \quad i = \phi, \theta, \chi. \quad (2.32)$$

At contact surface, $b_z = 0$,

$$r_i(b_z = 0) = b_i(b_z = 0) \quad (2.33)$$

$$\dot{r}_i(b_z = 0) = \dot{b}_i(b_z = 0) + 3\kappa_{i,3} b_z^2 \dot{b}_z = \dot{b}_i(b_z = 0) \quad (2.34)$$

$$\ddot{r}_i(b_z = 0) = \ddot{b}_i(b_z = 0) + 3\kappa_{i,3} b_z^2 \ddot{b}_z + 6\kappa_{i,3} b_z \dot{b}_z^2 = \ddot{b}_i(b_z = 0) \quad (2.35)$$

For illustration Fig. 2.4 shows the planar case in the vertical x-z-plane. Corresponding robot-object trajectory points are connected via a dashed line, and the robot's end effector orientation is indicated by a short solid line. Notice how the initially large orientation error is eliminated by (2.32) to match the object's angle, angular velocity and angular acceleration with respect to the robot's end effector.

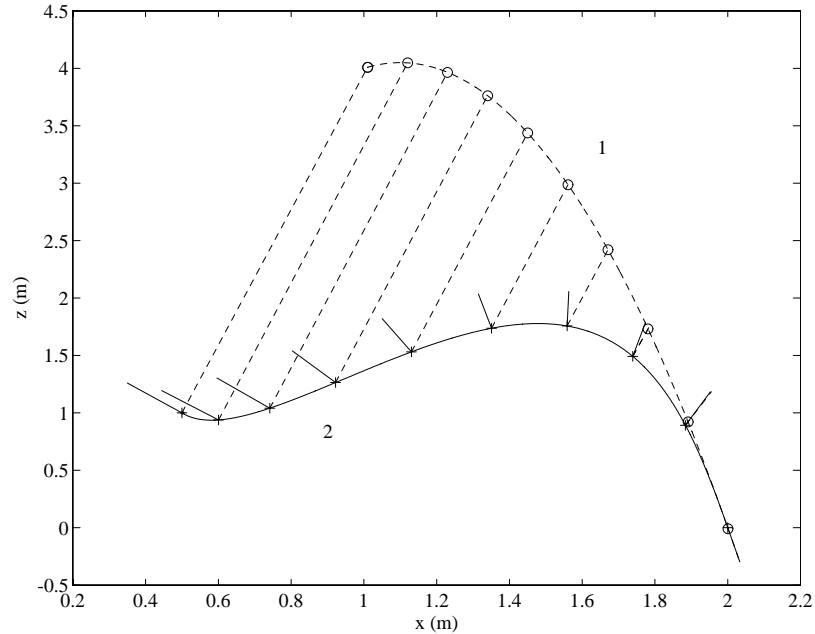


Figure 2.4: Orientation Control for Parabolic Object Trajectory, 1: object, 2: robot

2.6 Dynamic simulation and comparisons

In this section we compare the performance of the geometric control approach to smooth grasping as described above to the more conventional approach based on trajectory planning using intercept predictions based on current and past state measurements. Both versions, which assume either constant velocity or acceleration for estimating the future object trajectory are evaluated. For ease of exposition we limit ourselves to the planar problem, and endpoint position only. The results for the full spatial case with orientation would be analogous. Therefore the geometric control is completely described by equations (2.19) and (2.23). The results are shown in Fig. 2.5 and Fig. 2.6 for a parabolic, low and high frequency sinusoidal object trajectory. In

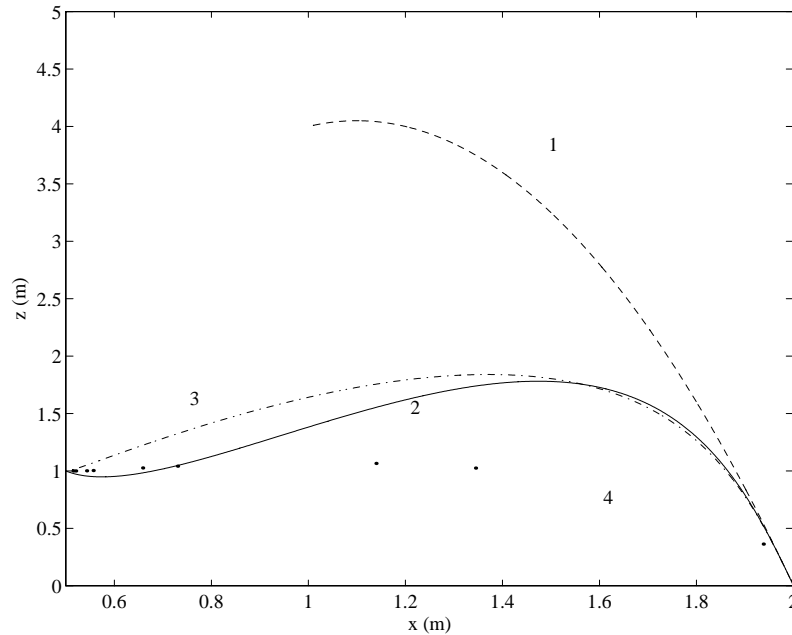


Figure 2.5: Parabolic Object Trajectories (1: object, 2: geometric controller, 3: replanning, const. acceleration, 4: replanning, const. velocity)

each plot the object trajectory is trajectory 1 (dashed line), the geometric controller is trajectory 2 (solid line), the replanning approach based on constant acceleration

object prediction is trajectory 3 (dashed dot) and the one on constant velocity object prediction is trajectory 4 (dot).

Overall, the robot trajectories generated by all three approaches are quite similar, despite the drastic difference between the replanning and the geometric controller. However, the geometric controller performs more closely to the better one of the replanning approaches.

For example, for the parabolic object trajectory (Fig. 2.5), the replanning approach assuming constant object acceleration (3) is predicting correctly. In this case the geometric controller (2) results in almost the same trajectory. In the case of the sinusoidal object trajectories (Fig. 2.6) and (Fig. 2.6), the replanning approach assuming constant object acceleration (3) predicts incorrectly, and the constant velocity (4) replanning approach results in a shorter, smoother trajectory. In this case, the geometric approach (2) results in a robot reference trajectory similar or better than the constant velocity approach.

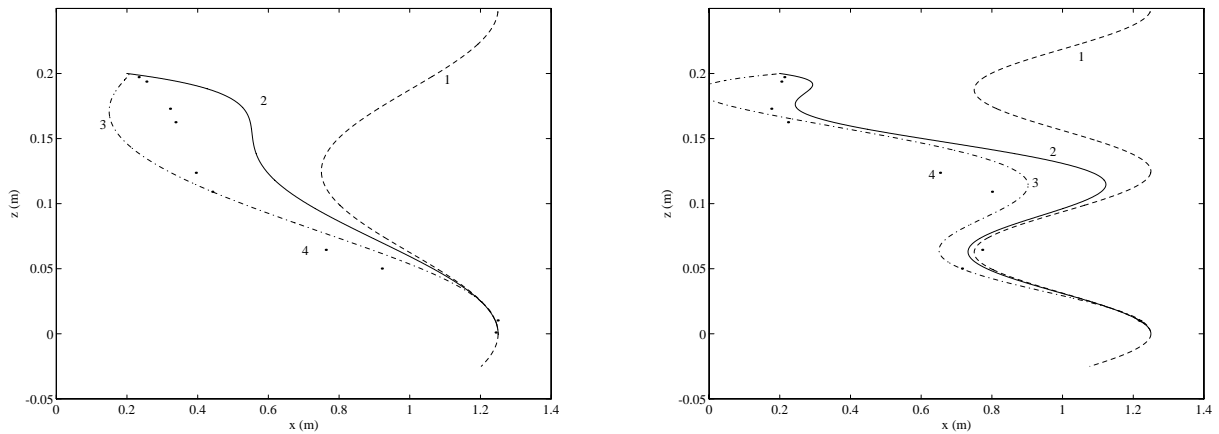


Figure 2.6: Sinusoidal Object Trajectories (1: object, 2: geometric controller, 3: replanning, const. acceleration, 4: replanning, const. velocity)

Chapter 3

Real-Time Implementation

The geometric controller generates the desired robot trajectory in Cartesian space for dynamic grasping. Real-time implementation involves the control in robot joint space in order to track the desired position and orientation, which leads to kinematics development. The special requirement arises from 7 degree of freedom Sarcos Dextrous Arm such as redundancy decoupling, solution analysis and selection. Then we deal with calibration problem, where the **open loop kinematic calibration** is issued in this chapter. We have successfully implemented the dynamic grasping based upon the previous works on kinematics and calibration. The dynamic grasping results analysis is presented in this chapter. In order to improve the tracking performance, servo controller gains have been optimized as well as gravity compensation are discussed.

3.1 Kinematics

3.1.1 Forward kinematics

The kinematical problem includes the **forward** and **inverse** kinematics. The forward kinematics problem can be stated as follows: Given the joint variables of the robot, determine the position and orientation of the end effector. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link

extension in the case of prismatic or sliding joints. The forward kinematics problem is to be contrasted with the inverse kinematics problem, which is studied in next section, and which can be stated as follows: Given a desired position and orientation for the end effector of the robot, determine a set of joint variables that achieve the desired position and orientation.

For the purpose of kinematic analysis, one can think of a robot as a set of rigid links connected together at various joints. While it is possible to carry out all of the analysis using an arbitrary frame attached to each link, it is helpful to be systematic in the choice of these frames. A commonly used convention for selecting frames of reference in robotic applications is the Denavit Hartenberg, or DH convention. Consider Fig. 3.1, the rotation axis z_i corresponds to joint angle θ_i between link i and $i + 1$. The internal coordinate system for link i is completed by defining x_i from the cross product $z_{i-1} \times z_i$, which also locates the origin, and $y_i = z_i \times x_i$. Neighboring coordinate systems are related by three parameters:

1. a_i is the distance between z_{i-1} and z_i measured along x_i .
2. s_i is the distance between x_{i-1} and x_i measured along z_{i-1} .
3. α_i is the angle between z_{i-1} and z_i measured in a right-hand sense about x_i .

In this convention, each homogeneous transformation \mathbf{A}_i is represented as a product of four “basic” transformations

$$\mathbf{A}_i = \mathbf{Rot}_{z,\theta_i} \mathbf{Trans}_{z,d_i} \mathbf{Trans}_{x,a_i} \mathbf{Rot}_{x,\alpha_i} \quad (3.1)$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

where the four quantities $\theta_i, a_i, d_i, \alpha_i$ are parameters of link i and joint i .

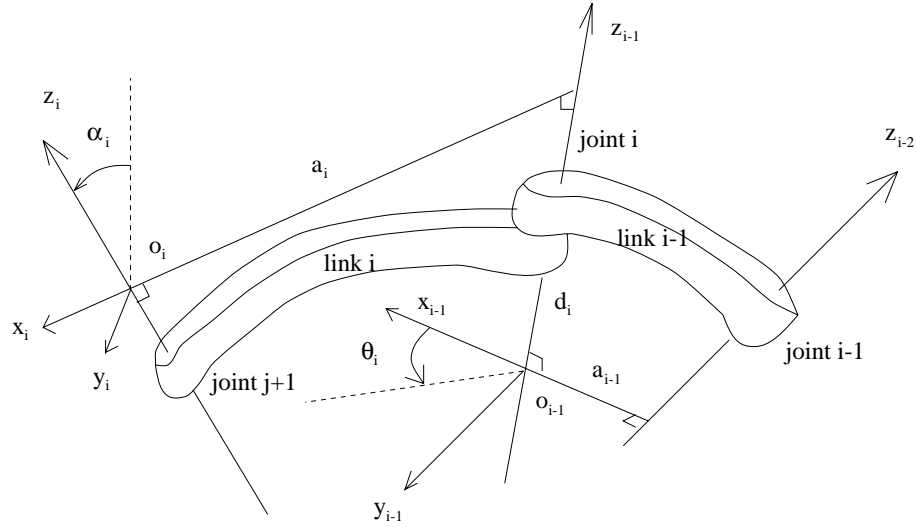


Figure 3.1: DH frame assignment

Each homogeneous transformation \mathbf{A}_i is of the form

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{R}_{i-1}^i & \mathbf{d}_{i-1}^i \\ \mathbf{0} & 1 \end{bmatrix} \quad (3.4)$$

Hence

$$\mathbf{T}_i^j = \mathbf{A}_{i+1} \cdots \mathbf{A}_j = \begin{bmatrix} \mathbf{R}_i^j & \mathbf{d}_i^j \\ \mathbf{0} & 1 \end{bmatrix} \quad (3.5)$$

The matrix \mathbf{R}_i^j express the orientation of frame j relative to frame i and is given by the rotational parts of the \mathbf{A} matrices as

$$\mathbf{R}_i^j = \mathbf{R}_i^{i+1} \cdots \mathbf{R}_{j-1}^j \quad (3.6)$$

The vectors \mathbf{d}_i^{j-1} are given recursively by the formula

$$\mathbf{d}_i^j = \mathbf{d}_i^{j-1} + \mathbf{R}_i^{j-1} \mathbf{d}_{j-1}^j \quad (3.7)$$

Denote the position and orientation of the end effector with respect to the inertial or the base frame by a 3×1 vector \mathbf{d}_0^n and a 3×3 rotation matrix \mathbf{R}_0^n , respectively, and show the structure of homogeneous matrix

$$\mathbf{T}_0^n = \begin{bmatrix} \mathbf{R}_0^n & \mathbf{d}_0^n \\ \mathbf{0} & 1 \end{bmatrix} \quad (3.8)$$

In principle, that is all there is to forward kinematics! Determine the functions $\mathbf{A}_i(\mathbf{q}_i)$, and multiply them together as needed. Now we develop the forward kinematics for the 7 dof Sarcos Dextrous Arm. Fig. 3.2 shows the structure of the Sarcos Dextrous Arm. The correspondent joint coordinates derived from Denavit-Hartenberg convention where the values for the three parameters for the arm a_i , s_i and α_i are given in the Table below:

Denavit-Hartenberg Parameters for Sarcos Arm

| Joint i | a_i (m) | s_i (m) | α_i (degree) |
|---------|-----------|-----------|---------------------|
| 1 | 0 | 0.1628 | 90 |
| 2 | 0 | 0 | -90 |
| 3 | 0 | 0.3467 | 90 |
| 4 | 0 | 0 | -90 |
| 5 | 0 | 0.32004 | 90 |
| 6 | 0 | 0 | -90 |
| 7 | 0 | 0 | 0 |

The directions for z_i and x_i are chosen so that when $\theta_i = 0$ the x_{i-1} and $x + i$ are parallel and pointing to the same direction. The arm has two spherical joints of the roll-pitch-roll configuration at shoulder and wrist, which is used to decouple the full degree of freedom in the inverse kinematics.

From equation(3.3) the link transformation matrices resulted from the coordinate

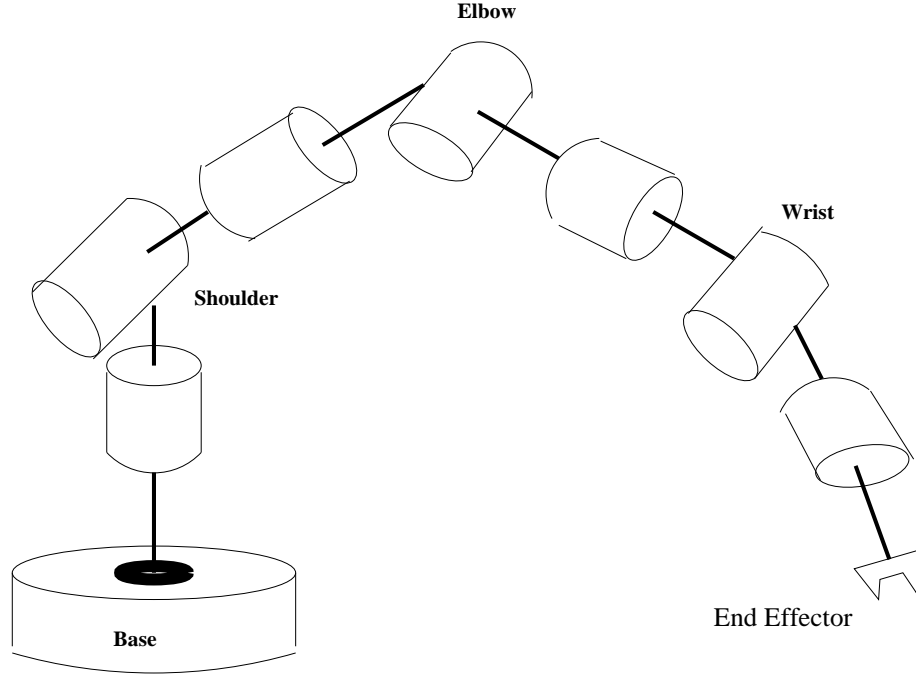


Figure 3.2: The Sarcos Dextrous Arm Geometry Structure

system shown are:

$$\mathbf{T}_1^0 = \begin{bmatrix} c_{\theta_1} & 0 & s_{\theta_1} & 0 \\ s_{\theta_1} & 0 & -c_{\theta_1} & 0 \\ 0 & 1 & 0 & 0.1628 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_1^0 & & 0 \\ & & & 0.1628 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

$$\mathbf{T}_2^1 = \begin{bmatrix} c_{\theta_2} & 0 & -s_{\theta_2} & 0 \\ s_{\theta_2} & 0 & c_{\theta_2} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_2^1 & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.10)$$

$$\mathbf{T}_3^2 = \begin{bmatrix} c_{\theta_3} & 0 & s_{\theta_3} & 0 \\ s_{\theta_3} & 0 & -c_{\theta_3} & 0 \\ 0 & 1 & 0 & 0.3467 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_3^2 & & 0 \\ & & & 0.3467 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

$$\mathbf{T}_4^3 = \begin{bmatrix} c_{\theta_4} & 0 & -s_{\theta_4} & 0 \\ s_{\theta_4} & 0 & c_{\theta_4} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_4^3 & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.12)$$

$$\mathbf{T}_5^4 = \begin{bmatrix} c_{\theta_5} & 0 & s_{\theta_5} & 0 \\ s_{\theta_5} & 0 & -c_{\theta_5} & 0 \\ 0 & 1 & 0 & 0.32004 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_5^4 & & 0 \\ & & & 0.32004 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.13)$$

$$\mathbf{T}_6^5 = \begin{bmatrix} c_{\theta_6} & 0 & -s_{\theta_6} & 0 \\ s_{\theta_6} & 0 & c_{\theta_6} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_6^5 & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.14)$$

$$\mathbf{T}_7^6 = \begin{bmatrix} c_{\theta_7} & 0 & -s_{\theta_7} & 0 \\ s_{\theta_7} & 0 & c_{\theta_7} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_7^6 & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.15)$$

where the abbreviations $s_\theta = \sin\theta$ and $c_\theta = \cos\theta$.

The end effector position vector \mathbf{p} and rotation matrix with respect to base coordinate \mathbf{R} are calculated from the multiplications of the homogeneous transformations \mathbf{T}_1^0 to \mathbf{T}_7^6 :

$$\mathbf{T}_7^0 = \mathbf{T}_1^0 \mathbf{T}_2^1 \cdots \mathbf{T}_7^6 \quad (3.16)$$

$$= \begin{bmatrix} & & & | & \\ & \mathbf{R} & & | & \mathbf{p} \\ & & & | & \\ - & - & - & | & - \\ 0 & 0 & 0 & | & 1 \end{bmatrix} \quad (3.17)$$

3.1.2 Inverse kinematics

Given end effector position and orientation (full 6 dof), joint angles are searched by inverse kinematics. Due to singularities, limited ranges of motion of joints, obstacles in workspace and torque limit, a general-purpose manipulator is needed to have more than 6 degree of freedom. A number of control schemes for determining joint trajectories for redundant manipulator have been suggested. For the zero-offset anthropomorphic manipulator – Sarcos Dextrous Arm, an optimum kinematics design for a 7 dof manipulator is given by Hollerbach [3]. From Fig. 3.2 all of axes line up

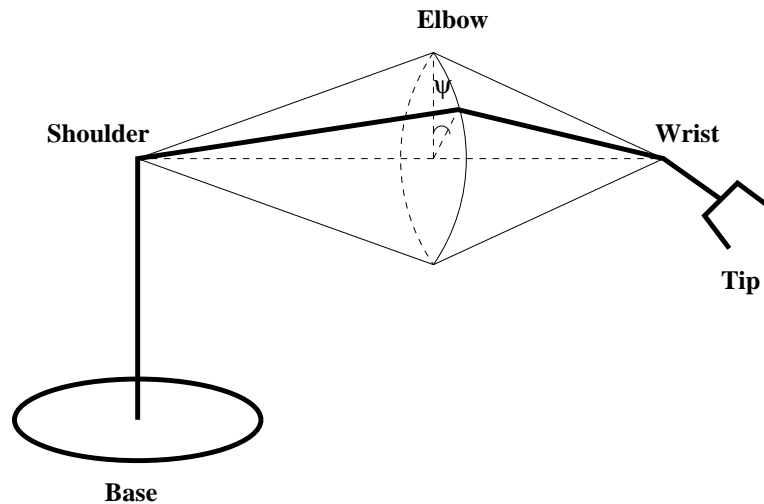


Figure 3.3: The redundancy corresponds to rotating the elbow about the shoulder/wrist line

such that joints 0, 1 and 2 intersect at the shoulder triple point; joints 2, 3 and 4 intersect at the elbow triple point; and joints 4, 5 and 6 intersect at the wrist triple point. The only length parameters required to parameterize the arm are the distances from shoulder to elbow s_3 and that from elbow to wrist s_5 .

Assuming that the redundancy is resolved by specifying the forearm-upper arm plane rotating ψ angle about an imaginary line from shoulder triple point to the wrist position (Fig. 3.3), the inverse kinematics solution of the arm can be derived as the following steps:

onto the \mathbf{x}_0 and \mathbf{y}_0 plane (Fig. 3.4).

$$\theta'_1 = \text{atan2}(p_{wy}, p_{wx}) \quad (3.19)$$

θ_4 is calculated by the cosine rule (Fig. 3.4) ,

$$p_w^2 = s_3^2 + s_5^2 - 2 s_3 s_5 \cos(\pi - \theta_4) \quad (3.20)$$

$$\cos\theta_4 = \frac{p_w^2 - s_3^2 - s_5^2}{2 s_3 s_5} \quad (3.21)$$

$$\theta_4 = \pm \cos^{-1}\left(\frac{p_w^2 - s_3^2 - s_5^2}{2 s_3 s_5}\right) \quad (3.22)$$

Depending which quadrant is chosen for θ_4 , we obtain an elbow-up or an elbow-down solution.

From the solution of θ'_1 , \mathbf{p}_w can be expressed in terms of link 1 coordinate as ${}^1\mathbf{p}'_w$:

$${}^1\mathbf{p}'_w = ({}^0\mathbf{R}'_1)^T {}^1\mathbf{p}_w = \begin{bmatrix} {}^1p'_{wx} \\ {}^1p'_{wy} \\ 0 \end{bmatrix} \quad (3.23)$$

At the same time, ${}^1\mathbf{p}'_w$ can be expressed in terms of joint axis vectors as

$${}^1\mathbf{p}'_w = s_3 \hat{\mathbf{z}}_4 + s_5 \hat{\mathbf{z}}_4 \quad (3.24)$$

$$= s_3 \begin{bmatrix} -\cos\theta'_2 \\ \sin\theta'_2 \\ 0 \end{bmatrix} + s_5 \begin{bmatrix} -\sin(\theta'_2 + \theta_4) \\ \cos(\theta'_2 + \theta_4) \\ 0 \end{bmatrix} \quad (3.25)$$

The θ'_2 is calculated from equations (3.23) and (3.25) as

$$\begin{bmatrix} {}^1p'_{wx} \\ {}^1p'_{wy} \end{bmatrix} = \begin{bmatrix} -(s_3 + s_5 \cos\theta_4) & -s_5 \sin\theta_4 \\ -s_5 \sin\theta_4 & s_3 + s_5 \cos\theta_4 \end{bmatrix} \begin{bmatrix} \sin\theta'_2 \\ \cos\theta'_2 \end{bmatrix} \quad (3.26)$$

Thus

$$\begin{bmatrix} \sin\theta'_2 \\ \cos\theta'_2 \end{bmatrix} = \begin{bmatrix} -(s_3 + s_5 \cos\theta_4) & -s_5 \sin\theta_4 \\ -s_5 \sin\theta_4 & s_3 + s_5 \cos\theta_4 \end{bmatrix}^{-1} \begin{bmatrix} {}^1p'_{wx} \\ {}^1p'_{wy} \end{bmatrix} \quad (3.27)$$

then $\theta'_2 = \text{atan2}(\sin \theta'_2, \cos \theta'_2)$

Step 2b: Find the elbow position by rotating the elbow point about \mathbf{p}_w by the angle ψ .

The elbow position corresponding to angle ψ is given by:

$$\mathbf{p}'_e = {}^0\mathbf{R}_1 {}^1\mathbf{R}_2 \begin{bmatrix} 0 \\ 0 \\ s_3 \end{bmatrix} = s_3 \begin{bmatrix} -\cos \theta'_1 \sin \theta'_2 \\ -\sin \theta'_1 \sin \theta'_2 \\ \cos \theta'_2 \end{bmatrix} \quad (3.28)$$

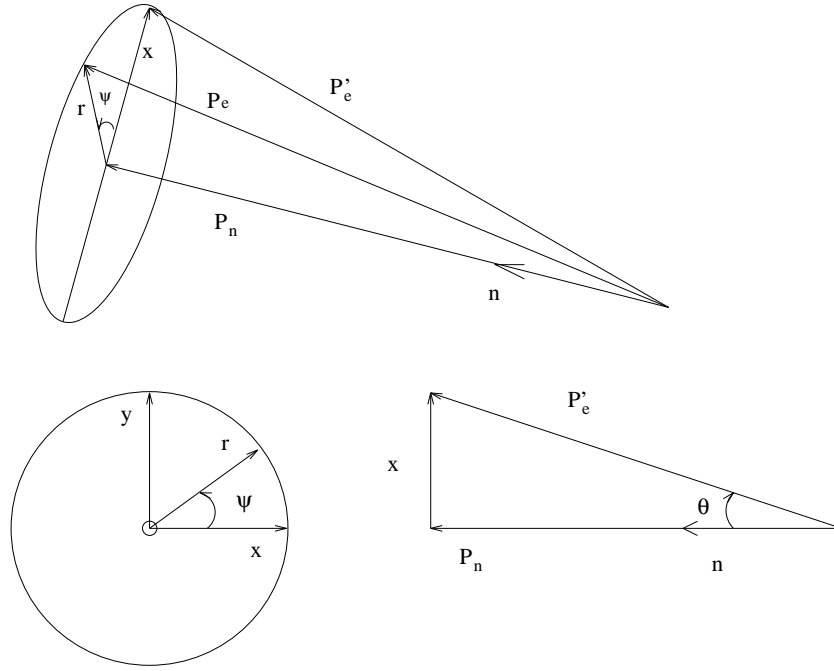


Figure 3.5: Elbow position correspond to rotational angle

Define that

$$\hat{\mathbf{n}} = \frac{\vec{\mathbf{p}}_w}{|\vec{\mathbf{p}}_w|}$$

the elbow position \mathbf{p}_e is generated by rotating \mathbf{p}'_e vector around the vector \mathbf{n} by angle ψ (Fig. 3.5).

$$\vec{\mathbf{p}}_e = \vec{\mathbf{p}}_n + \vec{\mathbf{r}} \quad (3.29)$$

where

$$\begin{aligned}
\vec{p}_n &= |\vec{p}'_e| \cos \theta \hat{n} = \langle \vec{p}'_e \cdot \hat{n} \rangle \hat{n} \\
\text{where } \langle \vec{p}'_e \cdot \hat{n} \rangle &= |\vec{p}'_e| |\hat{n}| \cos \theta = |\vec{p}'_e| \cos \theta \\
\vec{x} &= \vec{p}'_e - \vec{p}_n \\
\vec{y} &= \hat{n} \times \vec{x} \\
\vec{r} &= \vec{x} \cos \psi + \vec{y} \sin \psi
\end{aligned} \tag{3.30}$$

So

$$\vec{p}_e = \langle \vec{p}'_e \cdot \hat{n} \rangle \hat{n} + (\vec{p}'_e - \langle \vec{p}'_e \cdot \hat{n} \rangle \hat{n}) \cos \psi + \hat{n} \times (\vec{p}'_e - \langle \vec{p}'_e \cdot \hat{n} \rangle \hat{n}) \sin \psi \tag{3.31}$$

$$= \langle \vec{p}'_e \cdot \hat{n} \rangle \hat{n} (1 - \cos \psi) + \vec{p}'_e \cos \psi + \hat{n} \times \vec{p}'_e \sin \psi \tag{3.32}$$

Step 2c: Find the first two angles from the elbow position:

$$\mathbf{p}_e = {}^0\mathbf{R}_1 {}^1\mathbf{R}_2 \begin{bmatrix} 0 \\ 0 \\ s_3 \end{bmatrix} = s_3 \begin{bmatrix} -\cos \theta_1 \sin \theta_2 \\ -\sin \theta_1 \sin \theta_2 \\ \cos \theta_2 \end{bmatrix} = \begin{bmatrix} p_{ex} \\ p_{ey} \\ p_{ez} \end{bmatrix} \tag{3.33}$$

So that

$$\cos \theta_2 = \frac{p_{ez}}{s_3} \tag{3.34}$$

$$\theta_2 = \pm \arccos\left(\frac{p_{ez}}{s_3}\right) \tag{3.35}$$

while

$$\theta_1 = \text{atan2}\left(\frac{p_{ey}}{-s_3 \sin \theta_2}, \frac{p_{ex}}{-s_3 \sin \theta_2}\right) \tag{3.36}$$

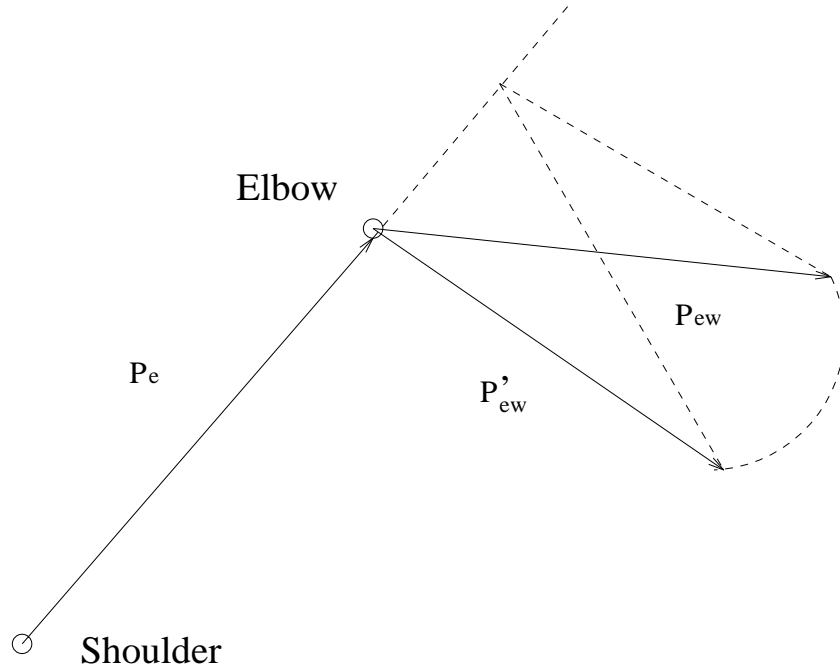
Step 2d: Find the roll angle θ_3 .

Consider that the wrist position that would be located if the arm was placed by the (unprimed) angles θ_1 , θ_2 , and θ_4 , with $\theta_3 = 0$ (Fig. 3.6). The vector from the elbow point to the wrist point \mathbf{p}'_{ew} is given by

$$\mathbf{p}'_{ew} = {}^0\mathbf{R}_4 \begin{bmatrix} 0 \\ 0 \\ s_5 \end{bmatrix}$$

The actual forearm vector is given by $\mathbf{p}_{ew} = \mathbf{p}_w - \mathbf{p}_e$. The vector normal to the $\mathbf{p}_e - \mathbf{p}_{ew}$ plane is calculated as $\mathbf{n} = \mathbf{p}_e \times \mathbf{p}_{ew}$; the vector normal to $\mathbf{p}_e - \mathbf{p}'_{ew}$ plane is calculated as $\mathbf{n}' = \mathbf{p}_e \times \mathbf{p}'_{ew}$. The θ_3 is then given by

$$\theta_3 = \text{atan2}\left(\frac{\mathbf{p}_e \times \mathbf{n}'}{|\mathbf{p}_e|}, \mathbf{n}' \cdot \mathbf{n}\right)$$

Figure 3.6: Find θ_3

Step 3: Find the hand orientation relative to the forearm.

$${}^4\mathbf{R}_7 = ({}^0\mathbf{R}_4)^T {}^0\mathbf{R}_7 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3.37)$$

So that ${}^4\mathbf{R}_7$ is calculated to be used in next step for finding θ_5 , θ_6 and θ_7 .

Step 4: Find the last three joint angles.

The last three angles θ_5 , θ_6 and θ_7 can be found by some matrix manipulations:

$${}^4\mathbf{R}_5 {}^5\mathbf{R}_6 = \begin{bmatrix} \cos \theta_5 & 0 & \sin \theta_5 \\ \sin \theta_5 & 0 & -\cos \theta_5 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos \theta_6 & 0 & -\sin \theta_6 \\ \sin \theta_6 & 0 & \cos \theta_6 \\ 0 & -1 & 0 \end{bmatrix} \quad (3.38)$$

$$= \begin{bmatrix} \cos \theta_5 \cos \theta_6 & -\sin \theta_5 & -\cos \theta_5 \sin \theta_6 \\ \sin \theta_5 \cos \theta_6 & \cos \theta_5 & -\sin \theta_5 \sin \theta_6 \\ \sin \theta_6 & 0 & \cos \theta_6 \end{bmatrix} \quad (3.39)$$

Since ${}^4\mathbf{R}_5 {}^5\mathbf{R}_6 {}^6\mathbf{R}_7 = {}^4\mathbf{R}_7 \implies {}^4\mathbf{R}_5 {}^5\mathbf{R}_6 = {}^4\mathbf{R}_7 {}^6\mathbf{R}_7^T$, and

$${}^4\mathbf{R}_7 {}^6\mathbf{R}_7^T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot \begin{bmatrix} \cos \theta_7 & \sin \theta_7 & 0 \\ -\sin \theta_7 & \cos \theta_7 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.40)$$

$$= \begin{bmatrix} r_{11} \cos \theta_7 - r_{12} \sin \theta_7 & r_{11} \sin \theta_7 + r_{12} \cos \theta_7 & r_{13} \\ r_{21} \cos \theta_7 - r_{22} \sin \theta_7 & r_{21} \sin \theta_7 + r_{22} \cos \theta_7 & r_{23} \\ r_{31} \cos \theta_7 - r_{32} \sin \theta_7 & r_{31} \sin \theta_7 + r_{32} \cos \theta_7 & r_{33} \end{bmatrix} \quad (3.41)$$

Compare the elements 33, 13 and 23 in (3.39) and (3.41), we have solved θ_6 and θ_5 , respectively

$$\theta_6 = \pm \arccos(r_{33}) \quad (3.42)$$

$$\theta_5 = \operatorname{atan2}\left(\frac{r_{23}}{-\sin \theta_6}, \frac{r_{13}}{-\sin \theta_6}\right) \quad (3.43)$$

Also from elements 12 and 22,

$$\begin{cases} -\sin \theta_5 = r_{11} \sin \theta_7 + r_{12} \cos \theta_7 \\ \cos \theta_5 = r_{21} \sin \theta_7 + r_{22} \cos \theta_7 \end{cases} \quad (3.44)$$

So

$$\begin{bmatrix} \sin \theta_7 \\ \cos \theta_7 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -\sin \theta_5 \\ \cos \theta_5 \end{bmatrix} \quad (3.45)$$

Hence $\theta_7 = \operatorname{atan2}(\sin \theta_7, \cos \theta_7)$.

By varying ψ angle, the flexible solutions are infinity. Also considering the fixed position and local flexible orientation, we have more solutions to select. The restraints are joint limits shown below and avoidance of switch between branches.

| Joint | Lower Limit | Upper Limit |
|--------------|--------------------|--------------------|
| i | (Degrees) | (Degrees) |
| 1 | -45.0 | 135.0 |
| 2 | -135.0 | 45.0 |
| 3 | -180.0 | 0.0 |
| 4 | -90.0 | 90.0 |
| 5 | 0.0 | 180.0 |
| 6 | 0.0 | 180.0 |
| 7 | 45.0 | 135.0 |

From fixed angle ψ comes the eight branches of solution to inverse kinematics on Sarcos Dextrous Arm(3.46):

$$\begin{aligned}
 \theta_2 > 0 & \left\{ \begin{array}{l} \theta_4 > 0 \\ \theta_4 < 0 \end{array} \right\} \left\{ \begin{array}{l} \theta_6 > 0 \text{ Solution1} \\ \theta_6 < 0 \text{ Solution2} \\ \theta_6 > 0 \text{ Solution3} \\ \theta_6 < 0 \text{ Solution4} \end{array} \right. \\
 \theta_2 < 0 & \left\{ \begin{array}{l} \theta_4 > 0 \\ \theta_4 < 0 \end{array} \right\} \left\{ \begin{array}{l} \theta_6 > 0 \text{ Solution5} \\ \theta_6 < 0 \text{ Solution6} \\ \theta_6 > 0 \text{ Solution7} \\ \theta_6 < 0 \text{ Solution8} \end{array} \right.
 \end{aligned} \tag{3.46}$$

We choose the shoulder down and elbow up solution for fixed $\psi = 0$ angle from eight branches of solutions with considering the joint angle limits.

3.2 Open Loop Kinematic Calibration

Calibration is an important issue for experimental robotics, since it is required for virtually any implementations. On the Sarcos Dextrous Arm, we need to calibrate parameters such as: position sensor reading offset and gain for each joint, robot end effector offset from the last joint coordinate origin, and robot base coordinate position offset and orientation matrix with respect to the sensor coordinate. In this section, we will develop the open loop kinematic calibration method [6].

3.2.1 The Kinematic Model

Both geometric and nongeometric parameters are required for kinematic calibration. The Denavit-Hartenberg convention is employed for the geometric parameters (Fig. 3.7). For a manipulator with n dof's, the end effector is located by the position

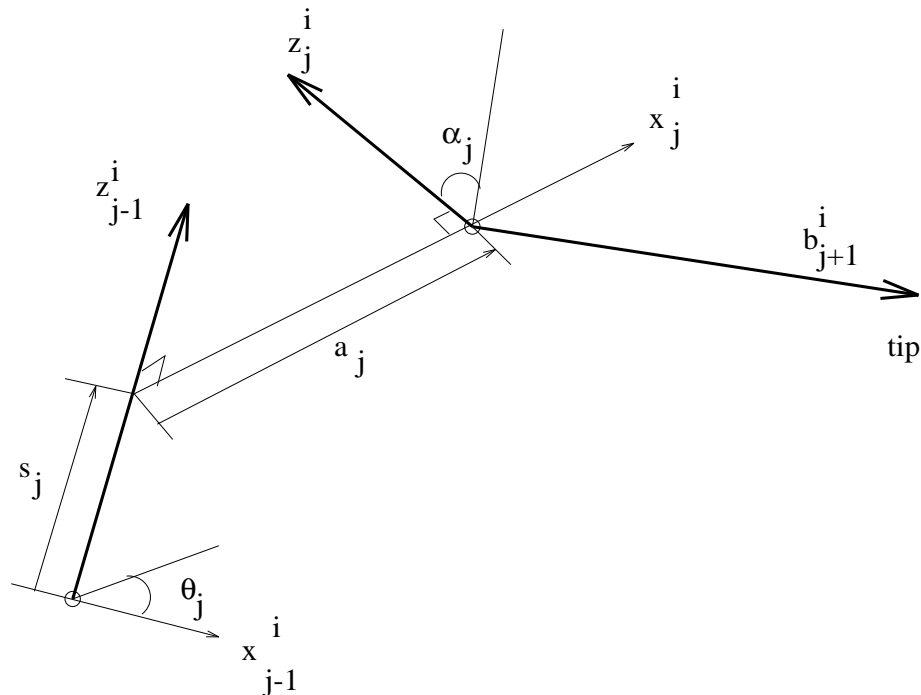


Figure 3.7: Denavit-Hartenberg coordinates and end effector vector \mathbf{b}_j^i

vector \mathbf{p}_c^i and the orientation matrix \mathbf{R}_c^i :

$$\mathbf{b}\mathbf{p}_c^i = \sum_{j=1}^n s_j \mathbf{z}_{j-1} + a_j \mathbf{x}_j \quad (3.47)$$

$$\mathbf{R}_c^i = \prod_{j=1}^n \mathbf{R}_z(\theta_j^i) \mathbf{R}_x(\alpha_j^i) \quad (3.48)$$

where $\mathbf{R}_z(\phi)$ and $\mathbf{R}_x(\phi)$ are 3×3 rotation matrices about the \mathbf{z} and \mathbf{x} axes by the angle ϕ , and the subscript c indicates that the position and orientation are computed from the model. The subscript i refers to the configuration of the manipulator, since in kinematic calibration it is placed into a number of configurations $\underline{\theta} = (\theta_1^i, \dots, \theta_m^i)$, $i = 1, \dots, m$. The required geometric parameters are s_j , α_j , and a_j for links $j = 1, \dots, n$.

The nongeometric parameters are focused at a joint and reflect errors between the true and measured joint angles; sources of error include backlash, gear eccentricity, joint compliance, and joint angle offsets. We model only the joint angle offset error θ_j^{off} , which needs to be identified. It is related to actual θ_j' and measured θ_j , the D-H joint angles by $\theta_j' = \theta_j + \theta_j^{off}$. All of unknown kinematic parameters are placed into a single vector $\underline{\varphi} = (\underline{\theta}_j^{off}, \underline{\alpha}, \underline{s}, \underline{a})$, where $\underline{s} = (s_1, \dots, s_n)$, etc.

Instead of orientation matrix \mathbf{R}_c^i , it is convenient to represent the orientation by the vector $\underline{\rho}_c^i = (\phi_x^i, \phi_y^i, \phi_z^i)$, representing the roll-pitch-yaw (ZYX) Euler angles: $\mathbf{R}_c^i = \mathbf{R}_z(\phi_z^i) \mathbf{R}_y(\phi_y^i) \mathbf{R}_x(\phi_x^i)$. The computed endpoint location $\underline{\mathbf{x}}_c^i = (\underline{\rho}_c^i, \mathbf{p}_c^i)$ may then be written as

$$\underline{\mathbf{x}}_c^i = \underline{f}(\underline{\theta}^i, \underline{\varphi}) \quad (3.49)$$

where the function \underline{f} is derived from (3.47) and (3.6). Its exact form is not required here.

3.2.2 Iterative identification

To estimate $\underline{\varphi}$, the manipulator must be moved into an adequate number m of configurations, in consideration of the large number of parameters in $\underline{\varphi}$ and of statistical

averaging. At each configuration i the actual endpoint location $\underline{\mathbf{x}}_a^i$ is measured. The goal is to determine the $\underline{\varphi}$ that best predict from the kinematic model(3.49) all of the endpoint measurement $\mathcal{X} = (\underline{\mathbf{x}}_a^1, \dots, \underline{\mathbf{x}}_a^m)$:

$$\mathcal{X} = \mathcal{F}(\underline{\varphi}) \quad (3.50)$$

where $\mathcal{F}(\underline{\varphi}) = (f(\underline{\theta}^1, \underline{\varphi}), \dots, f(\underline{\theta}^m, \underline{\varphi}))$.

Solving for $\underline{\varphi}$ from (3.50) is a nonlinear estimation problem, which can be done by linearization and iteration:

$$\Delta\mathcal{X} = \mathcal{C}\Delta\underline{\varphi} \quad (3.51)$$

where $\mathcal{C} = \partial\mathcal{F}/\partial\underline{\varphi}$. We can consider $\Delta\mathcal{X} = (\Delta\underline{\mathbf{x}}^1, \dots, \Delta\underline{\mathbf{x}}^m)$, with $\Delta\underline{\mathbf{x}}^i = \underline{\mathbf{x}}_a^i - \underline{\mathbf{x}}_c^i$, as the location errors. Similarly, $\Delta\underline{\varphi} = \underline{\varphi} - \underline{\varphi}_0$ is the error in the total parameter set, where $\underline{\varphi}_0$ is the current estimate and $\underline{\varphi}$ is the corrected estimate. In $\Delta\underline{\varphi}$, $\Delta\underline{\mathbf{s}} = \underline{\mathbf{s}} - \underline{\mathbf{s}}_0$, etc. An estimate of the parameter errors is provided by minimizing $LS = (\Delta\mathcal{X} - \mathcal{C}\Delta\underline{\varphi})^T(\Delta\mathcal{X} - \mathcal{C}\Delta\underline{\varphi})$, which yields

$$\Delta\underline{\varphi} = (\mathcal{C}^T\mathcal{C})^{-1}\mathcal{C}^T\Delta\mathcal{X} \quad (3.52)$$

Finally the guess at the parameters is updated as $\underline{\varphi} = \underline{\varphi}_0 + \Delta\underline{\varphi}$ and the iteration continues until $\Delta\mathcal{X} \rightarrow \mathbf{0}$.

The basis for linearization is the assumption that $\underline{\mathbf{x}}_c^i$ is close to $\underline{\mathbf{x}}_a^i$. Then

$$\Delta\underline{\mathbf{x}}^i = \underline{\mathbf{x}}_a^i - \underline{\mathbf{x}}_c^i = \begin{bmatrix} \Delta\underline{\rho}^i \\ \Delta\underline{\mathbf{p}}^i \end{bmatrix} \quad (3.53)$$

where $\Delta\underline{\rho}^i = (\partial\phi_x^i, \partial\phi_y^i, \partial\phi_z^i)$ is the incremental orientation error in terms of the Euler angles and $\Delta\underline{\mathbf{p}}^i = (\mathbf{d}\mathbf{x}^i, \mathbf{d}\mathbf{y}^i, \mathbf{d}\mathbf{z}^i)$ is the incremental position error. When $\underline{\varphi}_0$ is far from the final values, problems with this approach may occur, as discussed later.

The differential of (3.49) is

$$\Delta\underline{\mathbf{x}}^i = \frac{\partial\underline{\mathbf{x}}_c^i}{\partial\underline{\theta}}\Delta\underline{\theta} + \frac{\partial\underline{\mathbf{x}}_c^i}{\partial\underline{\alpha}}\Delta\underline{\alpha} + \frac{\partial\underline{\mathbf{x}}_c^i}{\partial\underline{\mathbf{s}}}\Delta\underline{\mathbf{s}} + \frac{\partial\underline{\mathbf{x}}_c^i}{\partial\underline{a}}\Delta\underline{a} = \mathbf{C}^i\Delta\underline{\varphi} \quad (3.54)$$

where

$$\mathbf{C}^i = \frac{\partial \underline{\mathbf{x}}^i}{\partial \underline{\varphi}} = \begin{bmatrix} \frac{\partial \underline{\mathbf{x}}^i}{\partial \theta} & \frac{\partial \underline{\mathbf{x}}^i}{\partial \alpha} & \frac{\partial \underline{\mathbf{x}}^i}{\partial \underline{\mathbf{s}}} & \frac{\partial \underline{\mathbf{x}}^i}{\partial \underline{\mathbf{a}}} \end{bmatrix} \quad (3.55)$$

and $\mathcal{C} = (\mathbf{C}^1, \dots, \mathbf{C}^m)$. The Jacobian \mathbf{C}^i is often formed through the use of differential homogeneous transformations, but to address identifiability we find the use of screw coordinates more convenient and transparent. To proceed, we must express differential rotations $\Delta \mathbf{r}^i = (\partial \mathbf{x}^i, \partial \mathbf{y}^i, \partial \mathbf{z}^i)$ about orthogonal axes rather than the differential rotations $\Delta \underline{\rho}^i$ about nonorthogonal axes. The two may be related by a matrix \mathbf{Q}^i , whose form depends on the particular Euler angles (ZYX here) chosen:

$$\Delta \mathbf{r}^i = \begin{bmatrix} 1 & 0 & -\sin \phi_y^i \\ 0 & \cos \phi_z^i & \sin \phi_z^i \cos \phi_y^i \\ 0 & -\sin \phi_z^i & \cos \phi_z^i \cos \phi_y^i \end{bmatrix} \Delta \underline{\rho}^i = \mathbf{Q}^i \Delta \underline{\rho}^i \quad (3.56)$$

Assuming the inverse of \mathbf{Q}^i exists, then

$$\Delta \mathbf{x}^i = \begin{bmatrix} \Delta \underline{\rho}^i \\ \Delta \mathbf{p}^i \end{bmatrix} = \begin{bmatrix} (\mathbf{Q}^i)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{r}^i \\ \Delta \mathbf{p}^i \end{bmatrix} = \underline{\mathbf{Q}}^i \begin{bmatrix} \Delta \mathbf{r}^i \\ \Delta \mathbf{p}^i \end{bmatrix} \quad (3.57)$$

where a 6×6 matrix $\underline{\mathbf{Q}}^i$ has been defined incorporating 3×3 zero matrices $\mathbf{0}$ and identity matrix \mathbf{I} .

The matrix \mathbf{Q}^i is not invertible when $\phi_y^i = \pi/2$, which is the singularity for this set of Euler angles. This problem is inescapably linked with the use of Euler angles, since no set of Euler angle exists that is the integral of angular velocity. At endpoint configurations with $\phi_y^i = \pi/2$ an alternate Euler angle formulation (say ZXZ) may be used to guarantee that \mathbf{Q}^i is always invertible. Without loss of generality we assume that appropriate Euler angle representations will be used for these few singular endpoint configurations and do not discuss them further.

The total incremental displacement $(\Delta \mathbf{r}^i, \Delta \mathbf{p}^i)$ is the sum of the instantaneous screws corresponding to variations in all of the parameters:

$$\begin{bmatrix} \Delta \mathbf{r}^i \\ \Delta \mathbf{p}^i \end{bmatrix} = \sum_{j=1}^n \Delta \theta_j \begin{bmatrix} \mathbf{z}_{j-1}^i \\ \mathbf{z}_{j-1}^i \times \mathbf{b}_j^i \end{bmatrix} + \Delta \alpha_j \begin{bmatrix} \mathbf{x}_j^i \\ \mathbf{x}_j^i \times \mathbf{b}_{j+1}^i \end{bmatrix} + \Delta s_j \begin{bmatrix} \mathbf{0} \\ \mathbf{z}_{j-1}^i \end{bmatrix} + \Delta a_j \begin{bmatrix} \mathbf{0} \\ \mathbf{x}_j^i \end{bmatrix} \quad (3.58)$$

where \mathbf{b}_{j+1}^i is a vector from the j th coordinate system to the endpoint(3.7). From the parameter-based screws on the right, Jacobian matrices can be formed whose columns j are

$$(\mathbf{J}_\theta^i)_j = \begin{bmatrix} \mathbf{z}_{j-1}^i \\ \mathbf{z}_{j-1}^i \times \mathbf{b}_j^i \end{bmatrix}, (\mathbf{J}_\alpha^i)_j = \begin{bmatrix} \mathbf{x}_j^i \\ \mathbf{x}_j^i \times \mathbf{b}_{j+1}^i \end{bmatrix}, \quad (3.59)$$

$$(\mathbf{J}_s^i)_j = \begin{bmatrix} \mathbf{0} \\ \mathbf{z}_{j-1}^i \end{bmatrix}, (\mathbf{J}_a^i)_j = \begin{bmatrix} \mathbf{0} \\ \mathbf{x}_j^i \end{bmatrix}, \quad (3.60)$$

where \mathbf{J}_β^i represents a Jacobian with regard to a particular parameter vector β ; thus \mathbf{J}_θ^i is the ordinary Jacobian related to joint angle displacement. More compactly,

$$\begin{bmatrix} \Delta \mathbf{r}^i \\ \Delta \mathbf{p}^i \end{bmatrix} = \begin{bmatrix} \mathbf{J}_\theta^i & \mathbf{J}_\alpha^i & \mathbf{J}_s^i & \mathbf{J}_a^i \end{bmatrix} \Delta \underline{\varphi} = \mathbf{A}^i \Delta \underline{\varphi} \quad (3.61)$$

From (3.61) and (3.57), the ensemble Jacobian \mathbf{A}^i is then related to the Jacobian \mathbf{C}^i by $\mathbf{C}^i = \underline{\boldsymbol{\Omega}}^i \mathbf{A}^i$. Define $\mathcal{A} = (\mathbf{A}^1, \dots, \mathbf{A}^m)$.

3.2.3 Parameter Scaling

For the kinematic method, the final calibration accuracy can depend on the numerical methods employed. Here we employ parameter scaling, which is one of the scaling methods available. Scaling is important for good numerical performance, but has not received that much attention in the calibration literature.

Consider the equation (3.51), right multiplication of \mathcal{C} by a weighting matrix \mathbf{H} results in parameter scaling:

$$\Delta \mathcal{X} = (\mathcal{C}\mathbf{H})(\mathbf{H}^{-1}\Delta \underline{\varphi}) = \tilde{\mathcal{C}}\Delta \tilde{\underline{\varphi}} \quad (3.62)$$

where $\tilde{\mathcal{C}} = \mathcal{C}\mathbf{H}$ and $\tilde{\underline{\varphi}} = \mathbf{H}^{-1}\Delta \underline{\varphi}$.

The most common approach towards parameter scaling is column scaling, which does not require a priori statistical information. Define a diagonal matrix $\mathbf{H} =$

$diag(h_1, \dots, h_r)$ with element

$$h_j = \begin{cases} \|\mathcal{C}_j\|^{-1} & \text{if } \|\mathcal{C}_j\| \neq 0 \\ 1 & \text{if } \|\mathcal{C}_j\| = 0 \end{cases} \quad (3.63)$$

where \mathcal{C}_j is the j th column of \mathcal{C} . Then the modified least square problem (3.62) becomes

$$\Delta \mathcal{X} = \sum_{j=1}^r \frac{\mathcal{C}_j}{\|\mathcal{C}_j\|} \Delta \varphi_j \|\mathcal{C}_j\| \quad (3.64)$$

3.2.4 Calibration on Sarcos Dextrous Arm

The simplified open loop calibration method is simulated for the Sarcos Arm in Matlab software package with end effector position measured only. For the first five joints, the optical encoders are used to measure the joint angle position with $\theta_i = K\psi_i + \theta_i^{off}$, where the ψ_i is the optical encoder reading, the gain is known as $K = \frac{\pi}{100000}(rad)$ and $i = 1, \dots, 5$. The last two joints use Rotary Variable Differential Transducers (RVDTs) to measure the joint angles with $\theta_i = k_i \psi_i + \theta_i^{off}$, where $i = 6, 7$, and gains are unknown. We add two extra coordinates \mathbf{O}_{-1} and \mathbf{O}_{-2} with eight DH parameters to be calibrated for the six unknown parameters in order to locate robot base coordinate with respect to global sensor coordinate. We set two DH parameters to be constants to reduce two extra degree of freedom. To calibrate the position offset from the reference point on the robot hand with respect to the seventh coordinate, we add two extra coordinate \mathbf{O}_8 and \mathbf{O}_9 with eight DH parameters. Same as before, we set five of them to be constants.

For the Jacobians of RVDT gains and offsets, using

$$\mathbf{J}_{\mathbf{k}}^i = \frac{\partial f}{\partial k_i} = \frac{\partial f}{\partial \theta_i} \cdot \frac{\partial \theta_i}{\partial k_i} = \psi \cdot \mathbf{J}_{\theta}^i \quad (3.65)$$

$$\mathbf{J}_{\theta^{off}}^i = \frac{\partial f}{\partial \theta_i^{off}} = \frac{\partial f}{\partial \theta_i} \cdot \frac{\partial \theta_i}{\partial \theta_i^{off}} = \mathbf{J}_{\theta}^i \quad (3.66)$$

then apply the open loop method discussed in the previous sections with column scaling for parameter identification.

3.3 Experiment Setup and Data Analysis

When the kinematics and calibration had been prepared, we set up the experiment in order to apply our geometric controller on real time dynamic tracking and grasping via OPTOTRAK, an Infrared motion tracking system using triangulation measurement (Fig. 3.8). We used a white plastic ball with four-inch diameter as our target, which

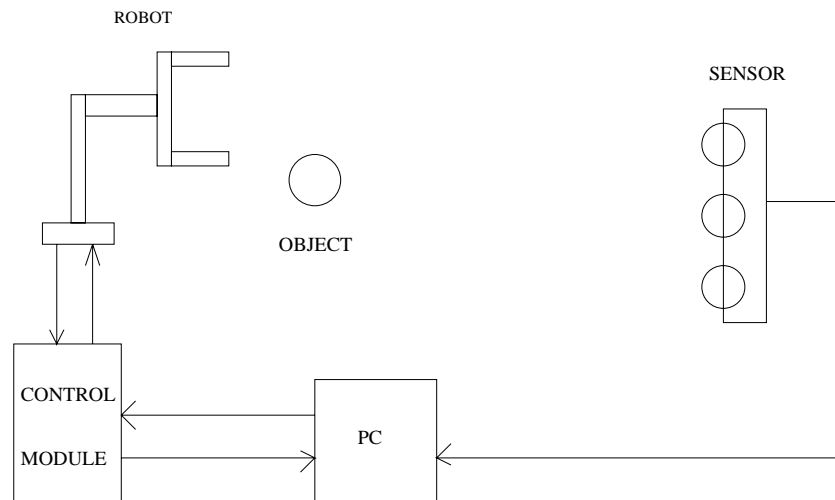


Figure 3.8: Experiment Setup

is attached at the end of an aluminum stick. Two infrared emitting diodes, we call them markers, are attached to the surface of the aluminum stick with known distance to the center of the ball. The positions of the marker are detected by three receiver formed OPTOTRAK motion tracking system. The data were collecting through the interface from OPTOTRAK to a PC and stored into a specific memory, which is used to determine the position of center of the ball. The sampling rate is over 1000Hz, fast enough to provide the updated global information. More markers up to six can be placed for full position and orientation of the object.

We have created software module in a C40 environment with the block diagram in Fig. 3.9. In the Initialization step, we set initial values to the variables. Next start the loop for the experiment. In the loop, first synchronize the time with checking the timer reading for the current loop, compared to the desired time, and wait until

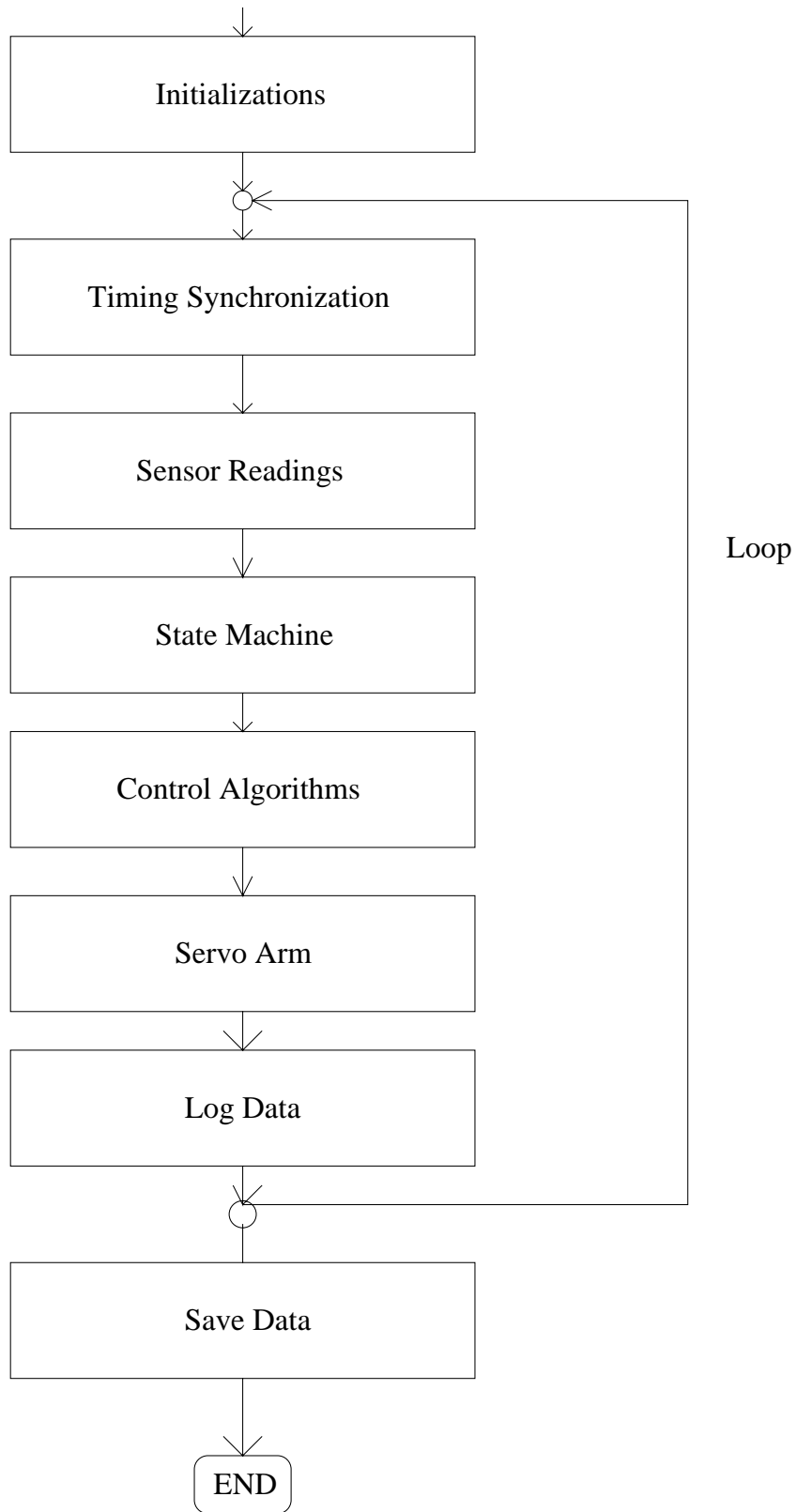


Figure 3.9: Experiment Block Diagram

arriving the desired time if necessary. If the time for current loop is late compare to the desired one, log the data, send error message and exit from the loop. This subroutine make sure that the sampling time period is identical for each loop and adjustable as we set. Timing synchronization is very important to implement on-line robotic manipulations. Next subroutine is reading sensors such as optical encoders, RVDTs and OPTOTRAK, where the updated readings from OPTOTRAK are obtained from specific location in the memory. We calculated the actual joint angles and object position from the sensor readings here. The state machine seems more complicated. It switches to different branches according to criterion and define the control algorithm. We define several branches as below:

- Sensor Initialize: Rotate the first five joints to pass certain index point so that optical encoder can be initialized. This is done once after the hydraulic system powered on.
- Home: Put arm to a desired rest pose named home.
- Joint space tracking: Control the robot to track a desired trajectory in joint space.
- Cartesian space tracking: Measure the object position via OPTOTRAK, invoke the inverse kinematics, and control the robot to reach the set points.
- Dynamic grasping: Apply our geometric controller to generate the desired Cartesian space position, invoke inverse kinematics, and control the arm to track the set points. When the position error is within the threshold, control the fingers to achieve the grasping.
- End: After the grasping is successfully performed, control the arm back to home pose

Next step is control algorithm which includes the joint space position control, joint space velocity control, Cartesian space position control where inverse kinematics is

invoked and grasping control by moving the 3 dof three-finger-hand from open to close. In servo arm subroutine, desired joint angles are written to the servo controller and drive the actuators to the desired position. In the servo controller, both torque and position control are available. In our experiment, we apply the position control with joint angle position, velocity feedback and valve compensation. We have optimized the gains for the PD controller and valve currents in order to eliminate the tracking error without causing any vibration on each joint. The data of loop time, desired and actual joint angles, object position, etc are logged in to an array in the log data subroutine. This loop is updated every 10 msec, at 100Hz sampling frequency. Finally the data will be saved into files as implementing record.

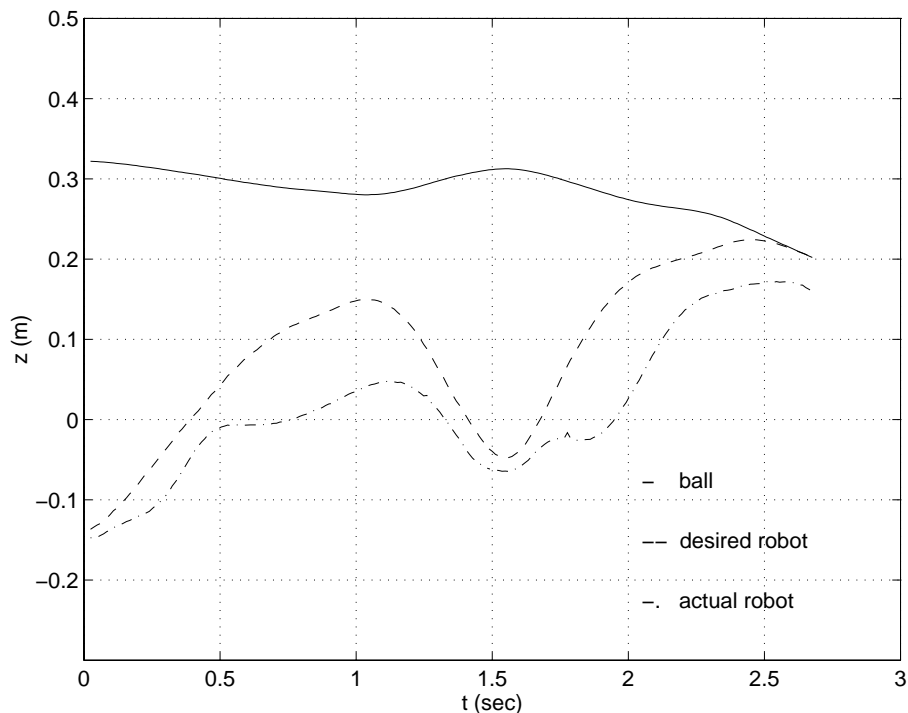


Figure 3.10: Dynamic Grasping Result in Z Dimension

In the experiment, we have successfully implemented the dynamic grasping in 3D space using our geometric controller to predict the desired robot trajectory and resulted in finally grasping the plastic ball and taking it to the home pose. The measured object position, desired and actual robot position during grasping are shown in

Fig. 3.10 for Z dimension and Fig. 3.11 for dimension X and Y, respectively.

We have seen that in Z dimension when object approaches and finally traverses the grasping surface, which has been defined as $b_z = 0.2$ here, the desired robot position, velocity and acceleration match those of the object's, while in the X and Y dimension the position matching have been synchronized as in Z dimension, as well as the velocity and acceleration matching in order to guarantee the dynamic grasping in theory. The robot arm tracks the desired trajectory with a position error less of 5 cm is not good enough but due to insufficient tracking of built-in joint level PD controller. This would be greatly improved by gravity compensation and Cartesian endpoint tracking. Due to lack of time, these methods were not implemented.

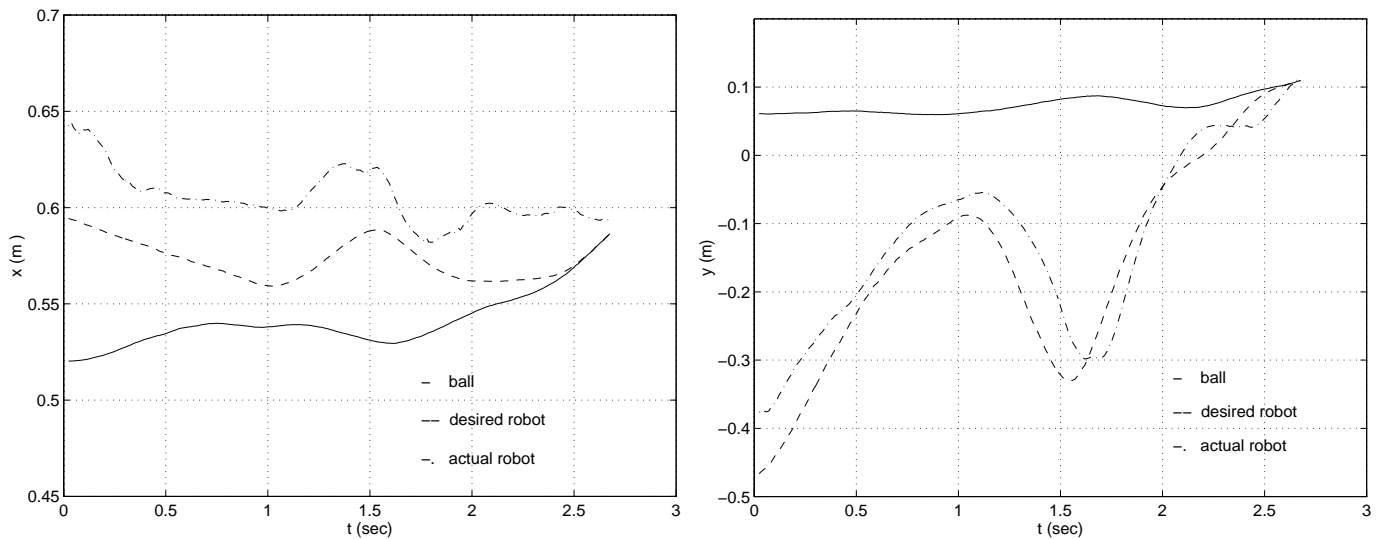


Figure 3.11: Dynamic Grasping Result in X and Y Dimension

The gravity influences can be eliminated by adding gravity compensation torque to the PD controller, which will be discussed in next section [16]. Regarding the global sensing error, local proximity sensing is developed to provide sufficient information of object's position related to the robot hand coordinate. Details will be presented in next chapter including local proximity sensor selecting, modelling, and calibration, as well as object location strategy and experiments.

3.4 Gravity Compensation

First we define some vectors shown in Fig. 3.12. Using the famous recursive **Newton-**

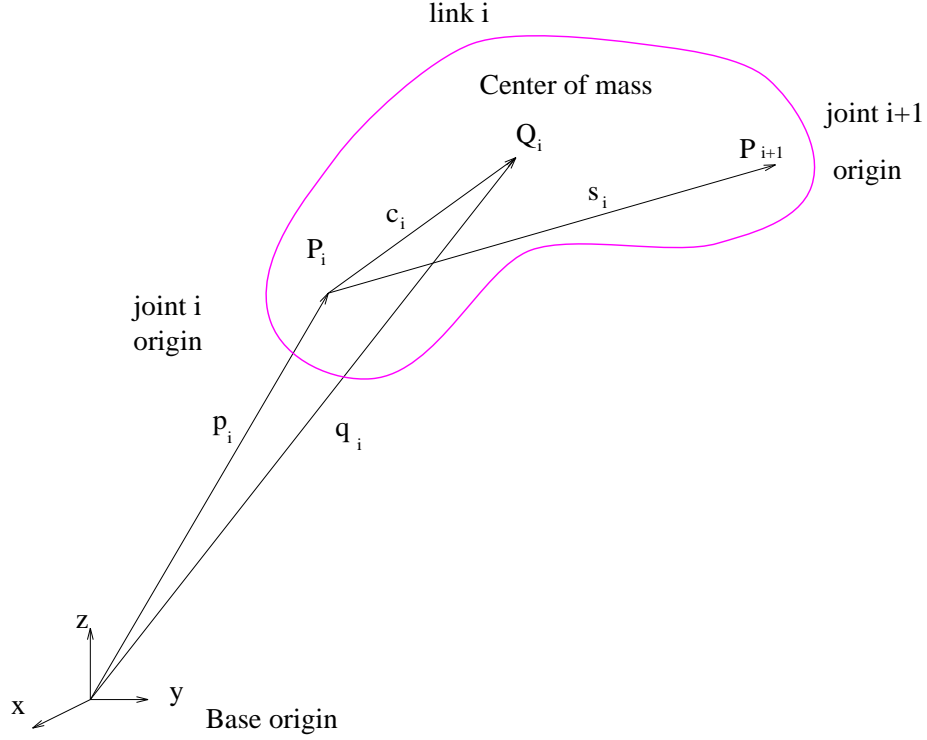


Figure 3.12: Vector Definition for Gravity Compensation

Euler formulation, we have been able to calculate the torque due to gravity on each joint. Considering $\ddot{\mathbf{p}}_i = \mathbf{0}$, $\dot{\boldsymbol{\omega}}_i = \mathbf{0}$ and $\ddot{\boldsymbol{\omega}}_i = \mathbf{0}$, the wrench acting on joint i from link i is

$$\mathbf{w}_{i,i} = \begin{bmatrix} \mathbf{f}_{i,i} \\ \mathbf{n}_{i,i} \end{bmatrix} = \begin{bmatrix} -\mathbf{g} & \mathbf{0} \\ \mathbf{0} & \mathbf{g} \times \end{bmatrix} \cdot \begin{bmatrix} m_i \\ m_i \mathbf{c}_i \end{bmatrix} = \mathbf{A}_i \cdot \boldsymbol{\phi}_i \quad (3.67)$$

The wrenches acting on joint i from adjacent link $i+1$ becomes

$$\mathbf{w}_{i,i+1} = \begin{bmatrix} \mathbf{f}_{i,i+1} \\ \mathbf{n}_{i,i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & \mathbf{0} \\ [\mathbf{s}_i \times] \cdot \mathbf{R}_i & \mathbf{R}_i \end{bmatrix} \cdot \begin{bmatrix} \mathbf{f}_{i+1,i+1} \\ \mathbf{n}_{i+1,i+1} \end{bmatrix} = \mathbf{T}_i \cdot \mathbf{w}_{i+1,i+1} \quad (3.68)$$

where \mathbf{R}_i is the transfer matrix from link $i+1$ coordinate to link i coordinate. Then we have

$$\mathbf{w}_{i,j} = \mathbf{T}_i \mathbf{T}_{i+1} \cdots \mathbf{T}_j \mathbf{w}_{j,j} = \mathbf{U}_{i,j} \boldsymbol{\phi}_j \quad (3.69)$$

where $\mathbf{U}_{i,j} = \mathbf{T}_i \mathbf{T}_{i+1} \cdots \mathbf{T}_j \mathbf{A}_j$ and $\mathbf{U}_{i,i} = \mathbf{A}_i$. The total wrench acting on joint i is

$$\mathbf{w}_i = \sum_{j=i}^n \mathbf{w}_{i,j} \quad (3.70)$$

Then we have the wrenches on all joints as

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_7 \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{1,1} & \mathbf{U}_{1,2} & \cdots & \mathbf{U}_{1,7} \\ \mathbf{0} & \mathbf{U}_{2,2} & \cdots & \mathbf{U}_{2,7} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{U}_{7,7} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_7 \end{bmatrix} \quad (3.71)$$

The torque due to gravity acting on joint i is then derived as

$$\tau_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \phi_i$$

and then

$$\underline{\tau} = \mathcal{K} \Phi \quad (3.72)$$

In formulation above, mass and center of mass of each link are required to calculate the gravity torque. This leads to the calibration issue of the required dynamic parameters by assuming that the robot kinematic parameters, torque sensor gains and offsets are known in advance, as well as gravity vector expressed in the robot base coordinates has been calibrated.

The calibration is applied the same iteration approach as open loop kinematic calibration in previous section with dynamic model (3.72). By minimizing the least square error $LS = (\underline{\tau} - \mathcal{K} \Phi)^T (\underline{\tau} - \mathcal{K} \Phi)$, which yields $\underline{\tau}_{estimate} = (\mathcal{K}^T \mathcal{K})^{-1} \mathcal{K}^T \underline{\tau}$. This method has been simulated for the Sarcos Dextrous Arm in Matlab software package.

Chapter 4

Infrared Proximity Sensing

When implementing the dynamic grasping, a problem arises from the global sensor system. Sometimes the object is undetected when it has been blocked by part of robot arm. In our software module, we check first if the IR diode is detectable, if not, we use the latest data to predict the desired robot trajectory. Obviously inefficient sensing increases tracking error and decreases the accuracy of grasping performance. This problem occurs more frequently when the robot end effector approaches to the object. Then we consider adding local sensors in order to provide sufficient global and local sensing information for dynamic grasping (Fig. 4.1). In this chapter, we describe an amplitude based electro-optical proximity sensor suitable for known environment and which meets our particular specifications. We summarize the model, calibration of the proximity sensor. An object localization strategy is presented as well with experimental results.

4.1 Local Sensor Requirements

The specifications for a local proximity sensor were derived by studying manipulation tasks in maintenance of live power lines and dynamic grasping of a hand tool, implemented on our Sarcos Dextrous Arm, a 7 dof robot arm with a three finger hand.

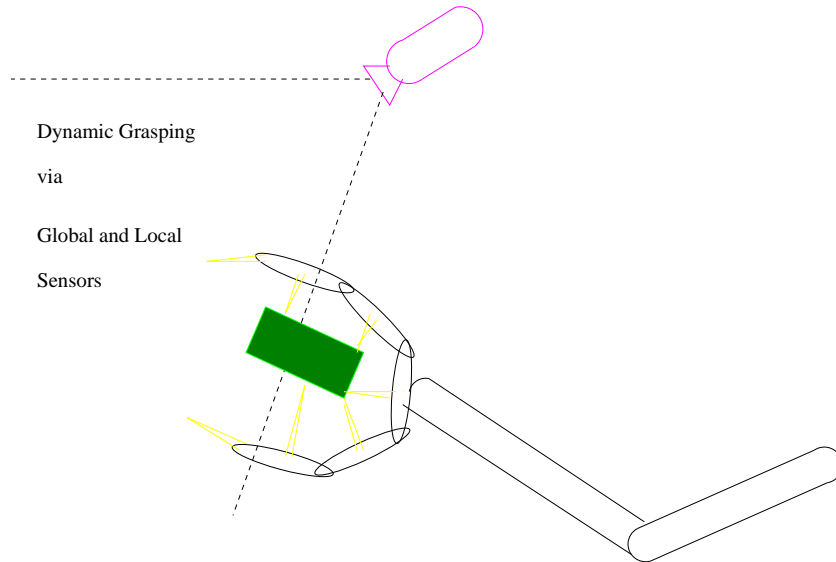


Figure 4.1: Global and Local Sensing for robotic manipulation

The two most stringed requirements arise from the combination of a working range down to $5mm$ and a size suitable for embedding in a human-size finger pad, limited to $5mm$ diameter. By recessing the sensor in the finger pad, an effective range down to $0mm$ can actually be achieved. Such an arrangement is also preferable for mechanical protection. Based on compliant finger pads, we estimate that an accuracy of $1mm$ just before contact is sufficient, with decreasing accuracy requirements as distance increases. In order to support low level sensor-based control, high sampling rates of at least $1kHz$ are required.

| | |
|---------------|-----------------------------|
| Working Range | 5-100 mm |
| Size | 5mm Diam., 5mm Height |
| Accuracy | 1mm (@ 5mm) – 10mm (@100mm) |
| Sampling Rate | 1kHz |

4.2 Local Sensor Options

For short range proximity sensing in robotic application, electro-optical sensors are at present the most appropriate. There are three basic approaches – triangulation, phase and amplitude based systems. The triangulation based approach includes Infrared Light Emitting Diode (IR LED) or Laser Diode, Optical Lens, and Position Sensor Diode (PSD) (Fig. 4.2). The output current from PSD is proportional to the offset x . The distance d from the object to the sensor pair is then calculated from (4.1).

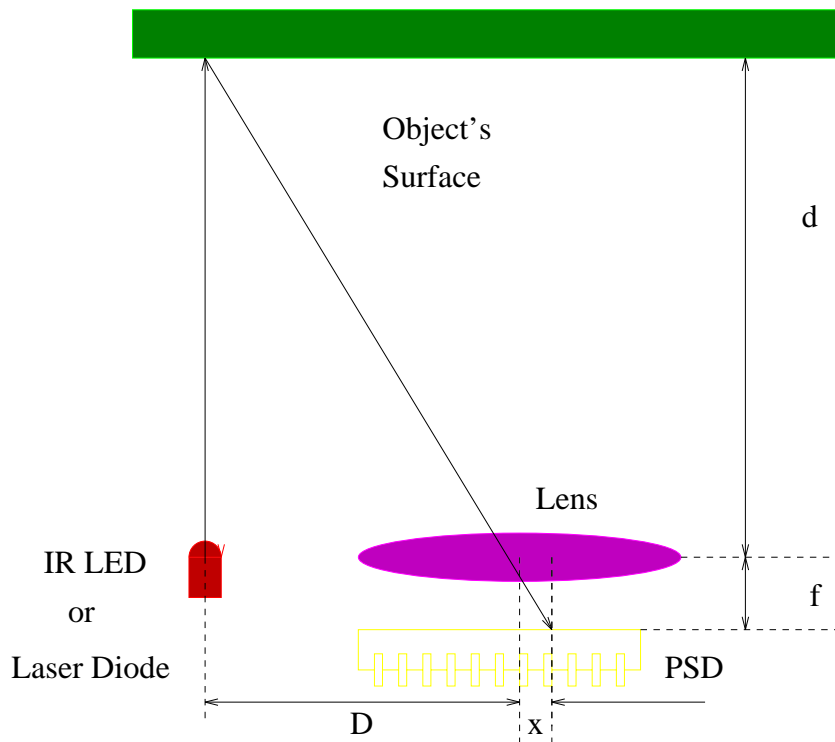


Figure 4.2: Triangulation Based local sensing system

$$d = \frac{f \cdot D}{x} \quad (4.1)$$

The phase based approach (Fig. 4.2) measured the phase difference ϕ between the signals received by the photo diode from IR LED **a** and IR LED **b**, respectively(4.2).

While triangulation and phase based systems are more independent from ambient light, surface material properties and surface orientation, they typically require a base separation between transmitter and receiver larger than $5mm$ which makes them ineligible for our purposes.

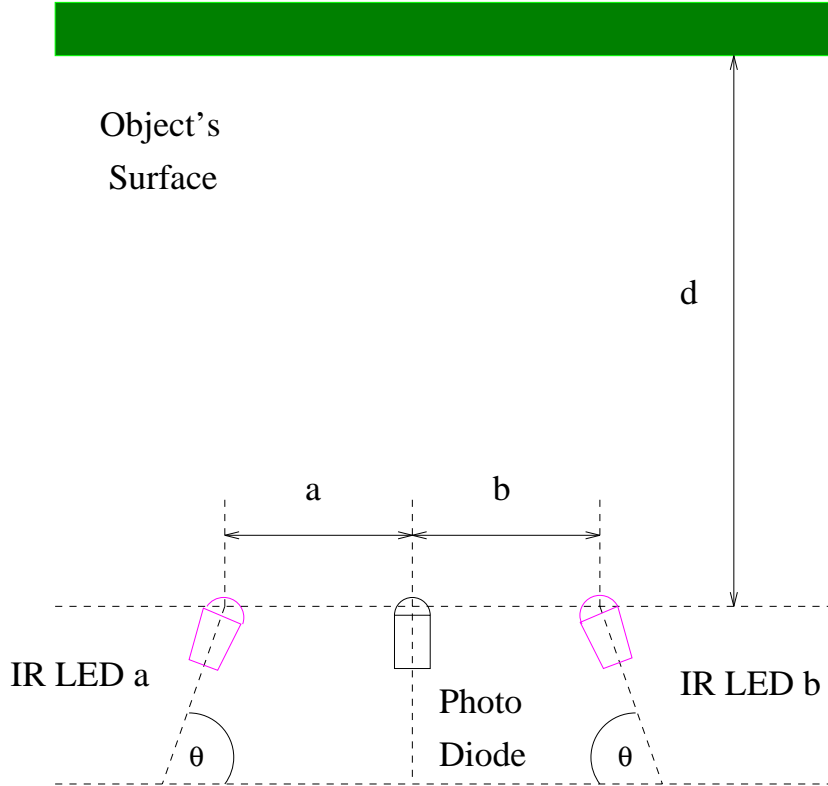


Figure 4.3: Phase Based local sensing system

$$\phi = \tan^{-1} \left[\frac{B}{A} \left(\frac{a^2 + d^2}{b^2 + d^2} \right)^{3/2} \right] \quad (4.2)$$

When the *magnitude* of the reflected energy is correlated to proximity, the transmitter and receiver pair can be placed as close together as physically possible, and we have built a prototype with less than $3mm$ diameter. Also due to its low cost, small size, high bandwidth, ruggedness, simplicity, ease of use, in our approach, we are going to employ several sensors (one sensor pair per finger pad), rely on signal processing to reduce ambient light dependence and exploit the knowledge of the object geometries

to achieve the accuracies required.

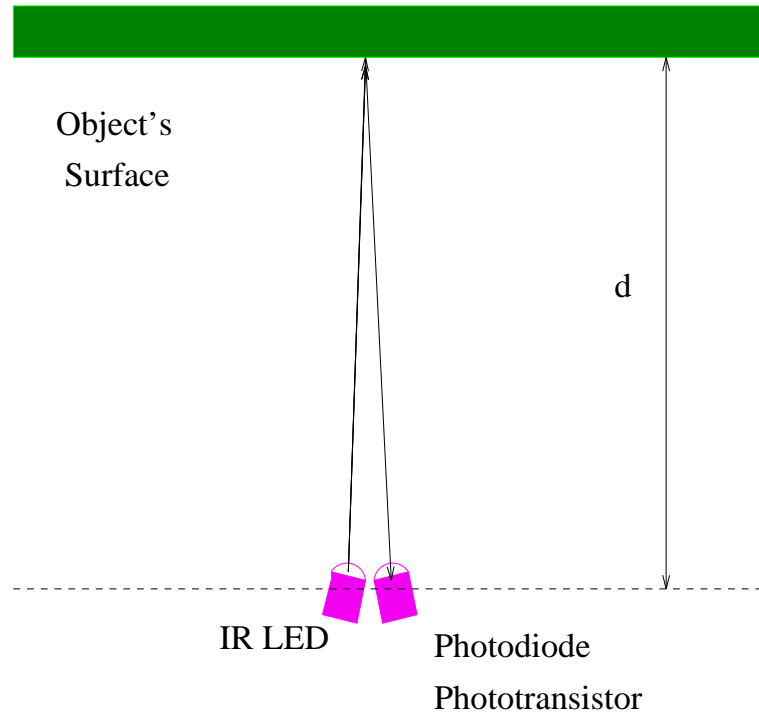


Figure 4.4: Amplitude Based local sensing system

$$d = g(I, \alpha, \lambda)$$

4.3 Infrared Sensor Model

The first system using infrared sensors seems to have been realized by Johnston at the Jet Propulsion Laboratory. A beam of infrared modulated light is emitted by a diode which is reflected by the target and demodulated: a focusing system is used. These sensors are principally employed in spatial applications to guide an operator, for example, by audio presentation of the proximity signals or in a multisensor configuration to locate an object by segmenting the sensitive area.

The infrared sensor system consists of one or more infrared emitting diodes (emitters) and a single silicon photo transistor (receiver). The wavelength of the infrared light is

centered at 890 nm and closely matches the spectral response of silicon photo transistors. The intensity of the infrared light changes versus the distance between the object and the sensor pair. The infrared light travels until it hits the surface of the object and is reflected back to the receiver. The photo transistor has narrow receiving angle which provides excellent on-axis coupling. The electrical signal given by the receiver depends on the intensity of the incident ray detected. The higher the intensity of the ray is received, the higher the current will be produced in the circuit of the receiver. So the voltage on the resistor will increase and in the mean time, the voltage between emitter and collector of the transistor will decrease. As the object is closed to the sensor pair, most of the ray will be reflected away from the receiving angle of receiver, so that the receiver can only obtain a little of it. Typically, the sensor sensitivity becomes insufficient beyond a distance of 10 cm from the object. If a larger distance is required, we can use series of higher pulse current in the circuit of the emitter to produce much higher intensity of the ray.

The intensity level measured via the IR sensor pair depends not only on the distance d , but on the orientation angle α (in planar case), and object surface properties. In the known environment, it is commonly assumed that the surface properties are known and favorable. A simple example is using white paper as the object. If the color of the object is black, there will be no ray reflected back at all, which is the extreme case. We define the albedo parameter λ to represent the object surface properties. In partially known environment, we will apply sensor fusion to calibrate the unknown albedo parameter.

Our model for amplitude based infrared sensor measurement is

$$v = f(d, \alpha, \lambda) \quad (4.3)$$

where v is the output voltage on photo transistor, d is the distance from the object to the infrared sensor pair, α is the angle between the normal vector of the object surface and the sensor pair axis, λ is the albedo parameter of the object surface properties.

To simplify the model, we linearize (4.3) as

$$v = \lambda f(d, \alpha) \quad (4.4)$$

For the standard known environment, let $\lambda = 1.0$, we calibrate function $f(d, \alpha)$. For the partially known environment, we can calibrate albedo parameter λ by sensor fusion (Fig. 4.5) and apply the same function as for known environment.

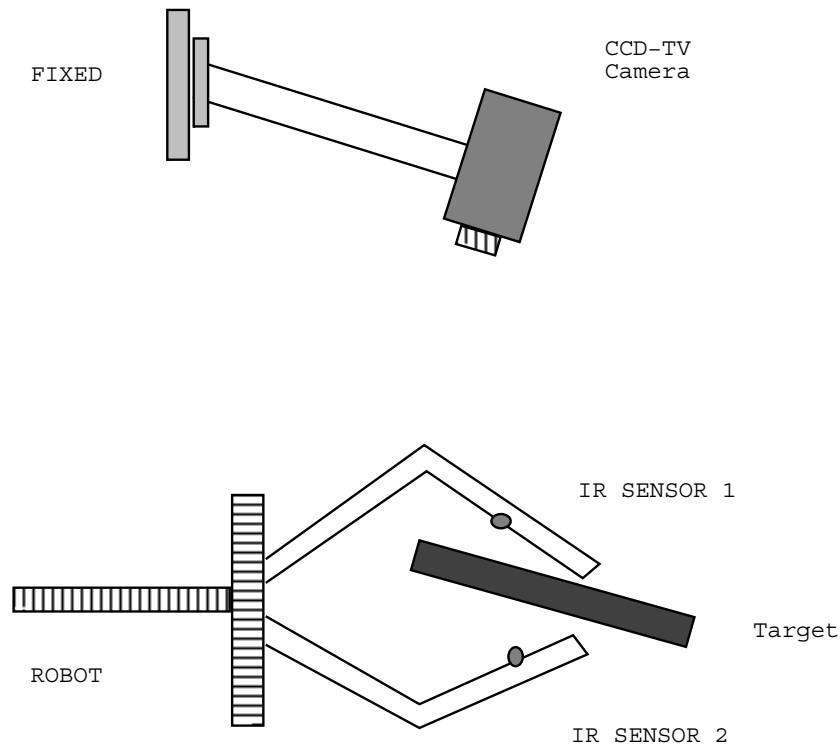


Figure 4.5: Albedo Parameter Calibration via Sensor Fusion

4.4 IR sensor Calibration and Characterization

In the calibration experimental setup (Fig. 4.4), the sensor device consists of a GaAIAs plastic infrared emitting diode—Type OP290A and a photo transistor—Type OP804SL. The measured object is a piece of white paper as standard workpiece which attached to a three degree of freedom motor driven platform. We drove the motors to move the

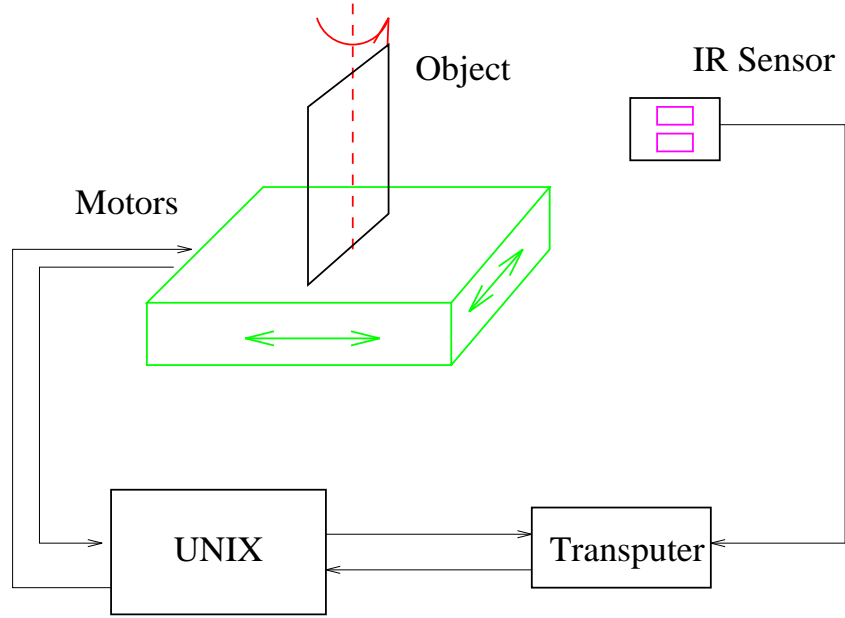


Figure 4.6: Calibration Experimental Setup

platform around the longitudinal and yaw axes for varied distances d and orientation angles α respect to sensor location. The measured data were collected through a transputer and transferred to UNIX system. The data processing are done using Matlab package including 2D surface fitting.

With a given albedo parameter $\lambda = 1.0$ of the object, in our case white paper, sample data showing the variation in sensor data as a function of object distance and orientation. By logging the sensor output while scanning a range of distances from $5mm$ to $80mm$ and surface orientations from 0 to $33deg$ the experimental surface plot shown in Fig. 4.8 was obtained. we also plot the output voltage versus distance and orientation angle, respectively, in Fig. 4.4. Applying the Least Square method for the 2D surface fitting in Matlab, we obtained satisfied estimation of the parameters for the model. The function of the output voltage versus the distance and the angle is shown as:

$$v = f(d, \alpha) = (\mathbf{C}_1 \cdot \underline{\alpha}) \times (\mathbf{C}_2 \cdot \underline{\mathbf{d}}) + (\mathbf{C}_3 \cdot \underline{\mathbf{d}}')$$

where $\mathbf{C}_1 = [4.9082, 2.7755e - 3, -1.2517e - 3, 6.4226e - 6, 9.7776e - 8]$,

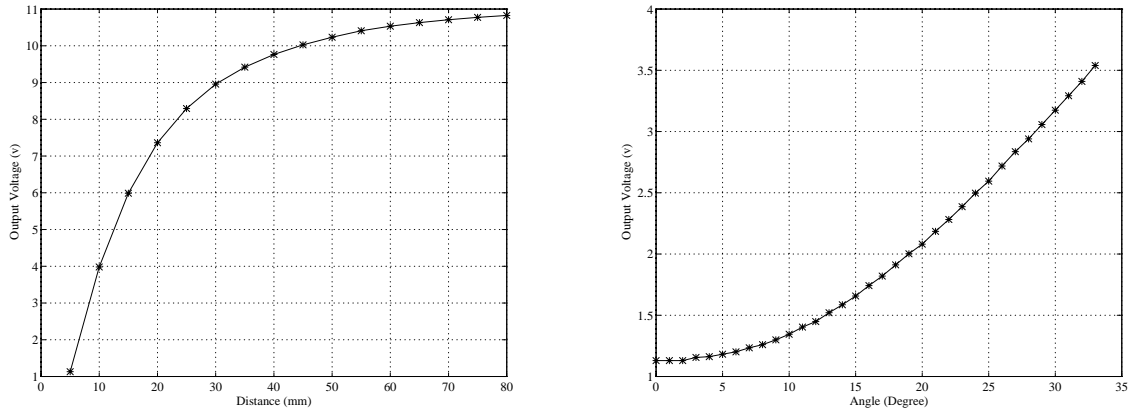


Figure 4.7: Output Voltage Curve versus Distance or Angle

$$\mathbf{C}_2 = [-3.4099, 1.8599e - 3, -4.5053e - 3, 5.1839e - 5, -2.22709e - 7],$$

$$\mathbf{C}_3 = [14.3440, -9.6711e - 2, 1.1178e - 3, -4.6416e - 6], \underline{\alpha} = [1, \alpha, \alpha^2, \alpha^3, \alpha^4]^T,$$

$$\underline{\mathbf{d}} = [1, d, d^2, d^3, d^4]^T, \underline{\mathbf{d}}' = [1, d, d^2, d^3]^T.$$

The surface fitting result is shown in Fig. 4.8 with the measured data.

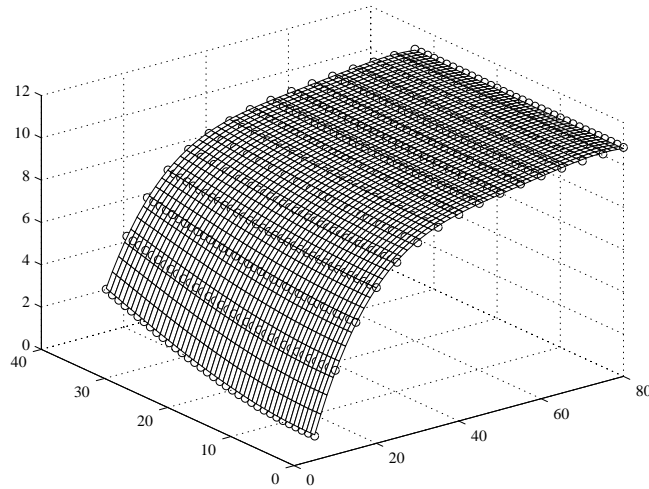


Figure 4.8: Surface fitting with measurement data (circles)

For the partially unknown environment, we calibrate the albedo parameter λ for

different colored paper in our experiment. The results is shown in Fig. 4.9. The model is modified by multiplying the albedo parameter with the function $f(d, \alpha)$ for different object surfaces.

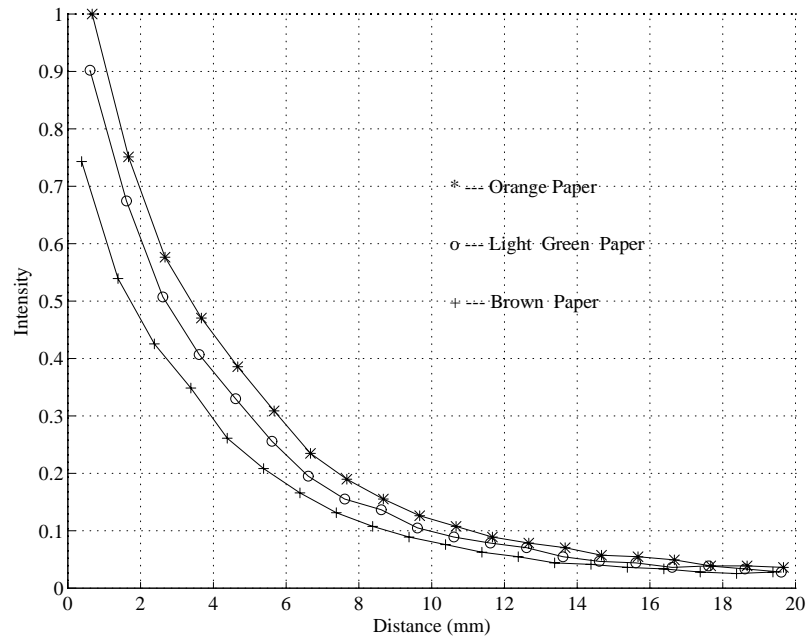


Figure 4.9: Albedo Parameter Calibration

4.5 Object Localization Strategy

We have calibrated the IR sensors and obtained the output voltage as a strictly monotonic function of the distance d , the surface angle α , and the albedo parameter λ ,

$$v = \lambda f(d, \alpha).$$

From the simultaneous measurements of two sensors which are a distance d_0 apart, we obtain two equations with four unknowns,

$$v_1 = \lambda f(d_1, \alpha_1)$$

$$v_2 = \lambda f(d_2, \alpha_1).$$

In the case of partially known environment, two additional equations can be added from the object geometry and we can solve the resulting system of equations. The parameters d_1 , d_2 , α_1 and α_2 describe the location and orientation of the object with respect to the sensor pairs. Without restriction of generality, we assume here that the two sensor pair lie in a plane. In general, this is not the case, for example, when each one is located in a separate finger pad. We now derive the three simple geometries for flat, rectangular and round object shapes.

Locally Flat Surface:

$$\alpha_1 = \alpha_2 = \alpha$$

$$\tan \alpha = \frac{d_2 - d_1}{d_0}.$$

From two measurements v_1, v_2 and d_0 (Fig. 4.10) we can solve for d_1 and d_2 .

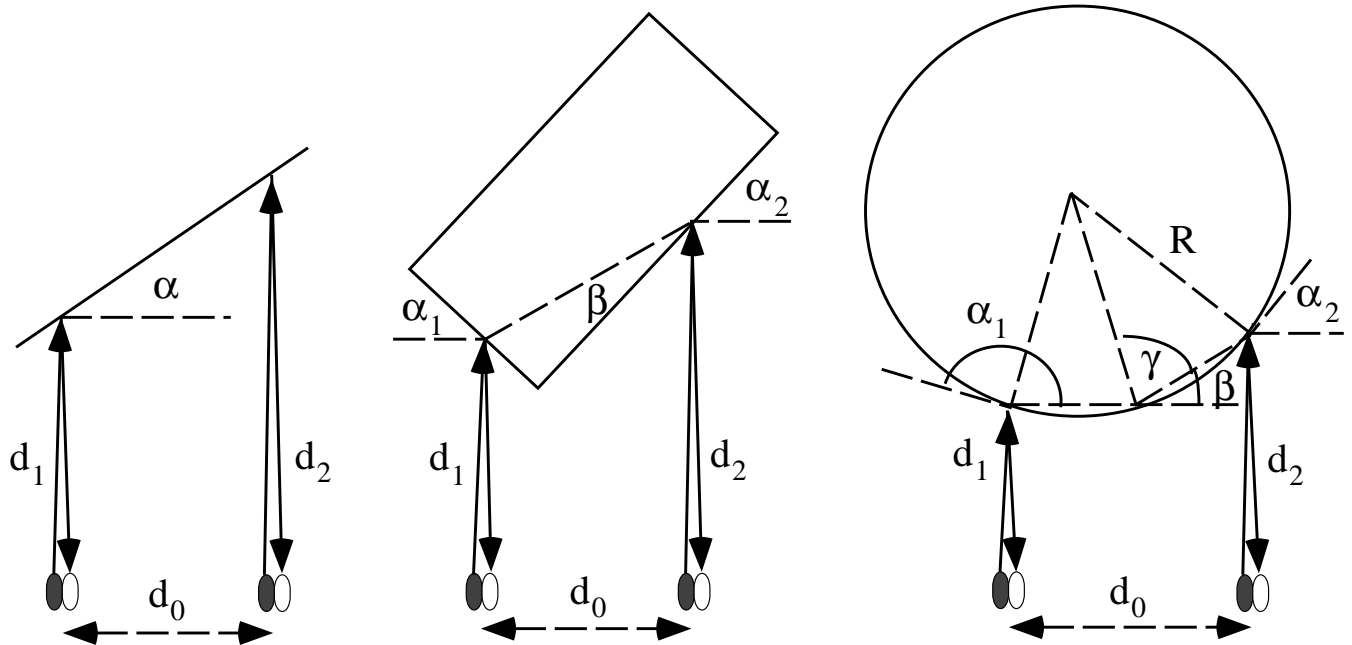


Figure 4.10: Sensing a Planar, Rectangular and Circular Object

Rectangular Object:

For a rectangle (Fig. 4.10), the added equations are

$$\begin{aligned}\frac{\pi}{2} - \alpha_1 + \alpha_2 &= 0 \\ d_2 - d_1 &= d_0 \tan(\alpha_2 - \beta), \\ \sin\beta &= \frac{(d_0 - \frac{d_2 - d_1}{\tan\alpha_2}) \sin\alpha_2}{\sqrt{d_0^2 + (d_2 - d_1)^2}}.\end{aligned}$$

Circular Surface:

In case of a circular or cylindrical object (Fig. 4.10), we use the same method and obtain the additional equations

$$\begin{aligned}d_0 &= 2R \cos(\alpha_1 - \frac{\pi}{2}) + \frac{d_2 - d_1}{\tan\beta}, \\ d_2 - d_1 &= 2R \cos\gamma \sin\beta,\end{aligned}$$

$$\text{where } \beta = \frac{\pi}{2} - \frac{\alpha_1 - \alpha_2}{2}, \gamma = \pi - \frac{\alpha_1 + \alpha_2}{2}.$$

This technique provides an effective method for locating the object from IR sensors, and is currently being extended to the spatial case with a larger number of sensors and more complex object geometries. The resulting system of equations are highly nonlinear and have to be solved online via efficient numerical methods.

4.6 Object Localization Experiment

For our object localization experiments we have selected a readily available and inexpensive transmitter/receiver pair, the Optek OP290A GaAlAs IR LED and the OP804SL photo transistor. Each device measures 1.3mm diameter and 3mm height, and satisfies the size constraints set out earlier. The albedo parameter of the object is given $\lambda = 1.0$, in our case white paper.

Based on this sensor characterization, we have implemented the object localization experiment for a planar surface as described above. From the two sensor measurements v_1, v_2 , the sensor pair separation d_0 , and the sensor characterization $f(d, \alpha)$ we

estimate the object pose using Matlab as $\hat{d}_1 = 4.82mm, \hat{d}_2 = 20.08mm, \hat{\alpha} = 15.25deg$. With a true pose of $d_1 = 5mm, d_2 = 20mm, \alpha = 15deg$, the errors in proximities $d_1 - \hat{d}_1 = 0.18mm, d_2 - \hat{d}_2 = -0.08mm$ are well within our requirements.

4.7 Future Work

Local proximity sensing is suitable to be combined with global sensing system in order to provide sufficient sensing information for robotic manipulation. There still exist some uncertainties such as ambient lights effect, environment and object dependence. But it shows great advantages in sensor fusion with other accurate global sensing systems.

Planar sensing issue has been discussed in this chapter, while our next stage will expand the sensing method to 3D space with multiple sensor pairs. An array of infrared sensor pairs will be located in the palm and fingers of the robot hand. How many sensor pairs is needed at least, where to locate each of them, how to design the hardware module, data processing of multiple sensing signals, and object pose estimation based on multiple sensing signals will be investigated. In the future more developments will be pursued in the field of local infrared proximity sensing for robotic manipulation.

Chapter 5

Conclusions

The main goal of developing and implementing the geometric controller is to generate a desired robot reference trajectory from object pose measurement for smooth grasping. This thesis has focused on developing the basic 1 dof atan function to the full 6 dof quintic polynomial which satisfies all the grasping conditions. It guaranteed to match the object's velocity and acceleration on a specified grasp surface, where the grasping location is considered instead of grasping time as in traditional approaches. Since it does not depend on future state predictions, no object model is required. Without trajectory prediction, the computational effort is drastically reduced, allowing for higher controller speed and tracking feedback gains. In contrast, the traditional approach keeps planning and replanning the robot reference trajectory from the latest measurements.

The real time implementation of our geometric controller shows that it satisfied experimental data while actually grasped the moving object in 3D space. Kinematics, calibration, servo controller optimization on the Sarcos Dextrous Arm have been implemented, as well as gravity compensation been discussed. Future work will include reducing the tracking errors and optimizing the grasping performances, for example, adding infrared sensor arrays in the robot hand to the global sensing system based on the IR model, calibration and object location strategy developed in this thesis.

More investigation will be pursued in the orientation control for dynamic grasping and singularities avoidance.

Bibliography

- [1] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Trajectory filtering and prediction for automated tracking and grasping of a moving object. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1850–1856, May 1992.
- [2] R. L. Andersson. Real time intelligent visual control of a robot. In *IEEE Workshop on Intelligent Control*, Aug 1986.
- [3] J. Baillieul, J.M. Hollerbach, and R. Brockett. Programming and control of kinematically redundant manipulators. In *Proc. 23rd IEEE Conf. Decision and Control*, pages 768–774, Las Vegas, Dec 1984.
- [4] D.J. Balek and R.B. Kelley. Using gripper mounted infrared proximity sensors for robot feedback control. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 282–7, Silver Spring, MD, March 1985. IEEE Computer Society Press.
- [5] A. K. Bejczy. Effect of hand-based sensors on manipulator control performance. *Mechanism and Machine Theory*, 12(5):547–67, 1977.
- [6] D.J. Bennett and J.M. Hollerbach. Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints. In *IEEE Trans. Robotics and Automation*, volume 7, pages 597–606, 1991.
- [7] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. A Simple Juggling Robot: Theory and Experimentation. In V. Hayward and O. Khatib, editors, *Experimental Robotics I*. Springer-Verlag, 1989.

- [8] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. A Family of Robot Control Strategies for Intermittent Dynamical Environments. *IEEE Control Systems Magazine*, 10(2), Feb 1990.
- [9] M. Buehler, D. E. Koditschek, and P. J. Kindlmann. Planning and Control of Robotic Juggling and Catching Tasks. *Int. J. Robotics Research*, 13(2):101–118, 1994.
- [10] G. Buttazzo, B. Allotta, and F. Fanizza. Mousebuster: A robot for real-time catching. *IEEE Control Systems Magazine*, 14(1), Feb 1994.
- [11] J.Y. Catros, A. Dore, B. Espiau, and J.Y. Yclon. Automatic grasping using infrared sensors. In *Proceedings of the 8th International Symposium on Industrial Robots*, pages 132–42, Stuttgart, Germany, May 1978.
- [12] J. Côte, C. Gosselin, and D. Laurendeau. A graphic simulator for the study of robotic tracking and catching operations. In *Computers in Engineering*, 1992.
- [13] J. Côté, C. Gosselin, and D. Laurendeau. Tracking a moving object with a 6-dof manipulator. *Machine Vision and Robotics*, 1964, 1993.
- [14] H. A. Ernst. *MH-1, a computer-operated mechanical hand*. PhD thesis, M.I.T., Dec 1961.
- [15] C. Gosselin, J. Côté, and D. Laurendeau. Inverse kinematic functions for approach and catching operations. *IEEE Trans. Systems, Man, and Cybernetics*, 1993, to appear.
- [16] J. M. Hollerbach and G. Sahar. Wrist-partitioned inverse kinematic accelerations and manipulator dynamics. In *Intl. J. Robotics Research*, pages 61–76, 1983.
- [17] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1990.

- [18] B. Hove and J.-J. E. Slotine. Experiments in robotic catching. In *American Control Conference*, 1991.
- [19] A.R. Johnston. Proximity sensing technology for manipulator end effectors. *Mechanism and Machine Theory*, 12:95–109, 1977.
- [20] D. E. Koditschek and M. Buehler. Analysis of a Simplified Hopping Robot. *Int. J. Robotics Research*, 10(6):587–605, Dec 1991.
- [21] Y.F. Li. Characteristics and signal processing of a proximity sensor. *Robotica*, 12:335–41, 1994.
- [22] Z. Lin, V. Zeman, and R. V. Patel. On-line robot trajectory planning for catching a moving object. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1989.
- [23] J. Lloyd and V. Hayward. Trajectory generation for dynamic, sensor-driven environments. *Int. J. Robotics Research*, 12(4), 1993.
- [24] J. Marszalec. A proximity sensing system for an intelligent optically-powered robot gripper. *International Journal of Optoelectronics*, 4(3/4):343–55, May-Aug 1989.
- [25] R. Masuda. Multifunctional optical proximity sensor using phase modulation. *Journal of Robotic Systems*, 3(2):137–47, 1986.
- [26] O. Partaatmadja, B. Benhabib, and A. Goldenberg. Analysis and design of a robotic distance sensor. *Journal of Robotic Systems*, 10(4):427–45, 1993.
- [27] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *Proc. IEEE Int. Conf. Robotics and Automation*, Atlanta, Georgia, May 1993.
- [28] M. Zhang and M. Buehler. Sensor-based online trajectory generation for smoothly grasping moving objects. In *Int. Symp. Intelligent Control*, pages 141–146, Columbus, OH, Aug 1994.

- [29] M. Zhang, J. Damianakis, and M. Buehler. Sensing and control for autonomous grasping in dynamic environments. In *World Congress on Intelligent Manufacturing Processes and Systems*, pages 641–649, Puerto Rico, Feb 1995.