NTNU
Norwegian University of
Science and Technology

Coordinated Control with Obstacle Avoidance for Robot Manipulators

Magnus Christian Bjerkeng

Master of Science in Engineering Cybernetics
Submission date: June 2010
Supervisor: Kristin Ytterstad Pettersen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

SINTEF and NTNU are, together with StatoilHydro, investigating how to remotely control operations on offshore oil platforms by employing robots. One of the key challenges for remote control of offshore oil platforms is how to endow an onshore operator with the right amount and type of information so that the operator can remotely control the various operations needed to be carried out on an oil platform (e.g., by controlling a robot manipulator which performs these operations). This can be partly solved by having a robot manipulator act as an automated camera platform that ensures a clear view of the robot operation carried out by a second robot controlled by the operator.

In order to develop such a system, a controller is needed that endows the robot with the camera mounted to follow the motion of the operator controlled robot without colliding with the other robot, itself, or its environment. Hence, a controller with collisions avoidance capabilities for coordinated control of robot manipulators is needed. Such coordinated control will be the topic of this project.

Preliminary set of tasks:

1. Perform a literature study on coordinated control of robot manipulators with collision avoidance.
2. Propose a strategy for coordinated control with obstacle avoidance for the two robot manipulators in the lab
3. Implement a simulator for the robot manipulator
4. Verify the proposed strategy by simulations

Assignment given: 11. January 2010
Supervisor: Kristin Ytterstad Pettersen, ITK

NTNU
Department of Engineering Cybernetics

Coordinated Control with Obstacle Avoidance for Robot Manipulators

Magnus C. Bjerkeng

Norwegian University of Science and Technology
Department of Engineering Cybernetics

June 2010

Supervisor
Kristin Y Pettersen

co-supervisors
Erik Kyrkjebø
Aksel Transeth

Summary

This thesis addresses the problem of robot synchronization with obstacle avoidance. While these two fields have been studied extensively on their own, they have not yet been considered together as one problem.

This thesis is divided roughly into four parts which are to some extent self contained. The theory is presented in a narrative that culminates with the stability proof of the proposed controller. Examples and figures are used in order to keep the material manageable and readable.

The introductory part of the thesis consists of chapters 1 and 2. We present the notation and some mathematical background which is necessary for the theoretical analysis. We go on to review the diversity of ways in which one may approach this problem from a control design standpoint.

We derive the robot dynamical model in chapter 3 as well as solve other modeling specific problems. This chapter is of little theoretical interest, but is needed to implement a simulator on which we may test our controller. This chapter contains no new contributions but can be read as a guide to robot modeling.

The first contributions in this thesis are found in chapter 4 where we propose a real time implementable solution for solving the shortest distance estimation problem. It is important to know the distance to an obstacle in order to avoid it. The solution is a dynamic implementation of a steepest descent optimization scheme which is suitable to run on-line.

Chapter 5 is an introduction to the involved control design found in chapter 6. We review results from obstacle avoidance literature and argue for our choice of using the task space control design method.

The main contribution of this thesis is found in chapter 6. A controller is developed and is shown to produce a stable closed loop system. We first develop a controller considering only collision for the end effector, and then we

extend this to work with full robot collision. The response of the robot is such that it will track a reference trajectory whenever it is locally possible. When one cannot track the reference trajectory because of obstacles hindering the movement, then the trajectory is tracked in all directions in which the robot can move freely. The controller is simple and elegant, and does not rely on heuristics common in traditional solutions to obstacle avoidance control.

Preface

This thesis is the culmination of my master studies at the department of Engineering Cybernetics at the Norwegian University of Science and Technology. The thesis was written over a 5 month period ending the 14th of June 2010.

I was guided throughout the process of writing this thesis by my main supervisor Professor Kristin Y. Pettersen and co-supervisors PhDs Erik Kyrkjebø and Aksel Transeth. They have provided me with theoretical support and research ideas as well as editorial suggestions. They have always been positive and encouraging which has helped me tremendously especially in times of slow progression. This thesis would not have been completed without their patience and support. They deserve my deepest thanks. I look forward their continuing support as I embark on my PhD starting fall 2010.

I would also like to thank my office mates Kristoffer Fikkan, Magnus Rørvik and Idar Haugstuen which together have made the my day a joy throughout the last year. Finally I would like to thank my family and especially my girlfriend Nathalie who apart from enduring my endless robot rants without going crazy, has given me lots of love and support throughout my master studies.

Magnus C. Bjerkeng

Trondheim 14th June 2010

Contents

List of Figures

Chapter 1

Introduction

We consider the problem of synchronization with obstacle avoidance in this thesis. The result of the thesis is a low level controller designed to achieve this goal. The theoretical results derived are backed up by simulation results and examples.

1.1 Motivation

Traditional industrial robots follow pre-programmed motion routines. This works perfectly if the robot is set to perform a given task in repetition in a static environment. The application we will consider is a space inhabited by two or more robots. A follower robot has the objective of executing a movement in synchrony with leader robot. The leader robot is controlled by some external means, and moves independently of the follower. They operate on an oil platform, where the environment may be changing over time, a picture of the setup is seen i figure 1.1. In this scenario where robots are working closely together it is imperative that the robots do not collide to prevent them from damaging themselves and their environment. For this purpose we need to bestow our robot with some apparent intelligence. We want the follower robot to move in synchrony with the leader without colliding with anything. When the robot is on a collision course we need it to take appropriate measures in order for it not to collide, but at the same time we want the robot to perform its assigned task predictably and in a desirable way even when its given task which is infeasible due to obstacles in its environment.

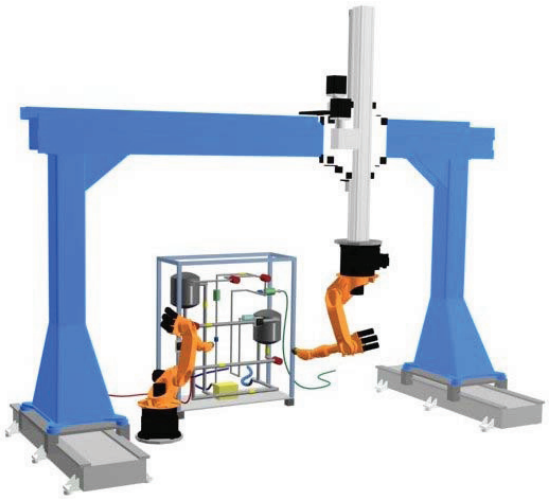


Figure 1.1: Two robots working in close proximity on a mock up process plant in the NTNU Robot Lab.

1.2 Preliminaries

We briefly present the notation we will be using and some background material needed for the analysis.

1.2.1 Notation and terminology

Vectors and matrices are denoted by uppercase and lowercase letters respectively. Dots denote differentiation with respect to time $\dot{x} = \frac{dx}{dt}$. The notation used is in compliance with the standard notation in control theory literature.

We adopt the following terminology from [10]. A **configuration** of a manipulator is a complete specification of the location of every point on the manipulator. The set of all configurations is called the **configuration space**. The vector q contains the joint angles of the robot labelled $q_1 - q_6$. The **joint space** is the vector space spanned by q. A robot has n **degrees of freedom** (DOF) if its configuration has be minimally specified by n parameters. The **workspace** of the robot is the set of all points that are reachable by the end effector (the tip of the robot arm). $SO(3)$ refers to the (special orthogonal) group of rotational matrices. The **Task space** or **Configuration space** is a vector space describing the position and orientation of the robots end effector. This space is spanned by the composite vector $[x, \Phi] \in \mathcal{R}^3 \times SO(3)$

where x is a position and Φ describes the orientation of the end effector.

1.2.2 Vector projection

An introduction to vector projection can be found in [4]. We will denote the projection of a vector $a \in \mathbb{R}^n$ along the vector $b \in \mathbb{R}^n$ as $a^{\parallel_b}$, and the projection of a perpendicular to b as $a^{\perp_b}$, see figure 1.2.

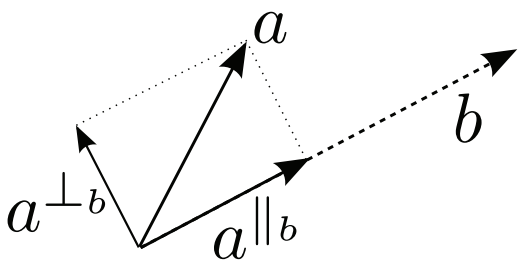


Figure 1.2: Two vectors a and b and the projections of a parallel to b and perpendicular to b.

We obtain them from the following relationship:

$$a^{\perp_b} = a - (a^T b)\frac{b}{\|b\|^2}, \quad a^{\parallel_b} = (a^T b)\frac{b}{\|b\|^2} \tag{1.1}$$

We see that $a = a^{\parallel_b} + a^{\perp_b}$. If b has length one, then the expression simplifies to:

$$a^{\perp_b} = a - (a^T b)b, \quad a^{\parallel_b} = (a^T b)b \tag{1.2}$$

The projection may also be expressed as a linear transformation using the **Transformation Matrix** $S(b)$[1] in the following way:

$$a^{\perp_b} = S(b)a, \quad a^{\parallel_b} = (I - S(b))a \tag{1.3}$$

Where I is the identity matrix. We can compute S in $\mathbb{R}^n$ as:

[1]This matrix is referred to as the select matrix in the field of hybrid force motion

$$S = \begin{bmatrix} \frac{\partial a - [a^T b]b}{\partial a_1} & \frac{\partial a - [a^T b]b}{\partial a_2} & \cdots & \frac{\partial a - [a^T b]b}{\partial a_n} \end{bmatrix} \tag{1.4}$$

S may be computed as the following in $\mathbb{R}^3$ and $\mathbb{R}^2$ assuming b is normalized:

$$S_3(b) = \begin{bmatrix} 1 - b_1^2 & -b_1 b_2 & -b_1 b_3 \\ -b_1 b_2 & 1 - b_2^2 & -b_2 b_3 \\ -b_1 b_3 & -b_2 b_3 & 1 - b_3^2 \end{bmatrix}, \quad S_2(b) = \begin{bmatrix} b_2^2 & -b_1 b_2 \\ -b_1 b_2 & b_1^2 \end{bmatrix} \tag{1.5}$$

S is symmetric, $\det(S(b)) = 0$ for all b and it satisfies $S(b)^k = S(b)$ for all positive integers k. The eigenvalues of S are either 0 or 1. The number of one-eigenvalues corresponds to the dimension of the vector space one is projecting into and the number of zero-eigenvalues corresponds to the dimension of the projected vector space of $(I - S)$. We have in addition that a non-trivial projection ($S \neq 0$) is diagonalizable with real eigenvectors since S is symmetric.

1.2.3 The model of a robot manipulator

A robot arm/manipulator is a kinematic chain of rigid bodies connected by joints. Joints come in two flavours and may either be rotational or translational. The model of a n-link robot manipulator can be derived via Lagrange's method and may be written on the standard form [10] p.290:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + B\dot{q} + G(q) = u \tag{1.6}$$

Where q is the vector of generalized joint coordinates that is either rotation or translation. M is the joint-space inertia matrix, C is the Coriolis and centrifugal force matrix, B is the friction term and G accounts for the gravity. u is the input vector of generalized forces corresponding to the generalized coordinates, either force or torque.

It is known that $\dot{M} - 2C$ is a skew symmetric matrix. C may be written in the following way:

$$C(q, \dot{q}) = \begin{bmatrix} \dot{q}^T C_1(q) \\ \vdots \\ \dot{q}^T C_6(q) \end{bmatrix} \tag{1.7}$$

Where $C_i \in \mathbb{R}^{n \times n}$ are symmetric matrices. We also know that $M \in \mathbb{R}^{n \times n}$ is symmetric positive definite and is norm bounded by constants.

Chapter 2

The Problem

We consider the following synchronization problem for two 6-DOF robot manipulators. One robot is the leader and the other is the follower. The robots are placed in a space containing obstacles with known shape and position. The leader is controlled by some unknown means such that no future reference information is available.

The follower's task is to move in synchrony with the leader in some predetermined way as well as to avoid collisions with itself, the leader and obstacles in the environment.

The control/synchronization problem pertaining to the follower will be studied in this thesis.

Definition 1 (State feedback synchronization with collision avoidance problem) *Consider a dynamical model of a 6-DOF robot manipulator given in the task space:*

$$\dot{x} = f(x, u) \tag{2.1}$$

Where $x \in \mathbb{R}^{12}$ is the state of the system and $u \in \mathbb{R}^6$ is the control input. Assume that a reference trajectory $x_r, \dot{x}_r$ is given. We also assume that there exist some collision free reference trajectory x_d^f and input u_f which satisfies $\dot{x} = f(x, x_r, u_r)$ such that x is collision free. Find an appropriate control law:

$$u = u(t, x_r, x) \tag{2.2}$$

Such that the closed loop system is asymptotically stable and collision free:

$$\lim_{t\to\infty} \|x(t) - x_d^f(t)\| = 0 \tag{2.3}$$

$$\lim_{t\to\infty} \|\dot{x}(t) - \dot{x}_d^f(t)\| = 0 \tag{2.4}$$

2.0.4 Assumptions

We make the following assumptions regarding our system:

We have full reference trajectory information up to the present time. The internal actuator dynamics in the joints are negligible. The robot is frictionless, or has known friction such that we may remove it with a controller. The robot it fully rigid, i.e. no joint flexibility. The geometry of the workspace is known.

2.1 Design considerations

We present two synchronization examples in order to illustrate the diversity of possible solutions to our problem.

Example: Dyssynchrony handling

Consider an example where our robot is for some reason in severe dyssynchrony, see figure 2.1.

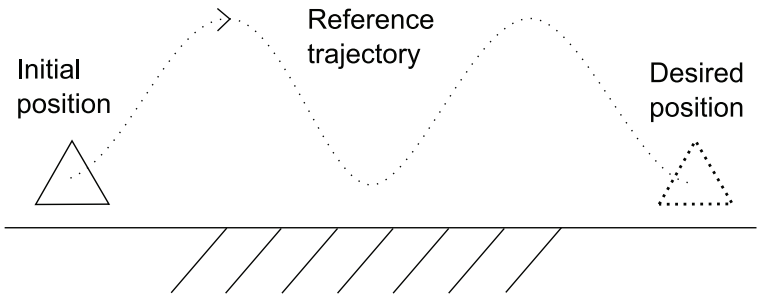
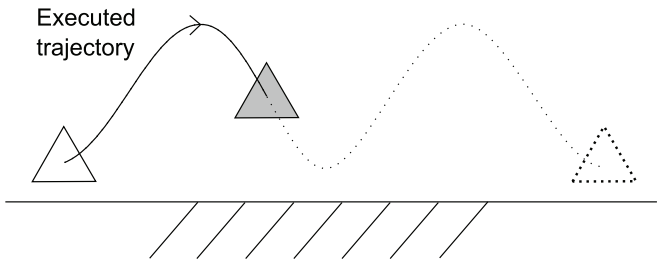
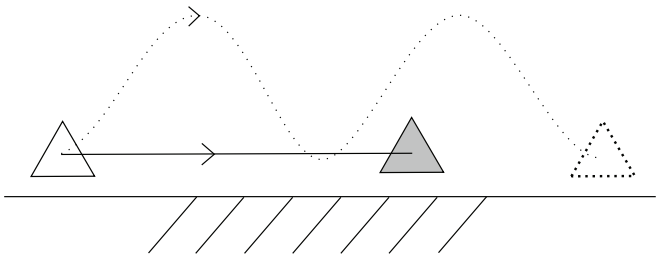


Figure 2.1: The robot, represented by a triangle, has a large synchronization error and the reference has moved along some trajectory

The question is how the controller should behave in order to catch up to the reference. Should it move along the reference trajectory faster than the reference to catch up? Or should it move as fast as possible towards the last available reference position? These cases are illustrated in figure 2.2.



(a) The robot moves, possibly faster than the reference, along the reference path.



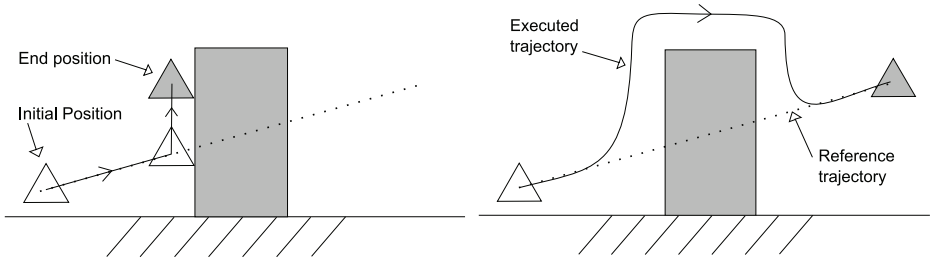
(b) The robot tries to minimize the current synchronization error as fast as possible.

Figure 2.2: Different handling of dissynchronization.

The difference is which synchronization error that gets penalized. If only the last available measurement is of interest, then figure 2.2b is the optimal solution. If the position error for all time is important then figure 2.2a would be an appropriate solution. Whether spatial or temporal errors are important depends on the task being carried out. Spatial errors would be more important for a welding-type robot than for a robot trying avoid a moving obstacle. A versatile design choice would seem to be a controller that weighs spatial and temporal errors arbitrarily. A "fast enough" control algorithm would nonetheless reduces the importance of this question.

Example: Collision avoidance design

Some action has to be taken when an impending collision is detected. One scenario is illustrated in figure 2.3 where the reference passes through an obstacle. The end effector is the triangle and the rest of the robot arm is suspended above the workspace.



(a) The controller stops moving in a direction that will lead to a collision.

(b) The controller finds a way around the obstacle.

Figure 2.3: Different ways of avoiding a collision when the end effector is given a reference passing through an obstacle.

Each choice for a control strategy has its advantages and downsides. The collision avoidance routine in scenario (a) is simple to implement, but results in a large synchronization error in the x-direction. The routine in scenario (b) gives a good steady state error, but is harder to implement. The end effector has to wait for a reference point that is collision free in order to proceed since no future reference information is available. And a trajectory then has to be found by some path planning or constrained nonlinear optimization algorithm. The result is a time period for which the synchronization error is large since this is computationally expensive. Consider a human leader who wants to operate on some thin plate and accidentally moves the reference point through the plate. The controller (a) will inhibit this motion while the controller (b) will try to move all the way around the plate. We therefore assume that some higher level controller produces a less pathological reference trajectory, and we chose the controller from scenario (a).

2.2 Overview of control design

We review two different controller implementations for the collision avoidance problem. One at the guidance level and one lower level integrated collision

avoidance controller.

2.2.1 Guidance level control design

The guidance level obstacle avoidance control scheme is traditionally implemented in three steps. First some collision free curve is generated considering only the geometry of the system using any path planning algorithm. This is done without considering any robot dynamics and may easily produce infeasible or undesirable curves. The second step is to solve a trajectory planning problem where a time parameterized trajectory is generated subject to certain constraints yielding a feasible trajectory. The last step is to solve the control problem where the objective is to follow the reference trajectory as closely as possible.

It is the job of the collision avoidance routine to produce a reference trajectory which, if perfectly tracked, will not lead to a collision while being close to the leader's trajectory with respect to some error measure. The collision problem is now removed from the controller since the reference trajectory is now collision free, see figure 2.4. Any sufficiently good synchronization controller may be implemented to follow the collision free reference trajectory.

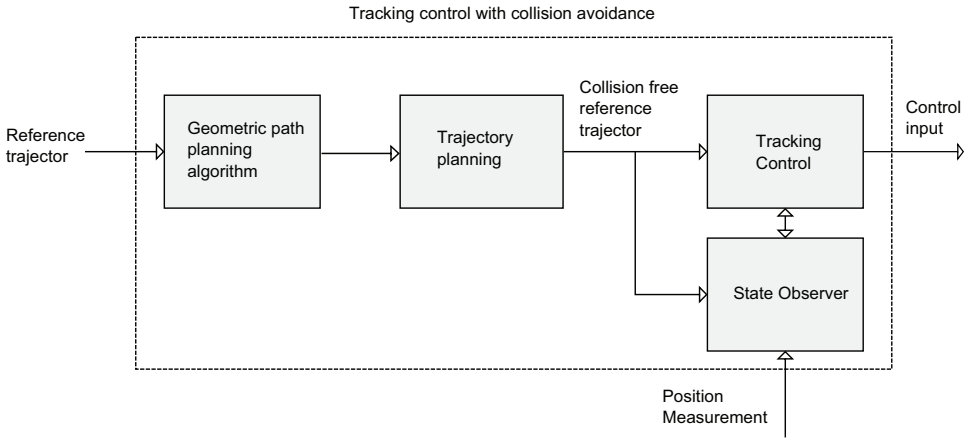


Figure 2.4: Box-diagram of the different modules in the tracking controller with obstacle avoidance at the guidance level.

The guidance level control design is attractive due to the simplicity arising from its modular nature and ease of implementation. One problem with this design is the following: We assuming that we have an algorithm that

takes reference trajectory samples as an argument, and returns collision free reference points. The collision avoidance problem is not yet solved even though this algorithm works to perfection. The reason for this is that the controller might not track this trajectory close enough, and deviations from the collision-free trajectory might produce collisions, see figure 2.6.

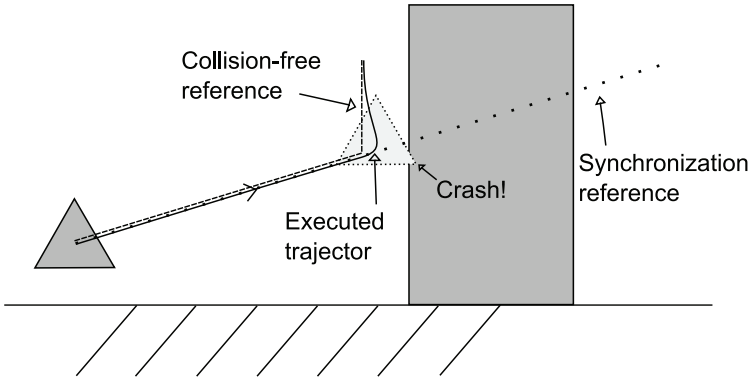


Figure 2.5: An example of how failing to follow the collision-free reference results in a collision.

We would need to implement a low level collision avoidance routine in addition to the path planning stage, to be certain that the robot does not crash. Path planning algorithms are generally slower than low level controllers and may be hard to implement for real time applications.

2.2.2 Control level collision avoidance design

The shortcomings of the open-loop trajectory generating algorithm may be avoided with a different control design. If we move the collision avoidance down to the synchronization controller level then we may add feedback control to both the synchronization controller and the collision avoidance controller. This effectively executes the three steps used in the guidance level control at once.

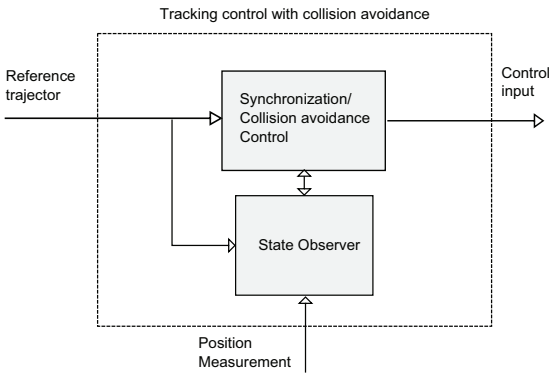


Figure 2.6: Box-diagram of the different modules in the tracking controller with integrated obstacle avoidance.

This can be thought of as the increased performance gained from the two controllers working together as opposed to working independently. This is an attractive feature since feedback on this level is suitable for on-line implementation as no slow path planning stage is required. The design will also be theoretically more interesting. We will for these reasons design our controller on this low level.

Chapter 3

Modeling

The robots dynamics is derived in this chapter. The analysis is text book standard and no new ideas or methods are presented. This chapter may be used as a guide to modeling a high degree of freedom revolute robot manipulator. The robot in question is seen in figure 3.1. Readers looking solely for obstacle avoidance theory may skip this chapter in its entirety. We need to perform the modeling in order to implement a simulator on which to test our controller, which is our motivation for including this chapter. We will first derive the forward kinematics and then the inverse kinematics. We will then present the robot model in full as well as derive som usefull functions needed for our controller. An overview of the methods we will use can be found in [10].

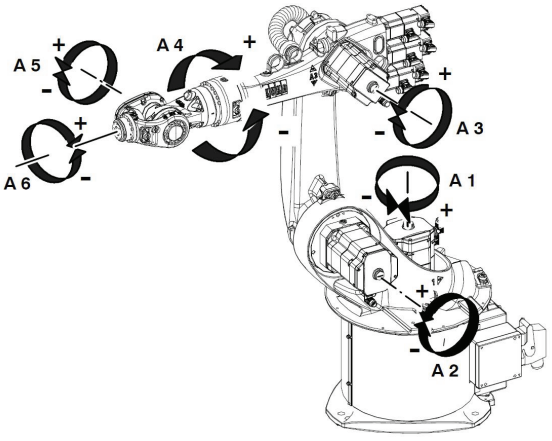


Figure 3.1: The KUKA KR16 robot with the rotational joints labeled $A_1 - A_6$ courtesy of KUKA.

3.1 Forward kinematics

The first step is to find the forward kinematics, i.e. the geometric relationship between the configuration of the robot and the position and orientation of the end effector. We derive this relationship using the DH-convention. The Denavit-Hartenberg (DH) convention is an algorithm used to derive the kinematics of a chain of rigid bodies using a small number of parameters, 4 instead of 6. This is achieved by cleverly choosing the coordinate axis for each joint. The joint diagram and appropriate coordinate axes is seen in figure 3.2.

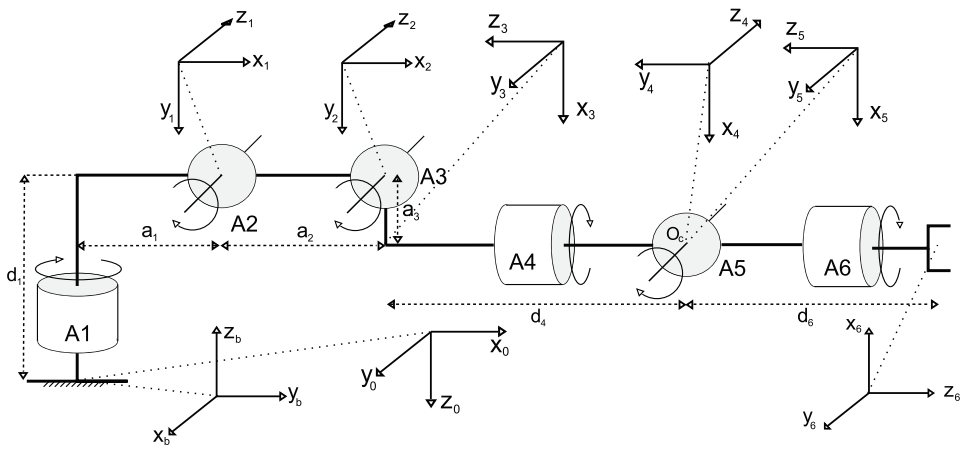


Figure 3.2: The joints of the robot labelled $A_1 - A_6$ with coordinate systems following the DH-convention. The arrows indicate the positive rotation direction. The lengths of the links are labelled a_i, d_i. The dotted lines indicate the placement of the coordinate axes.

We observe that the first three joints make up an elbow manipulator, and the last three are composed as a spherical wrist as defined in [10] p.87. We assign the right handed coordinate axis for each joint such that the z_i-vector is parallel with the rotation axis and the x_i-vector is perpendicular to z_i and z_{i-1}. The z vectors are placed such that the positive rotation angle is as it is defined for the robot from the manufacturer. The origin of each coordinate system is placed at the common normal between z_i and z_{i-1} and is arbitrary when they are parallel. The DH-parameters are given by the transformations from coordinate system i to $i + 1$ via the rotational and translational transformations given in (3.1)

	q	d	a	α
0	$\frac{\pi}{2}$	0	0	π
1	q_1	$-d_1$	a_1	$\frac{\pi}{2}$
2	q_2	0	a_2	0
3	$q_3 + \frac{\pi}{2}$	0	a_3	$-\frac{\pi}{2}$
4	q_4	$-d_4$	0	$\frac{\pi}{2}$
5	q_5	0	0	$-\frac{\pi}{2}$
6	$q_6 + \pi$	$-d_6$	0	π

Table 3.1: The DH-parameters for the KUKA-16K robot where $q_i = 0 \ \forall i$ results in the pose of figure 3.2. q_i are the joint variables.

$$T_i^{i+1} = \mathrm{Rot}_{z,q_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i} \tag{3.1}$$

Where Rot_{z,q_i} denotes a rotation q_i about z and Trans_{x,a_i} means a translation a_i along x. The parameters are given in table 3.1.

We get the homogeneous transformation from the base coordinate system to the end effector by multiplying the transformations in order:

$$T = T_b^6 = T_b^0 T_0^1 \cdots T_5^6 \tag{3.2}$$

We denote the base frame, or the world frame as b. The columns of forward kinematics homogeneous transformation is given by $T = [T_1, T_2, T_3, T_4]$ where $s_i = \sin(q_i)$,$c_i = \cos(q_i)$ and $c/s_{ij} = cos/sin(q_i + q_j)$.

$$T_1 = \begin{bmatrix} -c_6(c_5(c_1s_4 - s_{23}c_4s_1) - c_{23}s_1s_5) - s_6(c_1c_4 + s_{23}s_1s_4) \\ s_6(c_4s_1 - s_{23}c_1s_4) + c_6(c_5(s_1s_4 + s_{23}c_1c_4) + c_{23}c_1s_5) \\ -c_6(s_{23}s_5 - c_{23}c_4c_5) - c_{23}s_4s_6 \\ 0 \end{bmatrix} \tag{3.3}$$

$$T_2 = \begin{bmatrix} c_6(c_1c_4 + s_{23}s_1s_4) - s_6(c_5(c_1s_4 - s_{23}c_4s_1) - c_{23}s_1s_5) \\ s_6(c_5(s_1s_4 + s_{23}c_1c_4) + c_{23}c_1s_5) - c_6(c_4s_1 - s_{23}c_1s_4) \\ (s_{23}s_5 - c_{23}c_4c_5) \\ 0 \end{bmatrix} \tag{3.4}$$

$$T_3 = \begin{bmatrix} s_5(c_1s_4 - s_{23}c_4s_1) + c_{23}c_5s_1 \\ c_{23}c_1c_5 - s_5(s_1s_4 + s_{23}c_1c_4) \\ -s_{23}c_5 - c_{23}c_4s_5 \\ 0 \end{bmatrix} \tag{3.5}$$

$$T_4 = \begin{bmatrix} s_1(a_1 + d_4c_{23} - a_3s_{23} + a_2c_2) + d_6(s_5(c_1s_4 - s_{23}c_4s_1) + c_{23}c_5s_1) \\ c_1(a_1 + d_4c_{23} - a_3s_{23} + a_2c_2) - d_6(s_5(s_1s_4 + s_{23}c_1c_4) - c_{23}c_1c_5) \\ d_1 - a_3c_{23} - d_4s_{23} - a_2s_2 - d_6(s_{23}c_5 + c_{23}c_4s_5) \\ 1 \end{bmatrix} \tag{3.6}$$

3.2 Inverse kinematics

We derive the inverse kinematics of the robot in this section. This is a necessity if one chooses to employ joint space control as one needs to map the desired end effector trajectory to a trajectory given in the joint space. It may also be used when performing task space control to provide information about desirable poses. The inverse kinematics is the mapping from the end effector to the joint space:

$$q = T^{-1}(o, R) \tag{3.7}$$

Where $o = [x, y, z]^T$ is the position of the end effector given in the base frame and R is a rotation matrix describing the orientation of the end effector. The problem of determining the inverse kinematics is in general a nontrivial problem of solving 12 highly nonlinear equations with 6 unknowns. The problem is however not that hard given the specific robot configuration from figure 3.2 and forward kinematics in accordance with the DH-convention, [10] p.96. This stems from the fact that the three last joints make up a spherical wrist where the coordinate axes (4,5) have the same origin called the wrist center, o_c. We use this to split the inverse kinematics problem in two, a procedure that is called Kinematic decoupling.

The location of the wrist center with respect to the base frame is uniquely given by:

$$o_c = o - d_6R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

This allows us to first find $q_{1,2,3}$ using o_c, and then finding the last 3 using the orientation of the end effector.

3.2.1 The first three joint angles

We assume that the robot is in a nonsingular configuration. $o_c^T = [x_c, y_c, z_c]^T$ is the wrist center given in the base frame, using the translations from the base to o_c we see that:

$$o_c = \begin{bmatrix} s_1(a_1 + a_2c_2 + c_{23}d_4 - a_3s_{23}) \\ c_1(a_1 + a_2c_2 + c_23d_4 - a_3s_{23}) \\ d_1 - a_3c_{23} - a_2s_2 - d_4s_{23} \end{bmatrix} \tag{3.9}$$

We find q_1 easily since since the location of o_c in the x_by_b-plane is a function of q_1, i.e.:

$$\frac{x_c}{y_c} = \tan q_1 \tag{3.10}$$

This gives us the two possible solutions for q_1:

$$\{q_{11}, q_{12}\} = \{\text{Atan2}(x_c, y_c), \text{Atan2}(x_c, y_c) + \pi\}\,; \tag{3.11}$$

Where Atan2 $\in [0, 2\pi)$ is the four quadrant arctangent function. There are two possible values for q_1, these refer to the right arm or left arm configurations.

To find the second two angles we first rotate the robot into the y_bz_b-plane since we know q_1:

$$o_{2_{\text{rot}}} = \begin{bmatrix} 0 \\ a_1 \\ d_1 \end{bmatrix} \quad o_{c_{\text{rot}}} = Rot_{z_b,q_1} o_c = \begin{bmatrix} x_c\cos(q_1) - y_c\sin(q_1) \\ y_c\cos(q_1) + x_c\sin(q_1) \\ z_c \end{bmatrix} = \begin{bmatrix} 0 \\ o_{c1} \\ o_{c2} \end{bmatrix} \tag{3.12}$$

We find q_3 by observing that the distance between o_2 and o_c is a function of q_3:

$$d = \|o_c - o_2\|^4 = a_2^2 + a_3^2 + d_4^2 - 2a_2a_3\sin(q_3) + 2a_2d_4\cos(q_3) \tag{3.13}$$

The fourth power of the distance is used since the result is slightly simpler. This gives us the two solutions for q_3:

$$\{q_{31}, q_{32}\} = \left\{ -2\text{atan}\left(\frac{a_3 + \sqrt{a_3^2 + d_4^2 - D^2}}{d_4 + D}\right), -2\text{atan}\left(\frac{a_3 - \sqrt{a_3^2 + d_4^2 - D^2}}{d_4 + D}\right) \right\} \tag{3.14}$$

Where $D = (d - a_2^2 - a_3^2 - d_4^2)/(2a_2)$. If the plus sign is chosen, then we get the q_3 angle corresponding to $q_{1,1}$. If $\Delta_i = a_3^2 + d_4^2 - D_i^2 < 0$ then we have no solutions for the given q_{1i}.

The solution for q_2 is an exercise in trigonometry. The equations one needs to solve is the wrist center position as a function of q_2 and is given by:

$$\begin{bmatrix} o_{c1} \\ o_{c2} \end{bmatrix} = \begin{bmatrix} a_1 + d_4 c_{23} - a_3 s_{23} + a_2 c_2 \\ d_1 - a_3 c_{23} - d_4 s_{23} - a_2 s_2 \end{bmatrix} \tag{3.15}$$

It's solution is:

$$q_{2k} = \text{Atan2}(o_{c1} - a_1, o_{c2} - d_1) - \text{Atan2}\left[a_2 + \delta\cos(q_{3k}^{'}), -\delta\sin(q_{3k}^{'})\right] \tag{3.16}$$

Where $\delta = \sqrt{a_3^2 + d_4^2}$, $q_{3k}^{'} = q_{3k} + \text{atan}\left(\frac{a_3}{d_4}\right)$ and $k \in \{1, 2\}$ refers to the different solutions possible for q_3.

The solution set now looks like:

$$Q_1 = \Big\{\{q_{11}, q_{21}, q_{31}\}, \{q_{11}, q_{22}, q_{32}\}, \{q_{12}, q_{21}, q_{31}\}, \{q_{12}, q_{22}, q_{32}\}\Big\} \tag{3.17}$$

$$Q_2 = \Big\{\{q_{11}, q_{21}, q_{31}\}, \{q_{11}, q_{22}, q_{32}\}\Big\} \tag{3.18}$$

$$Q_3 = \Big\{\{q_{12}, q_{21}, q_{31}\}, \{q_{12}, q_{22}, q_{32}\}\Big\} \tag{3.19}$$

$$\{q_1, q_2, q_3\} \in \begin{cases} Q_1 & \text{if} \quad \Delta_{1,2} > 0 \\ Q_2 & \text{if} \quad \Delta_1 > 0, \Delta_2 < 0 \\ Q_3 & \text{if} \quad \Delta_1 < 0, \Delta_2 > 0 \\ \emptyset & \text{if} \quad \Delta_{1,2} < 0 \end{cases}$$

Where D_i corresponds to q_{1i} and the angles within a bracket depends upon each other in the following way:

$$\{q_{11}, q_{21}, q_{31}\} \triangleq \{q_{11}, q_{21}(q_{11}, q_{31}), q_{31}(q_{11})\} \tag{3.20}$$

We see that there are either zero,two or four solution for the first three angles. They correspond to the permutations of the elbow-up/down, arm right/left robot configurations as defined in [10] p.104.

3.2.2 The last three joint angles

The last three joint angles are the Euler parameter for a given rotation matrix. We may construct the rotation matrix from the wrist center to the end effector as a function of $q_{4,5,6}$ since we now know the first three joint angles. This is R_6^3:

$$R_6^3 = R_3^0 R = r_{ij} \tag{3.21}$$

We know from the forward kinematics that:

$$R_6^3 = \begin{bmatrix} s_4 s_6 - c_4 c_5 c_6 & -c_6 s_4 - c_4 c_5 s_6 & c_4 s_5 \\ -c_4 s_6 - c_5 c_6 s_4 & c_4 c_6 - c_5 s_4 s_6 & s_4 s_5 \\ -c_6 s_5 & -s_5 s_6 & -c_5 \end{bmatrix} \tag{3.22}$$

This gives us the solutions for the last angles:

$$\{q_{41}, q_{42}\} = \{\text{Atan2}(r_{23}, r_{13}), \text{Atan2}(r_{23}, r_{13}) + \pi\} \tag{3.23}$$
$$\{q_{51}, q_{52}\} = \{\text{acos}(r_{33}), -\text{acos}(r_{33})\} \tag{3.24}$$
$$\{q_{61}, q_{62}\} = \{\text{Atan2}(-r_{32}, -r_{31}), \text{Atan2}(-r_{32}, -r_{31}) + \pi\} \tag{3.25}$$

With the solution set:

$$\{q_4, q_5, q_6\} \in \Big\{\{q_{41}, q_{51}, q_{61}\}, \{q_{42}, q_{52}, q_{62}\}\Big\} \tag{3.26}$$

There are two solutions to this problem given a triplet $q_{1,2,3}$ and a nonsingular configuration. So we either have zero, four or eight solutions for the nonsingular inverse kinematics.

3.2.3 Singularities of the inverse kinematics

A singular configuration of the robot is the case where there are an infinite number of solutions to the inverse kinematics problem. Since we can calculate the first three angles independently of the last three, we know that the singularities for the first three joints are the same ones as for an elbow manipulator. This is when the wrist center is on the z_b axis such that $o_c^T = [0, 0, o_{cz}]$. q_1 hence becomes a free variable. The solution we use here is to pick the last known value for q_1 and use this to calculate the other angles as before.

The spherical joint is in a singular configuration without self intersection when $q_5 = 0$. The matrix R_6^3 now has the form:

$$R_6^3 = \begin{bmatrix} -c_{46} & -s_{46} & 0 \\ -s_{46} & c_{46} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.27}$$

As before we have a case where either q_4 or q_6 is a free variable. The solution is to assume that q_4 is the same as the last known value, and then get q_6 from the expression:

$$q_4 = \text{Atan2}(-r_{12}, -r_{11}); \tag{3.28}$$

3.3 Velocity kinematics

We consider the velocity kinematics which is the next step in the modeling procedure. The velocity kinematics describe the linear and angular velocity of points on the robot as a function of the joint velocities. The velocity is described by the Jacobian of the forward kinematics, and we specifically need the velocity of the mass centers of the links.

3.3.1 Positions of the mass centers

We obtain the position of the mass centers using the forward kinematics. We assume that the mass density in the joints are sufficiently uniform such that the mass center will be located somewhere along the line between two joints. We call these lengths r_i, and we readily compute the mass centers given in the base coordinate system:

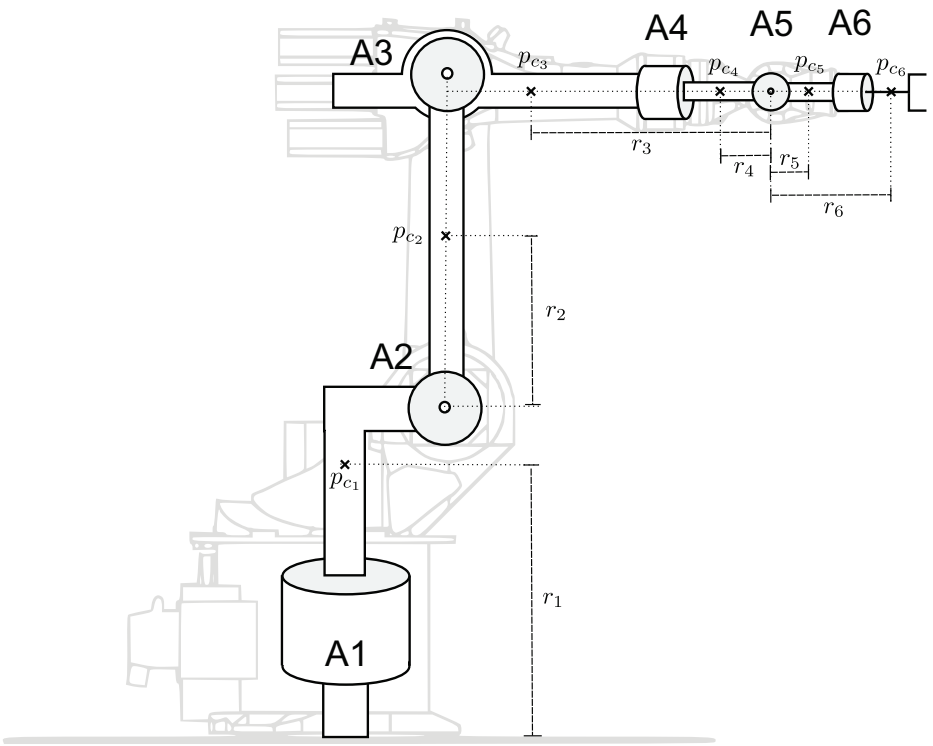


Figure 3.3: The KUKA robot with the centers of gravity labeled p_{c_i} for each link.

$$p^b_{c_1} = \begin{bmatrix} 0 \\ 0 \\ r_1 \end{bmatrix} \qquad p^b_{c_2} = o_1 + R^b_2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} r_2$$

$$p^b_{c_3} = o_5 + R^b_4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} r_3 \qquad p^b_{c_4} = o_5 + R^b_4 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} r_4$$

$$p^b_{c_5} = o_5 + R^b_5 \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} r_5 \qquad p^b_{c_6} = o_5 + R^b_5 \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} r_6$$

where

$$R^b_k = R^b_0 R^0_1 \cdots R^{k-1}_k, \quad T^b_k = \begin{bmatrix} R^b_k & o^b_k \\ 0 & 1 \end{bmatrix} \tag{3.29}$$

We note that the mass centers p_3, p_4 and p_5, p_6 are located on the same line.

3.3.2 The Jacobian

The Jacobian J is the mapping from the joint velocities to the linear and angular velocities of any given point on the robot. The Jacobian derived in this section is also referred to as the geometric Jacobian in order to separate it from the analytic Jacobian which we will not use. The relationship is the following:

$$v_i^b = J_{v_i}\dot{q}, \quad \omega_i^b = J_{\omega_i}\dot{q} \tag{3.30}$$

Where $\omega, v \in \mathbb{R}^{3\times 1}$ are respectively the linear and angular velocities and $J_{\omega_i}, J_{v_i} \in \mathbb{R}^{3\times 6}$. We get the Jacobian for the linear velocities of the mass centers either by differentiation of the mass centers with respect to time.

$$v_i^b = \dot{p}_{c_i}^b = J_{v_i}\dot{q} \quad \Rightarrow \quad J_{v_i} = \frac{\partial p_{c_i}^b}{\partial q} \tag{3.31}$$

Where $q = [q_1, \ldots, q_6]^T$. Alternatively we may use the handy formula from [10] p.133 which holds true for a *fixed* point p on the robot with revolute joints:

$$J_p(q) = \begin{bmatrix} z_0 \times (p - o_0) & z_0 \times (p - o_1) & \ldots z_0 \times (p - o_5) \end{bmatrix} \tag{3.32}$$

Where the row i of J_p is the zero vector if p is not a function of q_i. From [10] p.133 we have the following Jacobian for the angular velocities given revolute joints using the DH-convention:

$$J_{\omega_i} = \begin{bmatrix} z_0^b u(i-1) & z_1^b u(i-2) & z_2^b u(i-3) & \cdots & z_6^b u(i-6) \end{bmatrix} \tag{3.33}$$

Where u is the Heaviside function, $u(x) = 1 \,\forall\, x \geq 0, u(x) = 0 \,\forall\, x < 0$ and z_i^b is the z vector of the coordinate system i expressed in the base frame. We have for instance that:

$$J_{\omega_3} = \begin{bmatrix} z_0^b & z_1^b & z_2^b & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} R_0^b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_1^b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & R_2^b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

The Jacobian for the first two mass centers are as follows:

$$J_1 = \begin{bmatrix} J_{v_1} \\ J_{\omega_1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.35}$$

$$J_2 = \begin{bmatrix} J_{v_2} \\ J_{\omega_2} \end{bmatrix} = \begin{bmatrix} c_1(a_1 + r_2c_2) & -r_2s_1s_2 & 0 & 0 & 0 & 0 \\ -s_1(a_1 + r_2c_2) & -r_2c_1s_2 & 0 & 0 & 0 & 0 \\ 0 & -r_2c_2 & 0 & 0 & 0 & 0 \\ 0 & -c_1 & 0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.36}$$

We see that the linear velocity of the first mass center is zero and that $\omega_1 = -\dot{q}_1$ as is apparent by inspecting figure 3.3.

3.4 The singularities of the Jacobian

The Jacobian does not in general have full rank. The angles where J looses rank are called singularities and are of interest as a task space synchronization controller uses the inverse of the Jacobian. These singularities are the same as for the inverse kinematics as the nonsingularity of the Jacobian is a necessary and sufficient condition for the forward kinematics to be a diffeomorphism. These joint angle values have a physical interpretation as the robot looses one or more degrees of freedom at a singularity. This happens for instance when the robot is stretched out fully in one direction, and the end effector is on the boundary of the reachable workspace.

We present the determinant of the jacobian for inspection as we have allready considered the singularities in view of the inverse kinematics.

$$\det\{J\} = a_2s_5\left[a_3c_3 + d_4s_3\right]\left[a_1 + d_4c_{23} - a_3s_{23} + a_2c_2\right] \tag{3.37}$$

3.5 The joint space dynamical model

We are now ready to state the dynamical robot model given in the joint space. From [10] p.254 we have that the mass matrix M in equation (1.6) is given by:

$$M(q) = \sum_{i=1}^{6} \left\{ m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i^b(q) I_i R_i^b(q)^T J_{\omega_i}(q) \right\} \tag{3.38}$$

Where m_i are the mass of link i and I_i is the inertia matrix for link i about the mass center expressed in the body attached frame. We see from figure 3.2 that he body attached frames to the mass centers $[p_{c_1}, p_{c_2}, p_{c_3}, p_{c_4}, p_{c_5}, p_{c_6}]$ are in order $[R_1^b, R_2^b, R_3^b, R_4^b, R_5^b, R_6^b]$. We assume that the inertia matrices for the last three joints are diagonal since the last three joint are small compared to the first three, i.e:

$$I_k = \begin{bmatrix} I_{k_{xx}} & I_{k_{xy}} & I_{k_{xz}} \\ I_{k_{xy}} & I_{k_{yy}} & I_{k_{yz}} \\ I_{k_{xz}} & I_{k_{yz}} & I_{k_{zz}} \end{bmatrix} \forall\, k \in \{1,2,3\}, \quad I_k = \begin{bmatrix} I_{k_{xx}} & 0 & 0 \\ 0 & I_{k_{yy}} & 0 \\ 0 & 0 & I_{k_{zz}} \end{bmatrix} \forall\, k \in \{4,5,6\} \tag{3.39}$$

The full mass matrix is rather large, and we would need about 3 pages to write it out explicitly assuming one page has 50 lines.

The Coriolis matrix C is given by:

$$C(q,\dot{q}) = \begin{bmatrix} \dot{q}^T C_1(q) \\ \vdots \\ \dot{q}^T C_6(q) \end{bmatrix} \tag{3.40}$$

Where the elements of C_k are given by:

$$C_{k_{(i,j)}} = \frac{1}{2} \left\{ \frac{\partial M_{(k,j)}}{\partial q_i} + \frac{\partial M_{(k,i)}}{\partial q_j} - \frac{\partial M_{(k,i)}}{\partial q_k} \right\} \tag{3.41}$$

The Coriolis matrix is even larger than M and we would need about 5 pages to write it down. We note that both M and C are independent of q_1. This is justified by observing that the robots kinetic energy is independent of its base angle.

The gravity vector $G(q)$ is defined as:

$$G(q) = \frac{\partial P}{\partial q} \tag{3.42}$$

Where P is the potential energy of the robot given in the world frame as:

$$P_i = m_i g p^b_{c_1 z} \tag{3.43}$$

Where $p^b_{c_1 z}$ is the height of the centers of gravity for each link.

Some of the shortest elements of the system matrices are given by:

$$\begin{aligned}
M_{(5,6)} &= 0 \\
M_{(4,6)} &= I_{6_{zz}} \cos(q_5) \\
M_{(3,6)} &= M_{(2,6)} = I_{6_{zz}} \sin(q_4) \sin(q_5) \\
M_{(4,5)} &= -\cos(q_6) \sin(q_5) \sin(q_6)(I_{6_{xx}} - I_{6_{yy}}) \\
C_{(5,5)} &= \frac{1}{2} \dot{q}_6 \sin(2q_6)(I_{6_{xx}} - I_{6_{yy}}) \\
C_{(6,6)} &= 0
\end{aligned}$$

3.6 Task space mapping

In order to express our dynamical robot model in the task space, we need to do a little more work. We will need to find a representation of the orientation of the end effector using a $\mathbb{R}^{3\times 1}$ vector. We also need the derivative of the Jacobian with respect to time. We will not explicitly derive the task space dynamics as is common using the *Analytic Jacobian* [10] p.140. We will later use the roll pitch yaw angle error in place of the actual roll pitch yaw angles for the end effector when performing this mapping. We will see later why this is advantageous. Writing down an analytic Jacobian for the orientation error dynamics is possible, but one needs a matrix with a translational mapping to achieve this. This will be the final step of our lengthy modeling exercise.

3.6.1 A minimal representation $SO(3)$

We need to use a minimal representation of $SO(3)$ in order to find a robot model that is useful for task space control. The orientation information is given as a rotation matrix, which is subject to three normality constraints and three orthogonality constraints. So expressing an orientation using a rotational matrix is highly redundant as only three degrees of freedom are left after accounting for the constraints. Representing $SO(3)$ by three parameters may be done in a number of ways, but sadly there are no nonsingular minimal representations of $SO(3)$. That is, there exsists no mappings from $R \in SO(3) \mapsto \Phi \in \mathbb{R}^{3\times 1}$ such that the mapping and its inverse are defined for all $R \in SO(3)$ for both position and velocity. A nonsingular representation is however possible using quaternions, but a quaternion is not a minimal representation and using quaternions will result in nonlinear closed loop system. Commonly used minimal representations are axis/angle and Euler angles. We will use Euler angles, and specifically roll-pitch-yaw angles to parametrize orientation. We choose this parametrization since it is intuitive, easily implementable, and it produces a linear orientation error system. Used in a certain way it is also possible to place the singularity 90° away from the desired orientation.

The roll pitch yaw angles are defined as the ZYX Euler parameters ϕ, θ, ψ such that:

$$R = R_z(\phi)R_y(\theta)R_x(\psi) \tag{3.44}$$

Where $R_z(\phi)$ denotes a rotational about the z-axis with ϕ radians. Given a rotational matrix R, it is possible to extract the roll pitch yaw angles with:

$$\begin{aligned} \psi &= \text{Atan2}(R_{32}, R_{33}) \\ \phi &= \text{Atan2}(R_{21}, R_{11}) \\ \theta &= \text{Atan2}(-R_{31}, \cos(\phi)R_{11} + \sin(\phi)R_{21}) \end{aligned} \tag{3.45}$$

Where Atan2 is the two argument arcatan function. It is possible to find a linear relationship between the velocity of the Euler parameters and the angular velocity of a rotational matrix by differentiating the Euler parameters with respect to time. If we denote $\Phi = [\phi, \theta, \psi]^T$ as the vector consisting of the roll pitch yaw angles, and the angular velocity ω given by the relationship between the joint velocity and the geometric Jacobian $\omega = J_\omega \dot{q}$ then we have:

$$\omega = B\dot{\Phi} = \begin{bmatrix} 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 1 & 0 & -\sin(\theta) \end{bmatrix} \dot{\Phi} \tag{3.46}$$

3.6.2 The time derivative of the Jacobian

We will need the time derivative of the Jacobian in order to achieve a mapping from the joint space to the task space. This motivates us to find a closed form expression of the time derivative of the Jacobian as it is needed for our controller.

We recall that J consists of two matrices J_{v_p} and J_{ω_p} respectively mapping the joint velocities to the linear velocity of a fixed point p on the robot and its angular velocity. We assume that p is some point on the robot, the i'th column of J_{v_p} is either given by $z_i(q) \times (p(q) - o_i(q))$ or it is the zero vector. The i'th column of J_ω is given by $z_i(q)$ or zero.

The time derivative of J may be taken column wise, so we may without loss of generality consider a column of J_ω:

$$\frac{d}{dt} z_i = \omega_i \times z_i = J_{\omega_i}\dot{q} \times z_i \tag{3.47}$$

The result is straightforward to derive, and the derivation identities used may be found in [5] p.243. The derivative of a column of J_{v_p} is given by:

$$\Big[z_i(q)\times(p(q)-o_i(q))\Big] = \frac{d}{dt} z_i(q)\times(p(q)-o_i(q)) + z_i(q)\times\frac{d}{dt}(p(q)-o_i(q)) \tag{3.48}$$

Where we have used the fact that the cross product obeys the product rule. Substituting in $\dot{z}_i$, $\dot{q} = J_{v_q}\dot{q}$ and $\dot{o}_i = J_{v_{o_i}}\dot{q}$ gives:

$$\frac{d}{dt}\Big[z_i(q) \times (p(q) - o_i(q))\Big] = [(J_{\omega_i}\dot{q}) \times z_i] \times (p - o_i) + z_i \times [(J_{v_p} - J_{o_i})\dot{q}] \tag{3.49}$$

Where the k'th column of J_{v_p} and J_{o_i} are given by:

$$\mathrm{Col}_k\{J_{v_p}\} = z_k \times (p - o_k) \tag{3.50}$$
$$\mathrm{Col}_k\{J_{o_i}\} = z_k \times (o_i - o_k) \tag{3.51}$$

And are zero for $k > i$. All these expressions are given as explicit functions of q and $\dot{q}$ which make them suitable for on-line implementation.

Chapter 4

Geometry Representation and Distance Calculation

Finding the distance from the robot to an obstacle in real time is the subject of this chapter. The collision avoidance strategy we will implement uses robot to obstacle distance information. If the robot had distance sensors, then this section would be unnecessary. Our system however uses information obtained from cameras to generate data about the workspace obstacles. This obstacle representation was used in [9] together with a potential functions for path planing in joint space. It is as far as the author knows the first time the representation is utilized for real time distance estimation. Our shortest distance solution, however trivial it may be, has not been found in any publication.

4.1 Obstacle representation

We need to establish a framework for representing geometry which is both efficient and notationally manageable. In this section we will present our choice of obstacle representation, and see why it is a suitable choice for on-line distance estimation. It is common, especially in computer games to represent geometry as polytopes consisting of a set of vertices and normal vectors. While this is an approach where collision detection software is available, it is notationally hard to incorporate this into a dynamical system as the data is represented by a big array of ordered vectors. We will instead use a completely general implicit representation of an obstacle by a scalar function of the spatial coordinates.

An implicit representation of an object in $\mathbb{R}^n$ given by $\beta_i(x) : \mathbb{R}^n \mapsto \mathbb{R}$. We denote the interior of obstacle i as $\mathbb{R}^n$ as $\mathcal{I}_i$, the boundary as $\mathcal{B}_i$ and the exterior as $\mathcal{E}_i$. An implicit representation $\beta_i(x)$ satisfies the following properties:

$$\begin{aligned} \beta_i(x) < 0 \quad \forall \quad x \in \mathcal{I}_i \\ \beta_i(x) = 0 \quad \forall \quad x \in \mathcal{B}_i \\ \beta_i(x) > 0 \quad \forall \quad x \in \mathcal{E}_i \end{aligned} \tag{4.1}$$

We note that there are an infinite number of implicit representations of any given obstacle. For a sphere in $\mathbb{R}^3$ we see that $\beta(x) = x_1^2 + x_2^2 + x_3^2 - r^2$ is correct implicit representation, and it is unique up to a constant multiple. Using this method it is easy to check whether a given point is inside, on the boundary of, or outside an obstacle. The normal vector pointing out of the obstacle, which is convenient for a number of purposes is the gradient evaluated at the boundary.

Implicit representations of shapes with a boundary defined as a solution to a particular equation are readily given for simple geometric shapes like sphere and ellipses. These basic shapes are nice due to their simplicity, and may be used for a rough representation of the workspace. Their expressiveness is however rather limited.

We shortly review how implicit representations for general shapes may be obtained. For obstacles with piecewise continuous boundary we may use the following representation from [11]. [11] presents a union and a intersection function of k functions satisfying (4.1):

$$Z_1(\beta_1(x), \beta_2(x), ..., \beta_k(x)) = \beta_1(x) \cap \beta_2(x) \cap ... \cap \beta_k(x) \tag{4.2}$$

$$Z_2(\beta_1(x), \beta_2(x), ..., \beta_k(x)) = \beta_1(x) \cup \beta_2(x) \cup ... \cup \beta_k(x) \tag{4.3}$$

Where $Z_{1,2}$ may be recursively defined for as:

$$Z_{1,2}(\beta_1, \beta_2, ..., \beta_k) = Z_{1,2}(\beta_1, Z_{1,2}(\beta_2, ..., \beta_k)) \tag{4.4}$$

With

$$Z_1(\beta_1, \beta_2) = \beta_1 + \beta_2 + \|\beta_1 - \beta_2\| \tag{4.5}$$

$$Z_2(\beta_1, \beta_2) = \beta_1 + \beta_2 - \|\beta_1 - \beta_2\| \tag{4.6}$$

To shortly motivate why these functions are appropriate we note that the intersection function Z_1 is zero if and only if one of its arguments are zero and the other is negative, or both its arguments are zero. If one of its arguments are zero and one is positive, the Z_1 is positive. When both arguments are of the same sign Z_1 is has the same sign as its arguments. Thus Z_1 satisfies the properties of an implicit representation of a the intersection of its two arguments.

These functions can be used to implicitly represent any shape in $\mathbb{R}^n$ as a union and intersection of other implicit shapes with implicitly given boundary. We present two examples to show their usage. The functions presented in [11] are given as a lengthy expression of all k arguments. We will use just the functions Z_1 and Z_2 composed by each other. This results in shorter expressions and less computational expense than the form presented in [11].

Example 1

A cube in $(x, y) \in \mathbb{R}^2$ placed at the origin with unit sides is given by the intersection of following inequalities:

$$\beta_1 = |x| - \frac{1}{2}$$
$$\beta_2 = |y| - \frac{1}{2}$$

Where $|x|$ is the absolute value. We may also have used $\beta_1 = x^2 - \frac{1}{2}^2$ for the same result. Using Z_1 to obtain the intersection we get:

$$\beta_1 \cap \beta_2 = Z_1(\beta_1, \beta_2) = |x| + |y| + ||x| - |y|| - 1$$

The boundary of the cube is given by $Z_1(\beta_1, \beta_2) = 0$, see figure 4.1

A surface plot of Z_1 is shown in figure 4.2 where we see that Z_1 is negative inside the boundary of the box we constructed.

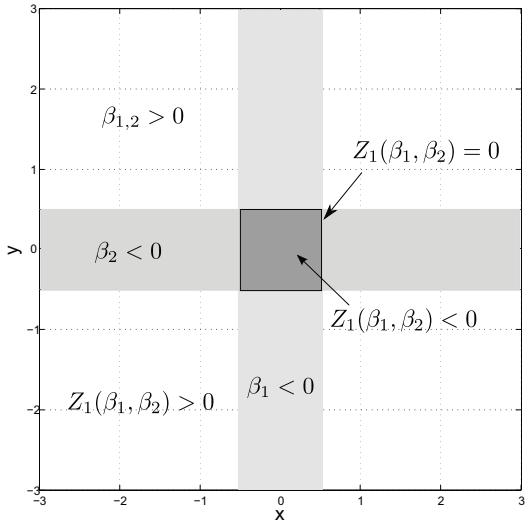


Figure 4.1: A cube in $\mathbb{R}^2$ given as the intersection of two inequalities.

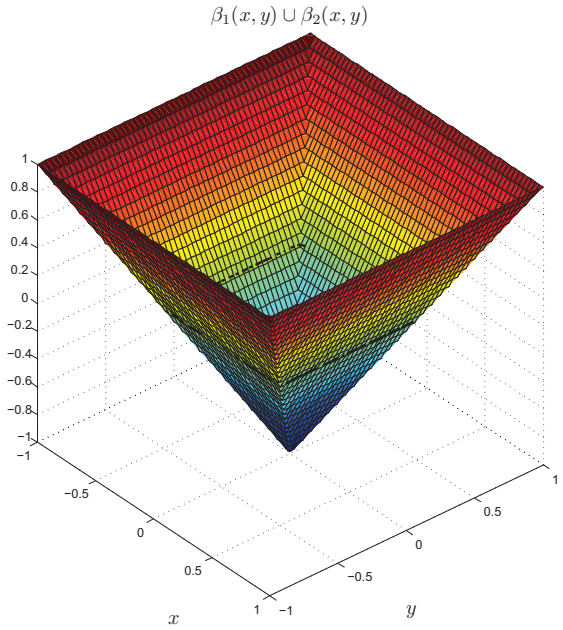


Figure 4.2: The function Z_1 describing a cube in $\mathbb{R}^2$.

Example 2

We present another example to illustrate the expressiveness of this method. Two boxes and four unit circles in $\mathbb{R}^2$ are represented by the inequalities:

$$\begin{aligned}
\beta_1 &= |x| - 2, \quad \beta_2 = |y| - \frac{1}{2} \\
\beta_3 &= |x| - \frac{1}{2}, \quad \beta_4 = |y| - 2 \\
\beta_5 &= (x-2)^2 + y^2 - 1 \\
\beta_6 &= (x+2)^2 + y^2 - 1 \\
\beta_7 &= x^2 + (y-2)^2 - 1 \\
\beta_8 &= x^2 + (y+2)^2 - 1
\end{aligned}$$

The operation $Z_2(Z_1(\beta_1, \beta_2), Z_1(\beta_3, \beta_4))$ takes the union of the two rectangles and constructs a cross. The union of this cross and the circles gives the shape seen in 4.3.

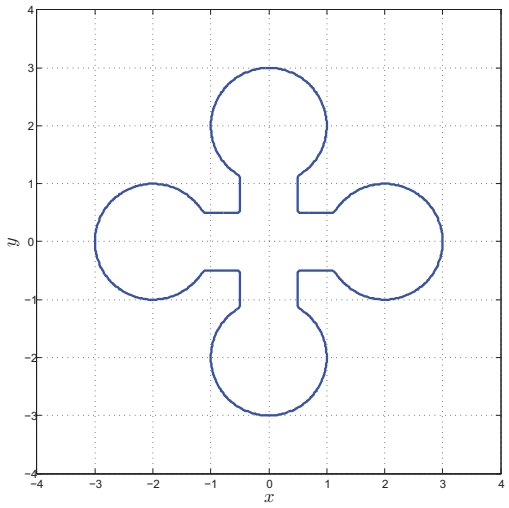


Figure 4.3: A strange cross in $\mathbb{R}^2$.

In implicit representation of the cross and the circles is given by:

$$Z_2(Z_1(\beta_1, \beta_2), Z_1(\beta_3, \beta_4), \beta_5, \beta_6, \beta_7, \beta_8) = (\beta_1 \cap \beta_2) \cup (\beta_3 \cap \beta_4) \cup \beta_5 \cup \beta_6 \cup \beta_7 \cup \beta_8 \tag{4.7}$$

Z_2 is negative in the interior of the strange cross, positive outside and zero if and only if (x, y) is on its boundary. The closed form expression is lengthy, the expression for the cross without the circles is:

$$\begin{aligned} Z_2(Z_1(\beta_1, \beta_2), Z_1(\beta_3, \beta_4)) = &||x| - |y| - 3/2| - |||x| - |y| + 3/2| \\ &-||x| - |y| - 3/2|| + ||x| - |y| + 3/2| + 2|x| + 2|y| - 5 \end{aligned} \tag{4.8}$$

4.2 Shortest distance between a convex shape and a line

Distance estimation between our robot and obstacles in the workspace is needed for our obstacle avoidance strategy to be successful. We need the shortest robot to obstacle distance in order to implement the obstacle avoidance potential function from [10]. This is straightforward for spherical obstacles, but this may be to restrictive for an accurate representation of the workspace obstacles. In this section we discuss an approach for the solution to the distance estimation problem for general obstacles. We finish the analysis with an example.

We assume that all obstacles are convex, or are representable as a set of possibly overlapping convex figures. We also assume that the robot is shrunk to a connected link of line segments, see figure 4.4. This is appropriate if we enlarge the workspace obstacle proportionally.

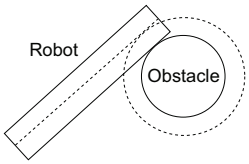


Figure 4.4: Illustration of how collision is invariant under proportionate rescaling of obstacle and robot sizes.

We will assume throughout the analysis that the gradient of Z is continuous for all points in $\mathbb{R}^n$. This is not true for the box from example 1 as the normal vector changes discontinuously at a corner. This did not present a problem in simulation however, but applying theory of solutions to vector fields with discontinuous right hand side is beyond the scope of this chapter.

The problem is to find the shortest distance from each link to each obstacle in real time for moving links and obstacles. To that end, we consider an arbitrary line segment $o_i, o_{i+1} \in \{x_1, x_2, x_3\}$ and an arbitrary convex shape given implicitly in the workspace as $Z(x) \leq 0$.

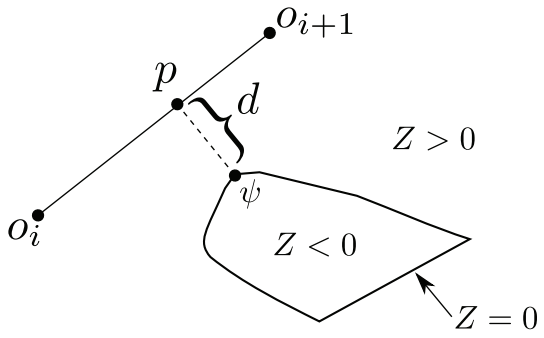


Figure 4.5: The shortest distance from a convex shape to a line segment problem.

The end points of the line segments are readily given by the robot forward kinematics, and an implicit representation of an obstacle is constructed as in 4.1. We consider a point on the line segment $o_i + p(o_{i+1} - o_i)$ parametrized by $p \in [0, 1]$ and a point on the boundary of the obstacle ψ. We use the square brackets to denote set inclusion. This gives us the following nonlinear optimization problem:

$$\min_{d} \quad \text{s.t.} \quad Z(\psi) = 0, \quad p \in [0, 1] \tag{4.9}$$

Where $d = \|o_i + p(o_{i+1} - o_i) - \psi\|$ is the distance between a point on the line segment parametrized by p and a point on the boundary of the obstacle. This could be solved as a nonlinear optimization problem using of the shelf optimization algorithms, but this may not be acceptable for real time implementation.

We propose a dynamical system as an optimization algorithm which will be more suitable for on-line implementation. Solving optimization problems using dynamical systems has for instance been used in [1] where it is applied to a path planning problem. We construct dynamical system with a stable equilibrium being the solution to (4.9). The system uses normalized steepest descent search and is continuous.

4.3 Constructing the steepest descent system

We will in this section construct the steepest descent dynamical solution to (4.9). Consider first the slightly simpler problem of finding a point ψ on the obstacles surface. If we assume that Z is constructed using linear and quadratic arguments such that ∇Z^2 has a globally stable equilibrium where $Z = 0$, then we will find this point in finite time with the steepest descent. This assumption is valid since the construction of Z is up to us.

$$\dot{\psi} = -K_1 \frac{(\nabla Z^2)^T}{\|\nabla Z^2\|} = -K_2 \text{Sign} Z(\psi) n \tag{4.10}$$

∇ denotes the gradient row vector:

$$\nabla(Z^2(\psi)) = 2Z\nabla(Z) = 2Z \left[\frac{\partial Z}{\partial x_1}, \frac{\partial Z}{\partial x_2}, \frac{\partial Z}{\partial x_3} \right] \tag{4.11}$$

And $K_1 > 0$ is a constant defining the convergence speed, see figure 4.9 for an illustation. We have also written the system as a function of $n(\psi) = \nabla Z(\psi)/\|\nabla Z(\psi)\|$ which is the normal vector pointing out of the shape in question. Its form now resembles a sliding mode controlled system. Notice that we do not need ψ to be on the boundary of the shape for n to describe the normal vector.

We consider next the problem of finding the closest point on a line given a point ψ. This problem is solvable in closed form and the solution for the parameter p is given by:

$$p(\psi) = \text{Sat} \left\{ \frac{(o_i - o_{i+1})^T (o_i - \psi)}{l_i^2} \right\} \tag{4.12}$$

Where $l_i = \|o_i - o_{i+1}\|$ is the length of link i and Sat is the saturation function between 0 and 1. So far we have managed to find a point on the boundary of an obstacle, and the closest point to this on a line. The last step we need is to find the closest point on the shape to the line. We will achieve this using a force perpendicular to n pulling ψ along the boundary of the obstacle. If we denote a "force" pulling ψ in the direction of p, as F_ψ, we have:

$$F_\psi = o_i + p(o_{i+1} - o_i) - \psi \tag{4.13}$$

The projected force along the boundary of the obstacle is given by:

$$F_{\psi}^{\perp n} = S(n)(o_i + p(o_{i+1} - o_i) - \psi) \tag{4.14}$$

Where S is the projection matrix of the appropriate dimension found in (1.4). See figures 4.6 and 4.7 for an illustration.

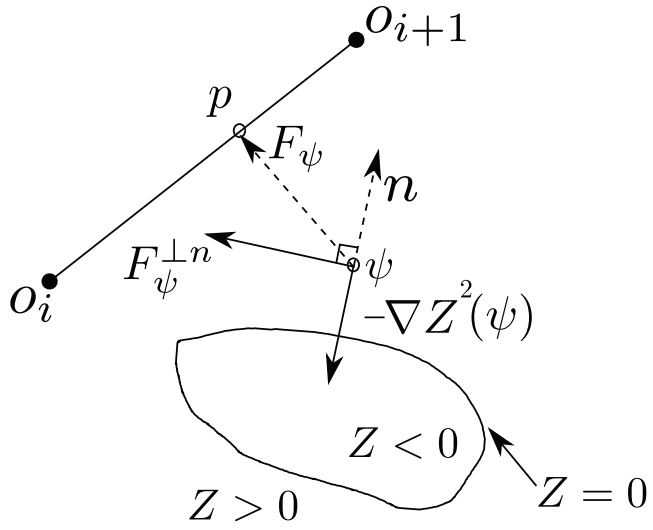


Figure 4.6: The "forces" involved in the shortest distance dynamical system.

If we combine the steepest descent towards the obstacle and the descent along the obstacle we have:

$$\dot{\psi} = K_2 \frac{F_{\psi}^{\perp n}}{\|F_{\psi}^{\perp n}\|} + K_1 \frac{\nabla(Z^2(\psi_i))^T}{\|\nabla(Z^2(\psi_i))\|} \tag{4.15}$$

Where we also define

$$F_{\psi}^{\perp n} = 0 \quad \Rightarrow \quad \frac{F_{\psi}^{\perp n}}{\|F_{\psi}^{\perp n}\|} = 0 \tag{4.16}$$

$$Z(\psi) = 0 \quad \Rightarrow \quad \frac{\nabla(Z^2(\psi_i))^T}{\|\nabla(Z^2(\psi_i))\|} = 0 \tag{4.17}$$

We will see that this system achieves finite time convergence to a point ψ_s, k_s which describe the shortest distance between the robot and a convex obstacle.

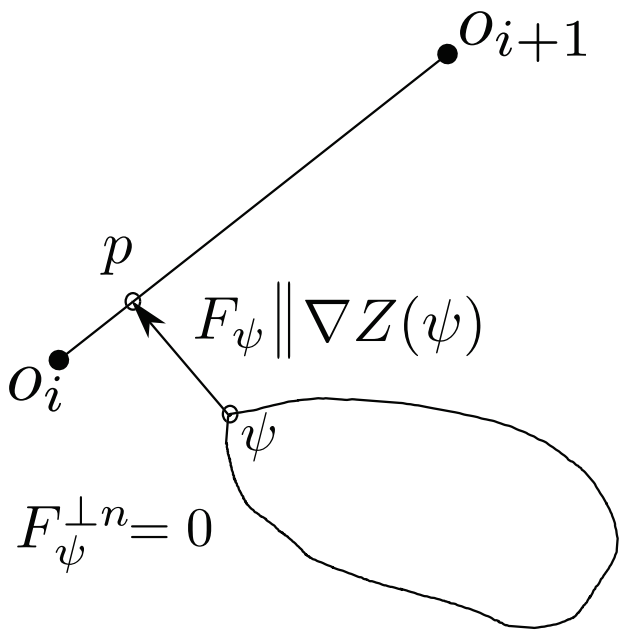


Figure 4.7: The steady state of the shortest distance system (4.15).

4.3.1 Equilibrium analysis

We will shortly discuss the stability of our system, and we start by finding the equilibria. We know by direct calculation that $o_i + p_s(o_{i+1} - o_i)$ is the closest point on the line segment $\{o_i, o_{i+1}\}$ to the obstacle, so we only need to consider the stability of ψ. For $\dot{\psi} = 0, \psi \Rightarrow \psi_s$ we have to solve the following equation:

$$K_2 \frac{F_\psi^{\perp n}}{\|F_\psi^{\perp n}\|} - K_2 \mathrm{Sign} Z(\psi) n = 0 \tag{4.18}$$

Since $F_{\perp n}$ is perpendicular to the gradient field vector at any ψ_s we see that for this to be zero, then the two terms individually have to be zero. $\nabla_x (Z^2(\psi_s))^T = 2Z(\psi_s)\nabla_x (Z(\psi_s))^T$ is by construction zero if and only $Z = 0$ and ψ_s is on the boundary of the obstacle.

$F_\psi^{\perp n}$ is zero if and only if $F^{\perp n}$ is parallel to the normal gradient field $\nabla Z(\psi_s)$. This happens for point ψ_s on a convex obstacle, one at the closest point, and one at the point farthest away from the robot on the boundary of the obstacle.

The closest of these two points satisfy the closest distance property. If we let the initial condition ψ_0 be somewhere closer to the robot than the obstacle, we know that we will converge on the correct equilibrium.

4.3.2 The gradient of Z

An efficient implementation of the shortest distance algorithm uses the gradient of the implicit representation of a shape. It would be convenient to have an easy way to compute this since Z may in general be quite complex. We therefore derive the gradient in closed form.

Remember that a given shape may be represented by a composition of intersection and union functions of k inequalities. We denote the composed functions with a superscript by slightly abusing notation.

$$Z(\beta_1(x), ..., \beta_n(x)) = Z^1(\beta_1, Z^2(\beta_2, Z^3(\dots, Z^{n-2}(\beta_{n-2}, Z^{n-1}(\beta_{n-1}, \beta_n)\dots)) \tag{4.19}$$

With $Z(a, b) = a + b - |a - b|$. We have by the chain rule:

$$\frac{\partial Z}{\partial x_i} = \frac{\partial Z}{\partial \beta_1}\frac{\partial \beta_1}{\partial x_i} + \dots + \frac{\partial Z}{\partial \beta_n}\frac{\partial \beta_n}{\partial x_i} \tag{4.20}$$

We define the following functions which are derivatives of Z^i with respect to their first argument of the union and intersection function:

$$\phi_i = \frac{\partial Z_1^i(\beta_i, Z_1^{i+1})}{\partial \beta_i} = 1 - \text{sign}(\beta_i - Z^{i+1}) \tag{4.21}$$

$$\phi_i = \frac{\partial Z_2^i(\beta_i, Z_2^{i+1})}{\partial \beta_i} = 1 + \text{sign}(\beta_i - Z^{i+1}) \tag{4.22}$$

We observe that the derivative of Z^i with respect to its second argument is given by $2 - \phi^i$ from the definition of Z, i.e:

$$\frac{\partial Z^i(\beta_i, b)}{\partial b} = 2 - \phi_i \tag{4.23}$$

This enables us to write the partial derivatives of Z with respect to β_i as:

$$
\begin{aligned}
\frac{\partial Z}{\partial \beta_1} &= \frac{\partial Z^1}{\partial \beta_1} = \phi_1 \\
\frac{\partial Z}{\partial \beta_2} &= \frac{\partial Z^1}{\partial Z^2}\frac{\partial Z^2}{\partial \beta_2} = (2-\phi_1)\phi_2 \\
\frac{\partial Z}{\partial \beta_3} &= \frac{\partial Z^1}{\partial Z^2}\frac{\partial Z^2}{\partial Z^3}\frac{\partial Z^3}{\partial \beta_3} = (2-\phi_1)(2-\phi_2)\phi_3 \\
&\vdots \\
\frac{\partial Z}{\partial \beta_{n-1}} &= \phi_{n-1}\prod_{i=1}^{n-2}(2-\phi_i) \\
\frac{\partial Z}{\partial \beta_n} &= \prod_{i=1}^{n-1}(2-\phi_i)
\end{aligned}
$$

We now the gradient is given by:

$$\nabla_x Z = \begin{bmatrix} Z_{x_1} \\ Z_{x_2} \\ Z_{x_3} \end{bmatrix} = J_\beta^T P \tag{4.24}$$

Where the subscripts indicates the partial derivative. where J_β is the Jacobian of all the β_i with respect to x_1, x_2, x_3.

$$J_\beta = \begin{bmatrix} \frac{\partial \beta_1}{\partial x_1} & \frac{\partial \beta_1}{\partial x_2} & \frac{\partial \beta_1}{\partial x_3} \\ \frac{\partial \beta_2}{\partial x_1} & \frac{\partial \beta_2}{\partial x_2} & \frac{\partial \beta_2}{\partial x_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial \beta_n}{\partial x_1} & \frac{\partial \beta_n}{\partial x_2} & \frac{\partial \beta_n}{\partial x_3} \end{bmatrix} \tag{4.25}$$

and P are the partial derivatives of Z with respect to β stacked on top of each other.

$$P = \begin{bmatrix} \phi_1 \\ (2-\phi_1)\phi_2 \\ \vdots \\ \prod_{i=1}^{n-1}(2-\phi_i) \end{bmatrix} \tag{4.26}$$

4.3.3 Example

We present a simple example with a planar elbow robot and a prismatic obstacle. A box in $\mathbb{R}^2$ with position $[x_b, y_b]$, width and height w, h and rotation θ is given by the intersection $\beta_1 \cap \beta_2$ where β_i may be represented by the following inequalities:

$$\beta_1 = \| \cos(\theta)(x - x_b) - \sin(\theta)(y - y_b)\|^2 - w^2 \tag{4.27}$$
$$\beta_2 = \| \sin(\theta)(x - x_b) + \cos(\theta)(y - y_b)\|^2 - h^2 \tag{4.28}$$

The intersection is:

$$Z = \beta_1 + \beta_2 + \|\beta_1 - \beta_2\|; \tag{4.29}$$

If we define $\phi_1 = 1 + \text{sign}(\beta_1 - \beta_2) = 2u_1(\beta_1 - \beta_2)$ then the gradient of Z is given by:

$$\nabla Z^T = \begin{bmatrix} \frac{\partial Z}{\partial x} \\ \frac{\partial Z}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \beta_1}{\partial x} & \frac{\partial \beta_2}{\partial x} \\ \frac{\partial \beta_1}{\partial y} & \frac{\partial \beta_2}{\partial y} \end{bmatrix} \begin{bmatrix} \phi_1 \\ 2 - \phi_1 \end{bmatrix} \tag{4.30}$$

$$\nabla Z^T = \begin{bmatrix} 2(x - x_b)c_\theta^2 - (y - y_b)s_{2\theta} & 2(x - x_b)s_\theta^2 + (y - y_b)s_{2\theta} \\ 2(y - y_b)s_\theta^2 - (x - x_b)s_{2\theta} & 2(y - y_b)c_\theta^2 + (x - x_b)s_{2\theta} \end{bmatrix} \begin{bmatrix} 2u_1(\beta_1 - \beta_2) \\ 2u_1(\beta_2 - \beta_1) \end{bmatrix} \tag{4.31}$$

A simulation trail is shown in figure 4.8

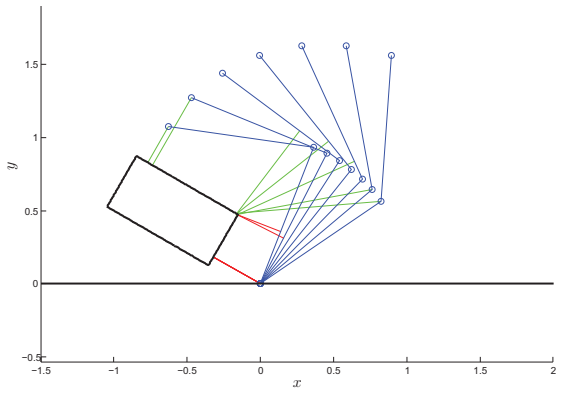


Figure 4.8: Screenshots of a robot moving close to a box obstacle with the dynamically computed shortest distances as lines.

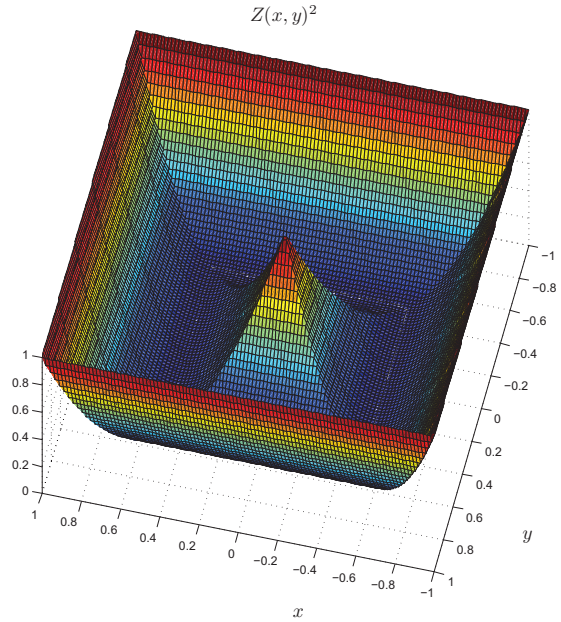


Figure 4.9: The function Z^2. We see that the gradient of Z^2 points towards the boundary of the box making the boundary of the box globally convergent using steepest descent.

4.4 Shortest distance between two convex shapes

Representing the robot as a link of line segments is inefficient if the geometry of the robot is asymmetric or irregular. This is because obstacles in the workspace have to be grown proportionate the outermost point on the robot. This will render the free configuration space much smaller than strictly necessary, and will make good synchronization harder to achieve. This motivates us to present a steepest descent system to find the shortest distance between two convex shapes. The algorithm is similar to the one previously stated and relies heavily on vector decomposition.

Given two convex shapes in the workspace $\beta_1(x), \beta_2(x)$, and two points ψ_1, ψ_2 we present the following system which will in finite time converge to the steady state ψ_1^s, ψ_2^s. These points have the property of being the closest points between the shapes, and $\|\psi_1^s - \psi_2^2\|$ is the shortest distance. This is the system:

$$\dot{\psi}_1 = \frac{F^{\perp n_1}}{\|F_{\perp n_1}\|} + K_{\psi_1} \frac{\nabla_x(\beta_1^2(\psi_1))^T}{\|\nabla_x(\beta_1^2(\psi_1))\|} \tag{4.32}$$

$$\dot{\psi}_2 = -\frac{F^{\perp n_2}}{\|F_{\perp n_2}\|} + K_{\psi_2} \frac{\nabla_x(\beta_2^2(\psi_2))^T}{\|\nabla_x(\beta_2^2(\psi_2))\|} \tag{4.33}$$

n_1 and n_2 are the normalized gradient vectors of β_1 and β_2 respectively. Where $F_{\perp n_1}$ is the orthogonal complement of $\psi_2 - \psi_1$ with respect to the vector pointing out of n_1 evaluated at ψ_1:

$$F^{\perp n_1} = S(n_1)(\psi_2 - \psi_1) \tag{4.34}$$

And $F_{\perp n_2}$ is similarly given by:

$$F^{\perp n_2} = S(n_2)(\psi_2 - \psi_1) \tag{4.35}$$

We do not need to consider the stability of (4.32) as these two systems are independently identical to (4.15) and the convergence properties we need follow directly.

4.4.1 Example

We revisit the planar elbow with a box example. We now assume that the robot is covered by ellipses. The following ellipse will cover the line segment $\{o_0, o_1\}$ with length l_1:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} - (l_1 + \epsilon)^2 = 0 \tag{4.36}$$

For some small $\epsilon \geq 0$. We place the two ellipses over the joints given the forward kinematics.

$$\beta_2 = \frac{h_1(x,q)^2}{a^2} + \frac{h_2(x,q)^2}{b^2} - (l_2 + \epsilon)^2 \tag{4.37}$$

Where

$$h(x,q) = T_1^{-1}(q) \left(\begin{bmatrix} x_1 \\ x_2 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{2}(o_0 + o_1) \right) \tag{4.38}$$

Where T_1 is the homogeneous transformation from the base to the first joint. β_2 in closed form is given by:

$$\beta_2 = \frac{(x_1 \cos(q_1) + x_2 \sin(q_1) - \frac{l1}{2})^2}{a^2} + \frac{(x_2 \cos(q_1) - x_1 \sin(q_1))^2}{b^2} - (l_1 + \epsilon)^2; \tag{4.39}$$

The ellipse for the second joint is given by:

$$\beta_3 = \frac{g_1(x,q)^2}{a^2} + \frac{g_2(x,q)^2}{b^2} - (l_2 + \epsilon)^2 \tag{4.40}$$

Where

$$g(x,q) = (T_1 T_2)^{-1}(q) \left(\begin{bmatrix} x_1 \\ x_2 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{2}(o_1 + o_2) \right) \tag{4.41}$$

Where T_1T_2 is the homogeneous transformation from the base to the first second joint. An illustration of the system is shown with a superelliptical obstacle with $a = 1, b = 0.2$.

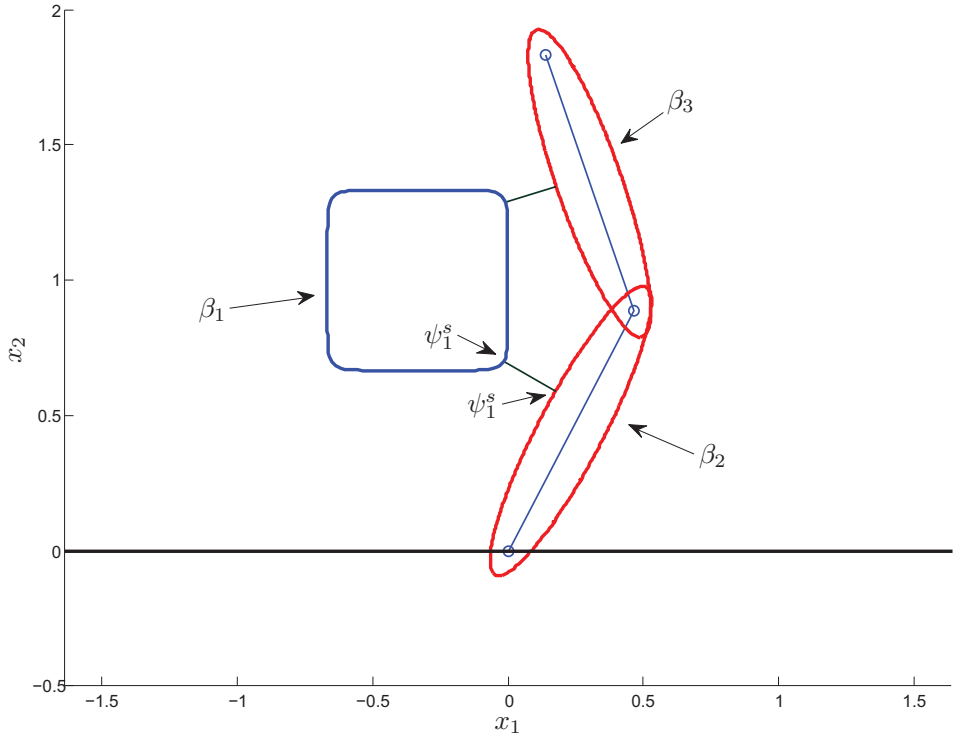


Figure 4.10: An elbow robot covered by ellipses where the shortest distance to a superellipse is computed on-line.

4.5 Complexity considerations

Shortest distance estimation is computationally expensive, and even though finite time convergence has been achieved we have no explicit bounds on the convergence time. In addition we need to perform distance estimation between each convex portion of the robot and each obstacle. While this number grows linearly with the number of obstacles one would spend a considerable time just for distance estimation for each time sample given a dense workspace. The method utilizes temporal coherency to a great extent if one assumes that the obstacles and the robot have not moved far over any given time sample. Optimization may be obtained by moving the ψ_i attached to

the robot in accordance with the movement of the robot such that when ψ_i reaches the robots surface it will stay there. Variable step length by using for instance the *Wolfe conditions* will speed up the convergence at the expense of continuity.

Chapter 5

Collision Avoidance Control

In this chapter we will review strategies for collision avoidance control and robot control in general. We will justify the design choices which we will use in the control design in the next chapter.

5.1 Collision avoidance in path planning

Collision avoidance for robot manipulators has been studied heavily in the field of path planning. The path planning problem is the following: Give two points a and b and a set of obstacles, find an executable trajectory between a and b such that no collision occurs. This problem seem similar to the synchronization with obstacle avoidance problem. We will therefore shortly review some results in the field of path planing.

One of the first successful solutions to the path planning is known as the *probabilistic road map method*, [6]. One first needs to construct the configuration space obstacle, then randomly occupy it with way points. A tree of feasible vertices are added, and then the shortest path is used to generate a trajectory. Another solution paradigm is called the artificial potential field method introduced in [7]. The essence of the controller is that is generates forces repelling the robot away from obstacles, and forces pulling the robot towards the control objective. The robot converges to the goal given a properly constructed potential field. The potential field is in this setting referred to as a navigation function. A third solution strategy is called the *elastic strip method* [1], which to some extent is a combination of the potential field approach and the road map method. The proposed solution is

to construct an elastic strip between the start and goal point consisting of way points. The elastic strip is deformed by a obstacle potential field and contracts such that a short collision free trajectory is generated as the equilibrium is reached. The elastic strip method may be used for global path planning or path deformation if a path is already given.

Different variations on these methods broadly represent the suggested solutions in path planning literature.

5.2 How synchronization differs from path planning

The path planning problem does seem like a similar problem to ours. The biggest difference is that we do not have a goal point, but a goal trajectory with unknown future behavior. Another crucial difference is that none of the methods impose any real time constraints on the trajectory or the speed at which it is executed. The elastic strip method may be used to deform an already provided trajectory, and can be said to account for trajectory restrictions, but we do not have a predefined trajectory to deform. The potential field approach may provide convergence to some goal point, but how the robot behaves between a and b cannot easily be specified. These strategies are also developed in joint space, which requires the construction of the joint space obstacle. These strategy cannot be applied directly to our problem as the configuration space obstacle is computationally infeasible to compute on-line even for a mildly complex workspace. So a solution to the path planning problem is not a solution to the synchronization with obstacle avoidance problem, but we will use ideas from the artificial potential field approach to develop a local synchronization controller.

5.3 The potential field

The potential field method was introduced by Khatib in 1986, [7] and has since then gained wide popularity in the control theory community. Our choice to use artificial potentials for our controller are due to the following: It produces a closed loop system which readily susceptible to stability analysis unlike heuristic tree-traversing methods. Different potentials may be used for

different applications making the method highly flexible. The method is also simple to implement.

The artificial potential field is a potential field in that it is the gradient of a scalar function, and it is artificial as it is imposed through control input unlike for instance gravity. We want the potential field to produce a force on the robot ensuring collision avoidance.

We impose the following restrictions on our repulsive force: We want the repulsive force to exert little or no influence far away from an obstacle. The magnitude of the force must tend to infinity close to the obstacle to ensure collision avoidance. We also want a continuous force such that we have existence of classical solutions.

The simplest approach is to construct a force solely dependent on the distance to the nearest obstacle. We have shown how to find the shortest distance from the robot to the workspace obstacle, and we will use this to construct a repulsive force.

The design of the repulsion force is entirely up to the designer, and may vary between different types of obstacles. An object where the goal is to actually crash, like the act of pushing a button, another force will need to be used. We will not consider cases like this, but it is important to recognize the versatility of the design methodology.

5.4 Different potential fields

A repulsive force with magnitude proportional to the inverse of the collision distance would meet the properties required to construct a proper repulsion force. To ensure that the force is zero outside a specified security zone around the obstacle we have from [10] p.172:

$$F_i(q) = \begin{cases} \eta_i \left(\frac{1}{d(q)} - \frac{1}{\rho_0}\right) \frac{1}{d(q)^2} \frac{p_i - p_{c_i}}{\|p_i - p_{c_i}\|} & d_i(q) \leq \rho_0 \\ 0 & d_i(q) > \rho_0 \end{cases} \tag{5.1}$$

Where ρ_0 is the security distance where the obstacle exerts a force on the robot, p_i is a point on the robot closest to the obstacle i and p_{c_i} is a point on the obstacle closest to the robot.

This repulsive force field is easily implementable for time-varying obstacles where the closest robot/obstacle distance is known at every time instance.

There are however several reasons why this is not an ideal choice. If for instance the synchronization objective is close to the boundary on obstacle, then even steady state convergence is infeasible as a local equilibrium will be located some distance away from the obstacle. The approach to this point may also be oscillatory. Other less than optimal behavior is the fact that obstacle exert repulsive forces even while moving away from, or parallel to the robot.

One way to remedy this is to use another force dependent on the estimated time to collision. Using the time to collision for obstacle avoidance hs been considered in [3]. The form of the repulsive force is however very different from the one used in [3].

This employs velocity information of the manipulator and its environment and is inherently designed to handle a time varying workspace. Assuming that a velocity estimate $\dot{d}$ of the distance d is available, then the distance at time $t+T$ assuming constant velocity is by the Taylor theorem:

$$d(t+T) = d(t) + \dot{d}(t)T + O(T^2) \tag{5.2}$$

We want an estimate for the time to collision, i.e. a time $T = T_c$ such that $d(t+T) = 0$. A linear estimate produces the following time to collision estimate:

$$T_c = -\frac{d(t)}{\dot{d}(t)} \tag{5.3}$$

T_c is negative if the obstacle is moving away from robot, and infinite if the robot is moving parallel to, or is relatively stationary to the obstacle surface. One simple controller based upon the estimated time to collision is the following:

$$F_i(q,\dot{q}) = \begin{cases} \eta_i \left(\frac{1}{T_c} - \frac{1}{T_0}\right) \frac{1}{T_c^2} \frac{p_i - p_{c_i}}{\|p_i - p_{c_i}\|} & T_c \leq T_0 \\ 0 & T_c > T_0 \quad \text{or} \quad T_c < 0 \end{cases} \tag{5.4}$$

Where T_0 is the threshold time to collision where the controller kicks in. p_i is the point on the robot closest to the obstacle, and p_{c_i} is a point on the obstacle closest to the robot, and $\eta_i > 0$ is the gain. As opposed to keeping the robot at a fixed distance from the obstacle, the controller seeks to keep the robot above some fixed positive time to collision. This results in the robot

slowing down relative to an obstacles speed as opposed to be pushed back by it. It will also allow the robot to move arbitrarily close to an obstacle as long as the relative speed towards the obstacle approaches zero. It will also give avoidance priority to avoiding obstacle moving very fast in the workspace. A more thorough analysis if found in section 6.6.2.

5.5 Joint space vs Task space control

Control of robot manipulators is traditionally carried out in two different design methodologies. These are joint space control and task space control. Perhaps the most crucial design choice one has to make when performing robot control is whether to use joint space control or task space control. We shortly review their strengths and weaknesses for our control problem to clarify our choice.

5.5.1 Task space control

A task space[1] control algorithm is carried out explicitly on the end effector position and orientation. The task space is the physical space inhabited by the robot plus the orientation space. This makes the task space approach more intuitive. We note the following: The synchronization objective is given in the task space. The robots geometry and the geometry of the obstacles are given in the task space. A minimal representation is however needed for orientation control as the orientation reference is given as a rotational matrix.

5.5.2 Joint space control

Joint space control is carried out considering desired joint angles and velocities. The control objective is not uniquely given in joint space, but is mapped to the joint space from the task space via the inverse kinematics. The global joint space obstacle is generally infeasible to compute on-line, but it is not needed as distance information in the task space may be mapped to the joint space. The robot is represented by a point mass in the joint space, which makes path planning conceptually easier. One does not need to consider

[1] This space is also referred to as the "work space" or "operational space" in robot control literature.

minimal representations of $SO(3)$ as the end effector orientation is mapped to a joint reference. The synchronization controllers in [8] is developed in joint space, and building upon these would be simpler in joint space. The robots dynamical model is also given in joint space, but is easily mapped to the task space by a change of coordinate given a minimal representation of $SO(3)$. A drawback with joint space control which is rarely discussed is the case where there are no solutions to the inverse kinematics, i.e. the trajectory is infeasible. While this problem is solvable in a number of ways, it would be preferable not to have to worry about it.

5.6 Feedback linearization

We will shortly review a technique used commonly in control of robot manipulators. The robot model (1.6) is on the input affine form, and is hence susceptible to feedback linearization. Assuming perfect state information we see that the input:

$$u = M(q)u_2 + G(q, \dot{q})\dot{q} + G(q) \tag{5.5}$$

Will yield a linear closed loop double integrator:

$$\ddot{q} = u_2 \tag{5.6}$$

The linearized system is desirable since is easy to control and analyze. Perfect feedback linearization is however impossible in practice due to model assumptions and uncertainties.

5.7 Resolved acceleration

Resolved accelerations is a technique used for mapping the joint space robot model to the task space. The Analytic Jacobian is also used for this purpose, but using it directly is not advisable when using roll pitch yaw angles as the minimal representation of $SO(3)$. This is because the mapping is undefined for certain configuration called representation singularities. We will limit the possibility of representation singularities using a angle error mapping. A review of the method for common parameterizations of $SO(3)$ can be found in [2].

We start with the feedback linearized system given in the joint space:

$$\ddot{q} = u_2 \tag{5.7}$$

We want to map the system to the task space. We use the geometric Jacobian J, and we know that

$$\begin{bmatrix} \dot{x} \\ \omega \end{bmatrix} = J(q)\dot{q}, \quad \begin{bmatrix} \ddot{x} \\ \dot{\omega} \end{bmatrix} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{5.8}$$

Setting u_2 to the solution of this with respect to $\ddot{q}$ gives us a mapping to the task space given in the variables x for position and ω for angular velocity of the end effector. We construct the input:

$$u_2 = J^{-1}(u_3 - \dot{J}\dot{q}) \tag{5.9}$$

Given a nonsingular Jacobian then the closed loop system becomes:

$$\ddot{q} = J^{-1}(u_3 - \dot{J}\dot{q}) \quad \Longleftrightarrow \quad J\ddot{q} - \dot{J}\dot{q} = u_3 \quad \Longleftrightarrow \quad \begin{bmatrix} \ddot{x} \\ \dot{\omega} \end{bmatrix} = u_3 \tag{5.10}$$

We will now split up the systems explicitly into the position system $\ddot{x} = u_p$ and the orientation system $\dot{\omega} = u_o$. If we first consider the position system, then a stable linear tracking controller is:

$$u_p = \ddot{x}_d - K_p(x - x_d) - K_d(\dot{x} - \dot{x}_d) \tag{5.11}$$

With the closed loop error system $e = x - x_d$:

$$\ddot{e} = -K_p e - K_d \dot{e} \tag{5.12}$$

which is exponentially stable for all $K_p, K_d > 0$.

5.7.1 The roll-pitch-yaw error system

We now need to parametrize $SO(3)$ to construct a system similar to (5.12) for the orientation. We may extract the roll pitch yaw angle difference directly from $R^T R_d$ using (3.45):

$$\Phi_e = f\left(R^T(q)R_d\right) \tag{5.13}$$

Where $\Phi_e = [\phi_e\, \theta_e\, \psi_e] \neq [\phi_d - \phi\, \theta_d - \theta\, \psi_d - \psi]$ is a measure of the roll pitch yaw angle difference. This is advantageous since representation singularities will occur only for an orientation synchronization error of $\theta = k\frac{\pi}{2}$. If we used the roll pitch yaw angles directly, then a desired pitch of 90° would be impossible to achieve.

$\dot{\Phi}_e$ and $\ddot{\Phi}_e$ are given by:

$$\omega_e = \omega_d - \omega = R(q)B(\Phi_e)\dot{\Phi}_e \tag{5.14}$$

$$\dot{\omega}_e = \left(\frac{d}{dt}R(q)B(\Phi_e)\right)\dot{\Phi}_e + R(q)B(\Phi_e)\ddot{\Phi}_e, \tag{5.15}$$

Choosing $u_o = \dot{\omega}_d - \left(\frac{d}{dt}R(q)B(\Phi_e)\right)\dot{\Phi}_e + R(q)B(\Phi)(K_{po}\Phi_e + K_{do}\dot{\Phi}_e)$ gives the closed loop orientation system:

$$\dot{\omega} - \dot{\omega}_d + \left(\frac{d}{dt}R(q)B(\Phi)\right)\dot{\Phi}_e = R(q)B(\Phi_e)(K_{po}\Phi_e + K_{do}\dot{\Phi}_e) \tag{5.16}$$

If we write the angular acceleration as a function of $\ddot{\Phi}$ using (5.14) we get:

$$-\dot{\omega}_e + \left(\frac{d}{dt}R(q)B(\Phi)\right)\dot{\Phi} = -R(q)B(\Phi_e)\ddot{\Phi}_e = R(q)B(\Phi_e)(K_{po}\dot{\Phi}_e + K_{do}\Phi_e) \tag{5.17}$$

Which gives us the exponentially stable closed loop system given $K_{po}, K_{do} > 0$:

$$\ddot{\Phi}_e = -K_{po}\dot{\Phi}_e - K_{do}\Phi_e \tag{5.18}$$

We have now successfully mapped the system to the task space and produced an exponentially decreasing tracking error in position and orientation.

5.8 Applying forces to the robot

Now that we have mapped the system to the task space and designed appropriate obstacle repulsion forces, we need to map these forces to the robot. If we assume that a force F_{rep} acts on the end effector then we can simply write:

$$\ddot{x} = u_2 = u_3 + F_{rep} \tag{5.19}$$

By Newtons 2d law, which gives us the closed loop system:

$$\ddot{e} = -K_p e - K_d \dot{e} + F_{rep} \tag{5.20}$$

The orientation system remains unchanged since no torque is exerted on the end effector by the obstacle. This is one of the differences between traditional hybrid force motion control and our approach as we may apply the force where we see fit as opposed to a "real" force which will appear in both the orientation system and the position system.

We now present an example which will illustrate our choice of performing task space control.

5.9 Example: Elbow robot near a wall

We present a simple example in order to motivate our design for choosing between the two robot representations. Consider a planar elbow robot in the vicinity of a static wall. Assume that the synchronization objective is a static point. The synchronization objective lies beyond or inside the wall and is infeasible. We want the robot to place the end effector close to the wall and with the same height as the desired value. This way we can at least say that we can achieve synchronization in height.

In figure 5.1 we see the setup.

On the left we see the steady state of a simple joint space controller, and on the left is a task space controller showing some initial configuration at t_1 and the steady state at t_2. x_d^o is the optimal end effector position. q_d are the desired joint values obtained from passing the desired end effector point x_d to the inverse kinematics. F_{rep} is a force induced on the robot from the

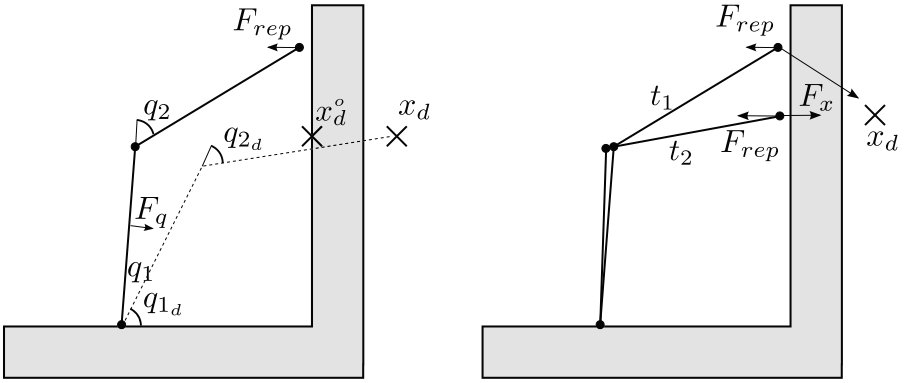


Figure 5.1: The elbow/wall synchronization problem.

obstacle via some properly constructed vector field. F_q is the force of some controller in joint space, and F_x is a force on the end effector in the task space.

We observe that the task space controller has achieved partial synchronization while the joint space controller robot has settled into some undesirable steady state.

Lets analyze the behavior of the controllers:

Joint space controller

We assume that feedback linearization has been composed with a PD controller such that the closed loop system is given as:

$$\ddot{q} = -K_p(q - q_d) - K_d(\dot{q}) + J_v^T(q)F_{rep}(q) \tag{5.21}$$

Where the repulsive force is mapped to the joint space via the velocity end effector Jacobian. The steady state, $\ddot{q} = \dot{q} = 0$ is given by:

$$K_p(q^s - q_d) = J_v^T(q^s)F_{rep}(q^s) \tag{5.22}$$

This equation is hard to solve analytically but solving it numerically gives the solution shown if figure 5.1. The controller seeks to minimize the angle error, in 5.1 we see that $q_2 - q_{2_d}$ is approximately zero. However we don't want to

minimize this error as we want to synchronize the end effector position. The picture becomes clearer if we look at the configuration space obstacle seen if figure 5.2.

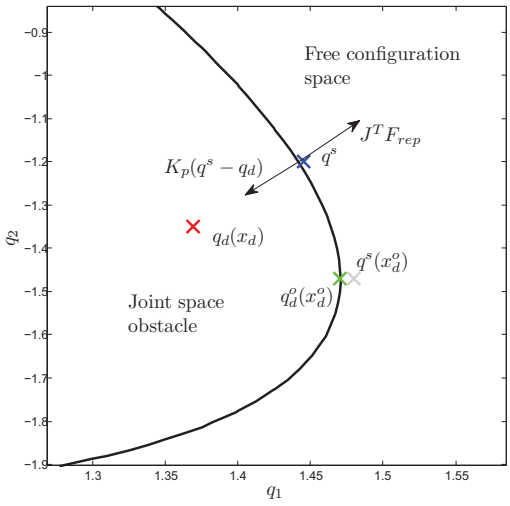


Figure 5.2: The elbow/wall synchronization problem in joint space close to the steady state.

The joint space obstacle is defined by $o_{2_x}(q) = \cos(q_1) + \cos(q_1 + q_2) > o_{c_x}$ where o_c defines the position of the wall and o_{2_x} is the x-coordinate of the end effector. We see that the minimum angle error with respect to the configuration space obstacle, q^s is not close to the projected reference q_d^o. Also shown is the steady state where q_d^o has been used as the reference position. The steady state error in this case is still not zero projected into the task space obstacle, even though it is closer than before.

One intuitive strategy to make the joint space controller work better is to explicitly project the reference point onto the obstacle surface, and then extract the desired joint angles from this point. We now get q_d^p such that $h(q_d^p) = x_d^o$ where h is the robots forward kinematics. The closed loop steady state is now:

$$K_p(q^s - q_d^p) = J_v^T(q^s)F_{rep}(q^s) \tag{5.23}$$

We see that the x_s^o is guaranteed to be reached if $F_{rep} = 0$. To have a repulsive force from an obstacle be zero on the obstacle surface is not advisable, so we would need to move the projected point some way away from the obstacle.

And even then we are not guaranteed that to have zero projected error. To make matters worse, consider the case where the reference is moving in and out of an obstacle. If we choose a certain distance from the obstacle at which we project the reference, then the time derivative of the error used in the controller is not defined at the time of projection since it not continuously differentiable at this point. So one either needs to slow the robot down to zero speed as one approaches this point, or filter the reference with some lowpass filter for classical solutions to the closed loop system to exist. If one employs a velocity error observer however, then this filtering is already carried out, making the problem less severe for systems without explicit velocity information. Even with all this work we further need to impose constraints on F_{rep} in order for the objective to be feasible. The point is that this approach is complicated, and we will show below why a task space controller is better suited for end effector synchronization with obstacle avoidance.

Workspace controller

We assume that a mapping to the task space has been done and feedback linearization has been employed with a PD controller with scalar gains such that the closed loop system is:

$$\ddot{x} = -k_p(x - x_d) - k_d(\dot{x}) + F_{rep} \tag{5.24}$$

The steady state of the system is given by:

$$-k_p(x - x_d) + F_{rep} = 0 \tag{5.25}$$

We see that the vectors $-k_p(x - x_d)$ and F_{rep} are pointing in opposite directions and are of the same length. If we assume that the vector $F_{rep}(x^s)$ is normal to the obstacle surface then we know that we are situated at some point where the error normal to the surface is zero. More formally we have that the projected error into the wall $(x - x_d)^p$ is the orthogonal complement of $(x - x_d)$ along F_{rep} which is:

$$(x - x_d)^p = (x - x_d) - \left[(x - x_d)^T F_{rep}\right] \frac{F_{rep}}{\|F_{rep}\|^2} \tag{5.26}$$

Substituting in $(x - x_d)^T = F_{rep}/k_p$ gives us:

$$(x - x_d)^p = (x - x_d) - \frac{1}{K_p}\left[F_{rep}^T F_{rep}\right]\frac{F_{rep}}{\|F_{rep}\|^2} \tag{5.27}$$

Which gives us that the projected error into the obstacle surface is zero:

$$(x - x_d)^p = (x - x_d) - \frac{1}{k_p}F_{rep} = 0 \tag{5.28}$$

Which is the desired behavior of obstacle avoidance controller in this case. Notice how much simpler the analysis is in the task space. The complexity in this approach is in the mapping from the joint space to the task space. Notice also that we have not imposed any explicit restrictions on q_d, such that either of the two possible robot poses are feasible steady state solution.

5.9.1 Mapping a contact force to end effector

We will now consider the how we map a force acting on an arbritrary point on the robot to the end effector. It would be nice if we could map a force from any point on the robot by simply adding them togeter in the position system as we did with the forces acting on the end effector. We have illustrated a case in figure 5.3 why this cannot be done. A scenario is shown where the sum of the forces are zero while a crash is imminent.

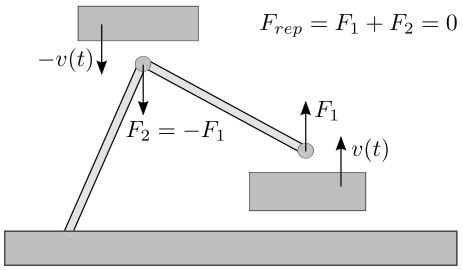


Figure 5.3: Two objects are approaching the robot at two different points excerting equal force. Adding these together will not achieve collision avoidance

The mapping of a force F_p acting on a given fixed point p on the robot to the joint space is given by $J_p^T F_p$, [10].

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u + J_p^T F_p \tag{5.29}$$

Where J_p^T denotes the geometric Jacobian for the point p. If we now revisit the mapping from a force acting on the end effector to the position system which we added in the following way:

$$\ddot{e} = f(e,\dot{e}) + F_x \tag{5.30}$$

Which is consistent with Newtons 2d law given a point mass. In doing this however we are actually performing an implicit mapping of a force from the jointspace to the task space from (5.29) to (5.30). We derive this mapping by applying the jointspace to task space mapping through intput with feedback linearization assuming that p is the end effector position:

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q) = M(q)(J^{-1}u_2-\dot{J}\dot{q})+C(q,\dot{q})\dot{q}+G(q)+J^T F \tag{5.31}$$

$$\begin{bmatrix}\ddot{x}\\ \dot{\omega}\end{bmatrix} = u_2 + JM^{-1}(q)J^T F \tag{5.32}$$

We observe that the force is mapped to the end effector with the linear transformation $JM^{-1}(q)J_p^T$. This is called the mobility tensor in the literature. Since we added the force F in the system (5.32) without the mobility tensor, it is appearant that the implicit mapping we were performing was $F_x = (J^{-T}MJ^{-1})(JM^{-1}J)F_p$ such that (5.29) and (5.30) are consistent. We may interpret this as scaling the force given in the taskspace such that the end effector is a point with unit mass. So to keep our force mapping consistent we need to apply the mapping from an arbitrary point on the robot to the end effector as:

$$F_{x_p} = (J^{-T}MJ^{-1})(JM^{-1}J_p)F_p = J^{-T}J_p^T F_p \tag{5.33}$$

If we consider (5.17) then we see that the mapping from a general force to the rotation system is:

$$F_o = B^{-1}(\Phi_e)R(q)^T W_o \tag{5.34}$$

Where W_o is the last three elements of $J^{-T}J_p^T F_p$.

Example

We see how one maps forces to the end effector in the scenario presented in figure 5.3. We consider an elbow robot where a force F_1 is applied to the second joint a force F_2 to the end effector. The robot has links of unit length, and the Jacobian for the second joint o_1 and the end effector o_2 are:

$$J_{o_1} = \begin{bmatrix} -\sin(q_1) & 0 \\ \cos(q_1) & 0 \end{bmatrix} \quad J_{o_2} = \begin{bmatrix} -\sin(q_1+q_2) - \sin(q_1) & -\sin(q_1+q_2) \\ \cos(q_1+q_2) + \cos(q_1) & \cos(q_1+q_2) \end{bmatrix}$$

We assume that at some time t_0 the robot is in the pose $q_1 = \pi/4, q_2 = -\pi/2$ and the obstacle avoidance forces $F_1 = [\,0 \; -1\,]^T$ is applied to the point o_1 and the force $F_2 = [\,0 \quad 1\,]^T$ is applied to the end effector. We map these forces to the end effector and add them together to find the resulting repulsive force felt by the end effector.

$$F_{1_x} = \left[J_{o_2}^{-1}\right]^T J_{o_1}^T F_1 \Big|_{q_1(t_0), q_2(t_0)} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} \tag{5.35}$$

$$F_x = \left[J_{o_2}^{-1}\right]^T J_{o_1}^T F_1 + \left[J_{o_2}^{-1}\right]^T J_{o_2}^T F_2 = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \tag{5.36}$$

Applying the force F_x, which now is not equal to zero, to the end effector will result in the robot successfully avoiding the collision. An illustration of the scenario is seen in figure 5.4.

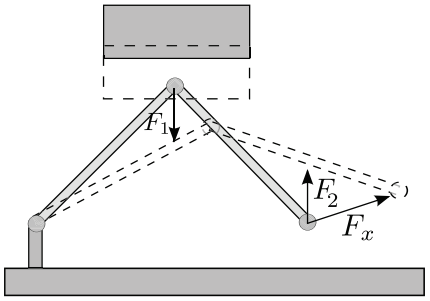


Figure 5.4: Two objects are approaching the robot at two different points excerting equal force. Mapping these to the end effector resuslts in collision avoidance

Chapter 6

The Synchronization Controller

The results derived in this chapter are the primary contributions to the field of obstacle avoidance and synchronization found in this thesis.

The synchronization controller is proposed in this chapter. It is developed in the framework of artificial repulsion forces. The obstacle avoidance controller is based upon the original found in [7], and is extended to also achieve partial synchronization. The novelty of this chapter lies in the combination of the two problems of synchronization with collision avoidance in a new way which allows us to present proofs regarding partial synchronization in the proximity of obstacles. The main contribution in this thesis is found in the problem formulation. The problem formulation is stated in a way which allows us to split the system up in different parts which we may look at independently. These subsystems are analyzed using standard stability theorems.

We begin by considering collision avoidance for only the end effector and planar obstacles. We then generalize this to include general obstacles. The last and most general case is where the entire robot is subject to collision from general obstacles.

6.1 The ideal controller

We now have everything we need in order to develop a synchronization with obstacle avoidance controller. We will do so in the workspace using feedback linearization and resolved acceleration. We develop the controller assuming perfect information. That is full state measurement and known synchronization position, velocity and acceleration for both position and orientation. We

also assume that the shortest distance to the workspace obstacle is known. We will first only consider collision between the end effector and the obstacles, and we assume that the distance to the obstacles and its time derivative is given. The problem becomes:

Given the dynamical system:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = u \tag{6.1}$$

And a synchronization objective:

$$\{x_d(t), R_d(t)\}, \quad \{\dot{x}_d(t), \omega_d(t)\}, \quad \{\ddot{x}_d(t), \dot{\omega}_d(t)\} \tag{6.2}$$

Find an input u such that:

$$\lim_{t\to\infty} \|x(t) - x_d(t)\| = 0 \tag{6.3}$$
$$\lim_{t\to\infty} \|R(t) - R_d(t)\| = 0 \tag{6.4}$$
$$\lim_{t\to\infty} \|\omega(t) - \omega_d(t)\| = 0 \tag{6.5}$$
$$\lim_{t\to\infty} \|\dot{\omega}(t) - \dot{\omega}_d(t)\| = 0 \tag{6.6}$$

If there is no collision. $x(t), R(t)$ are the position and orientation of the end effector, and $\dot{x}(t), \omega(t)$ are the linear and angular velocities. If we are in danger of colliding we want to achieve synchronization in the *feasible* movement direction. This means that we want to synchronize the robot as much as possible without colliding with an obstacle. If we denote x^f, R^f as the set of feasible trajectories which will not lead to a collision, then we want:

$$\lim_{t\to\infty} \|x^f(t) - x^f_d(t)\| = 0 \tag{6.7}$$
$$\lim_{t\to\infty} \|R^f(t) - R^f_d(t)\| = 0 \tag{6.8}$$
$$\lim_{t\to\infty} \|\omega^f(t) - \omega^f_d(t)\| = 0 \tag{6.9}$$
$$\lim_{t\to\infty} \|\dot{\omega}^f(t) - \dot{\omega}^f_d(t)\| = 0 \tag{6.10}$$

When we are in danger of a collision.

6.2 Projected synchronization error

We assume that the controller (5.19) has been applied to the system, and we are close to an obstacle such that $F_{rep} \neq 0$. We obviously cannot achieve perfect synchronization since we cannot at some point move further in the direction of F_{rep}. What we can achieve however is perfect tracking in the other directions directions orthogonal to F_{rep}. If we assume that $F_{rep} = n\|F_{rep}\|$ where n is the normal vector pointing out of the obstacle, then we can try to achieve synchronization in the tangent plane of n. This means that we don't care about the synchronization error along F_{rep} as it is infeasible. The orthogonal projection of the error vector field $\ddot{e} = f(e, \dot{e})$ is $\ddot{e}^{\perp_n} = f^{\perp_n}(e^{\perp_n}, \dot{e}^{\perp_n})$, see figure 6.1. We will write $e^{\perp_n} = e^{\perp}$ and drop the n in the superscript as n is the only vector which will be used for projection.

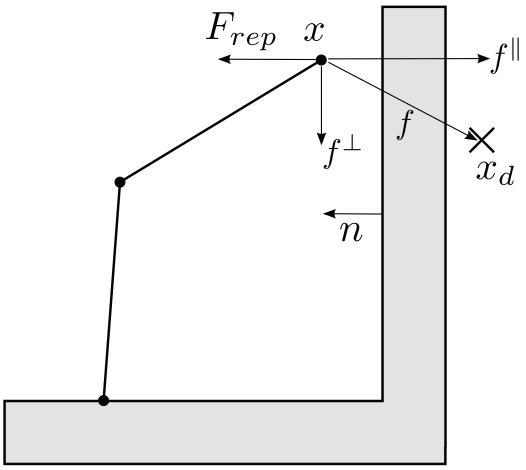


Figure 6.1: An illustration of the decomposition of our vector field in the vicinity of an obstacle.

6.2.1 The vector field $f^{\perp}$

We begin by finding the vector field perpendicular to the obstacle and we derive the conditions under which it is well behaved. The projection of the error e into the tangent plane with the normal vector n is:

$$e^{\perp} = e - [e^T n]n = S(n)e \tag{6.11}$$

We will use the transformation matrix S since it is notationally short. We differentiate this twice to obtain the projected dynamics.

$$\dot{e}^{\perp} = \dot{S}e + S\dot{e} \tag{6.12}$$

$$\ddot{e}^{\perp} = \ddot{S}e + 2\dot{S}\dot{e} + S\ddot{e} \tag{6.13}$$

When we are not close to an obstacle such that $F_{rep} = 0$, then we set $n = 0$ and the projection becomes the identity projection. We get the following if we assume that the normal vector is constant, or changing slowly such that $\dot{n} = \ddot{n} = 0 \Rightarrow \dot{S}(n) = \ddot{S}(n) = 0$:

$$\ddot{e}^{\perp} = S(n)\ddot{e} = S(n)f(e, \dot{e}) \tag{6.14}$$

This is applicable to flat surfaces such as walls, floors, slightly curved surfaces and is globally applicable if we choose to bound obstacles by bounding boxes, a common strategy in video game design. If we expand the simplified projected vector field (6.14) we get:

$$\ddot{e}^{\perp} = S(n)(-K_p e - K_d\dot{e} + n\|F_{rep}\|) = -S(n)K_p e - S(n)K_d\dot{e} + S(n)n\|F_{rep}\| \tag{6.15}$$

The repulsive force in the projected system is zero as can be seen by $S(n)n = n - (n^T n)n = 0$. This is and important step as we now don't need to worry about F_{rep} to show stability. The projected dynamics now look similar to a mass spring damper:

$$\ddot{e}^{\perp} = -S(n)K_p e - S(n)K_d\dot{e} \tag{6.16}$$

Which gives us a system dependent on the new state $(K_p e)^{\perp}$ and $(K_d\dot{e})^{\perp}$ which is not what we need since we need the states to be the same throughout the system. What we want is $\ddot{e}^{\perp} = f^{\perp}(e^{\perp}, \dot{e}^{\perp})$. We need to have K_p, K_d such that $S(n)Ke = \tilde{K}S(n)e$ where $\tilde{K}$ is a constant positive definite matrix. To find a solution for this we set $K = I^{3\times 3}k$:

$$S(n)Ke = S(n)kIe = kIS(n)e \tag{6.17}$$

We see that $\tilde{K} = K = kI^{3\times 3}$ where $k > 0$ is a scalar constant is a solution. We set $K_p = k_p I, K_d = k_d I$ to find our projected system:

$$\ddot{e}^{\perp} = -k_p S(n)e - k_d S(n)\dot{e} \tag{6.18}$$

Since $\dot{S} = 0$, we have from (6.12) that $\dot{e}^{\perp} = \dot{S}e + S\dot{e} = S\dot{e}$. This allows us to write the closed loop projected error as:

$$\ddot{e}^{\perp} = f^{\perp}(e^{\perp}, \dot{e}^{\perp}) = -k_p e^{\perp} - k_d \dot{e}^{\perp} \tag{6.19}$$

Which immediately tells us that the projected error system is exponentially stable for any scalar gain $k_p, k_d > 0$. The only assumption we imposed on the repulsive force F_{rep} was that it is pointing out of the obstacle. So we are assured projected synchronization even for a badly designed repulsion force as long as e is not unstable. We note that there are special cases where stability is ensured even for k_p, k_d not being scalar constants. This happens when the eigenbasis of S is the same as for the base coordinate system. We need to assume that the gains are scalar for the result to hold in general however.

So we now have successfully mapped the system to the task space, and we produced an exponentially decreasing tracking error in position if $F_{rep} = 0$ and a globally exponentially stable orientation error irrespective of F_{rep}. The fact that the repulsive force does not appear in the orientation system means that we can achieve perfect orientation tracking irrespective of any obstacles. It also means that the robot will not alter the pose of its spherical wrist in order to comply with the environment. This will limit the robots possible movement when avoiding a collision as it effectively now only has three degrees of freedom to use for obstacle avoidance. The choice however makes the stability analysis simple for the orientation.

6.3 The controller

To reiterate, the controller which achieves perfect orientation and projected position tracking in the presence of static planar obstacles assuming that F_{rep} is designed such that the system $f^{\parallel}$ stable, is:

$$u = C(q, \dot{q})\dot{q} + G(q) + M^{-1}J^{-1}(U - \dot{J}\dot{q}) \tag{6.20}$$

Where U is given by

$$U = \begin{bmatrix} \ddot{x}_d - k_p(x - x_d) - k_d(\dot{x} - \dot{x}_d) + F_{rep} \\ \dot{\omega}_d - \left(\frac{d}{dt}R(q)B(\Phi_e)\right)\dot{\Phi}_e + R(q)B(\Phi)(K_{po}\dot{\Phi}_e + K_{do}\Phi_e) \end{bmatrix} \tag{6.21}$$

Where $k_p, k_d \in \mathbb{R}^+$ and $K_{do}, K_{do} \in \mathbb{R}^{3\times 3}$ and are positive definite constant matrices. Φ_e is given from the expression:

$$\psi_e = \text{Atan2}(R^e_{32}, R^e_{33}) \tag{6.22}$$
$$\phi_e = \text{Atan2}(R^e_{21}, R^e_{11}) \tag{6.23}$$
$$\theta_e = \text{Atan2}(-R^e_{31}, \cos(\phi)R^e_{11} + \sin(\phi)R^e_{21}) \tag{6.24}$$

With $R^e = R^T(q)R_d(t)$.

$$B = \begin{bmatrix} 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 1 & 0 & -\sin(\theta) \end{bmatrix} \tag{6.25}$$

And $\frac{d}{dt}R(q)B(\Phi_e)$ can be shown by direct differentiation to be:

$$\frac{d}{dt}R(q)B(\Phi_e) = \dot{R}(q)B(\Phi_e) + R(q)\dot{B}(\Phi_e) = S(\omega)R(q)B(\Phi_e) + R(q)\dot{B}(\Phi_e) \tag{6.26}$$

Where:

$$\dot{B}(\Phi_e) = \begin{bmatrix} 0 & -\dot{\phi}_e\cos(\phi_e) & -\dot{\phi}_e\cos(\theta_e)\sin(\phi_e) - \dot{\theta}_e\cos(\phi_e)\sin(\theta_e) \\ 0 & -\dot{\phi}_e\sin(\phi_e) & \dot{\phi}_e\cos(\phi_e)\cos(\theta_e) - \dot{\theta}_e\sin(\phi_e)\sin(\theta_e) \\ 0 & 0 & -\dot{\theta}_e\cos(\theta_e) \end{bmatrix} \tag{6.27}$$

And we may find $\dot{\Phi}$ from inverting (5.14).

$$\dot{\Phi}_e = B^{-1}(\Phi_e)R(q)^T\omega_e \tag{6.28}$$

It is here that the nonsingularity of B is important.

Remarks

We have used projections on obstacle tangent planes in order to verify the stability of the closed loop system. It is important to note that no explicit projection is carried out in the controller. We simply add a repulsion force normal to the obstacle, add a scalar PD-gain, and then the stability results follow without us needing to perform any conditioning on the reference. This is a crucial property as no slow high level path planning or trajectory modification stage is included. The stability proof also provides us with a system describing the motion in the direction of the obstacle surface. It will be substantially easier to pick F_{rep} intelligently as this system tells us how the robot moves in the collision direction.

6.4 Synchronization over curved and rotating surfaces

We will now try to develop a controller equivalent to (6.90) for more complex surfaces.

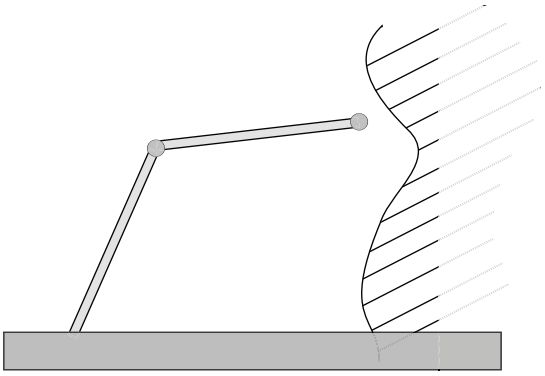


Figure 6.2: A robot in the vicinity of complicated moving or stationary obstacle.

This means that we no longer assume that the normal vector is constant. An obstacle with a nonconstant normal vector is sufficiently expressive for the representation of any moving rotation arbitrarily shaped obstacle. We will, as before, derive the dynamics in the tangent plane to the obstacle taken at the closest point between the end effector and the obstacle. The full dynamics of the projected vector field is given by:

$$\ddot{e}^{\perp} = \ddot{S}e + 2\dot{S}\dot{e} + S\ddot{e} \tag{6.29}$$

Our strategy is to find a new input $u_n = u + \gamma$ such that the closed loop system similar or identical to (6.19). The closed loop error system is then $\ddot{e} = -k_p e - k_d \dot{e} + F_{rep} + \gamma$. If we substitute this into (6.29) we get:

$$\ddot{e}^{\perp} = S[-k_p e - k_d \dot{e} + F_{rep} + \gamma] + 2\dot{S}\dot{e} + \ddot{S}e \tag{6.30}$$

If we expand we have:

$$\ddot{e}^{\perp} = -k_p Se - k_d S\dot{e} + S\gamma + 2\dot{S}\dot{e} + \ddot{S}e \tag{6.31}$$

We cannot say that $-k_d S\dot{e} = -k_d \dot{e}^{\perp}$ since since n is no longer constant. We add and subtract the term $k_p \dot{S}e$ to get $\dot{e}^{\perp}$ from (6.12).

$$\ddot{e}^{\perp} = -k_p Se - k_d[S\dot{e} + \dot{S}e] + k_d \dot{S}e + S\gamma + 2\dot{S}\dot{e} + \ddot{S}e \tag{6.32}$$

We can now write the desired projected system since $\dot{e}^{\perp} = S\dot{e} + \dot{S}e$:

$$\ddot{e}^{\perp} = -k_p e^{\perp} - k_d \dot{e}^{\perp} + k_d \dot{S}e + S\gamma + 2\dot{S}\dot{e} + \ddot{S}e \tag{6.33}$$

We write this in terms of the desired projected system $f^{\perp}$ and the undesirable disturbances F.

$$\ddot{e}^{\perp} = f^{\perp}(e^{\perp}, \dot{e}^{\perp}) + S\gamma + F(e, \dot{e}, n, \dot{n}, \ddot{n}) \tag{6.34}$$

Where the vector F is zero when the the normal vector is constant. It is hard to say anything about the stability of this system as F is nonlinear. Our aim is to find a γ such that $S(n)\gamma + F = 0$. The equation we need to solve is:

$$S\gamma = -F(e, \dot{e}, n, \dot{n}, \ddot{n}) \tag{6.35}$$

Since S is singular for all n then equation (6.35) has no solutions, and the strategy seemingly failed. The reason that this equation is unsolvable however is the fact that we are representing a projected vector field into $\mathbb{R}^2$ by vectors in $\mathbb{R}^3$. So our representation is redundant as one of the three equations of (6.29) are linearly dependent upon two others. So we will need to

remove this redundancy in order to proceed. If we go back to the beginning, we know that:

$$e^{\perp} = S(n)e, \quad e \in \mathbb{R}^3, \quad e^{\perp} \in \mathbb{R}^2 \tag{6.36}$$

Since we are using a vector in $\mathbb{R}^3$ to describe the projected dynamics which actually is in $\mathbb{R}^2$, we should be able to express the projected dynamics by its minimal representation. We will see that a time dependent equivalence transformation will suffice for this purpose. The transformation we want is the following:

$$e^{\perp} = S(n)e, \quad e_m^{\perp} = P(t)e^{\perp} \quad \text{such that} \quad e_m^{\perp} = \begin{bmatrix} e_{m_1}^{\perp} \\ e_{m_2}^{\perp} \\ 0 \end{bmatrix} \tag{6.37}$$

An illustration of the transformation is seen in figure 6.3

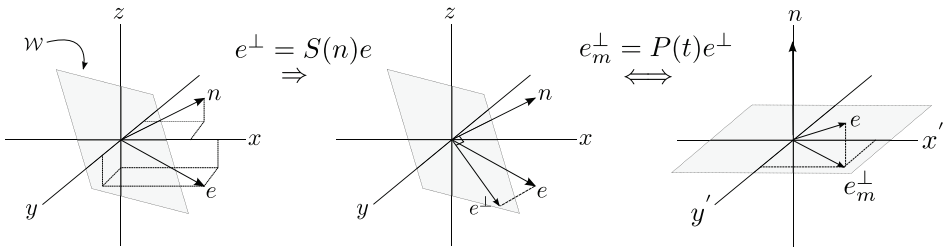


Figure 6.3: The transformation we need to find a minimal representation of $\mathcal{W}$ which is the tangent space of n.

One such transformation is a rotation which rotates the projected system onto its own basis, making one of its elements zero. A seemingly more elegant approach is to use the diagonalization of S to express $e^{\perp}$ is the eigenspace of $S(n)$. We know that it is possible to diagonalize S for all $S \neq 0$. This is not a problem for us since if $S = 0$, then we are not in the vicinity of an obstacle, and we do not need to perform any projection. The diagonalization of S may be written without loss of generality as:

$$S(n) = E(n)\Lambda E^{-1}(n) = E(n) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} E^{-1}(n) \tag{6.38}$$

Where Λ is a diagonal matrix containing the eigenvalues, being either zero or one, and E is the eigenvector matrix. The choice to place the zero eigenvalue in the bottom row is arbitrary. Placing it anywhere else will not change the result. We will see that using the eigenvectors or a rotation matrix will show the same result. So if we premultiply (6.39) by E^{-1} we have:

$$e^{\perp} = S(n)e = E(n)\Lambda E^{-1}(n)e \quad \Longleftrightarrow \quad E^{-1}(n)e^{\perp} = \Lambda E^{-1}(n)e \tag{6.39}$$

If we call the new projected error $e_m^{\perp} = E^{-1}(n)e^{\perp}$ we have that $e_{m_3}^{\perp} = 0$. The steps involved in this transformation is first a projection, then we multiply the system by an invertible matrix, which is an equivalence transformation. We have succeeded in eliminating the dependent degree of freedom. We may now explicitly remove this zero row and construct a new reduced system. We will however keep the zero row in order to keep our matrices square. The choice has no impact on the result. The minimal representation of the projected matrix is given by:

$$e_m^{\perp} = \Lambda E^{-1} e \tag{6.40}$$

We differentiate this twice to find a non redundant expression for the projected dynamical system.

$$\dot{e}_m^{\perp} = \Lambda[\dot{E}^{-1}e + E^{-1}\dot{e}] \tag{6.41}$$

$$\ddot{e}_m^{\perp} = \Lambda[\ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + E^{-1}\ddot{e}] \tag{6.42}$$

We substitute the closed loop system with the added input γ to get:

$$\ddot{e}_m^{\perp} = \Lambda E^{-1}(-k_p e - k_d \dot{e} + F_{rep} + \gamma) + \Lambda[\ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e}] \tag{6.43}$$

We need to add and subtract $k_d \Lambda \dot{E}^{-1} e$ to find the desired vector field in closed form.

$$\ddot{e}_m^{\perp} = -k_p \Lambda E^{-1} e - k_d(\Lambda E^{-1}\dot{e} + \Lambda \dot{E}^{-1} e) + k_d \Lambda \dot{E}^{-1} e + \Lambda[E^{-1}\gamma + \ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e}] \tag{6.44}$$

Where we also use $\Lambda E^{-1} F_{rep} = 0$. This simplifies to:

$$\ddot{e}_m^{\perp} = -k_p e^{\perp_m} - k_d \dot{e}_m^{\perp} + \Lambda[E^{-1}\gamma + \ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + k_d\dot{E}^{-1}e] \tag{6.45}$$

We can factor out Λ such that we now need to solve the equation:

$$E^{-1}\gamma = -\ddot{E}^{-1}e - 2\dot{E}^{-1}\dot{e} - k_d\dot{E}^{-1}e \tag{6.46}$$

It is possible to solve this directly since we now have E^{-1} instead of S as before. Since we have E readily available as the eigenvector matrix of S we solve the equation as:

$$\gamma = -E(\ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + k_d\dot{E}^{-1}e) \tag{6.47}$$

Or if we used a rotation matrix R_n we would have:

$$\gamma = -R_n(\ddot{R_n}^T e + 2\dot{R_n}^T\dot{e} + k_d\dot{R_n}^T e) \tag{6.48}$$

And we have found the γ we need to achieve closed loop stability. The closed loop system is now given by:

$$\ddot{e}_m^{\perp} = -k_p e_m^{\perp} - k_d\dot{e}_m^{\perp} \tag{6.49}$$

where $\ddot{e}_{m_3}^{\perp} = 0$ for all $e(t), n(t)$. We note that $\gamma = 0$ for $n = 0$ such that this new feedback term will not interfere when we are not in danger of colliding with an obstacle.

We can now seemingly achieve perfect position synchronization orthogonal to any obstacles given that the Jacobian is nonsingular, B is nonsingular and that the system $f^{\parallel}$ is stable.

6.4.1 Feasibility considerations

Even though it might not seem evident, the derived controller will only be implementable given planar or slightly curved surfaces. This is because we use the acceleration of the normal vector of F_{rep} in the controller. This will be a function of the obstacles acceleration as well as the robots acceleration given a curved surface. Since the robots acceleration is a function of γ, then our closed loop system is now no longer given in state space form. The dynamics are now given as an implicit equation in $\ddot{e}$ which will manifest itself discrete

implementation as a recurrence equation also called an algebraic loop. If we assume that $\ddot{e}$ was available for us to use in control, then we could derive nonsensical results. Consider for instance the closed loop system:

$$\ddot{e} = u \tag{6.50}$$

If we use the input $u = \ddot{e} - e$ we get the algebraic equation:

$$e = 0 \quad \forall \quad t \tag{6.51}$$

Which is clearly not feasible in any physical system. The best we can hope for is that the term $\ddot{E}^{-1}e$ is small such that it will not perturb our solutions too much.

6.4.2 Remarks

The choice for the equivalent transform mapping the projected system to a minimal representation is not unique. Generalized inverse solutions operating directly on S could also be employed. We chose the diagonalization as a basis for our transformation given its simplicity. We could also find γ using generalized inverse solutions of S, this would be less intuitive. Our choice will however not have an impact on the result.

6.5 The controller

To reiterate once again, assuming that F_{rep} is designed such that the system $f^{\|}$ stable, then the controller which achieves perfect orientation and projected position tracking in the presence of a moving and rotating planar obstacles is:

$$u = C(q, \dot{q})\dot{q} + G(q) + M^{-1}J^{-1}(U - \dot{J}\dot{q}) \tag{6.52}$$

Where U is given by

$$U = \begin{bmatrix} \ddot{x}_d - k_p(x - x_d) - k_d(\dot{x} - \dot{x}_d) + F_{rep} + \gamma \\ \dot{\omega}_d - \left(\frac{d}{dt}R(q)B(\Phi_e)\right)\dot{\Phi}_e + R(q)B(\Phi)(K_{po}\dot{\Phi}_e + K_{do}\Phi_e) \end{bmatrix} \tag{6.53}$$

And γ is given as:

$$\gamma = -E(n)(\ddot{E}^{-1}(n)e + 2\dot{E}^{-1}(n)\dot{e} + k_d\dot{E}(n)^{-1}e) \tag{6.54}$$

Where $E(n)$ is a matrix containing the eigenvectors of the projection matrix $S(n)$.

6.6 The vector field $f^{\|}$

We know that it is possible to achieve partial synchronization in the presence of an obstacle provided that the system $f^{\|}$ is stable. This system describes the dynamics in the direction of F_{rep}. We will call this system $\ddot{e}^{\|}$.

6.6.1 $f^{\|}$ for planar obstacles

We will start by assuming that $\dot{n} = \ddot{n} = 0$. The full system $\ddot{e} = \ddot{e}^{\perp} + \ddot{e}^{\|}$ is stable if and only if both $\ddot{e}^{\perp}$ and $\ddot{e}^{\|}$ are stable. We know this since they are orthogonal systems. We have that:

$$\ddot{e}^{\|} = (I - S(n))\ddot{e} = S_{\|}(n)\ddot{e} \tag{6.55}$$

We substitute in the vector field f, F_{rep} is invariant under a projection onto itself and remains unchanged.

$$\ddot{e}^{\|} = S_{\|}(-k_p e - k_d\dot{e} + F_{rep}) = -k_p e^{\|} - k_d\dot{e}^{\|} + F_{rep} \tag{6.56}$$

We use the diagonalization of $S_{\|}$ to map the system such that it is explicitly scalar. We can take the projection matrix assuming $n_3 \neq 0$:

$$\Lambda E^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -n_1 n_2 & n_1^2 + n_3^2 & -n_2 n_3 \\ \frac{-n_2^2 + n_3^2}{n_3} & \frac{n_1 n_2}{n_3} & n_1 \\ n_1 & n_2 & n_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ n^T \end{bmatrix} \tag{6.57}$$

The condition that $n_3 \neq 0$ does not limit the generality because if $n_3 = 0$ then we may use another projection matrix giving the same result. The projected system now has the particularly simple form, the single nonzero row of $\ddot{e}_m^{\perp}$ is:

$$\ddot{e}_m^{\parallel} = n^T f(e, \dot{e}) + n^T F_{rep} = -k_p e_m^{\parallel} - k_d \dot{e}_m^{\parallel} + n^T F_{rep} \tag{6.58}$$

Where $-k_p e_m^{\parallel} \in \mathbb{R}$, see figure 6.4.

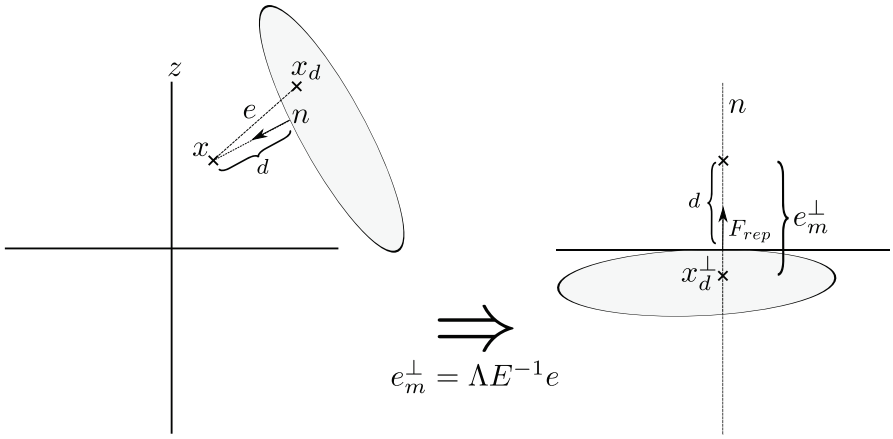


Figure 6.4: The projection and similarity transformation aligning the normal vector with the z-axis making the system $f^{\parallel}$ scalar.

The stability of this system is not that easy to analyze since it is time varying and nonlinear and it depends upon our particular choice of repulsion force. If we use the repulsive force F_{rep} which is only a function of the distance to the obstacle given in (5.1) such that whenever we are close enough to an obstacle we have:

$$F = \left(\frac{1}{d(x)} - \frac{1}{\rho_0} \right) \frac{1}{d(x)^2} \tag{6.59}$$

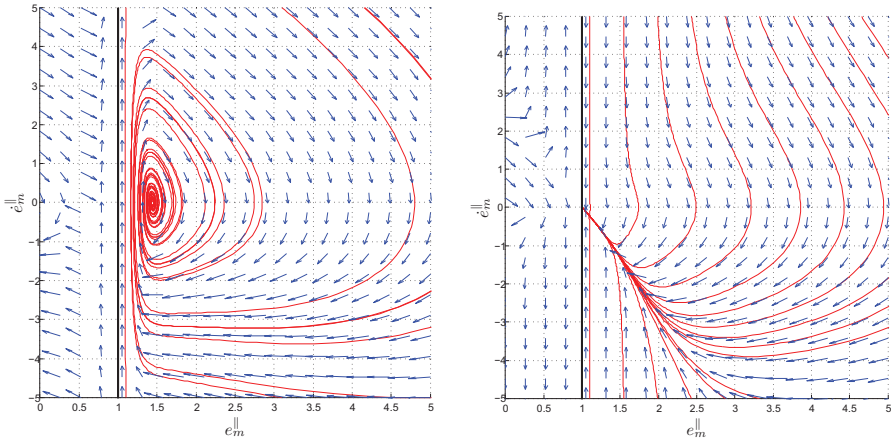
Where $d(x)$ is the distance to the obstacle as a function of the end effector position x and $e = x - x_d$. Since the shortest distance to the obstacle also is colinear with n we may write:

$$F = \left(\frac{1}{d(x_m)} - \frac{1}{\rho_0} \right) \frac{1}{d(x_m)^2} \tag{6.60}$$

Where x_m is a scalar. The only interesting case we need to consider is when $-k_p e_m^{\parallel} - k_d \dot{e}_m^{\parallel}$ is negative such that we are on a collision course. $n^T F_{rep}$ will be greater than or equal to zero for all collision free initial conditions.

6.6.2 Qualitative analysis of $f^{\parallel}$

We compare the response of the two potential fields (5.1) and (5.4) on our scalar system: The obstacle in the following plots is static and is situated at $x = 1$, and we plot the error $e = x - x_d$ for a constant reference, see figure 6.5. The time invariance is needed to keep the plots in 2D. Simulation with time varying a reference and obstacle are qualitatively similar.



(a) The end effector position and velocity error using (5.1) showing an oscillatory convergence on a point some distance away from the obstacle.

(b) The end effector position and velocity error using (5.4) showing a convergence to a point on the obstacle while slowing down linearly with the distance.

Figure 6.5: Different response for the two proposed repulsive forces.

The collision free equilibrium point for the distance dependent repulsive force is the solution e_s to:

$$-k_p e_s + \eta_i \left(\frac{1}{d(e_s)} - \frac{1}{\rho_0} \right) \frac{1}{d(e_s)^2} = 0 \tag{6.61}$$

Which is unique and approaches the obstacle as $k_p \to \infty$. It is also linearly stable with complex eigenvalues with negative real part. We also note that for any initial condition arbitrarily close to the boundary we have an arbitrarily larger positive acceleration assuring that we cannot cross the collision boundary unless we start on the boundary. There are in other words no solutions bringing our system to a collision.

The analysis for the velocity dependent repulsive force is slightly more involved. To see why the approach to the obstacle using the velocity dependent repulsive force is the way it is, we must look at the equilibrium of the system:

$$\ddot{e} = -k_p e - k_d \dot{e} + \eta_i \left(-\frac{\dot{d}(t)}{d(t)} - \frac{1}{T_0} \right) \frac{\dot{d}^2(t)}{d^2(t)} \tag{6.62}$$

Where we denote e as the scalar error projected along the normal vector. If we assume that solutions to this system exist, and solution converge on some equilibrium then we must have a real steady state e_s such that:

$$\lim_{e,\dot{e}\to 0} -k_p e - k_d \dot{e} + \eta_i \left(-\frac{\dot{d}(t)}{d(t)} - \frac{1}{T_0} \right) \frac{\dot{d}^2(t)}{d^2(t)} = 0 \tag{6.63}$$

If we substitute in $\dot{e} = \dot{d} = 0$, then this means that $e = 0 \Rightarrow x = x_d$, which means that we have collided. We know that no solutions exist passing through the obstacle however, so this implies that both $d \to 0$ and $\dot{d} \to 0$, so we need to take some care in evaluating the limit. If we convert the limit for to polar coordinates in d taking $d = r\cos(\theta)$ and $\dot{d} = r\sin(\theta)$ and let r go to zero, we find:

$$\lim_{r\to 0} F_{ref}(d, \dot{d}) = -k_p e_s + \eta_i \left(-\frac{\sin(\theta)}{\cos(\theta)} - \frac{1}{T_0} \right) \frac{\sin^2(\theta)}{\cos^2(\theta)} = 0 \tag{6.64}$$

This limit is independent of r and does not exist in the normal sense of limits in $\mathbb{R}^2$. But we have assumed that a solution does exist, so we need to consider the dependence of the angle of approach θ. This means we have to find a θ such that:

$$k_p e_s + \eta_i \left(\tan(\theta) + \frac{1}{T_0} \right) \tan^2(\theta) = 0 \tag{6.65}$$

There is one real solution to this equation which is dependent on k_p, T_0 and η_i. All the solution however fall in the region $\theta \in [0, -\pi/4]$ such that the trajectories are slowing down as they approach the obstacles. We have that $\theta = -\pi/4$ for $k_p/\eta_i \to \infty$ and $\theta = 0$ for $T_0 \to \infty$. These correspond to the limiting cases of extremely fast approach and extremely slow approach. So assuming that a solution to (6.62) exists with a collision free initial condition, then solutions approach the closest possible collision free point on the obstacle

as $t \to \infty$. We will in fact collide with the obstacle at $t = \infty$, but this happens however with zero speed relative to the obstacle.

We conclude that the system $f^{\|}$ with collision free initial conditions will not result in a collision and is stable since all other scenarios are inherently collision free.

6.7 $f^{\|}$ for general obstacles

We consider the possibility of a collision when n is not constant. The minimal representation of the error along n was found to be:

$$e_m^{\|} = n^T e \tag{6.66}$$

And the dynamics in $\mathbb{R}$ is given by:

$$\ddot{e}_m^{\|} = \ddot{n}^T e + 2\dot{n}^T e^T - k_p e_m^{\|} - k_d \dot{e}_m^{\|} + n^T F_{rep} \tag{6.67}$$

We note that $n^T \gamma = 0$ such that our synchronization error does not interfere with the dynamics in the direction of the obstacle. We may now attempt to remove these disturbances using another control input. This would be required if one wants to design an approach trajectory to the obstacle for manipulating a changing environment. We only want to avoid the obstacle however, so we will not do this. We will limit our analysis by assuming that n is continuously changing, such that $\|\dot{n}\|$ and $\|\ddot{n}\|$ are bounded function of time. It is now apparent that a collision is impossible using the same argument as for a constant normal vector.

6.7.1 Example

We see how the method applies to a planar elbow robot in the vicinity of a curved and/or moving obstacle. One such scenario is where a robot is carrying out some task in the vicinity or on a box dangling from a cable, see figure 6.6.

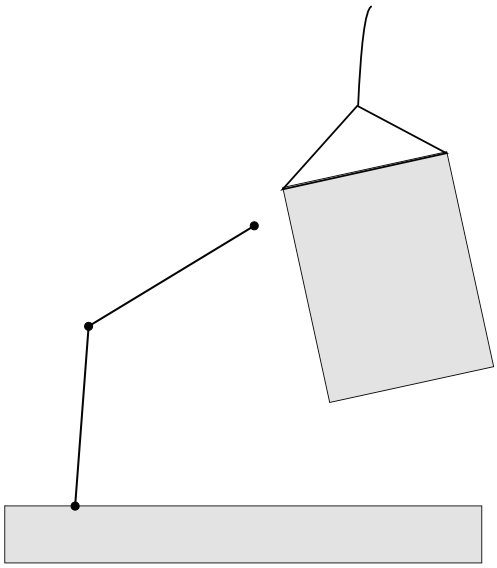


Figure 6.6: A robot operating in the vicinity of a moving box hanging from a cable.

If we ignore the orientation as this a two-valued function of the position, then the geometric Jacobian of the elbow manipulator is quadratic. This makes the elbow manipulator conceptually analogous to our full 6DOF robot in the plane as it is non redundant. The closed loop system after feedback linearization and task space mapping is:

$$\ddot{e} = -k_p e - k_d \dot{e} + Fn + \gamma \tag{6.68}$$

We identify the projection matrix S as:

$$S(n) = \begin{bmatrix} n_2^2 & -n_1 n_2 \\ -n_1 n_2 & n_1^2 \end{bmatrix} \tag{6.69}$$

The eigenvector matrix $E(n)$ and the eigenvalue matrix Λ is given by:

$$E(n) = \begin{bmatrix} n1 & -n2 \\ n2 & n1 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{6.70}$$

If we check, we will find that $E(n) \in SO(2)$[1] The minimal projection is now given as:

$$\Lambda E^{-1} = \begin{bmatrix} -n_2 & n1 \\ 0 & 0 \end{bmatrix} \tag{6.71}$$

So $e^{\perp}_{m_2} = 0$ as desired. The minimal representation of the projected error into n in the vicinity of the obstacle is now a scalar.

$$e^{\perp}_m = S_r(n)e = \Lambda E^{-1} e = \begin{bmatrix} e_2 n_1 - e_1 n_2 \\ 0 \end{bmatrix} \tag{6.72}$$

We get the closed loop projected system by differentiating (6.72) twice:

$$\ddot{e}^{\perp}_m = \Lambda[\ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + E^{-1}\ddot{e}] \tag{6.73}$$

$$\ddot{e}^{\perp}_m = \begin{bmatrix} -\ddot{n}_2 & \ddot{n}_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + 2 \begin{bmatrix} -\dot{n}_2 & \dot{n}_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} -n_2 & n_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{e}_1 \\ \ddot{e}_2 \end{bmatrix} \tag{6.74}$$

If substitute in our closed loop system $\ddot{e} = -k_p e - k_d \dot{e} + F_{rep} + \gamma$ we have:

$$\ddot{e}^{\perp}_m = \Lambda[\ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + E^{-1}(-k_p e - k_d \dot{e} + F_{rep} + \gamma)] \tag{6.75}$$

We remove the repulsive force as $\Lambda E^{-1} F_{rep} = 0$ and we collect the terms we want to keep for the projected system.

$$\ddot{e}^{\perp}_m = f^{\perp}_m(e^{\perp}_m, \dot{e}^{\perp}_m) + \Lambda[k_d \dot{E}^{-1}e + \ddot{E}^{-1}e + 2\dot{E}^{-1}\dot{e} + E^{-1}\gamma] \tag{6.76}$$

Where the first row of $f^{\perp}_m$ is given in terms of e and n as:

$$f^{\perp}_{m_1} = k_p(e_1 n_2 - e_2 n_1) + k_d(\dot{n}_2 e_1 - \dot{n}_1 e_2 - \dot{e}_2 n_1 + \dot{e}_1 n_2) \tag{6.77}$$

[1] Actually we have $det(E) = -1$, but we may choose to multiply one of the rows of E by -1 such that $det(E) = 1$ since the eigenvectors are invariant under scaling

And written in terms of $e_m^{\perp}$:

$$f_{m_1}^{\perp} = -k_p e_{m_1}^{\perp} - k_d \dot{e}_{m_1}^{\perp} \tag{6.78}$$

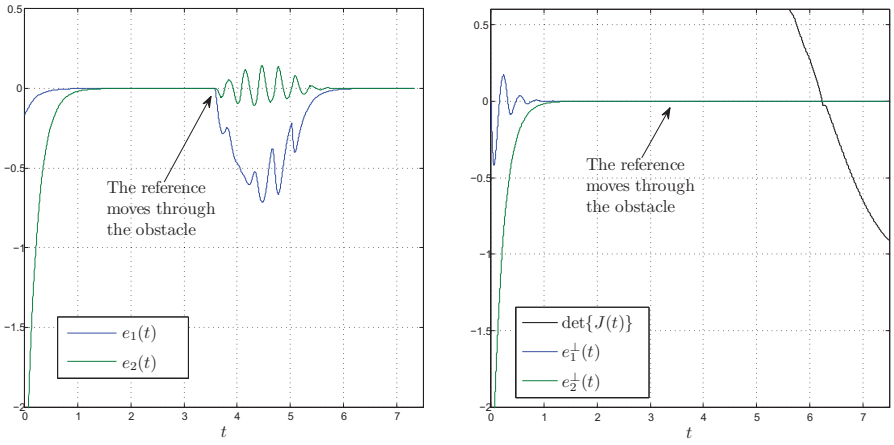
The rest of the terms cancel out given our choice of γ:

$$k_d \dot{E}^{-1} e + \ddot{E}^{-1} e + 2\dot{E}^{-1}\dot{e} + E^{-1} \underbrace{[-E(\ddot{E}^{-1}(n)e + 2\dot{E}^{-1}(n)\dot{e} + k_d \dot{E}(n)^{-1} e)]}_{\gamma} = 0 \tag{6.79}$$

Which is what we wanted. Perfect tracking perpendicular to the obstacle is ensured for any moving and rotating box. And the controller is given in closed form as:

$$u = G(q) + C(q,\dot{q})\dot{q} + M(q)(J^{-1}(\ddot{x}_d - k_p(x - x_d) - k_d(\dot{x} - \dot{x}_d) + F_{rep} + \gamma - \dot{J}\dot{q})) \tag{6.80}$$

We note that the control input γ only produces gain perpendicular to an obstacle when the end effector is in the vicinity of an obstacle. Simulation results are seen in 6.7.



(a) The end effector position error.

(b) The projected end effector position error. The robot moves through a singularity where J is singular. The damped inverse has been used to prevent a crash.

Figure 6.7: The end effector error while following a trajectory moving through an obstacle where rotating and translating obstacle.

Snapshots of a simulator is shown in figure 6.8.

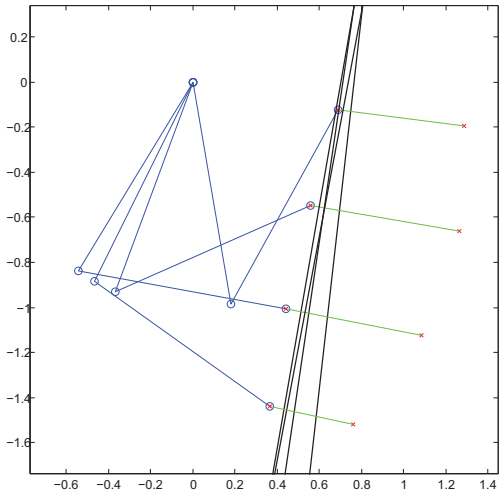


Figure 6.8: The elbow manipulator at different time instances. The reference is represented by a red cross, the end effector is alway in synchrony projected into the moving wall.

6.8 The minimal representation of the obstacle tangent space

We have seen how one may use the eigenvectors of the projection matrix to express the dynamics on a tangent plane using only two parameters. The proof will however work with any such mapping, and we therefore present two alternative representations. The first relies on explicitly rotating the normal vector to align with the z-axis in the world frame. This can be achieved with the following operation given n:

$$\theta_1 = \text{Atan2}(n_1, n_2) \tag{6.81}$$
$$n^{'} = R_z(\theta_1)n \tag{6.82}$$
$$\theta_2 = \text{Atan2}(n^{'}_2, n^{'}_3) \tag{6.83}$$
$$n^{''} = R_x(\theta_2)n^{'} \tag{6.84}$$

Now $n^{''} = [0\,0\,1]^T$ and $R_x(\theta_2)R_z(\theta_1)$ is the rotation matrix aligning the tangent space basis with the standard orthonormal basis in $\mathbb{R}^3$. We may compute a simple expression for this matrix if we use arctan instead of the two argument inverse tangent function:

$$R(n) = \begin{bmatrix} \frac{n_2}{\sqrt{n_1^2+n_2^2}} & \frac{-n_1}{\sqrt{n_1^2+n_2^2}} & 0 \\ \frac{n_1 n_3}{\sqrt{n_1^2+n_2^2}} & \frac{n_2 n_3}{\sqrt{n_1^2+n_2^2}} & -\sqrt{n_1^2+n_2^2} \\ n_1 & n_2 & n_3 \end{bmatrix} \tag{6.85}$$

This matrix is not defined for $n = [0\,0\,1]^T$, but we can make the definition consistent if we define $R([0\,0\,1]^T) = I$. Using this we would write the projected error as

$$e_m^{\perp} = R(n)S(n)e \tag{6.86}$$

If we use the eigenvector directly, we need to compute them, for the zero eigenvalue we have:

$$S(n)v_0 = 0 \tag{6.87}$$

Which has the solution $v_0 = n$. For the one eigenvectors we have:

$$(S(n) - I)v = \begin{bmatrix} -n_1(n_1v_1 + n_2v_2 + n_3v_3) \\ -n_2(n_1v_1 + n_2v_2 + n_3v_3) \\ -n_3(n_1v_1 + n_2v_2 + n_3v_3) \end{bmatrix} = 0 \tag{6.88}$$

Which tells us that any two linearly independent vectors v that satisfies $n \cdot v = 0$ will satisfy as eigenvectors. Since the rotational matrix R expresses a basis for n and the tangent space of n, then see that R^T is an eigenvector matrix of S.

6.9 Full robot collision for general obstacles

We have up to now only considered the case where the end effector interacts with the workspace obstacle. We obviously need a controller which will prevent the entire robot from colliding. We will show that this seemingly more complicated problem is conceptually identical to the end-effector collision problem which we have solved. We know from section 5.9.1 how to map a any force to the end effector.

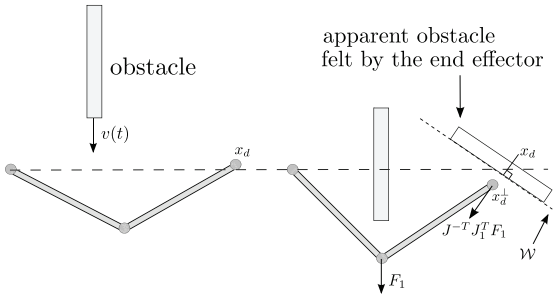


Figure 6.9: The robot seen from above is pushed sideways by an obstacle moving towards its second joint. The obstacle force is mapped to the end effector for which it will seem as an object is pushing it back. Synchronization in the tangent plane to this apparent obstacle is achieved.

The closed loop system is for this case given by:

$$\ddot{e} = -k_p e - k_d \dot{e} + J^{-T} J_p^T F_p \tag{6.89}$$

Our system now looks similar to the ones we have solved, apart from two things. The force now also appears in the orientation system since the three bottom elements of $J^{-T}J_p^T F_p$ are not in general zero. So we cannot as easily state any asymptotic stability property on the orientation. This will appear in the equation as an apparent torque acting on the end effector. The force which acts on the end effector is now not perpendicular to the actual obstacle, but is a complex function of the obstacle as well as the robots geometry and pose.

It is not as geometrically intuitive to apply projection mappings to the orientation system in order to achieve partial synchronization as it is for the position system. The form of the two equations are however identical, and we will handle it identically to the position system. This means that if we take the normal vector of F_o, and apply the projection transformation to it as we did for the position, all the results apart from the stability analysis of $f^{\|}$ hold. What we now are achieving partial synchronization with respect to is the feasible movement directions and feasible orientations for the robot. In other words, the robot will move towards the synchronization objective along some apparent curved surface which describes the boundary in which it can move without a collision in $\mathbb{R}^6$.

A question one might ask is whether or not this is the behavior we want. We will achieve synchronization tangent to some apparent obstacle which is a function of the robots pose and geometry, which will not necessarily transfer to any intuitive geometrical partial synchronization objective as before. Another question is whether or not we can do any better than this.

6.9.1 Full robot collision avoidance synchronization controller

Assuming that $f^{\|}$ is stable we have the following controller which will achieve partial synchronization for full robot collision avoidence. Since we now are concerned with the projected movement also in orientation we need to limit K_{po}, K_{do} to be scalars k_{po}, k_{do}. The controller in full is given by:

$$u = C(q,\dot{q})\dot{q} + G(q) + M^{-1}J^{-1}(U - \dot{J}\dot{q}) \tag{6.90}$$

If we write $J^{-T}J_pF_p = [W_p \quad W_o]^T$ the U is given by:

$$U = \begin{bmatrix} \ddot{x}_d - k_p(x - x_d) - k_d(\dot{x} - \dot{x}_d) + W_p + \gamma \\ \dot{\omega}_d - \left(\frac{d}{dt}R(q)B(\Phi_e)\right)\dot{\Phi}_e + R(q)B(\Phi)(k_{po}\dot{\Phi}_e + k_{do}\Phi_e + B^{-1}(\Phi)R^{-1}(q)W_o + \beta) \end{bmatrix} \tag{6.91}$$

γ is given as:

$$\gamma = -E(n)(\ddot{E}^{-1}(n)e + 2\dot{E}^{-1}(n)\dot{e} + k_d\dot{E}(n)^{-1}e) \tag{6.92}$$

Where $E(n)$ is a matrix containing the eigenvectors of the projection matrix $S(n)$ where $n = W_p/\|W_p\|$ and $n = 0$ for $W_p = 0$.

β is given as:

$$\beta = -E(n_o)(\ddot{E}^{-1}(n_o)\Phi_e + 2\dot{E}^{-1}(n_o)\dot{\Phi}_e + k_d\dot{E}(n_o)^{-1}\Phi_e) \tag{6.93}$$

Where $E(n_o)$ is a matrix containing the eigenvectors of the projection matrix $S(n_o)$ where $n_o = B^{-1}(\Phi)R^{-1}(q)W_o/\|B^{-1}(\Phi)R^{-1}(q)W_o\|$. and $n_o = 0$ for $B^{-1}(\Phi)R^{-1}(q)W_o = 0$.

6.10 Multiple obstacle interaction

We have seen how one can achieve synchronized tracking in the vicinity of an obstacle. What we have yet to discuss is the case where there robot comes in conflict with more than one obstacle. Consider a case where the end effector reference has moved into a corner, such that two repulsive forces are added in the controller, $F_{rep} = F_{wall} + F_{floor}$. See figure 6.10 for an illustration.

If we try to synchronize the end effector in the plane perpendicular to F_{rep} then we see that we can not succeed since we cannot freely move in this plane. What we need to do in order to find a projected system where we can guarantee synchronization is to project the system into the wall *and* the floor. if we denote the normal vector to the wall as n_w and the vector normal to the floor as n_f then we use the mapping:

$$e^{\perp_{n_w,n_f}} = S(n_w)S(n_f)e = S(n_f)S(n_w)e \tag{6.94}$$

We loose one degree of freedom for each linearly independent contact force. If two forces were parallel, between two walls, then we only loose one DOF.

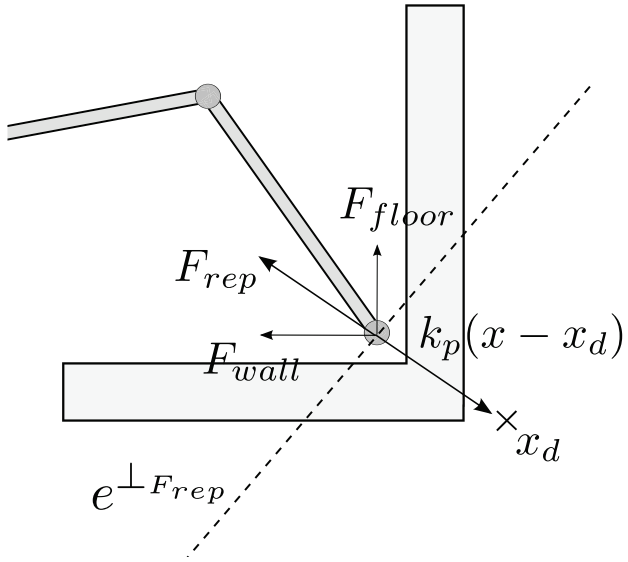


Figure 6.10: The end effector has reached a corner and is stopped by virtual forces excreted on the manipulator by the wall and the floor. The tangent plane/line of the sum of the forces is indicated by the dotted line.

If the robot in figure 6.10 is operating in $\mathbb{R}^3$, with basis vectors $\{i, j, k\}$ then we have the projected system which is now given in $\mathbb{R}$:

$$\ddot{e}^{\perp_{i,j}} = \ddot{e}_k = f^{\perp_{i,j}}(e_k, \dot{e_k}) \tag{6.95}$$

Such that we can achieve synchronization along the $z-$axis. The proof of this follows directly by iteratively applying the method given in 6.4 to the system. If we approach a corner in of a room, then we are restricted by three linearly independent forces, which means that the position synchronization error $e^{\perp}$ is zero and $\ddot{e}^{\perp} = 0$. Our projected state space has been reduced to the trivial vector space consisting only of the zero vector. This means that we cannot move in any direction with a decreasing position error. By our measure however, perfect synchronization is achieved. This example illustrates the usefulness of our interpretation as we can achieve perfect projected synchronization even when we cannot move in any directions. Orientation synchronization may however still be feasible. For the system $f^{\parallel}$ it however suffices to consider the sum of the forces and our previous arguments will suffice.

6.11 Self collision

One important collision scenario we have yet to discuss is the possibility of the robot colliding with itself. It is important to account for self collision in order to keep the robot from damaging itself. This is an easier problem to tackle than the ones we have previously discussed for the following reason. The geometry of the robot is known and is constant, this means that we may calculate the global joint space obstacle in closed form represented by joint angle intervals. Self collision avoidance may be handled in hardware, or on a low control level as joint saturation. We may also solve this problem using artificial repulsive forces. It is simplest to tackle this in joint space since the self collision problem is inherently given in this space. But it is possible to map the controller to the task space. We consider a joint q_i and its collision free set given in joint space as:

$$q_{i_{\text{free}}} \in (\, q_{i_{\min}}(q), \quad q_{i_{\max}}(q) \,) \tag{6.96}$$

We use the parentheses to denote exclusion. Note that the limits for the joint angle may be constant or a more complex function of other joint angles and the robots geometry. We construct the following new control input u_c : after feedback linearization

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u \quad \Longleftrightarrow \quad \ddot{q} = u_2 + u_c \tag{6.97}$$

Where an element u_{c_i} of u_c is given by:

$$u_{c_i} = -\eta_i \left(\frac{1}{|q_i - q_{i_{\min}}|} - \frac{1}{\rho_0} \right) \frac{1}{(q_i - q_{i_{\min}})^2} + \eta_i \left(\frac{1}{|q_i - q_{i_{\max}}|} - \frac{1}{\rho_0} \right) \frac{1}{(q_i - q_{i_{\max}})^2} \tag{6.98}$$

And the two terms are zero if $|q_i - q_{i_{\min/\max}}| > \rho_0$. $\eta_i > 0$ is scalar gain and ρ_0 defines a distance where the control input is turned on. Notice that the first term is negative. This is all we need to define the direction of the control gain. This control input may be mapped to the task space by multiplying with the inverse transposed Jacobian.

Another approach which is more compatible with the previously developed strategy is the following: Construct an artificial force between two links in danger of collision, then map this to the end effector, see figure 6.11.

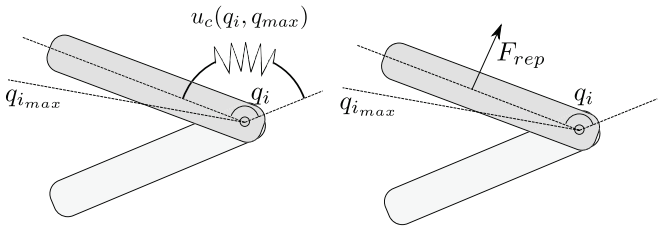


Figure 6.11: Two different approaches to avoiding self collision. The explicit joint difference is used on the left. Some repulsive force is constructed on the right.

We now also have a choice of exactly where we wish to place the force. This design choice between these two approaches is again up to the designer. The second approach fits nicely into the developed framework, while the first is simpler and may be easier to tune. Note that the joint force u_c is mapped to the task space as a force, such that these two approaches may be considered equivalent.

6.12 Similarities with force control

The method proposed in this chapter for collision avoidance is largely similar to techniques used in the field of force motion control. In force control one actually wants to crash, and then one impose limitations on the movement of the end effector through constraints imposed using control inputs. Differences between our solution and a typical force motion controller is the following: We have full control of the forces which are applied to the robot as we generate them. This gives us flexibility in terms of design freedom. One does not have this freedom in force control as the force is something "real". We do not impose any explicit constraints on our robot through control. We solve the problem of constrained motion through imposing restrictions on the PD-controller used in the inner control loop. We also have the possibility of achieving synchronized motion close to a moving surface using a feed forward term. Given the similarities between the controllers, it would seem like a promising prospect to achieving hybrid control, obstacle avoidance and synchronization control using one unified controller.

Chapter 7

Simulation Results

We perform simulations of the full 6DOF robot in various scenarios to test our controller. We use the damped inverse Jacobian instead of the inverse Jacobian with damping constant 10^{-3} to prevent the controller from crashing in a singular configuration, [10]. We construct a desired trajectory which the leader robot will try to follow. The leader is fitted with a badly tuned PD controller such that we can see that the follower is actually tracking the leader. The follower will attempt to follow the leaders trajectory while avoiding collisions. We only plot one value for the position and and orientation when we compare the reference, leader and follower in order to keep the plots readable. The orientation measurements are given as roll pitch yaw angles in radians. One important thing to note throughout these simulations is the fact that perfect synchronization is not possible most of the time. The robot recovers gracefully after some time whenever an obstacles pushes it out of its desired trajectory.

7.1 No collision avoidance

The simplest case is where there are no obstacles to avoid. The simulation output is seen in figure 7.1. We see that the position and orientation error have minor disturbances, this disturbance comes from passing through a singular position. We have also plotted the determinant of the Jacobian, we see that singular configurations is common for the generated trajectory.

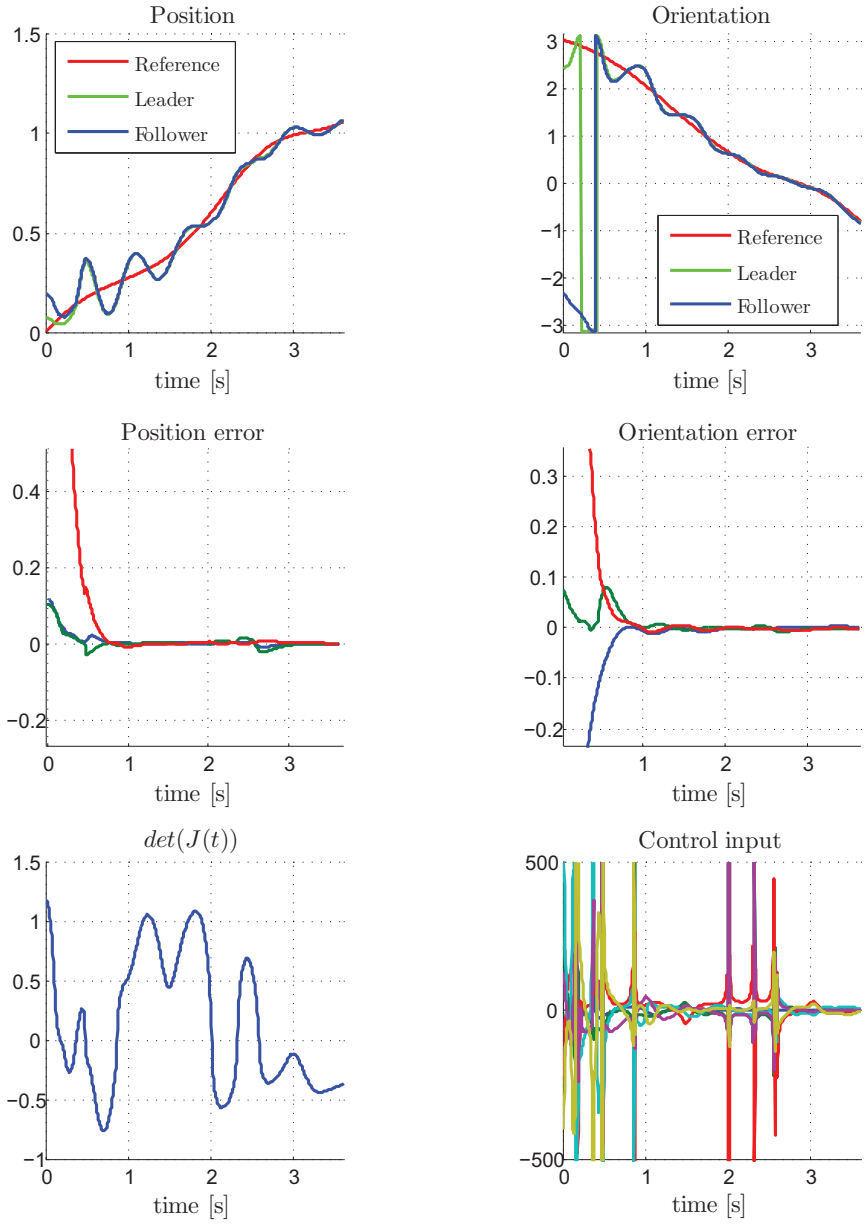


Figure 7.1: Synchronization control in the absence of obstacles.

7.2 Planar obstacles for the end effector

The robot is placed in the middle of a square room where the obstacle consists of four walls and the floor. Only the end effector is effected by the obstacles. The leader which is placed somewhere else and is purposefully generating a trajectory which will lead the robot to a collision. The controller used is from section 6.3. We see the end effector position traced out by the leader and the follower in figure 7.2.

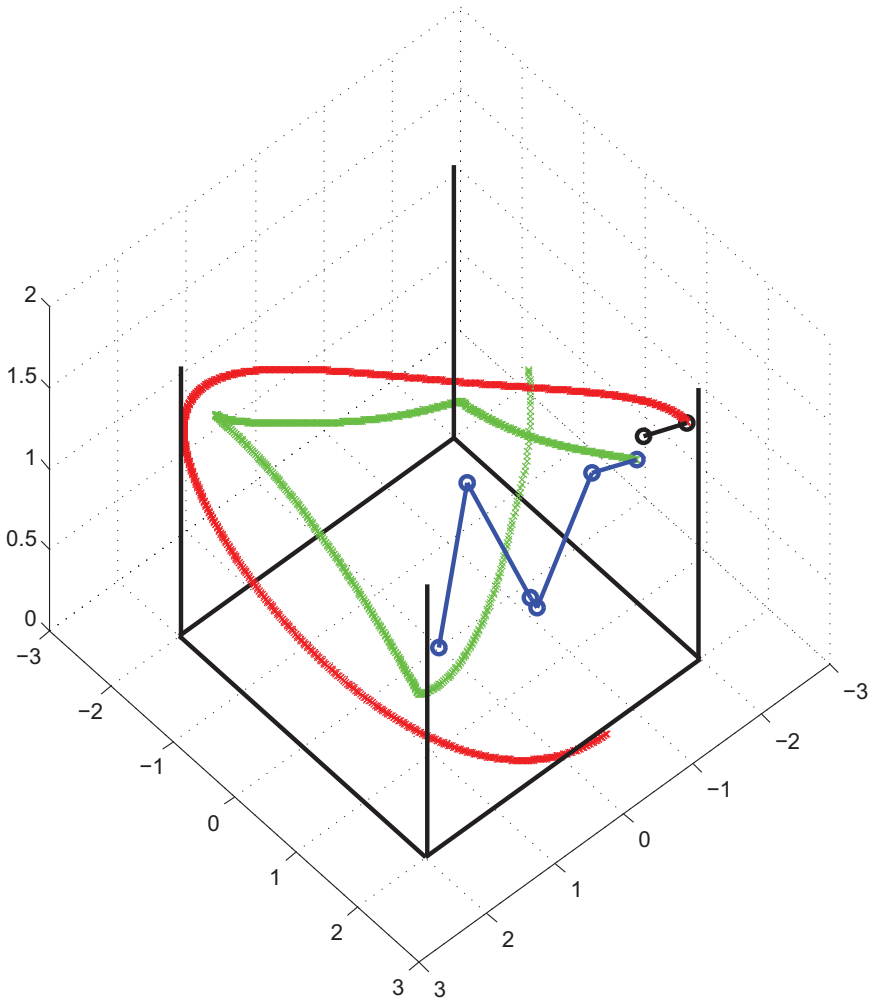
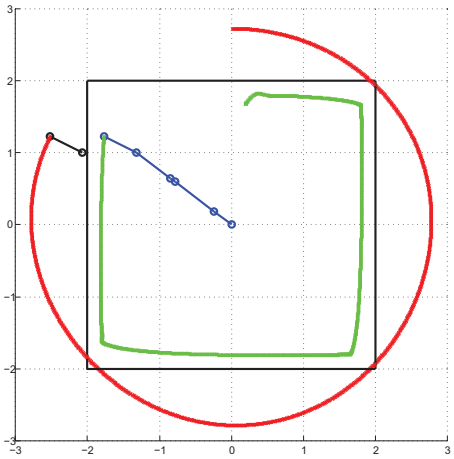
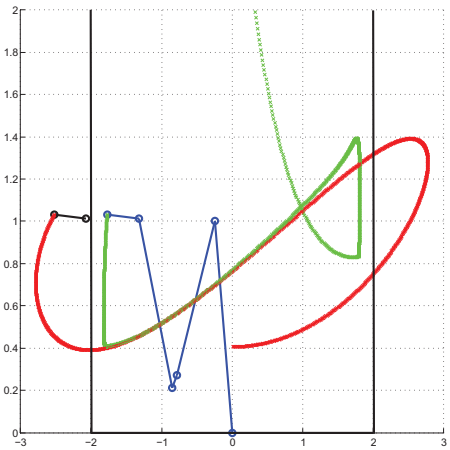


Figure 7.2: Synchronization control When the robot is confined within a small room.

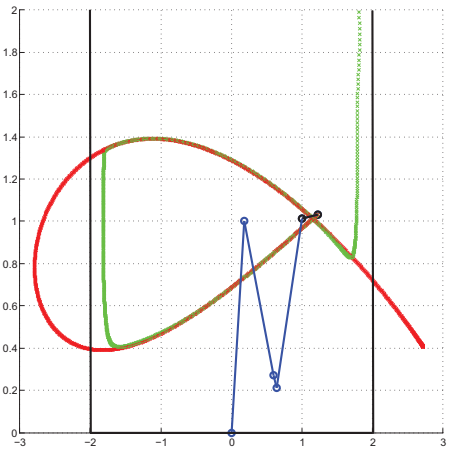
We also show the errors projected into the walls and the floor in figure 7.3. Notice that tracking perpendicular to the wall is achieved.



(a) XY view.



(b) ZY-view



(c) YZ-view

Figure 7.3: The robot tracking a reference while being confined in a small room.

Time plots of the key values of a trail with a reference which is harder to follow is seen in figure 7.4. We will test both the distance dependent repulsion force (figure 7.4) and the velocity dependent repulsive force (figure 7.7).

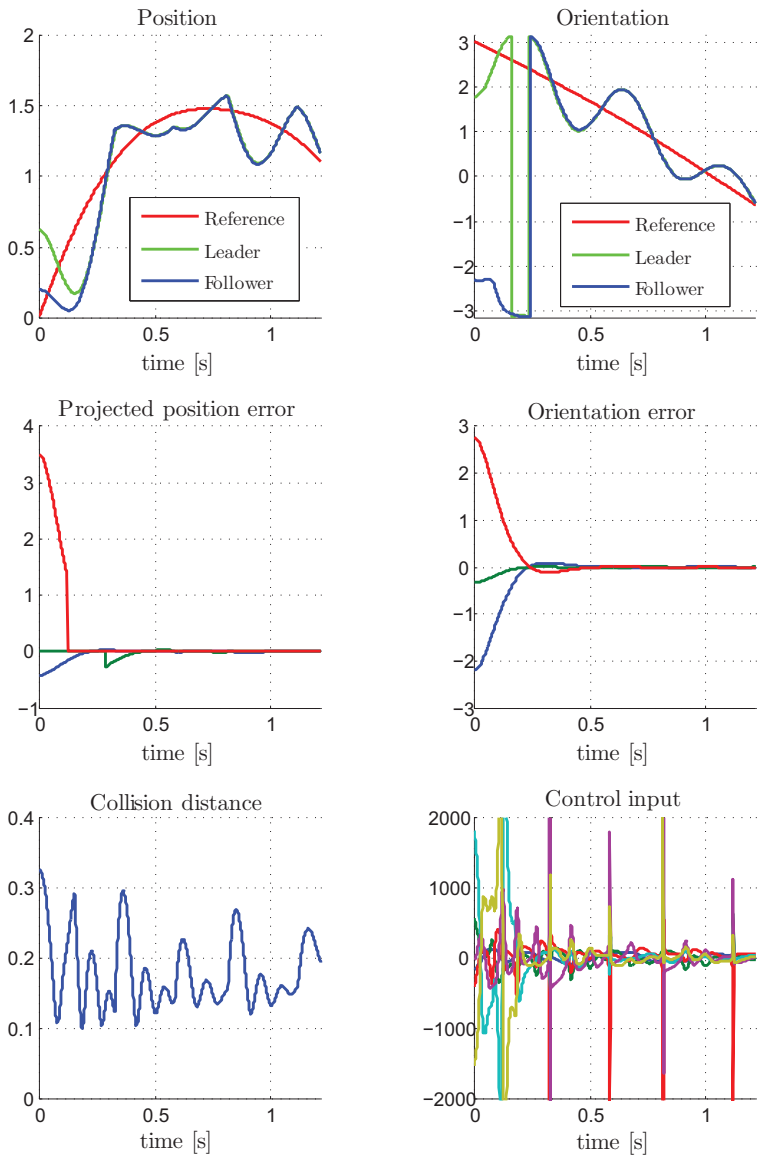


Figure 7.4: Synchronization control in the presence of planar obstacles with the distance dependent repulsive force. The position shown is the x-value.

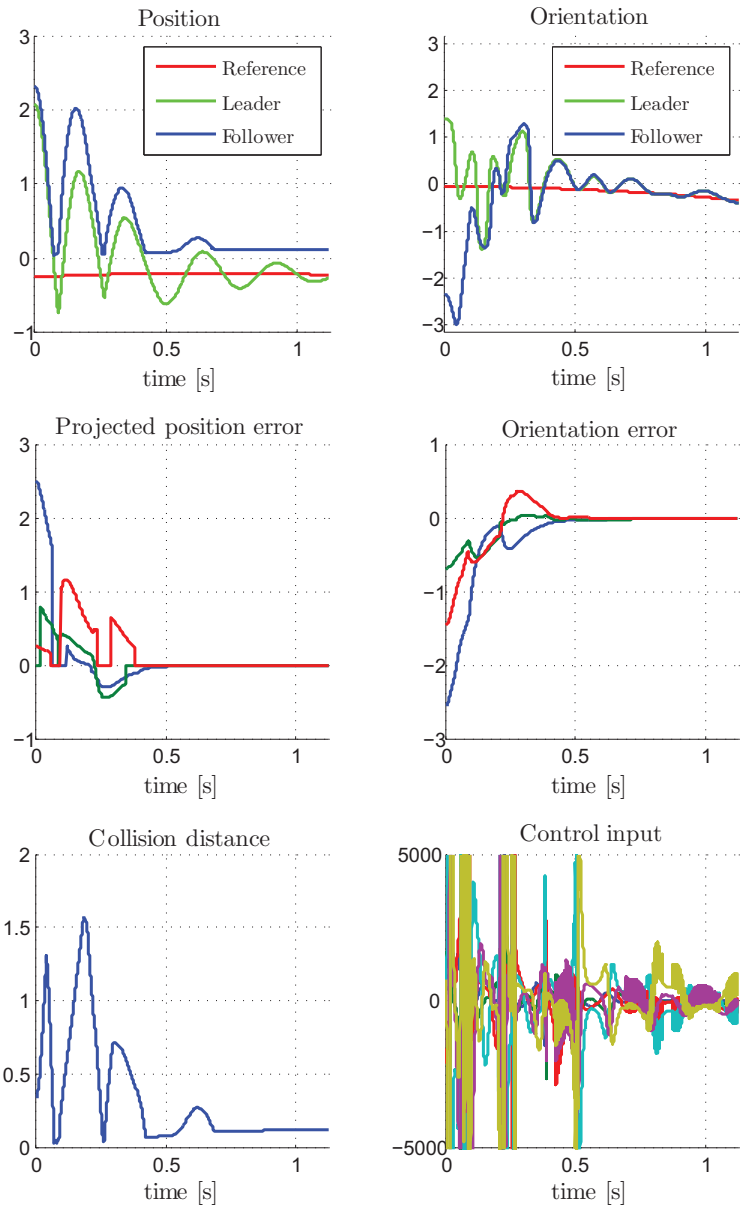


Figure 7.5: Synchronization control in the presence of planar obstacles with the velocity dependent repulsive force. The position shown is the z-position, and the reference has moved through the floor.

We notice that the collision distance is less oscillatory when using the velocity dependent repulsion force, but the control gain is larger.

7.3 Moving spherical obstacles for the end effector

The robot is placed in the middle of a square room where the obstacle consists of four walls and the floor as well as a ball moving around. Only the end effector is effected by the obstacles. The ball is on a collision course at the start of the simulation. The robot successfully manges to navigate around the ball and recovers gracefully. The controller used is from section 6.5 without the accleration term in γ.

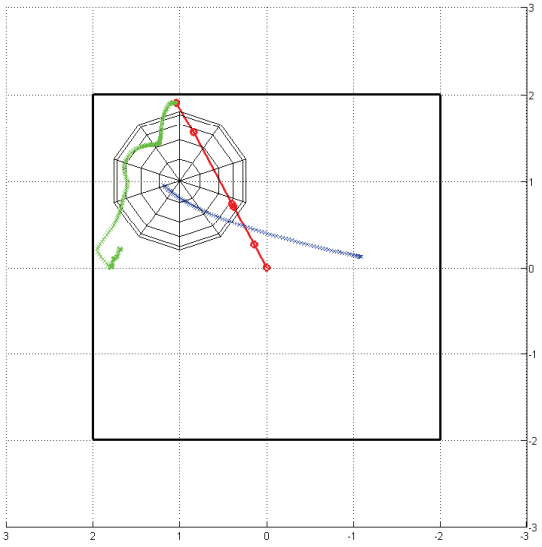


Figure 7.6: Synchronization control in the presence of planar obstacles and a moving ball. The ball is moving straight down on top of the robot.

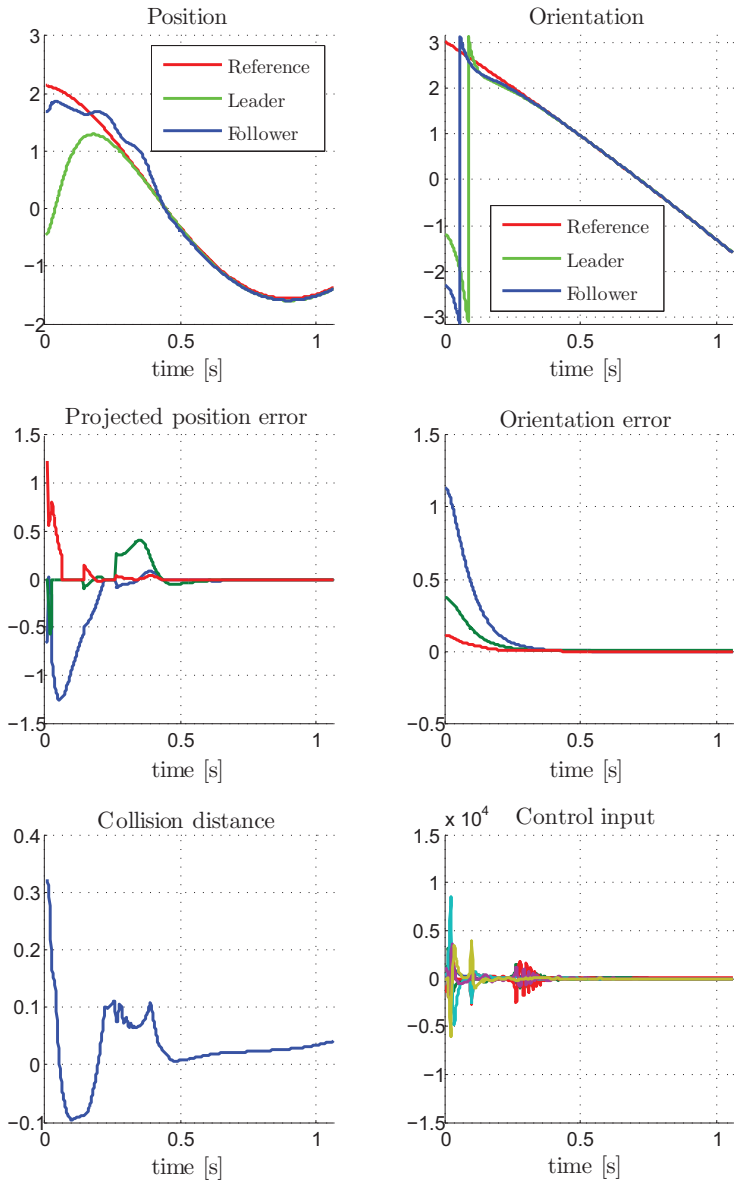


Figure 7.7: Synchronization control in the presence of planar obstacles and a moving ball. The position shown is the y-position.

7.4 Planar obstacles with full robot collision

The next simulation involves the robot confined in a small room where the entire robot is subject to collision. The robot is placed in a pose which will subject its third joint to collide with the floor. We will for the rest of the simulations only consider the distance dependent repulsion force.

We will now not achieve the same performance as before in orientation tracking because obstacle forces now also effect the orientation system. In figure 7.9 we show data from this simulation. The projected error is now not projected along the forces applied to different points on the robot. We have chosen to only use that obstacle tangent plane for the projection so we will easily see the actual end effector error achieved. Note that the error introduced by being in a bad pose may be removed by choosing another pose for our robot by some heuristic.

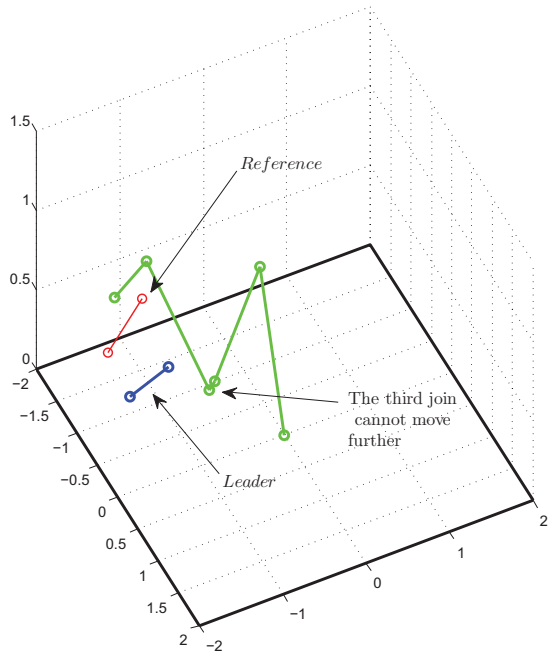


Figure 7.8: A screenshot of the simulation where the particular pose of the robot prevents it from reducing the synchronization error.

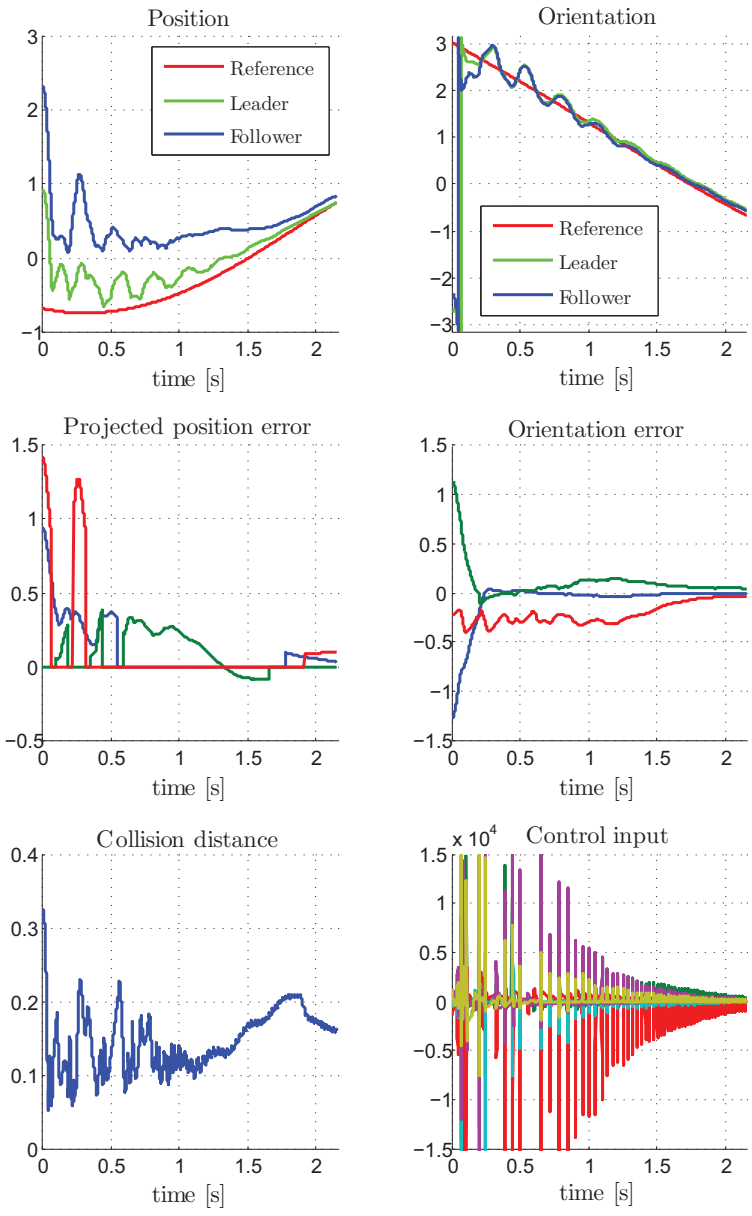


Figure 7.9: Synchronization control in the presence of planar obstacles with full robot collision avoidance. The position shown is the x-value.

7.5 Moving obstacles with full robot collision

The last experiment is the most complex. The robot is yet again placed in a room, this time we will use full robot obstacle avoidance with moving spheres representing the leader and other moving obstacles. We will now plot the actual position error as opposed to the projected error as the projected error along a general contact force is hard to interpret from a plot. We perform two trails, one with the elbow up pose as the initial condition, and the other with the elbow down initial configuration. The data output is seen in figures 7.12 and 7.13. We will see that the performance is heavily effected by the initial condition as the projected reference is not feasible from the elbow down position. Snapshots of the animation with the elbow up pose is seen in figure 7.11. We have not included the feedforward term γ in the controller from section sec:controller3 in order to better simulate feasible obstacle information.

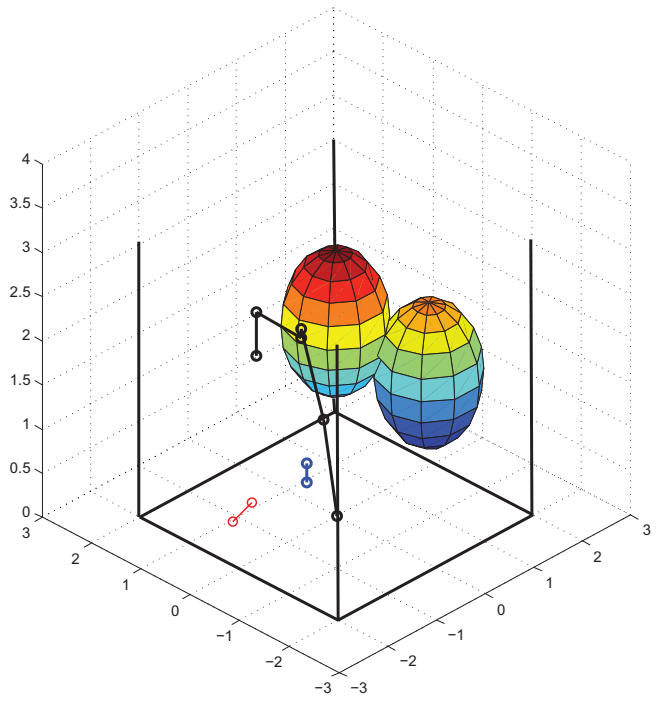


Figure 7.10: The elbow up initial condition for our experiment.

7.11.

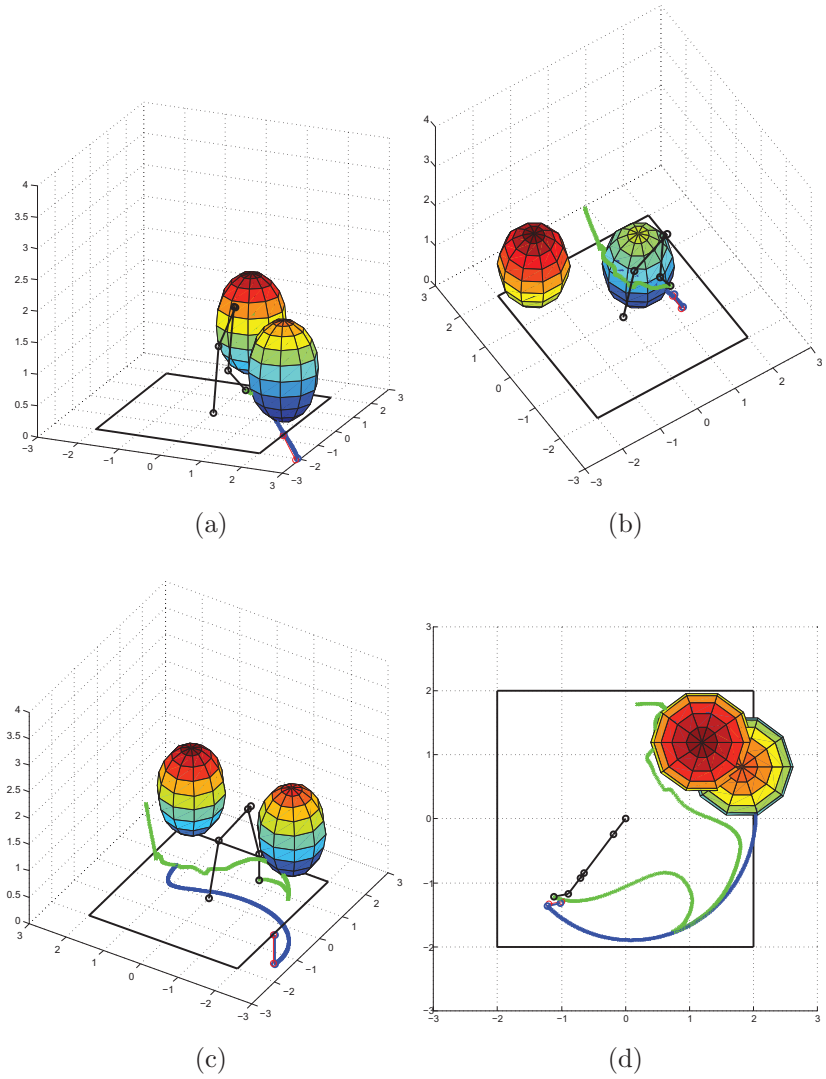


Figure 7.11: The robot in a workspace occupied by moving spheres under full robot obstacle avoidance.

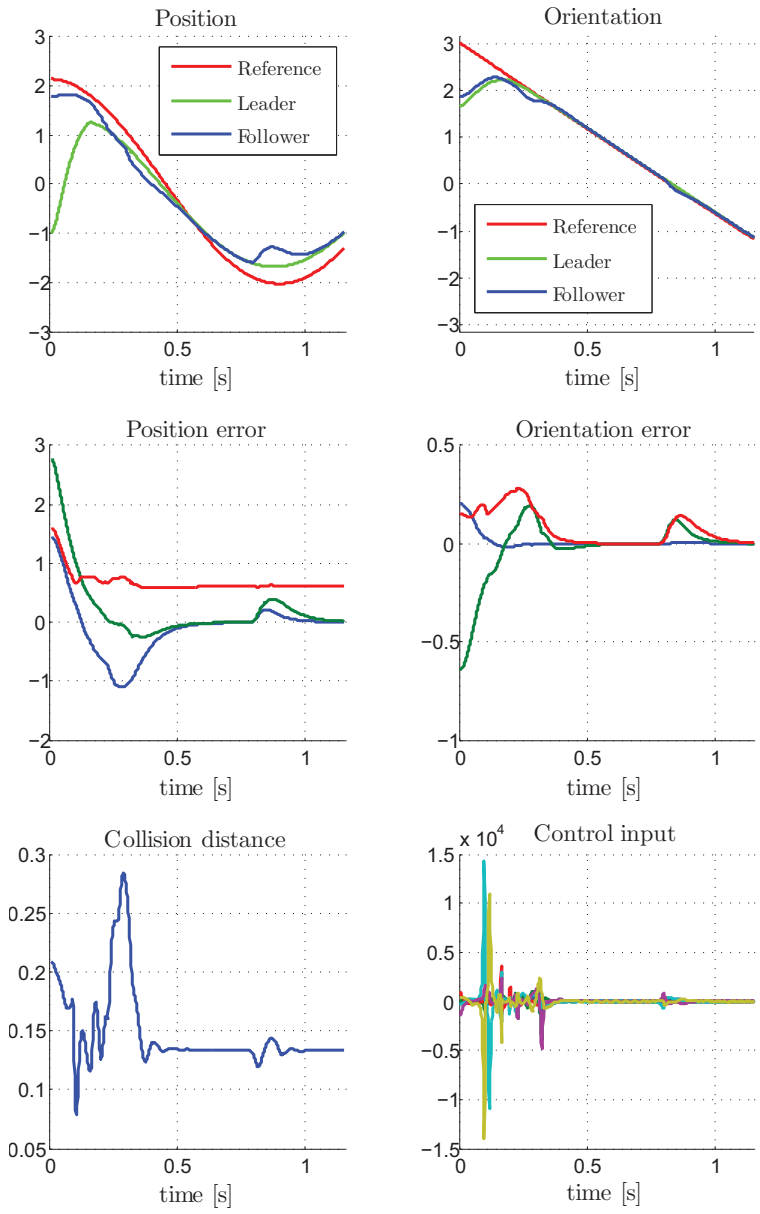


Figure 7.12: Synchronization control in the presence of planar and spherical obstacles with full robot collision avoidance with the elbow up initial condition.

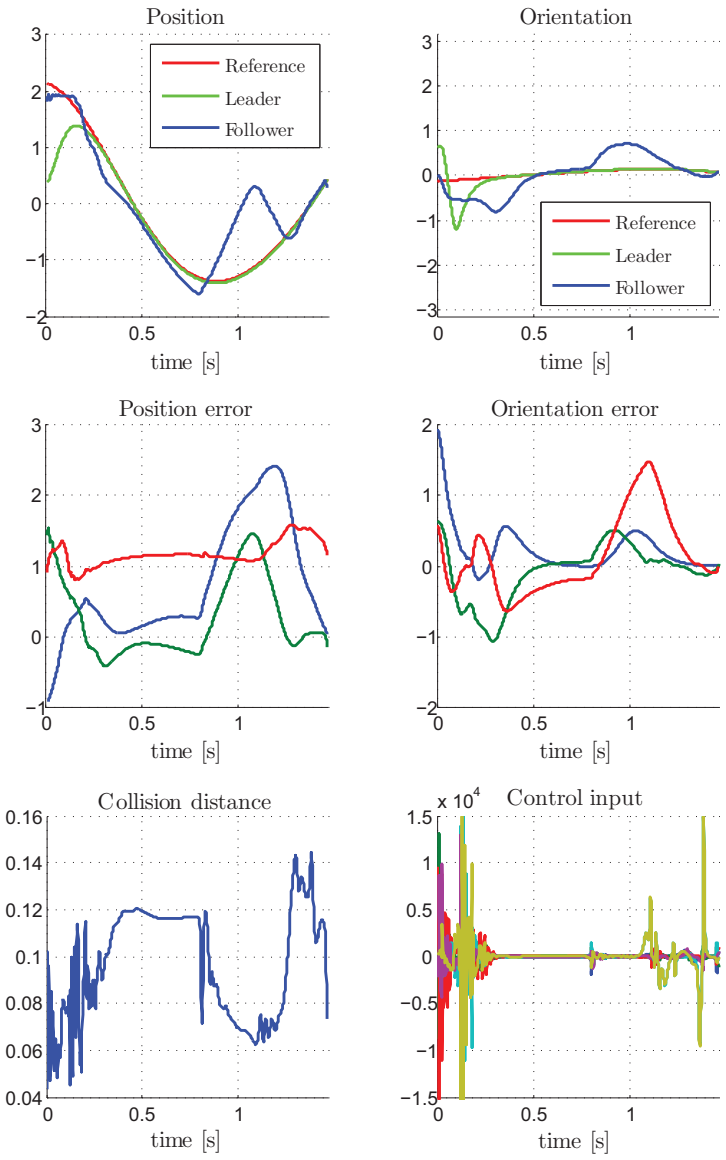


Figure 7.13: Synchronization control in the presence of planar and spherical obstacles with full robot collision avoidance with the elbow down initial condition.

Chapter 8

Conclusion and Further Work

8.1 Conclusion

A controller was developed which achieved full obstacle avoidance for any stationary, moving and rotating obstacle. synchronization in feasible movement direction was achieved for moving and rotating planar obstacles. The controller was developed in the framework of artificial potential functions. A simulator of a 6DOF robot was developed on which the controller was tested. The controller is developed in a general framework making it possible to employ it for a wide variety of applications. The controller is implementable in a low level of control and does not rely on any heuristics which are common in the development of collision avoidance controllers. The stability of the solution is local in the sense that the robot may not converge on an optimal solution even if one exists. Efficient heuristics may be employed to remedy this with the cost of losing generality.

A general representation of the workspace environment is presented along with a method which will dynamically solve the shortest distance problem. The representation is compatible with the camera sensors provided. The results in this thesis are verified using numerical simulations and examples are provided in order to illuminate and motivate the development.

The mathematical development in this thesis is apart from the modeling chapter largely self contained. The developed framework in which stability is proven is constructive in that it gives us a scalar system where detailed design of task specific behavior may be easily studied for different repulsion forces. This generality allows us to use one controller for a multitude of different

tasks. The control framework is highly compatible with controllers developed in the field of hybrid force motion control. This opens up the possibility of uniting the three field of hybrid force motion control, synchronization control and obstacle avoidance control in an elegant and provably correct manner.

8.2 Further work

This work will be carried on in a PhD thesis starting fall 2010. The crucial next step in the development of this controller is to make it robust with respect to uncertainties in state measurements, synchronization objective, obstacle representation and modeling errors. This is important as limited information is a reality in robotics. It is however a necessity to first study the nominal system in order to pinpoint how this lack of information will impact our system. Further work also includes the combination of our system with hybrid force motion control. This is a logical step as a useful robot system will not only need to avoid collision, but also manipulate its environment. We are hopeful that results will follow as the first step on the way to a practical implementation of a synchronization with obstacle avoidance system is complete.

Bibliography

[1] A. Ahmadzadeh, N. Motee, A. Jadbabaie, and G. Pappas. Multi-vehicle path planning in dynamically changing environments. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2148–2153. Institute of Electrical and Electronics Engineers Inc., The, 2009.

[2] F. Caccavale, C. Natale, B. Siciliano, and L. Villani. Resolved-acceleration control of robot manipulators: A critical review with experiments. *Robotica*, 16(05):565–573, 1998.

[3] S.I. Choi and B.K. Kim. Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, 18(02):143–151, 2000.

[4] C.H. Edwards and D.E. Penney. *Elementary linear algebra*. Prentice-Hall Englewood Cliffs, New Jersey, USA, 1988.

[5] O. Egeland and J.T. Gravdahl. *Modeling and simulation for automatic control*. Marine Cybernetics, 2002.

[6] LE Kavraki, P. Svestka, J.C. Latombe, and MH Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[7] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90, 1986.

[8] H. Nijmeijer and A. Rodriguez-Angeles. *Synchronization of mechanical systems*. World Scientific Pub Co Inc, 2003.

[9] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on robotics and automation*, 8(5):501–518, 1992.

[10] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. Wiley New Jersey, 2006.

[11] OV Zenkin. Analytical description of geometrical shapes. *Cybernetics and Systems Analysis*, 6(4):481–489, 1970.