Robotic Work-Space Sensing and Control

Magnus Linderoth

Department of Automatic Control
Lund University
Lund, June 2011

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--3251--SE

Abstract

Industrial robots are traditionally programmed using only the internal joint position sensors, in a sense leaving the robot blind and numb. Using external sensors, such as cameras and force sensors, allows the robot to detect the existence and position of objects in an unstructured environment, and to handle contact situations not possible using only position control.

This thesis presents work on how external sensors can be used in robot control. A vision-based robotic ball-catcher was implemented, showing how high-speed computer vision can be used for robot control with hard time constraints. Special attention is payed to tracking of a flying ball with an arbitrary number of cameras, how to initialize the tracker when no information about the initial state is available, and how to dynamically update the robot trajectory when the end point of the trajectory is modified due to new measurements. In another application example, force control was used to perform robotic assembly. It is shown how force sensing can be used to handle uncertain positions of objects in the workspace and detect different contact situations.

Acknowledgments

First of all I would like to thank my supervisors Rolf Johansson and Anders Robertsson, who introduced me to the field of robotics. They complement each other well and have given me important support and guidance in the different aspects of my development as a researcher. I also want to thank Klas Nilsson, who in practice has acted as an extra supervisor. Much of my work has been done in close cooperation with Andreas Stolt, who has been a very good work partner. Coming up with solutions to problems and implementing them always works smoothly with Andreas. I also want to thank the rest of my colleagues in the ROSETTA project for many fruitful discussions and feed-back on my work.

I want to thank all my colleagues at the Department of Automatic Control for making it such a nice environment to work in. The coffee breaks and lunches often offer interesting discussions on all imaginable topics. Some people are particularly important for making the work run smoothly. Anders Blomdell, with his vast programming experience, is the living book of answers when I have programming problems, and the robots in the lab could not run without him. Leif has given me important help when typesetting this document in LaTeX and getting the computers to work the way I want them to. Eva Schildt, Eva Westin, Ingrid Nilsson and Britt-Marie Mårtensson have helped me handle all administrative matters

Special thanks go to Kristian Soltész, who (unintentionally) had the biggest impact on me when I decided to do a PhD. He is a good friend and colleague, inspiring me in my research, and he has a never

ending supply of interesting, challenging problems for me to try to solve.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 – Challenge 2 – Cognitive Systems, Interaction, Robotics – under grant agreement No 230902 - ROSETTA. This document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained herein.

Contents

1

Introduction

1.1 Motivation and Background

The traditional way of programming industrial robots is based on position control and the trajectories are tracked using the internal position sensors. This kind of programming is widely used in industry and has become indispensable in many kinds of manufacturing. The robots are fast and have good repetitional accuracy, outperforming humans in many applications. These systems, however, only work in very structured environments.

Currently, there is a trend to extend robotic operation to task execution in time-varying dynamic, non-deterministic, real-life environments. Such robotics systems must be capable of responding in a timely and sensible manner and with a suitable degree of autonomy to uncertain, incomplete knowledge, and to situations not anticipated at design time. In addition to work-space sensing supported by suitable robotic software architectures, there is a need for extraction of real-time sensor information at various abstraction levels for feedback control of actuator control, motion control, trajectory control, force control, task-execution control, and supervisory control. To the purpose of effective and efficient task execution, it is necessary to include feedback control ranging from fast force feedback to reactive event-oriented feedback based on cognitive interpretation of sensor data. In this context, a whole range of new control challenges appear with a potential to reshape the current dominance of fixture-based position-controlled industrial robotics.

In this thesis, the research field of robotic task execution based on sensing and control is studied. Particular attention is given to two challenging application cases, *i.e.*, a ball-catching industrial robot, and force-controlled assembly. Both cases require high-rate feedback from senors external to the robot.

Several skills are required for catching a ball. The ball has to be detected and its position must be estimated and extrapolated to the future. In order to catch as many balls as possible, the robot has to start moving as soon as an estimate of the catching position is available, and then the trajectory must be modified while moving, as the catching position is modified due to new measurements. All of these actions have to be performed under hard time constraints so that the robot reaches the catching position before the ball reaches the robot.

Using force sensing in assembly provides important advantages over position based control. The ability to sense and act upon different contact situations makes it possible to handle, *e.g.*, uncertain positions or shapes of objects, which mimics the capability of humans to perform very complex assembly operations in spite of our poor position accuracy.

1.2 Publications

The thesis is based on the following publications.

Linderoth, M., A. Robertsson, K. Åström, and R. Johansson (2009): “Vision based tracker for dart-catching robot” In *Preprints 9th IFAC International Symposium on Robot Control (SYROCO’09)*. Gifu, Japan, pp. 883–888.

Linderoth, M., A. Robertsson, K. Åström, and R. Johansson (2010): “Object tracking with measurements from single or multiple cameras” In *Proc. 2010 IEEE International Conference on Robotics and Automation (ICRA2010)*. Anchorage, AK, pp. 4525–4530.

Linderoth, M., K. Soltesz, A. Robertsson, and R. Johansson (2011): “Initialization of the Kalman filter without assumptions on the initial state" In *Proc. 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*. Shanghai, China, pp. 4992–4997.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2011): "Force Controlled Assembly of Emergency Stop Button" In *Proc. 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*. Shanghai, China, pp. 3751–3756.

Other Publications

Stolt, A., M. Linderoth, A. Robertsson, M. Jonsson, and T. Murray (2011): "Force Controlled Assembly of Flexible Aircraft Structure" In *Proc. 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*. Shanghai, China, pp. 6027–6032.

1.3 Contributions

The thesis contains the following contributions

- a method for initializing the Kalman filter when no *a priori* information about the initial state is available;
- a method for tracking objects using single or multiple cameras;
- a method for generating robot trajectories when the target point is getting updated during the execution of the trajectory;
- a framework for performing force-controlled robotic assembly.

1.4 Thesis Outline

Chapter 2 describes hardware and interfaces used for robot control. Part I, including Chapters 3–6, presents work done in the context of a robotic ball-catcher. Part II, including Chapters 7–10, describes work done in the context of force-controlled robotic assembly. Finally, conclusions are given in Chapter 11.

Part I starts with Chapter 3, introducing the ball-catching problem and describing the physical setup. Chapter 4 presents a way of initializing the Kalman filter. In Chapter 5 visual tracking of a flying ball is described. Chapter 6 presents a method for generating robot trajectories when the target point is updated during the execution.

Part II starts with Chapter 7, introducing the topic of force-controlled assembly. Chapter 8 presents the framework used to describe the assembly operations. Chapter 9 presents work done on a snap fit assembly application. In Chapter 10 lead-through control of two virtually connected robots is described.

2

Hardware and Interfaces

All the experiments described in this thesis were performed in the Robotics Lab of the Department of Automatic Control at LTH, Lund University. The following sections give a short overview of equipment, computer architecture and interfaces used.

2.1 Robots

Two kinds of robots were used for the experiments described in this thesis, an IRB140 and an IRB2400-16, both produced by ABB [ABB Robots, 2011]. The IRB2400-16 was controlled by an S4Cplus controller cabinet and the IRB140 by a controller of the newer IRC5 type. The IRB140, shown in Fig. 2.1, has a maximum payload of 6 [kg], reach of 810 [mm] and position repeatability of ±0.03 [mm]. The IRB2400-16, shown in Fig. 2.2, has a maximum payload of 20 [kg], reach of 1550 [mm] and position repeatability of ±0.06 [mm].

2.2 Force/torque sensing

Each robot was equipped with a wrist-mounted JR3 100M40A force/torque sensor [JR3, 2011], which can be seen as a blue cylinder attached to the robot flange in Fig. 2.1 and 2.2. It can measure 3D forces up to 400 [N] and 3D torques up to 40 [Nm] with a data rate of 8 [kHz].

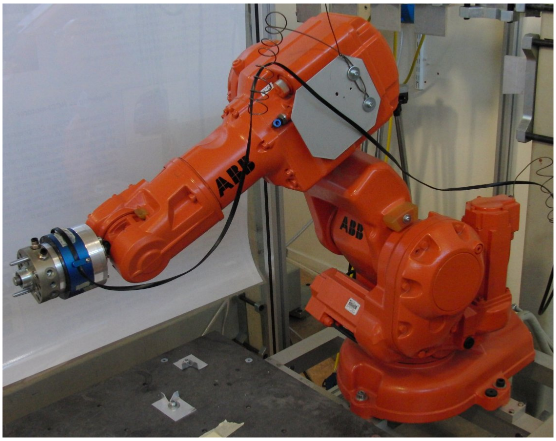


Figure 2.1 The IRB140 robot from ABB equipped with a JR3 force/torque sensor.

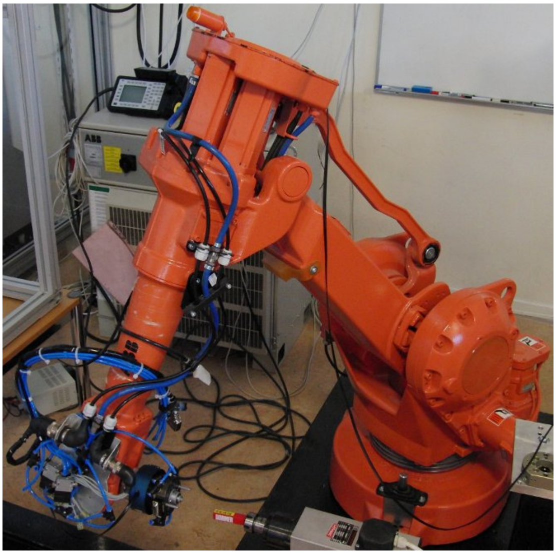


Figure 2.2 The IRB2400-16 robot from ABB equipped with a JR3 force/torque sensor.

2.3 Robot Controller Interface

The robots were controlled using the Extctrl interface [Blomdell *et al.*, 2005; Blomdell *et al.*, 2010]. It connects to the low-level axis controllers of the robot with a sample rate of 250 [Hz], and lets the higher level functionality be executed in an external controller. For each joint the external controller can set the reference position, and provide feed-forward data for the velocity and motor torque. Measured positions *etc.* are sent back from the robot control cabinet to the external controller.

The data is transferred using the LabComm protocol [LabComm, 2011], which allows the specification of data types that should be sent over a socket. The communication overhead has been kept to a minimum and the protocol is appropriate for sending samples of process data in real-time.

Part I

Ball-Catching Robot

3

Ball-Catching Robot

3.1 Introduction

The following chapters describe how high-speed computer vision can be used in a motion control application. The problem considered is a ball-catching robot. A sketch describing the problem can be seen in Fig. 3.1. A box with a hole was mounted on a robot and when a ball was thrown toward the box, the robot should move the box so the ball always hit the hole. The detection of the ball was performed with cameras that provided data to the estimation of the position and future trajectory of the ball. In turn, this information was used to move the box to the correct position.

Ball-catching robots have been described in [Frese *et al.*, 2001] and [Bäuml *et al.*, 2011]. The ball-catching application is closely related to the table tennis playing robot, which has been treated in numerous works, *e.g.* [Matsushima *et al.*, 2003], which used an adaptive black-box model for prediction.

3.2 Robot

The robot used for the ball-catcher was an ABB IRB140. For more details, see Sec. 2.1.

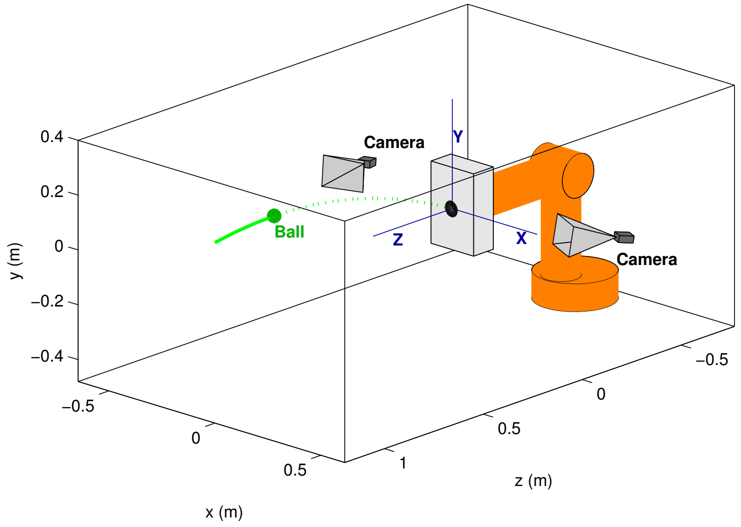


Figure 3.1 Ball-catching robot.

3.3 Cameras

To the purpose of stereo vision, two cameras were located in the robot workspace, one on each side of the robot. The application required good real-time camera performance and for this two Basler A602fc [Basler, 2011] with an IEEE1394 serial interface (also known as FireWire or i.LINK) were used. They could supply color images at resolutions up to 656 $\times$ 490 pixels with a maximum frame rate of 100 [fps] at full resolution. When two cameras were connected to the same FireWire card, the frame rate was limited to 50 [fps], which was the frame rate used for the ball-catcher.

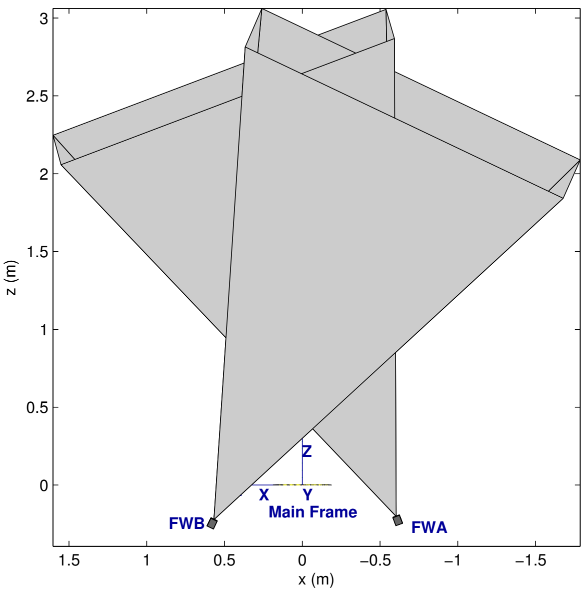


Figure 3.2 Top view of the setup illustrating the stereo coverage. The cones show the field of view for the respective cameras.

3.4 Camera Positioning

Many different ways of positioning the cameras were possible. The positions chosen were on each side of the robot, pointing toward the thrower, illustrated in Fig. 3.2. This positioning allowed the cameras to see the ball at a big distance if the thrower was straight in front of the robot. The ball-catcher application had a higher requirement on the accuracy of the estimated ball trajectory when the ball was close to the robot, which was fulfilled by the chosen camera positioning, since the ball was close to the cameras and observed from favorable angles when it was close to the robot.

3.5 Image Analysis

The algorithms used to find the ball in the images were implemented in C and run on a desktop PC. Processing of an image pair took approximately 6 [ms]. For more details, see [Linderoth, 2008].

3.6 Modeling of the Ball

The thrown ball was modeled as a point mass flying in a gravity field with negligible air drag. Using the coordinate system in Fig. 3.1, the dynamics were described by

$$m \begin{bmatrix} \ddot{x}_b \\ \ddot{y}_b \\ \ddot{z}_b \end{bmatrix} = \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix} \tag{3.1}$$

where m is the mass of the ball and g is the earth gravity acceleration constant. Introducing the state vector

$$x = \begin{bmatrix} x_b & y_b & z_b & \dot{x}_b & \dot{y}_b & \dot{z}_b \end{bmatrix}^T \tag{3.2}$$

equation (3.1) is equivalent to

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -g \\ 0 \end{bmatrix} \tag{3.3}$$

Discretizing (3.3) and introducing noise, results in a state space model on the form

$$\begin{aligned} x(k+1) &= \Phi(k)x(k) + \Gamma(k)u(k) + v(k) \\ y(k) &= C(k)x(k) + e(k) \end{aligned} \tag{3.4}$$

where x is the state, u is the input and y is the measurement. The disturbances v and e are assumed to be white noise processes with zero mean values, $\mathrm{E}[v(k)v^T(k)] = R_v(k) \succeq 0$, $\mathrm{E}[e(k)e^T(k)] = R_e(k) \succ 0$, and $\mathrm{E}[v(k)e^T(k)] = 0$. Further,

$$\Phi(k) = \begin{bmatrix} 1 & 0 & 0 & h & 0 & 0 \\ 0 & 1 & 0 & 0 & h & 0 \\ 0 & 0 & 1 & 0 & 0 & h \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \forall k \tag{3.5}$$

$$\Gamma(k) = \begin{bmatrix} 0 \\ h^2/2 \\ 0 \\ 0 \\ h \\ 0 \end{bmatrix}, \forall k \tag{3.6}$$

$$u(k) = -g, \ \forall k \tag{3.7}$$

where h is the sampling period. The matrix C will vary depending on the measurement method.

When analyzing measured position data from thrown balls, it could be seen that there was a noticable air drag. The ball-catcher still worked well, and hence the drag was never modeled.

4

Initialization of the Kalman Filter without Assumptions on the Initial State

4.1 Introduction

When performing state estimation on dynamical systems, the Kalman filter [Kalman, 1960] is a very commonly used tool. Just as for other recursive algorithms, initialization is a necessary computational step and such initialization may be accomplished in a variety of different approaches, *e.g.*, probabilistic (Gaussian, Bayesian [Bayes, 1763]), geometric and information-theoretical approaches. In the original formulation of the Kalman filter, it was assumed that the initial value of the state had a known mean value and variance. If no such data are available, the estimate will have a transient in the initial phase of the filtering. If it is possible to start the estimation well before the estimate is to be used, this causes no problem, since the estimate will have time to converge. The transient can also be reduced by letting the initial covariance matrix of the estimate have very large eigenvalues.

However, if the estimate is needed as soon as possible after the start of the estimation, it is desirable that not even the first estimates are affected by the guess of the initial state. One such example is a

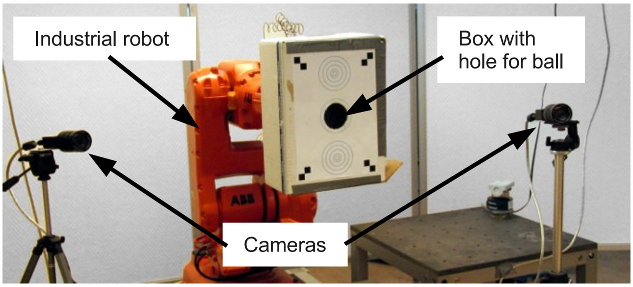


Figure 4.1 Picture of a ball-catching robot.

ball-catching robot, which has been treated in, *e.g.*, [Frese *et al.*, 2001], [Linderoth *et al.*, 2009], [Linderoth *et al.*, 2010] and [Bäuml *et al.*, 2011]. A photo of such a setup is shown in Fig. 4.1. When a ball is thrown toward the robot, the box is moved to make the ball hit the hole. High-speed cameras provide information about the position of the ball, and a Kalman filter is used to estimate the position of the ball and predict its future trajectory. Only a limited number of measurements are available during the flight of the ball and due to the limited acceleration of the robot, it has to start moving as soon as the catching position can be estimated. Thus, it is essential to have a good estimate from the very start of the measurement series so the robot can get to its target in time.

Widely used algorithms are based on the extended Kalman filter (EKF), whose application to SLAM (Simultaneous localization and mapping) problems was developed in a series of seminal papers [Smith and Cheeseman, 1986; Moutarlier and Chatila, 1990; Smith *et al.*, 1990]. The EKF calculates a Gaussian posterior over the locations of environmental features and the robot itself. Information-oriented Kalman-type filters were proposed for feature-based SLAM [Walter *et al.*, 2007; Thrun *et al.*, 2004] and for maneuvering target tracking [Farooq and Bruder, 1990] with attention to approximation and computational speed.

Robot vision is commonly used for vision-based tracking in robotic applications. An introduction to available algorithms is given in [Ma *et al.*, 2003].

Many specialized approaches for making an informed initialization of the Kalman filter have been proposed for specific problems and can be found in, *e.g.*, [Einhorn *et al.*, 2007], [Hyland, 2002] and [Weiner, 1981]. A more general way of initializing a system on the linear state-space form (4.1) can be done by using information filter theory, but then with additional requirements, *e.g.*, that Φ is non-singular or that (4.1) is controllable [Kailath *et al.*, 2000].

This thesis presents a method to initialize the Kalman filter without making any assumptions on the initial value, only assuming that the system is on the linear state-space form (4.1). This is done by transforming the state estimate $\hat{x}$ into a space with a basis separating directions of infinite variance from those with finite variance. A different approach for solving the same problem is presented in [Hagander, 1973].

4.2 Preliminaries

State Space Description

The problem considered relates to discrete-time time-varying linear systems on the form

$$\begin{aligned} x(k+1) &= \Phi(k)x(k) + \Gamma(k)u(k) + v(k) \\ y(k) &= C(k)x(k) + e(k) \end{aligned} \tag{4.1}$$

where x is the state, u is the input and y is the measurement. The disturbances v and e are assumed to be white-noise processes with zero mean values, $\mathrm{E}[v(k)v^T(k)] = R_v(k) \succeq 0$, $\mathrm{E}[e(k)e^T(k)] = R_e(k) \succ 0$, and $\mathrm{E}[v(k)e^T(k)] = 0$.

The Kalman Filter

The Kalman filter can be used to estimate the state of (4.1) recursively as described by (4.2), $\hat{x}(l|k)$ denoting the estimate of $x(l)$ based on measurements up to sample k and $P(l|k)$ being the covariance matrix of $\hat{x}(l|k)$.

$$\begin{aligned}
\hat{x}(k|k) &= \hat{x}(k|k-1)+ \\
&\quad + K(k)\left(y(k) - C(k)\hat{x}(k|k-1)\right) \\
K(k) &= P(k|k-1)C^T(k)\cdot \\
&\quad \cdot \left(C(k)P(k|k-1)C^T(k) + R_e(k)\right)^{-1} \\
P(k|k) &= P(k|k-1) - K(k)C(k)P(k|k-1) \\
\hat{x}(k+1|k) &= \Phi(k)\hat{x}(k|k) + \Gamma(k)u(k) \\
P(k+1|k) &= \Phi(k)P(k|k)\Phi^T(k) + R_v(k)
\end{aligned} \tag{4.2}$$

Singular Value Decomposition

Consider a matrix $A \in \mathbb{R}^{m\times n}$ with $\text{rank}(A) = r$. Using singular value decomposition (SVD) it can be factorized as

$$A = U\Sigma V^T \tag{4.3}$$

where $U \in \mathbb{R}^{m\times r}$ satisfies $U^TU = I$, $V \in \mathbb{R}^{n\times r}$ satisfies $V^TV = I$ and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$

4.3 Optimal Solution of a Linear System of Equations with Noise

This section describes methods for solving linear systems of equations, which will be used later.

Over-determined System

Consider a system of linear equations with disturbances:

$$z = Gx + w \tag{4.4}$$

where $z \in \mathbb{R}^m$ and $G \in \mathbb{R}^{m\times n}$ are known and $w \in \mathbb{R}^m$ is a disturbance with $\text{E}[w] = 0$ and $\text{E}[ww^T] = R_w \succ 0$.

Assume that $\text{rank}(G) = n < m$, *i.e.*, the system is over-determined. Let $\hat{x}$ denote the minimum variance unbiased estimate of x, which can be obtained from the Gauss-Markov theorem [Kailath *et al.*, 2000], with the solution

$$\hat{x} = (G^T R_w^{-1} G)^{-1} G^T R_w^{-1} z \tag{4.5}$$

$$R_x = \mathrm{E}\left[(\hat{x} - x)(\hat{x} - x)^T\right] = (G^T R_w^{-1} G)^{-1} \tag{4.6}$$

Under-determined System

Again consider the system (4.4), but now assume that $\operatorname{rank}(G) = r < n$, *i.e.*, the system of equations is under-determined. Still, x can be partly determined. By singular value decomposition G can be factorized as

$$G = U \Sigma V^T \tag{4.7}$$

It is possible, as a part of the SVD algorithm, to construct

$$S = \begin{bmatrix} S_f & S_i \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{4.8}$$

such that $S_f = V$ and $S^T S = I$. Define $x_f \in \mathbb{R}^r$ and $x_i \in \mathbb{R}^{n-r}$ as the unique solution to

$$x = S \begin{bmatrix} x_f \\ x_i \end{bmatrix} = S_f x_f + S_i x_i \tag{4.9}$$

Note that x_f is a parametrization of the part of x that can be estimated by (4.4), and x_i is a parametrization of the null space of G. Inserting (4.9) into (4.4) and noting that $V^T [S_f \; S_i] = [I \; 0]$ one obtains

$$\begin{aligned} z &= Gx + w \\ &= U \Sigma V^T (S_f x_f + S_i x_i) + w \\ &= U \Sigma x_f + w \end{aligned} \tag{4.10}$$

Since $U\Sigma$ has full rank, one can solve (4.10) for $\hat{x}_f$, using the method described in the previous subsection.

4.4 Filter Initialization

State Partitioning

During the initialization of the Kalman filter there may be times when the variance of the state estimate is finite in some directions of the state space and infinite in other directions. To handle this situation the state can by a linear transformation be reoriented to a space where the directions with infinite variance are orthogonal to as many base vectors as possible. Let the state $\bar{x}$ in this alternative space be defined by

$$T\bar{x} = x \tag{4.11}$$

where $T \in \mathbb{R}^{n \times n}$ and $T^T T = I$. Let $\hat{\bar{x}}$ denote the estimate of $\bar{x}$, and denote the estimation error by

$$\tilde{\bar{x}} = \bar{x} - \hat{\bar{x}} \tag{4.12}$$

Throughout the chapter it is assumed that the estimates are designed to be unbiased, *i.e.*, so that $\mathrm{E}\left[\tilde{\bar{x}}\right] = 0$. The state can be partitioned as

$$\bar{x} = \begin{bmatrix} \bar{x}_f \\ \bar{x}_i \end{bmatrix} \tag{4.13}$$

such that the covariance-matrix of $\hat{\bar{x}}_f$ is finite and the variance of $\hat{\bar{x}}_i$ is infinite in all directions. Define n_f and n_i such that $n_f + n_i = n$, and $\bar{x}_f \in \mathbb{R}^{n_f}$, $\bar{x}_i \in \mathbb{R}^{n_i}$. Similarly, the transformation matrix T can be partitioned as

$$T = \begin{bmatrix} T_f & T_i \end{bmatrix}, \quad T_f \in \mathbb{R}^{n \times n_f}, \quad T_i \in \mathbb{R}^{n \times n_i} \tag{4.14}$$

Since T is orthonormal we have

$$T^{-1} = \begin{bmatrix} T_f & T_i \end{bmatrix}^{-1} = \begin{bmatrix} T_f & T_i \end{bmatrix}^T = \begin{bmatrix} T_f^T \\ T_i^T \end{bmatrix} \tag{4.15}$$

Note for future reference that

$$\begin{bmatrix} \bar{x}_f \\ \bar{x}_i \end{bmatrix} = \begin{bmatrix} T_f^T \\ T_i^T \end{bmatrix} x \tag{4.16}$$

and

$$x = T_f \bar{x}_f + T_i \bar{x}_i \tag{4.17}$$

which shows that T_f spans the directions in which $\hat{x}$ has finite variance and T_i spans the directions in which $\hat{x}$ has infinite variance.

Let $\bar{P}_f$ denote the covariance matrix of $\hat{\bar{x}}_f$:

$$\bar{P}_f = \mathrm{E}\left[(\hat{\bar{x}}_f - \bar{x}_f)(\hat{\bar{x}}_f - \bar{x}_f)^T\right] \tag{4.18}$$

In the remainder of the chapter all quantities may be appended with time indices so that, *e.g.*, $\hat{\bar{x}}_f(l|k)$ is the estimate of $\bar{x}_f(l)$ based on measurements up to sample k, and $\hat{x}(l|k) = T(l|k)\hat{\bar{x}}(l|k)$. Note, however, the slightly different case $x(l) = T(l|k)\bar{x}(l|k)$. The actual state x has only a single time index, since the second time index is meaningful only for estimates. Still $\bar{x}$ has two time indices to indicate which T was used for the transformation.

To conclude, all the knowledge about $\hat{x}(l|k)$ can be fully specified by $T(l|k)$, $\hat{\bar{x}}_f(l|k)$ and $\bar{P}_f(l|k)$.

Time Step

Assume that $T(k|k)$, $\hat{\bar{x}}_f(k|k)$ and $\bar{P}_f(k|k)$ are known. The state model (4.1) gives the time update

$$x(k+1) = \Phi(k)x(k) + \Gamma(k)u(k) + v(k). \tag{4.19}$$

The purpose of the time step is to calculate $T(k+1|k)$, $\hat{\bar{x}}_f(k+1|k)$ and $\bar{P}_f(k+1|k)$.

Choose $T(k+1|k)$ such that

$$T_f^T(k+1|k)\Phi(k)T_i(k|k) = 0 \tag{4.20}$$

$$n_f(k+1|k) = n - \operatorname{rank}\left(\Phi(k)T_i(k|k)\right) \tag{4.21}$$

$$T^T(k+1|k)T(k+1|k) = I \tag{4.22}$$

This can be interpreted as finding a $T(k+1|k)$ such that it is orthonormal and its $n_f(k+1|k)$ leftmost columns span the left null space of $\Phi(k)T_i(k|k)$, which can be done, *e.g.*, by means of SVD. Note that

$$n_f(k+1|k) \geq n_f(k|k) \tag{4.23}$$

where strict inequality holds if and only if $\Phi(k)$ is singular and its null space satisfies $\mathcal{N}(\Phi(k)) \cap \mathcal{R}(T_i(k|k)) \neq \emptyset$.

Premultiplying (4.19) with $T_f^T(k+1|k)$ gives

$$\begin{aligned}
&\bar{x}_f(k+1|k) \\
&= T_f^T(k+1|k)\Phi(k)x(k) \\
&\quad + T_f^T(k+1|k)\Gamma(k)u(k) \\
&\quad + T_f^T(k+1|k)v(k) \\
&= T_f^T(k+1|k)\Phi(k)\left(T_f(k|k)\bar{x}_f(k|k) + T_i(k|k)\bar{x}_i(k|k)\right) \\
&\quad + T_f^T(k+1|k)\Gamma(k)u(k) \\
&\quad + T_f^T(k+1|k)v(k) \\
&= T_f^T(k+1|k)\Phi(k)T_f(k|k)\bar{x}_f(k|k) \\
&\quad + T_f^T(k+1|k)\Gamma(k)u(k) \\
&\quad + T_f^T(k+1|k)v(k)
\end{aligned} \tag{4.24}$$

where the second and third equalities result from (4.17) and (4.20) respectively. Here the advantage of choosing $T(k+1|k)$ according to (4.20) becomes clear. Because of this choice $\bar{x}_f(k+1)$ is independent of $\bar{x}_i(k)$ and only depends on quantities with finite variance. Condition (4.21) guarantees that $T_f(k+1)$ has the highest possible rank.

Motivated by (4.24), let the update of the state estimate be defined by

$$\begin{aligned}
\hat{\bar{x}}_f(k+1|k) = {} & T_f^T(k+1|k)\Phi(k)T_f(k|k)\hat{\bar{x}}_f(k|k) \\
& + T_f^T(k+1|k)\Gamma(k)u(k)
\end{aligned} \tag{4.25}$$

The estimation error is then given by

$$\begin{aligned}
\tilde{\bar{x}}_f(k+1|k) &= \bar{x}_f(k+1|k) - \hat{\bar{x}}_f(k+1|k) \\
&= T_f^T(k+1|k)\Phi(k)T_f(k|k)\tilde{\bar{x}}_f(k|k) \\
&\quad + T_f^T(k+1|k)v(k)
\end{aligned} \tag{4.26}$$

It is easily verified that $\mathrm{E}\left[\tilde{\bar{x}}_f(k+1|k)\right] = 0$ as required. The variance of the estimate becomes

$$\begin{aligned}\bar{P}_f(k+1|k) &= \mathrm{E}\left[\tilde{\bar{x}}_f(k+1|k)\tilde{\bar{x}}_f^T(k+1|k)\right] \\ &= Q\bar{P}_f(k|k)Q^T \\ &\quad + T_f^T(k+1|k)R_v(k)T_f(k+1|k), \\ Q &= T_f^T(k+1|k)\Phi(k)T_f(k|k)\end{aligned} \tag{4.27}$$

Correction Step

Assume that $T(k|k-1)$, $\hat{\bar{x}}_f(k|k-1)$ and $\bar{P}_f(k|k-1)$ are known. The state model (4.1) gives the measurement

$$y(k) = C(k)x(k) + e(k) \tag{4.28}$$

The purpose of the correction step is to calculate $T(k|k)$, $\hat{\bar{x}}_f(k|k)$ and $\bar{P}_f(k|k)$.

Combining (4.12) and (4.16) gives

$$\begin{aligned}\hat{\bar{x}}_f(k|k-1) &= \bar{x}_f(k) - \tilde{\bar{x}}_f(k|k-1) \\ &= T_f^T(k|k-1)x(k) - \tilde{\bar{x}}_f(k|k-1)\end{aligned} \tag{4.29}$$

Equations (4.28) and (4.29) can be formulated as a single linear system of equations:

$$\begin{aligned}&\underbrace{\begin{bmatrix} y(k) \\ \hat{\bar{x}}_f(k|k-1) \end{bmatrix}}_{z} \\ &= \underbrace{\begin{bmatrix} C(k) \\ T_f^T(k|k-1) \end{bmatrix}}_{G} x(k) + \underbrace{\begin{bmatrix} e(k) \\ -\tilde{\bar{x}}_f(k|k-1) \end{bmatrix}}_{w}\end{aligned} \tag{4.30}$$

which can be solved by the method described in Sec. 4.3 with

$$R_w = \begin{bmatrix} R_e(k) & 0 \\ 0 & \bar{P}_f(k|k-1) \end{bmatrix} \tag{4.31}$$

The solution is given by

$$T(k|k) = S \tag{4.32}$$
$$\hat{\bar{x}}_f(k|k) = (\Sigma U^T R_w^{-1} U \Sigma)^{-1} \Sigma U^T R_w^{-1} z \tag{4.33}$$
$$\bar{P}_f(k|k) = (\Sigma U^T R_w^{-1} U \Sigma)^{-1} \tag{4.34}$$
$$n_f(k|k) = \text{rank}(\Sigma) = \text{rank}(G) \tag{4.35}$$

where U, Σ and S are defined in (4.7) and (4.8).

From the definition of G in (4.30) it can be seen that

$$\text{rank}(G) \geq \text{rank}\left(T_f(k|k-1)\right) \tag{4.36}$$

The orthonormality of $T(k|k-1)$ in combination with (4.14) gives

$$\text{rank}\left(T_f(k|k-1)\right) = n_f(k|k-1) \tag{4.37}$$

Combining (4.35) - (4.37) results in

$$n_f(k|k) \geq n_f(k|k-1) \tag{4.38}$$

where equality holds if and only if $\mathcal{R}(C^T(k)) \subseteq \mathcal{R}(T_f(k|k-1))$. Equations (4.23) and (4.38) together show that n_f never decreases and give conditions for when n_f increases.

If G has full rank the variance of $\hat{x}(k|k)$ will be finite in all directions and $n_f(k|k) = n$.

Remark: For $n_f(k|k-1) = n$ and $T_f(k|k-1) = I$ it can be shown that the solution of (4.30) is equivalent to the correction step of the ordinary Kalman filter (4.2).

How to start and when to stop

Assuming that nothing is known about x when the estimation starts out ($n_f = 0$), the first thing to do is to apply the correction step to the first measurement. The lower blocks of the matrices z and G, and all blocks except R_e in R_w, will then be empty.

If the initial variance of x is infinite only in some directions ($0 < n_f < n$), the available information can be represented by a triple of matrices, T, $\hat{\bar{x}}_f$ and $\bar{P}_f$, and then plugged into the algorithm without any modification.

If the measurements provide enough information, the variance of the estimate will be finite in all directions ($n_i = 0$) after a number of iterations of the filter. Then, it is no longer necessary to use the algorithm described in this section and one can just as well use the standard Kalman filter (4.2), since the methods are equivalent for $n_i = 0$.

4.5 Simulation

To illustrate the use of the filter, consider a ball flying in a gravity field and with negligible air drag. The ball is tracked by a vision system, where each camera can provide an estimate of the line that intersects both the ball and the focal point of the camera, but no depth information is available. The process model is given on state-space form (4.1) with

$$\Phi(k) = \begin{bmatrix} 1 & 0 & 0 & h & 0 & 0 \\ 0 & 1 & 0 & 0 & h & 0 \\ 0 & 0 & 1 & 0 & 0 & h \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \forall k \tag{4.39}$$

$$\Gamma(k) = \begin{bmatrix} 0 \\ 0 \\ h^2/2 \\ 0 \\ 0 \\ h \end{bmatrix}, \forall k \tag{4.40}$$

$$u(k) = -g, \ \forall k \tag{4.41}$$

$$R_v(k) = 10^{-6} I_{6\times 6}, \ \forall k \tag{4.42}$$

where the state vector $x = [x_b \ \ y_b \ \ z_b \ \ \dot{x}_b \ \ \dot{y}_b \ \ \dot{z}_b]^T$ consists of three positions followed by three velocities. To make the example easy to follow, use the time step $h = 1$ and the earth gravitation constant

$g = 10$. Let the initial state of the system be $x(0) = [1\ \ 2\ \ 3\ \ 0\ \ 1\ \ 4]^T$. The trajectory of the ball is shown as a black curve in Fig. 4.2. The positions of the ball at the measuring instants are marked with green circles and the corresponding lines that are extracted from the images are marked in red. One camera observes the ball at time steps 0 and 2, and a second camera observes the ball at time step 1. The simulated measurements are given by

$$y(0) = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, C(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.43}$$

$$y(1) = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, C(1) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.44}$$

$$y(2) = \begin{bmatrix} 1 \\ -1.9 \end{bmatrix}, C(2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.3 & 0 & 0 & 0 \end{bmatrix} \tag{4.45}$$

$$R_e(0) = R_e(1) = R_e(2) = 10^{-4} I_{2\times 2} \tag{4.46}$$

Performing the state estimation on the given data gives the following results:

$$T_f(0|0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.47}$$

$$T_i(0|0) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.48}$$

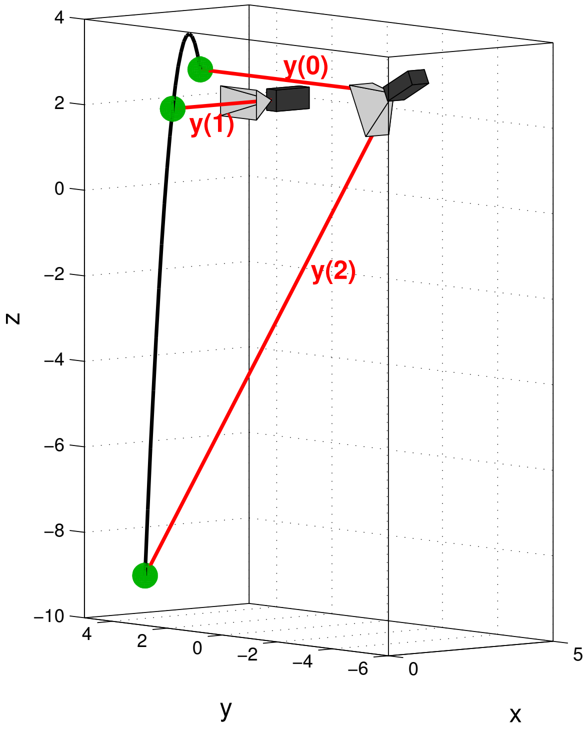


Figure 4.2 Simulated ball trajectory and measurements marked as green balls.

$$\hat{\bar{x}}_f(0|0) = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \tag{4.49}$$

$$\bar{P}_f(0|0) = 10^{-4} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.50}$$

$$T_f(1|0) = \begin{bmatrix} 0.707 & 0 \\ 0 & 0 \\ 0 & 0.707 \\ -0.707 & 0 \\ 0 & 0 \\ 0 & -0.707 \end{bmatrix} \tag{4.51}$$

$$T_i(1|0) = \begin{bmatrix} 0 & 0.707 & 0 & 0 \\ -0.707 & 0 & 0 & 0.707 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 0.707 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix} \tag{4.52}$$

$$\hat{\bar{x}}_f(1|0) = \begin{bmatrix} 0.707 \\ 5.657 \end{bmatrix} \tag{4.53}$$

$$\bar{P}_f(1|0) = 10^{-4} \begin{bmatrix} 0.51 & 0 \\ 0 & 0.51 \end{bmatrix} \tag{4.54}$$

$$T_f(1|1) = \begin{bmatrix} 0.707 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.707 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.55}$$

$$T_i(1|1) = \begin{bmatrix} 0 & -0.707 \\ 0 & 0 \\ 0 & 0 \\ 0 & -0.707 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.56}$$

$$\hat{\bar{x}}_f(1|1) = \begin{bmatrix} 0.707 \\ 3 \\ 2 \\ -6 \end{bmatrix} \tag{4.57}$$

$$\bar{P}_f(1|1) = 10^{-4} \begin{bmatrix} 0.51 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 2.02 \end{bmatrix} \tag{4.58}$$

$$T_f(2|1) = \begin{bmatrix} 0.447 & 0 & 0 & 0 \\ 0 & 0.707 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.894 & 0 & 0 & 0 \\ 0 & -0.707 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.59}$$

$$T_i(2|1) = \begin{bmatrix} 0 & -0.894 \\ 0.707 & 0 \\ 0 & 0 \\ 0 & -0.447 \\ 0.707 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.60}$$

$$\hat{\bar{x}}_f(2|1) = \begin{bmatrix} 0.447 \\ 2.121 \\ -9 \\ -16 \end{bmatrix} \tag{4.61}$$

$$\bar{P}_f(2|1) = 10^{-4} \begin{bmatrix} 0.214 & 0 & 0 & 0 \\ 0 & 0.51 & 0 & 0 \\ 0 & 0 & 5.03 & 3.02 \\ 0 & 0 & 3.02 & 2.03 \end{bmatrix} \tag{4.62}$$

$$T_f(2|2) = I_{6\times 6} \tag{4.63}$$

$$T_i(2|2) \in \mathbb{R}^{6\times 0} \tag{4.64}$$

$$\hat{\bar{x}}_f(2|2) = \begin{bmatrix} 1 \\ 2 \\ -9 \\ 0 \\ -1 \\ -16 \end{bmatrix} \tag{4.65}$$

$$\bar{P}_f(2|2) = 10^{-4} \begin{bmatrix} 1 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 9.08 & -3.77 & 0 & 9.08 & -2.27 \\ 0 & -3.77 & 5.03 & 0 & -3.77 & 3.02 \\ 0.5 & 0 & 0 & 0.518 & 0 & 0 \\ 0 & 9.08 & -3.77 & 0 & 10.1 & -2.27 \\ 0 & -2.27 & 3.02 & 0 & -2.27 & 2.03 \end{bmatrix} \tag{4.66}$$

Most insight on the estimation progress is given by studying T_f. The first measurement locates the ball on a line in the y_b-direction, which gives information about the position in the x_b- and z_b-directions. This is reflected in the columns of $T_f(0|0)$. After the time step the position is no longer known. Only linear combinations of the positions and velocities can be determined, as seen in $T_f(1|0)$. With the second measurement y_b and z_b are given. Since it is the second measurement in the z-direction, $\dot{z}_b$ can be determined. Still, no information about $\dot{y}_b$ is available and hence y_b is no longer known after the time step, as indicated by $T_f(2|1)$. The last measurement gives information in the remaining directions with infinite variance, and thus $T_f(2|2)$ spans the entire $\mathbb{R}^n$ and an estimate $\hat{x}(2|2) = T_f(2|2)\hat{\bar{x}}(2|2)$ can finally be calculated.

4.6 Discussion

Alternative frameworks to the one used in this thesis would be Bayesian networks or conditional expectations [Pearl, 1985].

The reason for doing the partitioning suggested in this thesis, is the difficulty of representing matrices with infinite singular values. An alternative approach to this is used in information filters [Kailath *et al.*, 2000]. Instead of the covariance matrix P and the state estimate $\hat{x}$, the information matrix $Y = P^{-1}$ and the information vector $\hat{y} = P^{-1}\hat{x}$ are used to represent the information about the system. If no information about the state is available, then this circumstance is conveniently represented by $Y = 0$. Measurement updates get a very simple form with information filters. However, the time update is complicated and does not work for a general system on the form (4.1), as stated in the introduction.

The equations (4.20) - (4.22) and (4.30) do not in general have a unique solution for T. In this thesis SVD based methods for solving the equations are suggested, but other methods can be used. The transformation T can be replaced by any T' fulfilling (4.11) - (4.18) such that $\mathcal{R}(T_f) = \mathcal{R}(T'_f)$. Of course $\hat{\bar{x}}_f$ and $\bar{P}_f$ have to be modified accordingly. In the example in Sec. 4.5 the T matrices were chosen to align the base vectors of $\bar{x}_f$ with the base vectors of the original state space as far as possible to improve human readability.

The presented initialization procedure is useful when very little is known about the initial state. If *a priori* knowledge is available, this should of course be used to improve the estimate.

The state $x = x(k)$, and hence also $\bar{x}_f$ and $\bar{x}_i$, are assumed to have exact and finite time-varying values, although not known exactly. More

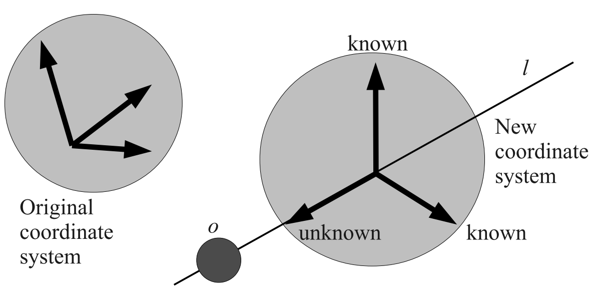


Figure 4.3 Example illustrating state partitioning. The position of the object o along the line l is unknown, and a new coordinate system, where as many basis vectors as possible are orthogonal to the line, is chosen.

specifically it is assumed that no information at all is available about $\bar{x}_i$, which is modeled as $\hat{\bar{x}}_i$ having infinite variance.

In general, it is not meaningful to give any numerical values of $\hat{x}$ if $n_f < n$. To see this recall (4.17). If a row in T_i has any non-zero element, the corresponding element of $\hat{x}$ is completely unknown. The knowledge about x can, however be described by T, $\hat{\bar{x}}_f$ and $\bar{P}_f$.

Even though it may not be possible to calculate any value of $\hat{x}$ in the original state space, the information in $\hat{\bar{x}}_f$ can still be useful. For instance, it may be of interest to know the altitude of an aerial vehicle before its longitude and latitude can be estimated.

As an example of state partitioning, consider the scenario in Fig. 4.3, where an object o is known to be near a given line l in 3D-space, but nothing is known about its position along the line. Choose a coordinate system such that its first two base vectors are orthogonal to l and the third base vector is parallel to l. The position of o can then be partly described by the first two components with a finite covariance matrix, even though the variance in the direction of the third component (parallel to l) is infinite.

4.7 Conclusions

A new way of initializing the Kalman filter has been presented, making it possible to calculate a state estimate that is not influenced by any guess of the initial value of the state. Instead, the estimate can be determined completely based on the first measurements.

5

Object Tracking with Measurements from Single or Multiple Cameras

5.1 Introduction

To be able to determine the position of a static object in 3D space by means of computer vision, the object has to be seen by cameras from at least two different view points, assuming that the size of the object is not known. The same applies for measuring the position of a moving object based on images captured at one single time instant. However, if the cameras are not synchronized in time, or if a moving object is not visible in all images, one can not rely on using matching pictures for making accurate position estimates of dynamical objects.

This chapter describes a strategy to track an object with known dynamical model, using a series of images where no pair has to be captured simultaneously. It even allows tracking of a point object in 3D space using a single static camera, and provides a convenient way of fusing data from multiple cameras.

The properties presented in the previous paragraph are very desirable for a ball-catching robot. They allow any number of cameras to be used, though at least two cameras are needed for good accuracy.

If the position of the ball can not be extracted from an image, due to occlusions or failed image analysis, the data from the other images can still be used to improve the estimate of the state of the ball.

An overview of tracking techniques is given in [Yilmaz *et al.*, 2006]. In the literature there are numerous examples of how computer vision and Kalman filtering can be used for robot control, *e.g.*, [Olsson *et al.*, 2003], [Olsson *et al.*, 2006] and [Marayong *et al.*, 2008]. In [Jia *et al.*, 2005] dynamic objects are tracked from a mobile platform using an extended Kalman filter and multiple interacting models. A method for tracking rigid objects that were partly occluded by other objects was suggested in [Sato *et al.*, 2003]. A method for exploiting the epipolar constraint between a pair of images is described in [Lamiroy *et al.*, 2000], to the purpose of visual servoing.

Tracking of flying balls, to the purpose of catching them with a robot, was previously treated in [Birbach *et al.*, 2008], [Birbach and Frese, 2009], and [Birbach *et al.*, 2011].

5.2 Problem Formulation

The goal is to track an object with a known process model. This is to be done with images captured from different view points and where possibly no images are captured simultaneously. The object to be tracked is described by the discrete-time state-space model

$$x(k+1) = \Phi(k)x(k) + \Gamma(k)u(k) + v(k) \tag{5.1}$$

where x is the state vector and u is a known input signal. Measurements are assumed to be on the form

$$y(k) = C(k)x(k) + e(k) \tag{5.2}$$

Note that the C-matrix is time dependent and will depend on the camera parameters and the image coordinates at that particular time step. The disturbances v and e are discrete-time white-noise processes with zero mean value and covariances

$$\begin{aligned} \mathrm{E}\left[v(i)v^T(j)\right] &= R_v\delta_{ij} \\ \mathrm{E}\left[v(i)e^T(j)\right] &= R_{ve}\delta_{ij} \\ \mathrm{E}\left[e(i)e^T(j)\right] &= R_e\delta_{ij} \end{aligned} \tag{5.3}$$

5.3 Methods

State Estimation

A Kalman filter was used to estimate the state of the tracked object. The update law can be described by (5.4) - (5.9). Here, for example, $\hat{x}(k+1|k)$ denotes the estimate of x at sample $k+1$ based on measurements up to sample k, and $P(k+1|k)$ the covariance of that estimate.

$$\begin{aligned}
\hat{x}(k|k) &= \hat{x}(k|k-1) + K_f\left(y(k) - C(k)\hat{x}(k|k-1)\right) && (5.4)\\
K_f(k) &= P(k|k-1)C^T(k) && (5.5)\\
&\quad \cdot \left(R_e(k) + C(k)P(k|k-1)C^T(k)\right)^{-1} &&\\
P(k|k) &= P(k|k-1) - K_f(k)C(k)P(k|k-1) && (5.6)\\
\hat{x}(k+1|k) &= \Phi(k)\hat{x}(k|k-1) + \Gamma(k)u(k) && (5.7)\\
&\quad + K(k)\left(y(k) - C(k)\hat{x}(k|k-1)\right) &&\\
K(k) &= \left(\Phi(k)P(k|k-1)C^T(k) + R_{ve}(k)\right) && (5.8)\\
&\quad \cdot \left(R_e(k) + C(k)P(k|k-1)C^T(k)\right)^{-1} &&\\
P(k+1|k) &= \Phi(k)P(k|k-1)\Phi^T(k) + R_v(k) && (5.9)\\
&\quad - K(k)\left(\Phi(k)P(k|k-1)C^T(k) + R_{ve}(k)\right)^T &&
\end{aligned}$$

Homogeneous Coordinates

Homogeneous coordinates [Möbius, 1827; Graustein, 1930] will be used extensively in this thesis. To simplify the notation, "$\sim$" is defined according to

$$\begin{aligned}
&\mathbf{x_1} \sim \mathbf{x_2} \iff \\
&\mathbf{x_1} = \lambda \mathbf{x_2} \text{ for some } \lambda \neq 0
\end{aligned} \qquad (5.10)$$

where $\mathbf{x_1}$ and $\mathbf{x_2}$ are homogeneous coordinate vectors.

The projection matrix, $P_c \in \mathbb{R}^{3\times 4}$, of a camera is a matrix such that

$$\mathbf{x} \sim P_c\mathbf{X} \qquad (5.11)$$

where $\mathbf{X}$ is a point in space and $\mathbf{x}$ is its projection onto the image plane.

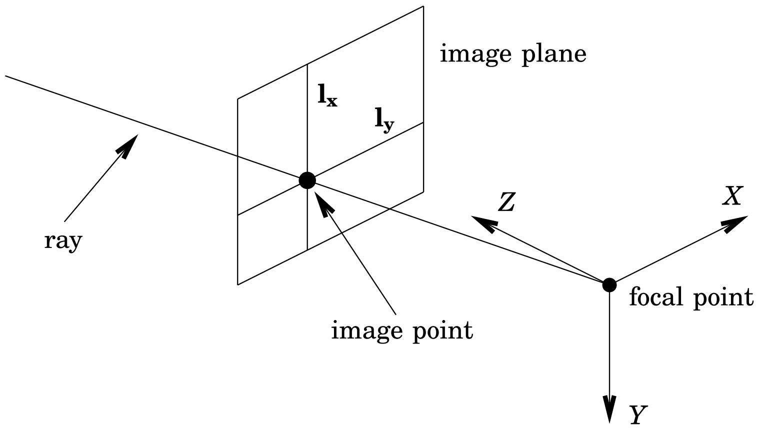


Figure 5.1 Illustration of a ray. An object projected onto some image point can be located anywhere along the corresponding ray.

5.4 Transforming Image Data for the Kalman Filter

From the position of a feature point in a single image it is possible to determine a line in 3D-space along which the point must be. In this thesis this line will be referred to as a ray (cf. Fig. 5.1). The ray parameters then have to be related to the process model states in some way in order to be used as input to the Kalman filter.

Extracting Rays

To determine the ray for a point in an image, first two lines through the point in the image are chosen. The easiest way to do this is to take one horizontal and one vertical line. Each one of these lines uniquely defines a plane through the line and the focal point. Finally, the ray can be determined as the intersection of these two planes. A line **l** in an image can be represented as the points fulfilling the equation

$$\mathbf{l}^T\mathbf{x} = 0 \tag{5.12}$$

where $\mathbf{l} = [l_1\ l_2\ l_3]^T$ and $\mathbf{x} = [x\ y\ 1]^T$.

A plane $\boldsymbol{\pi}$ in space can similarly be represented by the points fulfilling

$$\boldsymbol{\pi}^T\mathbf{X} = 0 \tag{5.13}$$

where $\boldsymbol{\pi} = [\pi_1\ \pi_2\ \pi_3\ \pi_4]^T$ and $\mathbf{X} = [X\ \ Y\ \ Z\ \ 1]^T$.

Moreover, if the plane $\boldsymbol{\pi}$ is projected onto the line $\mathbf{l}$, then

$$\mathbf{x} \sim P_c\mathbf{X} \tag{5.14}$$

where P_c is the projection matrix of the camera. Inserting (5.14) into the equation for the image line (5.12) we get

$$0 = \mathbf{l}^T\mathbf{x} \sim \mathbf{l}^T P_c\mathbf{X} = (P_c^T\mathbf{l})^T\mathbf{X} \tag{5.15}$$

Hence it can be can concluded that the points that project onto the line $\mathbf{l}$ reside in the plane defined by

$$\boldsymbol{\pi} = P_c^T\mathbf{l} \tag{5.16}$$

If the coordinate of a feature point in an image is (x, y), a convenient choice of lines through this point is $\mathbf{l_x} = [-1\ 0\ x]^T$ and $\mathbf{l_y} = [0\ \ -1\ y]^T$ (cf. Fig. 5.1) corresponding to the planes defined by

$$\begin{aligned} \boldsymbol{\pi_x} &= P_c^T\mathbf{l_x} \\ \boldsymbol{\pi_y} &= P_c^T\mathbf{l_y} \end{aligned} \tag{5.17}$$

The ray can then be described as the intersection of the planes $\boldsymbol{\pi_x}$ and $\boldsymbol{\pi_y}$.

Flying Ball Example

To illustrate how the rays are transformed into Kalman filter input data, a simple example of a dynamical model is used; a ball flying under the influence of gravity and without air friction. The state vector is

$$x = \begin{bmatrix} x_b & y_b & z_b & \dot{x}_b & \dot{y}_b & \dot{z}_b \end{bmatrix}^T \tag{5.18}$$

where the first three states represent the position and the last three states represent the velocity in the coordinate system of Fig. 3.1. The discrete-time process model in the form of (5.1) is

$$\begin{aligned}
\Phi(k) &= \begin{bmatrix} 1 & 0 & 0 & h & 0 & 0 \\ 0 & 1 & 0 & 0 & h & 0 \\ 0 & 0 & 1 & 0 & 0 & h \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \forall k \\
\Gamma(k) &= \begin{bmatrix} 0 \\ h^2/2 \\ 0 \\ 0 \\ h \\ 0 \end{bmatrix}, \forall k \\
u(k) &= -g, \ \forall k
\end{aligned} \tag{5.19}$$

where g is the earth gravity constant and h is the sample period.

Including Constraints into the Kalman Filter

The planes in (5.17) each put the constraint

$$\boldsymbol{\pi}^T \mathbf{X} = 0 \tag{5.20}$$

on the feature point position $\mathbf{X}$. This can be rewritten as a constraint on the state vector (5.18) of the process:

$$0 = \boldsymbol{\pi}^T \mathbf{X} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ z_d \\ 1 \end{bmatrix} \Longleftrightarrow$$

$$\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = -\pi_4 \Longleftrightarrow$$

$$\begin{aligned} -\pi_4 &= cx \\ \text{where } c &= \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & 0 & 0 & 0 \end{bmatrix} \\ \text{and } x &= \begin{bmatrix} x_d & y_d & z_d & \dot{x}_d & \dot{y}_d & \dot{z}_d \end{bmatrix}^T \end{aligned} \tag{5.21}$$

If $\boldsymbol{\pi}$ is normalized so $\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2} = 1$, $-\pi_4$ can be interpreted as the signed length of the orthogonal projection of the feature point position onto the direction $[\pi_1 \; \pi_2 \; \pi_3]$.

Each image with a measurement of a feature position gives two constraints, each specified by a row vector c in (5.21); one in the x direction and one in the y direction of the image. If several images are available, this can be handled simply by adding rows in the C-matrix of (5.2):

$$C = \begin{bmatrix} c_{ax} \\ c_{ay} \\ c_{bx} \\ c_{by} \\ \vdots \end{bmatrix} \tag{5.22}$$

where a and b denote different images and x and y the different measurement directions. Similarly, the measurement vector y of (5.2) is composed of the negative fourth component of the planes $\boldsymbol{\pi}$ in (5.17):

$$y = \begin{bmatrix} -\pi_{4ax} \\ -\pi_{4ay} \\ -\pi_{4bx} \\ -\pi_{4by} \\ \vdots \end{bmatrix} \tag{5.23}$$

If no measurement at all is available at some time step, the last term of (5.4), (5.6), (5.7), and (5.9) simply disappears.

5.5 Detection of False Positives

When using computer vision, two different kinds of noise are common. If the correct object is detected in the image, the value of the position estimate has a probability distribution close to the true value. There is also a risk that the position estimate is based on the wrong feature, which typically results in outliers, *i.e.*, estimates with values far away from the main body of the distribution. For good tracker performance, the outliers should be detected and discounted.

Outlier Detection

In this subsection the time index is left out in the notation, since it is the same for all quantities.

The measurement is given on the form

$$y = Cx + e \tag{5.24}$$

with $\mathrm{E}[e] = 0$ and $\mathrm{E}[ee^T] \equiv R_e$. Based on the state estimate $\hat{x}$ the expectation value of the measurement is

$$\hat{y} = C\hat{x} \tag{5.25}$$

The state estimate error is $\tilde{x} = x - \hat{x}$ with $\mathrm{E}[\tilde{x}] = 0$, $\mathrm{E}[\tilde{x}\tilde{x}^T] = R_x$ and $\mathrm{E}[\tilde{x}e^T] = 0$. The best estimate of the measurement error is

$$\tilde{y} = y - \hat{y} = C\tilde{x} + e \tag{5.26}$$

with the properties $\mathrm{E}[\tilde{y}] = 0$ and $R_y \equiv \mathrm{E}[\tilde{y}\tilde{y}^T] = CR_xC^T + R_e$.

If all distributions are Gaussian the probability density function for $\tilde{y}$ is

$$f(\tilde{y}) = \frac{1}{(2\pi)^{n/2}\,|R_y|^{1/2}} \exp\left(-\frac{1}{2}\tilde{y}^T R_y^{-1}\tilde{y}\right) \tag{5.27}$$

where n is the dimension of the state vector. Then it is natural to consider the measurement as an outlier if a condition on the form

$$\tilde{y}^T R_y^{-1}\tilde{y} > p^2 \tag{5.28}$$

is fulfilled, where p is a tuning parameter. Inserting (5.28) into (5.27) it can easily be seen that all points outside the decision boundary have a lower probability density than all points inside it.

Managing of Multiple Trajectories Simultaneously

It is useful to have a layer on top of the Kalman filter, keeping track of several state vectors, each representing the trajectory of one ball. The obvious advantage of this is that the trajectories of several balls can be tracked simultaneously. A more important advantage has to do with outlier detection. In each iteration of the Kalman filter, the measurements are discarded if they are not close to what is expected based on the previous estimates, as described in the previous subsection. If the measurement initiating the trajectory is a false positive, successive correct measurements will be discarded, since they are not close to what is expected after the incorrect measurement. This means that one incorrect measurement will block the system. The algorithm used to handle this situation is described by the following pseudo code, which is run on every measurement:

```
for all trajectories
    if the measurement is not an outlier to this trajectory
        perform iteration of Kalman filter
    if the trajectory has not received a measurement for a while
        throw it away
if the measurement did not match any existing trajectory
    create a new trajectory
```

5.6 Results

Figure 5.2 shows example images of a thrown ball from the perspectives of two cameras. The detected positions of the ball from a sequence of such images are marked with green dots connected with lines. Different subsets of these data were used as input to the tracker described in this chapter. The results are shown in Figs. 5.3 – 5.6. For all cases the estimate was initialized by the method described in Chapter 4. The model (5.19) was used with

$$R_v = 0.001^2 \cdot \mathrm{diag}[\ 2^2 \quad 2^2 \quad 2^2 \quad 5^2 \quad 5^2 \quad 5^2\ ] \tag{5.29}$$

$$R_{ve} = 0 \tag{5.30}$$

$$R_v = \mathrm{diag}[\ 0.01^2 \quad 0.01^2\ ] \tag{5.31}$$

The accuracy of the estimates are illustrated by ellipsoids on the form $(z - \hat{z})^T R^{-1} (z - \hat{z}) = p^2$, where z is the position or velocity and

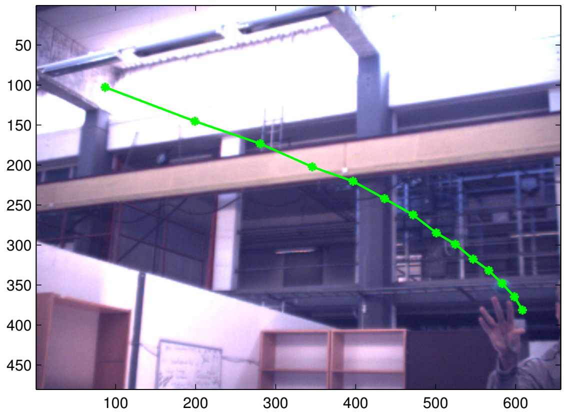
R is the covariance matrix of its estimate $\hat{z}$. In Figs. 5.3 – 5.6 $p = 3$, which means that there is a 97 % probability that the real value is within the ellipsoid, assuming that the distributions are Gaussian.

Figures 5.3 and 5.4 show the estimates based on all the data points in Fig. 5.2. The position estimates formed a smooth parabola and the estimated velocities decreased in the vertical direction linearly with time, as expected by the model due to the gravity. The size of the uncertainty ellipsoids decreased with the number of measurements. The tracker classified one measurement as a false positive, which is indicated by a red circle in Fig. 5.2.

It is possible to track the ball using only a single fixed camera, since the ball is observed from slightly different angles as it moves across the image. In Fig. 5.5 the blue curve shows the estimates based only on measurements from the lower image in Fig. 5.2. The uncertainty ellipsoids were initially very oblong, since the position in the direction of the rays was very uncertain. As the ball moved and got observed from different angles, the estimate variance decreased drastically. The estimated trajectory based on the images from both cameras is plotted in red for reference. The uncertainty ellipsoids of the estimates in the blue trajectory enclose the corresponding estimates in the red trajectory as expected. When both cameras are used a full state estimate can be obtained after two time samples, while three time samples are needed when only one camera is used. Hence the first point in the red trajectory has no corresponding position estimate in the blue trajectory.

When the blue trajectory in Fig. 5.6 was generated, only the data from the odd time indices in the upper image and the data from the even time indices in the lower image of Fig. 5.2 were used. After four measurements, the estimate converged to almost coincide with the red curve, generated by all the data points in both images of Fig. 5.2, and the 97 % confidence ellipsoids are almost too small to be visible. The example illustrates that the filter works well if images from cameras at two different locations are provided, even if the images are not captured simultaneously.

The tracker was implemented on the ball-catcher described in Chapter 3, and was successfully used to catch thrown balls. The performance of the ball-catcher was limited by the accuracy of the image analysis and the speed and acceleration of the robot, which were limited to 0.5 [m/s] and 13 [m/s^2] respectively. The diameters of the ball and the hole



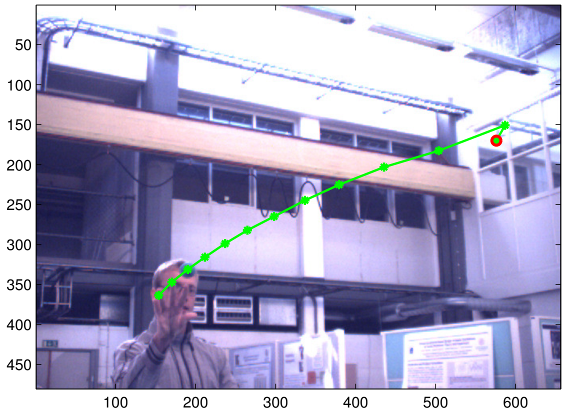


Figure 5.2 Example images from a sequence of images of a thrown ball, as seen from the two different cameras. The detected ball positions from the entire sequence are marked with green dots. A detected ball position that was classified as an outlier by (5.28) is marked with a red circle. In these example images the ball can be seen at the third green dot in the trajectory.

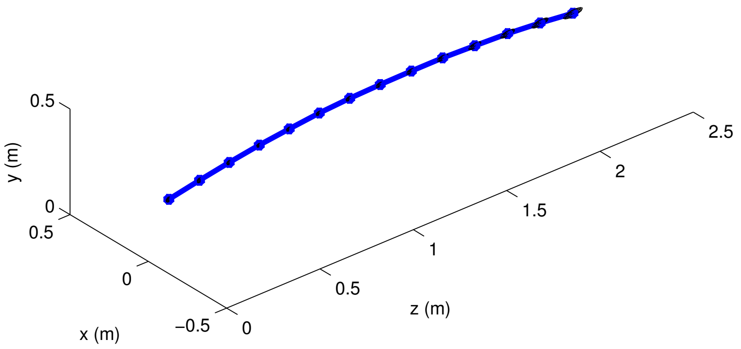


Figure 5.3 Position estimates based on all measured image coordinates from both cameras in Fig. 5.2.

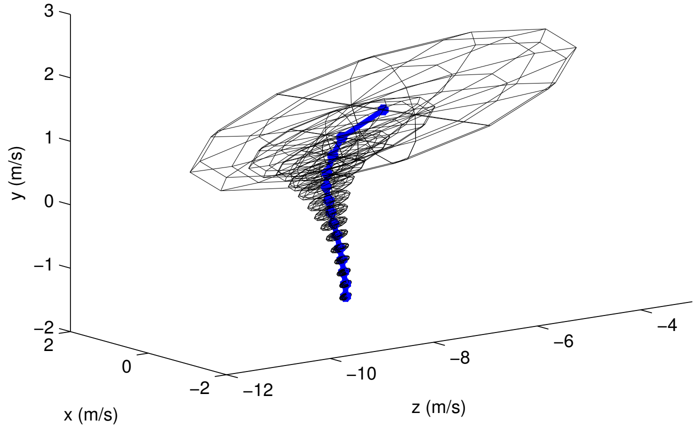


Figure 5.4 Velocity estimates based on all measured image coordinates from both cameras in Fig. 5.2.

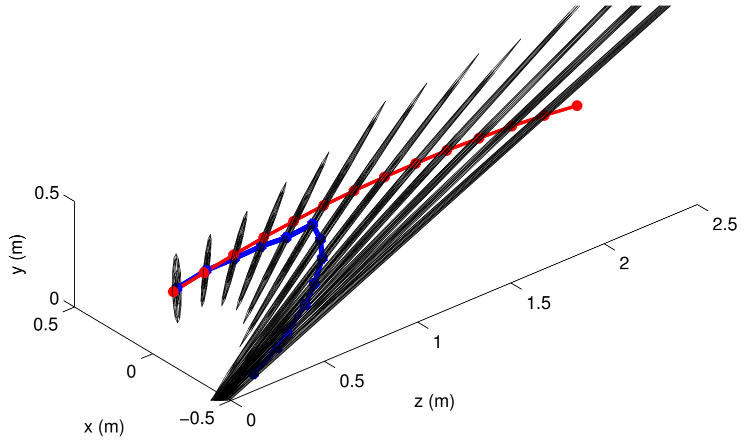


Figure 5.5 The blue curve shows position estimates based on the measured image coordinates from only the upper image in Fig. 5.2. For comparison, estimates based on the data in both images of Fig. 5.2 are shown in red.

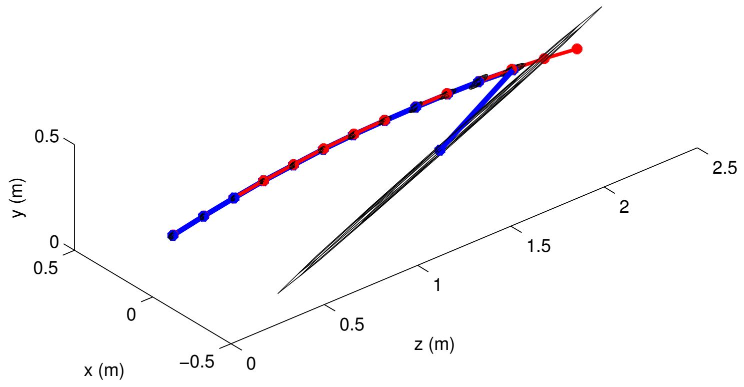


Figure 5.6 The blue curve shows estimates based on data from only one camera for odd time indices, and data from another camera for even time indices. Estimates based on all the data from both cameras are shown in red.

were 5 [cm] and 6 [cm] respectively. A video of the robot catching balls is available on Youtube [Ball-Catcher Video, 2009].

5.7 Discussion

No explicit triangulation was done in the proposed procedure. However, if the rows in C of (5.22) correspond to measurements in many different directions over time, an estimate with low variance in all directions can be obtained.

A benefit of the tracking approach described in this chapter is that it easily handles any number of cameras. In (5.22) and (5.23) two more rows are simply added to C and y for every camera that captured an image at the same sampling instant. Another advantage is that a measurement can be used even if there are no valid measurements from any other cameras, so that no 3D position estimate could be made from the data from only that sampling instant. It was shown that a ball can be tracked using a single camera. Accuracy is, however, significantly improved if two or more cameras at different positions are used.

If a continuous-time model of the process is available, the procedure can be generalized to completely aperiodic measurements. The expressions (5.1) - (5.9) then have to be recalculated at every measurement to reflect the actual time that passed since the previous measurement.

Each row of C and y in (5.22) and (5.23) describes a hyperplane in the state space. The above proposed procedure can thus be extended to any feature that can be expressed as a hyperplane in the state space.

5.8 Conclusions

This chapter describes a strategy for tracking dynamical objects with computer vision. A novel way of mapping image space data to a Kalman filter in 3D space is described. It provides a simple way to use data from an arbitrary number of pictures captured simultaneously. In combination with a dynamical model of the tracked object, it enables determination of the position of the object in 3D space without any two images being captured at the same time. It also allows tracking of objects in 3D space, using only a single static camera.

A method to determine whether a measurement is an outlier, based on the covariances of the measurement and the position estimate, is presented.

6

Robot Trajectory Generation with Uncertain Target Point

6.1 Introduction

The tracker described in Chapter 5 can be used to estimate the position where a ball should be caught. When few measurements are available, the variance of the estimated state of the ball may be large, and since there is a long distance to the robot, the uncertainties will be magnified when the trajectory is extrapolated to the future. As the ball moves along the trajectory and more measurements of the position are performed, the variance of the state estimate decreases. In order to have good performance of the ball-catcher, the robot should start moving the box toward the catching position as soon as there is an estimate, and then modify the end point of the trajectory on the fly when better estimates become available.

6.2 Problem Formulation

The task is to generate a trajectory for moving a robot from one pose to another and reach the final position before a given deadline. While the robot is moving along the trajectory, the final point may change both in space and time and the trajectory should be modified accordingly.

In order to reduce wear on the mechanical parts it is desirable not to shake the robot excessively, even if the target point moves around very much.

6.3 Method

Decomposition

Since only the end-point of the trajectory is of interest for the ball-catcher, trajectory generation in joint space and in Cartesian space are both possible solutions. Computation in joint space, however, leads to simpler calculations and is less sensitive to kinematic singularities. The strategy suggested is thus to solve the inverse kinematics of the target point, *i.e.*, find the target position for each joint, and then do trajectory generation for each joint separately.

Assumptions

When a new destination point is acquired at time t_0 the joint has an initial position x_0 and initial velocity v_0. The time left to the deadline t_d is $T = t_d - t_0$ and the final position is x_f, where the velocity should be zero. The velocity and acceleration are limited to V and A respectively. The total distance left to go is denoted by $X = x_f - x_0$.

Trajectory Primitive

In the following, two strategies for trajectory generation will be described. They both split the movement into three parts:

- apply acceleration a^* during time t_1
- apply acceleration 0 during time t_2
- apply acceleration $-a^*$ during time t_3

This trajectory primitive can be seen as bang-bang control in the acceleration, used to minimize the maximum acceleration, when moving a given distance within a given time. The middle time interval with zero acceleration is needed in case the velocity reaches its maximum value.

Let v_t be the velocity in the time interval when the acceleration is zero. Since the acceleration has different signs before and after this interval, v_t is always an extremum of the velocity, and is hence convenient for determining whether an attempt to generate a trajectory violates the velocity constraints.

Strategy 1: Minimize Maximum Acceleration

This subsection describes a strategy that uses the smallest possible acceleration needed to reach the destination before the deadline. If the deadline cannot be met, the destination will be reached at time t_f, with $t_f > t_d$. The smallest possible t_f that does not violate the constraints on velocity and acceleration will then be used. It will be shown that there always is exactly one solution.

The problem can be formally described by equations (6.1)–(6.11), where $x(t)$, $v(t)$ and $a(t)$ are the position, velocity and acceleration as functions of time.

Initial conditions

$$x(t_0) = x_0 \tag{6.1}$$
$$v(t_0) = v_0 \tag{6.2}$$

Dynamics

$$v(t) = v_0 + \int_{t_0}^{t} a(t)\,dt \tag{6.3}$$
$$x(t) = x_0 + \int_{t_0}^{t} v(t)\,dt \tag{6.4}$$

Constraints

$$x(t_f) = x_f \tag{6.5}$$
$$v(x_f) = 0 \tag{6.6}$$

$$t_1 + t_2 + t_3 = t_f - t_0 \tag{6.7}$$
$$t_1 \geq 0, \quad t_2 \geq 0, \quad t_3 \geq 0 \tag{6.8}$$

$$-V \leq |v(t)| \leq V, \forall t \tag{6.9}$$
$$-A \leq |a(t)| \leq A, \forall t \tag{6.10}$$

$$a(t) = \begin{cases} a^*, & t_0 \leq t < t_0 + t_1 \\ 0, & t_0 + t_1 \leq t < t_0 + t_1 + t_2 \\ -a^*, & t_0 + t_1 + t_2 \leq t \leq t_0 + t_1 + t_2 + t_3 \end{cases} \tag{6.11}$$

The goal is to find t_1, t_2, t_3 and a^* that solve (6.1)–(6.11). The quantities x_0, v_0, x_f, V and A are given as a part of the problem formulation.

Inserting (6.11) into (6.3) and (6.4) results in

$$\begin{aligned} X &= x_f - x_0 \\ &= v_0(t_1 + t_2 + t_3) + a^* \left(\frac{t_1^2}{2} + t_1 t_2 + t_1 t_3 - \frac{t_3^2}{2} \right) \end{aligned} \tag{6.12}$$

$$\begin{aligned} v_t &= v(t), \ t_0 + t_1 \leq t < t_0 + t_1 + t_2 \\ &= v_0 + a^* t_1 \end{aligned} \tag{6.13}$$

$$v(t_f) = v_0 + a^*(t_1 - t_3) \tag{6.14}$$

where v_t is the velocity when the acceleration switches sign. Combining (6.6), (6.13) and (6.14) gives

$$v_t = v_0 + a^* t_1 = a^* t_3 \iff \tag{6.15}$$

$$\begin{cases} t_1 = (v_t - v_0)/a^* \\ t_3 = v_t / a^* \end{cases} \tag{6.16}$$

No saturation of velocity, no saturation of acceleration. First assume the case where there is no saturation of the velocity, which gives

$$t_2 = 0 \tag{6.17}$$

Further assume that the deadline can be met without violating the acceleration constraints, which gives the constraint

$$T = t_1 + t_3 \tag{6.18}$$

Inserting (6.16) and (6.17) into (6.18) and (6.12) gives

$$T = \frac{2v_t - v_0}{a^*} \tag{6.19}$$

$$X = \frac{2v_t^2 - v_0^2}{2a^*} \tag{6.20}$$

By taking X/T, a^* can be eliminated, resulting in

$$\begin{aligned}
\frac{X}{T} &= \frac{v_t^2 - v_0^2/2}{2v_t - v_0} \Longleftrightarrow \\
0 &= v_t^2 - 2\frac{X}{T}v_t + \frac{X}{T}v_0 - \frac{v_0^2}{2} \\
&= \left(v_t - \frac{X}{T}\right)^2 - \frac{X^2}{T^2} + \frac{X}{T}v_0 - \frac{v_0^2}{2} \\
&= \left(v_t - \frac{X}{T}\right)^2 - \left(\frac{X}{T} - \frac{v_0}{2}\right)^2 - \left(\frac{v_0}{2}\right)^2 \Longleftrightarrow \\
v_t &= \frac{X}{T} + k\sqrt{\left(\frac{X}{T} - \frac{v_0}{2}\right)^2 + \left(\frac{v_0}{2}\right)^2}
\end{aligned} \tag{6.21}$$

where $k = \pm 1$. To determine the sign of k, consider (6.16). In order for the times to be non-negative, as required by (6.8), both v_t and $(v_t - v_0)$ must have the same sign as a^* or be zero, which implies

$$0 \leq v_t(v_t - v_0) \tag{6.22}$$

Inserting (6.21) into (6.22) gives.

$$0 \leq 2\left(\frac{X}{T} - \frac{v_0}{2}\right)\left(\left(\frac{X}{T} - \frac{v_0}{2}\right) + k\sqrt{\left(\frac{X}{T} - \frac{v_0}{2}\right)^2 + \left(\frac{v_0}{2}\right)^2}\right) \tag{6.23}$$

Since

$$\sqrt{\left(\frac{X}{T}-\frac{v_0}{2}\right)^2+\left(\frac{v_0}{2}\right)^2} \geq \left|\frac{X}{T}-\frac{v_0}{2}\right| \tag{6.24}$$

(6.23) is fulfilled for

$$k = \begin{cases} 1, & X/T \geq v_0/2 \\ -1, & X/T \leq v_0/2 \end{cases} \tag{6.25}$$

If $X/T = v_0/2$, either sign of k is valid. The difference it will make is whether $t_1 = 0$ or $t_3 = 0$. In both cases $a(t)$ is the same.

The acceleration can be solved for in, *e.g.* (6.19), giving

$$a^* = \frac{2v_t - v_0}{T} \tag{6.26}$$

To see that a^* does not have different sign than v_t or $(v_t - v_0)$, and hence fulfilling (6.8) and (6.16), remember that v_t and $(v_t - v_0)$ do not have different signs and that $T \geq 0$. Then consider

$$\begin{aligned} v_t + (v_t - v_0) &= 2v_t - v_0 \\ &= a^*T \end{aligned} \tag{6.27}$$

Saturation of velocity, no saturation of acceleration. Now consider the case when the velocity is saturated and hence

$$v_t = V\,\mathrm{sign}(X) \tag{6.28}$$

It is assumed that the deadline can be met without violating the acceleration constraints, which gives the constraint

$$\begin{aligned} &T = t_1 + t_2 + t_3 \iff \\ &t_2 = T - t_1 - t_3 \end{aligned} \tag{6.29}$$

Inserting (6.29) into (6.12) gives

$$\begin{aligned} X &= v_0 T + a^* \left(\frac{t_1^2}{2} + t_1(T - t_1 - t_3) + t_1 t_3 - \frac{t_3^2}{2} \right) \\ &= (v_0 + a^* t_1)T - a^* \left(\frac{t_1^2}{2} + \frac{t_3^2}{2} \right) \\ &= v_t T - \frac{(v_t - v_0)^2 + v_t^2}{2a^*} \end{aligned} \tag{6.30}$$

where the last equality follows from (6.16). Solving (6.30) for a^* gives

$$a^* = \frac{(v_t - v_0)^2 + v_t^2}{2(v_t T - X)} \tag{6.31}$$

Now we have v_t and a^* and can get t_1 and t_3 from (6.16) and get t_2 from (6.29). For the solution to be valid we must have $t_3 \geq 0$, which in combination with (6.16) gives that v_t and a^* must have the same sign. Applied to (6.31) this gives that v_t and $(v_t T - X)$ must have the same sign, which in combination with (6.28) gives

$$|X| < VT \iff \frac{|X|}{T} < V \tag{6.32}$$

This requirement is very intuitive, since it means that the mean velocity must be less than the maximum velocity.

No saturation of velocity, saturation of acceleration. In this case it is assumed that the velocity will not be saturated, so $t_2 = 0$. The deadline, however, can not me met, so (6.18) must be relaxed and the maximum possible acceleration will be applied to reach the target as soon as possible. This gives $a^* = \pm A$, where the sign will be determined later. The conditions for (6.20) being valid are fulfilled, and solving for v_t gives

$$v_t = k\sqrt{Xa^* + v_0^2/2} \tag{6.33}$$

where $k = \pm 1$. It will now be shown that there is always exactly one solution to (6.33) fulfilling

$$Xa^* + v_0^2/2 \geq 0 \tag{6.34}$$
$$t_1 = (v_t - v_0)/a^* \geq 0 \tag{6.35}$$
$$t_3 = v_t/a^* \geq 0 \tag{6.36}$$

The condition (6.36) gives

$$\begin{aligned} 0 \leq \frac{v_t}{a^*} = \frac{k}{a^*}\sqrt{Xa^* + v_0^2/2} \iff \\ k = \operatorname{sign}(a^*) \end{aligned} \tag{6.37}$$

- Now assume that $\boxed{a^* = A > 0}$.
  Through (6.33) and (6.37) this gives $v_t \geq 0$. Condition (6.34) is then equivalent to

  $$2XA \geq -v_0^2 \tag{6.38}$$

  Condition (6.35) is trivial to verify if $v_0 \leq 0$ and in the case $v_0 > 0$ it is equivalent to

  $$\begin{aligned} v_t - v_0 \geq 0 \iff \\ \sqrt{XA + v_0^2/2} \geq v_0 \iff \\ XA + v_0^2/2 \geq v_0^2 \iff \\ 2XA \geq v_0^2 \end{aligned} \tag{6.39}$$

  Condition (6.38) is valid for all v_0 and (6.39) only for $v_0 > 0$. Both conditions are fulfilled if and only if

  $$2XA \geq v_0|v_0| \tag{6.40}$$

- Now assume that $\boxed{a^* = -A < 0}$.
  Through (6.33) and (6.37) this gives $v_t \leq 0$. Condition (6.34) is then equivalent to

  $$2XA \leq v_0^2 \tag{6.41}$$

Condition (6.35) is trivial to verify if $v_0 \geq 0$ and in the case $v_0 < 0$ it is equivalent to

$$\begin{aligned} v_t - v_0 \leq 0 &\iff \\ -\sqrt{-XA + v_0^2/2} \leq v_0 &\iff \\ -XA + v_0^2/2 \geq v_0^2 &\iff \\ 2XA \leq -v_0^2 \end{aligned} \tag{6.42}$$

Condition (6.41) is valid for all v_0 and (6.42) only for $v_0 < 0$. Both conditions are fulfilled if and only if

$$2XA \leq v_0|v_0| \tag{6.43}$$

Since either (6.40) or (6.43) is fulfilled for all values of X and v_0, there is always a solution for either $a^* = A$ or $a^* = -A$. On the curve $2XA = v_0|v_0|$ both values of a^* are valid. The sign of a^* will only affect whether $t_1 = 0$ or $t_3 = 0$ and in both cases $a(t)$ will be the same.

Inserting (6.33) into (6.36) gives

$$\begin{aligned} t_3 &= \frac{v_t}{a^*} \\ &= \frac{\operatorname{sign}(a^*)}{a^*}\sqrt{Xa^* + v_0^2/2} \\ &= \frac{1}{A}\sqrt{Xa^* + v_0^2/2} \end{aligned} \tag{6.44}$$

Combining (6.35) and (6.36) gives

$$\begin{aligned} t_1 &= \frac{v_t}{a^*} - \frac{v_0}{a^*} \\ &= t_3 - \frac{v_0}{a^*} \end{aligned} \tag{6.45}$$

Saturation of velocity, saturation of acceleration. In this case it is assumed that both velocity and acceleration become saturated so

$$v_t = V \operatorname{sign}(X) \tag{6.46}$$

$$a^* = A \operatorname{sign}(X) \tag{6.47}$$

and the deadline can not be met. Inserting (6.16) into (6.12) gives

$$\begin{aligned} X &= \frac{v_t v_0 - v_0^2}{a^*} + v_0 t_2 + \frac{v_t v_0}{a^*} \\ &\quad + a^* \left(\frac{v_t^2 - 2v_t v_0 + v_0^2}{2a^{*2}} + \frac{t_2}{a^*}(v_t - v_0) + \frac{v_t^2 - v_t v_0}{a^{*2}} - \frac{v_t^2}{2a^{*2}} \right) \\ &= v_t t_2 + \frac{2v_t^2 - v_0^2}{2a^*} \end{aligned} \tag{6.48}$$

Solving (6.48) for t_2 gives

$$\begin{aligned} t_2 &= \frac{X}{v_t} - \frac{v_t}{a^*} + \frac{v_0^2}{2v_t a^*} \\ &= \frac{|X|}{V} - \frac{V}{A} + \frac{v_0^2}{2VA} \end{aligned} \tag{6.49}$$

The durations t_1 and t_3 can be obtained from (6.16).

Summary of solution. In the following the solution to equations (6.1)–(6.11) for different cases of saturation will be summarized. First try using Case 1, where neither velocity nor acceleration is saturated.

- **Case 1: velocity not saturated, acceleration not saturated**

$$\begin{aligned} v_t &= \begin{cases} X/T + \sqrt{(X/T - v_0/2)^2 + (v_0/2)^2}, & X/T \geq v_0/2 \\ X/T - \sqrt{(X/T - v_0/2)^2 + (v_0/2)^2}, & X/T \leq v_0/2 \end{cases} \\ a^* &= (2v_t - v_0)/T \\ t_1 &= (v_t - v_0)/a^* \\ t_2 &= 0 \\ t_3 &= v_t/a^* \end{aligned}$$

If $|v_t| > V$ the velocity constraint is violated and the solution is not valid. Try using Case 2.

If $|a^*| > A$ the acceleration constraint is violated and the solution is not valid. Try using Case 3. The deadline will not be met.

- **Case 2: velocity saturated, acceleration not saturated**

$$\begin{aligned}
v_t &= V\,\mathrm{sign}(X) \\
a^* &= \frac{(v_t - v_0)^2 + v_t^2}{2(v_t T - X)} \\
t_1 &= (v_t - v_0)/a^* \\
t_3 &= v_t/a^* \\
t_2 &= T - t_1 - t_3
\end{aligned}$$

  If $VT < |X|$ or $|a^*| > A$ the solution is not valid. Use Case 4. The deadline will not be met.

- **Case 3: velocity not saturated, acceleration saturated**

$$\begin{aligned}
a^* &= \begin{cases} A, & 2XA \geq v_0|v_0| \\ -A, & 2XA \leq v_0|v_0| \end{cases} \\
v_t &= \mathrm{sign}(a^*)\sqrt{Xa^* + v_0^2/2} \\
t_3 &= \frac{1}{A}\sqrt{Xa^* + v_0^2/2} \\
t_2 &= 0 \\
t_1 &= t_3 - v_0/a^*
\end{aligned}$$

  If $|v_t| > V$ the velocity constraint is violated and the solution is not valid. Use Case 4.

- **Case 4: velocity saturated, acceleration saturated**

$$\begin{aligned}
a^* &= A\,\mathrm{sign}(X) \\
v_t &= V\,\mathrm{sign}(X) \\
t_1 &= (v_t - v_0)/a^* \\
t_2 &= \frac{|X|}{V} - \frac{V}{A} + \frac{v_0^2}{2VA} \\
t_3 &= V/A
\end{aligned}$$

The above calculations are performed for every joint every time new estimates of x_f and t_d become available. If one joint can not meet the deadline, the other joints are slowed down so they all reach the end point at the same time. This is an attempt to limit the deviation of the tool point from the plane where x_0 and all x_f are, and can also decrease the acceleration of some joints. The tool will not reach its target position before all joints reach their target position

Strategy 2: Use Maximum Acceleration

This strategy applies the maximum acceleration to move the joint to x_f as fast as possible. Hence the value of T is only used for determining whether the deadline could be met. The motion will be jerkier than with Strategy 1, but the deadlines may be met more often.

The expressions for a, t_1, t_2 and t_3 are the same as for the case with saturation of the acceleration in Strategy 1.

6.4 Experimental Results

Simulations Based on Real Data

Figures 6.1 - 6.4 show simulated robot trajectories based on real data generated by the tracker described in Chapter 5. The simulations assumed that the robot had two independent actuators that could move the tool in the x and y directions respectively. The velocity was limited to $V = 2$ [m/s] and the acceleration was limited to $A = 20$ [m/s^2], giving the robot properties similar to those of an IRB140.

Figures 6.1 and 6.2 are based on the same data, data set 1, but in Fig. 6.1 the trajectory is generated using method 1 and in Fig. 6.2 method 2 is used. Similarly, Figs. 6.3 and 6.4 are based on the same data, data set 2, but in Fig. 6.3 the trajectory is generated using method 1 and in Fig. 6.4 method 2 is used.

In the upper part of each figure the target points estimated by the tracker are marked with blue stars. The stars are connected by blue lines in chronological order and the final target point is marked in red. The simulated robot trajectory is shown as a green curve. The positions the robot was at when new target points arrived are marked with green stars.

The lower left part of each figure shows the position, velocity and acceleration in the x-direction as functions of time, and the lower right part shows the corresponding functions in the y-direction. The final target position and time is marked by a star.

Comparing Fig. 6.1 and Fig. 6.2 it can be seen that method 1 used smaller accelerations than method 2 in the beginning and hence avoided making a detour to the first bad estimates of the target position. Due to a late update of the target position method 1 was 0.4 [mm] away from the target at the deadline and reached the target 6 [ms] later. Method 2 was only 0.02 [mm] away from the target at the deadline and reached the target 1 [ms] later.

Comparing Fig. 6.3 and Fig. 6.4 it can again be seen that the trajectory generated by method 2 made a bigger detour to the first bad estimates of the target position. Both methods reached the target in time and method 2 even finished 7 [ms] before the deadline. Method 1, however, used less acceleration.

Real-Time Execution

The trajectory generation methods were implemented in Java and used in real-time to catch balls, as described in Chapter 3. The robot used was an IRB140, and the first three joints were used to move the hole of the box to the catching position. The choice was made to always try to catch the ball where it intersected a vertical plane in front of the robot. For each joint a trajectory was generated according to the following sequence every time a new target position was estimated by the tracker

- Go to the target position before the deadline;
- Wait 0.5 [s];
- Go to the home position in 1 [s].

Usually only the first part of the trajectory was executed when a new target estimate became available and a new trajectory was generated from the current position. Only the trajectory based on the last measurement of a throw was executed to the end. Since a safe return to the home position was generated every time a trajectory was generated, no special action had to be taken for the last estimate, and there was no risk for a trajectory to end while the robot was moving. The trajectory generation took approximately 0.1 [ms] on a desktop PC with 3 [GHz] processor.

Both methods of trajectory generation were tested. Method 1 generated significantly smoother motions and less noise from the robot than method 2. A video of the robot catching balls is available on Youtube [Ball-Catcher Video, 2009].

6.5 Discussion

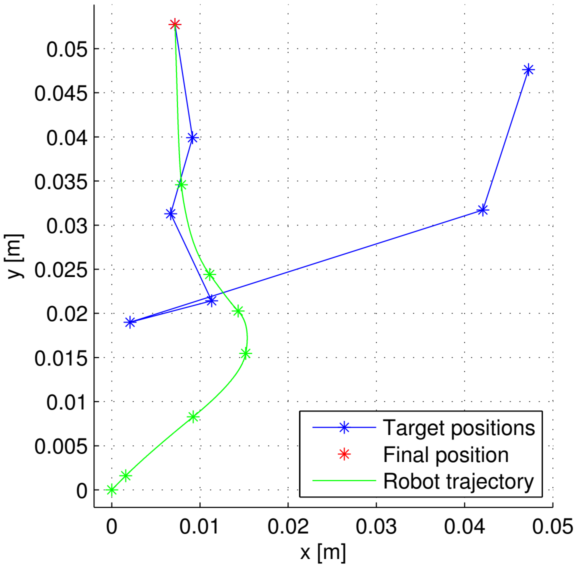
Alternative approaches for generating trajectories are, *e.g.*, dynamic programming [Bellman, 1954] or MPC (model predictive control) [Maciejowski, 2002]. They are, however, based on sampling of the trajectory and require much more computations than the suggested approach.

Related work on real-time trajectory generation can be found in, *e.g.*, [Lambrechts *et al.*, 2004; Bruyninckx, 2005; Macfarlane, 2001].

The suggested method has a constant computation time with respect to the length of the trajectory, and the generated trajectory can be represented as at most three concatenated second order polynomials. The trajectory will still have to be sampled before sending it to the robot controller, but this can be done just before the value is needed. There is no need to sample the entire trajectory if only the first part will be used before a modified trajectory is generated.

The closed form solutions presented on pages 69 – 70 can be computed very quickly. In the worst case scenario three of the cases have to be tested to find a valid solution.

For simplicity, it was chosen to aways catch the ball where it intersected a specified vertical plane. It would be possible to instead optimize along the trajectory of the ball, to find the position where the ball could be caught the most easily.



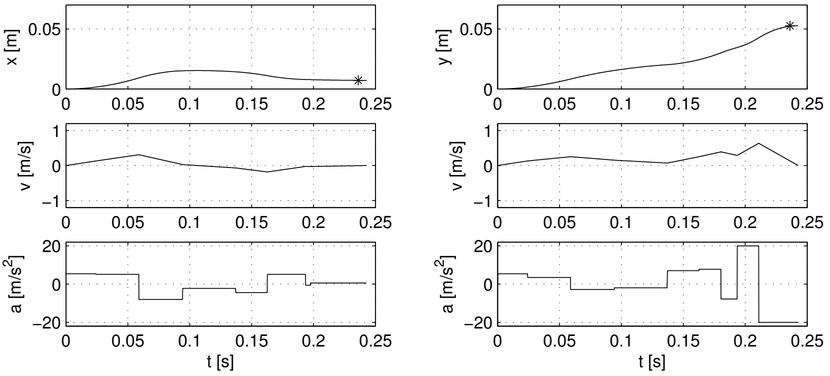
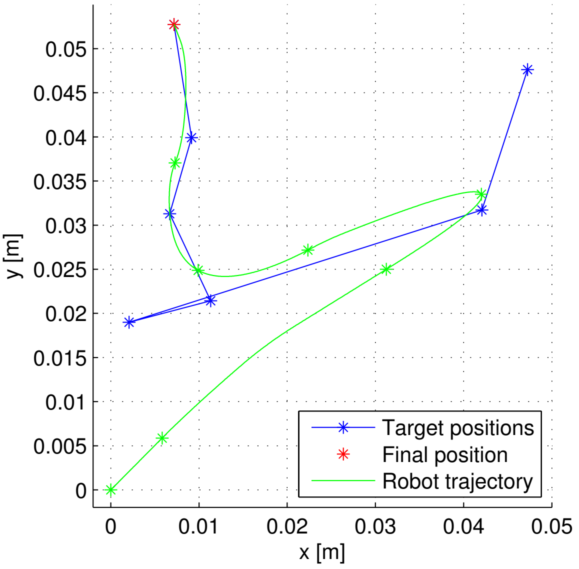


Figure 6.1 Generated trajectory based on data set 1, using trajectory generation method 1. (*Upper*) Target points and robot trajectory in the *xy*-plane. (*Lower left*) Time plots of position, velocity and acceleration in the *x*-direction. (*Lower right*) Time plots of position, velocity and acceleration in the *y*-direction.



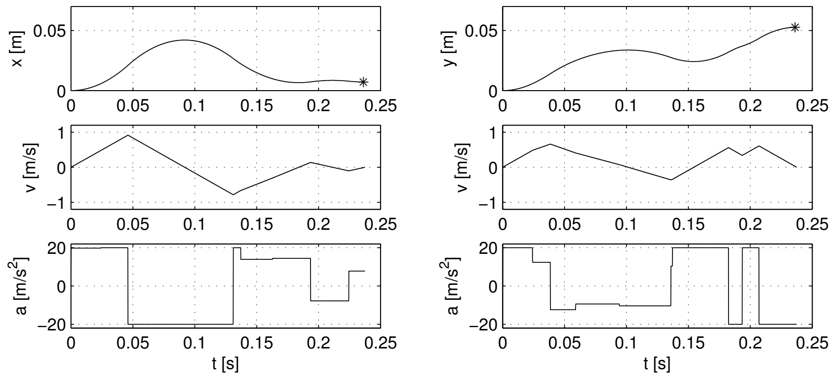


Figure 6.2 Generated trajectory based on data set 1, using trajectory generation method 2. (*Upper*) Target points and robot trajectory in the *xy*-plane. (*Lower left*) Time plots of position, velocity and acceleration in the *x*-direction. (*Lower right*) Time plots of position, velocity and acceleration in the *y*-direction.

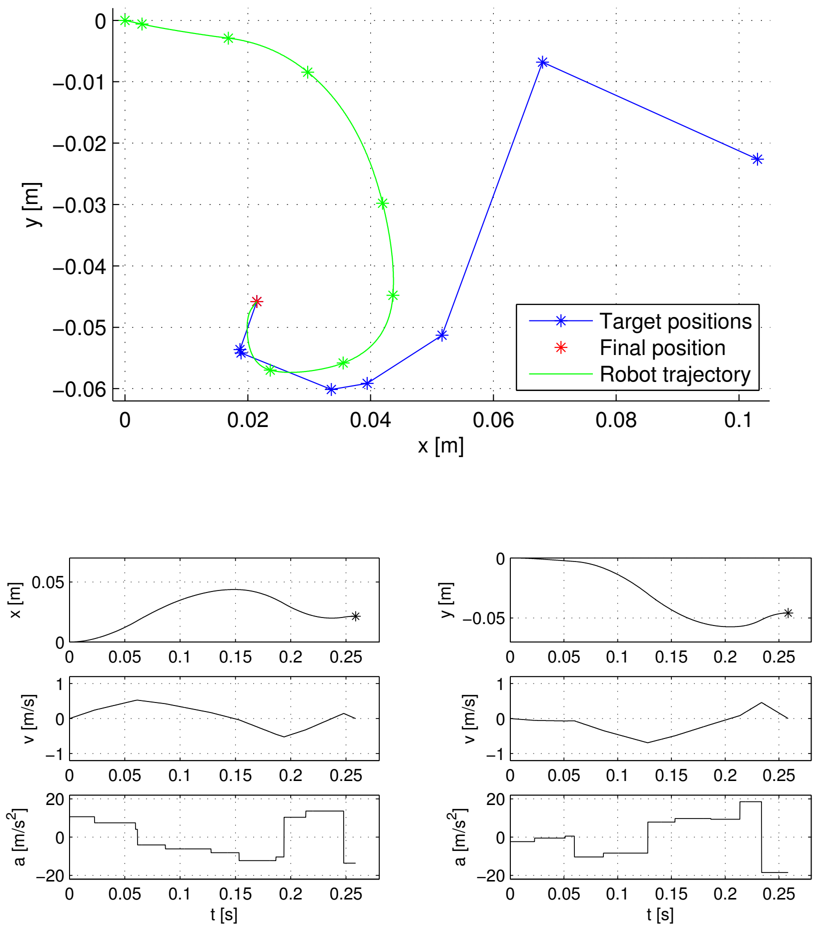


Figure 6.3 Generated trajectory based on data set 2, using trajectory generation method 1. (*Upper*) Target points and robot trajectory in the *xy*-plane. (*Lower left*) Time plots of position, velocity and acceleration in the *x*-direction. (*Lower right*) Time plots of position, velocity and acceleration in the *y*-direction.

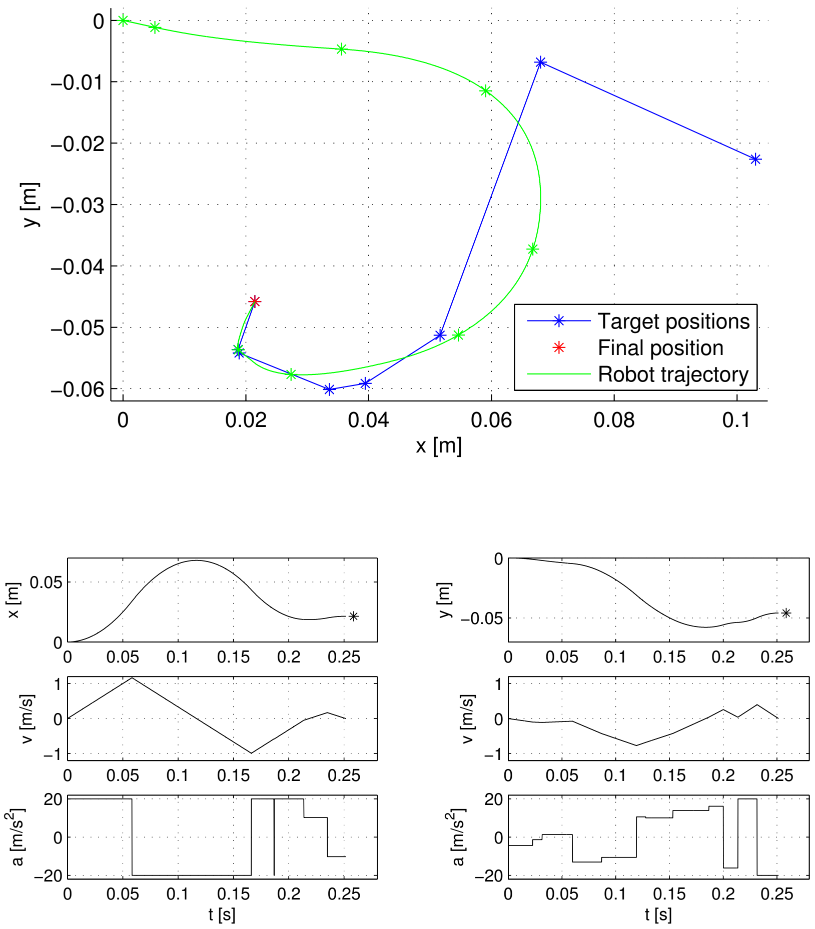


Figure 6.4 Generated trajectory based on data set 2, using trajectory generation method 2. (*Upper*) Target points and robot trajectory in the *xy*-plane. (*Lower left*) Time plots of position, velocity and acceleration in the *x*-direction. (*Lower right*) Time plots of position, velocity and acceleration in the *y*-direction.

Part II

Robotic Assembly

7

Introduction

Chapters 7–10 present work done on robotic assembly. Traditional position-based robot control is very successful in many applications. In assembly, however, the relative position of the objects to be assembled may require an accuracy that can not be accomplished by a position-controlled robot. Additional sources of uncertainty may result from difficulties in gripping the objects accurately.

If contacts can be detected by force sensors, many assembly operations can be performed even if the position accuracy is bad. The following chapters describe a framework that allows a programmer to define and execute force controlled assembly tasks in a convenient way.

The work described in Chapters 7–10 has been done in close cooperation with Andreas Stolt and we assert equal contribution, except for the work on learning in Sec. 9.3, which was done by me.

A previous approach for performing robotic force control was described in [Khatib, 1987]. In [Bruyninckx *et al.*, 2001] a survey of the requirements for autonomous robotic assembly was presented. In [Olsson *et al.*, 2005] force sensing and vision were used to handle contact transitions. Additional examples of force control were given in [Arai *et al.*, 2006], [Di Xiao *et al.*, 2000] and [Jörg *et al.*, 2000].

8

Control Framework

8.1 Task Specification

For the specification of the assembly operations the constraint-based task specification methodology [De Schutter *et al.*, 2007] was used. An overview of the parts relevant for the following chapters will be given here.

The constraint-based task specification methodology provides a convenient way of describing robot motions without being restricted to joint space or Cartesian space. One or several feature chains are defined, relating the tool frame of the robot to the world frame. Each feature chain has six DOFs (degrees of freedom) χ_f, which typically are chosen to represent relevant quantities for the task.

The transformation from the world frame to the tool frame can through the robot kinematics be described as a function of the joint positions

$$T_q(q) \tag{8.1}$$

The pose of the tool frame can also be described by a feature chain, which typically consists of a series of translations and reorientations. The transformation from the world frame to the tool frame can be described as a function of the feature coordinates

$$T_f(\chi_f) \tag{8.2}$$

The geometric Jacobians of (8.1) and (8.2) are $J_q(q)$ and $J_f(\chi_f)$, respectively. The condition that (8.1) and (8.2) describe the same tool frame pose gives the equations

$$T_q(q) = T_f(\chi_f) \tag{8.3}$$

$$\begin{aligned} J_q\dot{q} &= J_f\dot{\chi}_f \Longleftrightarrow \\ \dot{\chi}_f &= J_f^{-1}J_q\dot{q} \end{aligned} \tag{8.4}$$

The outputs y are defined by

$$f(q,\chi_f) = y \tag{8.5}$$

$$C_q = \frac{\partial f}{\partial q} \tag{8.6}$$

$$C_f = \frac{\partial f}{\partial \chi_f} \tag{8.7}$$

Combining equations (8.4)-(8.7) gives

$$\begin{aligned} \dot{y} &= C_q\dot{q} + C_f\dot{\chi}_f \\ &= A\dot{q} \end{aligned} \tag{8.8}$$

$$A = C_q + C_fJ_f^{-1}J_q \tag{8.9}$$

Object and Feature Frames

For convenience, object and feature frames can be introduced to facilitate the specification of the feature chain. The relations between the frames are depicted in Fig. 8.1 and an example is shown in Fig. 9.2. Below is a description of the frames and guidelines for how to choose them.

- $o1$ – object frame 1. Usually attached to the object to manipulate;
- $f1$ – feature frame 1. Usually attached to a feature on the object to manipulate;
- $f2$ – feature frame 2. Usually attached to a feature on the robot;
- $o2$ – object frame 2. Usually attached to the robot.

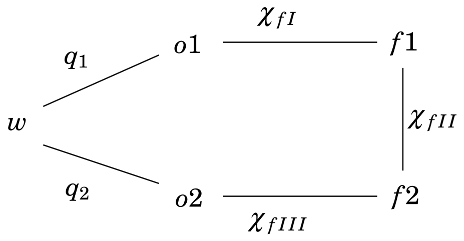


Figure 8.1 Illustration of frame interconnections.

The feature coordinate vector χ_f can be partitioned into three parts according to Fig. 8.1 with $\chi_f = ({\chi_{fI}}^T \;\; {\chi_{fII}}^T \;\; {\chi_{fIII}}^T)^T$. The six DOFs of χ_f can be distributed between χ_{fI}, χ_{fII} and χ_{fIII} as appropriate for the task at hand. In this thesis q_1 will be constant unless otherwise stated.

Control

The robot motion is described in terms of the outputs on the velocity level:

$$\dot{y} = \underbrace{\dot{y}_d + C}_{\dot{y}_d^\circ} \tag{8.10}$$

where $\dot{y}_d$ is feed-forward for the desired trajectory and C is a controller using feed-back from, *e.g.*, force sensors, cameras or measured robot joint angles. The desired robot joint velocities can then be obtained as the solution to

$$\dot{y}_d^\circ = A\dot{q}_d \tag{8.11}$$

Unless y and q have the same dimensions and A has full rank, some kind of pseudoinverse of A has to be used to solve (8.11). For under-constrained systems a possibility is

$$\dot{q}_d = A^\# \dot{y}_d^\circ, \qquad A^\# = M^{-1}A^T \left(AM^{-1}A^T\right)^{-1} \tag{8.12}$$

which is the solution to the optimization problem

$$\begin{aligned} &\min_{\dot{q}_d} \dot{q}_d^T M \dot{q}_d \\ &\text{s.t.}\, \dot{y}_d^{\circ} = A\dot{q}_d \end{aligned} \tag{8.13}$$

The matrix M is a weighting matrix that is used to choose the optimization criterion. If the desired weighting is more easily described in Cartesian space, then this property can easily be handled by using $M = J_q^T(q)\bar{M}J_q(q)$, where $\bar{M}$ describes the weights in the Cartesian space.

The Extctrl interface [Blomdell *et al.*, 2010] to the robot allows sending of joint angle references and feed-forward values for velocities and torques. The $\dot{q}_d$ from (8.12) is thus integrated to obtain the robot joint angle references.

State Estimation

The pose of the tool frame can be obtained from the robot joint angles q through (8.1). The corresponding feature coordinates χ_f that solve (8.3) can be obtained iteratively. Assume that there is an initial guess $\hat{\chi}_f^0$ of χ_f. If $\hat{\chi}_f^i \neq \chi_f$ this results in an error T_d^i satisfying

$$T_q(q) = T_d^i T_f(\hat{\chi}_f^i) \tag{8.14}$$

$$T_d^i = \begin{bmatrix} R^i & t^i \\ 0 & 1 \end{bmatrix} \tag{8.15}$$

where R^i is a rotation matrix which can be converted to a rotation vector v^i in axis/angle representation [Spong *et al.*, 2006]. The feature coordinates χ_f can then be obtained iteratively by

$$\hat{\chi}_f^{i+1} = \hat{\chi}_f^i + J_f^{-1}(\hat{\chi}_f^i)\begin{bmatrix} t^i \\ v^i \end{bmatrix} \tag{8.16}$$

If the system were linear, a single iteration would be needed to arrive at the correct solution. If χ_f is estimated continuously, then the estimate from the previous time step can be used as an initial guess, and one or a few iterations are usually sufficient for convergence.

8.2 Control Strategies

Three different control strategies were used in (8.10). Independent scalar controllers were applied to the different components of y.

Position Control

Position control in the operational space was implemented as PI controllers. Each position-controlled component of the output y was governed by

$$\dot{y}_d^\circ = \underbrace{\dot{y}_d}_{0} + K\left((y - y_{ref}) + \frac{1}{T_i}\int (y - y_{ref})dt\right) \tag{8.17}$$

where y_{ref} is the reference output value and K and T_i are controller parameters. The output of the controller is hence the time derivative of the desired output y_d.

Velocity Control

Velocity control was implemented like (8.17), but with non-zero $\dot{y}_d$ and updating y_{ref} according to $\dot{y}_{ref} = \dot{y}_d$.

Impedance Control

Impedance control [Hogan, 1985] was implemented to make the robot behave as if it were a body with mass M, a viscous damping D and a force F_{ref} acting on it. This was realized by

$$\ddot{y}_d^\circ = \frac{1}{M}\left(F - F_{ref} - D\dot{y}_d^\circ\right) \tag{8.18}$$

where F is the measured force acting on the robot, transformed through the feature coordinates to the outputs.

8.3 Software

A framework for specifying and executing tasks described by the constraint-based task specification methodology was implemented in Matlab/Simulink. The programs were compiled by the Real-Time Workshop toolbox [Real-Time Workshop, 2011] and executed on a real-time

Linux PC [Xenomai, 2011], which connected to the robot controllers via the Extctrl interface [Blomdell *et al.*, 2010].

The program was structured to strictly separate task independent parts from task specific parts. The task independent parts included initialization, controllers, coordinate transformations, kinematics calculations, gravity compensation and overload protection.

All task specific parts were to be implemented in a state machine. State machines used were implemented in, *e.g.*, Stateflow [Stateflow, 2011], JGrafchart [JGrafchart, 2011] or Java [Java, 2011]. The output from the state machine to the task independent parts of the software consisted of the following time-variable parameters:

- feature chains to be used, described as sequences of translations and reorientations;
- active outputs;
- reference values for the outputs;
- controller types (position, velocity or impedance);
- controller parameters.

The state machines made state transitions based on, *e.g.*, robot joint positions or force measurements and varied the parameters to the task independent parts through the execution of the task.

9

Snap Fit Assembly of Emergency Stop Switch

9.1 Introduction

Generally, assembly without well-defined fixtures is difficult without support from sensors. This chapter illustrates a successful approach to solve this problem, using feedback from a force/torque sensor. Figure 9.1 shows a partly assembled emergency stop button that was used as an example application of force controlled assembly. This chapter will describe how to attach the dark gray switch to the light gray box using a snap fit.

9.2 Task Description

The constraint-based task specification methodology (Sec. 8.1) was used to describe the assembly operation. The object and feature frames, illustrated in Fig. 9.2, were chosen as follows:

- $o1$ was attached to the box and was related to the world frame by a fix transformation;
- $f1$ was attached to a point on the switch and had the same ori-

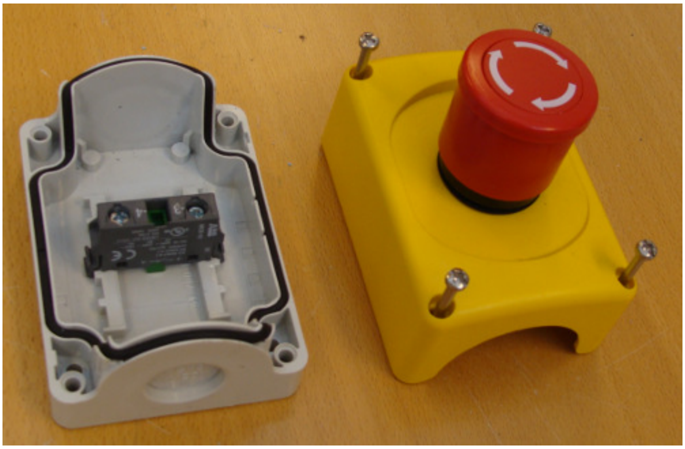


Figure 9.1 Emergency stop button that is to be assembled.

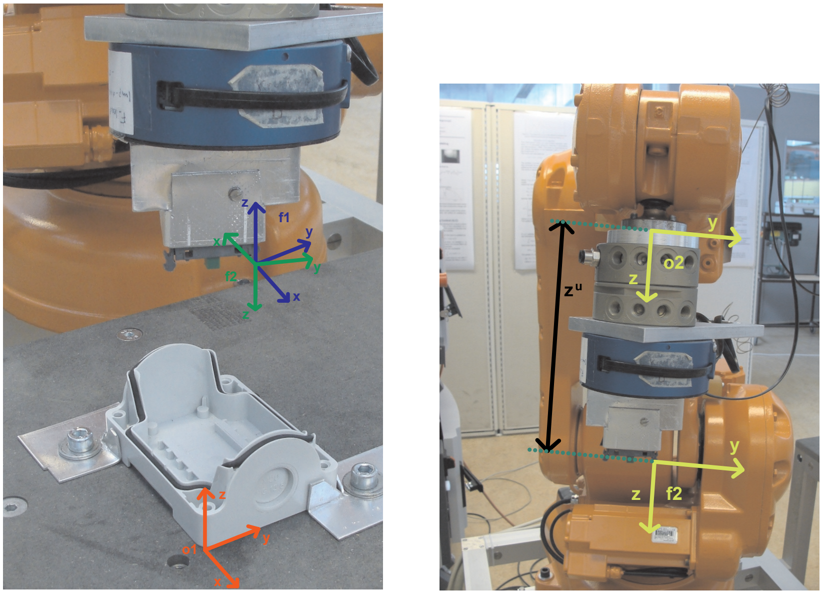


Figure 9.2 Illustration of object and feature frames.

entation as $o1$;

- $f2$ had its origin coinciding with the origin of $f1$, but the orientation was the same as for the robot flange frame;
- $o2$ coincided with the robot flange frame.

The feature coordinates were defined as:

- $\chi_{fI} = (x \quad y \quad z)^T$ described the translation from $o1$ to $f1$ with Cartesian coordinates in $o1$;
- $\chi_{fII} = (\varphi \quad \theta \quad \psi)^T$ described the reorientation from $f1$ to $f2$ in ZYX Euler angles;
- χ_{fIII} was empty since the transformation between $f2$ and $o2$ was fix.

The process outputs were chosen to be the feature coordinates:

$$y = \chi_f = (x \quad y \quad z \quad \varphi \quad \theta \quad \psi)^T \tag{9.1}$$

It was assumed that the position of the box, onto which the switch should be snapped, was only known with an accuracy of a few mm. Under these circumstances position controlled assembly was not possible. Instead, a wrist-mounted force/torque sensor was used to search for contact with the box and hence resolve the uncertainties.

The state machine describing the assembly sequence can be seen in Fig. 9.3 and snapshots from the execution are shown in Fig. 9.4. In order to perform the assembly, the different controller strategies described in Sec. 8.2 were applied to different components of the output y at different stages of the execution.

It was assumed that the position and orientation of the box was known well enough for the switch to hit the bottom of the box in front of the slot where the switch should be mounted. First, all outputs y were position controlled to position the switch over the box. Then a velocity controller on z was used to lower the switch toward the box. When a big contact force was detected in the z-direction, the transition to the next state was triggered and an impedance controller was started. This was used during the rest of the execution to maintain contact in the z-direction. Similarly, the other contact surfaces were found with guarded search motions and then maintained with impedance

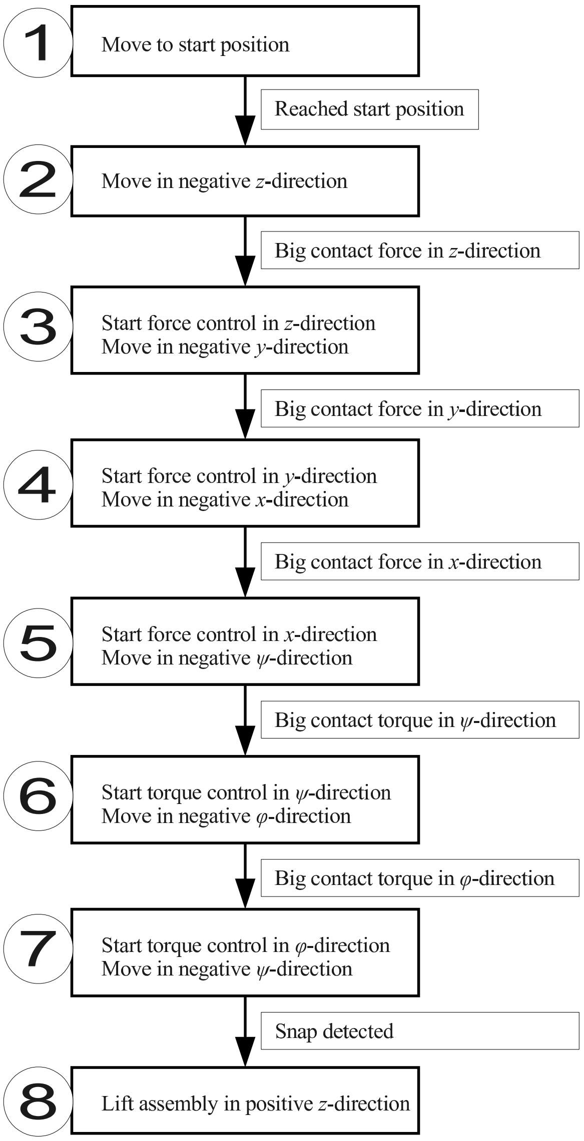

1
Move to start position
Reached start position
2
Move in negative z-direction
Big contact force in z-direction
3
Start force control in z-direction
Move in negative y-direction
Big contact force in y-direction
4
Start force control in y-direction
Move in negative x-direction
Big contact force in x-direction
5
Start force control in x-direction
Move in negative ψ-direction
Big contact torque in ψ-direction
6
Start torque control in ψ-direction
Move in negative φ-direction
Big contact torque in φ-direction
7
Start torque control in φ-direction
Move in negative ψ-direction
Snap detected
8
Lift assembly in positive z-direction

Figure 9.3 State machine describing the assembly sequence.

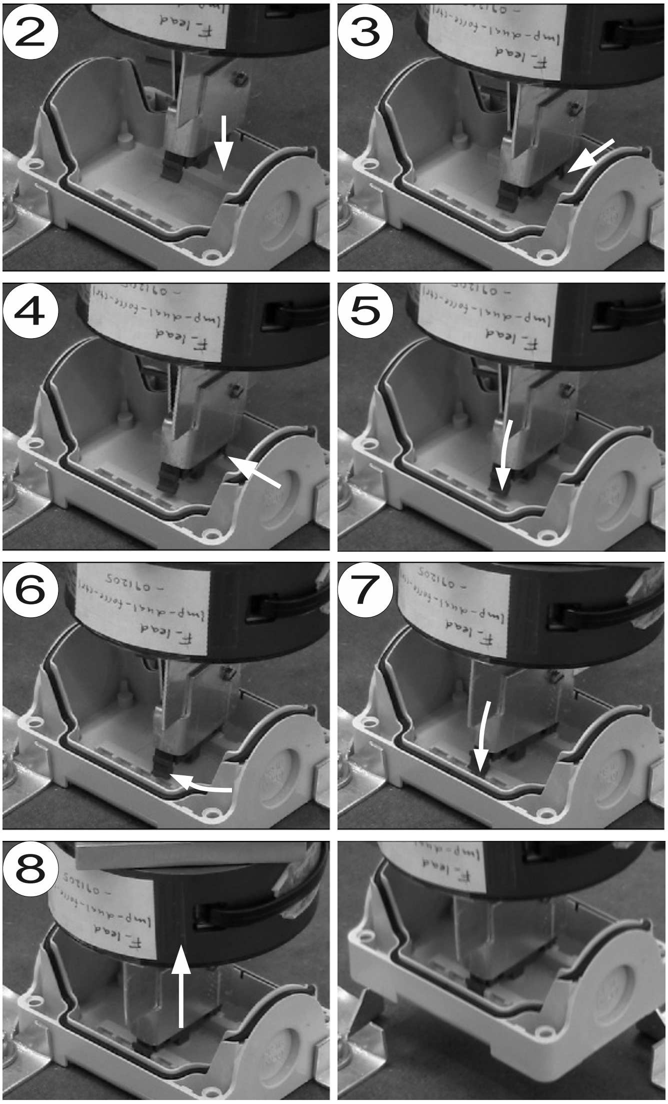


Figure 9.4 Snapshots from the execution of the assembly with the state numbers indicated in each image. The arrows indicate the direction of movement.

controllers. Finally the snap was detected and the assembled parts lifted from the fixture.

9.3 Learning

When searching for contact between the robot and another object, there is a delay between the moment when contact is made and when the robot actually stops. If the materials are stiff, the contact forces increase very rapidly during this delay and components may break. Hence, the robot has to move quite slowly when searching for contact.

A learning strategy was applied to the assembly operation to reduce the execution time. In the initial implementation of the assembly operation the parameters (initial position, velocities etc.) were set manually. Then, the assembly operation was performed 13 times and for each search operation the value of the corresponding feature coordinate was recorded when contact was made. The empirical distribution of the contact positions was used to determine a region where contact was expected to be made. The velocity of the robot could then be increased in the initial part of each search operation and when it reached the region where contact was expected to be made, the velocity was reduced to the same value as before the learning strategy was applied

Experimental Results

Figure 9.5 illustrates the performance improvement gained by the learning strategy. The figure shows the states described in Fig. 9.3 as a function of time, with and without learning. As can be seen, the total execution time was reduced from 8.6 [s] to 3.5 [s]. This simple learning strategy shows that big performance improvements can be achieved by using learning and feed-forward data. A video of the assembly being performed with and without learning is available through [Stolt *et al.*, 2011].

9.4 Transient Detection

The condition for transitioning from state 7 to state 8 in Fig. 9.3 was that the switch snapped in place. When the snap occurred, the signals

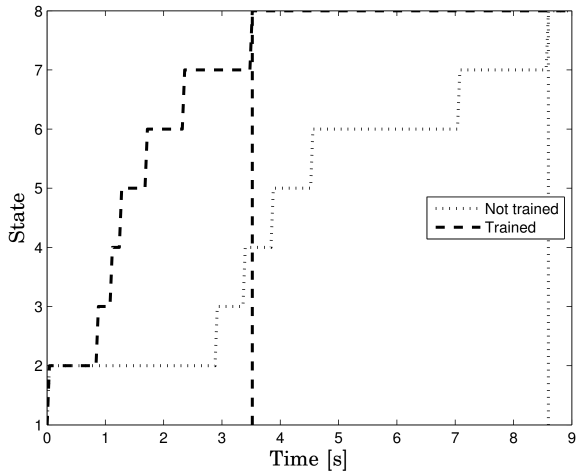


Figure 9.5 Time evolution of the state sequence from the assembly task with and without training. The vertical lines indicate when the assembly operations finish.

of the z-force and the ψ-torque followed a specific signature. An automatic snap detector was developed to determine whether a sequence of measurements were caused by a snap or not.

An assumption was made that the sequence of force and torque measurements could be considered as an n-dimensional multivariate Gaussian random variable, where n is the number of measurements used for the classification. With mean value μ and covariance matrix R the probability distribution function is

$$f_{\mu,R}(x) = \frac{1}{(2\pi)^{n/2}|R|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T R^{-1}(x-\mu)\right) \tag{9.2}$$

As a classification criterion to determine whether the measurement sequence x was caused by a snap, the hyper-ellipsoid defined by

$$h(x) < c^2 \tag{9.3}$$

was used, where

$$h(x) = (x - \mu)^T R^{-1}(x - \mu) \tag{9.4}$$

Using this criterion was motivated by the fact that all points inside the hyper-ellipsoid have higher probability density than all points outside the hyper-ellipsoid, which can easily be verified by combining (9.2) and (9.4).

The probability of a snap being detected as a snap is given by (9.5) for even n and (9.6) for odd n.

$$\begin{aligned} p &= \int_{(x-\mu)^T R^{-1}(x-\mu)<c^2} f_{\mu,R}(x)dx \\ &= 1 - \exp\left(-\frac{c^2}{2}\right) \sum_{i=0}^{n/2-1} \frac{(c^2/2)^i}{i!} \end{aligned} \tag{9.5}$$

$$\begin{aligned} p &= \int_{(x-\mu)^T R^{-1}(x-\mu)<c^2} f_{\mu,R}(x)dx \\ &= \operatorname{erf}\left(\frac{c}{\sqrt{2}}\right) - \sqrt{\frac{2}{\pi}} \exp\left(-\frac{c^2}{2}\right) \sum_{i=1}^{(n-1)/2} \frac{c^{2i-1}}{\prod_{j=1}^{i}(2j-1)} \end{aligned} \tag{9.6}$$

The formulas (9.5) and (9.6) look quite complicated, but are straight forward to evaluate on a computer, and a value of c giving an appropriate level of confidence could be found off-line. During execution μ, R and c were kept constant and at each sampling instant the latest measurement sequence x was applied to (9.3) to determine whether a snap occurred.

Experimental Results

Two specimens of the switch were available for training of the snap detector. Data from 10 snaps were recorded for each switch. The sampling instant when the snap occurred was manually marked for each data sequence. The 9 samples centered around the snap were then selected and a weighted mean value subtracted. The data from the

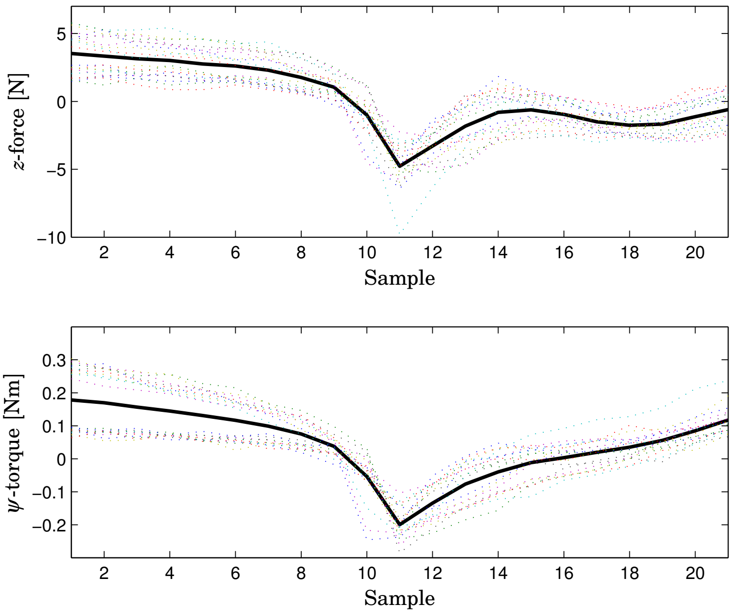


Figure 9.6 The data used for training of the snap detection. The dotted curves show the results for the 20 different experiments and the thick curve shows the point-wise mean. Only samples 7-15 were used for the classification. In the plots the data was aligned so the snap occurred at sample 11, and the offset was compensated for by subtracting a weighted mean of samples 7-15.

z-force and ψ-torque were then considered as the outcome of an 18-dimensional random variable, that would be used for detection. The empirical mean value $\mu \in \mathbb{R}^{18}$ and covariance matrix $R \in \mathbb{R}^{18\times 18}$ for the 20 training snaps were calculated. The training data are shown in Fig. 9.6.

When the detector was used, the 9 latest samples of the z-force and ψ-torque were used to evaluate the snap detection criterion (9.3) at every sampling instant. In Fig. 9.7 the snap detection is evaluated by applying the method to 6 snaps that were not used in the training;

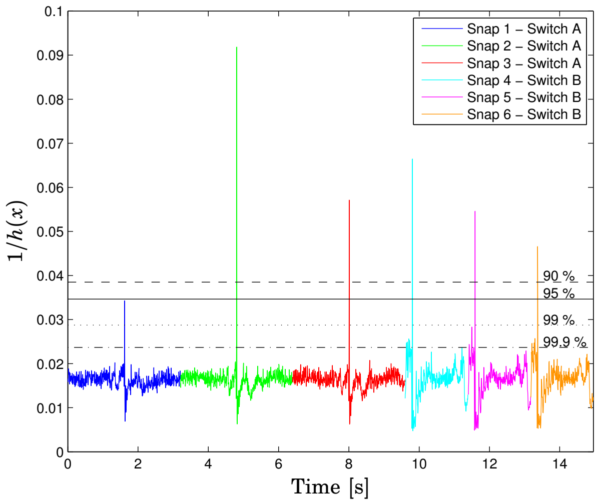


Figure 9.7 Results from the snap detection. In the plot data from 6 snaps, each with a different color, are concatenated and $h(x)$ in (9.4) is evaluated. The horizontal lines show the decision boundaries for different confidence levels. For 99 % of the snaps, the peak will be higher than the line marked with 99 % and so on.

3 snaps from each specimen of the switch. The six sequences were put side by side with $1/h(x)$ plotted. The snaps can be seen as very distinct peaks.

In the lower part of Fig. 9.6, it can seen that the two different switches had different signatures, which probably was a consequence of one of the switches being worn out, since it had been snapped in place many times before. The different signatures may have affected the classification performance. In a real application scenario, however, a new switch would be used every time, and if a new switch would be used for every training snap too, the training data would probably be more realistic.

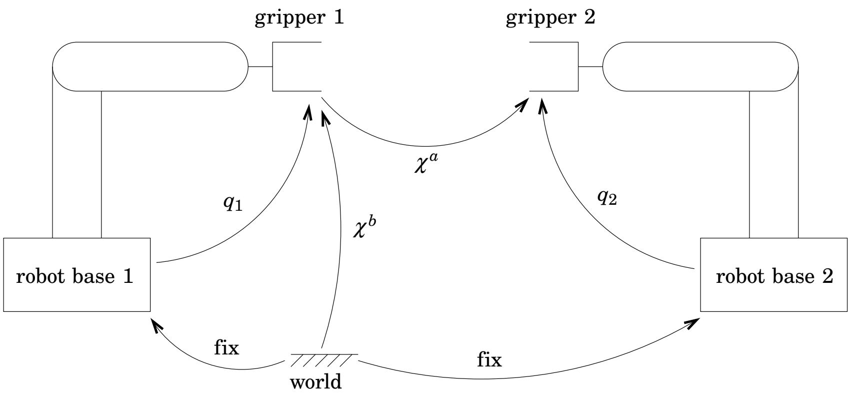


Figure 9.8 Schematic overview of cooperating robots performing snap fit assembly. The robot joint angles for the different robots are denoted by q_1 and q_2, and the feature coordinates for the two feature chains are denoted by χ^a and χ^b.

9.5 Cooperating Robots

In the assembly operations described in the previous sections of this chapter, the switch was held by a robot and the bottom box was mounted in a fixture. The same assembly was also performed with the bottom box held by a second robot. The setup is schematically described in Fig. 9.8. The box was held by gripper 1 and the switch by gripper 2. The transformation between the box and the switch, described by feature coordinates χ^a, was identical to the case with the box in a fixture. Hence the state machine describing the assembly could be reused without modifications. The remaining feature coordinates, χ^b, could be kept constant to keep gripper 1 fixed, or be used to superimpose a motion of the parts without affecting the relative position of the grippers. Figure 9.9 shows a photo of an IRB140 and an IRB2400 performing assembly together.

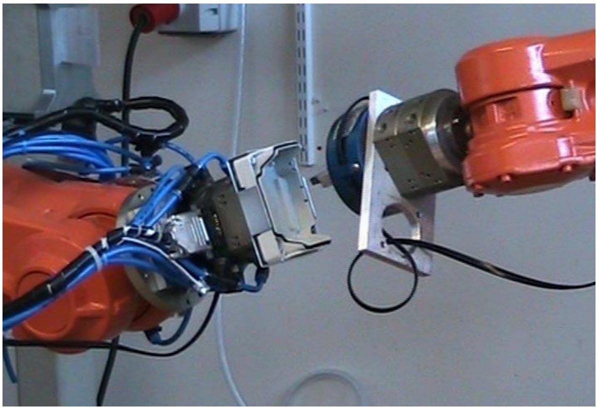


Figure 9.9 Photo of cooperating robots performing snap fit assembly.

10 Dual Robot Lead-Through

10.1 Introduction

This chapter describes the implementation of a lead-through system for two robots, shown in Fig. 10.1. Each robot was equipped with a handle mounted on a 6-DOF force/torque sensor, which in turn was mounted on the flange of the robot. The robots were constrained to move so they always had identical positions relative to their respective initial positions. The simultaneous motion of the robots was governed by an impedance controller, giving the robots a virtual mass, moment

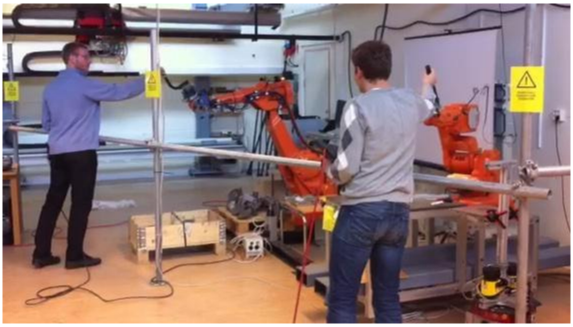


Figure 10.1 Lead-through of virtually connected robots.

of inertia and damping.

Forces applied to the handle of one robot, were felt by a hand holding the handle of the other robot. A system like this can hence be used as a haptic device for teleoperation. By applying different gains on the two robots it can be used to, *e.g.*, increase the strength of the user.

10.2 System Structure

The forward kinematics will in this chapter be computed for the tool frame, which has the same orientation as the flange frame and has its origin where the hand grips the handle. All robot Jacobians used will be described in the tool frame. The subscript i is used to denote the two different robots, and the subscript j is used to denote one of the six components in a force/torque vector or a translational and rotational velocity vector.

The initial positions of the robots are denoted by

$$T_{0i} \tag{10.1}$$

where $i = 1$ for one of the robots and $i = 2$ for the other. Since both robots have the same motion relative to their initial positions, the output of the controller can easily be described in a robot independent way. The reference position T_i is described by the displacement T_d for the respective robots:

$$T_i(q_i) = T_{0i}T_d \tag{10.2}$$

which through the inverse kinematics gives the reference positions for the robot joints q_i. The impedance controller outputs the desired translational and rotational velocity v of the tool frame described in the tool frame, fulfilling

$$v = J_i(q_i)\dot{q}_i \tag{10.3}$$

where J_i is the Jacobian of robot i, and v has the structure

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} \dot{t} \\ \omega \end{bmatrix} \tag{10.4}$$

where $\dot{t}$ is the translational velocity and ω is the rotational velocity. The desired robot joint velocities can be calculated as

$$\dot{q}_i = J_i^{-1}(q_i)v \tag{10.5}$$

The controller is implemented as six independent scalar impedance controllers according to

$$\dot{v}_j = \frac{1}{M_j}(F_j - D_j v_j) \tag{10.6}$$

where v_j is one of the six components of v, F_j is the applied force/torque in the corresponding direction, D_j is the virtual damping and M_j is the virtual mass/moment of inertia. The output of the controller, to be used in (10.5), is then obtained by integrating the left hand side of (10.6); $v_j = \int \dot{v}_j dt$.

Since the output of the controller is a velocity and the interface to the robots requires a reference position, the controller output has to be integrated somehow. This is done in the joint space of robot 1:

$$q_1 = \int J_1^{-1}(q_1)v\,dt \tag{10.7}$$

From the forward kinematics $T_1(q_1)$ the displacement can then be calculated as

$$T_d = T_{01}^{-1}T_1(q_1) \tag{10.8}$$

The force vector F, whose components are used in (10.6), is calculated as a combination of the forces F_{si} measured by the force/torque sensors and virtual torques τ_{qi} applied to avoid joint limits:

$$F = k_1 F_{s1} + k_2 F_{s2} + J_1^{-T}(q_1)\tau_{q1} + J_2^{-T}(q_2)\tau_{q2} \tag{10.9}$$

where k_1 and k_2 are gains. Since the transformations from the sensor frames to the respective tool frames are fix and the control is done in the tool frame, the sensor measurements can be transformed to F_{s1} and F_{s2} using fix transformations. The gains k_1 and k_2 can be used to scale how forces applied to one robot are perceived at the other robot.

When a joint limit was close to being violated, a virtual linear spring applied a torque to the corresponding joint in τ_{qi}, and the torque was transformed to the tool frame by means of the robot Jacobian as described in (10.9). By setting the joint limits properly this could also be used to avoid kinematic singularities.

For safety reasons the robots were in teach mode when doing the lead-through, which meant that the robot was to do an emergency stop if any of the motors exceeded a threshold speed. These speed limits were easily exceeded, and to avoid the robot doing emergency stops all the time, a speed limitation was added to the controller on the motor level.

To avoid the robots drifting away, due to noise and offsets of the force measurements, the applied forces to the handles had to exceed some threshold values before the robots started moving.

10.3 Experimental Results

The system was successfully implemented, using an ABB IRB140 and an ABB IRB2400, which proves the ability of the system to handle robots with different kinematics. If a person grabbed the handle of one of the robots, the robot could be moved with a perceived virtual mass, moment of inertia and damping. Both robots would make identical motions and if the joint limits of any of the robots were close to being violated, the force of virtual springs were felt in the handle. If a robot made contact with an object, the contact force was perceived by the person holding the handle of the other robot. In one experiment a spatula was attached to one of the robots and used to cut a birthday cake.

11

Conclusions

A new way of initializing the Kalman filter has been presented, making it possible to calculate a state estimate that is not influenced by any guess of the initial value of the state. Instead the estimate can be determined completely based on the first measurements. This is accomplished by choosing a new state space where the directions with infinite variance are orthogonal to as many basis vectors as possible.

A strategy for tracking dynamical objects with computer vision was described. It provides a simple way to fuse data from an arbitrary number of pictures captured simultaneously. In combination with a dynamical model of the tracked object, it enables determination of the position of the object in 3D space without any two images being captured at the same time. It also allows tracking of objects in 3D space, using only a single static camera.

A method for generating trajectories with uncertain target points was described. It pays attention to the deadline and uses the smallest acceleration possible to reach the goal in time. When a new target point is received, the trajectory is smoothly modified on the fly.

A method for performing force controlled assembly was described and used to perform snap fit assembly of an emergency stop switch. Learning was used to reduce the execution time.

A

Bibliography

ABB Robots (2011): "`http://www.abb.com/product/us/9AAC100735.aspx`."

Arai, T., N. Yamanobe, Y. Maeda, H. Fujii, T. Kato, and T. Sato (2006): "Increasing Efficiency of Force-Controlled Robotic Assembly::-Design of Damping Control Parameters Considering Cycle Time." *CIRP Annals-Manufacturing Technology*, **55:1**, pp. 7–10.

Ball-Catcher Video (2009): "`http://www.youtube.com/watch?v=Fxzh3pFr3Gs`."

Basler (2011): "`http://www.baslerweb.com/beitraege/unterbeitrag_en_23042.html`."

Bäuml, B., F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, O. Eiberger, and G. Hirzinger (2011): "Catching flying balls and preparing coffee: Mobile humanoid Rollin' Justin perfoms dynamic and sensitive tasks." In *Proc. IEEE Intl. Conf. Robotics and Automation (ICRA2011), May 9-13, Shanghai, China*, pp. 3443–3444.

Bayes, T. (1763): "An essay towards solving a problem in the doctrine of chances." *Phil. Trans. of the Royal Soc. of London*, **53**, pp. 370–418.

Bellman, R. (1954): "The theory of dynamic programming." *Bulletin of the American Mathematical Society 60*, pp. 503–516.

Birbach, O. and U. Frese (2009): "A multiple hypothesis approach for a ball tracking system." In Fritz *et al.*, Eds., *ICVS*, vol. 5815 of *Lecture Notes in Computer Science*, pp. 435–444. Springer.

Birbach, O., U. Frese, and B. Bäuml (2011): “Realtime perception for catching a flying ball with a mobile humanoid.” In *Proc. IEEE Intl. Conf. Robotics and Automation (ICRA2011), May 9-13, Shanghai, China*, pp. 5955–5962. IEEE.

Birbach, O., J. Kurlbaum, T. Laue, and U. Frese (2008): “Tracking of ball trajectories with a free moving camera-inertial sensor.” In Iocchi *et al.*, Eds., *RoboCup*, vol. 5399 of *Lecture Notes in Computer Science*, pp. 49–60. Springer.

Blomdell, A., G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. Wang (2005): “Extending an industrial robot controller–Implementation and applications of a fast open sensor interface.” *IEEE Robotics & Automation Magazine*, **12:3**, pp. 85–94.

Blomdell, A., I. Dressler, K. Nilsson, A. Robertsson, and I. Dressler (2010): “Flexible application development and high-performance motion control based on external sensing and reconfiguration of abb industrial robot controllers.” In *Proc. ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*. Anchorage, AK.

Bruyninckx, H. (2005): “`http://people.mech.kuleuven.be/~bruyninc/blender/doc/interpolation-ap%i.html`.”

Bruyninckx, H., T. Lefebvre, L. Mihaylova, E. Staffetti, J. De Schutter, and J. Xiao (2001): “A roadmap for autonomous robotic assembly.” In *Proc. Intl. Symp. Assembly and Task Planning*. Fukuoka, Japan.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007): “Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty.” *Intl. J. Robotics Research*, **26**, pp. 433–455.

Di Xiao, B., N. Xi, and T. Tarn (2000): “Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment.” *IEEE Trans. Control Systems Technology*, **8:4**, p. 635.

Einhorn, E., C. Schröter, H.-J. Böhme, and H.-M. Gross (2007): “A hybrid Kalman filter based algorithm for real-time visual obstacle detection." *Proc. 52nd Intl. Scientific Colloquium (IWK)*, **II**, pp. 353–358.

Farooq, M. and S. Bruder (1990): "Information type filters for tracking a maneuvering target." *Trans. Aerospace and Electronic Systems*, **26:3**, pp. 441–454.

Frese, U., B. Baeuml, G. Schreiber, I. Schaefer, M. Haehnle, G. Hirzinger, and S. Haidacher (2001): "Off-the-shelf vision for a robotic ball catcher." In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS2001), October 2001, Maui*, pp. 1623–1629.

Graustein, W. C. (1930): *Homogeneous Cartesian Coordinates. Linear Dependence of Points and Lines. Ch. 3 in Introduction to Higher Geometry*. Macmillan, New York.

Hagander, P. (1973): *Operator Factorization and Other Aspects of the Analysis of Linear Systems*. PhD thesis TFRT-1005, Department of Automatic Control, Lund University, Sweden.

Hogan, N. (1985): "Impedance control: An approach to manipulation." *ASME J. Dynamic Systems, Measurement, and Control*, **107**, pp. 1–24.

Hyland, J. (2002): "An iterated-extended Kalman filter algorithm for tracking surface and sub-surface targets." *OCEANS '02 MTS/IEEE*, **3**, pp. 1283–1290.

Java (2011): "`http://www.oracle.com/technetwork/java/index.html`."

JGrafchart (2011): "`http://www.control.lth.se/Research/tools/grafchart.html`."

Jia, Z., A. P. Balasuriya, and S. Challa (2005): "Sensor fusion based 3D target visual tracking for autonomous vehicles with IMM." In *Proc. IEEE Intl. Conf. Robotics and Automation (ICRA 2005), April 18-22, 2005, Barcelona, Spain*, pp. 1829–1834.

Jörg, S., J. Langwald, C. Natale, J. Stelter, and G. Hirzinger (2000): "Flexible robot-assembly using a multi-sensory approach." In *IEEE Intl. Conf. Robotics and Automation (ICRA2000)*, vol. 4, pp. 3687–3694. San Francisco.

JR3 (2011): “`http://www.jr3.com`.”

Kailath, T., A. Sayed, and B. Hassibi (2000): “Linear estimation.” *Prentice Hall, Upper Saddle River, NJ*, pp. 310–361.

Kalman, R. (1960): “A new approach to linear filtering and prediction problems.” *Transactions of the ASME–J. Basic Engineering*, **82:Series D**, pp. 35–45.

Khatib, O. (1987): “A unified approach for motion and force control of robot manipulators: The operational space formulation.” *IEEE J. Robotics and Automation*, **3:1**, pp. 43–53.

LabComm (2011): “`http://torvalds.cs.lth.se/moin/LabComm`.”

Lambrechts, P., M. Boerlage, and M. Steinbuch (2004): “Trajectory planning and feedforward design for high performance motion systems.” In *American Control Conference*, pp. 4637–4642.

Lamiroy, B., B. Espiau, N. Andreff, and R. Horaud (2000): “Controlling robots with two cameras: How to do it properly.” In *IEEE Intl. Conf. Robotics and Automation (ICRA2000)*, pp. 2100–2105. San Francisco.

Linderoth, M. (2008): “Vision based tracker for dart catching robot.” Master’s Thesis ISRN LUTFD2/TFRT--5830--SE. Department of Automatic Control, Lund University, Sweden.

Linderoth, M., A. Robertsson, K. Åström, and R. Johansson (2009): “Vision based tracker for dart-catching robot.” In *Preprints 9th IFAC Intl. Symp. Robot Control (SYROCO’09)*. Gifu, Japan, September 9-12, 2009, pp. 883-888.

Linderoth, M., A. Robertsson, K. Åström, and R. Johansson (2010): “Object tracking with measurements from single or multiple cameras.” In *Proc. Intl. Conf. Robotics and Automation (ICRA2010)*, pp. 4525–4530. Anchorage, AK.

Ma, Y., S. Soatto, J. Kosecka, and S. S. Sastry (2003): *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag.

Macfarlane, S. (2001): “On-line smooth trajectory planning for manipulators.” M.A.Sc. thesis. Dept. Mech. Eng., Univ. British Columbia, Vancouver, Canada.

Maciejowski, J. (2002): *Predictive Control with Constraints*. Prentice Hall, Pearson Education, England.

Marayong, P., G. D. Hager, and A. M. Okamura (2008): "Control methods for guidance virtual fixtures in compliant human-machine interfaces." In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS2008), September 22-26, 2008, Nice, France*, pp. 1166–1172.

Matsushima, M., T. Hashimoto, and F. Miyazaki (2003): "Learning to the robot table tennis task – ball control & rally with a human." *IEEE Intl. Conf. Systems Man and Cybernetics*, **3**, pp. 2962 – 2969.

Möbius, F. (1827): *Die Barycentrische Calcül, Reprinted 1967 in Gesammelte Werke, vol. 1, pp. 36–49*. Dr. M. Saendig oHG, Wiesbaden, Germany.

Moutarlier, P. and R. Chatila (1990): "An experimental system for incremental environment modelling by an autonomous mobile robot." In *The First Intl. Symp. Experimental Robotics I*, pp. 327–346. Springer-Verlag, London, UK.

Olsson, T., J. Bengtsson, A. Robertsson, and R. Johansson (2003): "Visual position tracking using dual quaternions with hand-eye motion constraints." In *IEEE Intl. Conf. Robotics and Automation (ICRA2003)*, pp. 3491–3496. Taipei, Taiwan.

Olsson, T., R. Johansson, and A. Robertsson (2005): "Force/vision based active damping control of contact transition in dynamic environments." In *Proc. 10th IEEE Intl. Conf. Computer Vision, Workshop on Dynamical Vision*. Beijing.

Olsson, T., R. Johansson, and A. Robertsson (2006): "High-speed visual robot control using an optimal linearizing intensity-based filtering approach." In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS2006)*. Beijing, China.

Pearl, J. (1985): "Bayesian networks: A model of self-activated memory for evidential reasoning." In *Proc. 7th Conference of the Cognitive Science Society*, pp. 329–334. Irvine, CA.

Real-Time Workshop (2011): "`http://www.mathworks.com/products/simulink-coder/index.html`."

Sato, Y., J. Takamatsu, H. Kimura, and K. Ikeuchi (2003): "Recognition of a mechanical linkage based on occlusion-robust object tracking." In *Proc. IEEE Conf. Multisensor Fusion and Integration for Intelligent Systems*, pp. 329–334. Tokyo, Japan.

Smith, R., M. Self, and P. Cheeseman (1990): *Estimating uncertain spatial relationships in robotics*, pp. 167–193. Springer-Verlag New York, Inc., New York, NY, USA.

Smith, R. C. and P. Cheeseman (1986): "On the representation and estimation of spatial uncertainly." *Intl. J. Robotics Research*, **5**, December, pp. 56–68.

Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006): *Robot Modeling and Control.* John Wiley & Sons, Hoboken, NJ.

Stateflow (2011): "`http://www.mathworks.com/products/stateflow`."

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2011): "Force controlled assembly of emergency stop button." In *Proc. 2011 IEEE Intl. Conf. Robotics and Automation (ICRA2011)*, pp. 3751–3756. May 9-13, 2011, Shanghai, China.

Thrun, S., Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte (2004): "Simulataneous localization and mapping with sparse extended information filters." *Intl. J. Robotics Research*, **23:7–8**, pp. 693–716.

Walter, M. R., R. M. Eustice, and J. J. Leonard (2007): "Exactly sparse extended information filters for feature-based SLAM." *Intl. J. Robotics Research*, **26:4**, pp. 335–359.

Weiner, L. B. (1981): "Kalman filter initialization with large initial uncertainty and strong measurement nonlinearity." *Proc. Region 3 Conf. and Exhibit, Huntsville, AL; United States*, pp. 150–151.

Xenomai (2011): "`http://www.xenomai.org`."

Yilmaz, A., O. Javed, and M. Shah (2006): "Object tracking: A survey." *ACM Comput. Surv.*, **38:4**.

Department of Automatic Control
Lund University
Box 118
SE-221 00 Lund Sweden

Document name
LICENTIATE THESIS

Date of issue
June 2011

Document Number
ISRN LUTFD2/TFRT--3251--SE

Author(s)
Magnus Linderoth

Supervisor
Rolf Johansson, Anders Robertsson

Sponsoring organisation
ROSETTA (EU FP7)

Title and subtitle
Robotic Work-Space Sensing and Control

Abstract

Industrial robots are traditionally programmed using only the internal joint position sensors, in a sense leaving the robot blind and numb. Using external sensors, such as cameras and force sensors, allows the robot to detect the existence and position of objects in an unstructured environment, and to handle contact situations not possible using only position control.

This thesis presents work on how external sensors can be used in robot control. A vision-based robotic ball-catcher was implemented, showing how high-speed computer vision can be used for robot control with hard time constraints. Special attention is payed to tracking of a flying ball with an arbitrary number of cameras, how to initialize the tracker when no information about the initial state is available, and how to dynamically update the robot trajectory when the end point of the trajectory is modified due to new measurements. In another application example, force control was used to perform robotic assembly. It is shown how force sensing can be used to handle uncertain positions of objects in the workspace and detect different contact situations.

Key words
Robotics, computer vision, visual tracking, filter initialization, on-line trajectory generation, force control, assembly, cooperating robots

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
114

Security classification

Recipient's notes