Preface

This work was done in the framework of the ECMI program, ECMI being the European Consortium for Mathematics in Industry. One of the main points of ECMI is that the research done within the program should have a strong industrial connection.

Parts of the work in this thesis is based on a project with a company based in Särö, outside Gothenburg, called Netzler and Dahlgren AB. They develop guidance systems for autonomous vehicles that are used in factory automation. The systems that we have worked on are vehicles which are guided by a laser scanner. This laser scanner can be seen as a camera with a one-dimensional retina. The equations governing one-dimensional vision are similar to those used in two-dimensional computer vision but as will be seen there are advantages of using normal camera instead.

The thesis is partly based on an earlier publication and an accepted paper

[HS1] H. Stewenius Automatic calibration of vehicle parameters from visual data. In *Proc. SSAB, Lund, Sweden*, 2002.

[HS2] H. Stewenius Simplified Vehicle Calibration using Multilinear Constraints. Accepted for *SCIA, Gothenburg, Sweden* 2003

[HS3] H. Stewenius and K. Åström Hand-Eye Calibration using Multilinear Constraints. Submitted.

Subsidiary Paper

[HS4] H. Malm and H. Stewenius Direct Calibration for Robot Vision. Submitted

Acknowledgments

First of all I would like to thank my two supervisors Kalle Åström and Gunnar Sparr. They have been vital in providing ideas and their careful reading has helped me to learn how to write a thesis that can hopefully by read and understood.

The members of the Mathematical Imaging Group at the department and in particular Henrik Malm has provided me with helpful advice and interesting discussions.

I would like to thank all the PhD-students at the department for providing an enjoyable work atmosphere.

Finally I would like to thank my family and friends for being there.

Contents

Chapter 1

Introduction

Thought is impossible without an image
Aristotle

1.1 Motivation

This thesis introduces an alternative way of thinking for vehicle and camera calibration for autonomously guided vehicles as well as for hand-eye calibration. Parts of the work have been done in cooperation with **Netzler & Dahlgren AB, NDC**, designs guidance systems for autonomous vehicles. **Autonomous guided vehicles**, called **AGV**, are important components of factory automation.

Traditional methods for camera based calibration often use a calibration target or are based on solving the structure and motion problem before solving for the parameters. In this thesis it is shown how to compute optimal parameters immediately from the observed data. This method of computation relies on using the Continuous Multilinear Constraint problem for many observations in a least squares sense by representing this least squares problem with a polynomial.

Two different situations for **AGV** are investigated. In the first case, with a laser sensor instead of a camera, it is shown that the distance measurements from the laser enable an analytic solution. In the second case, the same situation is investigated for an ordinary camera. As an additional example, it is shown how to use the method for hand-eye calibration.

Navigation of Autonomous Vehicles

Autonomous vehicles are already important in factory automation and will certainly be even more important in the future. To see that this technology will be advantageous is not difficult. Unfortunately, robot and autonomous vehicle technology can also lead to bad things in that it reduces the work force and in war. Fortunately this discussion is beyond the scope of this thesis and only the mathematical aspects of calibrating the vehicle motion and camera parameters will be considered here.

Why is Calibration interesting

In hand-eye calibration as well as for mobile robot camera calibration the data that we wish to compute are lengths and angles that should have have been known from construction of the system or could have been measured after construction. Why do we use

computer vision instead of simpler methods? The reasons to search for these measurements from images or other data are twofold, the first reason is that it is possible that it is easier to forget the information of the blueprints and measure again or that the blueprints are not available. The other reason is that in some cases, it is possible to get better accuracy when measuring this way than was possible in the process of manufacturing the robot.
For the Autonomously Guided Vehicles that will be the main part of this study a sensor is mounted on an existing vehicle and the exact parameters of this vehicle are unknown. The sought parameters are usually the geometric properties of the vehicle and the camera placement on this vehicle.
A related question is the problem of hand-eye calibration when a camera is mounted on a robot hand and we know the motion of the robot hand but would like to calculate the translation and rotation of the camera relative to the robot hand as well as the internal calibration of the camera.
With a calibrated vehicle it is possible to integrate data from both visual and odometric information of the vehicle. Odometric data is information from steering and velocity. This combination of data is important in the process of building a map of the environment as well as navigating on this map. It is theoretically possible both to build the map and to use it without a calibrated vehicle but known calibration simplifies the task.
An important situation for visual calibration is the hand-eye calibration problem in robotics. A camera is mounted on a robot hand for which the motion is known and we want to find the the location and orientation of the camera relative to the robot hand. There are many different ways to do this but most rely on either a measured calibration target or on a reconstruction of structure and motion.

When is it correct?

The simplest way to judge if the calibration is correct is to compare with measured values. Unfortunately such values are not in general available. Another way to see if the results are correct is to use the parameters for map-construction or similar work and if this is possible the residuals in this step are an indication if the parameters are good or bad.
The methods are tested on simulated data to check against bias and other systematic errors. For a laser-guided vehicle calibration as well as for hand-eye calibration there are also tests on real data.

1.2 Organization of the thesis

In chapter 2 a brief introduction will be given to map-building, computer vision with special attention to to the multilinear constraints. Vehicle calibration as well as hand eye calibration will also be introduced in this chapter together with the motion equations for

the vehicles to be considered.
Some early thought and ideas on how to perform calibration are introduced in chapter 3 along with the ideas used in later chapters. In chapter 4 a vehicle equipped with a laser-sensor is considered and different lines of thoughts are developed and results on real data presented.
The core of the thesis is chapter 5 where it is shown how to perform hand eye as well as vehicle calibration using Multilinear Constraints.
In an appendix some of the polynomial coefficients used in other chapters are listed.

Chapter 2

Background in computer vision , map building and robot calibration

The knowledge at which geometry aims is the knowledge of the eternal.
Plato (ca 429-347 BC) Republic, VII, 52.

2.1 Map Building

Maps of the environment have been constructed and used for a very long time. With maps there are two fundamental questions, the first and most difficult one is how to build the map, the second one is how to use the map.
Here we will be concerned with building a map of a set of points in the plane. For practical purposes there are two types of measurements. Absolute measurement of position relative to a fixed coordinate system (longitude latitude) , the early measurements in this category were obtained by observing the sun and the stars with additional information from a chronometer, later other absolute methods were developed using radio like GPS. The second type of measurements aim at finding the relation of neighboring points to each other. In this category falls measuring distances and measuring the the angle from one visible point to another. Computation of a position relative to a fixed coordinate system then has to be made through a long chain of relative measurements back to the origin of the coordinate system.

Autosurveyor

One application that uses automatic map-building is the **AGV** (Autonomously Guided Vehicle) control systems of Netzler & Dahlgren AB shown in figure (2.1). These systems use a one-dimensional laser-scanner and odometry information from the vehicle. The laser-scanners are intended to detect reflections from strips of a reflective material (beacons) in the environment and detect angle and approximate distance to the beacons relative to the vehicle . There is no identifying information in the beacons. This system is described at length in [22, 23, 3, 1, 16], and one part of a map is shown in figure (2.2). If correctly calibrated, the system can, automatically build a map of its surroundings and will then use this map for navigation. If the calibration is approximately correct it is possible to use the system for minor adjustments of calibration parameters but this procedure is very slow. If the initial guess on the parameters is not correct, the system fails.

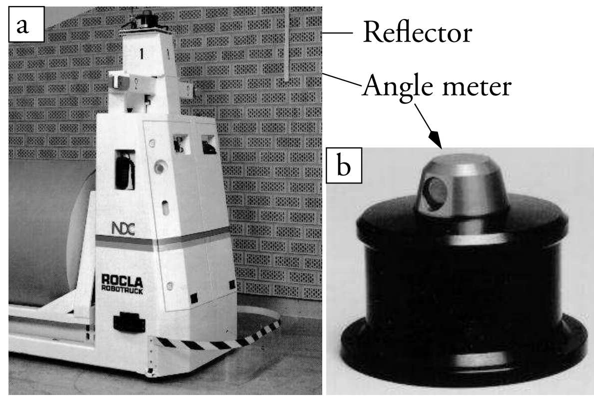

Figure 2.1: The figure illustrates a typical truck with a laser scanner mounted on top. On the walls the reflective tapes can be seen.

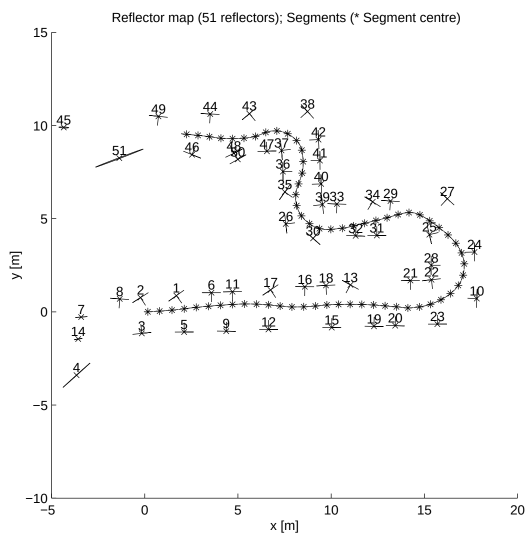

Figure 2.2: Map built with Autosurveyor. Stars are vehicle positions, numbered points are beacon positions.

2.2 Computer Vision

Computer vision deals with the problem of finding a 3D structure from 2D gray-scale or colour images. A common procedure is to use some point tracking algorithm to simplify the problem to a problem of labeled points. One such tracker is the KLT tracker [25]. In this thesis it will be assumed that the tracking step has already been made and that we have no problem in associating corresponding points in different images, hopefully globally, but at least locally.

Projective Geometry

Every point, $\mathbf{u}$, in the projective space $\mathbb{P}^n$ of n dimensions correspond to a unique line in $\mathbb{R}^{n+1}$ passing through origo. As every such line is defined by any non-zero point on the line, it is possible to represent any point in $\mathbb{P}^n$ by a point in $\mathbb{R}^{n+1}\setminus 0$. The coordinates of these corresponding points in $\mathbb{R}^{n+1}\setminus 0$ are called homogeneous coordinates of the point u. A hyper-plane in $\mathbb{P}^n$ is defined by an $(n+1)$-tuple $l \neq 0$ and consists of all points in $\mathbb{P}^n$ with homogeneous coordinates u fulfilling $l^t u = 0$. Angles between lines and distances between are not defined in projective spaces. A projective transformation correspond to multiplication of the homogeneous coordinates with a non-singular $(n+1) \times (n+1)$ matrix. By choosing one hyper-plane in $\mathbb{P}^n$ as the plane at infinity, the affine geometry is constructed. In affine geometry lines are considered parallel if they intersect at the plane at infinity.
By choosing a particular conic at the plane at infinity, i.e. the so called absolute conic, and by choosing a scale, Eucledian structure is imposed. This makes it possible to define distances and angles.
Example. The point (X, Y, Z) in $\mathbb{R}^3$ corresponds to a point in $\mathbb{P}^3$ that is represented by the set $\{(X, Y, Z, 1)t; t \neq 0\}$ in $\mathbb{R}^4\setminus\{0\}$ ■
Example. In $\mathbb{P}^2$ the point represented by $(3, 2, 1)$ can also be represented by $(6, 4, 2)$ and $(-3, -2, -1)$ but not $(0, 0, 0)$. ■

A problem with projective geometry is that there is no concept of distances or angles and we have to impose Eucledian Structure to get these notions. Saying that something is known up to a projective transformation means that the true structure is known to be a member of the set of structures generated by projective transformations on the class member that we have computed.

The camera equation

Assuming the pinhole camera model is used, a point in space represented by homogeneous coordinates $(X, Y, Z, 1)$ is mapped to a point in the image plane with homogeneous coordinates $(x, y, 1)$, see figure (2.2). This gives the standard projective camera equation [12]

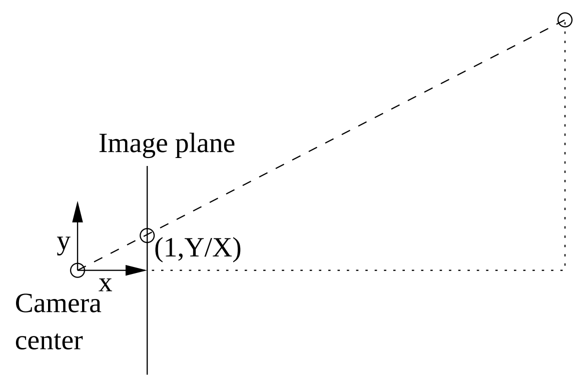

Figure 2.3: Projection formula of a pinhole camera

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

where the camera matrix $P = K\,[R \quad -RT]$ is a 3×4 matrix containing information on both the internal parameters, K, of the camera as well as the external parameters, $[R \;\; RT]$, that is position and orientation, of the camera. If the camera is calibrated it can after a suitable choice of image coordinate system be assumed $K = I$, the identity matrix. For 1D cameras in the plane the corresponding equation is written

$$\lambda \begin{pmatrix} x \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}.$$

Here P is a 2×3 matrix. Here too P can be decomposed as $P = K\,[R \quad -RT]$ where

$$K = \begin{bmatrix} f & x_0 \\ 0 & 1 \end{bmatrix}$$

are the internal calibration parameters and (R, T) represents position and orientation of the camera with respect to the world coordinate system. All other equations in computer vision are then derived from the camera equation, sometimes by imposing additional constraints.

Most of the time the notation U will be chosen for points in space and u for points in the image plane

$$\lambda u = PU. \tag{2.1}$$

Reconstruction

Using the theory for structure and motion it is possible to make a reconstruction of the structure of the scene, the motion of the camera and the internal calibration of the camera from a sequence of images. This reconstruction can be made both in 3D for 2D cameras and in 2D for 1D cameras. All textbooks and most articles on Computer Vision deal with some aspect of the structure and motion problem, here only two recent textbook will be mentioned for reference [10, 12].

The usual way to perform structure and motion reconstruction is to begin from a minimal configuration and solve for structure and motion. The points and cameras thus calculated are then used with the information in other images and of other points to generate more points and cameras locations. In order to deal with problems coming from uncertainty in point estimates and numerical instability a numerical optimization scheme, bundle adjustment, is applied.

There are many more or less automatic systems for the reconstruction. One of the more impressive ones is from the group in Oxford [31].

Having cameras of constant calibration limits the amount of unknown variables and having known calibration is even more restrictive. If we have uncalibrated cameras the solution can only be computed up to a projective transformation. There are also problems with the stability of the solutions.

2.3 The Multilinear Constraints

With a known camera moving in a known way the image of a known point in space would have a well known orbit in the image. Even if some of the above quantities are unknown there are restrictions on their possible values. Two such constraints are the discrete and the continuous multilinear constraints relating movement in the images to possible movements of the cameras. A complete treatment can be found in [14, 15, 9, 27].

Definition 2.3.1. The **nth order discrete multilinear constraint in space** is

$$\operatorname{rank}\begin{bmatrix} P(t_0) & u(t_0) & \mathbf{0} & \dots & \mathbf{0} \\ P(t_1) & \mathbf{0} & u(t_1) & \dots & \mathbf{0} \\ P(t_2) & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P(t_n) & \mathbf{0} & \mathbf{0} & \dots & u(t_n) \end{bmatrix} \leq n+4 \ .$$

■

Theorem 2.3.1. *In a sequence of discrete images corresponding points with coordinates* $u(t_0), u(t_1), \ldots, u(t_n)$ *obey the nth order discrete multilinear constraint.*

Proof. This can be seen by lining up each projection constraint

$$\lambda(t_i)u(t_i) = P(t_i)U$$

in a linear matrix equation

$$\underbrace{\begin{bmatrix} P(t_0) & u(t_0) & \mathbf{0} & \ldots & \mathbf{0} \\ P(t_1) & \mathbf{0} & u(t_1) & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P(t_n) & \mathbf{0} & \mathbf{0} & \ldots & u(t_n) \end{bmatrix}}_{M} \begin{bmatrix} -U \\ \lambda(t_0) \\ \lambda(t_1) \\ \vdots \\ \lambda(t_n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} .$$

Since this system has a non-trivial solution, M cannot have full rank. In space M is of size $(3n + 3 \times n + 5)$ and therefore $\operatorname{rank}(M) \leq n + 4$. ∎

Considering the first order constraint $(n = 1)$ in space gives

$$\operatorname{rank} \begin{bmatrix} P(t_0) & u(t_0) & \mathbf{0} \\ P(t_1) & \mathbf{0} & u(t_1) \end{bmatrix} \leq 5.$$

which as the above matrix is of quadratic of size 6 is equivalent to

$$\det \begin{bmatrix} P(t_0) & u(t_0) & \mathbf{0} \\ P(t_1) & \mathbf{0} & u(t_1) \end{bmatrix} = 0. \tag{2.4}$$

Sometimes it is easier to deal with derivatives and Heyden and Åström introduced in 1998 [4] a continuous analogue of the above theorem.

In

$$\lambda(t)u(t) = P(t)U \ . \tag{2.5}$$

Use the Taylor series expansions

$$\begin{aligned} \lambda(t) &= \lambda^{(0)} + \lambda^{(1)}t + \ldots, \\ u(t) &= u^{(0)} + u^{(1)}t + \ldots, \\ P(t) &= P^{(0)} + P^{(1)}t + \ldots \ . \end{aligned} \tag{2.6}$$

Recall that superscript $^{(i)}$ denotes the ith coefficient in the Taylor series expansion, e.g. $\lambda^{(k)} = \frac{1}{k!}\frac{d^k\lambda}{dt^k}$. Substituting the Taylor series expansions into (2.5) gives

$$\begin{aligned} &(\lambda^{(0)} + \lambda^{(1)}t + \lambda^{(2)}t^2 + \ldots)(u^{(0)} + u^{(1)}t + u^{(2)}t^2 + \ldots) = \\ &\qquad = (P^{(0)} + P^{(1)}t + P^{(2)}t^2 + \ldots)\,U. \end{aligned} \tag{2.7}$$

Identifying the coefficients of t^i for $i = 0, \ldots, n$ gives

$$\begin{bmatrix} P^{(0)} & u^{(0)} & \mathbf{0} & \dots & \mathbf{0} \\ P^{(1)} & u^{(1)} & u^{(0)} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P^{(n)} & u^{(n)} & u^{(n-1)} & \dots & u^{(0)} \end{bmatrix} \begin{bmatrix} -U \\ \lambda^{(0)} \\ \lambda^{(1)} \\ \vdots \\ \lambda^{(n)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} .$$

Since this set of equations has a non-trivial solution the leftmost matrix cannot have full rank.

Definition 2.3.2. The **nth order continuous constraint** is

$$\operatorname{rank} \begin{bmatrix} P^{(0)} & u^{(0)} & \mathbf{0} & \dots & \mathbf{0} \\ P^{(1)} & u^{(1)} & u^{(0)} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P^{(n)} & u^{(n)} & u^{(n-1)} & \dots & u^{(0)} \end{bmatrix} \leq n + 4. \tag{2.9}$$

where $f^{(i)}$ is the Taylor coefficient of order i of f. ■

Theorem 2.3.2. *In a continuous sequence of images the image coordinates $u^{(0)}$ and their derivatives, $u^{(i)}$, up to order n at the same instant of time, obey the nth order continuous constraint.*

Proof. Follows from the derivation above. ■

First order constraint in space

In space the first order continuous constraint can be written as

$$\operatorname{rank} M \leq 5, \tag{2.10}$$
$$M = \begin{bmatrix} P^{(0)} & u^{(0)} & \mathbf{0} \\ P^{(1)} & u^{(1)} & u^{(0)} \end{bmatrix}$$

but as M is 6×6 this imposes

$$\det \begin{bmatrix} P^{(0)} & u^{(0)} & \mathbf{0} \\ P^{(1)} & u^{(1)} & u^{(0)} \end{bmatrix} = 0. \tag{2.11}$$

With $P(t) = A[R(t)\; R(t)T(t)]$, $T(0) = 0$ and $R^0 = I$ we rewrite this as

$$\det \begin{bmatrix} A & \mathbf{0} & u^0 & \mathbf{0} \\ AR^1 & AT^1 & u^1 & u^0 \end{bmatrix} = 0. \tag{2.12}$$

2.4 Calibration

In many cases a construction is known up to some unknown system parameters. If these numbers can be found the performance of the system will be improved. Such parameters can be camera parameters, parameters in motion equations or other parameters of importance. The most obvious way to get these parameters is to measure them when the system is built, either in the blueprints or on the system itself. Sometimes the demands in precision on the parameters are beyond the reliability of the manufacturing process and it is difficult to measure the parameters from the object directly, sometimes the robot is assembled from parts from different sources and it would be convenient not having to keep track of all values. Another possibility is that the parameters are changed after the object was put to use. When the parameters are unknown it would be helpful if the robot was able to calculate by itself in order to perform self-calibration.
In computer vision mainly two types of calibration are investigated. First the problem of camera calibration and secondly the problem of hand-eye calibration. Camera calibration deals with how to determine a set of parameters of a camera. In hand-eye calibration the camera is rigidly attached to a robot hand for which the motion is known, the task is to determine the translation and the rotation from the hand to the camera as well as the internal parameters of the camera. The hand-eye problem has been solved in many different ways. Tsai and Lenz solve the problem by solving for camera positions using a well-known calibration target [18] and then factoring out the unknown transformation. They also solve the problem of unknown parameters in the robot [17].
Most other authors solve the problem in a similar way by solving for the unknown cameras and then factoring out the affine transformation, Andreff, Horaud and Espiau [2] use elegant linear formalism to reduce the problem to a linear problem, Dornaika and Horaud[8] use quaternions to linearize the problem.
Wei, Arbter and Hirzinger [28] use the robot motion to solve for the unknown calibration point and to calibrate the camera, the hand-eye translation and the lens distortion. They also consider motion planning for the calibration process. Dias et al [7] propose to use linear methods together with a Kalman or a recursive filter to increase stability. Zhuang and Qu [30] propose an improved Jacobian for use in a one-stage iterative algorithm based on the reprojection errors. Malm and Heyden deviate strongly from the other methods in that they use a motion field-based method instead of a point-based method [20].

AGV Literature

The question of auto-calibration for AGV robots is related to conventional calibration. Unfortunately there is less freedom of movement and sometimes necessary to find certain vehicle parameters as well. The amount of literature concerning calibration of AGV is quite limited.
McNamee, Petriu and Spoelder [19] take images of a mobile robot and compute the

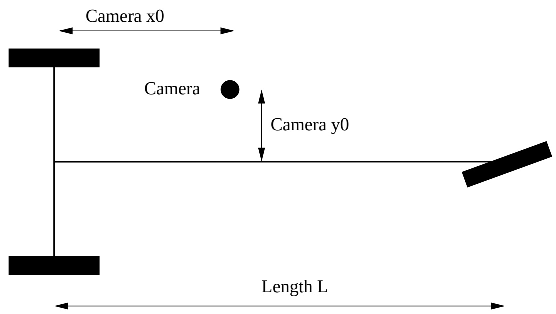

Figure 2.4: Parameters for a three wheeled cart

positions of a target on the robot and then use a Kalman model to find wheel radius and wheel distance on a 4 wheeled vehicle with differential control. An early paper on the topic is the one by Chang and Aggarwal [6] describing how to use rectangular corners in space to compute the motion of the vehicle and then compute the unknown parameters.

2.5 Vehicle Motion

For this kind of application it is important to choose a vehicle for which the motion is not too complicated, in industrial applications there are two-wheel configurations that are especially convenient when it comes to calculations. In both cases the wheels can be considered as non-slipping, that is, each wheel moves only in the direction in which it is pointing. This is not the case for all wheels of an ordinary car.

3 wheels

A common wheel configuration is one with three wheels, two non-driving fixed wheels at the rear axis and a driving steering wheel at a distance L from the rear axis. Control signals for this configuration are the speed $v(t)$ and steering angle $s(t)$ of the front wheel. The translation of the center of the rear axis is orthogonal to the rear axis. The speed of the translation is the projection of the speed of the front wheel to the x-axis , that is $v(t)\sin(s(t))$.

To investigate the rotation we see that the vehicle will travel on a circle whose center belongs to both wheel axes. The radius of the circle that to the front wheel travels along is $L/\sin(s(t))$ and therefore

$$\dot{\theta}(t) = v\sin(s(t))/L. \tag{2.13}$$

To compute the movement of the camera the effects of the rotation are added to the translation of the rear wheel

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = v \begin{bmatrix} -x_0 l \sin(s) \\ y_0 l \sin(s) - \cos(s) \end{bmatrix}.$$

Substitution with

$$\begin{aligned} l &= 1/L, \\ X &= x_0/L, \\ Y &= y_0/L \end{aligned}$$

gives

$$T' = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = v \begin{bmatrix} -X \sin(s) \\ Y \sin(s) - \cos(s) \end{bmatrix}. \tag{2.14}$$

In order to compute derivatives in a coordinate system fixed on the floor, positions are computed by integration of movement in this coordinate system

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \int_{\tau=t_0}^{t} R(\theta(\tau)) \begin{bmatrix} v_x(\tau) \\ v_y(\tau) \end{bmatrix} dt \tag{2.15}$$

differentiating twice in a fixed coordinate frame

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = R(\theta(t)) \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix}, \tag{2.16}$$

after a third differentiation

$$\begin{bmatrix} x''(t) \\ y''(t) \end{bmatrix} = R'(\theta(t)) \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix} + R(\theta(t)) \begin{bmatrix} v'_x(t) \\ v'_y(t) \end{bmatrix}. \tag{2.17}$$

Where rotation and rotational velocity can be expressed

$$R(\theta(t)) = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) \\ -\sin(\theta(t)) & \cos(\theta(t)) \end{bmatrix}, \tag{2.18}$$

$$R'(\theta(t)) = \dot{\theta} \begin{bmatrix} -\sin(\theta(t)) & \cos(\theta(t)) \\ -\cos(\theta(t)) & -\sin(\theta(t)) \end{bmatrix}, \tag{2.19}$$

$$\begin{aligned} R''(t) = {} & \dot{\theta}^2 \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) \\ -\sin(\theta(t)) & -\cos(\theta(t)) \end{bmatrix} + \\ & + \ddot{\theta(t)} \begin{bmatrix} -\sin(\theta(t)) & \cos(\theta(t)) \\ -\cos(\theta(t)) & -\sin(\theta(t)) \end{bmatrix}. \end{aligned} \tag{2.20}$$

Evaluating 2.18, 2.19 and 2.20 in 0 gives

$$\begin{aligned}
R(0) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, && (2.21)\\
R'(0) &= \dot{\theta} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, && (2.22)\\
R''(0) &= \begin{bmatrix} \dot{\theta}^2 & \ddot{\theta(t)} \\ -\ddot{\theta(t)} & -\dot{\theta}^2 \end{bmatrix}. && (2.23)
\end{aligned}$$

This can be inserted into 2.15,2.16 and 2.17 to compute the value at 0

$$\begin{aligned}
\begin{bmatrix} x(0) \\ y(0) \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, && (2.24)\\
\begin{bmatrix} x'(0) \\ y'(0) \end{bmatrix} &= \begin{bmatrix} v_y \\ v_x \end{bmatrix}, && (2.25)\\
\begin{bmatrix} x''(0) \\ y''(0) \end{bmatrix} &= \begin{bmatrix} \dot{\theta}(Yv\sin(s) - v\cos(s)) \\ -Xv_p\sin(s) - Xvs_p\cos(s) \\ \dot{\theta}Xv\sin(s) - v_p\cos(s) + vs_p\sin(s) \\ +Yv_p\sin(s) + Yvs_p\cos(s) \end{bmatrix}. && (2.26)
\end{aligned}$$

As seen, all motion and motion derivatives can be computed if signals (v, s) and parameters (l, X, Y) are known. As there is no motion along the z-axis, camera matrices and their derivatives can be assembled

$$P(t) = \begin{bmatrix} R(t) & R(t)T(t) \end{bmatrix}.$$

4 wheels

Carts with wheels according to fig 2.6 are controlled by differential control. The motion equations are quite simple with linear expressions in all control variables. Assuming that the distance L between the driving wheels is unknown as well as the odometric constants k_1 and k_2 for the wheels and the camera position (x_0, y_0) on the cart we get a parameter set (L, k_1, k_2, x_0, y_0) with five unknowns for the movements of the camera center.

For the center of the cart

$$\begin{aligned}
v_x &= \frac{k_1v_1 + k_2v_2}{2}, \\
v_y &= 0, \\
\dot{\theta} &= \frac{k_1v_1 - k_2v_2}{L}.
\end{aligned}$$

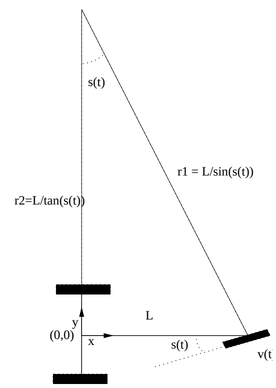

Figure 2.5: Movement calculations of the cart

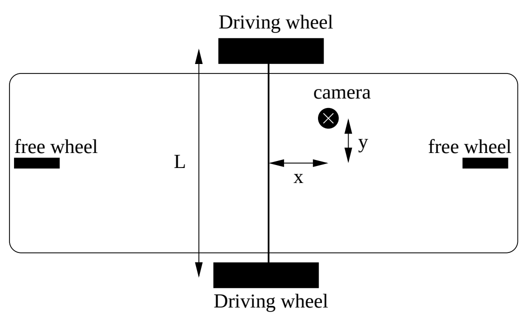

Figure 2.6: Parameters for a four wheeled cart.

Motion for the camera center is derived

$$\begin{aligned} v_x &= \frac{k_1 v_1 + k_2 v_2}{2} - y\dot{\theta}, \\ v_y &= 0 + x\dot{\theta}, \\ \dot{\theta} &= \frac{k_1 v_1 - k_2 v_2}{L}. \end{aligned}$$

Simplifying with $l = 1/L$, $X = x_0/L$ and $Y = y_0/L$

$$\begin{aligned} v_x &= \frac{k_1 v_1 + k_2 v_2}{2} - Y(k_1 v_1 - vb), \\ v_y &= X(k_1 v_1 - k_2 v_2), \\ \dot{\theta} &= l(k_1 v_1 - k_2 v_2). \end{aligned}$$

In the parameter set $x = (l, k_1, k_2, X, Y)$ the scale that cannot be reconstructed from images alone and it is advisable to lock one of the parameters. It is in some cases known that $k_1 = k_2$ from construction. Integrating to get absolute positions

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix}(t) &= \int_{\tau=0}^{t} R(\theta(t)) \begin{bmatrix} v_x \\ v_y \end{bmatrix} d\tau, \\ \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}(t) &= R'(\theta(t))\dot{\theta}(t) \begin{bmatrix} v_x \\ v_y \end{bmatrix} + R(\theta(t)) \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \end{bmatrix} \end{aligned}$$

but

$$\begin{aligned} R(\theta(0)) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ R'(\theta(0)) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

so

$$\begin{aligned} [\ddot{x}, \ddot{y}](0) &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \dot{\theta}(t) \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \end{bmatrix} \\ &= \begin{bmatrix} -\dot{\theta} v_y + \dot{v_x} \\ \dot{\theta} v_x + \dot{v_y} \end{bmatrix}. \end{aligned} \tag{2.27}$$

The second order derivatives are needed to use second order Continuous Multilinear Constraints.

2.6 Numerics

Optimization

The functions to be minimized will be polynomials and in some cases it is possible to reduce the number of variables by using the fact that there is an analytic solution to the minima of second order quadratic forms.

Theorem 2.6.1. *If A is a positive definite symmetric matrix, b is a vector, c is a constant. Then $x^T Ax + b^T x + c$ is minimized by $x = -(2A)^{-1}(b^t)$.*

The above theorem is useful when the target-function is polynomial of total degree less than two in a subset of the variables. It is then possible to eliminate some variables and thus constrain the dimensionality of the problem.

Richardson extrapolation

Some of the derivatives involved in the continuous multilinear constraint cannot be calculated analytically but have to be estimated numerically. This is the case for the derivative u' of the image point u. When numerically computing a derivative we want $\Delta t \to 0$ but small values of Δt lead to strong amplification of noise levels while high Δt in some instances leads to systematic errors. In order to reduce this problem the Richardson Extrapolation method is used, a more detailed introduction to this method can be found in[13].

A function $f(h)$ is defined for $h > 0$ and we wish to compute $lim_{h\to 0} f(h)$. Assuming that $f(h)$ has a polynomial approximation $p(h)$ close to 0. Then p is estimated by functional evaluations close to 0 and evaluated in $h = 0$

Example. Due to noise and quantization errors in the data it is impossible to calculate directly

$$f'(t) = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t\Delta t)}{2\Delta t}.$$

Calculate $f'_{\Delta t}(t)$ for some different Δt, find a polynomial approximation and compute $f'_0(t)$. ■

Chapter 3

Attempts at Calibration

Where there is matter, there is geometry. (Ubi materia, ibi geometria.)
Kepler, Johannes (1571-1630)

3.1 Problem formulation

Given a three wheeled cart with unknown parameters x. By performing an experiment with a system (a robot, a vehicle) with observations $\mathbf{o} = \{o_i\}_i$ consisting of measured odometry data and also measured visual data $\mathbf{u} = \{u_i\}_i$. The main idea is to construct a function $f(x, \mathbf{o})$ such that f is minimal when for the correct parameters and also such that f is not computationally expensive neither in preparation time not in iteration time when optimizing.

The most reliable target function for calibration would be to use the parameters in our motion model to get vehicle positions and from these positions calculate point coordinates for the points in the images. As two points suffice to calculate point coordinates, points visible in more images will lead to over-determined systems and a residual can be calculated. The problem with this approach is that each iteration requires a full reconstruction and is thus very costly.

For the hand-eye calibration problem the motion of the hand is known and the possibility to make translations in all directions simplifies the situation.

3.2 Early Attempts

With a full and correct reconstruction of the structure and motion problem, the issue of calibration would be quickly solved. Unfortunately the full and stable reconstruction is difficult to obtain and the task would at best be quite time consuming.
Attempts were made in this direction but problems were encountered. The main reason to avoid this approach is that a full reconstruction contains much more information than is asked for.

3.3 Linear Function Spaces

When constructing f, it is important to get f to behave nicely over summation. Let $\mathbf{o} = \{o_j\}$ be the observed data and x the vehicle and camera parameters. We would like

f to fulfill

$$f(x, \mathbf{o}) = \sum_j f(x, o_j) = \sum_{i=1}^{N} p_i(x) \sum_j g_i(o_j). \tag{3.1}$$

with N fixed and not too big. The author has been unable to build a function f based on the reconstruction fulfilling the above conditions. Using the discrete and continuous multilinear constraints it is possible to build such functions.
Example. With $f_1(x, y) = \sin(xy)$ we get

$$f_1(x, 4.3) + f_1(x, 3.7) + f_1(x, 6.3) = \sin(4.3x) + \sin(3.7x) + \sin(6.3x).$$

There are no interesting simplifications except for Taylor expansions, but with $f_2(x, y) = yx^2$ we get

$$f_2(x, 4.3) + f_2(x, 3.7) + f_2(x, 6.3) = 4.3x^2 + 3.7x^2 + 6.3x^2 = 14.3x^2$$

in this we thus have

$$\sum_i f_2(x, c_i) \quad = \quad f_2(x, \sum_i c_i) \tag{3.2}$$

. ■

Experiments with Taylor expansions show some promise but also substantial problems, mainly with stability. The goal in the following is to build target functions of the type in equation (3.1)

3.4 Using Continuous multilinear constraints

The determinant of a matrix is a polynomial in the elements of the matrix. If the elements of the matrix are polynomial in the vehicle parameters, the determinant will also be a polynomial in these parameters. The continuous constraint is the determinant of a matrix with elements that are polynomials in the parameters x and constants calculated from the measured data. As the determinant is a polynomial in the matrix elements, the continuous constraint is a polynomial in x and coefficients calculated from known data.
A full description is given in chapter 5.

3.5 2D case with distance

In the special case when distances as well as angles to the tracked points are known it is possible to avoid the use of the second order continuous constraint by estimating motion directly from the measurements and then calculate a polynomial target function for the

parameters.
It is also possible to use the motion constraint to immediately compute a target function that is minimized to get the parameters.

Chapter 4

A first simple case

> ...by natural selection our mind has adapted itself to the conditions of the external world. It has adopted the geometry most advantageous to the species or, in other words, the most convenient. Geometry is not true, it is advantageous.
> *Poincaré, Jules Henri (1854-1912) Science and Method.*

4.1 2D with distances, the growth of an idea

A simplified case is a robot vehicle with a laser scanner instead of a camera moving in an environment marked with reflective strips. This system is described in [16]. This is similar to measurements in ordinary vision with the following characteristics:

- The image is 1D
- The field of vision is 360^o and the internal parameters of the camera are known.
- Approximate range measurements are available
- The hardware used here has a built in point detector that detects points marked with reflective tape and hopefully nothing else.

This gives several advantages over a traditional camera as the feature detection is made by the camera and the distances are most helpful when determining corresponding points. Having distances also enables us to solve for vehicle motion and compare with vehicle motion values computed from odometry.

Usually, algorithms are developed for automatic feature detection, feature association and structure and motion estimation, cf. [21, 24]. However, these algorithms require a calibrated vehicle. There are also methods for finding the calibration parameters of the vehicle, but these require a well known map. Here it is shown how to estimate the calibration parameters from raw unassociated data. The result can be used as a preprocessing step to building a map.

A key observation made is that the problem can be treated more efficiently if one uses the derivatives of vehicle motion. In this paper it is shown how these derivatives can be estimated from raw partially associated data if approximate distances to the strips of reflective material are known. It is also shown how odometry data, calibration data and motion derivatives are connected. This makes it possible to estimate the calibration

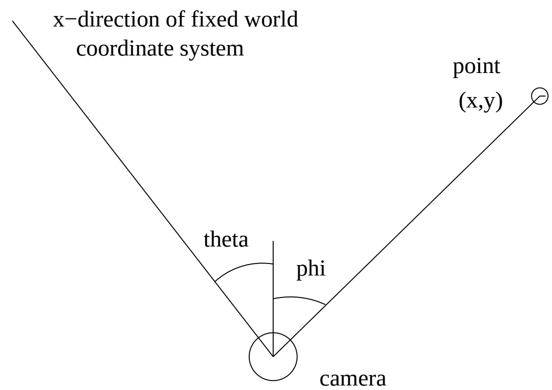

Figure 4.1: A camera with x-axis pointing forward. Angles ϕ and θ to point and world coordinate systems respectively

parameters. Instead of solving globally for structure, local solutions for motion will be computed. Associating corresponding reflexes is done by a linear model of reflex movement in the field of vision. By applying very strict constraints of acceptance, most false correspondences can be eliminated.

By choosing a coordinate system centered on the laser sensor the vehicle is immobile while the world and the detected points are moving. The angle from the vehicle coordinate system to the world coordinate system is θ. Point i is at position $(x, y)_i$ with a measured angle ϕ_i and moving with velocity $(v_x, v_y)_i$. As all points in the world are rigidly related to each other they all move with the same velocity (V_x, V_y). From figure (4.1)

$$\theta = \phi_i + \arctan \frac{y_i}{x_i}.$$

We differentiate and set $r^2 = x^2_{reflex} + y^2_{reflex}$. Let V_o be the rotation speed of the vehicle. Remembering that $\frac{d(x_i, y_i, \theta)}{dt} = (V_x, V_y, V_o)$ we now get:

$$V_o = \dot{\phi}_i + \frac{V_y x_i - V_x y_i}{r^2}. \tag{4.1}$$

4.2 Optimizing the parameters

In order to find the correct parameters some different methods were tested. The first two are very similar and will be described first, then a quite different method will be described.

Reconstruction and the parameter fitting

First attempts were made to estimate the structure and motion This was partially successful but it was quite time consuming due to repeated use of bundle adjustment and there was also problems with numerical stability.

Iterative and Analytic solutions

We remind of equation 4.1 From our measurements we get ϕ_i and r_i and from these $(x, y)_i$ for all visible beacons. We now have a number of equations and only 3 unknown variables and can in most cases solve this system to get (V_x, V_y, V_o). In this way vehicle motion can be estimated for most times t. To get a more precise estimate of movement only allow values based on over determined systems with limited residuals will be used.

For most times the movement can be calculated from laser measurements and at these times movement can also be computed for any set of parameters

W , velocities calculated from odometric data. With s the steering angle, v_r the measured velocity of the front wheel and x the vehicle parameters we have $(s, v_r, x) \rightarrow W$.

V , velocities calculated from laser data.

These velocities are subtracted to compute the error

$$
\begin{aligned}
V - W &= \begin{bmatrix} V_x - W_x \\ V_y - W_y \\ V_o - W_o \end{bmatrix} \\
&= \begin{bmatrix} V_x \\ V_y \\ V_o \end{bmatrix} - k_v v_r R \begin{bmatrix} \cos k_a s - y_0 \sin k_a s/L \\ x_0 \sin k_a s/L \\ \sin k_a s/L \end{bmatrix}
\end{aligned}
$$

with $R(\theta)$ the estimate for the offset rotation of the laser

$$
R = \begin{bmatrix} cos(\theta_0) & sin(\theta_0) & 0 \\ -sin(\theta_0) & cos(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

$R(\theta)$ is a rotation matrix and therefore orthogonal

$$
V - W = R \left(R^T v - \begin{bmatrix} (k_v v_r \cos k_a s - k_v v_r y_0 \sin k_a s/L) \\ k_v v_r x_0 \sin k_a s/L \\ k_v v_r \sin k_a s/L \end{bmatrix} \right).
$$

Since s is small in most parts of the data set, $\sin(k_a s) \approx k_a s$. This implies that it will be difficult to estimate k_a and L separately. However, the relation k_a/L can be estimated reliably. In the sequel this problem is avoided by setting $k_a = 1$.

$$\begin{array}{rcl@{\qquad}rcl} k_1 &=& k_v, & k_4 &=& k_v/L, \\ k_2 &=& k_v y_0/L, & a &=& v_r \cos k_a s, \\ k_3 &=& k_v x_0/, L & b &=& v_r \sin k_a s \end{array}$$

we get

$$V - W = R \left(R^T v - \begin{bmatrix} ak_1 + bk_2 \\ bk_3 \\ bk_4 \end{bmatrix} \right).$$

For each t we set $f = (v-w)^T(v-w)$ and use

$$z = \begin{bmatrix} ak_1 + bk_2 \\ bk_3 \\ bk_4 \end{bmatrix}$$

to rewrite f as

$$\begin{array}{rcl} f &=& (v-w)^T(v-w) \\ &=& (R(R^T v - z))^T(R(R^T v - z)) \\ &=& (R^T v - z)^t(R^T v - z) \\ &=& v^T R^T R v - v^T R^T z - z^T R v + z^T z \\ &=& v^T v - 2v^T R^t z + z^T z. \end{array}$$

Notice that z is linear in $\{k_i\}$. Consequently f is bilinear in the $\{k_i\}$. The sum of f for different time instants t_j is chosen as target function

$$F(z) = \sum_j f(z, t_j).$$

To optimize this function we calculate partial derivatives and set these to 0. This gives a system of 5 equations.

$$\tfrac{dF}{dk_i} = 0, \qquad \tfrac{dF}{d\theta} = 0.$$

With the data as column vectors we substitute

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl} r_1 &=& av'_x, & r_3 &=& bv'_x, & r_5 &=& aa', \\ r_2 &=& av'_y, & r_4 &=& bv'_y, & r_6 &=& bb', \\ & & & r_8 &=& bv'_o, & r_7 &=& ab'. \end{array}$$

It is important to note that all r_i are scalars and can be calculated without knowing the vehicle parameters. The conditions on the partial derivatives can be written

$$\begin{cases} r_1\cos\theta - r_2\sin\theta \; -r_5k_1 \; -r_7k_2 & = 0 \\ r_3\cos\theta - r_4\sin\theta \; -r_7k_1 \; -r_6k_2 & = 0 \\ r_4\cos\theta + r_3\sin\theta \qquad -r_6k_3 & = 0 \\ r_6k_4 & = r_8 \\ (r_2k_1+r_4k_2-r_3k_3)\cos\theta + (r_1k_1+r_3k_2+r_4k_3)\sin\theta & = 0 \end{cases} . \tag{4.2}$$

When $(r_5r_6 - r_7^2) \neq 0$, $r_6 \neq 0$ we can solve the first four rows for k_i with respect to $\cos(\theta)$ and $\sin(\theta)$. The k_i can then be substituted into the fifth row. With substitutions

$$\begin{aligned} \beta_1 &= \frac{-r_1r_2r_6 + r_2r_3r_7 + r_1r_4r_7 - r_3r_4r_5}{r_5r_6 - r_7^2} + \frac{r_3r_4}{r_6} \\ &= \frac{(r_3r_7 - r_1r_6)(r_2r_6 - r_4r_7)}{r_6(r_5r_6 - r_7^2)}, \\ \beta_2 &= \frac{-r_1^2r_6 + 2r_1r_3r_7 + 2r_2^2r_6 - r_2r_4r_7 - r_3^2r_5 + r_4^2r_5}{r_5r_6 - r_7^2} - \frac{r_4^2 - r_3^2}{r_6}. \end{aligned}$$

we get

$$\beta_1(\cos^2(\theta) - \sin^2(\theta)) + \beta_2(\cos(\theta)\sin(\theta)) = 0$$

or equivalently

$$\beta_1 \cos(2\theta) + \beta_2 \sin(2\theta) = 0.$$

Assuming that the laser sensor is pointing forward the solution closest to 0 of the above equation is chosen. With θ computed $(k_1, k2, k_3, k_4)$ can be solved for in least squares sense in equation (4.2). In figure 4.2 the computed values are compared with the values from the blueprints for a number of different vehicles and different datasets. It is important to note that for our measured values the scale comes from the distances given by the laser-scanner and these measurements can be biased.

Polynomial search method

We will use equation (4.1):

$$v_o = \dot{\phi}_i + \frac{v_y x_i - v_x y_i}{r^2}.$$

In the previous method velocities were solved for even if they were not explicitly asked for. though this data was not asked for. Here the vehicle parameters will be inserted directly into (4.1). Multiplication by r^2 gives

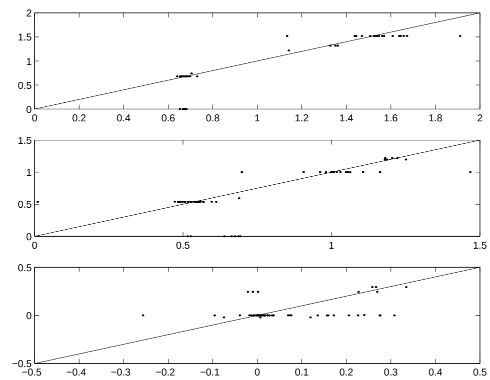

Figure 4.2: (l, x_0, y_0) Estimated vs "measured" parameters. Unknown values are set to 0. Measured parameters are only approximate.

$$r^2 v_o - r^2 \dot{\phi}_i - v_y x_i + v_x y_i = 0. \tag{4.3}$$

We know that

$$\begin{aligned} \begin{bmatrix} v_x \\ v_y \end{bmatrix} &= vR(\theta_0) \begin{bmatrix} -x_0 l \sin(s) \\ y_0 l \sin(s) - \cos(s) \end{bmatrix}, \\ v_o &= v \sin(s) l. \end{aligned}$$

Inserting this into (4.3) gives

$$\begin{aligned} &\dot{\phi} r^2 + k_v v \sin(s) X r \cos(\theta_0) \cos(phi) - k_v v Y \sin(s) r \sin(\theta_0) \cos(\phi) + \\ &k_v v Y \sin(s) r \cos(\theta_0) \sin(\phi) - \sin(s) k_v v l r^2 + \\ &k_v v \sin(s) X r \sin(\theta_0) \sin(\phi) + k_v v \cos(s) r \sin(\theta_0) \cos(\phi) \\ &\qquad - k_v v \cos(s) r \cos(\theta_0) \sin(\phi) = 0, \end{aligned}$$

where $(s, v, \phi, \dot{\phi}, r)$ are known and the remaining parts is a polynomial in $(k_v, l, X, Y, \cos(\theta_0), \sin(\theta_0))$. One such equation is generated for each time step. These equations are solved in least squares sense. That is the equations are squared and added to each other and the sum is minimized. Let F be the likelihood function, then

$$F(x) = \sum_i p_i(x) \sum_j B_i(o_i),$$

with $x = (k_v, l, X, Y, \cos(\theta_0), \sin(\theta_0))$ and $o_i = (s_i, v_i, \phi_i, \dot{\phi}_i, r_i)$. The $B_j(o_i)$ are found in the appendix.

4.3 Problems

The length scale in the estimate is given by the length measurements of the laser and systematic errors in the laser measurements give errors of scale in the estimated parameters. For this reason all estimates have been normalized before comparing to blueprint values. In the expression for the motion L and k_s appear as $\sin K_s s/L$ and as the steering angle usually is small it is impossible to distinguish between L and k_S and one of the values has to be set, we chose $k_s = 1$.

4.4 Possible applications

Except for the obvious application of parameter estimation, there is a possibility to use these ideas for surveillance of vehicle status. If there is an error in the sensors the error value between the motion calculated from the laser and the corresponding value from vehicle sensors will be large, indicating an error and hopefully the source of the error as well.
For parameter surveillance there are two main possibilities:

- Using the distances given by the laser to compute the motion. The advantage of this is the simplicity shown above.
- As this kind of system is run to find its own position on a map the vehicle will after the initiation phase, have a map of the surroundings and will then have precise information on estimated distances to different reflexes.

4.5 Conclusions

We have shown that it is not necessary to reconstruct the map to be able to compute the vehicle parameters. An important point is that position derivatives are easier to work with than the actual position and that, in some cases, the optimization can be made without the need for time-consuming iterative optimization routines

Chapter 5

Using Multilinear Constraints

Inspiration is needed in geometry, just as much as in poetry.
Pushkin, Aleksandr Sergeyevich (1799 - 1837)

In this chapter will be shown how to use the discrete and continuous multilinear constraints in hand-eye and vehicle calibration. It will first be demonstrated how the discrete multilinear constraint can be used for hand-eye calibration giving good results both with simulated and real data. Then the method with continuous multilinear constraints is demonstrated on the hand eye calibration problem and it is shown that this method has some problems related to the stability when applied to real data. The last part of the chapter shows how to apply the discrete and continuous constraints to solve for the unknown parameters of a vehicle equipped with a camera. As before it is assumed that only tracked points are dealt with. One possible point tracker is the KLT tracker[25].

5.1 Hand-Eye Calibration

A robot hand (gripper) is given for which the rotation R_H and the translation T_H with regard to the world coordinate system are known. On this hand a camera is rigidly mounted with translation T_{HE} and rotation R_{HE} from the hand in the hand coordinate system. The internal calibration of the camera is K, an upper triangular 3×3 matrix. By moving the robot hand and observing points in images the goal is to compute R_{HE}, T_{HE} and K. Camera motion is written

$$P(t) = KR_{HE}\begin{bmatrix} R_H(t) & -T_{HE} - R_H(t)T_H(t) \end{bmatrix}.$$

With $A = KR_{HE}$, this can be written

$$P(t) = A\begin{bmatrix} R_H(t) & -T_{HE} - R_H(t)T_H(t) \end{bmatrix}. \tag{5.1}$$

5.1.1 Hand Eye Calibration Using Discrete Multilinear Constraints

Reminding of equation (2.4) which is a reformulation of the first order discrete multilinear constraint in space :

$$\det\begin{bmatrix} P(t_0) & u(t_0) & \mathbf{0} \\ P(t_1) & \mathbf{0} & u(t_1) \end{bmatrix} = 0.$$

It is important to note that (2.4) gives a constraint for each pair of images of the same point and that constraints coming from different points may be used together without introducing new unknown variables.

Solving for camera rotation and internal calibration using translations

With a suitable choice of robot coordinate system, it can with pure translations assume that $R_H(t) = I$. Then (5.1) is simplified to

$$P(t) = A \begin{bmatrix} I & -T_{HE} - T_H(t) \end{bmatrix} . \tag{5.2}$$

Inserting this into (2.4) gives

$$\det \begin{bmatrix} A & A(-T_{HE} - T_H(t_i)) & u(t_i) & \mathbf{0} \\ A & A(-T_{HE} - T_H(t_j)) & \mathbf{0} & u(t_j) \end{bmatrix} = 0$$

which is equivalent to

$$\det \begin{bmatrix} A & A(-T_{HE} - T_H(t_i)) & u(t_i) & \mathbf{0} \\ \mathbf{0} & A(-T_H(t_j) + T_H(t_i)) & -u(t_i) & u(t_j) \end{bmatrix} = 0.$$

Substituting with $\Delta T_H = T_H(t_j) - T_H(t_i)$, the above is rewritten

$$\begin{aligned} \det A \det \begin{bmatrix} -A\Delta T_H & -u(t_i) & u(t_j) \end{bmatrix} &= 0 \\ &\Longleftrightarrow \\ -\det A \det \begin{bmatrix} -A\Delta T_H & u(t_i) & u(t_j) \end{bmatrix} &= 0 \\ &\Longleftrightarrow \\ \det \begin{bmatrix} A\Delta T_H & u(t_i) & u(t_j) \end{bmatrix} &= 0. \end{aligned}$$

since $\det(A) \neq 0$. Decomposing A into elements, this is written

$$\det \begin{bmatrix} \begin{bmatrix} a_{11}\, a_{12}\, a_{13} \\ a_{21}\, a_{22}\, a_{23} \\ a_{31}\, a_{32}\, a_{33} \end{bmatrix} \Delta T_H & u(t_i) & u(t_j) \end{bmatrix} = 0 \tag{5.3}$$

$$\Longleftrightarrow$$

$$\sum_{k=1,l=1}^{3,3} a_{kl} \det \begin{bmatrix} I(i,j)T_H & u(t_i) & u(t_j) \end{bmatrix} = 0 \tag{5.4}$$

where $I(i,j)$ is a 3×3 matrix with a 1 in place (i,j) and zeros elsewhere. Let $B(t_i,t_j)$ and $\tilde{A}$ be column vectors with elements defined by

$$\begin{aligned} B_{3(k-1)+l}(t_i,t_j) &= \det \begin{bmatrix} I(k,l)T_H & u(t_i) & u(t_j) \end{bmatrix} \\ \tilde{A}_{3(k-1)+l} &= a_{kl}. \end{aligned}$$

Then (5.4) can be rewritten

$$B(t_i,t_j)^T\tilde{A}=0$$

implying that

$$\tilde{A}^T(B(t_i,t_j)^T B(t_i,t_j))\tilde{A}=0.$$

The above has to be fulfilled for all pairs of measurements (t_i,t_j) and summing the squares of these equations we get

$$\begin{aligned}&\sum_{i,j}\tilde{A}^T(B(t_i,t_j)^T B(t_i,t_j))\tilde{A}=0\\ &\qquad\qquad\qquad\qquad\qquad\Longleftrightarrow\\ &\tilde{A}^T\underbrace{(\sum_{i,j}B(t_i,t_j)^T B(t_i,t_j))}_{M}\tilde{A}=0.\end{aligned} \tag{5.5}$$

As A has an arbitrary scale and $\det(A)\neq 0$ it can be assumed that $\|A\|_F=1$ where $\|A\|_F=\|\tilde{A}\|_2=\sqrt{\sum_{k,l}A_{kl}^2}$. A solution to this minimization problem is given by a singular value decomposition of M [11] as M is a symmetric matrix.

Computing the translation of a camera with known orientation and internal parameters

From the previous step, the matrix A is known. By a change of the image coordinate system, without loss of generality it can be assumed that $A=I$. The camera matrix $P(t)$ of size (3×4) can be written

$$P(t)=\begin{bmatrix}R_H(t) & -T_{HE}-R_H(t)T_H(t)\end{bmatrix}$$

and the constraint (2.4) is simplified to

$$\begin{aligned}\det\begin{bmatrix}R_H(t_i) & -T_{HE}-R_H(t_i)T_H(t_i) & u(t_i) & \mathbf{0}\\ R_H(t_j) & -T_{HE}-R_H(t_j)T_H(t_j) & \mathbf{0} & u(t_j)\end{bmatrix}&=0\\ &\Longleftrightarrow\\ C(t_i,t_j)^T T_{HE}+D(t_i,t_j)&=0,\end{aligned} \tag{5.6}$$

where

$$D(t_i,t_j) \;=\; \det\begin{bmatrix}R_H(t_i) & R_H(t_i)T_H(t_i) & u(t_i) & \mathbf{0}\\ R_H(t_j) & R_H(t_j)T_H(t_j) & \mathbf{0} & u(t_j)\end{bmatrix}$$

is a scalar and $C(t_i,t_j)$ is a column matrix with elements

$$c_k(t_i,t_j) \;=\; \det\begin{bmatrix}R_H(t_i) & I(k) & u(t_i) & \mathbf{0}\\ R_H(t_j) & I(k) & \mathbf{0} & u(t_j)\end{bmatrix}.$$

Here $I(i)$ is column i of the 3×3 identity matrix. Equation (5.6) implies that

$$\begin{aligned} (C(t_i,t_j)^T T_{HE} + D(t_i,t_j))^2 &= 0 \\ &\Longleftrightarrow \\ D^2(t_i,t_j) + 2D(t_i,t_j)C(t_i,t_j)^T T_{HE} & \\ +T_{HE}^T C(t_i,t_j)C(t_i,t_j)^T T_{HE} &= 0. \end{aligned}$$

This has to be fulfilled for all pairs of measurements, giving

$$\begin{aligned} \sum_{i,j}(D^2(t_i,t_j)) + 2D(t_i,t_j)C(t_i,t_j)^T T_{HE} & \\ +T_{HE}^T C(t_i,t_j)C(t_i,t_j)^T T_{HE} = 0 & \\ \Longleftrightarrow & \\ (\sum_{i,j} D^2(t_i,t_j)) + 2(\sum_{i,j} D(t_i,t_j)C(t_i,t_j)^T)T_{HE} & \\ +T_{HE}^T(\sum_{i,j} C(t_i,t_j)C(t_i,t_j)^T)T_{HE} = 0. & \end{aligned}$$

This is a second order polynomial for which the minimizing T_{HE} is the least squares solution to (5.6) for all observed image pairs.

5.1.2 Hand Eye Calibration Using Continuous Multilinear Constraints

Study the continuous multilinear constraint (2.11),

$$\det \begin{bmatrix} P & u & \mathbf{0} \\ P' & u' & u \end{bmatrix} = 0.$$

In the robot coordinate system the camera matrix at time t is

$$P(t) = A \begin{bmatrix} I & -T_{HE} \end{bmatrix} \begin{bmatrix} R_H(t) & -R_H(t)T_H(t) \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5.7}$$

By changing coordinate system so that

$$\begin{cases} \mathbb{R}_H(0) = I \\ T_H(0) = 0 \end{cases},$$

we get

$$\begin{aligned} P'(t) &= A \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \begin{bmatrix} R'_H(t) & -R'_H(t)T_H(t) - R'_H(t)T_H(t) \\ mnoll & 1 \end{bmatrix} && (5.8) \\ &= A \begin{bmatrix} R'_H(t) & -R'_H(t)T_H(t) - R'_H(t)T_H(t) \end{bmatrix}. && (5.9) \end{aligned}$$

Evaluating (5.7) and (5.9) at $\mathbf{0}$ gives

$$\begin{aligned}
P(0) &= A\left[I - T_{HE}\right] \\
P'(0) &= A\left[R_H'(0) \quad -R_H'(0)T_H(0) - R_H(0)T_H(0)'\right] \\
&= A\left[R_H'(0) \quad -T_H(0)'\right].
\end{aligned}$$

Inserting this into (2.11) we get

$$\det\begin{bmatrix} A & -AT_{HE} & u & \mathbf{0} \\ AR_H' & -AT_H' & u' & u \end{bmatrix} = 0. \tag{5.10}$$

This expression will be used as starting point for both steps in a two-step method.

Internal calibration as well as camera rotation from pure translations

In the case of pure translations, that is $R_H' = \mathbf{0}$, (5.10) gives

$$\det\begin{bmatrix} A & -AT_{HE} & u & \mathbf{0} \\ \mathbf{0} & -AT_H' & u' & u \end{bmatrix} = 0.$$

Using $\det(A) \neq 0$ this is equivalent to

$$\det\begin{bmatrix} AT_H' & u' & u \end{bmatrix} = 0. \tag{5.11}$$

Substituting

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

and

$$T_H = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

into (5.11) gives

$$\begin{aligned}
0 = {} & -a_1u_2't_1' - a_2u_2't_2' - a_3u_2't_3' + a_4u_1't_1' \\
& +a_5u_1't_2' + a_6u_1't_3' + a_7(t_1'u_1u_2' - t_1'u_1'u_2) \\
& +a_8(t_2'u_1u_2' - t_2'u_1'u_2) + a_9(t_3'u_1u_2' - t_3'u_1'u_2).
\end{aligned}$$

The constraint is linear in A and can be written

$$0 = \begin{bmatrix} -u_2' t_1' \\ -u_2' t_2' \\ -u_2' t_3' \\ u_1' t_1' \\ u_1' t_2' \\ u_1' t_3' \\ t_1'(u_1 u_2' - u_1' u_2) \\ t_2'(u_1 u_2' - u_1' u_2) \\ t_3'(u_1 u_2' - u_1' u_2) \end{bmatrix}^T \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{bmatrix}.$$

Using the notation

$$\tilde{A} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 \end{bmatrix}^T$$

we get

$$0 = \underbrace{\begin{bmatrix} -u_2' T_H' \\ u_1' T_H' \\ (u_1 u_2' - u_1' u_2) T_H' \end{bmatrix}^T}_{C(o_i)} \tilde{A} \tag{5.12}$$

This is the same expression as (5.4) when using the discrete constraint with the difference that the distance between observations is infinitesimal in this case.
The camera orientation and calibration matrix A is computed by singular value decomposition.

Computing the translation of a camera with known orientation and internal parameters

The camera matrix A is known from the previous step. By change of image coordinate system it can without loss of generality be assumed that $A = I$. The constraint (5.10) then becomes

$$\det \begin{bmatrix} I & -T_{HE} & u & \mathbf{0} \\ R_H' & -T_H' & u' & u \end{bmatrix} = 0. \tag{5.13}$$

With

$$R_H' = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix},$$

$$T_{HE} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix},$$

the constraint (5.13) becomes

$$\det \begin{bmatrix} 1 & 0 & 0 & -\tau_1 & u_1 & 0 \\ 0 & 1 & 0 & -\tau_2 & u_2 & 0 \\ 0 & 0 & 1 & -\tau_3 & 1 & 0 \\ 0 & -r_3 & r_2 & -t_1' & u_1' & u_1 \\ r_3 & 0 & -r_1 & -t_2' & u_2' & u_2 \\ -r_2 & r_1 & 0 & -t_3' & 0 & 1 \end{bmatrix} = 0$$

$$\Longleftrightarrow$$

$$C_{\tau 1}\tau_1 + C_{\tau 2}\tau_2 + C_{\tau 3}\tau_3 + m_0 = 0 \tag{5.14}$$

$$\Longleftrightarrow$$

$$\begin{bmatrix} C_{\tau 1} & C_{\tau 2} & C_{\tau 3} \end{bmatrix} T_{HE} + m_0 = 0, \tag{5.15}$$

where

$$\begin{aligned} C_{\tau 1} = {} & r_3^2 u_2 + r_3 u_1' - r_3 r_2 - r_2^2 u_2 + r_2 u_2^2 r_3 \\ & + r_2 u_1' u_2 - r_2 u_1 r_1 - r_2 u_1 u_2' + r_3 u_1 u_2 r_1, \\ C_{\tau 2} = {} & -r_3 u_2 u_1 r_2 - r_3^2 u_1 + r_3 u_2' + r_3 r_1 + r_1^2 u_1 \\ & - r_1 u_1^2 r_3 + r_1 u_1 u_2' + r_1 u_2 r_2 - r_1 u_1' u_2, \\ C_{\tau 3} = {} & -r_2 u_2' - r_1 u_1' - r_2 u_2^2 r_1 + r_2^2 u_2 u_1 + r_2 u_1 r_3 \\ & - r_1^2 u_1 u_2 + r_1 u_1^2 r_2 - r_1 u_2 r_3, \\ m_0 = {} & t_1'(u_2' + r_1 - u_1 r_3 + u_2^2 r_1 - u_2 u_1 r_2) \\ & - t_2'(u_1' - r_2 + u_2 r_3 - u_1^2 r_2 + u_1 u_2 r_1) \\ & + t_3'(u_1^2 r_3 - u_1 r_1 + u_2^2 r_3 + u_1' u_2 - u_2 r_2 - u_1 u_2'). \end{aligned}$$

Equation (5.15)is linear in T_{HE} and has to be fulfilled for all observed points. $T_H E$ is computed by solving this system in least squares sense.

5.1.3 Reprojection Based method

In the situation of only one tracked point the setup is very similar to a special case of [28]. If the robot hand is only translated it is possible to compute A using each of the above proposed methods. With a known camera performing known relative movements a point in space corresponding to the tracked image points can be computed using intersection.
Assuming a value for T_{HE} it is possible to compute camera movement even in the case when there are rotations in the hand movements. With a known camera position and a known point in space it is possible to compute a synthetic image point of where the point in space would have been projected.
The computed image point is compared with the tracked image point and an error function is computed by summation of this error over all images. With this error function it is possible to find T_{HE} by conventional optimization. This alternative approach can

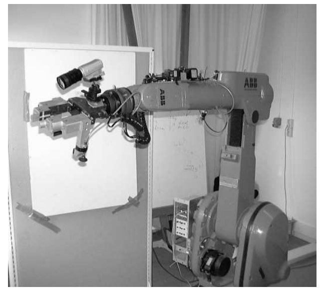

Figure 5.1: The robot used in the real experiments.

be used to compute T_{HE} and will be used for verification as the laboratory experiments were made using only one point.

5.1.4 Experimental validation

For simulated data both the discrete and the continuous multilinear constraint work well for real data, even if it is necessary to keep small enough step-lengths in the continuous case.
A modified ABB IRB2003 robot, capable of moving with 6 degrees of freedom, has been used to test the methods. The camera was mounted using a ball head camera holder on the hand (gripper) of the robot so that the orientation of the camera could be changed with respect to the orientation of the robot hand coordinate system, cf. Figure 5.1. It is difficult to test the values computed from real data. Visual inspection of the robot tells us that the answer we get is realistic; it also gives a small reprojection error if a point is computed by intersection and then reprojected. All tests have been made by tracking a single point as this was easier to handle.

The step computing a is identical for both the discrete and the continuous formulation even if it is only proven for small step lengths in the continuous case. Using the "Camera Calibration Tool-box for Matlab" by Bouguet [5] implementing the calibration method of [29] it is possible to get an estimate for the camera parameters using a calibration plane

$$\begin{aligned} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} &= \begin{bmatrix} 1001.3 \pm 5.8 \\ 1041.3 \pm 6.0 \end{bmatrix} \\ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} &= \begin{bmatrix} 317.8 \pm 7.3 \\ 240.5 \pm 6.6 \end{bmatrix} . \end{aligned}$$

Tracking one point in 37 observations and using rather long steps and all possible pairs we get camera internal calibration matrix

$$K = \begin{bmatrix} 1000.6 & -0.1 & 326.0 \\ 0 & 1036.6 & 232.1 \\ 0 & 0 & 1 \end{bmatrix}.$$

This is close to the values from the Calibration Tool-box.

When estimating T_{HE} the discrete and the continuous methods are different. Since the ground truth for T_{HE} is not available we are restricted to looking at the variability of the estimated values as well as comparing with the reprojection based method. Unfortunately the experiments were made on different days with different camera-settings.

Estimating T_{HE} using discrete multilinear constraints

Using a sequence with 72 camera positions and images a value for T_{HE} was calculated. An attempt to validate the method was made by selecting 30 random observations and computing T_{HE}, this was repeated 100 times. Mean value and variance of T_{HE} are computed

$$\begin{aligned} \text{mean}(T_{HE}) &= \begin{bmatrix} -103.8 \\ -117.3 \\ -221.9 \end{bmatrix} \\ \sigma(T_{HE}) &= \begin{bmatrix} 4.6 \\ 6.8 \\ 4.1 \end{bmatrix}. \end{aligned}$$

using the reprojection method T_{HE} was calculated to

$$T_{HE} = \begin{bmatrix} -101.6 \\ -106.6 \\ -221.2 \end{bmatrix}.$$

Matlab code for this implementation can be found on the author's home-page [26].

Estimating T_{HE} using continuous multilinear constraints

It is difficult to say if the estimated values for T_{HE} are correct or not. Hand translations and rotations were generated using the following matlab code

```
R   = randn(1,3)/R_div ;
Rp  = eye(3)+ [ 0 -R(3) R(2) ;
                R(3) 0 -R(1) ;
               -R(2) R(1) 0 ];
Thp = T_mult*randn(3,1);
```

Sequences 1 to 4 each containing 100 camera positions generated using

Sequence	1	2	3	4
R_{div}	100	200	400	400
T_{mult}	30	30	30	5

the fifth sequence is the sum of the other sequences. Using the A computed above and the method based on the continuous constraints the following results were computed for T_{HE}

1	2	3	4	5
-98.9	-102.4	-75.6	-95.6	-93.7
-98.0	-128.3	-124.80	-113.5	-107.1
-262.3	-265.0	-218.5	-262.0	-261.4

after refining the above values with the reprojection based method the following values were computed

1	2	3	4	5
-98.3	-118.9	-103.2	-69.1	-103.2
-104.4	-101.9	-100.7	-113.1	-104.4
-254.3	-254.1	-219.1	-233.8	-252.0

Visual inspection of the camera mount suggest that the values computed are plausible but it is hard to tell which one is correct.
These numbers show that the method using the continuous constraints has problems to estimate T_{HE} in a reliable way and should not be used when it is possible to use the discrete multilinear constraints.

Speed

These costs are computed assuming that we have a set of tracked points and corresponding robot movements $R_H(t)$ and $T_H(t)$. Tracked points can be obtained by using the KLT-tracker [25]. Estimates for flops are based on the numbers from Matlab. For the first step both methods are identical in time consumption. For the first step of finding the internal camera parameters as well camera orientation we have

- 330 flops per point pair for calculating M in (5.5)

- Finding the camera from M requires 15795flops divided into 15232 flops for the SVD and 554 flops for RQ-factoring.

The second stage is not identical for the two methods but requires approximately 500-700 flops per observed point pair. This means that for 100 points in each step arranged so that all pairs can be used, we need roughly $5Mflops$. A typical CPU of 2003 has performance in gigaflops.

5.2 Vehicle Calibration with known Camera

Reminding of (2.11)

$$\det \begin{bmatrix} P & u & \mathbf{0} \\ P' & u' & u \end{bmatrix} = 0. \tag{5.16}$$

With a known camera it can be assumed that $A = I$ and by choosing a coordinate system centered at the camera at time t , equation (5.16) is rewritten

$$\det \begin{bmatrix} I & \mathbf{0} & u & \mathbf{0} \\ R' & T' & u' & u \end{bmatrix} = 0. \tag{5.17}$$

The equations for the motion derivatives (2.13), (2.22) and (2.14) can after adding a third coordinate be written

$$\begin{aligned} T' &= \begin{bmatrix} 0 \\ -Xv\sin(s) \\ v(Y\sin(s) - \cos(s)) \end{bmatrix}, \\ R' &= \dot{\theta} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \\ \dot{\theta} &= v\sin(s)l \end{aligned}$$

where $o_i = (v_i, s_i, u_i, u_i')$ is measured odometry and image information and $x = (l, X, Y)$ vehicle parameters. Equation (5.17) is then written

$$\det \begin{bmatrix} 1 & 0 & 0 & 0 & u_1 & 0 \\ 0 & 1 & 0 & 0 & u_2 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dot{u_1} & u_1 \\ 0 & 0 & \dot{\theta} & -Xv\sin(s) & \dot{u_2} & u_2 \\ 0 & -\dot{\theta} & 0 & v(Y\sin(s) - \cos(s)) & 0 & 1 \end{bmatrix} = 0.$$

This equation is a polynomial in (l, X, Y) and corresponding coefficients are found in (A.9). This equation holds for all observations and a least squares solution leads to the problem of minimizing the polynomial

$$\sum_i (p_i(x) \sum_j A_i(o_j)),$$

where the $A_i(o_j)$ are found in (A.8).

Optimization Step

In the 3D case with fixed camera $F(l, X, Y)$ is a second order polynomial X and Y with coefficients being second order polynomials in l. The function F is written

$$F(l, X, Y) = \begin{bmatrix} X & Y \end{bmatrix} \alpha(l) \begin{bmatrix} X \\ Y \end{bmatrix} + \beta(l) \begin{bmatrix} X \\ Y \end{bmatrix} + \gamma(l)$$

minimizing this over (X, Y) and assuming $\det(\alpha) \neq 0$ we get

$$\min_{(X,Y)} F(l, X, Y) = -\frac{1}{4}\beta(l)'\alpha(l)^{-1} + \gamma(l)$$

which as α is a 2×2 matrix is a rational function of order 6 in the nominator and order 4 in the denominator. This reduces the search to one-dimensional optimization.

Numerical Experiments

In the absence of real data, the method has been tested on simulated data. In all the examples below a the parameters of a simulated vehicle has been set to $(l, x_0, y_0) = (1, 2, 0)$ and for each number of samples the process of estimation has been repeated 100 times and estimates, standard deviation and mean error are calculated and plotted.

Bias of the results

In figure 5.7 there is a clear and visible bias, this bias comes from the fact that the derivatives are calculated by subtraction and this often gives an error in the derivatives. If Δt is decreased the bias will decrease but unfortunately this also affects the stability when calculating the image derivatives. Our choice stands between on the one hand stability and reasonably fast convergence and on the other hand the elimination of bias. Also, it is important to point out that the different parts of the parameter set are more or less easy to estimate. The parameter l is linked to rotation and is easier to estimate than x_0 and y_0 which are linked to minor variations in the translation velocity. When computing a derivative in the image this has to be done by numerical methods e.g.

$$f'(t) = \frac{f(t + \epsilon/2) - f(t - \epsilon/2)}{\epsilon}$$

and with a small ϵ there will be a strong amplification of the errors in the image data. With a large ϵ there will instead be an error in estimating the derivative that will give a biased result when computing (x_0, y_0).
To eliminate the bias, the Richardson extrapolation scheme was used with promising results. That is, the vehicle parameters are computed for some different values of Δt and a polynomial is fitted to the results. The estimated polynomial is evaluated at 0. Second order polynomials give good results.

Computation time

The theoretical runtime is very low. Provided that tracking and derivation are already accounted for, each additional sample adds about 200 flops. This should give a throughput of millions of samples on computers from 2003. In situations without noise in the signals, an estimate for the parameters can be computed with a very low number of points. There are two ways to get increased precision for the parameter estimates, the first is to run longer series in the generation of polynomials and the second is to repeat the estimation process several times and then calculate an average.

5.3 Vehicle Calibration with Unknown Camera

An important restriction in the method mentioned above is the inability to deal with cameras of unknown orientation. Being able to to freely translate the camera the methods described for hand-eye calibration would easily find the orientation and internal calibration of the camera. Unfortunately, vehicle movement is quite restricted and to avoid too complex polynomials only calibrated cameras of unknown direction are considered.
The method is a two step method where the first step determines one axis of the camera and the second step determines vehicle parameters as well as the rotation of the camera around it's already known axis. The reason for using a two-step method is to limit the total number of coefficients as well as the total degree of the polynomial.

Step 1, Pure translation

Motion is restricted to pure translation by setting $s = 0$. This setting is very similar to the first step of the hand-eye calibration but unfortunately only one direction of translation is possible. With this information the continuous multilinear constraint equation (2.11) is rewritten

$$\det \begin{bmatrix} A & \mathbf{0} & \mathbf{u} & \mathbf{0} \\ \mathbf{0} & A \begin{bmatrix} 0 \\ 0 \\ -v \end{bmatrix} & \dot{\mathbf{u}} & \mathbf{u} \end{bmatrix} = 0.$$

The camera matrix A is decomposed as

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix},$$

which gives

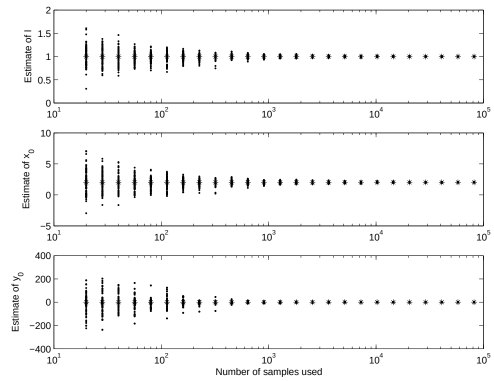

Figure 5.2: 100 estimates computed for each number of samples.

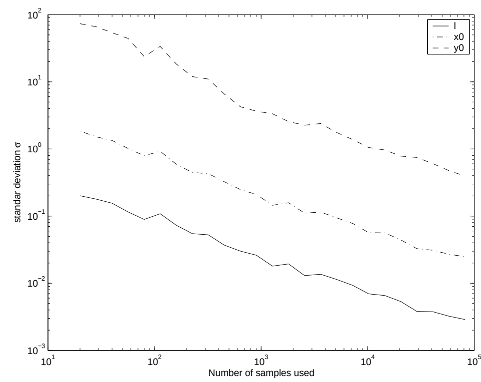

Figure 5.3: Standard deviation for different number of samples used in estimate.

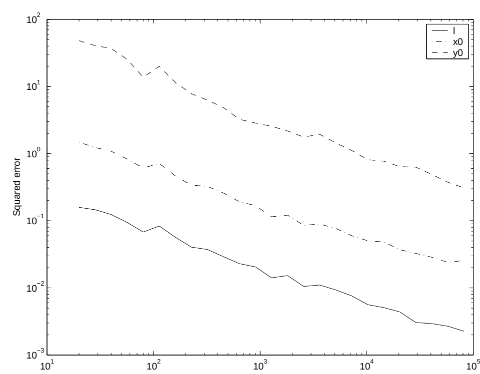

Figure 5.4: Error from known values against number of samples.

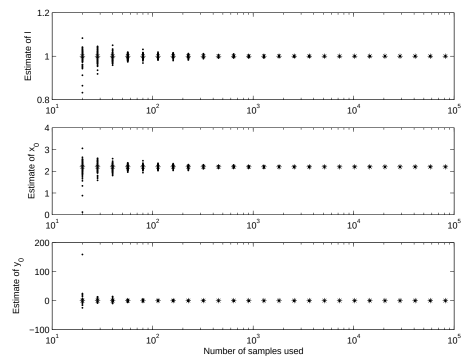

Figure 5.5: 100 estimates computed for each number of samples, high Δt.

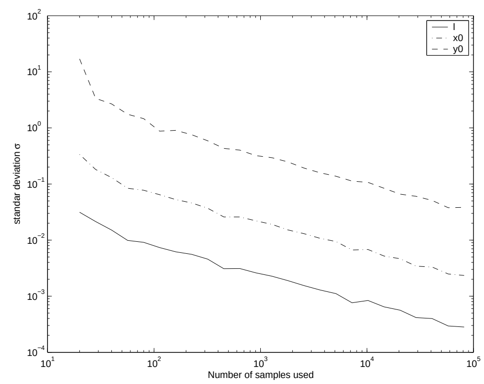

Figure 5.6: Standard deviation for different number of samples used in estimate, high Δt.

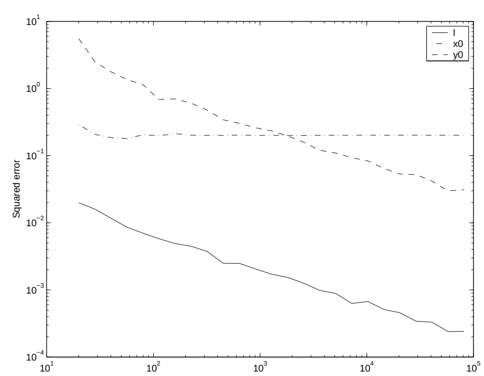

Figure 5.7: Error from known values against number of samples, high Δt.

$$
\begin{aligned}
0 &= \det \begin{bmatrix} A & \mathbf{0} & \mathbf{u} & \mathbf{0} \\ \mathbf{0} & A \begin{bmatrix} 0 \\ 0 \\ -v \end{bmatrix} & \dot{\mathbf{u}} & \mathbf{u} \end{bmatrix} \\
&= \begin{bmatrix} a_3 & u'_1 & u_1 \\ a_6 & u'_2 & u_2 \\ a_9 & 0 & 1 \end{bmatrix} \det(A).
\end{aligned}
$$

Since $\det(A) \neq 0$ this is equivalent to

$$
\begin{aligned}
0 &= \begin{bmatrix} a_3 & u'_1 & u_1 \\ a_6 & u'_2 & u_2 \\ a_9 & 0 & 1 \end{bmatrix} \\
&= ((a_3 - a_9 u_1) u'_2 - (a_6 - a_9 u_2) u'_1 \\
&= \begin{bmatrix} u'_2 & -u'_1 & -u_1 u'_2 + u'_1 u_2 \end{bmatrix} \begin{bmatrix} a_3 \\ a_6 \\ a_9 \end{bmatrix}.
\end{aligned}
$$

This equation is fulfilled for all times and measurements and is solved for available data in a least squares sense. This gives $A_3 = (a_3, a_6, a_6)^T$ up to scale. With a calibrated camera $||A_3|| = 1$ and A_3 can be calculated up to sign. As seen in 5.1.1 the constraint on translative motion can also be formulated with discrete multilinear constraints and this eliminates the problem of step-related bias.

Step 2 , free movement

Assuming internal calibration of the camera to be known and A_3 computed up to sign. As $A^T A = 1$ and $\det(A) = 1$ there is only one degree of freedom left for A. Introducing

$$
B = \begin{bmatrix} \frac{-a_3 * a_9}{\alpha} & \frac{a_6}{\alpha} & a_3 \\ \frac{-a_6 * a_9}{\alpha} & \frac{-a_3}{\alpha} & a_6 \\ \alpha & 0 & a_9 \end{bmatrix}
$$

with

$$
\alpha = \sqrt{a_3^2 + a_6^2}
$$

and

$$
C(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

All possible A can now be written

$$\begin{aligned} A(\phi) &= BC(\phi) \\ &= \begin{bmatrix} -\frac{a_3 a_9 \cos(\phi)+a_6\sin(\phi)}{\alpha} & \frac{-a_3 a_9\sin(\phi)+a_6\cos(\phi)}{\alpha} & a_3 \\ \frac{-a_6 a_9\cos(\phi)+a_3\sin(\phi)}{\alpha} & -\frac{a_6 a_9\sin(\phi)+a_3\cos(\phi)}{\alpha} & a_6 \\ \alpha\cos(\phi) & \alpha\sin(\phi) & a_9 \end{bmatrix}. \end{aligned}$$

Using this A the continuous multilinear constraint is computed

$$\det(M) = \sum_{i=1}^{11} c_i q_i(\cos\phi, \sin\phi, l, X, Y),$$

where the $q_i(\cos\phi, \sin\phi, l, X, Y)$ are monomials. Only 11 of the c_i are non-zero. As the equations will be solved in a least squares sense we compute

$$(\det(M)^2) = \sum_{i=1}^{54} C_i p_i(\cos\phi, \sin\phi, l, X, Y),$$

where $p_i(\cos\phi, \sin\phi, l, X, Y)$ are monomials and only 54 of the A_i are non zero. $(\det(M))^2$ is a polynomial of total degree less than 8 and

- Degree of trigonometric components is up to 4
- Degree of l is up to 2
- Degree of (X, Y) is up to 2

These coefficients can be added for several observations and used to calculate a least squares solution for our parameters given our observations.
The coefficients are given in (A.3).

Numerical experiments

Due to lack of a suitable robot with a camera no real data has been available to test these methods. All tests have therefore been done on simulated data.
These numerical experiments have provided an opportunity to get the polynomial coefficients right and get some some information on what behavior that can be expected.

Step 1

This step is quite stable and it tolerates long time-steps. When dealing with real data a probable source of error is if the vehicle fails to travel along a straight line.

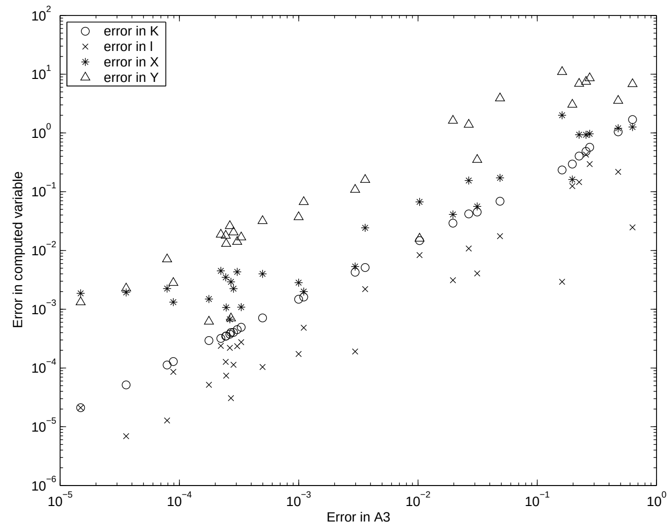

Figure 5.8: Dependence on step 1 in step 2 calculations. Notice that the error decreases quickly with a good estimate for A_3. It is also important to note that the estimate for l is good even for quite bad estimates for A_3.

Step 2

Step 2 is unfortunately dependent on the results from step 1 and errors from the first step tend to propagate into step 2. Figure (5.8) shows the dependence on A_3 when estimating K and l. In figure (5.8) it is also visible that the estimate for x behaves in a good way while the estimate for y is less reliable, it is also clear that there is a limit on the performance even for good estimates for A_3. This error is a bias error and is shown in figures (5.9) and (5.10). The bias errors come from the fact that the continuous constraints assume derivatives to be calculated with infinitesimal time steps.

Aspects on computational time

The limiting factor will be the point-tracker and not these parts of the code.

5.4 Degenerate case

It is possible to get a situation of poor numerical stability. Equation 2.11 :

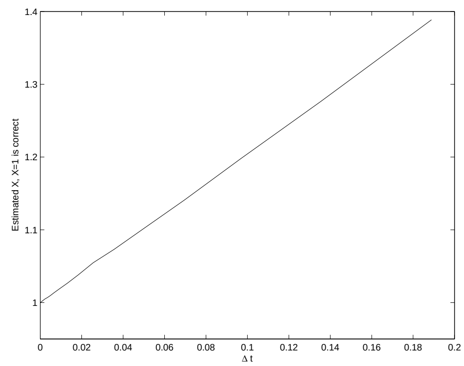

Figure 5.9: Bias in X vs Δt.

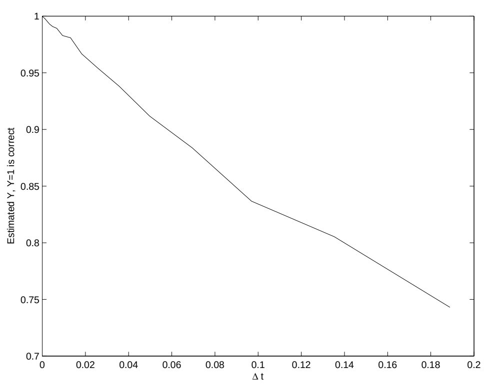

Figure 5.10: Bias in Y vs Δt.

$$\det\begin{bmatrix} A & \mathbf{0} & u & \mathbf{0} \\ AR' & AT' & u' & u \end{bmatrix} = 0$$
$$\Leftrightarrow$$
$$\det\begin{bmatrix} I & \mathbf{0} & A^{-1}u & \mathbf{0} \\ R' & T' & A^{-1}u' & A^{-1}u \end{bmatrix} = 0 \tag{5.18}$$

can with (2.14)

$$T' = v\begin{bmatrix} 0 \\ -X\sin(s) \\ Y\sin(s) - \cos(s) \end{bmatrix}$$

be simplified to

$$(cos(s) + Y sin(s))C + X sin(s)D = 0 \tag{5.19}$$

where both C and D are independent of (X, Y). Let

$$f(X,Y) = (cos(s) + Y sin(s))C + X sin(s)D = 0$$

then

$$\begin{aligned} \frac{df}{dX} &= sin(s)D, \\ \frac{df}{dY} &= sin(s)C. \end{aligned} \tag{5.20}$$

By solving for C in (5.19) the above can be rewritten

$$\begin{aligned} \frac{df}{dY} &= \frac{-sin(s)sin(s)XD}{-cos(s) + Y sin(s)} \\ &\approx sin^2(s)XD \end{aligned}$$

as $cos(s) - Y sin(s) \approx 1$. It is now possible to compare df/dX and df/dY without computing C and D

$$\frac{df/dY}{df/dX} \approx -sin(s)X.$$

Therefore the solution to $\min_{(X,Y)} \sum_i f_i^2$ is very sensitive in the Y direction, especially when $X = 0$. This degeneracy effects both the case with known direction and the case with unknown direction.

Chapter 6

Conclusions

In this thesis it has been shown how to perform hand-eye calibration as well as how to find the direction and location of a camera on a mobile robot. The proposed methods are based on collecting a large amount of data and computing the most probable relative camera position and camera orientation and calibration. For situations involving a 2D camera, the condition used has been derived from the first order multilinear constraint while a simpler geometric relation has been used for the laser-scanner.

The demand that large amounts of data must be handled has led to simplifications in how to extract information from image and odometry data and used to calculate polynomial coefficients representing the data.

The number of non-zero coefficients of these polynomials is limited and not dependent on the amount of data used in the computation of these polynomials. For the hand-eye calibration problem the resulting polynomial is quadratic while the polynomials for the mobile camera calibration leads to polynomials of a higher degree for which the optima has to be found numerically.

For the hand-eye calibration and the calibration of a 1D camera on an autonomous robot the method has been tested on real data.

Appendix A

Appendix

> Geometry enlightens the intellect and sets one's mind right. All of its proofs are very clear and orderly. It is hardly possible for errors to enter into geometrical reasoning, because it is well arranged and orderly. Thus, the mind that constantly applies itself to geometry is not likely to fall into error. In this convenient way, the person who knows geometry acquires intelligence.
> *Ibn Khaldun (1332-1406) The Muqaddimah.*

All these sets of coefficients have been computed with the aid of symbolic algebra software and the code is in some cases written by scripts.

A.1 Coefficients for laser scanner

The variables are $x = (l, X, Y, \cos(\theta_0), , \sin(\theta_0))$. We simplify

$$\begin{aligned} C &= \cos(\theta_0), && \text{(A.1)} \\ S &= \sin(\theta_0). && \text{(A.2)} \end{aligned}$$

. The coefficients for the squares constraint can be written as

$$\begin{aligned} f(x, o_i) = \; & B_{lK}lK + B_{lS}lS + B_{lC}lC + B_{lYS}lYS + B_{lYC}lYC && \text{(A.3)} \\ & + B_{lXS}lXS + B_{lXC}lXC + B_{KK}K^2 + B_{KS}KS \\ & + B_{KC}KC + B_{KYS}KYS + B_{KYC}KYC + B_{KXS}KXS \\ & + B_{KXC}KXC + B_{SS}S^2 + B_{SC}SC + B_{YSS}YS^2 \\ & + B_{YSC}YSC + B_{XSS}XS^2 + B_{XSC}XSC \\ & + B_{CC}C^2 + B_{YCC}YC^2 + B_{XCC}XC^2 \\ & + B_{YYSS}Y^2S^2 + B_{YYSC}Y^2SC + B_{YYCC}Y^2C^2 \\ & + B_{XXSS}X^2S^2 + B_{XXSC}X^2SC + B_{XXCC}X^2C^2 \\ & + B_{XYSS}XYS^2 + B_{XYSC}XYSC + B_{ll}l^2 + \\ & + B_{XYCC}XYC^2, \end{aligned}$$

where

$$S^2 + C^2 = 1. \quad \text{(A.4)}$$

With

$$
\begin{aligned}
B_{ll} &= A_l A_l, & B_{YSS} &= 2A_S A_{YS},\\
B_{lK} &= 2A_l A_K, & B_{YSC} &= 2(A_S A_{YC} + A_C A_{YS}),\\
B_{lS} &= 2A_l A_S, & B_{XSS} &= 2A_S A_{XS},\\
B_{lC} &= 2A_l A_C, & B_{XSC} &= 2(A_S A_{XC} + A_C A_{XS}),\\
B_{lYS} &= 2A_l A_{YS}, & B_{CC} &= A_C A_C,\\
B_{lYC} &= 2A_l A_{YC}, & B_{YCC} &= 2A_C A_{YC},\\
B_{lXS} &= 2A_l A_{XS}, & B_{XCC} &= 2A_C A_{XC},\\
B_{lXC} &= 2A_l A_{XC}, & B_{YYSS} &= A_{YS} A_{YS},\\
B_{KK} &= A_K A_K, & B_{YYSC} &= 2A_{YS} A_{YC},\\
B_{KS} &= 2A_K A_S, & B_{YYCC} &= A_{YC} A_{YC},\\
B_{KC} &= 2A_K A_C, & B_{XXSS} &= A_{XS} A_{XS},\\
B_{KYS} &= 2A_K A_{YS}, & B_{XXSC} &= 2A_{XS} A_{XC},\\
B_{KYC} &= 2A_K A_{YC}, & B_{XXCC} &= A_{XC} A_{XC},\\
B_{KXS} &= 2A_K A_{XS}, & B_{XYSS} &= 2A_{XS} A_{YS},\\
B_{KXC} &= 2A_K A_{XC}, & B_{XYSC} &= 2(A_{XS} A_{YC} + A_{YS} A_{XC}),\\
B_{SS} &= A_S A_S, & B_{XYCC} &= 2A_{XC} A_{YC},\\
B_{SC} &= 2A_S A_C, & &
\end{aligned}
\tag{A.5}
$$

where the unsquared coefficients are

$$
\begin{aligned}
A_l &= -vr\sin(s), & A_{XS} &= v\sin(s)\sin(\phi),\\
A_K &= r\dot{\phi}, & A_{XC} &= v\sin(s)\cos(\phi),\\
A_{YS} &= -v\sin(s)\cos(\phi), & A_S &= v\cos(s)\cos(\phi),\\
A_{YC} &= v\sin(s)\sin(\phi), & A_C &= -v\cos(s)\sin(\phi).
\end{aligned}
\tag{A.6}
$$

A.2 Coefficients in 3D, non-rotational case

The coefficients with a calibrated camera with known direction. $x = (l, X, Y)$.

$$\begin{aligned} f(x, o_i) &= B_{l0XX}X^2 + B_{l0YY}Y^2 + B_{l0XY}XY + B_{l0X}X + B_{l0Y}Y \\ &+B_{l0} + B_{l1XX}lX^2 + B_{l1YY}lY^2 + B_{l1X}lX + B_{l1Y}lY \\ &+B_{l1}l + B_{l2XX}l^2X^2 + B_{l2YY}l^2Y^2 \\ &+B_{l2XY}l^2XY + B_{l2X}l^2X + B_{l2Y}l^2Y + B_{l2}l^2, \end{aligned} \tag{A.7}$$

where

$$\begin{array}{lcllcl} B_{l0XX} &=& A_{l0X}A_{l0X}, & B_{l1X} &=& 2(A_{l0}A_{l1X} + A_{l0X}A_{l1}), \\ B_{l0YY} &=& A_{l0Y}A_{l0Y}, & B_{l1Y} &=& 2(A_{l0}A_{l1Y} + A_{l0Y}A_{l1}), \\ B_{l0XY} &=& 2A_{l0X}A_{l0Y}, & B_{l1} &=& 2A_{l0}A_{l1}, \\ B_{l0X} &=& 2A_{l0X}A_{l0}, & B_{l2XX} &=& A_{l1X}A_{l1X}, \\ B_{l0Y} &=& 2A_{l0Y}A_{l0}, & B_{l2YY} &=& A_{l1Y}A_{l1Y}, \\ B_{l0} &=& A_{l0}A_{l0}, & B_{l2XY} &=& 2A_{l1X}A_{l1Y}, \\ B_{l1XX} &=& 2A_{l0X}A_{l1X}, & B_{l2X} &=& 2A_{l1X}A_{l1}, \\ B_{l1YY} &=& 2A_{l0Y}A_{l1Y}, & B_{l2Y} &=& 2A_{l1Y}A_{l1}, \\ B_{l1XY} &=& 2(A_{l0X}A_{l1Y} + A_{l0Y}A_{l1X}) & B_{l2} &=& A_{l1}A_{l1} \end{array} \tag{A.8}$$

where

$$\begin{array}{lcllcl} A_{l1X} &=& -v^2\sin^2(s)u_2u_1, & A_{l0X} &=& v\sin(s)u_1', \\ A_{l1Y} &=& v^2\sin^2(s)u_1, & A_{l0Y} &=& v\sin(s)(u_1'u_2 - u_1u_2'), \\ A_{l1} &=& -v^2\sin(s)u_1\cos(s), & A_{l0} &=& -v\cos(s)(u_1'u_2 - u_1u_2'). \end{array} \tag{A.9}$$

A.3 Unknown Camera, step 2

The number of coefficients is quite high but only a fraction of the number if the all polynomials up to the involved degrees had been represented. $x = (\cos(\phi), , \sin(\phi), l, X, Y)$. The coefficients for the second order polynomial. The subscript indicates the powers in the corresponding polynomial.

$$
\begin{aligned}
B_{c0s0l0X0Y0} &= A_0 A_0, \\
B_{c0s0l0X0Y1} &= A_0 A_{0Y} + A_{0Y} A_0, \\
B_{c0s0l0X0Y2} &= A_{0Y} A_{0Y}, \\
B_{c0s0l1X1Y0} &= A_0 A_{0lX} + A_{0lX} A_0, \\
B_{c0s0l1X1Y1} &= A_{0Y} A_{0lX} + A_{0lX} A_{0Y}, \\
B_{c0s0l2X2Y0} &= A_{0lX} A_{0lX}, \\
B_{c1s0l0X1Y0} &= A_0 A_{c1X} + A_{c1X} A_0, \\
B_{c1s0l0X1Y1} &= A_{0Y} A_{c1X} + A_{c1X} A_{0Y}, \\
B_{c1s0l1X0Y0} &= A_0 A_{c1l} + A_{c1l} A_0, \\
B_{c1s0l1X0Y1} &= A_0 A_{c1lY} + A_{0Y} A_{c1l} + A_{c1l} A_{0Y} + A_{c1lY} A_0, \\
B_{c1s0l1X0Y2} &= A_{0Y} A_{c1lY} + A_{c1lY} A_{0Y}, \\
B_{c1s0l1X2Y0} &= A_{0lX} A_{c1X} + A_{c1X} A_{0lX}, \\
B_{c1s0l2X1Y0} &= A_{0lX} A_{c1l} + A_{c1l} A_{0lX}, \\
B_{c1s0l2X1Y1} &= A_{0lX} A_{c1lY} + A_{c1lY} A_{0lX}, \\
B_{c0s1l0X1Y0} &= A_0 A_{s1X} + A_{s1X} A_0, \\
B_{c0s1l0X1Y1} &= A_{0Y} A_{s1X} + A_{s1X} A_{0Y}, \\
B_{c0s1l1X0Y0} &= A_0 A_{s1l} + A_{s1l} A_0, \\
B_{c0s1l1X0Y1} &= A_0 A_{s1lY} + A_{0Y} A_{s1l} + A_{s1l} A_{0Y} + A_{s1lY} A_0, \\
B_{c0s1l1X0Y2} &= A_{0Y} A_{s1lY} + A_{s1lY} A_{0Y}, \\
B_{c0s1l1X2Y0} &= A_{0lX} A_{s1X} + A_{s1X} A_{0lX}, \\
B_{c0s1l2X1Y0} &= A_{0lX} A_{s1l} + A_{s1l} A_{0lX}, \\
B_{c0s1l2X1Y1} &= A_{0lX} A_{s1lY} + A_{s1lY} A_{0lX}, \\
B_{c2s0l0X2Y0} &= A_{c1X} A_{c1X}, \\
B_{c2s0l1X1Y0} &= A_0 A_{c2lX} + A_{c1X} A_{c1l} + A_{c1l} A_{c1X} + A_{c2lX} A_0, \\
B_{c2s0l1X1Y1} &= A_{0Y} A_{c2lX} + A_{c1X} A_{c1lY} + A_{c1lY} A_{c1X} + A_{c2lX} A_{0Y}, \\
B_{c2s0l2X0Y0} &= A_{c1l} A_{c1l}, \\
B_{c2s0l2X0Y1} &= A_{c1l} A_{c1lY} + A_{c1lY} A_{c1l}, \\
B_{c2s0l2X0Y2} &= A_{c1lY} A_{c1lY}, \\
B_{c2s0l2X2Y0} &= A_{0lX} A_{c2lX} + A_{c2lX} A_{0lX},
\end{aligned}
\tag{A.10}
$$

$$
\begin{aligned}
B_{c1s1l0X2Y0} &= A_{s1X}A_{c1X} + A_{c1X}A_{s1X}, \\
B_{c1s1l1X1Y0} &= A_0A_{c1s1lX} + A_{s1X}A_{c1l} + A_{s1l}A_{c1X} + A_{c1X}A_{s1l} \\
&\quad + A_{c1l}A_{s1X} + A_{c1s1lX}A_0, \\
B_{c1s1l1X1Y1} &= A_{0Y}A_{c1s1lX} + A_{s1X}A_{c1lY} + A_{s1lY}A_{c1X} + A_{c1X}A_{s1lY}, \\
&\quad + A_{c1lY}A_{s1X} + A_{c1s1lX}A_{0Y}, \\
B_{c1s1l2X0Y0} &= A_{s1l}A_{c1l} + A_{c1l}A_{s1l} \\
B_{c1s1l2X0Y1} &= A_{s1l}A_{c1lY} + A_{s1lY}A_{c1l} + A_{c1l}A_{s1lY} + A_{c1lY}A_{s1l}, \\
B_{c1s1l2X0Y2} &= A_{s1lY}A_{c1lY} + A_{c1lY}A_{s1lY}, \\
B_{c1s1l2X2Y0} &= A_{0lX}A_{c1s1lX} + A_{c1s1lX}A_{0lX}, \\
B_{c0s2l0X2Y0} &= A_{s1X}A_{s1X}, \\
B_{c0s2l1X1Y0} &= A_{s1X}A_{s1l} + A_{s1l}A_{s1X}, \\
B_{c0s2l1X1Y1} &= A_{s1X}A_{s1lY} + A_{s1lY}A_{s1X}, \\
B_{c0s2l2X0Y0} &= A_{s1l}A_{s1l}, \\
B_{c0s2l2X0Y1} &= A_{s1l}A_{s1lY} + A_{s1lY}A_{s1l}, \\
B_{c0s2l2X0Y2} &= A_{s1lY}A_{s1lY}, \\
B_{c3s0l1X2Y0} &= A_{c1X}A_{c2lX} + A_{c2lX}A_{c1X}, \\
B_{c3s0l2X1Y0} &= A_{c1l}A_{c2lX} + A_{c2lX}A_{c1l}, \\
B_{c3s0l2X1Y1} &= A_{c1lY}A_{c2lX} + A_{c2lX}A_{c1lY}, \\
B_{c2s1l1X2Y0} &= A_{s1X}A_{c2lX} + A_{c1X}A_{c1s1lX} + A_{c1s1lX}A_{c1X} + A_{c2lX}A_{s1X}, \\
B_{c2s1l2X1Y0} &= A_{s1l}A_{c2lX} + A_{c1l}A_{c1s1lX} + A_{c1s1lX}A_{c1l} + A_{c2lX}A_{s1l}, \\
B_{c2s1l2X1Y1} &= A_{s1lY}A_{c2lX} + A_{c1lY}A_{c1s1lX} + A_{c1s1lX}A_{c1lY} + A_{c2lX}A_{s1lY}, \\
B_{c1s2l1X2Y0} &= A_{s1X}A_{c1s1lX} + A_{c1s1lX}A_{s1X}, \\
B_{c1s2l2X1Y0} &= A_{s1l}A_{c1s1lX} + A_{c1s1lX}A_{s1l}, \\
B_{c1s2l2X1Y1} &= A_{s1lY}A_{c1s1lX} + A_{c1s1lX}A_{s1lY}, \\
B_{c4s0l2X2Y0} &= A_{c2lX}A_{c2lX}, \\
B_{c3s1l2X2Y0} &= A_{c1s1lX}A_{c2lX} + A_{c2lX}A_{c1s1lX}, \\
B_{c2s2l2X2Y0} &= A_{c1s1lX}A_{c1s1lX}
\end{aligned}
$$

with first unsquared coefficients

$$
\begin{array}{lcl}
A_{c2lX} & = & -2v\sin(s)^2(a_9u_1u_2\alpha^2 - a_3a_6a_9uu - 2a_6^2u_1a_9u_2 \\
 & & -a_3u_2\alpha^2 + a_6u_1\alpha^2)/\alpha^2, \\
A_{c1s1lX} & = & v\sin(s)2(-4a_3u_2a_6u_1 - a_6^2\alpha 2uu + 2a_6^2uu - \alpha^2uu + \alpha^4uu \\
 & & -\alpha^4 + 2a_3\alpha^2a_9u_1 + 2a_6\alpha^2a_9u_2 + u_2^2\alpha^4 + 2a_3u_2a_6u_1\alpha^2)/\alpha^2, \\
\beta_1 & = & v\sin(s)(-a_6u_2 - a_3u_1 - u_2^2a_9\alpha^2 + 2a_6\alpha^2u_2 - 2a_6u_2a_3u_1a_9 \\
 & & +\alpha^2a_9 + a6^2a9uu - a_9\alpha^2uu + 2a_3\alpha^2u_1)/\alpha, \\
A_{c1lY} & = & \sin(s)\beta_1, \\
A_{c1l} & = & -\cos(s)\beta_2, \\
A_{c1X} & = & -\sin(s)(a_3\dot{u}_1 + a_6\dot{u}_2)/\alpha, \\
\beta_2 & = & v\sin(s)(\alpha^2u_1u_2 - 2a_6^2u_2u_1 + a_9a_3u_2 - a_9a_6u_1 - a_3a_6uu)/\alpha, \\
A_{s1lY} & = & \sin(s)\beta_2, \\
A_{s1l} & = & -\cos(s)\beta_2, \\
A_{s1X} & = & \sin(s)(\alpha^2u_1\dot{u}_2 - \alpha^2\dot{u}_1u_2 + a_9a3\dot{u}_2 - a_9a6\dot{u}_1)/\alpha, \\
A_{0lX} & = & v\sin(s)2(a9u_1u_2\alpha^2 - a_3a_6a_9uu - 2a_6^2u_1a_9u_2 \\
 & & -a_3u_2\alpha^2 + a_6u_1\alpha^2)/\alpha^2, \\
\beta_3 & = & (a9\dot{u}_1u_2 - a9u_1\dot{u}_2 + a3\dot{u}_2 - a6\dot{u}_1), \\
A_{0Y} & = & \beta_3\sin(s), \\
A_0 & = & -\beta_3\cos(s),
\end{array}
\tag{A.11}
$$

with

$$
\alpha = \sqrt{a3^2 + a6^2}, \qquad uu = u1^2 - u2^2.
$$

Bibliography

[1] U. Andersson. Trajectory estimation and control of autonomous guided vehicles. Licentiate thesis, Luleå University of Technology, Luleå, Sweden, 1989.

[2] N. Andreff, R. Horaud, and B. Espiau. On-line hand-eye calibration. In *Proceedings. Second International Conference on 3-D Digital Imaging and Modeling*, pages 430–436, 1999.

[3] K. Åström. *Invariancy Methods for Points, Curves and Surfaces in Computational Vision*. PhD thesis, Dept of Mathematics, Lund University, Sweden, 1996.

[4] K. Åström and A. Heyden. Continuous time matching constraints for image streams. *Int. Journal of Computer Vision*, 28(1):85–96, 1998.

[5] JY. Bouget. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[6] Y.-L. Chang and J.K. Aggarwal. Calibrating a mobile camera's extrinsic parameters with respect to its platform. In *Procedings of the 1991 International Symposium on Intelligent Control*, pages 443–448, 1991.

[7] J. Dias, A. de Almeida, H. Araujo, and J. Batista. Improving camera calibration by using multiple frames in hand-eye robotic systems. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 1, pages 285–291, 1991.

[8] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Transactions on Robotics and Automation*, 14(4):617–622, 1998.

[9] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Proc. 5th Int. Conf. on Computer Vision, MIT, Boston, MA*, pages 951–956, 1995.

[10] O. Quang-Tuan Luong Faugeras. *The Geometry of Multiple Images*. MIT Press, Cambridge, Massachusetts, 2001.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[12] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ. Press, 2000.

[13] M. T. Heath. *Scientific Computing : An introductory Survey*. McGraw-Hill, 1996.

[14] A. Heyden. Reduced multilinear constraints: Theory and experiments. *Int. Journal of Computer Vision*, 30(1):5–26, 1998.

[15] A. Heyden and K. Åström. A canonical framework for sequences of images. In *Proc. IEEE workshop on Representation of Visual Scenes, MIT, Cambridge, MA, USA*, pages 45–52, 1995.

[16] K. Hyyppä. Optical navigation system using passive identical beacons. In *Proceedings intelligent autonomous systems, Amsterdam*, 1987.

[17] R.K. Lenz and R.Y. Tsai. Calibrating a cartesian robot with eye-on-hand configuration independent of eye-to-hand relationship. *IEEE Transactions on Robotics and Automation*, 11(9):916 –928, 1989.

[18] R.K. Lenz and R.Y. Tsai. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345 –358, 1989.

[19] E.M. Petriu L.P. McNamee and H.J.W. Spoelder. Photogrammetric calibration of a mobile robot model. In *IEEE Instrumental and Measuerement Technology Conferance*, Budapest, 2001.

[20] H. Malm and A. Heyden. Hand-eye calibration from image derivatives. In *Proc. 6th European Conf. on Computer Vision, Dublin, Ireland*, volume 2, pages 493–507, June 2000.

[21] D. Nister. *Automatic dense reconstruction from uncalibrated video sequences*. PhD thesis, KTH, Sweden, 2001.

[22] M. Oskarsson. One-dimensional retina vision with applications in structure and motion estimation. Licentiate thesis, Centre for Mathematical Sciences, Lund Institute of Technology, Sweden, 2000.

[23] M. Oskarsson. *Solutions and their Ambiguities for Structure and Motion Problems*. Phd thesis, Centre for Mathematical Sciences, Lund Institute of Technology, Sweden, 2002.

[24] M. Oskarsson and K. Åström. Automatic geometric reasoning in structure and motion estimation. *Pattern recognition letters*, 21(13-14):1105–1113, 2000.

[25] J. Shi and C. Tomasi. Good features to track. In *Proc. Conf. Computer Vision and Pattern Recognition*, 1994.

[26] H. Stewenius. Homepage with code for hand-eye calibration using multilinear constraints.
http://www.maths.lth.se/matematiklth/personal/stewe/.

[27] B. Triggs. Matching constraints and the joint image. In *Proc. 5th Int. Conf. on Computer Vision, MIT, Boston, MA*, pages 338–343, 1995.

[28] Guo-Qing Wei, K. Arbter, and G. Hirzinger. Active self-calibration of robotic eyes and hand-eye relationships with model identification. *IEEE Transactions on Robotics and Automation*, 14(1):158–166, 1998.

[29] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. 7th Int. Conf. on Computer Vision, Kerkyra, Greece*, 1999.

[30] Hanqi Zhuang and Zhihua Qu. A new identification jocaobain for robotic hand/eye calibration. *IEEE Transactions on Systems, Man and Cybernetics*, 24(8):1284–1287, 1994.

[31] A. Zisserman, A. Fitzgibbon, and G. Cross. Vhs to vrml: 3d graphical models from video sequences. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 51–57, 1999.