Working paper
2005:08

Åsa Windfäll

PRINCIPLES OF ROBUST AND ACCURATE COMPUTATIONAL 3D POSITIONING FROM 2D IMAGE

UNIVERSITY
TROLLHÄTTAN • UDDEVALLA

Department of Technology, Mathematics & Computer Science

Principles of Robust and Accurate Computational 3D Positioning from 2D Image Data

Åsa Windfäll

March 14, 2005

Abstract

Principles of Robust and Accurate Computational 3D Positioning from 2D Image Data

This thesis considers how to compute the 3D position of a camera which can be placed for example on a robot hand. This is done by taking 2D images of reference points that are placed in the room. The image is a mapping from the 3D world frame to a 2D image plane. To find the camera position in the world frame a camera model is used that describes the relationship between the different coordinate systems in terms of rotation and translation. There are basically three parts in this thesis. The first part is the camera model formulation. The second part deals with real-time determination of the position when the robot is moving. Two algorithms have been used, the Levenberg-Marquardt method and the Ortogonal Iteration algorithm. The third part is about calibration. That is when the reference points and the camera parameters are to be determined, which is done before the robot has started to move. Here the Levenberg-Marquardt method is used.

Sammanfattning

Principer för robusta och noggranna beräkningar för 3D positionsbestämning från 2D bild data

Detta examensarbete behandlar principer för att beräkna en kameras 3D position då kameran är placerad på en robothand. Detta görs genom att ta 2D bilder på referenspunkter som är placerade i rummet. Bilderna är en avbildning från det tre-dimensionella rummet rill det två-dimensionella bildplanet. En kameramodell som beskriver relationen mellan de olika koordinatsystemen i termer av rotation och translation används för att hitta kamerapositionen. Detta arbete är uppdelat i tre delar. Första delen behandlar kameramodellen. Den andra delen handlar om positionsbestämning av roboten i realtid, d.v.s. när roboten rör sig. För detta har två metoder använts, Levenberg-Marquardts metod och Orthogonal-Iteration algoritm. Den tredje och sista delen handlar om kalibrering, vilket innebär att referenspunkterna och kameraparametrarna beräknas. Detta görs innan roboten börjar röra på sig, och här används Levenberg-Marquardts metod.

Acknoledgement

This report is a master thesis in Scientific Computing for the Royal Institute of Technology (KTH) in Stockholm, department of Numerical Analysis and Computer Science (Nada). The thesis work has been performed at the University of Trollhättan/Uddevalla (HTU), department of Technology, Mathematics and Computer Science. I Would like to thank my supervisor Kenneth Eriksson, HTU for all inspiration and advice. I would also like to thank Anna-Karin Christiansson, HTU, Elinore Bengtsson, University of Stockholm, and Gunilla Kreiss, Nada/KTH, for all their help during this project. Finally I would like to thank my wonderful husband Johan for all his love and support during my time as a student.

Contents

1 Introduction

With a GPS-device of today the position can be determined with an accuracy of something like one meter, and this is enough when you want to know where you are. But if you want to determine the position of an industrial robot hand the accuracy has to be far better. One common technique today is to calculate the movement of the robot. This could be done by knowing the position of one point at the robot, for example its foot, and then steer the robot to the wanted position. One problem with this technique is that the position is also affected by different disturbances. For example if the robot has a tool in its hand it could cause the hand to be lower due to the weight. This could of course be included in the computation or one could build a more robust robot, which is more expensive. There are different ways to solve this problem, but there can always be some disturbance. Therefore it would be good to always know the true position of the robot hand, so that one can adjust the position if it happens to be incorrect.

One way to solve the problem is to put a small camera on the robot hand and use the principles of photogrammetry, which is to take 2-dimensional (2D) photos of 3-dimensional (3D) reference points from different angles and then determine the position of the camera from the 2D images. The reference points could be light emitting diodes or some kind of reflectors, that are lit by a light source, e.g. light emitting diodes. This thesis deals with this problem. Since one wants to know the position of the moving robot constantly the calculations has to be in real time, and therefor it is very important to have a fast method to solve the pose estimation problem (i.e. determine the position of the robot/camera).

The first part of this thesis is the model formulation, which describes the correspondence between the coordinates of the reference points, the camera and the image. The problem can be formulated as a non-linear minimization problem, where the deviation between the true image coordinates and the image coordinates described by the model is to be minimized. The model used here is a commonly used model, [3], [4], [5], [9], [10], [11]. The second part of the thesis proposes methods to solve the problem. The first problem to be solved is when the reference points are known. This will be done with two different methods, one classical nonlinear optimizations algorithm - Levenberg-Marquardt method [7], and the more modern Orthogonal Iteration Algorithm [3]. The third part of the thesis deals with calibration, which includes both calibration to find reference points and camera calibration. This section also deals a little

bit about distortion. Finally some tests have been performed by taking some real photos with a digital camera.

2 Background

There has been a lot of research done in this area. One person who plays an important role is D.C Brown [2]. He deals with camera calibration in close range photogrammetry in general, but the principles are the same as for camera calibration in positioning of robots. In [2] Brown develops a theory to account for the variation of lens distortion and a method for calibrating radial and decentering distortion of close-range cameras. Roger Y. Tsai and Reimar K. Lenz are also important names and they have written a number of articles about camera calibration for robot positioning, for example, [10],[11],[9],[13],[12]. In [9], Tsai writes about a two-stage technique where he first computes the location of the camera parameters (camera calibration). This is done in a non-iterative way. Gregory D. Hager, Chien-Ping Lu and Eric Mjolsness [3] have written about pose estimation of robots and some of their material was the start for this thesis. The work of A. Wisser et al [4] and Heikkilä [5] are extensions of [3]. They deal with both pose estimation and camera calibration, and they use iterative search methods, which is most commonly used.

The aim of this thesis is to get an overview of how to handle the whole process, from calibration of reference points to the real time pose estimation problem. The articles mentioned above deal with camera calibration and the pose estimation problem, but nothing is said about how to calibrate the reference points.

3 Camera Model

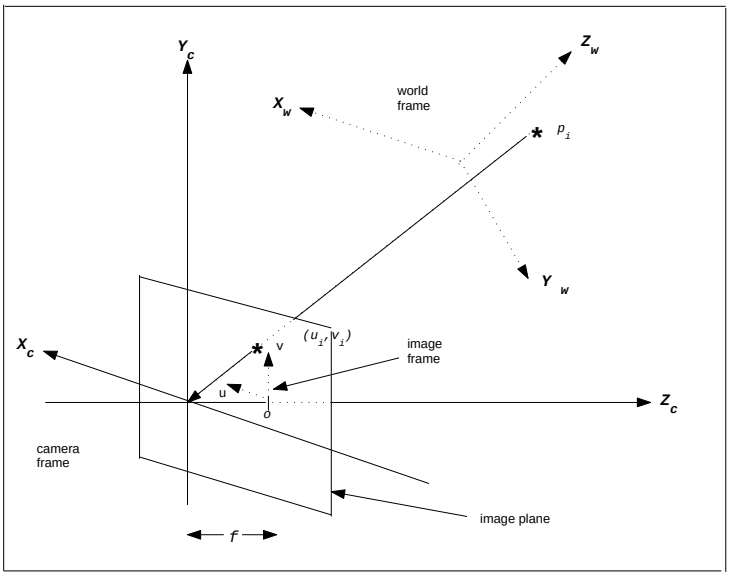

Figure 1: Pinhole camera model here p_i is the reference point, f is the focal length, o is the principal point (u_0, v_0) and (u_i, v_i) is the image point in the image frame that corresponds to p_i in the world frame. (X_c, Y_c, Z_c) defines the camera frame and (X_w, Y_w, Z_w) defines the world frame.

In this camera model there are three different coordinate systems. The world frame,(X_w, Y_w, Z_w), where the reference points (p_i) are, the camera frame, (X_c, Y_c, Z_c), for the camera coordinates and the image plane, (U, V, f), which the reference points are projected on. The third coordinate, f, in the image plane is the focal length, and is fixed somewhere at the Z_c-axis in the camera frame, so the image points (u_i, v_i, f) will be referred to as just the (U, V) coordinates, (u_i, v_i). The idea with this model is to describe a correspondence between the reference points, p_i, the camera coordinates, q_i, and the image points (u_i, v_i), and thereby by able to determine the true position of the camera.

Consider the pinhole camera model [3],[5] illustrated in Figure 1. Given a number of non collinear 3D reference points,
$p_i = (x_{wi}, y_{wi}, z_{wi})^T$, $i = 1, ..., n,\ n \geq 3$, the corresponding camera coordinates, $q_i = (x_{ci}, y_{ci}, z_{ci})^T$ can be described by

$$q_i = P(Rp_i + t), \tag{1}$$

where

$$P = \begin{pmatrix} sf & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

is the perspective transformation matrix, f is the focal length, s is the aspect ratio and (u_0, v_0) is the center of projection on the image plane, which is also called the principal point.

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \in \mathcal{R}^3 \tag{3}$$

is the translation vector and

$$R = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \in \mathcal{R}^{3x3} \tag{4}$$

is the rotation matrix parameterized using Euler angles, ψ, ϕ, θ, e.g. $R = ABC$ where

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{pmatrix}$$

$$B = \begin{pmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{pmatrix}$$

$$C = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

ψ, ϕ, θ and $t = (t_x, t_y, t_z)^T$ are called extrinsic parameters.

The reference points are projected on the image plane with focal length f, e.g. $z_w = f$, and the image coordinates becomes $\mathbf{v}_i = (u_i, v_i, f)$. f, s, u_0, v_0 are called intrinsic parameters and will be determined later in the part about camera calibration. Since we have a pinhole camera model $\mathbf{v}_i$, q_i and the center of projection, (u_0, v_0), are collinear and the image coordinates can be expressed as follows:

$$\begin{pmatrix} u_i \\ v_i \\ f \end{pmatrix} \propto \begin{pmatrix} \lambda u_i \\ \lambda v_i \\ \lambda \end{pmatrix} = q_i = P(Rp_i + t), \tag{5}$$

where

$$\lambda = r_3 p_i + t_z.$$

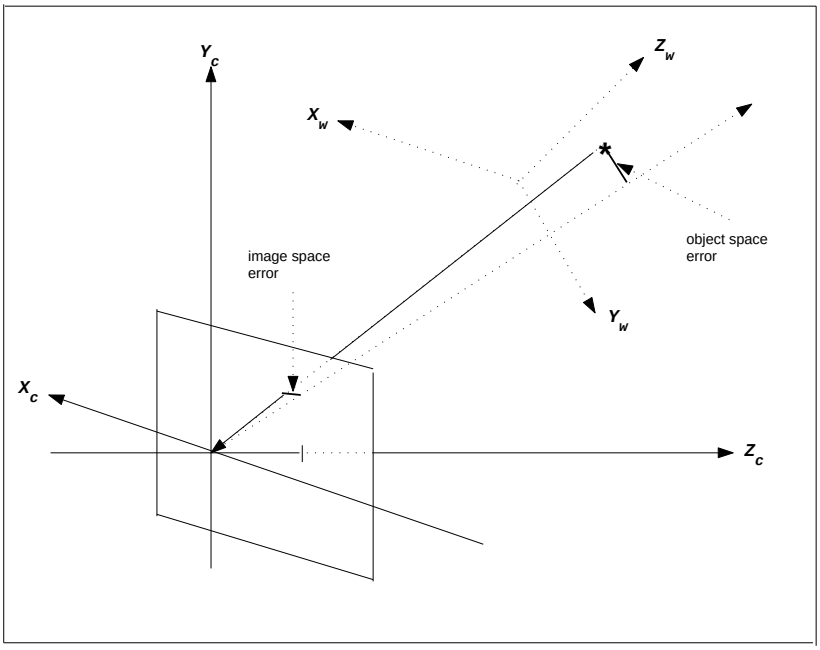

Figure 2: Image space error minimized in section 4.1, and object space error minimized in section 4.2.

(5) gives that

$$u_i = u_o + sf\frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} \tag{6}$$

$$v_i = v_o + f\frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \tag{7}$$

To start with let $f = 1,\ s = 1$ and $(u_0, v_0) = (0, 0)$. Then $P = I$ and equations (1), (6) and (7) become

$$q_i = Rp_i + t, \tag{8}$$

$$u_i = \frac{r_1^T p_i + t_x}{r_3 p_i^T + t_z} \tag{9}$$

$$v_i = \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \tag{10}$$

These equations can be used to solve the pose estimation problem, i.e. determine the camera position, in two different way. One way is to minimize the image space error (Figure 2) which is to minimize the deviation between the observed image points $(\hat{u}_i, \hat{v}_i)$ and the

calculated image points (6) and (7). The other way is to minimize the object space error (Figure 2), which is to minimize the residual error $||Rp_i + t - q_i||$. These two minimization problems will be explained further and used in the next section.

To be able to determine the camera position we have to know the rotation matrix R, i.e. the Euler angles ψ, ϕ, θ, and the translation vector $t = (t_x, t_y, t_z)^T$. This means that there are six unknown parameters. Every reference point gives rise to two equations (6) and (7) and therefore it is necessary to have at least three reference points to fully determine the camera position.

4 Methods

Two different methods have been used to solve the pose estimation problem. The first is the classic nonlinear iterative optimization method called Levenberg-Marquardt method, which is used to minimize the object space error (Figure 5). The other, the Orthogonal Iteration algorithm, is a newer pose estimation algorithm, and is used to minimize the image space error (Figure 5) under the constraint that the rotation matrix is orthogonal, i.e. $R^T R = I$.

4.1 Levenberg-Marquardt Method

As mentioned earlier, one can formulate the pose estimation problem as the minimization of the deviation between the observed image points $(\hat{u}_i, \hat{v}_i)$ and the calculated image points (6) and (7),

$$||F||_2^2 \sum_{i=1}^{n} \left[\left(\hat{u}_i - \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} \right)^2 + \left(\hat{v}_i - \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \right)^2 \right]. \tag{11}$$

The Levenberg-Marquardt method [7] is used to minimize equation (11), where the unknowns is $x = (\psi, \phi, \theta, t_x, t_y, t_z)$. Here, the objective function (11) in vector form is:

$$F(x) = \begin{bmatrix} \left(\hat{u}_1 - \frac{r_1^T p_1 + t_x}{r_3^T p_1 + t_z} \right)^2 \\ \left(\hat{v}_1 - \frac{r_1^T p_1 + t_y}{r_3^T p_1 + t_z} \right)^2 \\ \vdots \\ \left(\hat{u}_n - \frac{r_1^T p_n + t_x}{r_3^T p_n + t_z} \right)^2 \\ \left(\hat{v}_n - \frac{r_1^T p_n + t_y}{r_3^T p_n + t_z} \right)^2 \end{bmatrix} \tag{12}$$

Levenberg-Marquardt method is a mixture of the Gauss-Newton method and a steepest descent method. When the current solution is far from the true solution Levenberg-Marquardt method behaves like a steepest descent method. It converges slow, but convergence is guaranteed. When the current solution is close to the true solution Levenberg-Marquardt behaves like Gauss-Newton, which is much faster than the steepest descent method but must have a good initial guess to converge to the true solution. This method is commonly used for nonlinear least-squares problems [3].

One drawback with the Levenberg-Marquardt method is that it is a fairly slow method, but one benefit is that this method is very

Algorithm 1 Levenberg-Marquardt algorithm

$$\begin{aligned}(J(x_k)^T J(x_k) + \lambda_k I)d_k &= -J(x_k)F(x_k) \\ x_{k+1} &= x_k + d_k\end{aligned}$$

where d_k is the search direction, F is the objective function to minimize and J is the Jacobian matrix to F. λ is a scalar chosen so that $F(x_k + d_k) < F(x_k)$ holds true. To choose λ is the main difficulty with this method but can be done as follows:

Let $\nu > 1$ (here $\nu = 10$ has been used)
Let $\lambda^{(k-1)}$ denote the value of λ from the previous iteration. Initially let $\lambda^{(0)} = 10^{-2}$. Compute $F(\lambda^{(k-1)})$ and $F(\lambda^{(k-1)}/\nu)$.

* If $F(\lambda^{(k-1)}/\nu) \leq F(x_k)$, let $\lambda^{(k)} = \lambda^{(k-1)}/\nu$.
* If $F(\lambda^{(k-1)}/\nu) > F(x_k)$ and $F(\lambda^{(k-1)}) \leq F(x_k)$, let $\lambda^{(k)} = \lambda^{(k-1)}$.
* If $F(\lambda^{(k-1)}/\nu) > F(x_k)$ and $F(\lambda^{(k-1)}) > F(x_k)$, increase λ by successive multiplication by ν until for some smallest integer ω, $F(\lambda^{(k-1)}\nu^{\omega}) \leq F(x_k)$. Let $\lambda^{(r)} = \lambda^{(r-1)}\nu^{\omega}$.

easy to extend to find more unknown parameters, which will be done in the part about calibration (section 5).

4.2 The Orthogonal Iteration Algorithm

As stated in equation (8) we have $q_i = Rp_i + t$ for every observed reference point. To determine R and t we minimize the residual error, i.e. minimize the object space error, see Figure 2. For now let q_i be considered as known, estimated in some way, and later it will be shown how to estimate them when they are unknown as they are in reality.

$$\|e\|^2 = \min_{R,t} \sum_{i=1}^{n} \|Rp_i + t - q_i\|^2, \text{ s.t } R^T R = I. \tag{13}$$

Let $\{p_i\}$ and $\{q_i\}$ denote the set of points related by (13) and define

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i, \ \bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i, \tag{14}$$

$$p_i' = p_i - \bar{p}, \ q_i' = q_i - \bar{q}, \tag{15}$$

$$M = \sum_{i=1}^{n} q_i' p_i'^T. \tag{16}$$

For both q'_i and p'_i yields that $\sum_{i=1}^{n} q'_i = 0$ and $\sum_{i=1}^{n} p'_i = 0$.

The contents of the following two theorems are found in [3]. Here they are stated as theorems and proved, since they play a central rule in the Orthogonal Iteration algorithm. Here Tr denotes the trace of a matrix.

Theorem 1. *If R^* and t^* minimize (13), then they satisfy*

$$R^* = argmax_R \mathrm{Tr}(R^T M) \tag{17}$$

$$t^* = \bar{q} - R^* \bar{p} \tag{18}$$

Proof. The squared error e in (13) can be rewritten as

$$\|e\|^2 = \sum_{i=1}^{n} \|q'_i - Rp'_i - t'\|^2 \tag{19}$$

or

$$\|e\|^2 = \sum_{i=1}^{n} \|q'_i - Rp'_i\|^2 - 2t' \cdot \sum_{i=1}^{n} (q'_i - Rp'_i) + n\|t'\|^2 \tag{20}$$

where

$$t' = t - \bar{q} + R\bar{p}.$$

Since $\sum_{i=1}^{n} q'_i = 0$ and $\sum_{i=1}^{n} p'_i = 0$ the second sum in (20) is zero. The first term does not depend on t' and the last term could not be less than zero. Therefore the total error is minimized with

$$t' = t - \bar{q} + R\bar{p} = 0 \tag{21}$$

and (19) becomes

$$\|e\|^2 = \sum_{i=1}^{n} \|q'_i - Rp'_i\|^2 = \sum_{i=1}^{n} \|q'_i\|^2 - 2\sum_{i=1}^{n} {q'}_i^T Rp'_i + \sum_{i=1}^{n} \|Rp'_i\|^2 \tag{22}$$

It is obvious that to minimize (22) R must be chosen so that $\sum_{i=1}^{n} {q'}_i^T Rp'_i$ is as large as possible.
Since

$$\mathrm{Tr}(R^T ab^T) = \sum_j \sum_i r_{ij} a_i b_j = \sum_i a_i \sum_j r_{ij} b_j = a^T Rb$$

$$\sum_{i=1}^{n} {q'}_i^T Rp'_i = \mathrm{Tr}\left(R^T \sum_{i=1}^{n} q'_i {p'}_i^T \right) = \mathrm{Tr}\left(R^T M\right). \tag{23}$$

Thus, (17) and (18) minimize the residual error. □

Theorem 1 shows that $R^* = \text{argmax}_R \text{Tr}(R^T M)$ minimizes (13) and the following theorem shows how to find $R^* = \text{argmax}_R \text{Tr}(R^T M)$.

Theorem 2. *If M is the matrix defined in (16), then the solution to (13) is*

$$R^* = M(M^T M)^{-1/2} \tag{24}$$

Proof. Let (U, Σ, V) be a singular value decomposition (SVD) of M,

$$M = U\Sigma V^T = US$$

then $(M^T M)^{1/2} = ((US)^T(US))^{1/2} = (S^T U^T U S^{1/2} = (S^T S)^{1/2}$ since U is orthogonal.

$M^T M$ can be expressed as $M^T M = \sigma_1 g_1 + \sigma_2 g_2 + \sigma_3 g_3$, where σ_i, $i = 1, 2, 3$ are the singular values to M and $g_i = v_i v_i^T$ where $g_i \perp g_j$, $i \neq j$ and $||g_i|| = 1$. $M^T M$ is positive definite so all eigenvalues are positive and S can be expressed as

$$S = (M^T M)^{1/2} = \sqrt{\sigma_1} g_1 + \sqrt{\sigma_2} g_2 + \sqrt{\sigma_3} g_3.$$

If R^* is the solution to (13) then R^* should maximize $\text{Tr}(R^T M) = \text{Tr}(R^T U S)$.

$$\text{Tr}(R^T U S) = \sqrt{\sigma_1}\text{Tr}(R^T U g_1 g_1^T) + \sqrt{\sigma_2}\text{Tr}(R^T U g_2 g_2^T) + \sqrt{\sigma_3}\text{Tr}(R^T U g_3 g_3^T)$$

$$\text{Tr}(R^T U g_i g_i^T) = \text{Tr}(g_i^T R^T U g_i) = \text{Tr}((Rg_i)^T U g_i) = (Rg_i)^T U g_i$$

Ordinary properties for trace and transpose are used in above results, see [8].
Since $||g_i|| = 1$ and both U and R are orthonormal transformations, $(Rg_i)^T g_i \leq 1$ with equality if and only if $Rg_i = Ug_i$. Therefore the maximum of $\text{Tr}(R^T M)$ is obtained when $R = U$.

$$M = US \Rightarrow U = MS^{-1} = M(M^T M)^{-1/2}$$

and therefore the solution to (13) is $R^* = M(M^T M)^{-1/2}$ □

These results are important in the Orthogonal algorithm (OI-algorithm), and it can be shown that the algorithm is globally convergent (for proof see [3]). But even though the algorithm always converges to a solution for any starting point, $x^{(0)}$, it is not guaranteed that it converges to the true pose, it might converge to a local minimum point if the initial guess is to bad.

Now to the developing of the orthogonal Iteration algorithm. Let

$$e_i = (I - \hat{V}_i)(Rp_i + t) \tag{25}$$

be the object space error vector, and

$$\hat{V}_i = \frac{\hat{\mathbf{v}}_i \hat{\mathbf{v}}_i^T}{\hat{\mathbf{v}}_i^T \hat{\mathbf{v}}_i}$$

is the observed line-of-sight projection matrix, and $\hat{\mathbf{v}} = (u_i, v_i, f)$.

Minimizing the squared sum of the errors gives the following optimization problem:

$$\min_{R,t} \sum_{i=1}^{n} \|e_i\|^2 = \min_{R,t} \sum_{i=1}^{n} \|(I - \hat{V}_i)(Rp_i + t)\|^2 \tag{26}$$

If the true rotation matrix and translation vector are obtained, then $\sum_{i=1}^{n} e_i = 0$.

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (I - \hat{V}_i)(Rp_i + t) = 0 \Rightarrow$$

$$(nI - \sum_{i=1}^{n} \hat{V}_i)t = \sum_{i=1}^{n} (\hat{V}_i - I)Rp_i \Rightarrow$$

$$t = \frac{1}{n}\left(I - \frac{1}{n}\sum_j \hat{V}_j\right)^{-1} \sum_j (\hat{V}_j - I)Rp_j. \tag{27}$$

Given this formula for the optimal translation vector as a function of R, $t(R)$, define

$$q_i(R) = \hat{V}_i(Rp_i + t(R)) \text{ and } \bar{q}(R) = \frac{1}{n}\sum_{i=1}^{n} q_i(R),$$

then (26) can be written as

$$\min_{R,t} \sum_{i=1}^{n} \|e_i\|^2 = \min_{R,t} \sum_{i=1}^{n} \|Rp_i + t(R) - q_i(R)\|^2. \tag{28}$$

Define

$$M(R) = \sum_{i=1}^{n} q_i'(R) p_i'^T, \text{ where } p_i' = p_i - \bar{p},\ q_i'(R) = q_i(R) - \bar{q}(R).$$

Algorithm 2 The Orthogonal Iteration algorithm

Assume that the kth estimate of R is $R^{(k)}$, $t^{(k)} = t(R^{(k)})$ and $q_i^{(k)} = R^{(k)}p_i + t^{(k)}$. Then $R^{(k+1)}$ is determined by solving

$$R^{(k+1)} = \text{argmin}_R \sum_{i=1}^{n} \|Rp_i + t - \hat{V}_i q_i^{(k)}\|^2 = \text{argmax}_R \text{Tr}\left(R^T M R^{(k)}\right) \quad (29)$$

In this form $R^{(k+1)}$ is given by (24), and $t^{(k+1)} = t\left(R^{(k+1)}\right)$ is given by (27).

4.3 Comparison of Levenberg-Marquardt Method and the Orthogonal Iteration Method

Since the pose estimation should be in real time it is necessary to have a fast algorithm that solves the problem. The Levenberg-Marquardt (LM) algorithm and the Orthogonal Iteration (OI) algorithm have both been implemented in `Matlab`. In reality it would be better to implement the algorithm in some faster language, like `C` or `Fortran`, however a comparison in `Matlab` shows the relative difference between the methods. To compare the two algorithms the programs have been executed with $n = (10, 20, 30, 40, 50)$ (number of reference points) and a tolerance $\propto 10^{-5}$, where the tolerance is defined as norm $(R_{true} - R_{comp})$. Where R_{comp} is the computed rotation matrix and R is the true rotation matrix. R is known since this is created when simulating taking pictures. For each n the OI-algorithm and LM-method has been executed 1000 times and the result is an average of these. For both methods the reference points are selected randomly for each execution, and the program simulates taking a picture which gives the image coordinates (u_i, v_i). The initial guess is the same for all simulations, $x^{(0)} = \text{ones(n,1)}$. Figure (3) shows the computing time for the two algorithms when n is increased, and Figure (4) shows the number of iterations. One can clearly see in Figure (3) that the OI-algorithm is a much more efficient algorithm. The number of iterations, see Figure (4)is decreasing in the OI-algorithm when the number of reference points is increased, but for the LM-algorithm the number of iterations does not change when the number of reference points are increased. Figure 5 shows that the computing time for the LM-method increases when the tolerance decreases , especially when the number of reference points increase, while the computing time for the OI-algorithm is about the same. All tests show that the OI-algorithm always is the fasted, and should therefore be used to solve the pose estimation problem.

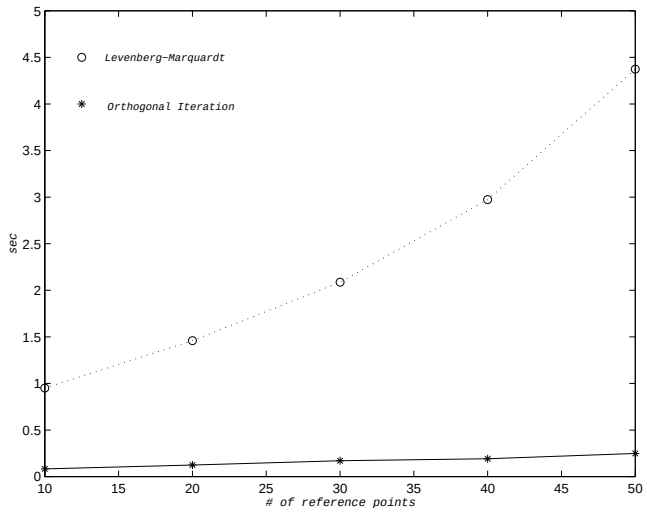

Figure 3: Running times for Levenberg-Marquardt method and the Orthogonal Iteration method. Each point in the plot represents an average of 1000 trials.

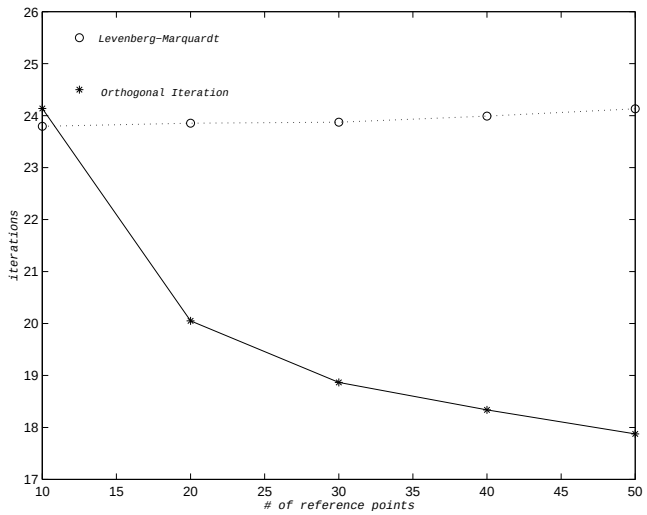

Figure 4: Number of iterations for Levenberg-Marquardt method and the Orthogonal Iteration method. Each point in the plot represents an average of 1000 trials.

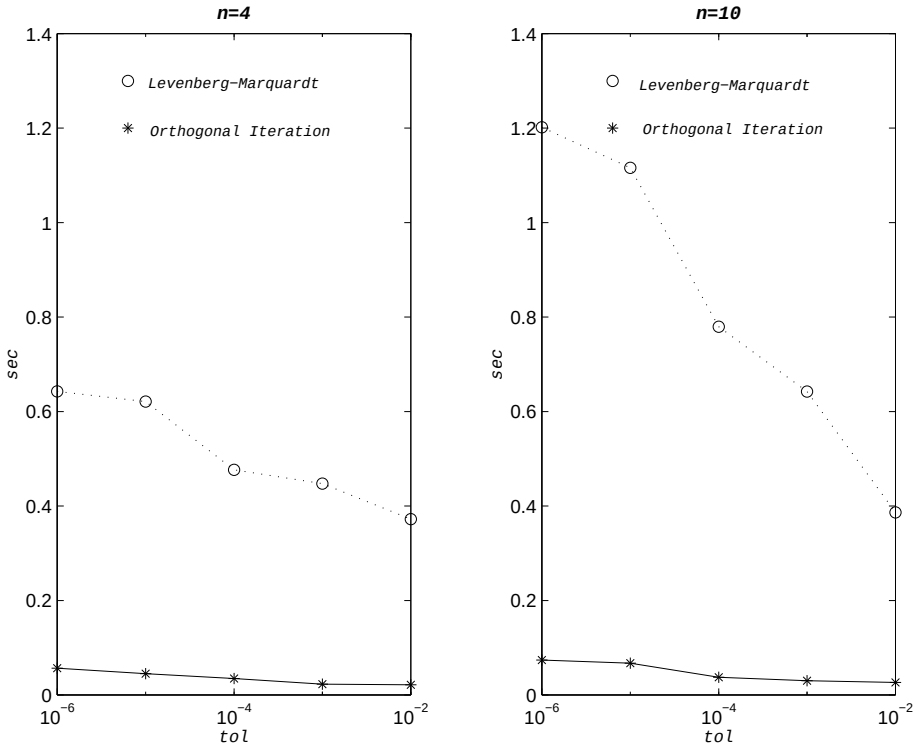

Figure 5: Time in seconds for Levenberg-Marquardt method and the Orthogonal Iteration method for $n = 4$ and $n = 10$, where n is the number of reference points, and different tolerance = norm$(R - R_{comp})$. Each point in the plot represents an average of 1000 trials.

5 Calibration

This section is about calibration which is divided into three parts. The first part is calibration of reference points, i.e. how to find the location of the reference points. The second part deals with the camera calibration which is to find the intrinsic camera parameters (the perspective transformation matrix R in equation (1)). The third part discusses distortion, which is closely related to camera calibration.

5.1 Calibration of Reference Points

Until now the reference points have been given, but to know where the reference points are one has to calibrate. When the reference points were known, there were six parameters to determine, see 4.1. Now all the reference points are unknown, and for each point there are three coordinates, (x_{wi}, y_{wi}, z_{wi}). Therefor, for every unknown reference point there are 3 unknowns. The problem could still be formulated as a minimization problem. The only difference here from (11) is that we not only seek the rotation matrix R and translation matrix t, but also the unknown reference points $p_i,\ i = 1, ..., n$. The squared sum to minimize looks the same as in (11) but the reference points are added to the unknowns, and $x = (\Psi, \Phi\theta, t_x, t_y, t_z, x_{w1}, y_{w1}, z_{w1}, \ldots x_{wn}, y_{wn}, z_{wn})$.

$$\sum_{i=1}^{n} \left[\left(\hat{u}_i - \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} \right)^2 + \left(\hat{v}_i - \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \right)^2 \right]. \tag{30}$$

Totally in (30) there are $6+3n$ unknowns, and from one image we get $2n$ equations, see (12). Obviously there are not enough equations to solve the problem, since $2n < 6 + 3n\ \forall n$. This could be solved by taking more images. Every image gives rise to $2n$ equations, so m images gives $2mn$ equations. And the sum to minimize is

$$\sum_{i=1}^{mn} \left[\left(\hat{u}_i - \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} \right)^2 + \left(\hat{v}_i - \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \right)^2 \right]. \tag{31}$$

where the object function, F, looks like

$$F = \begin{bmatrix} \left(\hat{u}_{11} - \frac{r_1^T p_1 + t_x}{r_3^T p_1 + t_z}\right)^2 \\ \left(\hat{v}_{11} - \frac{r_1^T p_1 + t_y}{r_3^T p_1 + t_z}\right)^2 \\ \vdots \\ \left(\hat{u}_{1n} - \frac{r_1^T p_n + t_x}{r_3^T p_n + t_z}\right)^2 \\ \left(\hat{v}_{1n} - \frac{r_1^T p_n + t_y}{r_3^T p_n + t_z}\right)^2 \\ \vdots \\ \left(\hat{u}_{m1} - \frac{r_1^T p_1 + t_x}{r_3^T p_1 + t_z}\right)^2 \\ \left(\hat{v}_{m1} - \frac{r_1^T p_1 + t_y}{r_3^T p_1 + t_z}\right)^2 \\ \vdots \\ \left(\hat{u}_{mn} - \frac{r_1^T p_n + t_x}{r_3^T p_n + t_z}\right)^2 \\ \left(\hat{v}_{mn} - \frac{r_1^T p_n + t_y}{r_3^T p_n + t_z}\right)^2 \end{bmatrix} \tag{32}$$

For every new image one has to calculate a new rotation matrix, R, and translation vector, t. This means $6m$ unknowns if there are m images. The reference points are the same all the time so there are $3n$ unknowns regardless of how many images that are used. To solve the minimization problem the following should hold:

$$2mn \geq 6m + 3n, \tag{33}$$

number of ref. points, n	images required, m
3	-
4	6
5	4
6	3
$\vdots$	$\vdots$
11	3
12	2
$\vdots$	$\vdots$
∞	2

Table 1: Number of images needed, m, for different number of reference points, n, when the reference points are unknown.

The minimum of reference points are $n = 4$, since if $n = 3$, (33) becomes $6m \geq 6m + 9$. The minimum of images are $m = 2$, since if $m = 1$, (33) becomes $2n \geq 3n + 6$. The calibration part is only done once, and therefore the computing time is not so relevant. Just small adjustments have to be done in the program compared to before. The only difference is that the objective function holds more equations and that the number of unknowns is larger. Since the Levenberg-Marquardt method is so easy to extend it has been used here.

5.2 Camera Calibration

Recall equation (1). Until now the perspective transformation matrix

$$P = \begin{pmatrix} sf & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

has been the unit matrix. The focal length, f, aspect ratio, s, and the principal point, (u_0, v_0) are called the intrinsic camera parameters and can be determined by extending the minimization problem. As in the case with the unknown reference points, the intrinsic camera parameters only have to be determined once, and therefor the Levenberg-Marquardt method can be used also when adding more unknowns. The sum to minimize now is

$$\sum_{i=1}^{mn} \left[\left(\hat{u}_i - \left(u_o + sf \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} \right) \right)^2 + \left(\hat{v}_i - \left(v_o + f \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} \right) \right)^2 \right] \tag{34}$$

There are four more unknowns and to be able to solve the minimization problem now

$$2mn \geq 6m + 3n + 4, \tag{35}$$

must hold.

number of ref. points, n	images required, m
3	-
4	8
5	5
6	4
7	4
8	4
$\vdots$	$\vdots$
16	2
$\vdots$	$\vdots$
∞	2

Table 2: Number of images needed, m, for different number of reference points, n, when the reference points and the intrinsic parameters are unknown.

5.3 Distortion

The camera model described so far gives ideal image coordinates (u, v) from the reference point p, where the light rays pass through the optical center linearly. But in practice lens systems are composed of several optical elements introducing nonlinear distortion. There are mainly two kinds of distortion, radial- and tangential distortion. Radial distortion is an alternation in magnification from the center of the field to any point in the field, measured in a radial direction from the center of the field, see Figure 6. Positive radial distortion is when the image point moves radially outwards from the image center (Figure 6a) and negative radial distortion is when each image points moves radially towards the image center (6b). Tangential distortion is an image defect, usually caused by errors of centration, that results in the displacements of image points perpendicular to a radius from the center of the field. Radial distortion is the most common distortion that is why this thesis only deals with that.

Knowing the observed image coordinates $\hat{\mathbf{v}}_i = (\hat{u}_i, \hat{v}_i)$, the ideal coordinates (the coordinates we would have if the lens was ideal) $\mathbf{v}_{Ii} = (u_{Ii}, v_{Ii})$ could be approximated by

$$\mathbf{v}_{Ii} = \hat{\mathbf{v}}_i + \mathcal{F}(\hat{\mathbf{v}}_i, x) \tag{36}$$

where

$$\mathcal{F}(\hat{\mathbf{v}}_i, x) = \begin{bmatrix} \bar{u}_i(k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \ldots) \\ \bar{v}_i(k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \ldots) \end{bmatrix}, \tag{37}$$

$$\bar{u}_i = \hat{u}_i - u_0,\ \bar{v}_i = \hat{v}_i - v_0,\ r_d = \sqrt{\bar{u}_i^2 + \bar{u}_i^2}$$

and $x = (k_1, k_2, k_3, \ldots)$ are the coefficients for the radial distortion [5].

By replacing the distorted observed image points $(\hat{u}_i, \hat{v}_i)$ with the approximation of the undistorted ideal image points $\mathbf{v}_{Ii} = (u_{Ii}, v_{Ii})$ in (34), we get a new minimization problem with some more unknowns, $(k_1, k_2, k_3, \ldots)$. According to Tsai in [10] only one distortion term, k_1, is needed since more distortion terms would not help and could cause numerical instability. The term to minimize is

$$\sum_{i=1}^{mn} \left[\left(\hat{u}_i + \bar{u} k_1 r_d^2 - \left(sf \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z} + u_0 \right) \right)^2 + \left(\hat{v}_i + \bar{v} k_1 r_d^2 - \left(f \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z} + v_0 \right) \right)^2 \right] \tag{38}$$

More about lens distortion modelling can be found in [1].

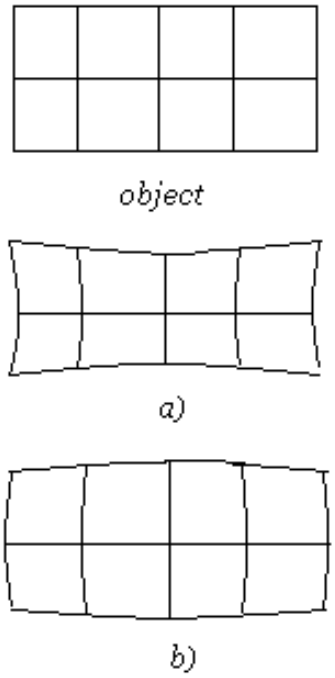

Figure 6: Radial distortion, a) Positive distortion, b) Negative distortion

6 Experiment

When simulating taking images in the computer the model used in this thesis works very well. But what happens when using this in reality? To test this five images were taken from different views on a paper with dots, where the dots were the reference points. The experiment was performed in two steps.

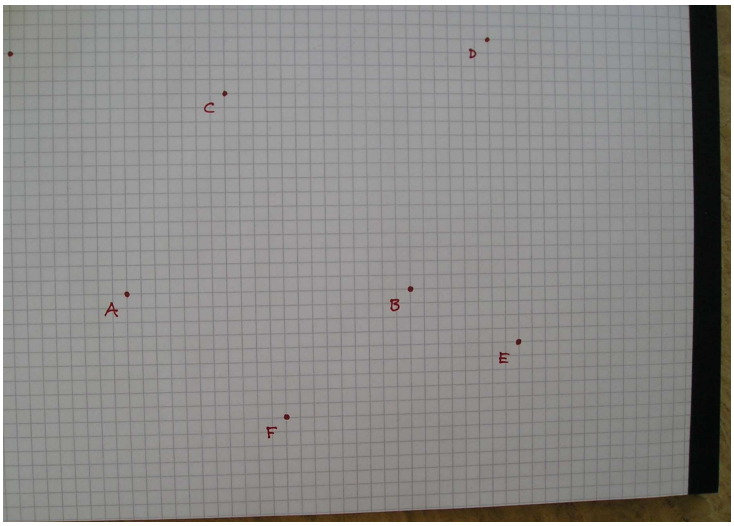

Figure 7: Image of reference points in experiment

6.1 Known Reference Points

In the first test the reference points were known. Since all points are located in the same plane, the z coordinate in the world coordinate system is set to zero. The x and y coordinates are measured by a ruler where origin is dot "A", se Figure 7. When the images are taken all the image coordinates u and v are measured with a ruler. Here the origin in the image plane is in the upper left corner on every image. The images are 26.8×20.1 cm so the principal point (u_0, v_0) should be at $(13.4, 10.05)$. Table 3 shows the result of the camera parameters when all five images are used. As one can see in the table the computed camera parameters differ a bit from the true camera parameters. This is not a surprise since the dots and the reference points are measured in an inaccurate way.

As initial guess the rotation angles in the rotation matrix R and the translation in the translation vector t is estimated by looking at

the images, and the camera parameters are set to $f = 1$, $s = 1,\ u_0 = 10,\ v_0 = 10,\ k_1 = 0,\ k_2 = 0$.

camera parameter	computed camera parameter	true camera parameter
u_0	13.488	13.4
v_0	10.41	10.05
f	26.073	-
s	0.9994	1
k_1	0.078	-
k_2	0.102	-

Table 3: Result from camera calibration in cm

6.2 Unknown Reference Points

Here even four of the seven reference points were considered to be unknown. One has to know at least tree points to define the coordinate system. If there are no known (or one or two known reference points) the world coordinate system could be any coordinate system. The unknown reference points are A,B,C and D, and the initial guess for these are $(4, 4, 4)$ for all four points. For the rotation matrix, R, translation vector, t, and the camera parameters the initial guess is the same as in the previous section.

camera parameter	computed camera parameter	true camera parameter
u_0	13.757	13.4
v_0	10.395	10.05
f	25.073	-
s	0.9984	1
k_1	0.0231	-
k_2	0.0975	-
reference point	computed reference point	true reference point
A	(-0.072,0.004,0.071)	(0,0,0)
B	(9.992,-0.001,0.002)	(10,0,0)
C	(3.444,7.091,-0.013)	(3.5,7,0)
D	(12.974,9.061,-0.054)	(13,9,0)

Table 4: Result from camera and reference point calibration in cm

This experiment shows that the model used seems to be a good one to use. Thought one can not draw any conclusions about the accuracy from this experiment since the accuracy in measuring the observed image point are not especially good.

7 Conclusion

As mentioned in section 4.2 it is not assured that the OI-algorithm converges to the true solution even though the algorithm is globally convergent. This could happen if the initial guess is far away from the true solution. The same holds for the LM-method. Therefore it is important to choose an initial guess that is good enough. That might be a problem in the calibration part, but when the calibrating is done, and the robot has started to move, one can use the last pose as starting point. This could be done since this should be in real time and the robot has not moved far between one image to another. Since the OI-algorithm is faster than the LM-method the OI-algorithm should be used when the calibration part is finished and the reference points and the camera parameters are known, i.e. the OI-algorithm should be used when the robot has started to move and the calculation should be in real time. The LM-method is used in the calibration part, which is performed once before the robot is used in production. Thereby the calculations do not have to be done in real time. In the experiment all five images were used to have a better solution, even though that is more information than needed mathematically. When using this method in reality it is good to have more information than necessary, since then the disturbances do not become as large as they could be otherwise. One drawback with having too many images is that it becomes computationally expensive, especially when there are many reference points. One have to find a balance, that fits the purpose, between efficiency and accuracy. In the calibration part (as in the experiment) accuracy is more important than time. There is a lot more to do before what has been described in this thesis can be used in reality. One thing is to find a way to determine a good initial guess for the calibration part. Another thing that has to be solved is how to identify the reference points, since if images are taken from different locations it is impossible to know which point in the image that corresponds to a particular reference point. When simulating taking pictures in the computer it is easy to know which image point that belongs to a certain reference point and in the test that have been performed in this thesis the reference points have been marked. In reality one can not solve the identification problem in this way. One thing that can be done is to form a number of reference points in different pattern, which easily can be identified, and thereby come to conclusion which reference point that corresponds to a certain image point. The problems mentioned above are things that have to be calculated, but there are of course some technical things that have to be done. For example one has to have a program that recognize the image points in the computer. Even though there is more work to do, the model used in this thesis is an indication of how it is possible to treat the whole process, from finding reference points to estimate the pose of the robot in real time.

References

[1] Amer. Soc of Photogrammetry. *Manual of Photogrammetry*, 4 edition, 1980.

[2] D.C. Brown. Close-range camera calibration. *Photogrammetric Engeneering*, (37):855–866, 1971.

[3] Gregory D. Hager Chien-Ping Lu and Eric Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, June 2000.

[4] A. Visser L.O. Hertzberg G.D. van Albada, J.M. Lagerberg. A low-cost pose-measuring system for robot calibration. *Robotics and Autonomous Systems*, (15):207–227, March 1995.

[5] Janne Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, October 2000.

[6] Guillaume Lievain and Damien Rupil. Study of different sensor and investigation about pos-eye for positioning. Technical report, University of Trollhättan/Uddevalla, 2004.

[7] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Jornula of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963.

[8] Lennart Rade and Bertil Westergren. *Mathematics Handbook for Science and Engeneering*, volume 4. 1998.

[9] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. Technical report, IBM T.J Watson Research Center, Yorktown Heights, NY 10598, 1986.

[10] Roger Y. Tsai. A versitile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[11] Roger Y. Tsai and Reimar K. Lentz. Real time robotics hand/eye calibration using 3d machine vision. Technical report, IBM T.J Watson Research Center, Yorktown Heights, NY 10598, 1988.

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

[13] Roger Y. Tsai and Reimar K. Lentz. Calibrating a cartesian robot with eye-on-hand configuration independent of eye-to-hand relationship. Technical report, IBM T.J Watson Research Center, Yorktown Heights, NY 10598, 1998.

Working Papers from the University of Trollhättan/Uddevalla

1999
Perneman, Jan-Erik, Vilken kunskap lyfter? Om Kunskapslyftet i norra Bohuslän och Fyrstad - erfarenheter från första året och några frågor för framtiden. 99:1

2001
Liss, Ann, Utvärdering av kompetensutvecklingsprogrammet Autism 2000. 01:01

Fredriksdotter Larsson, Ewa, Hur ofta fattar du beslut? - om målbilder och beslut hos företagare och chefer. 01:02

Rydstedt, Leif, Odhner, Miguel, Pettersson, Kristina, På väg till nollvisionen: Nationellt projekt i Trollhättan. En arbets- och organisationspsykologisk utvärdering. 01:03

2002
Kjellén, Bengt, Fear in the Workplace - Meanings, Manifestations and Management.
2002:01

2003
Perneman, Jan-Erik, Johansson, Elisabeth, Arbetsintegrerat lärande (AIL) - om integrationens olika former ur ett kunskapsperspektiv. 2003:01

2004
Fredrikson, Magnus, Dalsland i nyheterna. En innehållsanalys av Göteborgs-Postens bevakning av Dalsland 1997-2003. 2004:01

Karlholm, Gudrun (red), Lärande för pedagogisk utveckling vid HTU. Några exempel på utvecklingsarbeten i lärarens egen undervisning. 2004:02

Kauffeldt, Anders, Welander, Eva, Att handleda handledare - ett led i arbetsintegrerat lärande. 2004:03

Krohn, Karin, Claesson, Silwa, Blivande språklärares syn på sin utbildning. 2004:04

Lievain, Guillaume, Rupil, Damien, Study of different sensor systems and investigation about PosEye for positioning. 2004:05

2005
Persson, Tore, Ottermark, Kenth, En kritisk analys av företagsnätverk. 2005:01

Nilsson, Henrik, Metal Deposition Experiments: Electrode angle and weld speed. 2005:02

Nilsson, Henrik, Patents for Metal Deposition. 2005:03

Nilsson, Henrik, Geometry measurement using M-spot 90. 2005:04

Nilsson, A. Lena, Brukets nya döttrar: Pedagogiska perspektiv i en mållös utvärdering av ett sex- och samlevnadsprojekt. 2005:05

Martin, Anneli, Silkepapper och rosa gräs: En dokumentation och utvärdering av Kulturbyrån i Trollhättan 2000 - 2003. 2005:06

Chavier, Emmanuel, Investigations of PosEye - a photogrammetry position measuring system.
2005:07

Windfäll, Åsa, Principles of Robust and Accurate Computational 3D Positioning from 2D Image.
2005:08

UNIVERSITY
TROLLHÄTTAN • UDDEVALLA

ISSN 1653-2384

All working papers are obtainable through the University of Trollhättan/Uddevalla,
Box 936 SE 461 29 Trollhättan Sweden, phone +46 520 - 47 50 00, fax +46 520 - 47 51 99, www.htu.se