The Registration Problem Revisited: Optimal Solutions From Points, Lines and Planes

Carl Olsson
calle@maths.lth.se

Fredrik Kahl
fredrik@maths.lth.se

Magnus Oskarsson
magnuso@maths.lth.se

Centre for Mathematical Sciences
Lund University, Sweden

Abstract

In this paper we propose a practical and efficient method for finding the globally optimal solution to the problem of pose estimation of a known object. We present a framework that allows us to use both point-to-point, point-to-line and point-to-plane correspondences in the optimization algorithm. Traditional methods such as the iterative closest point algorithm may get trapped in local minima due to the non-convexity of the problem, however, our approach guarantees global optimality.

The approach is based on ideas from global optimization theory, in particular, convex under-estimators in combination with branch and bound. We provide a provably optimal algorithm and demonstrate good performance on both synthetic and real data.

1. Introduction

A frequently occurring and by now a classical problem in computer vision, robotic manipulation and photogrammetry is the registration problem, that is, finding the transformation between two coordinate systems, see [13, 6, 11] and the references therein. The problem appears in several contexts: relating two stereo reconstructions, solving the hand-eye calibration problem and finding the absolute pose of an object given 3D measurements.

There are a number of solutions proposed and perhaps the most well-known is by Horn *et al* [7]. They derive a closed-form solution for the Euclidean (or similarity) transformation that minimizes the sum of squares error between the transformed points and the measured points. As pointed out in [8], this is not an unbiased estimator if there are measurement errors on both point sets.

The more general problem of finding the registration between two 3-D shapes was considered in [2], where the *iterative closest point* (ICP) algorithm was proposed to solve the problem. The algorithm is able to cope with different geometric primitives, like point sets, line segments and different kinds of surface representations. However, the algorithm requires a good initial transformation in order to converge to the globally optimal solution, otherwise only a local optimum is achieved. A number of approaches have been devoted to make the algorithm more robust to such difficulties, e.g. [5, 4], but the algorithm is still plagued by local minima as no guarantee of global optimum is obtained.

In this paper, we generalize the method of Horn *et al* [7] by incorporating point, line and plane features in a common framework. Given *point-to-point*, *point-to-line*, or *point-to-plane* correspondences, we demonstrate how the transformation (Euclidean or similarity) relating the two coordinate systems can be computed based on a geometrically meaningful cost-function. Though the resulting optimization problem becomes much harder - the cost-function is a polynomial function of degree four in the unknowns and there may be several local minima. Still, we present an efficient algorithm that guarantees global optimality. A variant of the ICP-algorithm with a point-to-plane metric was presented in [4], but it is based on local iterative optimization.

The algorithm presented in this paper is based on relaxing the non-convex problem by convex under-estimators and then using branch and bound to focus in on the global solution [9]. The under-estimators are obtained by replacing bilinear terms in the cost-function with convex and concave envelopes, see [10] for further details.

In summary, our main contributions are:

- A generalization of Horn's method for the registration problem using points, lines and planes.
- An efficient algorithm for computing the global optimum of the corresponding quartic polynomial cost-function.
- The introduction of convex and concave relaxations of monomials in the computer vision literature. This opens up the possibility of attacking similar problems for which so far only local algorithms exist.

0-7695-2646-2/06 $20.00 (c) 2006 IEEE

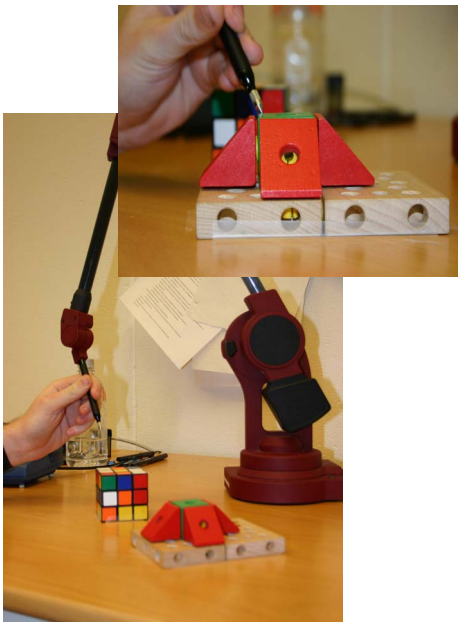

Figure 1. The experimental setup for the tests done in Section 5.

2. The Registration Problem

We will now review the methods of Horn *et al* as presented in [7]. Given two corresponding point sets we want to find the best transformation that maps one set onto the other. The best is here taken to mean the transformation that minimizes the sum of squared distances between the points, i.e.

$$\sum_{i=1}^{m} ||T(x_i^p) - y_i^p||_2^2, \tag{1}$$

where x_i^p and y_i^p, $i = 1, .., m$, are the 3D points in the respective coordinate systems. Here we assume T to be either a Euclidean or a similarity transformation,

$$T(x) = sRx + t,$$

with $s \in \mathbb{R}_+$, $R \in SO(3)$ and $t \in \mathbb{R}^3$ ($s = 1$ corresponds to the Euclidean case). Following Horn [7], it turns out that the translation t is given by

$$t = \frac{1}{m}\sum y_i^p - R\frac{1}{m}\sum x_i^p = \bar{y}^p - R\bar{x}^p. \tag{2}$$

This will turn equation (1) for the Euclidean case into

$$\sum_{i=1}^{m} ||Rdx_i^p - dy_i^p||_2^2 = \tag{3}$$

$$= \sum_{i=1}^{m} (dx_i^p)^T dx_i^p + (dy_i^p)^T dy_i^p - 2(dx_i^p)^T R^T dx_i^p, \tag{4}$$

with $dx_i^p = (x_i^p - \bar{x}^p)$ and $dy_i^p = (y_i^p - \bar{y}^p)$. Due to the orthogonality of R this expression becomes linear in R. Now

Corresp. type	Euclidean/ Similarity	Affine
Point-Point	Horn [7]	Linear Least Squares
Point-Plane	Our algorithm	Linear Least Squares
Point-Line	Our algorithm	Linear Least Squares
Combination	Our algorithm	Linear Least Squares

Table 1. Methods available for estimating the registration for different types of correspondences and transformations.

R can be determined from the singular value decomposition of a matrix constructed from dx_i^p and dy_i^p. The details can be found in [7].

In this paper we will consider not only *point-to-point* correspondences but also *point-to-line* and *point-to-plane* correspondences. In the following sections it will be shown why the extension to these types of correspondences result in more difficult optimization problems. In Figure 1, a measurement device is shown which generates 3D point coordinates and which will be used for validation purposes. Table 1 describes different methods available for global optimization. Note that if the transformation considered is affine, i.e., $T(x) = Ax + t$, then the problem is simply a linear least squares problem.

2.1. Point-to-Plane Correspondences

We will now consider the point-to-plane problem. Suppose we have a number of planes π_i in one coordinate system and points x_i^π $i = 1, ..., m_\pi$ in another, and we assume that point x_i^π lies on plane π_i. Let $d(x, \pi)$ be the minimum distance between a point x and a plane π. The problem is now to find $s \in \mathbb{R}_+$, $R \in SO(3)$ and $t \in \mathbb{R}^3$ that minimizes

$$f_\pi(s, R, t) = \sum_{i=1}^{m_\pi} d(sRx_i^\pi + t, \pi_i)^2. \tag{5}$$

From elementary linear algebra we know that this can be written as

$$f_\pi(s, R, t) = \sum_{i=1}^{m_\pi} \left((sRx_i^\pi + t - y_i^\pi) * n_i\right)^2, \tag{6}$$

where y_i^π is any point on the plane π_i, n_i is a unit normal of the plane π_i and $*$ is the inner product in $\mathbb{R}^3$. Thus we want to solve the problem

$$\min_{s\in\mathbb{R}_+, R\in SO(3), t\in\mathbb{R}^3} \sum_{i=1}^{m_\pi} \left(n_i^T(sRx_i^\pi + t - y_i^\pi)\right)^2. \tag{7}$$

In order to reduce the dimensionality of this problem we now derive an expression for the translation t. This is similar to the approach by Horn *et al.* (see [7]) in which all measurements are referred to the centroids.

If we let R be any 3×3 matrix we can consider the problem (7) as minimizing (6) with the constraints $g_{ij}(s, R, t) =$

$r_i^T r_j - \delta_{ij}\ i,j = 1,2,3$, where r_i is the columns of R and δ_{ij} is the Dirac function. From the method of Lagrange multipliers we know that for a local minimum (s^*, R^*, t^*) (and hence a global) there must be numbers $\lambda_{i,j}$ such that

$$\nabla f_\pi(s^*, R^*, t^*) + \sum_{i=1}^{3}\sum_{j=1}^{3} \lambda_{ij} \nabla g_{ij}(s^*, R^*, t^*) = 0. \quad (8)$$

Here the gradient is taken with respect to all parameters. We see that the constraint is independent of the translation t, thus it will disappear if we apply the gradient with respect to t. Moreover we see that we will get a linear expression in t and, thus we are able to solve for t. It follows that

$$t = N^{-1} \sum_{i=1}^{m_\pi} n_i n_i^T (y_i^\pi - sRx_i^\pi), \quad (9)$$

where $N = \sum_{j=1}^{m_\pi} n_j n_j^T$. Note that if N is not invertible then there are several solutions for t. However if t and $\tilde{t}$ are two such solutions then their difference $\hat{t} = t - \tilde{t}$ is in the null space of N thus $\sum n_j n_j^T \hat{t} = 0$. Now one can easily prove by inserting $\tilde{t}$ into the objective function (6) that $f_\pi(s, R, \tilde{t}) = f_\pi(s, R, t)$. This means that there are infinitely many solutions and thus the problem is not well-posed.

Next we turn to the problem of dealing with the rotation and scaling. A common way to parametrize rotations is to use quaternions (see [1]). Let $q = (q_1, q_2, q_3, q_4)^T$ be the unit quaternion parameters of the rotation matrix R. If the scale factor s is free to vary, one can equivalently drop the condition $||q|| = 1$. We note that for vectors a and b, both in $\mathbb{R}^3$, we can rewrite the term $a^T sRb$ as the quadratic form $q^T Bq$, where q is the 4×1 vector containing the quaternion parameters and B is a 4×4 matrix that depends on a and b. If we substitute (9) into (6) we see that the objective function f can be written as

$$f(q) = \sum_{i=1}^{m_p} (q^T B_i q + k_i)^2, \quad (10)$$

where k_i are constants. Hence our problem can be viewed as minimization of this 4th degree polynomial in 4 unknown variables.

2.2. Point-to-Line Correspondences

The problem of point-to-line correspondences can be treated in a similar way as in the case of point-to-plane correspondences. One difference is that in this case every point-to-line correspondence gives three squared terms instead of one as in the point-to-plane case. Let x_i^l be the measured 3D points and let l_i, $i = 1, ..., m_l$, be the corresponding lines. Then the sum of squared distances between the transformed points and the lines can be written

$$f_l(s, R, t) = \sum_{i=1}^{m_l} ||(I - v_i v_i^T)(sRx_i^l + t - y_i^l)||^2, \quad (11)$$

where v_i is a unit direction vector for the line l_i and y_i^l is any point on the line l_i. Note that the three components of $(I - v_i v_i^T)(Rx_i^l + t - y_i^l)$ are linearly dependent since $(I - v_i v_i^T)$ is a rank 2 matrix. However it would not make sense to remove any of them since we would then not be optimizing the geometrical distance any more.

Set $V_i = (I - v_i v_i^T)$. Note that $V_i^T V_i = V_i V_i = V_i$. Using Lagrange multipliers we can now derive the following expression for t

$$t = V^{-1} \left(\sum_{i=1}^{m_l} V_i (y_i^l - sRx_i^l) \right), \quad (12)$$

where $V = \sum_{i=1}^{m_l} V_i$. To see that we can formulate (11) as a sum of squared quadratic forms we note that

$$f_l(s, R, t) = \sum_{k=1}^{3}\sum_{i=1}^{m_l} ||v_i^k (Rx_i^l + t - y_i^l)||^2, \quad (13)$$

where v_i^k is the row vector containing the k'th row of V_i. Substituting (12) into (13) yields again a function of the form

$$f_l(q) = \sum_{i=1}^{3m_l} (q^T B_i q + k_i)^2. \quad (14)$$

2.3. Point-to-Point Correspondences

The case of point-to-point correspondences is the easiest one. Let x_i^p be the measured points and y_i^p be the corresponding points $i = 1, ..., m_p$. The objective function can in the same way as for the point-to-line case be written as

$$f_p(s, R, t) = \sum_{k=1}^{3}\sum_{i=1}^{m_p} ||e_k (sRx_i^p + t - y_i^p)||^2 \quad (15)$$

where e_k is the k'th row of the identity matrix. Substituting (2) into (15) yields a function of the desired form:

$$f_p(q) = \sum_{i=1}^{3m_p} (q^T B_i q + k_i)^2. \quad (16)$$

2.4. Merging the Different Kinds of Correspondences.

If we have many types of correspondences and want to combine them we find the goal function by adding (6), (13)

and (15). We then use Lagrange multipliers to compute the expression for t. One gets

$$t = M^{-1}\Big(\sum_{i=1}^{m_p}(y_i^p - Rx_i^p) + \tag{17}$$

$$+\sum_{i=1}^{m_l} V_i(y_i^l - Rx_i^l) + \sum_{i=1}^{m_\pi} n_i n_i^T(y_i^\pi - Rx_i^\pi)\Big),$$

where $M = m_p I + \sum_{i=1}^{m_l} V_i + \sum_{j=1}^{m_\pi} n_j n_j^T$. Substituting this into the objective function we can now find an expression of the type

$$f(q) = \sum_{i=1}^{m_\pi+3m_l+3m_\pi} (q^T B_i q + k_i)^2. \tag{18}$$

3. Convex Optimization and Branch-and-Bound Algorithms.

In this section we briefly present some notations and concepts of convex optimization and branch and bound algorithms. For a more detailed introduction see [3] and [9].

A convex optimization problem is a problem of the form

$$\begin{array}{ll} \min & g(x) \\ \text{such that} & h_i(x) \leq b_i \quad i = 1, ..., m. \end{array} \tag{19}$$

Here $x \in \mathbb{R}^n$ and both the objective function $g(x) : \mathbb{R}^n \mapsto \mathbb{R}$ and the constraint functions $h_i(x)\mathbb{R}^n \mapsto \mathbb{R}$ are convex functions. Convex problems have the very useful property that a local minimizer to the problem is also a global minimizer. Therefore it fits naturally into our framework.

The convex envelope of a function $h : S \mapsto \mathbb{R}$ (denoted h_{conv}) is a convex function which fulfills:

1. $h_{conv}(x) \leq h(x), \forall x \in S$.
2. If $u(x)$ is convex on S and $u(x) \leq h(x), \forall x \in S$ then $h_{conv}(x) \geq u(x), \forall x \in S$.

Here S is a convex domain. The concave envelope is defined analogously. The convex envelope of a function has the nice property that it has the same global minimum as the original function. However computing the convex envelope usually turns out to be just as difficult as solving the original minimization problem.

3.1. Branch and Bound Algorithms

Branch and bound algorithms are iterative methods for finding global optima of non-convex problems. They work by calculating sequences of provable lower bounds which converges to the global minima. The result of such an algorithm is usually an $\epsilon-$suboptimal solution i.e. a solution that is at most ϵ from the global minimum.

Consider the following problem. We want to minimize a non-convex scalar function $f(t)$ over a rectangle D_0. For any rectangle $D_n \in D_0$ let $f_{min}(D_n)$ be the minimum value of f on D_n and $f_l(D_n)$ be a lower bound for f on D_n. Also we require that the approximation gap $f_{min}(D_n) - f_l(D_n)$ uniformly goes to zero as the maximum length of the sides of D_n (denoted $|D_n|$) goes to zero. Or in terms of (ϵ,δ) we require that

$$\forall \epsilon > 0, \exists \delta > 0 \text{ s.t } \forall D_n \in D_0,$$
$$|D_n| \leq \delta \Rightarrow f_{min}(D_n) - f_l(D_n) \leq \epsilon.$$

If such a function can be obtained then a strategy to obtain an ϵ-suboptimal solution is to divide the domain into rectangles with sides δ and compute f_l in each rectangle. However the number of such rectangles increases exponentially with $1/\delta$ and therefore this may not be feasible. To avoid this problem a strategy to create as few rectangles as possible can be deployed. Assume that we know that $f_{min}(D) < k$. If $f_l(D_n) > k$ for some n then there is no point in refining D_n further since the minimum will not be attained in D_n. Thus D_n and all $D_k \subseteq D_n$ can be discarded.

The branch and bound algorithm begins by computing $f_l(D_0)$ and the point $q^* \in D_0$ which minimizes $f_l(q), \forall q \in D_0$. This is our current best estimate of the minimum. If $f(q^*) - f_l(D_0) \leq \epsilon$ then q^* is ϵ-suboptimal and the algorithm terminates. Otherwise the rectangle D_0 is partitioned into rectangles $(D_1, ..., D_k)$ (with $k \geq 2$) and one gets the lower bounds $f_l(D_n)$ and the points q_i in which these bounds are obtained. The new best estimate of the minimum is then $q^* := \text{argmin}_{\{q_i\}_{i=1}^k} f(q_i)$. If $f(q^*) - \min_{1\leq i\leq k} f_l(D_i) \leq \epsilon$, then q^* is suboptimal and the algorithm terminates. Otherwise D_0 is refined further, however, the rectangles for which $f_l(D_i) > f(q^*)$ are not considered. This algorithm is guaranteed to find an ϵ-suboptimal solution for any $\epsilon > 0$ but the worst case complexity is exponential. In practice one obtains relatively fast convergence.

4. Application to the Pose Estimation Problem

Recall that the problem is to minimize a function of the type

$$f(q) = \sum_{i=1}^{m}(q^T B_i q + k_i)^2 \tag{20}$$

where $q = (q_1, q_2, q_3, q_4)$. In order to be able to use the branch and bound algorithm we also have to have bounds $q_i^L \leq q_i \leq q_i^U$, $i = 1, 2, 3, 4$. In practice these bounds are usually known since the scale factor can not be arbitrarily large. The quadratic forms $q^T B_i q + k_i$ contain terms of the form $b_{ij} q_i q_j$, and therefore we introduce the new variables $s_{ij} = q_i q_j$, $i = 1, .., 4$, $j = 1, ..., 4$. Now consider the constraints $s_{ij} = q_i q_j$, or equivalently

$$s_{ij} \leq q_i q_j, \tag{21}$$
$$s_{ij} \geq q_i q_j. \tag{22}$$

In the new variables the objective function is convex. The constraints (22) are convex if $i = j$ and (21) is not convex for any i, j. If we replace $q_i q_j$ in (21) with the concave envelope of $q_i q_j$ then (21) will be a convex condition. We also see that by doing this we expand the domain for s_{ij} and thus the minimum for this problem will be lower or equal to the original problem. Similarly we can relax $q_i q_j$ in (22) by its convex envelope and obtain a convex problem which gives a lower bound on the global minimum of the objective function f.

The convex envelope of $q_i q_j$ $i \neq j$, $q_i^U \leq q_i \leq q_i^L$, $q_j^U \leq q_i \leq q_j^L$, is well known (*e.g.* [10]) to be

$$(q_i q_j)_{conv} = \max \left\{ \begin{array}{c} q_i q_j^U + q_i^U q_j - q_i^U q_j^U \\ q_i q_j^L + q_i^L q_j - q_i^L q_j^L \end{array} \right\} \leq q_i q_j, \tag{23}$$

and the concave envelope is

$$(q_i q_j)_{conc} = \min \left\{ \begin{array}{c} q_i q_j^L + q_i^U q_j - q_i^U q_j^L \\ q_i q_j^U + q_i^L q_j - q_i^L q_j^U \end{array} \right\} \geq q_i q_j. \tag{24}$$

Thus the equations (21) and (22) for $i \neq j$ can be relaxed by the linear constraints

$$-s_{ij} + q_i q_j^U + q_i^U q_j - q_i^U q_j^U \geq 0, \tag{25}$$

$$-s_{ij} + q_i q_j^L + q_i^L q_j - q_i^L q_j^L \geq 0, \tag{26}$$

$$s_{ij} - (q_i q_j^L + q_i^U q_j - q_i^U q_j^L) \geq 0, \tag{27}$$

$$s_{ij} - (q_i q_j^U + q_i^L q_j - q_i^L q_j^U) \geq 0. \tag{28}$$

If $i = j$ we need to relax q_i^2 in (21) with its concave envelope. However this is simply a line $aq_i + b$, where a and b is determined by noting that the values $(q_i^L)^2$ and $(q_i^U)^2$ should be attained at the points $q_i = q_i^L$ and $q_i = q_i^U$, respectively. Figure 2 shows the upper and lower bounds of s_1 when $-1 \leq q_1 \leq 1$. We see that even when the interval has only been divided four times the upper bound is quite close to the lower bound. This gives some indication on how the lower bounds on the problem may converge quite rapidly. Since all the non-convex constraints have been replaced by convex constraints, the relaxed problem is now convex. Thus we can minimize this problem to obtain a lower bound on the original problem and as we subdivide the domain these lower bounds will tend to the global minimum of the original problem. To summarize we now state the relaxed problem. Let $\hat{q}$ be a 14×1 vector containing the parameters $(q_1, ..., q_4, s_{11}, s_{12}, ..., s_{44})$. The term $q^T B_i q$ can then be relaxed by the term $b_i^T \hat{q}$ where b_i is a vector of the same size as $\hat{q}$. Note that entries of b_i are obtained directly from the entries of B_i. All the linear constraints can be written as $c_j - a_j^T \hat{q} \geq 0$ where c_j is a constant and a_j is a vector of same size as $\hat{q}$. These constraints can be written as a matrix inequality $c - A^T \hat{q} \geq 0$ where c is a vector whose entries are the c_j and A is a matrix whose columns are the

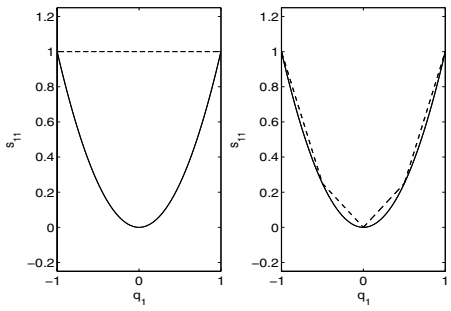

Figure 2. Upper and lower bounds of s_{11}, which relaxes q_1^2 in the interval $[-1, 1]$. **Left**: the initial bound. **Right**: when the interval has been divided four times. Note that the lower bound is exact since $q_1^2 \leq s_{11}$ is convex.

a_j's. Then the relaxed problem can be written

$$\begin{array}{ll} \min & \sum_{i=1}^{m} (b_i^T \hat{q})^2 \\ \text{subject to} & s_{ii} - q_i^2 \geq 0 \quad i = 1, ...4 \\ & c - A^T \hat{q} \geq 0. \end{array} \tag{29}$$

In the case of finding a Euclidean transformation instead of a similarity transformation we can simply add the extra linear constraint

$$s_{11} + s_{22} + s_{33} + s_{44} - 1 = 0. \tag{30}$$

Also we have to check if the problem is feasible in each rectangle D_n - if it is not, the rectangle can be removed for further consideration. To find new optimal function values we check the values of $f(q_n^*/||q_n^*||)$ to make sure that we get a Euclidean transformation. Recall that q_n^* is the minimizer of the relaxed problem.

5. Experiments and Discussion

5.1. Local Minima

The problems usually exhibit local minima. To show some typical behavior of these kinds of functions we generated a problem with eight plane-to-point correspondences. This was done in the following way. We randomly generated eight planes π_i and then picked a point y_i^π from each plane. Then the points $x_i^\pi = R^T(y_i^\pi - t)$ were calculated for known R and t. We used the Matlab built-in function **fmincon** to search for the minima from some different starting points. Table 2 shows the results. Note that the third point is the global minimum. To get an idea of the shape of the function, Figure 3 plots the function-values along a line from one local minimum to the global minimum. To the left is the values on the line from the first point in Table 2 and on the right is the second point.

local min point	*objective function value*
(0, -0.0781, -1.0000, -0.4857)	2.1365
(0, 0.1189, -0.3349, -0.9316)	2.2161
(0.5917, 0.0416, 0.6995, 0.4088)	3.7123e-04
(0.6551, 0.2226, 0.7166, 0.2306)	0.0018
(0,0,0,0)	65.1556

Table 2. Local minima found by the Matlab function **fmincon.**

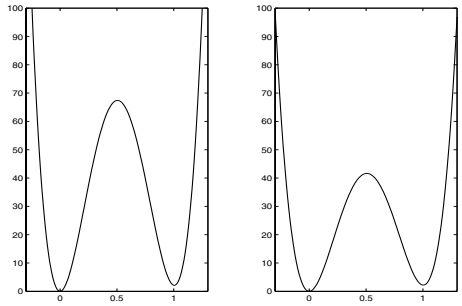

Figure 3. The function values on two lines between the local minima and the global minimum

5.2. Implementation

The implementation of the algorithm was done in Matlab. We basically used the algorithm described in Sections 3.1 and 4. At each iteration the problem (29) is solved for each rectangle D_n to obtain a lower bound on the function D_n. To speed up convergence we use minimizer q_n^* of the problem (29) as a starting guess for a local minimization of the original objective function f. We then compare the function-value at the local optimizer q_n^{loc} and at q_n^*. If any of these values are lower than the current best minimum the lowest one is deemed the new best minimum. Even thou q_n^* is the optimal value of the relaxed problem (29) it is often the case that $f(q_n^*)$ is larger that $f(q_n^{loc})$. Therefore we reach lower values faster if we use the local optimizer and thus intervals can be thrown away faster. If an interval is not thrown away then we divide it into two. We divide the intervals along the dimension which has the longest side. In this way the worst case would be that the number of intervals doubles at each iteration. However we shall see later that in practice this is not the case. As a termination criterion we use the total 4-dimensional volume of the rectangles. One could argue that it would be sufficient to terminate when the approximation gap (see Section 3.1) is small enough, however this does not necessarily mean that the ϵ-suboptimal solution is close to the real minimizer.

To solve the relaxed problem we used SeDuMi (see [12]). SeDuMi is a free add-on for Matlab that can be used to solve problems with linear, quadratic and semi-definiteness constraints. For the local minimization we used the built in Matlab function **fmincon**.

Figure 4 shows the performance of the algorithm in two cases. In both cases the data has been synthetically generated. The first problem is the case of ten point-to-plane, four point-to-line and four point-to-point correspondences. The solid line to the left in Figure 4 shows the number of feasible rectangles for this problem at each iteration. The solid line to the right shows the fourth root of the total volume of the rectangles. Recall that this is a 4-dimensional problem and therefore the fourth root gives an estimate of the total length of the sides in the rectangles. This case exhibits the typical behavior for this algorithm. The second case is the minimal case of 7 point-to-plane correspondences. This seems to be the case where our algorithm has the most difficulties. The dashed lines shows the performance for this case. Note that this case could probably be solved much more efficiently with a minimal case solver, it is merely included to show the worst case behavior of the algorithm. For comparison the dotted line to the left shows what the number of rectangles would be if no rectangles where thrown away. In the first case the algorithm terminated after 38 iterations, and in the second after 39.

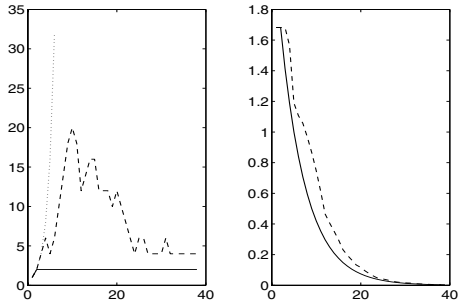

Figure 4. Left, the number of feasible rectangles at each iteration. Right, the fourth root of the total volume of the rectangles.

5.3. Experiments on Real Data

This section presents two experiments made with real data. The experimental setup can be viewed in figure 1. We used a MicroScribe-3DLX 3d scanner to measure the 3D-coordinates of some points on two different objects. The 3D-scanner consists of a pointing arm with five degrees of freedom and a foot pedal. It can be connected to the serial-port on a PC. To measure a 3D-coordinate one simply moves the pointing arm to the point and presses the pedal. The accuracy of the device is not very high. If one tries to measure the same point but varies the pose of the pointer one can obtain results that differ by approximately half a millimeter. The test objects are the ones that are visible in figure 1, namely the Rubik's cube and the toy model. By request of the designer will refer to the toy model as the space station.

5.3.1 Rubik's Cube Experiment

The first experiment was done by measuring on a Rubik's cube. The Rubik's cube contains both lines planes and points and therefore suits our purposes. We modeled three of the sides of the cube and we measured nine point-to-plane, two point-to-line and three-point-to-point correspondences on these sides. Figure 5 shows the model of the Rubik's cube and the points obtained when applying the estimated transformation to the measured data points. The points marked with crosses are points measured on the

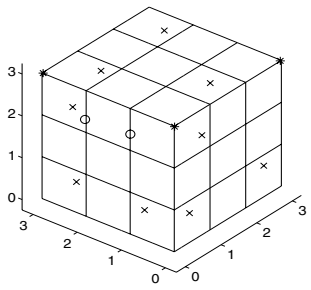

Figure 5. The model of the Rubik's cube.

planes, the points marked with rings are measured on lines and the points marked with stars are measured on corners. It is difficult to see from this picture how well the points fit the model, however to the left in figure 6 we have plotted the residual errors of all the points. Recall that the residuals are the squared distances. The first nine are the point-to-plane, the next two are the point-to-line and the last four are the point-to-point correspondences. We see that the point-

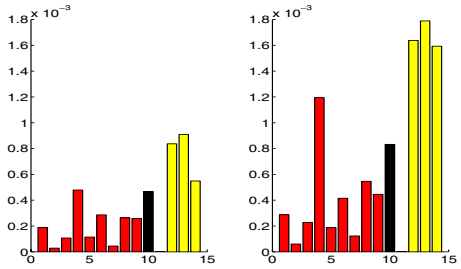

Figure 6. Left, Residual errors for all correspondences. Right, Leave-one-out residuals.

to-point residuals are somewhat larger than the rest of the residuals. There may be several reasons for this, one is that using the 3D-scanner it is much harder to measure corners than to measure planes or lines. This is because a corner is relatively sharp and thus the surface to apply the pointer to is quite small making it easy to slip. To the right in figure 6 is the result from a leave-one-out test. Each bar represents an experiment where we leave one data point out, calculate the optimal transformation and measure the residual of the left out point. Again the first nine are the point-to-plane, the next two are the point-to-line and the last four are the point-to-point correspondences.

For comparison we also implemented the algorithm by Horn *et al.* [7] and an algorithm based on linear least squares. The last algorithm first finds an optimal affine transformation $y = Ax + b$ and then finds sR by minimizing $||A - sR||_F$ where $||\cdot||_F$ is the frobenius norm. The translation is then calculated from equation (18). Note that in order to calculate the affine transformation one needs at least 12 equations. Table 3 shows the results obtained from the three methods when solving the Rubik's cube experiment. The residuals stated are the sum of the different types of correspondence residuals. Note that this experiment is somewhat unfair to the algorithm by Horn *et al.* since it only optimizes the point-to-point correspondences. However due to the lack of other alternatives we still use it for comparison. As one would expect the solution obtained by Horn's algorithm has a lower residual sum for the point-to-point correspondences, since this is what it optimizes. Our algorithm has a lower total residual sum since this is what we optimize.

Residuals:	*Our Alg.*	*Horn*	*Lin.Least Sq.*
point-point	0.0023	1.3e-04	0.0051
point-line	4.7e-04	0.0016	0.0027
point-plane	0.0018	0.0095	0.0049
Total	0.0045	0.0113	0.0127

Table 3. Residuals of the cube problem.

5.3.2 Space Station Experiment

The next experiment was done by measuring on the space station. It is slightly more complicated than the Rubik's cube and it contains more planes to measure from. We measured 27 point-to-plane, 12 point-to-line and 10-point-to-point correspondences on the space station. Figure 7 shows the model of the space station and the points obtained when applying the estimated transformation to the measured data points.

To the left in figure 8 we have plotted the residual errors of all the points, and to the right are the results of the leave-one-out test.

Table 3 shows the results obtained from the three methods when solving the space station experiment. Note that in this case the algorithm by Horn *et al.* [7] preforms better since we have included more point-to-point correspondences then in the Rubik's cube experiment. Again our algorithm has the lowest total residual sum, while Horn has the lowest point-to-point residual.

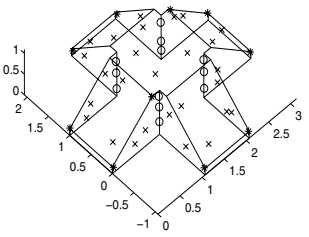

Figure 7. The model of the Space Station.

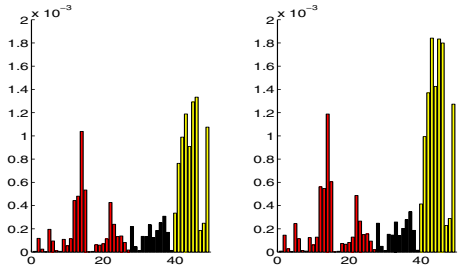

Figure 8. Left, Residual errors for all correspondences. Right, Leave-one-out residuals.

Residuals:	*Our Alg.*	*Horn*	*Lin.Least Sq.*
point-point	0.0083	0.0063	0.0221
point-line	0.0018	0.0036	0.0015
point-plane	0.0046	0.0098	0.0046
Total	0.0147	0.0197	0.0282

Table 4. Residuals of the space station problem.

Residuals:	*Our Alg.*	*Horn*	*Lin.Least Sq.*
point-point	0.0286	0.0220	0.0316
point-line	6.0630e-04	0.4986	0.0072
point-plane	0.0118	0.0404	0.0111
Total	0.0410	0.5610	0.0499

Table 5. Residuals of the simulated data for the Euclidean algorithm.

5.4. Euclidean Algorithm with Synthetic Data

For completeness we also include an experiment where the Euclidean version of the algorithm is tested against the Euclidean versions of Horn and the linear least squares. We artificially generated six point-to-point, three point-to-line and seven point-to-plane correspondences. The results can be seen in Table 5.

6. Conclusions and Future Work

We have presented a unified framework for the registration problem using points, lines and planes. Based on geometric distances, a practical algorithm that computes the globally optimal solution has been developed and tested on realistic scenarios.

Future work includes to investigate degenerate cases and the use of robust norms to improve the general applicability of the approach. In addition, the performance of the algorithm should be tested on a wider range of experiments. Another natural path for further investigation is to incorporate the methodology in the ICP algorithm in order to improve robustness with respect to local minima.

7. Aknowledgements

This work has been funded by the European Commission's Sixth Framework Programme (SMErobot grant no. 011838), and by the Swedish Research Council (grants no. 2004-4579, and no. 2005-3230).

References

[1] S. Altmann. *Rotations, Quaternions and Double Groups*. Clarendon Press, 1986. 3

[2] P. Besl and N. McKay. A method for registration two 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):232–256, 1992. 1

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 4

[4] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *International Conference on Robotics and Automation*, volume 3, pages 2724–2729, 1991. 1

[5] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the icp algorithm. In *3D Digital Imaging and Modeling (3DIM 2003)*, 2003. 1

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition. 1

[7] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using ortonormal matrices. *Journal of the Optical Society of America A*, 5, 1988. 1, 2, 7

[8] K. Kanatani. Unbiased estimation and statistical analysis of 3-d rigid motion from two views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(1):37–50, 1993. 1

[9] B. Kolman and R. E. Beck. *Elementary Linear Programming with Applications*. Academic Press, 1995. 1, 4

[10] H. S. Ryoo and N. V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, 19:403–424, 2001. 1, 5

[11] C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, 4:th edition, 1984. 1

[12] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. 1998. 6

[13] E. H. Thompson. An exact linear solution of the problem of absolute orientation. 15(4):163–179, 1958. 1