Branch-and-Bound Methods for Euclidean Registration Problems

Carl Olsson, *Student Member*, *IEEE*, Fredrik Kahl, and Magnus Oskarsson, *Member*, *IEEE*

Abstract—In this paper, we propose a practical and efficient method for finding the globally optimal solution to the problem of determining the pose of an object. We present a framework that allows us to use point-to-point, point-to-line, and point-to-plane correspondences for solving various types of pose and registration problems involving euclidean (or similarity) transformations. Traditional methods such as the iterative closest point algorithm or bundle adjustment methods for camera pose may get trapped in local minima due to the nonconvexity of the corresponding optimization problem. Our approach of solving the mathematical optimization problems guarantees global optimality. The optimization scheme is based on ideas from global optimization theory, in particular convex underestimators in combination with branch-and-bound methods. We provide a provably optimal algorithm and demonstrate good performance on both synthetic and real data. We also give examples of where traditional methods fail due to the local minima problem.

Index Terms—Registration, camera pose, global optimization, branch-and-bound.

1 Introduction

A frequently occurring and, by now, classical problem in computer vision, robotic manipulation, and photogrammetry is the **registration problem**, that is, finding the transformation between two coordinate systems; see [1], [2], [3]. 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. A related problem is the camera pose problem, that is, finding the perspective mapping between an object and its image. In this paper, we will develop algorithms for computing globally optimal solutions to these registration problems within the same framework.

There are a number of proposed solutions to the registration problem and perhaps the most well known is that by Horn et al. [4]. They derive a closed-form solution for the euclidean (or similarity) transformation that minimizes the sum-of-square error between the transformed points and the measured points. As pointed out in [5], 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 3D shapes was considered in [6], where the *iterative closest point* (ICP) algorithm was proposed to solve the problem. The algorithm is able to cope with different geometric primitives such as 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 attained. A number of approaches have been devoted to making the algorithm more robust to such difficulties, e.g., [7], [8], but the algorithm is still plagued by local minima problems. In [9], a (local) convergence analysis for algorithms similar to ICP is given. Furthermore, a comparison with respect to global convergence is given.

In this paper, we generalize the method of Horn et al. [4] 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. However, 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 [8] based on the same idea, but it is based on a local iterative optimization method.

The **camera pose problem** has been studied for a long time, that is, estimating the camera pose given 3D points and corresponding image points. The minimal amount of data required to solve this problem is three point correspondences and, for this case, there may be up to four solutions. The result has been shown a number of times, but the earliest solution is to our knowledge due to Grunert, already in 1841 [10]. A good overview of the minimal solvers and their numerical stability can be found in [11]. Given at least six point correspondences, the predominant method to solve the pose estimation problem is by estimating the solution linearly using the DLT algorithm, and then use this as a starting solution for a gradient descent method; see [2], [12], [3]. There are also quasi-linear methods that give a unique solution given at least four point correspondences; see [13], [14]. These methods solve a number of algebraic equations and do not minimize the reprojection error. Given a Gaussian noise model for the image measurements and assuming i.i.d. noise, the

• *The authors are with the Centre for Mathematical Sciences, Lund University, Box 118, 221 00 Lund, Sweden.*
E-mail: {calle, fredrik, magnuso}@maths.lth.se.

Manuscript received 12 June 2007; revised 26 Mar 2008; accepted 7 May 2008; published online 20 May 2008.
Recommended for acceptance by M. Pollefeys.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-06-0350.
Digital Object Identifier no. 10.1109/TPAMI.2008.131.

Maximum-Likelihood (ML) estimate is computed by minimizing the norm of the reprojection errors.

The algorithms presented in this paper are based on relaxing the original nonconvex problems by convex underestimators and then using branch and bound to focus in on the global solution [15]. The underestimators are obtained by replacing bilinear terms in the cost function with convex and concave envelopes; see [16] for further details. Branch-and-bound methods have been used previously in the computer vision literature. Perhaps the most similar work to ours is the approach pioneered by Breuel, see [17], [18]. Here, the branch and bound framework is applied to estimate euclidean transformations in the plane and globally optimal matching solutions are obtained. However, there are important differences. While for our problems the correspondences are assumed to be known, the goal in [17] is to find geometrically consistent pairings as well as the corresponding transformation. This makes the framework more general, but there is a price to pay. As stated in [17], a direct application to 3D transformations is often impractical due to the high computational complexity. Therefore, only 2D euclidean transformations with 3 degrees of freedom are experimented upon. In our case, known correspondences allow us to construct tight bounding functions which makes it possible to handle full 3D euclidean (or similarity) transformations with 6 (or 7) degrees of freedom, respectively. Another example of branch and bound applications is [19], where various multiple view geometry problems are considered in a fractional programming framework.

In summary, our main contributions are

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

Although we only give experimental results for point, line, and plane correspondences, it is shown that our approach is applicable to more general settings. In fact, correspondence features involving any convex sets can be handled. The work in this paper is based on the conference papers [20], [21].

2 A General Problem Formulation

We will now make a general formulation of the problem types we are interested in solving. The stated problem is to find a transformation relating two coordinate systems: the model coordinate system and the measurement coordinate system. In the measurement system, we have points that we know should map to some convex sets in the model system. Typically, in a practical situation, the convex sets consist of single points, lines, or planes, but in principle, they may be any convex set. We want to determine a transformation such that the mapping of the measurement points come as close to their corresponding convex sets as possible. The transformation is usually a similarity or a euclidean transformation. We will also consider the transformation consisting of a euclidean transformation followed by a perspective mapping in the camera pose problem.

Let $\|\cdot\|$ be any norm. We define the distance from a (closed) set C as

$$d_C(x) = \min_{y \in C} \|x - y\|. \tag{1}$$

Let $\{x_i\}_{i=1}^m$ be the point measurements and $\{C_i\}_{i=1}^m$ be the corresponding model sets. The registration (or pose) problem can then be written as

$$\min_T \sum_{i=1}^m d_{C_i}(T(x_i)), \tag{2}$$

where T is in the set of feasible transformations. In the case of a similarity transform $T(x) = sRx + t$, with $s \in \mathbb{R}_+$, $R \in SO(3)$, and $t \in \mathbb{R}^3$. ($s = 1$ corresponds to the euclidean case.) Our goal is to find the global optimum of (2). Note that the sets C_i do not have to be distinct. If two points x_i and x_j map to the same set, then we take $C_i = C_j$.

The following theorem shows why it is convenient to only work with convex sets.

Theorem 1. *d_C is a convex function if and only if C is a convex set.*

Proof. Since C can be written as the zero level set $\{x; d_C(x) \leq 0\}$, C must be convex if d_C is convex.

Next, assume that C is convex. Let $z = \lambda x + (1-\lambda)y$ for some scalar $\lambda \in [0,1]$. Pick $x', y', z' \in \partial C$ such that

$$d_C(x) = \min_{u \in C} \|x - u\| = \|x - x'\|, \tag{3}$$

$$d_C(y) = \min_{u \in C} \|y - u\| = \|y - y'\|, \tag{4}$$

$$d_C(z) = \min_{u \in C} \|z - u\| = \|z - z'\|. \tag{5}$$

Since d is a norm $\lambda x' + (1-\lambda)y' \in C$, we obtain

$$\begin{aligned} d_C(\lambda x + (1-\lambda)y) &= \|z - z'\| \leq \|z - (\lambda x' + (1-\lambda)y')\| \\ &\leq \lambda \|x - x'\| + (1-\lambda)\|y - y'\| \\ &= \lambda d_C(x) + (1-\lambda) d_C(y). \end{aligned} \tag{6}$$

The inequality $\|z - z'\| \leq \|z - (\lambda x' + (1-\lambda)y')\|$ is true since $z' = \arg\min_{u \in C} \|z - u\|$. □

Theorem 1 shows that if T depends affinely on some parameters, then (2) is a convex optimization problem in these parameters, which can be solved efficiently using standard optimization methods [20]. This is, however, not the case for euclidean and similarity transformations. In these cases, (2) becomes highly nonconvex due to the orthogonality constraints $R^T R = I$. Instead, we will use larger sets of transformations, which can be parameterized as affine combinations of some parameters, and use (2) to obtain lower bounds. These bounds can then be used to find the global optimum in a branch-and-bound scheme. Note that the convexity of d_C does not depend on the particular choice of norm. We will typically use the 2-norm $\|x\|_2^2 = x^T x$, but our methods are easily extended to norms such as $\|x\|_A^2 = x^T A x$, where A is a positive definite matrix, making it possible to include uncertainty weights.

3 Global Optimization

In this section, we briefly present some notation and concepts of convex optimization and branch-and-bound algorithms. This is standard material in (global) optimization theory. For a more detailed introduction, see [22] and [15].

3.1 Convex Optimization

A convex optimization problem is a problem of the form

$$\begin{aligned} \min \quad & g(x) \\ \text{such that} \quad & h_i(x) \leq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{7}$$

Here, $x \in \mathbb{R}^n$ and both the cost (or 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 that fulfills the following:

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.

3.2 Branch-and-Bound Algorithms

Branch-and-bound algorithms are iterative methods for finding global optima of nonconvex problems. They work by calculating sequences of provable lower bounds that converge to the global minimum. The result of such an algorithm is an ϵ-suboptimal solution, that is, a solution that gives a cost of at most ϵ more than the global minimum. By setting ϵ small enough, we can get arbitrarily close to the global optimum.

The idea of branch and bound is simple, straightforward, and, at the same time, very powerful. Suppose that computing the global minimum *directly* from the objective function $f(t)$ over a set D_0 is too hard. Instead, we first construct a lower bounding function $f_{lower}(t)$ over the domain D_0 and then minimize $f_{lower}(t)$ over D_0. Hence, we require that $f_{lower}(t) \leq f(t)$ for all $t \in D_0$ and that computing the minimum of f_{lower} is (relatively) easy. If the difference between the bounding function and the original cost function when evaluated at the minimum of f_{lower} is less than ϵ, then we have found our ϵ-suboptimal solution. Otherwise, we subdivide (branch) the set D_0 into smaller sets and perform bounding on each subset. If the lower bound for a subset is higher than the value of the best solution found so far, then this subset can be discarded from further consideration. Again, if the difference between the lower bounding function and the cost function (at the optimum of the lower bounding function) is larger than ϵ, we need to subdivide the set further. This process is repeated until the global minimum has been found. In order for this procedure to converge, there are some requirements that the bounding function must fulfill. Let us examine these conditions more closely.

Consider the following (abstract) problem: We want to minimize a nonconvex scalar-valued function $f(t)$ over a set D_0. One may assume that D_0 is closed and compact so a minimum is attained within the domain. For any closed set $D \subset D_0$, let $f_{min}(D)$ denote the minimum value of f on D and let $f_{lower}(D)$ denote the minimum value of the lower bounding function $f_{lower}(t)$ on D, that is, $f_{lower}(D) = min_{t \in D} f_{lower}(t)$. Also, we require that the approximation gap $f_{min}(D) - f_{lower}(D)$ goes uniformly to zero as $\max_{x,y \in D} \|x - y\|$ (denoted $|D|$) goes to zero. Or, in terms of (ϵ, δ), we require that

$$\begin{aligned} &\forall \epsilon > 0, \quad \exists \delta > 0 \quad \text{s.t.} \quad \forall D \subset D_0, \\ &|D| \leq \delta \Rightarrow f_{min}(D) - f_{lower}(D) \leq \epsilon. \end{aligned}$$

If such a lower bounding function f_{lower} can be constructed, then a strategy to obtain an ϵ-suboptimal solution is to divide the domain into sets D_i with sizes $|D_i| \leq \delta$ and compute $f_{lower}(D_i)$ in each set. However, the number of such sets increases quickly with $1/\delta$, and therefore, this may not be feasible. To avoid this problem, a strategy to create as few sets as possible can be deployed. Assume that we know that $f_{min}(D_0) < l$ for some scalar l. If $f_{lower}(D) > l$ for some set D, then there is no point in refining D further since the global minimum will not be contained in D. Thus, D and all possible refinements $D_i \subseteq D$ can be discarded.

The branch-and-bound algorithm begins by computing $f_{lower}(D_0)$ and the actual point $t^* \in D_0$, which is the minimizer. This is our current best estimate of the minimum. If $f(t^*) - f_{lower}(D_0) \leq \epsilon$, then t^* is ϵ-suboptimal and the algorithm terminates. Otherwise, the set D_0 is partitioned into subsets $(D_1, \ldots, D_k)$ with $k \geq 2$ and one gets the lower bounds $f_{lower}(D_i)$ and the points t_i^* in which these bounds are attained. The new best estimate of the minimum is $t^* := \arg\min_{\{t_i^*\}_{i=1}^k} f(t_i^*)$. If $f(t^*) - \min_{1 \leq i \leq k} f_{lower}(D_i) \leq \epsilon$, then t^* is ϵ-suboptimal, and the algorithm terminates. Otherwise, the subsets are refined further; however, the sets for which $f_{lower}(D_i) > f(t^*)$ can be discarded immediately and need not be considered for further computations. This algorithm is guaranteed to find an ϵ-suboptimal solution for any $\epsilon > 0$. The drawback is that the worst case complexity is exponential. In practice, one can achieve relatively fast convergence if the obtained bounds are tight.

In most cases, we do not want to minimize the function over the entire domain but rather to a subset. Then, we also need to check if the problem is feasible in each set D_i—if it is not, the set can be removed from further consideration.

4 Convex Relaxations of $SO(3)$

In this section, we derive relaxations of $SO(3)$ and the group of similarity transformations that allows us to parameterize the transformations affinely. The goal is to obtain easily computable lower bounding functions for cost functions involving euclidean transformations.

Recall from Section 2 that (2) is convex if T depends affinely on its parameters. A common way to parameterize

rotations is to use quaternions (see [23]). Let $q = (q_1, q_2, q_3, q_4)^T$, with $\|q\| = 1$, 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 a point x in $\mathbb{R}^3$, we can rewrite the 3-vector sRx as a vector containing quadratic forms in the quaternion parameters in the following way:

$$sRx = \begin{pmatrix} q^T B_1(x) q \\ q^T B_2(x) q \\ q^T B_3(x) q \end{pmatrix}, \tag{8}$$

where $B_l(x)$, $l = 1, 2, 3$, are 4×4 matrices whose entries depend on the elements of the vector x.

In order to be able to use the branch-and-bound algorithm, it is required that we have bounds on q such that $q_i^L \leq q_i \leq q_i^U$, $i = 1, 2, 3, 4$. In practice, these bounds are usually known, since the scale factor cannot be arbitrarily large. The quadratic forms $q^T B_l(x) q$ contain terms of the form $b_{lij}(x) q_i q_j$ and, therefore, we introduce new variables $s_{ij} = q_i q_j$, $i = 1, \ldots, 4$, $j = 1, \ldots, 4$, in order to linearize the problem.

Now, consider the constraints $s_{ij} = q_i q_j$ or, equivalently,

$$s_{ij} \leq q_i q_j, \tag{9}$$

$$s_{ij} \geq q_i q_j. \tag{10}$$

In the new variables s_{ij}, the cost function becomes convex (cf. (2)), but the domain of optimization given by the above constraint functions is not a convex set.

The constraints in (10) are convex if $i = j$ and (9) is not convex for any i and j. If we replace $q_i q_j$ in (9) with the concave envelope of $q_i q_j$, then (9) 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 than or equal to the original problem. Similarly, we can relax $q_i q_j$ in (10) by its convex envelope and obtain a convex problem that gives a lower bound on the global minimum of the cost function (2).

The convex envelope of $q_i q_j$, $i \neq j$, $q_i^L \leq q_i \leq q_i^U$, $q_j^L \leq q_j \leq q_j^U$, is well known (e.g., [16]) to be

$$(q_i q_j)_{conv} = \max \begin{Bmatrix} q_i q_j^U + q_i^U q_j - q_i^U q_j^U \\ q_i q_j^L + q_i^L q_j - q_i^L q_j^L \end{Bmatrix} \leq q_i q_j \tag{11}$$

and the concave envelope is

$$(q_i q_j)_{conc} = \min \begin{Bmatrix} q_i q_j^L + q_i^U q_j - q_i^U q_j^L \\ q_i q_j^U + q_i^L q_j - q_i^L q_j^U \end{Bmatrix} \geq q_i q_j. \tag{12}$$

Thus, (9) and (10) for $i \neq j$ can be relaxed by the linear constraints

$$-s_{ij} + q_i q_j^U + q_i^U q_j - q_i^U q_j^U \geq 0, \tag{13}$$

$$-s_{ij} + q_i q_j^L + q_i^L q_j - q_i^L q_j^L \geq 0, \tag{14}$$

$$s_{ij} - \left(q_i q_j^L + q_i^U q_j - q_i^U q_j^L \right) \geq 0, \tag{15}$$

$$s_{ij} - \left(q_i q_j^U + q_i^L q_j - q_i^L q_j^U \right) \geq 0. \tag{16}$$

If $i = j$, we need to relax q_i^2 in (9) with its concave envelope. However, this is simply a line $aq_i + b$, where a and b are 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. Fig. 2 shows the upper and lower bounds of s_{11} 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 of how the lower bounds on the problem may converge quite rapidly. Since all of the nonconvex 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 that will serve as a bounding function in our branch-and-bound algorithm. For now, we restrict T to be the set of euclidean (or similarity) transformations, that is, $x \mapsto Rx + t$, where R denotes a rotation matrix, and t denotes a translation vector. In a later section, we will generalize this to include perspective mappings in order to solve the camera pose problem. Let $\hat{q}$ be a 14×1 vector containing the parameters $(q_1, \ldots, q_4, s_{11}, s_{12}, \ldots, s_{44})$. The quadratic forms $q^T B_l(x_i) q$ in (8) can then be relaxed by the terms $b_l(x_i)^T \hat{q}$, where $b_l(x_i)$ is a vector of the same size as $\hat{q}$ whose entries depend on x_i. Note that entries of $b_l(x_i)$ are obtained directly from the entries of $B_l(x_i)$. The vector sRx_i can now be written as $\hat{B}(x_i)\hat{q}$, where $\hat{B}(x_i)$ is a 3×14 matrix with rows $b_l(x_i)^T$. All of the linear constraints in (13)-(16), as well as the bounds on the quaternion parameters q_i, 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 a_js. To improve the bound further, we may also add the constraint that the symmetric matrix S with entries s_{ij} in position (i, j) should be positive definite. (Note that $s_{ij} = s_{ji}$.) Then, the relaxed problem can be written as (cf. (2))

$$\begin{aligned} \min_{\hat{q}, t} \quad & \sum_{i=1}^{m} d_C\left(\hat{B}(x_i)\hat{q} + t\right) \\ \text{subject to} \quad & s_{jj} - q_j^2 \geq 0, \quad j = 1, \ldots, 4, \\ & c - A^T \hat{q} \geq 0, \quad S \succeq 0. \end{aligned} \tag{17}$$

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

$$s_{11} + s_{22} + s_{33} + s_{44} - 1 = 0. \tag{18}$$

Thus, the original nonconvex problem has been convexified by expanding the domain of optimization and, hence, the optimal solution to the relaxed problem will give a lower bound on the global minimum of the original problem.

4.1 Implementation

The implementation of the algorithm was done in Matlab. We basically used the algorithm described in Section 3.2

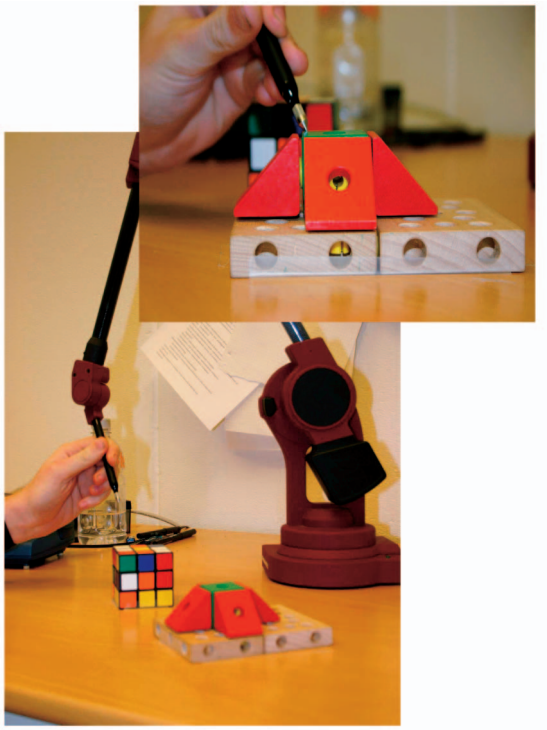

Fig. 1. The experimental setup for the tests done in Section 5.3.

and the relaxations from Section 4. At each iteration, (17) is solved for all rectangles D_i to obtain a lower bound on the function over D_i. To speed up convergence, we use the minimizer t_i^* of (17) as a starting guess for a local minimization of the original nonconvex problem. We then compare the function value at the local optimizer t_i^{loc} and at t_i^*. If any of these values are lower than the current best minimum, the lowest one is deemed the new best minimum. Even though t_i^* is the optimal value of the relaxed problem (17), it is often the case that the function values are larger at t_i^* than at t_i^{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 subdivide it into two. We divide the intervals along the dimension that 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 4D volume of the rectangles. One could argue that it would be sufficient to terminate when the approximation gap (see Section 3.2) 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 [24]). SeDuMi is a free add-on for Matlab that can be used to solve problems with linear, quadratic, and semidefiniteness constraints. For the local minimization, we used the built-in Matlab function **fmincon**.

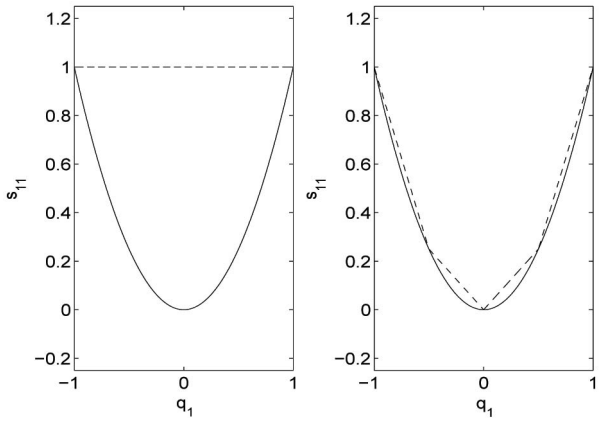

Fig. 2. Upper and lower bounds of s_{11}, which relaxes q_1^2 in the interval $[-1, 1]$. (Left) The initial bound. (Right) When the interval has been subdivided four times. Note that the lower bound is exact since $q_1^2 \leq s_{11}$

5 Applications I: Registration Using Points, Lines, and/or Planes

We will now review the methods of Horn et al. as presented in [4]. Given two corresponding point sets, we want to find the best transformation that maps one set onto the other. The best is taken here to mean the transformation that minimizes the sum-of-square distances between the points, i.e.,

$$\sum_{i=1}^{m} \left\| T(x_{p_i}) - y_{p_i} \right\|_2^2, \tag{19}$$

where x_{p_i} and y_{p_i}, $i = 1, \ldots, m$, 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 the work of Horn et al. [4], it is easily shown that the optimal translation t is given by

$$t = \frac{1}{m}\sum y_{p_i} - R\frac{1}{m}\sum x_{p_i} = \bar{y}_p - R\bar{x}_p. \tag{20}$$

This will turn (19) for the euclidean case into

$$\sum_{i=1}^{m} \left\| R\delta x_i - \delta y_i \right\|_2^2 \tag{21}$$

$$= \sum_{i=1}^{m} (\delta x_i)^T \delta x_i + (\delta y_i)^T \delta y_i - 2(\delta y_i)^T R \delta x_i, \tag{22}$$

with $\delta x_i = x_{p_i} - \bar{x}_p$ and $\delta y_i = y_{p_i} - \bar{y}_p$. Due to the orthogonality of R, this expression becomes linear in R. Now, R can be determined from the singular value decomposition (SVD) of a matrix constructed from δx_i and δy_i. The details can be found in [4].

In this application, 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 results in more difficult optimization problems. In Fig. 1, a measurement device is shown, which generates 3D point

TABLE 1
Methods Available for Estimating the Registration for Different Types of Correspondences and Transformations

Corresp. type	Euclidean/ Similarity	Affine
Point-Point	Horn [4]	Linear Least Squares
Point-Plane	Our algorithm	Linear Least Squares
Point-Line	Our algorithm	Linear Least Squares
Combination	Our algorithm	Linear Least Squares

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.

5.1 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_{π_i}, $i = 1, \ldots, m_\pi$, in another, and we assume that point y_{π_i} lies on plane π_i. Let $d_\pi(x)$ 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 minimize

$$f_\pi(s, R, t) = \sum_{i=1}^{m_\pi} \left(d_{\pi_i}(sRx_{\pi_i} + t)\right)^2. \tag{23}$$

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

$$f_\pi(s, R, t) = \sum_{i=1}^{m_\pi} \left((sRx_{\pi_i} + t - y_{\pi_i}) \cdot n_i\right)^2, \tag{24}$$

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

$$\min_{s \in \mathbb{R}_+, R \in SO(3), t \in \mathbb{R}^3} \sum_{i=1}^{m_\pi} \left(n_i^T (sRx_{\pi_i} + t - y_{\pi_i})\right)^2. \tag{25}$$

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 [4]) in which all measurements are referred to the centroids.

If we let R be any 3×3 matrix, we can consider (25) as minimizing (24) 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 one), there must be numbers $\lambda_{i,j}$ such that

$$\nabla f_\pi(s^*, R^*, t^*) + \sum_{i=1}^{3} \sum_{j=1}^{3} \lambda_{ij} \nabla g_{ij}(s^*, R^*, t^*) = 0. \tag{26}$$

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

$$t = N^{-1} \sum_{i=1}^{m_\pi} n_i n_i^T (y_{\pi_i} - sRx_{\pi_i}), \tag{27}$$

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 cost function (24) that $f_\pi(s, R, \tilde{t}) = f_\pi(s, R, t)$. This means that, in this case, there are infinitely many solutions and, thus, the problem is not well posed.

Next, we state the relaxed problem in the form similar to (17). In this case, the distance function is $d_{\pi_i}(x) = (n_i^T(x - y_{\pi_i}))^2$. If we parameterize the problem using quaternions, as in (17), and introduce the relaxations in terms of the $\hat{q}$ parameters, we get

$$\sum_{i=1}^{m_\pi} d_{\pi_i}(sRx_{\pi_i} + t) = \sum_{i=1}^{m_\pi} \left(n_i^T(\hat{B}_i \hat{q} - y_{\pi_i})\right)^2 = \sum_{i=1}^{m_\pi} (\tilde{B}_i \hat{q} - k_i)^2, \tag{28}$$

where $\tilde{B}_i = n_i^T \hat{B}_i$ is a 1×14 vector and $k_i = n_i^T y_{\pi_i}$. Together with the constraints from (17), this gives us our relaxed lower bounding problem. Note that it is easy to incorporate uncertainty weights into (28) simply by replacing $\tilde{B}_i$ and k_i by $w_i \tilde{B}_i$ and $w_i k_i$, respectively.

5.2 Point-to-Line and Point-to-Point Correspondences

The case of point-to-line correspondences can be treated in a similar way as in the case of point-to-plane correspondences. Let x_{l_i}, $i = 1, \ldots, m_l$, be the measured 3D points and let l_i, $i = 1, \ldots, m_l$, be the corresponding lines. In this case, the sum of the distance functions can be written as

$$d_{l_i}(x) = \left\| (I - v_i v_i^T)(x - y_{l_i}) \right\|^2, \tag{29}$$

where v_i is a unit direction vector for the line l_i and y_{l_i} is any point on the line l_i. Note that the three components of $(I - v_i v_i^T)(x - y_{l_i})$ 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. In the same way as in the point-to-plane correspondence, we can eliminate the t parameters and write our cost function in the form (28).

The case of point-to-point correspondences is the easiest one. Let x_{p_i} be the measured points and y_{p_i} be the corresponding points $i = 1, \ldots, m_p$. The distance function is just

$$d_{y_{p_i}}(x) = \|x - y_{p_i}\|^2. \tag{30}$$

Again, our cost function can be put in the same form as (28).

When we have combinations of different correspondences, we proceed in exactly the same way. Note, however, that we have to add the cost functions before eliminating the t-variables since they will depend on all the correspondences. When doing this, one gets the expression for t as

TABLE 2
Local Minima Found by the Matlab Function **fmincon**

local min point	*cost function value*
(0, -0.078, -1.00, -0.49)	2.1
(0, 0.12, -0.33, -0.93)	2.2
(0.59, 0.042, 0.70, 0.41)	0.00037
(0.66, 0.22, 0.72, 0.23)	0.0018
(0,0,0,0)	65

$$t = M^{-1}\left(\sum_{i=1}^{m_p}(y_{p_i} - Rx_{p_i}) + \sum_{i=1}^{m_l}(I - v_i v_i^T)(y_{l_i} - Rx_{l_i}) + \sum_{i=1}^{m_\pi} n_i n_i^T (y_{\pi_i} - Rx_{\pi_i})\right), \tag{31}$$

where $M = m_p I + \sum_{i=1}^{m_l}(I - v_i v_i^T) + \sum_{j=1}^{m_\pi} n_j n_j^T$. Substituting this into the cost function, we can now find an expression of the type (28).

5.3 Experiments and Results

5.3.1 Local Minima

Nonconvex 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_{π_i} from each plane. Then, the points $x_{\pi_i} = R^T(y_{\pi_i} - t)$ 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, Fig. 3 plots the function values along a line from one local minimum to the global minimum. The values on the line from the first point in Table 2 are in Fig. 3 left and the second point is in Fig. 3 right.

5.3.2 Iterations

Fig. 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 10 point-to-plane, 4 point-to-line, and 4 point-to-point correspondences. The solid line in Fig. 4 left shows the number of feasible rectangles for this problem at each iteration. For each rectangle, the algorithm solves the program (17) using the standard solver SeDuMi [24] with Matlab. The solid line in Fig. 4 right shows the fourth root of the total volume of the rectangles. Recall that this is a 4D 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 total execution time for this problem is around 10 seconds. The second case is the minimal case of seven point-to-plane correspondences. This seems to be the case where our algorithm has the most difficulties. The dashed lines show 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. The total execution time for this problem is around 30 seconds. For comparison, the dotted line in Fig. 4 left shows what the number of rectangles would be if no rectangles were thrown away. In the first case, the algorithm terminated after 38 iterations and, in the second, it terminated after 39, which is a typical behavior of the algorithm.

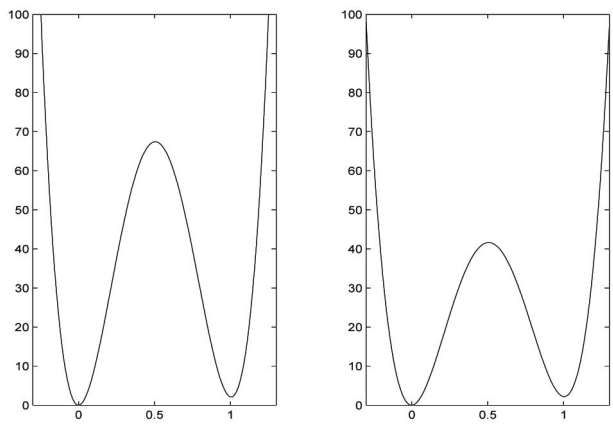

Fig. 3. The function values on two lines between a local minimum and the global minimum.

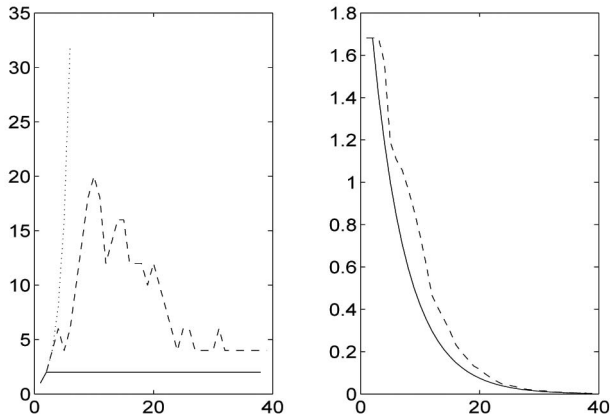

Fig. 4. (Left) The number of feasible rectangles at each iteration for two experiments (solid and dashed lines, see text for details). For comparison, the dotted line shows the theoretical worst case performance where no intervals are discarded. (Right) The fourth root of the total volume of the rectangles.

5.3.3 Experiments on Real Data

Next, we present two experiments made with real data. The experimental setup can be viewed in Fig. 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 5 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 Fig. 1, namely, the Rubik's cube and the toy model. By request of the designer, we will refer to the toy model as the space station.

Rubik's cube experiment. The first experiment was done by measuring on a Rubik's cube. The Rubik's cube contains

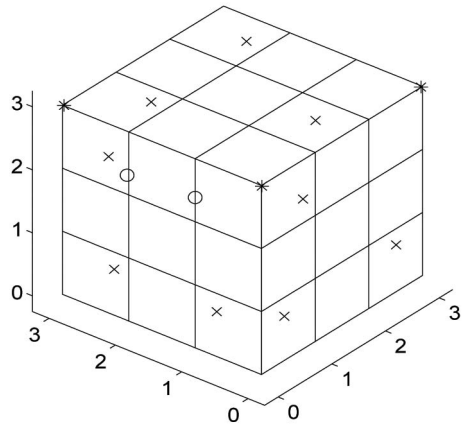

Fig. 5. The model of the Rubik's cube.

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. Fig. 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 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, in Fig. 6 left, 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-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. In Fig. 6 right 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

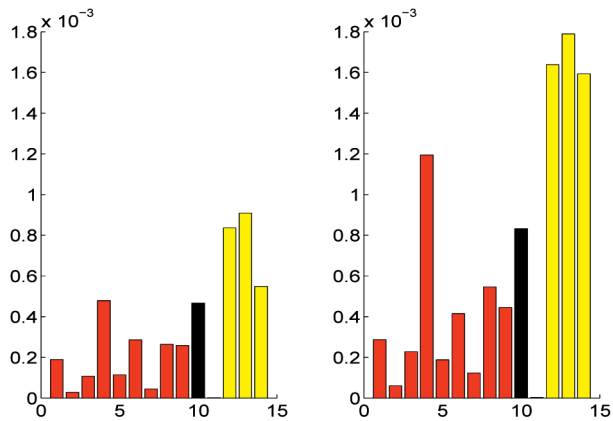

Fig. 6. (Left) Residual errors for all correspondences. (Right) Leave-one-out residuals. (See text for details.)

TABLE 3
Residuals of the Cube Problem

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

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. [4] 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 (31). 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.

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. Fig. 7 shows the model of the space station and the points obtained when applying the estimated transformation to the measured data points.

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

Table 4 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. [4] performs

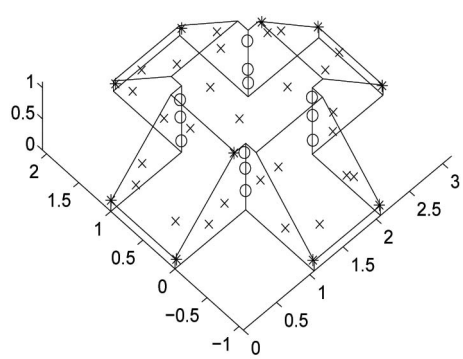

Fig. 7. The model of the space station.

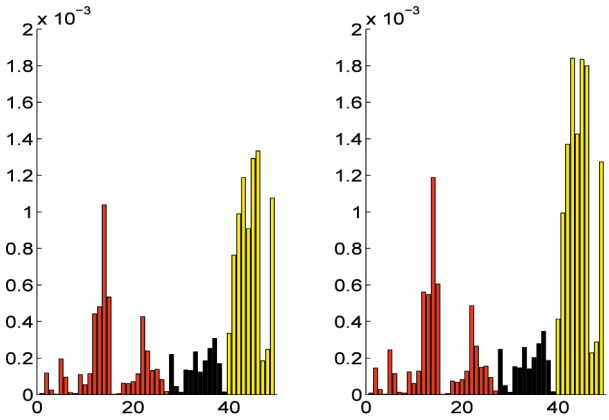

Fig. 8. (Left) Residual errors for all correspondences. (Right) Leave-one-out residuals. (See text for details.)

better since we have included more point-to-point correspondences than in the Rubik's cube experiment. Again, our algorithm has the lowest total residual sum, while that of Horn et al. has the lowest point-to-point residual.

5.3.4 Experiments on Synthetic Data: A Comparison with the Work of Horn et al.

For completeness, we also include an experiment where the euclidean version of the algorithm is tested against the euclidean versions of Horn et al. 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. The residuals for the point-point correspondences will be lower for Horn et al.'s method as expected, while the total error is larger than that for our algorithm. By approximating the euclidean transformation by an affine one, the problem becomes a linear least squares problem. A euclidean transformation can be obtained a posteriori from the affine estimate by taking the closest euclidean transformation. As can be seen in the table, the linear algorithm does ok, but the optimum is not obtained.

6 Applications II: Camera Pose Using Points

One of the basic problems both in computer vision and in photogrammetry is the camera pose estimation problem. Given a number of correspondences between points in 3D space and the images of these points, the pose estimation problem consists of estimating the rotation and position of the camera. Here, the camera is assumed to be calibrated. A typical application is shown in Fig. 9, where the object is to relate a camera to a scanned 3D model of a chair.

TABLE 4
Residuals of the Space Station Problem

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 5
Residuals of the Simulated Data for the Euclidean Algorithm

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

Given m world points X_i (represented by 3-vectors) and corresponding image points $x_i = [x_i^1\ x_i^2]^T$, we want to find the camera translation t and rotation R that minimizes f:

$$f(R,t) = \sum_{i=1}^{m} d(x_i, \pi(X_i))^2 = \sum_{i=1}^{m} \left(\left(x_i^1 - \frac{r_1^T X_i + t_1}{r_3^T X_i + t_3} \right)^2 + \left(x_i^2 - \frac{r_2^T X_i + t_2}{r_3^T X_i + t_3} \right)^2 \right), \tag{32}$$

where $\pi(\cdot)$ is the perspective projection with $R^T = [r_1\ r_2\ r_3]$ and $t^T = [t_1\ t_2\ t_3]$.

We will, in the next section, show how to derive a branch-and-bound algorithm that finds the global optimum for (32).

6.1 Parameterization

Recall that the problem is to find a rotation R and a translation t such that (32) is minimized, which is similar to the registration problem in Section 5. Using unit quaternions for parameterizing the rotation matrices, the cost function (32) can be rewritten as

$$f(q,t) = \sum_{i=1}^{2m} \left(\frac{q^T A_i q + a_i^T t}{q^T B_i q + b_i^T t} \right)^2, \tag{33}$$

where A_i and B_i are 4×4 matrices and a_i and b_i are 3×1 vectors, determined by the data points. This is a nonconvex rational function in seven variables.

To obtain lower bounds on this function, we proceed by formulating a convex optimization problem for which the solution gives a lower bound. The quadratic forms $q^T A_i q + a_i^T t$ and $q^T B_i q + b_i^T t$ contain terms of the form $q_i q_j$. We therefore introduce the new variables $s_{ij} = q_i q_j$ analogously to (9) and (10). In turn, these nonconvex constraints can be relaxed (convexified) using convex and concave envelopes. The terms $(q^T A_i q + a_i^T t)^2$ and $(q^T B_i q + b_i^T t)^2$ in the cost

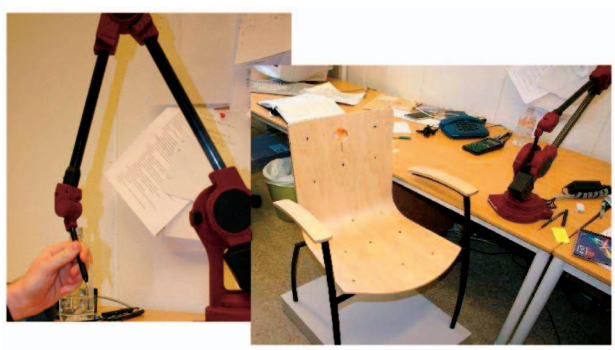

Fig. 9. Chair experiment with 3D scanner.

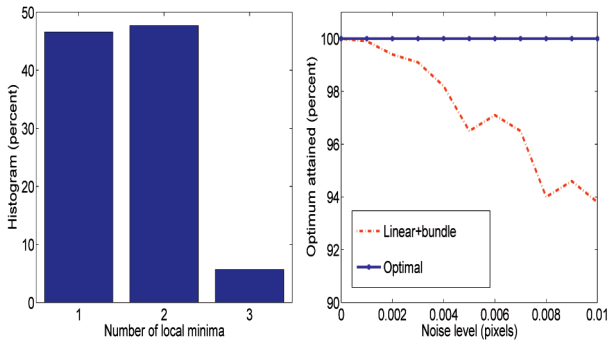

Fig. 10. Local minima plots. (Left) Histogram of local minima with four points. (Right) The percentage of times the global optimum is attained for six points.

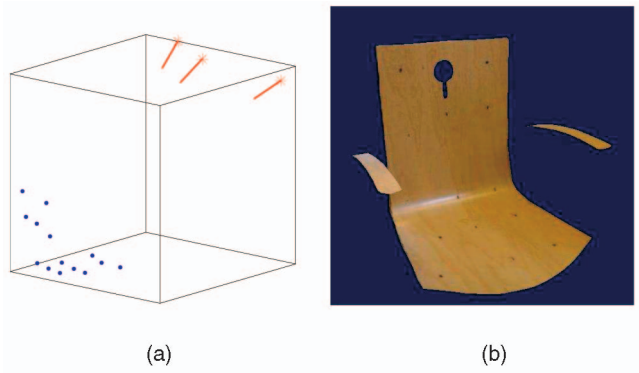

Fig. 11. (a) Reconstructed cameras. (b) Resulting VRML model.

function (33) can now be rewritten as $(\hat{A}_i\hat{q})^2$ and $(\hat{B}_i\hat{q})^2$, respectively. Here, $\hat{q}$ is a 13×1 vector containing the parameters $(t_1, \ldots, t_3, s_{11}, s_{12}, \ldots, s_{44})$ and $\hat{A}_i$ and $\hat{B}_i$ are 1×13 matrices.

It is well known that a rational function with a linear denominator and a quadratic numerator of the form x^2/y for $y > 0$ is convex. Therefore, by replacing the denominator with the affine function $c_i(\hat{B}_i\hat{q}_i) + d_i$ (that is, the concave envelope of $(\hat{B}_i\hat{q})^2$), a convex underestimator of the original cost function is obtained. The constants c_i and d_i are determined from the bounds on q_i and t_i. The full convex function is then

$$f_{lower}(\hat{q}) = \sum_{i=1}^{2m} \frac{(\hat{A}_i\hat{q})^2}{c_i(\hat{B}_i\hat{q}) + d_i}. \tag{34}$$

Minimizing this function subject to the relaxation constraints on s_{ij} yields a lower bound on the cost function (32). At the same time, one can compute the actual value of the cost function f. If the difference is small (less than ϵ), one can stop. As the intervals on q_i, $i = 1, \ldots, 4$, are divided into smaller ones, one can show that the lower bound f_{lower} converges uniformly to (33) (cf. [16]).

To simplify the problem further, we make the following modifications: Without loss of generality, one can choose the world coordinate system such that $X_i = [0\ 0\ 0]^T$ for some i since the cost function is independent of the world coordinate frame. Further, as the cost function is a rational function of homogeneous quantities, one can dehomogenize by setting $t_3 = 1$. Thus, for point i, we obtain

$$d(x_i, \pi(X_i))^2 = \left(x_i^1 - t_1\right)^2 + \left(x_i^2 - t_2\right)^2. \tag{35}$$

Note that this is a convex function. Further, we can restrict the search space by enforcing a maximum error bound on the reprojection error for this point, say, γ_{max} pixels. This results in bounds on t_j such that $x_i^j - \gamma_{max} < t_j < x_i^j + \gamma_{max}$ for $j = 1, 2$.

Since t_3 can be geometrically interpreted as the distance from the camera center to the point X_i (along the optical axis), we have effectively normalized the depth to one. This affects the bounds on q_i as well. Suppose a lower bound on t_3 for the original homogeneous camera is t_3^{low}, then the new bounds for the dehomogenized quaternions become $-1/t_3^{low} \leq q_i \leq 1/t_3^{low}$ for $i = 1, \ldots, 4$. A conservative lower bound on t_3 can easily be obtained by examining the distances between the given world points.

6.2 Experiments and Results

6.2.1 Local Minima

Even though the cost function is highly nonconvex, one might ask if local minima actually occur for realistic scenarios and, if so, how often. Therefore, we first generated random 3D points within the unit cube $[-1, 1]^3$ and a random camera (with principal point at the origin, unit focal length, and skew zero) having a viewing direction toward the origin at a distance of two units. Then, the projected image coordinates were perturbed with independent Gaussian noise with different noise levels. In Fig. 10 left, a histogram of the number of local minima that occur for four points is plotted. The local minima have been computed with random initializations (in total, 100 tries) and the experiment has been repeated 1,000 times. Note that all local minima have positive depths (that is, world points are in front of the camera). There are typically 4-6 additional local minima with negative depths. In Fig. 10 right, the percentage of times the correct global optimum is reached for six points is shown. For our algorithm (optimal), the global optimum is of course always obtained. The traditional way is to apply a linear algorithm (DLT), which requires at least six points, and then do local refinements (bundle adjustment) [2]. As one can see, one might get trapped in a local minimum even for small noise levels.

6.2.2 Experiments on Real Data

The following two subsections present experiments made with real data: the chair and the dinosaur experiment.

Chair experiment. The setup for the chair experiment can be viewed in Fig. 9. We used a MicroScribe-3DLX 3D scanner to measure the 3D coordinates of the black points on the chair. For the first experiment, we took three images of the chair and used the images of the 14 scanned 3D points to calculate the rotation and translation using our method. The intrinsic camera calibration was computed with standard techniques [2]. The reconstructed cameras are shown in Fig. 11. Using the images and the reconstructed cameras, it is easy to get a textured 3D model from the scanned model; see Fig. 11. Table 6 shows the resulting reprojection error measured in pixels. For this particular camera, the resolution was 1,360 × 2,048, and the focal length was 1,782. For comparison, we also tried to solve the

TABLE 6
The RMS Reprojection Error Measured in Pixels, Obtained When Using All 14 Points

Residuals:	*Our Alg.*	*Lin.Method*	*Lin.+Bundle*
camera 1	1.351	10.76	1.351
camera 2	0.939	44.60	10.01
camera 3	0.950	4.741	0.950

problem with a linear method with and without bundle adjustment. The linear method first calculates a projective $3\times$ camera matrix P and then use SVD factorization to find the closest camera matrix such that $\hat{P} = [R|t]$, where R is a scaled rotation matrix. As expected, the linear method without bundle performs poorly. Also, note that for the second chair image, the bundle adjustment yields a local minimum. In Fig. 12, we illustrate the difference between the solutions obtained when not using all points. Here, we regard the solution obtained when using all 14 points as the true solution. In Fig. 12, we have plotted the angles between the principal axis of the resulting camera matrix and the principal axis of the true solution. The red bars are the results when using 4 to 13 points in the first image, the black ones are the results when using the second image, and the yellow ones are from the third image. Note that the computed solution is very similar already from four points to that of using all 14 points.

To illustrate the convergence of the algorithm, Fig. 13 shows the performance of the algorithm for the two first images of the chair. In Fig. 13 left, we have plotted the number of feasible rectangles at each iteration for the first chair image using 6 (red), 9 (green), and 12 (blue) points. In Fig. 13 right is the same plot for the second chair image.

Dinosaur experiment. To further demonstrate the robustness of the algorithm, we have tested the algorithm on the publicly available turntable sequence of a dinosaur; see Fig. 14 for one of the 36 images. The full reconstruction of 3D points and camera motion are also available, obtained by standard structure and motion algorithms [2]. For each of the 36 views, we have taken four randomly chosen points visible in that image and then estimated the camera pose. The resulting camera trajectory including viewing direction is compared to the original camera motion in Fig. 14 (right). Note that, even though only four points have been used, the camera motion (*blue curve*) is very close to that of full bundle adjustment using all points (*red curve*).

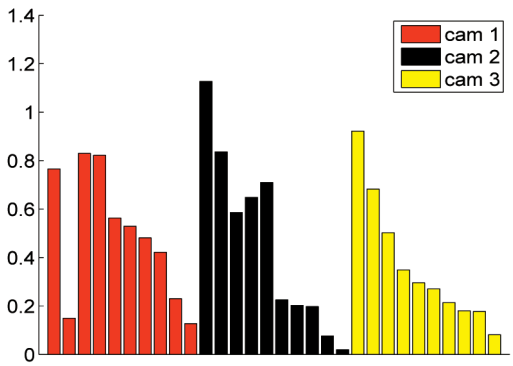

Fig. 12. Angles in degrees between the principal axis of the optimal solution with 14 points and the solutions with 4-13 points for the three chair experiments.

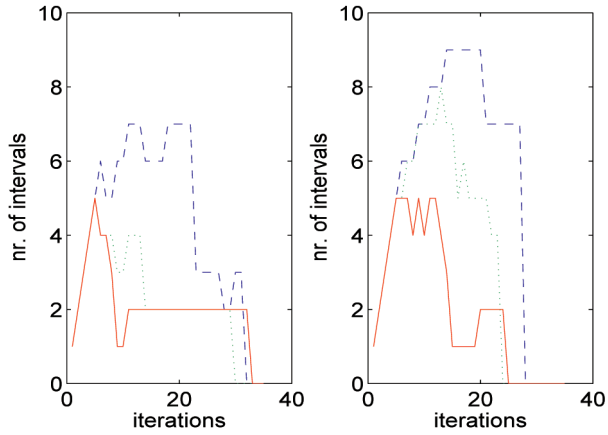

Fig. 13. Convergence of the branch-and-bound algorithm. The number of feasible rectangles at each iteration for two of the chair images using 6 (red), 9 (green), and 12 (blue) points.

7 Conclusions and Future Work

Optimization over the manifold of euclidean transformations is a common problem for many applications. The resulting mathematical optimization problems are typically nonconvex and hard to solve in general. In this paper, we have shown how global optimization techniques can be applied to yield ϵ-suboptimal solutions, where ϵ can be arbitrarily small.

Future work includes investigating 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 for general shapes in order to improve the robustness with respect to local minima.

Acknowledgments

This work has been funded by the European Commission's Sixth Framework Programme under Grant 011838 as part of the Integrated Project SMErobot, the Swedish Research

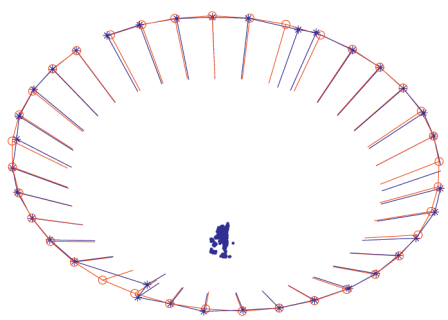

Fig. 14. The recovered camera motion for the dinosaur experiment. Camera motion from full bundle adjustment (*red curve*) and using only four points (*blue curve*).

Council (Grant 2007-6476), the Swedish Foundation for Strategic Research (SSF) through the programs Future Research Leaders and Vision in Cognitive Systems (VISCOS), and the European Research Council (GlobalVision Grant 209480).

REFERENCES

[1] E.H. Thompson, "An Exact Linear Solution of the Problem of Absolute Orientation," vol. 15, no. 4, pp. 163-179, 1958.
[2] R.I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision,* second ed. Cambridge Univ. Press, 2004.
[3] *Manual of Photogrammetry,* C. Slama, ed., fourth ed. Am. Soc. Photogrammetry, 1984.
[4] B.K. Horn, H.M. Hilden, and S. Negahdaripour, "Closed-Form Solution of Absolute Orientation Using Ortonormal Matrices," *J. Optical Soc. Am. A,* vol. 5, pp. 1127-1135, 1988.
[5] K. Kanatani, "Unbiased Estimation and Statistical Analysis of 3-D Rigid Motion from Two Views," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 1, pp. 37-50, Jan. 1993.
[6] P. Besl and N. McKay, "A Method for Registration Two 3-D Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 232-256, Feb. 1992.
[7] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically Stable Sampling for the ICP Algorithm," *Proc. Fourth Int'l Conf. 3-D Digital Imaging and Modeling,* 2003.
[8] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," *Proc. IEEE Int'l Conf. Robotics and Automation,* vol. 3, pp. 2724-2729, 1991.
[9] H. Pottman, Q. Huang, Y. Yang, and S. Hu, "Geometry and Convergence Analysis of Algorithms for Registration," *Int'l J. Computer Vision,* vol. 67, no. 3, pp. 277-296, 2006.
[10] J.A. Grunert, "Das Pothenot'sche Problem in Erweiterter Gestalt; Nebst Bemerkungen über Seine Anwendung in der Geodäsie," *Grunert Archiv der Mathematik und Physik,* vol. 1, no. 3, pp. 238-248, 1841.
[11] R.M. Haralick, C.N. Lee, K. Ottenberg, and M. Nolle, "Review and Analysis of Solutions of the 3-Point Perspective Pose Estimation Problem," *Int'l J. Computer Vision,* vol. 13, no. 3, pp. 331-356, Dec. 1994.
[12] I.E. Sutherland, "Sketchpad: A Man-Machine Graphical Communications System," Technical Report 296, MIT Lincoln Laboratories, 1963.
[13] L. Quan and Z. Lan, "Linear N-Point Camera Pose Determination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 8, pp. 774-780, Aug. 1999.
[14] B. Triggs, "Camera Pose and Calibration from 4 or 5 Known 3D Points," *Proc. Eighth Int'l Conf. Computer Vision,* pp. 278-284, 1999.
[15] B. Kolman and R.E. Beck, *Elementary Linear Programming with Applications.* Academic Press, 1995.
[16] H.S. Ryoo and N.V. Sahinidis, "Analysis of Bounds for Multilinear Functions," *J. Global Optimization,* vol. 19, pp. 403-424, 2001.
[17] T. Breuel, "Implementation Techniques for Geometric Branch-and-Bound Matching Methods," *Computer Vision and Image Understanding,* vol. 90, no. 3, pp. 258-294, 2003.
[18] T. Breuel, "Geometric Aspects of Visual Object Recognition," PhD dissertation, Massachusetts Inst. of Technology, 1992.
[19] F. Kahl, S. Agarwal, M.K. Chandraker, D.J. Kriegman, and S. Belongie, "Practical Global Optimization for Multiview Geometry," *Int'l J. Computer Vision,* vol. 79, no. 3, pp. 271-284, 2008.
[20] C. Olsson, F. Kahl, and M. Oskarsson, "The Registration Problem Revisited: Optimal Solutions from Points, Lines and Planes," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 1206-1213, 2006.
[21] C. Olsson, F. Kahl, and M. Oskarsson, "Optimal Estimation of Perspective Camera Pose," *Proc. 18th Int'l Conf. Pattern Recognition,* pp. 5-8, 2006.
[22] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge Univ. Press, 2004.
[23] S. Altmann, *Rotations, Quaternions and Double Groups.* Clarendon Press, 1986.
[24] J.F. Sturm, *Using Sedumi 1.02, a Matlab Toolbox for Optimization over Symmetric Cones,* 1998.

Carl Olsson received the MSc degree in electrical engineering from the University of Lund in 2004. His master's thesis was devoted to the study of shape optimization for minimizing drag in incompressible laminar flows. He is currently a PhD student in the Centre for Mathematical Sciences, Lund University. His main research interests are optimization with applications in computer vision, robotics, and related areas. In his spare time, he enjoys playing the guitar and listening to rock'n roll. He is a student member of the IEEE.

Fredrik Kahl received the MSc degree in computer science and technology in 1995 and the PhD degree in mathematics in 2001. His thesis was awarded the Best Nordic Thesis Award in pattern recognition and image analysis 2001-2002 at the Scandinavian Conference on Image Analysis 2003. He was a postdoctoral research fellow at the Australian National University from 2003 to 2004 and at the University of California, San Diego, from 2004 to 2005. He is currently an associate professor in the Centre for Mathematical Sciences, Lund University, Sweden. His primary research areas include geometric computer vision problems and optimization methods for both continuous and discrete domains. In 2005, together with Didier Henrion, he received the ICCV 2005 Marr Prize for work on global optimization methods applied to geometric reconstruction problems.

Magnus Oskarsson received the MSc degree in engineering physics and the PhD degree in mathematics from the University of Lund, Sweden, in 1997 and 2002, respectively. His thesis work was devoted to computer vision with applications for autonomous vehicles. He is currently an assistant professor in the Centre for Mathematical Sciences, Lund University, where his teachings include undergraduate and graduate courses in mathematics and image analysis. His main research interests are in algebra and optimization with applications in computer vision, cognitive vision, and image enhancement. He is a member of the IEEE.