

Robust Optimal Pose Estimation

Olof Enqvist and Fredrik Kahl

Lund University, Box 118, 22100 Lund, Sweden

Abstract. We study the problem of estimating the position and orientation of a calibrated camera from an image of a known scene. A common problem in camera pose estimation is the existence of false correspondences between image features and modeled 3D points. Existing techniques such as RANSAC to handle outliers have no guarantee of optimality. In contrast, we work with a natural extension of the L_∞ norm to the outlier case. Using a simple result from classical geometry, we derive necessary conditions for L_∞ optimality and show how to use them in a branch and bound setting to find the optimum and to detect outliers. The algorithm has been evaluated on synthetic as well as real data showing good empirical performance. In addition, for cases with no outliers, we demonstrate shorter execution times than existing optimal algorithms.

1 Introduction

Camera pose estimation is a well studied problem in both computer vision and photogrammetry [1, 2] and one of the earliest references on the topic dates back to 1841 [3]. The problem is important on its own as a core problem within the field of multiple view geometry, but it also appears as a subproblem for many other vision applications, like motion segmentation [4], object recognition [5–7], and more generally model matching and fitting problems, see [4–10]. Yet, previous approaches for solving the camera pose problem have not been able to solve the problem in the presence of outliers with a guarantee of global optimality. In this paper, an efficient algorithm is developed for achieving these criteria.

Often the determination of camera pose is divided into two steps. First feature points are extracted from the image and matched to a 3D model of the scene. In the next step, correspondence pairs from the matching procedure are used to estimate the position and orientation of the camera. It is this second step that we will consider in this article for the pinhole camera model.

Fitting problems with outliers are known to be hard optimization problems. We formulate the pose problem as a mixed integer problem and apply branch and bound to find the optimal solution. For such problems it is important to have bounding functions that are fast to compute and give hard constraints on the solution. We derive a bounding function which fulfills both these requirements using a classical result from geometry. This is the key component for the main contribution of the paper: an efficient, robust and optimal solution to the pose problem.

1.1 Related work

The minimal number of correspondence pairs necessary to obtain a solution for the camera pose problem is three and it is well-known that there may be up to four solutions in this case, see [11] for an overview. For four points, it can be solved linearly [12] and for six points or more, one can use the standard Direct Linear Transform (DLT) scheme [1], but these approaches optimize an algebraic cost function. Refinement using non-linear optimization techniques like Levenberg-Marquardt is of course possible, but the solution may get trapped in a local minimum. Recent work in multiple view geometry has focused on obtaining global solutions, see [13] for an overview, and this paper is no exception in that sense.

The first globally optimal algorithm for this problem using a geometric error norm was presented in [14]. They also investigate the problem of local minima for the pose problem and show that this is indeed a real problem for small number of correspondences. For large number of correspondence pairs or small noise levels, the risk of getting trapped in a local minimum is small. This is our experience as well. In their work the L_2 norm of the reprojection errors is used, but the algorithm converges rather slowly. In [15], the authors choose to work with the L_∞ norm instead and present a more efficient algorithm that finds the optimum by searching in the space of rotations and solving a series of second order cone programs. The reported execution times for both these algorithms are in the order of several minutes, while our algorithm performs the same task within a few seconds. Perhaps more importantly though is that our algorithm is capable of discarding outliers. If some correspondences are incorrect, then fitting a solution to all data might give a very bad result.

A method for detecting outliers for L_∞ solutions was given in [16] but it only applies to quasiconvex problems. The uncalibrated pose problem - often referred to as camera resectioning - is quasiconvex, but the calibrated pose problem is not. Further, the strategy in [16] for removing outliers is rather crude - all measurements that are in the support set are discarded. Hence, inlier correspondences may also be removed. Possible solutions to this problem was given in [17, 18] for outliers, but they are computationally expensive and also restricted to quasiconvex problems. Another well-known approach for estimating camera pose in cases where it is hard to find correct correspondences is to apply RANSAC-type algorithms [19]. Such algorithms offer no type of optimality.

Our work is also related to the rich body of literature on matching problems, see [4–10]. Many of these algorithms are quite sophisticated and have been an inspiration to our work. However, some do not guarantee any kind of optimality [4, 10], while others do [8, 9]. Another difference to our work is that simplified camera models like affine approximations are used [6–9]. Other drawbacks include simplified cost functions which are not based on reprojection errors.

2 Problem Formulation

Given an image from a calibrated camera and a 3D model of the scene, we want to estimate the position C , and orientation \mathbf{R} , of the camera. If we have more than three points this is generally an overdetermined problem. Thus we cannot expect an exact solution and need to decide on an error measure. The widely accepted standard is to consider some error norm on the reprojection errors. We will follow this standard as well.

Since the camera is calibrated we can choose to represent the image as a sphere (rather than an image plane) and detected feature points in the image as points on the sphere or unit vectors, x_j . We also have a set of hypothetical correspondences between points in the model and points in the image (X_i, x_i) . We will work with two natural extensions of the L_∞ norm to the outlier case.

Problem 1 *For a prescribed threshold $\varepsilon > 0$, find the rotation \mathbf{R} and position C that maximizes $|I|$ where I is the set of indices i such that*

$$\angle(x_i, \mathbf{R}(X_i - C)) < \varepsilon. \quad (1)$$

Here $\angle(u, v)$ denotes the angle between vectors u and v . Thus, we are seeking the largest consistent subset of all hypothetical correspondences such that the angular error is less than ε for each correspondence.

An alternative would be to try to minimize ε such that $|I|$ is larger than some predefined threshold, K , an approach used in [17] for the triangulation problem. Thus we seek the smallest ε such that there is at least K correspondences that satisfy (1). In cases with no outliers, this becomes the standard L_∞ norm formulation.

Problem 2 *For a prescribed $K \in \mathbb{N}$, find the rotation \mathbf{R} and position C that solves*

$$\min \varepsilon \quad \text{s.t.} \quad |I| \geq K.$$

where I is the set of indices i such that

$$\angle(x_i, \mathbf{R}(X_i - C)) < \varepsilon.$$

In either formulation, one has to specify a modeling parameter (ε or K). It may seem more convenient to prescribe ε since one usually has some idea of the noise level in the measurements. A side effect of this choice is that there may be a whole set of solutions in the space of Euclidean motions that satisfies (1). This can be regarded as a good feature as one gets uncertainty estimates of the camera positions for free, but in other circumstances, it may be preferable to have a unique solution. From an algorithmic point of view we have found that it is more practical to specify the maximum number of outliers and optimize over ε . We have experimented on both formulations.

3 Pumpkin Constraints

In this section we derive approximate constraints for L_∞ optimality. By considering the angle between two image vectors x and y we get constraints on the camera position that do not involve the orientation.

Assume for a moment that we have no noise and no false correspondences. Then, given two world points, X and Y and the corresponding image vectors x and y , we have

$$\angle(X - C, Y - C) = \angle(x, y). \quad (2)$$

This yields a constraint on the camera position C , that we intend to study more closely. On the left hand side of this equation, X and Y are given by our model of the scene. On the right hand side, we have the angle between two image vectors, which is simply a constant that can be calculated from the measured image coordinates. We denote it by α . We seek the points C in space that form exactly the angle α with X and Y .

So for which points does this hold? First consider a plane through X and Y . It is a well known result from classical geometry that if X and Y are two fixed points on a circle, then the angle $XC Y$ is equal for all points C on the arc from X to Y . This tells us that the points C such that $XC Y$ is equal to α form two circular arcs in the plane as shown in Figure 1. Also note that if for some other camera position \bar{C} the angle $X\bar{C}Y$ is larger than α , then \bar{C} lies in the set enclosed by the two arcs.

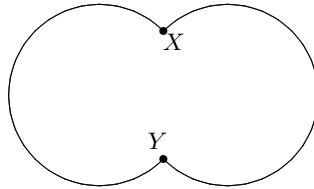


Fig. 1. The points C for which $XC Y = \alpha$ form two circular arcs in the plane.

In space the points C for which $XC Y = \alpha$ form a surface which is obtained by rotating the circular arcs around the line through X and Y (Figure 2). Like in the planar case any \bar{C} such that $X\bar{C}Y > \alpha$ lies in the set enclosed by this surface. We define

$$M_\alpha(X, Y) = \{C \in \mathbb{R}^3 : XC Y > \alpha\}.$$

If $\alpha < \pi/2$, this set will be non-convex and shaped like a pumpkin.

Now assume we have found an optimum (\mathbf{R}, C) in the sense given by Problem 1 or Problem 2. Let X and Y be two points that satisfy (1). Then by the spherical version of the triangle inequality,

$$\angle(X - C, Y - C) \leq \alpha + 2\varepsilon$$

$$\angle (X - C, Y - C) \geq \alpha - 2\varepsilon.$$

Note that these constraints are weaker than the L_∞ norm, so they could be used to produce tight bounds on the L_∞ optimum. We propose a branch and bound algorithm, evaluating these constraints for smaller and smaller boxes.

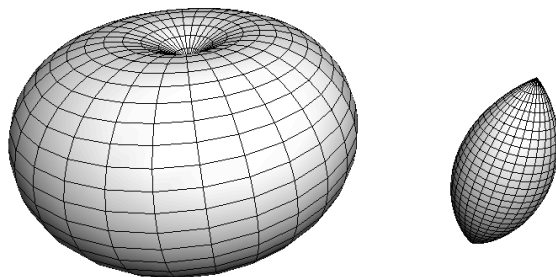


Fig. 2. M_α for angles less than (*left*) and larger than (*right*) $\pi/2$ respectively.

4 An Algorithm

The constraints from the previous section have the important characteristic that they do not involve the orientation of the camera. Thus we can seek the optimal camera centre by a branch and bound search over a subset of \mathbb{R}^3 . For each camera centre we evaluate the pumpkin constraints and get a bound on the L_∞ norm of this specific position. As we will show later, evaluating constraints of this type can be done quickly so there is chance of a reasonably fast algorithm. More importantly, the fact that each constraint depends only on two correspondence pairs makes it possible to handle outliers. Suppose, for example, that a certain camera centre violates the constraints from k disjoint pairs (X_i, X_j) . Then a solution with this camera centre will have at least k outliers.

We will now present an algorithm to estimate the optimal camera position with respect to any of the error measures presented in Section 2. Starting with a rough lower bound on the optimum, we use a branch and bound approach to create tighter and tighter bounds on the optimal solution. The aim is to restrict the solution to a small enough volume in space. The box below presents the overall structure of the algorithm and we will now go through the main steps in greater detail.

Algorithm 1

Initialize to get a bounded set in \mathbb{R}^3 .

Iterate until desired precision is reached:

1. *Pick a box from the queue.*
2. *Evaluate all pumpkin constraints for this box.*
3. *Try to detect and remove outliers.*
4. *Try to discard the box.*
5. *If the box cannot be discarded:*
 - *Divide the box and update the queue.*
 - *Try to update the lower bound on the optimum.*
6. *Remove the box from the queue.*

Initialization. To start our branch and bound loop, we need a bounded set of possible camera centres. In many cases we can get this from our knowledge of the scene. For example, we might know the size of the building we are working with. Otherwise we propose an initialization scheme by considering the pumpkin constraints directly on sets of the kind $|C_i| > b$ where C_i is one of the coordinates of the camera centre. Since for most pumpkin constraints $\alpha > 0$, they will be bounded sets in space and thus there will be a maximal b such that they intersect $|C_i| > b$.

Evaluating the constraints. Consider two world points X and Y and their corresponding image points x and y . If the angle $\alpha = \angle(x, y)$ is smaller than $\pi/2$, the constraint from Section 3 will be non-convex. So how do we determine whether any part of a given box (in our branch and bound algorithm) intersects this pumpkin?

Remember the way we derived the pumpkin constraints, a circular arc was rotated around the line through X and Y . Now consider the curve formed by the circle centre when rotating. This is itself a circle centred around $(X + Y)/2$ and the pumpkin is simply all points in space with less than a certain distance to this circle. To check if a given box intersects such a pumpkin we inscribe the box in a sphere and calculate the shortest distance from the sphere to the central circle of the pumpkin, see Figure 3. We know that the vector from the centre of the sphere to the closest point on the circle is perpendicular to the circle at this point. Thus we can find the shortest distance by studying a plane through the centre of the sphere and the centre of the circle (being $(X + Y)/2$) that intersects the circle under a right angle. A similar discussion tells us how to handle the case $\alpha > \pi/2$.

Detecting outliers and discarding boxes. Though the method we present can compete with other optimal methods when it comes to speed, a greater advantage is the ability to deal with outliers. The basic idea is very simple. If

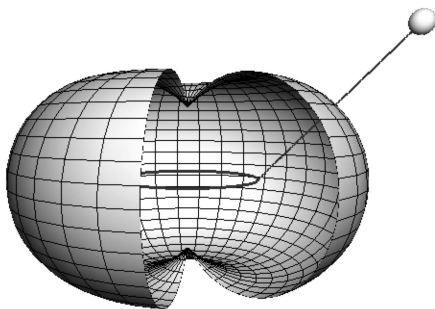


Fig. 3. The central circle inside of M_α centred around $(X + Y)/2$ and the line segment between the central circle and the sphere (circumscribing a box).

there are no outliers, we only need to find one violated constraint to discard a box in the branch and bound. In the case of outliers we need to find a larger number of violated constraints.

Actually, it is often even possible to pinpoint which correspondences are the outliers. Suppose a certain point is involved in, say, 10 different violated constraints. If this point were to be an inlier, then the other 10 points would have to be outliers. In this manner it is possible for each box in the branch and bound queue to keep track of which correspondences one needs to consider.

Updating the lower bound. To detect outliers and discard boxes in the branch and bound algorithm, we need some lower bound on the number of inliers of the optimal solution. At the initial stage this could be an educated guess or the result of an approximate method. For example one could use a RANSAC solution as a lower bound on the optimal solution. However, as the algorithm progresses we need to update this bound. In the next section we will show how to do this formally with respect to the error measures of Section 2. Here, we give an approximate method that works well in practice.

Consider a box that could not be eliminated in step 4 of Algorithm 1. A hypothesis is that the optimal camera position lies close to the centre of this box. Following this idea, we fix the camera position to the centre of this box and estimate the camera rotation. Since we have already eliminated most of the outliers we can estimate the rotation (for the remaining points) with Horn's closed form solution using unit quaternions [20]. Now when we have fixed both camera position (to the box centre) and rotation (to the Horn solution) we estimate the error as in Section 2. This gives us a new lower bound on the optimal solution.

5 Optimal Rotation

In the previous section, we presented a practical approach to find the camera centre that is optimal with respect to the L_∞ norm of the reprojection errors. It works to find smaller and smaller sets which are guaranteed to contain the optimal solution. We say that the algorithm has converged when a set is found which is small enough for the user's needs. However, Algorithm 1 is not guaranteed to converge to any prescribed precision. In this section, we discuss how to securely find the optimum by searching the space of rotations, whence giving sufficient conditions for obtaining globally optimal solutions.

It should be noted that in our experiments, the basic Algorithm 1 produces the desired precision and solves the optimization problem. Thus the modifications of this section are only required to guarantee convergence. Though we have implemented and tested Algorithm 2, the performance in Section 6 are from using Algorithm 1 with Horn's method to estimate rotations.

Searching rotation space. We will use the same approach to refine two different steps in Algorithm 1, discarding boxes in step 4 and updating the lower bound on the optimum in step 5. We will mainly describe the second problem here, but the same approach can be used for the first problem. Just as in the previous section, we start with a box that could not be discarded. To get a lower bound on the optimum, we fix the camera position to the centre of this box. Assuming that we use the formulation from Problem 1, we want to solve

Problem 3 *Given a camera position \tilde{C} and $\varepsilon > 0$, find the rotation \mathbf{R} that maximizes $|I|$, where I is the set of inliers as defined in Problem 1.*

Algorithm 2

Start with a set of spherical triangles covering the sphere.

Iterate until desired precision is reached:

1. *Pick a triangle from the queue.*
2. *Try to discard the triangle.*
3. *If the triangle cannot be discarded:*
 - *Divide the triangle and update the queue.*
 - *Try to update the lower bound on the optimum.*
4. *Remove the triangle from the queue.*

Our measured image is represented with unit vectors x_i . Since we have fixed the camera centre to \tilde{C} and have a 3D model of the scene we can calculate a modeled image, i.e. what the image should look like, given the 3D model and the fixed camera centre. We represent the modeled image with unit vectors

$$\tilde{x}_i = (X_i - \tilde{C}) / \|X_i - \tilde{C}\|.$$

Problem 3 consists in finding the rotation that maps as many of the \tilde{x}_i 's as possible to the corresponding x_i 's, within the prescribed tolerance ε . As a parameterization for rotations we use a unit vector ζ and an angle γ . The unit vector specifies the point in the modeled image that will be mapped to the z axis of the measured image and γ is the rotation angle around this axis.

To find the optimal rotation, we propose a branch and bound search over the possible ζ 's. The unit sphere is divided into spherical triangles. A simple test is then used to determine if a given triangle can contain the optimal solution, if so it is divided into four new triangles. Figure 5 shows the evaluated triangles for one search.

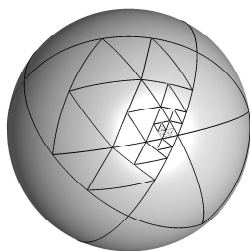


Fig. 4. Plot of the considered triangles in a particular search over the sphere. The data used are from the Notre Dame data (see Section 6).

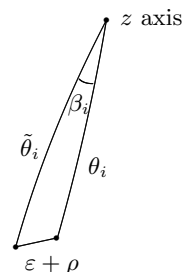


Fig. 5. The figure shows the modeled image superposed with the measured. The lower triangle corners are the modeled image point and the measured respectively. Their distance must not be larger than $\varepsilon + \rho$.

Discarding triangles. It remains to discuss how to discard triangles in the branch and bound search. Given a (spherical) triangle of possible ζ 's we want to determine if the optimal solution can lie within this triangle. Let ζ_c be the centre of the triangle and ρ the largest distance from the centre to a point in the triangle. It is easy to show that the reprojection errors are Lipschitz continuous as functions of ζ . This implies that if the optimal rotation is in the considered triangle mapping k points with error less than ε , then there is a rotation with $\zeta = \zeta_c$ that maps k points with error less than $\varepsilon + \rho$. Thus we assume that $\zeta = \zeta_c$ and look for a γ such that as many points as possible are mapped with error less than $\varepsilon + \rho$.

First transform all image points \tilde{x}_i in the modeled image to the measured with a rotation R that maps ζ to the z axis. Now that we have all points in the same coordinate system we switch to spherical coordinates

$$x_i = (\sin \theta_i \cos \phi_i, \sin \theta_i \sin \phi_i, \cos \theta_i)$$

and similarly for the modeled image points. Note that θ_i is the angle between the image point and the z axis. The γ -rotation will just add to the ϕ -angles of

the modeled image points. Thus, if there were no noise or errors we would have $\theta_i = \tilde{\theta}_i$ and $\phi_i = \gamma + \tilde{\phi}_i$ for all i 's. In practice this will not be the case. Instead we seek γ such that our error measure is minimized. This is equivalent to

$$|\tilde{\phi}_i + \gamma - \phi_i| < \beta_i \quad (3)$$

for as many i 's as possible. The angle β_i depends on the tolerance ε and the triangle size ρ . Figure 5 illustrates how to calculate β_i using the spherical law of cosines.

We have thus to decide a γ that satisfies (3) for as many i 's as possible. Since the constraints are intervals this is rather straightforward and we will not go into the details here. The maximal number of satisfied constraints gives us an upper bound on the number of inliers of the best solution in the current triangle. If it is lower than the number of inliers of the best solution so far, we can discard this triangle.

Updating the lower bound. If a triangle cannot be discarded as above, we try to update the lower bound on the optimum. As a candidate we use $\zeta = \zeta_c$, being the centre of the triangle that we could not discard. Then we repeat the procedure above, but without the extra tolerance ρ .

6 Experiments

To make it easier to compare the performance of our method to existing ones we have tested it on data without as well as with outliers. The timings presented are for a simple C++ implementation on a 1.2 GHz iBook G4.

For some data sets, like the Notre Dame scene described below, the size of the different scenes varies dramatically. To get a reasonable measure of performance we have normalized the size of the different scenes so that both 3D points and the camera centre fit into a $10 \times 10 \times 10$ voxel cube. The same normalization was done on the dinosaur data set.

Dinosaur data. As a first practical experiment of our algorithm, we ran it on the well-known and publicly available dinosaur images for which a 3D reconstruction of the object is available. In total the data consists of 328 object points and 36 images. Between 22 and 154 points are visible in each view and there are no outliers.

For all 36 views of the dinosaur data we estimated the camera pose. The maximal reprojection errors of the presented solutions are between 0.0003 and 0.001 radians. The median computation time was 0.7 s and the maximal computation time 4.8 s. In Figure 6, the resulting camera trajectory is plotted. As a comparison, we have included the result using standard bundle adjustment of the L_2 error cost function. Note that the two camera trajectories differ little which could be expected as the geometry is not very challenging.

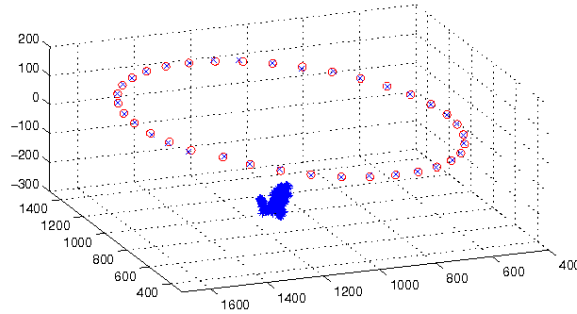


Fig. 6. Estimated camera motion for the dinosaur data using our approach (blue x's) and bundle adjustment (red o's).

Notre Dame data. We also tested our algorithm on the Notre Dame data set (which was used in [21]) consisting of 595 images with computed 3D scene structure. From each of the 595 images we picked sets of between 6 and 30 points on which we tested our algorithm. We ran a fixed number of iterations and measured the remaining volume in the space of camera positions. In 96% of the cases the remaining volume of camera positions was less than 0.01. The median computation time was 0.6 s but since some images took quite a long time the average computation time was as much as 4.5 s. In the cases where we were not so successful in terms of execution times the point configurations were almost degenerate. The camera was far away from the scene compared to the size of the scene. Figure 7 shows the image points and the calculated camera positions in two cases, one typical scene and one almost degenerate.

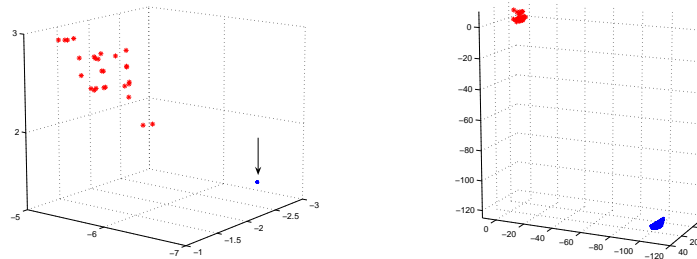


Fig. 7. Two examples from the Notre Dame data. The left plot shows 3D points (red *'s) and feasible camera positions (blue) in a standard case. The right plot shows one of the difficult cases with slower convergence.

Toy truck data. We also did some testing on real data with outliers. Figure 8 shows one of 18 images of a scene with a toy truck. We used two of the views to calculate a 3D model of the scene and then matched the other images to this model to get hypothetical correspondences for our experiment. Matching hypotheses were obtained with SIFT descriptors. The number of matched points in each case varied between 23 and 60 and the amount of outliers was around 20%. The average execution time was 8.47 s and the average remaining volume was 0.0005 compared to a scene of size approximately $1 \times 1 \times 1$.



Fig. 8. Example of a real case with outliers.

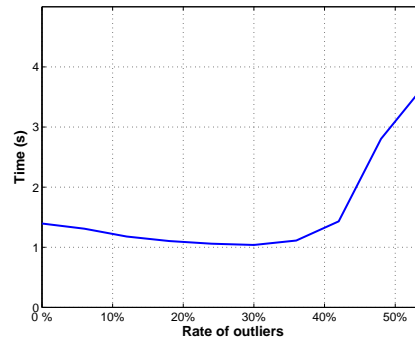


Fig. 9. Mean run times for the synthetic experiments with outliers.

Synthetic data. To get a larger amount of data with outliers we also evaluated our algorithm on synthetic data. A set of 50 random 3D points were scattered in a cube with side length 10 units. The camera was placed at distance 10 from the cube. Image vectors were computed and random angular noise with truncated normal distribution (original standard deviation 0.0015, truncated at 0.002) was added to the image vectors. Then a number of the 3D points were exchanged for outlier data with the same distribution. In this case the formulation from Problem 2 was used. The algorithm was run until the remaining volume was less than 0.001. Figure 9 shows the mean run times over 50 experiments for different rate of outliers.

The results indicate that our approach is applicable even with a considerable amount of outliers and that the execution times are competitive. It might seem strange that the run times in Figure 9 initially decrease with increasing rate of outliers. However, since all examples have the same total number of correspondences, the examples with many outliers get easier as the outliers are eliminated. The experiment shows that in this setting and with reasonable rates of outliers, removing the outliers is not the most time-consuming task.

7 Conclusions

The proposed algorithm is a first step towards practical L_∞ pose estimation with outliers. Experiments demonstrate the validity of the approach by computing globally optimal estimates while discarding outliers at the same time. We also believe that this work can be generalized to other problems in computer vision, especially multiview geometry problems. For instance, exactly the same idea can be applied to the triangulation problem and this gives us immediately an optimal algorithm for outlier removal.

The speed of the present algorithm is already attractive for many practical vision problems, but there is still work to be done in this direction for handling a larger percentage of outliers. Sorting the order of the pumpkin constraints that are processed in the branch and bound process based on violation performance will improve speed since non-interesting boxes are cut off early. Another research direction we intend to pursue is the possibility of a GPU implementation as the algorithm is easily parallelizable.

8 Acknowledgments

This work has been funded by the European Commission's Sixth Framework Programme (SMERobot grant no. 011838), by the European Research Council (GlobalVision grant no. 209480) and by the Swedish Foundation for Strategic Research through the programme Future Research Leaders.

References

1. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004) Second Edition.
2. Atkinson, K.B.: Close Range Photogrammetry and Machine Vision. Whittles Publishing (1996)
3. Grunert, J.A.: Das pothenot'sche problem in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. Grunert Archiv der Mathematik und Physik **1** (1841) 238–248
4. Olson, C.: A general method for geometric feature matching and model extraction. Int. Journal Computer Vision **45** (2001) 39–54
5. Cass, T.: Polynomial-time geometric matching for object recognition. Int. Journal Computer Vision **21** (1999) 37–61
6. Jacobs, D.: Matching 3-d models to 2-d images. Int. Journal Computer Vision **21** (1999) 123–153
7. Huttenlocher, D., Ullman, S.: Object recognition using alignment. In: Int. Conf. Computer Vision, London, UK (1987) 102–111
8. Jurie, F.: Solution of the simultaneous pose and correspondence problem using gaussian error model. Computer Vision and Image Understanding **73** (1999) 357–373
9. Breuel, T.: Implementation techniques for geometric branch-and-bound matching methods. Computer Vision and Image Understanding **90** (2003) 258–294

10. David, P., DeMenthon, D., Duraiswami, R., Samet, H.: SoftPOSIT: Simultaneous pose and correspondence determination. *Int. Journal Computer Vision* **59** (2004) 259–284
11. Haralick, R.M., Lee, C.N., Ottenberg, K., Nolle, M.: Review and analysis of solutions of the 3-point perspective pose estimation problem. *Int. Journal Computer Vision* **13** (1994) 331–356
12. Quan, L., Lan, Z.: Linear $n \leq 4$ -point camera pose determination. *IEEE Trans. Pattern Analysis and Machine Intelligence* **21** (1999) 774–780
13. Hartley, R., Kahl, F.: Optimal algorithms in multiview geometry. In: *Asian Conf. Computer Vision*, Tokyo, Japan (2007)
14. Olsson, C., Kahl, F., Oskarsson, M.: Optimal estimation of perspective camera pose. In: *Int. Conf. Pattern Recognition. Volume II.*, Hong Kong, China (2006) 5–8
15. Hartley, R., Kahl, F.: Global optimization through searching rotation space and optimal estimation of the essential matrix. In: *Int. Conf. Computer Vision*, Rio de Janeiro, Brazil (2007)
16. Sim, K., Hartley, R.: Removing outliers using the L_∞ -norm. In: *Conf. Computer Vision and Pattern Recognition*, New York City, USA (2006) 485–492
17. Li, H.: A practical algorithm for L_∞ triangulation with outliers. In: *Conf. Computer Vision and Pattern Recognition*, Minneapolis, USA (2007)
18. Olsson, C., Enqvist, O., Kahl, F.: A polynomial-time bound for matching and registration with outliers. In: *CVPR 2008*. (2008)
19. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.* **24** (1981) 381–395
20. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A* **4** (1987)
21. Snavely, N., Seitz, S., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. *ACM SIGGRAPH* **25** (2006) 835–846