New Extensions of the 3-Simplex for Exterior Orientation

John M. Steinbis Tyrone L. Vincent William A. Hoff
Colorado School of Mines
jsteinbis@gmail.com tvincent@mines.edu whoff@mines.edu

Abstract

Object pose may be determined from a set of 2D image points and corresponding 3D model points, given the camera's intrinsic parameters. In this paper, two new exterior orientation algorithms are proposed and then compared against the Efficient PnP Method and Orthogonal Iteration Algorithm. As an alternative to the homogeneous transformation, both algorithms utilize the 3-simplex as a pose parameterization. One algorithm uses a semidefinite program and the other a Gauss-Newton algorithm.

1. Introduction

By applying computer vision techniques, object pose may be determined from a set of 2D image points and corresponding 3D model points, given the camera's intrinsic parameters. In general, the problem has been given the name "exterior orientation", and for an arbitrary number of points it is called "perspective-n-point" or PnP. Difficulty arises due to image point noise, nonlinearity of perspective projection, and the constraints placed on the elements of the 6-DOF pose.

There is a long history of study in this area and many algorithms have been proposed. Linear algorithms can be very fast but are typically not that accurate due to relaxation of pose constraints. Iterative algorithms constrain the pose directly and are usually very accurate, but require a reasonably good initial guess for fast and proper convergence to the global minima. A review and comparison of existing exterior orientation algorithms can be found in the papers [9], [2], and [1].

Recently, Moreno-Noguer et al, [8], proposed an efficient PnP algorithm (EPnP) that relies on the 3-simplex and a linear combination of multiple eigenvectors. Results showed that the eigenvector combination can improve pose accuracy. The scale factors within the eigenvector combination are computed by a linearization method. Additionally, projection error is not considered when calculating the scale factors which can lead to inaccuracies.

In this paper, two new exterior orientation algorithms are proposed and then compared against the EPnP and classic Orthogonal Iteration Algorithm of [7]. Due to space limitations, some details are omitted but a more complete exposition can be found in [10].

2. Linear 3-Simplex

The n-simplex is an n-dimensional analogue of the triangle, and has n+1 vertices; e.g. the 3-simplex is a tetrahedron with four vertices. Barycentric coordinates, α, describe the location of points with respect to the simplex vertices. Suppose we want to describe the location of 3D points on a rigid body, referred to as model points. These points are assumed to be known in the local reference frame attached to the rigid body, called the model frame. The four vertices of the simplex, ${}^{C}P_s^j,\ j = 1,\ldots,4$, are chosen. Then, the ith model point in the model frame, ${}^{C}P_m^i$, may be written as ${}^{M}P_m^i = \sum_{j=1}^{4} \alpha_{ji}\ {}^{M}P_s^j$, with the constraint that the sum of each barycentric coordinate set, $\sum_{j=1}^{4} \alpha_{ji}$, equals $+1$. In this context, the 3-simplex may be considered a generalization of the homogeneous transformation.

The utility of the simplex in an exterior orientation algorithm is that the relationship between vertices, points, and barycentric coordinates is rotation and translation invariant. Thus, if ${}^{C}_{M}H$ is the homogeneous transformation from model to camera coordinates,

$$\begin{bmatrix} {}^{C}P_m^* \\ 1 \end{bmatrix} = {}^{C}_{M}H \begin{bmatrix} {}^{M}P_m^* \\ 1 \end{bmatrix} = {}^{C}_{M}H \begin{bmatrix} {}^{M}P_s^* \\ 1 \end{bmatrix} \alpha = \begin{bmatrix} {}^{C}P_s^* \\ 1 \end{bmatrix} \alpha \tag{1}$$

In other words, the same set of barycentric coordinates still describes the relationship between simplex vertices and points even when expressed in a different reference frame.

Suppose an image of an object is available and we wish to recover the pose of the object with respect to the

camera. Given a postulated pose, the 3D model points expressed in the camera frame can be projected to specific points, u_i and v_i, on the image plane, if the camera's intrinsic parameters, f_u, f_v, u_0, and v_0 are known. The image space error is the difference between the measured image points and the projected model points on the image plane. The image space error equations can be rearranged into a form which is linear in the model points expressed in the camera frame,

$$\begin{bmatrix} e_u^i \\ e_v^i \end{bmatrix} = \begin{bmatrix} f_u & 0 & (u_0 - u_i) \\ 0 & f_v & (v_0 - v_i) \end{bmatrix} {}^C P_m^i \ . \tag{2}$$

Since the *true* projection results in zero error, the null space of this linear relationship consists of all points in the camera frame that could have created the image points. An alternative error metric, object space error, proposed by [7] measures the error in 3D camera space. Similarly, the object space error equations can be rearranged into a linear form,

$$\begin{bmatrix} e_x^i \\ e_y^i \\ e_z^i \end{bmatrix} = \begin{bmatrix} (f_{11}^i - 1) & f_{12}^i & f_{13}^i \\ f_{21}^i & (f_{22}^i - 1) & f_{23}^i \\ f_{31}^i & f_{32}^i & (f_{33}^i - 1) \end{bmatrix} {}^C P_m^i \ , \tag{3}$$

where f_i are elements of a projection matrix. The expression for model points in terms of the simplex vertices and barycentric coordinates are substituted into the projection error equations above. Notice that the equations are linear in the simplex vertices and may therefore be written in matrix form. Let ${}^C\vec{P}_s$ equal the vector of simplex vertices in the camera frame so that $\begin{bmatrix} e_u^i & e_v^i \end{bmatrix}^T = M_i \, {}^C\vec{P}_s$ or $\begin{bmatrix} e_x^i & e_y^i & e_z^i \end{bmatrix}^T = M_i \, {}^C\vec{P}_s$.

The error equations for the entire point set, $1, \ldots, n$, are used to construct a large matrix referred to as the "measurement matrix", $M = \begin{bmatrix} M_1 & \ldots & M_n \end{bmatrix}^T$, which is of size $2\,n \times 12$ or $3\,n \times 12$ in either image space or object space. The null space of the measurement matrix, $\mathbb{V} = \left\{ {}^C\vec{P}_s \in R^{12} \,|\, M \, {}^C\vec{P}_s = 0 \right\}$, is a vector of simplex vertices that result in zero projection error.

Regardless of which error space is used, a basis for the measurement matrix null space may be computed by eigenvalue decomposition of $M^T M$, where v_i are the eigenvectors and λ_i are the positive real eigenvalues. If noise is present, there will be no strictly zero eigenvalue and the projection error of each basis vector equals

$$v_i^T M^T M \, v_i = \|M v_i\|^2 = \lambda_i \|v_i\| \ . \tag{4}$$

In this section, we are only interested in the eigenvector, v_1, with the smallest eigenvalue, λ_1, which is reshaped back into 3×4 matrix form, $v_1 \rightarrow {}^C\tilde{P}_s^*$. A nonzero scaling exists because the null space is a vector space, and vector spaces are closed under scalar multiplication and vector addition. In this context, the *final* scale factor, β_F, must be determined in order to estimate model points in the camera frame that are approximately the same size as the points in the model frame.

Fortunately, the scale factor may be easily computed, in a least-squares sense, from the sum of squared error about the model point centroids. The scale factor is the square root of the distance ratio. After the scale factor is recovered and model points in the camera frame are estimated by ${}^C P_m^* = \beta_F \, {}^C\tilde{P}_s^* \, \alpha$, the "absolute orientation" concept of [4] or [5] is applied to find the relative transformation between the point sets, ${}^C_M H$. This approach forces the final rotation to be special orthogonal (SO3).

By using only one eigenvector, pose may be determined non-iteratively by relaxing size and shape constraints of the simplex within eigenvalue decomposition. The linear 3-simplex algorithm can be very fast and also very accurate if image noise is low and many points are available. The scale factor applied to one eigenvector only corrects size but not shape. Correction of shape from multiple eigenvectors is the focus of the next section.

3. 3-Simplex-SDP

The algorithm presented in this section is a technique that produces a more accurate result from the same framework, which uses multiple eigenvectors and a polynomial semidefinite program (SDP). However, it is iterative and inevitably slower than the linear method.

Recall from the last section that only the eigenvector of $M^T M$ with the smallest eigenvalue was of interest, and in a noisy system there will be no perfectly zero eigenvalue. However, with noise, all of the eigenvectors provide *some* information in decreasing order of significance as the size of the eigenvalue increases. If the first four eigenvectors, $v_a, \ldots, v_d$, are used in a linear combination,

$$(\beta_a \, v_a + \beta_b \, v_b + \beta_c \, v_c + \beta_d \, v_d) \rightarrow {}^C\tilde{P}_s^* \ , \tag{5}$$

four scale factors, $\beta_a, \ldots, \beta_d$, are required.

From the eigenvector combination, model points in camera frame are estimated by ${}^C P_m^* = \beta_F \, {}^C\tilde{P}_s^* \, \alpha$, and then the absolute orientation concept is applied to find the relative transformation. Projection error of the eigenvector combination is given by

$$\begin{aligned} E &= \|M(\beta_a \, v_a + \beta_b \, v_b + \beta_c \, v_c + \beta_d \, v_d)\|^2 \\ &= \beta_a^2 \lambda_a \|v_a\| + \beta_b^2 \lambda_b \|v_b\| + \beta_c^2 \lambda_c \|v_c\| + \beta_d^2 \lambda_d \|v_d\|. \end{aligned} \tag{6}$$

As suggested by [8], the betas are calculated from non-linear rotation and translation invariant shape constraints that could not be imposed during eigenvalue decomposition. Euclidean distance between vertex pairs is a logical choice, and a set of four vectices has six possible combinations. In a noiseless system, distances in the model and camera frames should be equal,

$$\left\| {}^{C}P_s^i(\beta) - {}^{C}P_s^j(\beta) \right\|^2 = \left\| {}^{M}P_s^i - {}^{M}P_s^j \right\|^2 \ . \quad (7)$$

The dot product of a vertex triplet is another rotation and translation invariant quadratic constraint, and a set of four vectices has twelve possible combinations. In a noiseless system, the direction cosine that the dot product represents should be equal in the camera and model frames,

$$\begin{aligned} &\left({}^{C}P_s^i(\beta) - {}^{C}P_s^k(\beta) \right) \cdot \left({}^{C}P_s^j(\beta) - {}^{C}P_s^k(\beta) \right) \\ &= \left({}^{M}P_s^i - {}^{M}P_s^k \right) \cdot \left({}^{M}P_s^j - {}^{M}P_s^k \right) \ . \end{aligned} \quad (8)$$

However, with noise present, a technique is needed to minimize shape error by comparing distances and dot products in both frames.

In [8] a linearization method was used to calculate the βs which leads to a suboptimal solution. In addition, there was no consideration for project error resulting from the vector combination. In what follows, we present a globally convergent algorithm that balances shape and projection error without approximation. In particular, we see to find β to minimize the following error function

$$\begin{aligned} E(\beta) = &\sum_{l=1}^{18} \left({}^{C}d_l(\beta) - {}^{M}d_l \right)^2 \\ &+ w \left[\left(\frac{\lambda_b}{\lambda_a} \right) \beta_b^2 + \left(\frac{\lambda_c}{\lambda_a} \right) \beta_c^2 + \left(\frac{\lambda_d}{\lambda_a} \right) \beta_d^2 \right] \ , \end{aligned} \quad (9)$$

where the ${}^{C}d_l$ and ${}^{M}d_l$ define the polynomial shape constraints and w is a weight that helps restrict the amount of allowable projection error that results from improving shape. As w increases, the strength of the projection error penalty grows. If excessively large, β_a will always be the only nonzero scale factor found by the SDP, defeating the purpose of the minimization. If excessively small, the shape error is minimized with essentially no projection error penalty.

Note that $E(\beta)$ is a quartic multivariate polynomial and additionally a sum of squares. The work of Lasserre, [6], shows how minimizing a polynomial that is a sum-of-squares can be transformed into a convex linear matrix inequality (LMI) problem and then efficiently solved in a semidefinite program (SDP). Hence, the name "3-Simplex-SDP" that was given to the algorithm.

SDP is a broad generalization of linear programming (LP), to the case of symmetric matrices. A linear objective function is optimized over linear equality constraints with a semidefinite matrix condition. The LMI minimization problem is written in *standard* form as

$$\min_X \ E = P^T X \text{ , such that } M \geq 0 \ , \quad (10)$$

where E is the objective function and M is the semidefinite matrix of the linear inequality constraint. The key feature of SDPs is their convexity, since the feasible set defined by the constraints is convex. In this context, the SDP is used to solve convex LMI relaxations of multivariate real-valued polynomials.

In this algorithm, the vector P contains the coefficients of the polynomial $E(\beta)$, X is the optimization variable that encodes β, and M is the block diagonal semidefinite matrix constructed from three smaller matrices, M_0, M_1, M_2, which are semidefinite themselves. The constraints of M_0 force the magnitudes of the scale factors into the correct order, $\beta_a^2 \geq \beta_b^2 \geq \beta_c^2 \geq \beta_d^2$, which is also indirectly enforced through the projection error penalty. M_1 and M_2 are both "moment" (or covariance) matrices that are constructed from multiplication of monomial vectors from the sum-of-squares decomposition. Unfortunately, the details are too lengthy to include here but can be found in [10] and [6].

After the SDP terminates, the βs are recovered from the optimal vector X. Because the sum-of-squares decomposition is not unique, a (trivial) sign ambiguity exists that places the model points in front or behind the camera.

In Matlab, the SeDuMi package, [11], is used to solve the sum-of-squares problem as a LMI. YALMIP, SOSTOOLS, and GloptiPoly are interfaces that can convert the problem into a form that is compatible with SeDuMi.

4. 3-Simplex-GNA

In this section, global convergence is compromised for fast runtime in a new algorithm that also utilizes the 3-simplex parameterization and existing framework. The algorithm requires an initial guess that could come from numerous sources, including the linear 3-Simplex algorithm, the 3-Simplex-SDP, or even a recursive estimator in a tracking situation such as augmented reality.

The measurement matrix can actually be decomposed into a 12×12 upper triangular matrix. From SVD

we obtain

$$M^T M = (VSU^T)(USV^T) = (VS)(SV)^T = \hat{M}^T \hat{M} \ . \tag{11}$$

This is a very significant reduction if the number of points is large. QR decomposition is perhaps a quicker way to compute the reduced measurement matrix, $\hat{M}$. It can be shown, [3], that minimizing a residual with the reduced measurement matrix is the same as minimizing a residual with the full size matrix,

$$\min_{{}^C\vec{P}_s} \|\hat{M}\, {}^C\vec{P}_s\|^2 = \min_{{}^C\vec{P}_s} \|M\, {}^C\vec{P}_s\|^2 \ . \tag{12}$$

Sub-optimality of the linear 3-simplex algorithm might be attributed in part to the relaxation of size and shape constraints on the simplex vertices. Specifically, the simplex vertices are given 12-DOF even though they really only have six. For example, distances between all vertex pairs should be the same in the model and camera frames. Six constraints are placed on twelve parameters leaving only 6-DOF.

Fortunately, it is easy to enforce these nonlinear constraints in an iterative algorithm by parameterizing the simplex. Instead of choosing an arbitrary simplex, it is logical to choose a specific shape, such as a unit length right-handed orthogonal triad. e.g. ${}^M P_s^* = \begin{bmatrix} I_3 & 0 \end{bmatrix}$.

One method of constructing such a triad is by decomposing the model point set with SVD. The triad origin is placed at the centroid, ${}^M t_s = {}^M\bar{P}_m$ and the legs are made to point in the principle directions, ${}^M\hat{X}_s = v_1$, ${}^M\hat{Y}_s = v_2$, ${}^M\hat{Z}_s = \pm v_3$, such that $\det(R) = +1$. If chosen this way, another reference frame, $\{\mathrm{S}\}$, has been effectively created. More analysis is needed to determine if this is the optimal selection of the simplex.

If a unit length orthogonal triad is chosen, the final pose solution may be quickly computed by ${}^C_M H = {}^C_S H\ {}^S_M H$, where ${}^M_S H$ is equivalent to the chosen simplex and ${}^C_S H$ is equivalent to the unknown simplex.

Four points are required to define an orthogonal triad which can move freely in space. One point defines the position and the other three define orthogonal direction vectors along the legs. The rotation and translation of the homogeneous transform above can be used to construct an orthogonal triad,

$${}^M P_s^* = \begin{bmatrix} {}^M\hat{X}_s + {}^M t_s & {}^M\hat{Y}_s + {}^M t_s & {}^M\hat{Z}_s + {}^M t_s & {}^M t_s \end{bmatrix} . \tag{13}$$

Euler angles can be used to parameterize the rotational components of the unknown traid, $R(\alpha, \beta, \gamma) = \begin{bmatrix} {}^C\hat{X}_s & {}^C\hat{Y}_s & {}^C\hat{Z}_s \end{bmatrix}$, and thus, a six element vector will parameterize the entire triad, $\Theta = \begin{bmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{bmatrix}^T$.

A standard GNA may be used to minimize projection error of $f(\Theta) = \|\hat{M}\, {}^C\vec{P}_s(\Theta)\|^2$ and will have good convergence properties if the initial guess is in the neighborhood of the global minima.

By using the reduced measurement matrix, the Jacobian computed each iteration will always be of size 12×6, no matter how many points are in the full set. Thus, a significant time reduction occurs for many points over a few iterations.

5. Results

Simulations are used to compare exterior orientation algorithm performance. Metrics include position and orientation error, number of outliers, and runtime. An 800 pixel focal length is used to project ten 3D points onto the image plane, 1024×768, and then Gaussian noise with ten pixel standard deviation is added to the image points.

In each trial, a new point cloud is randomly generated to lie approximately 5' in front of the camera with 1' spread. The results of 1000 iterations are displayed in the form of a box plot. Instead of plotting the location of outliers, the total number is placed above the top whisker (non-standard).

In the 3-Simplex-SDP section, only the four eigenvector combination is discussed, but of course, two and three vector combinations are possible too. Figure 1 shows the pose error versus number of eigenvectors. These results show that pose error decreases with the number of eigenvectors used and, perhaps more importantly, the four eigenvector combination is consistently better. It is desirable to use additional eigenvectors within the SDP, but there is a performance/complexity trade-off, and it is not recommended to use more than four eigenvectors.

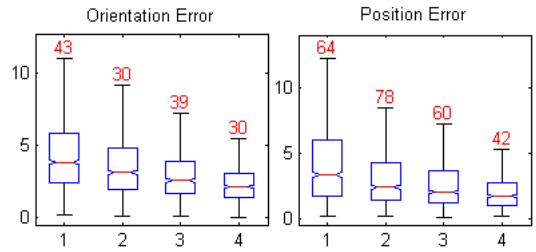

Figure 1. Pose Error: Eigenvector Combinations

Figure 2 compares the 3-Simplex-SDP (four eigenvector) to a modified EPnP method. The modification to the EPnP of [?] is the use of a GNA to further refine the shape fit, although it still does not consider projection error. The figure shows that the pose error of the 3-Simplex-SDP (four eigenvector) is significantly less

than the EPnP method. The figure also shows, by the relative number of outliers, that the convergence of the 3-Simplex-SDP is much better than the EPnP method. However, the performance increase comes at the expense of additional algorithm complexity when converted to a sum-of-squares, but in many situations accuracy is more important than runtime.

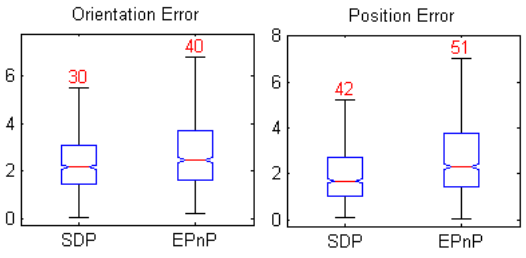

Figure 2. Group A: Pose Error

Figure 3 shows that the pose error and convergence of the 3-Simplex-GNA and Orthogonal Iteration Algorithm (OIA) are equivalent. However, Figure 4 shows that the runtime of the 3-Simplex-GNA is much better than the OIA. The data reduction to $\hat{M}$ and asymptotic efficiency of Gauss-Newton result in a very fast algorithm - possibly a lower bound on runtime.

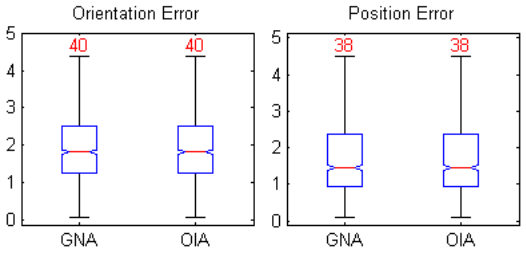

Figure 3. Group B: Pose Error

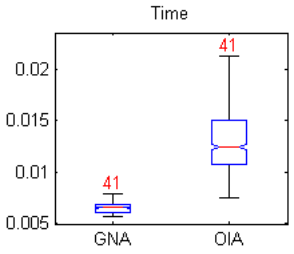

Figure 4. Group B: Runtime

6. Conclusions

Two new exterior orientation algorithms were introduced and compared against existing algorithms. The 3-Simplex-SDP can be useful in situations when a good initial guess is unavailable. It is globally convergent and more accurate than the EPnP method. However, the algorithm is not particularly fast, although a special implementation of the SDP solver may help improve runtime. The 3-Simplex-GNA is very fast, but it requires a reasonably good initial guess and is not globally convergent.

7. Acknowledgments

This work was supported by the National Science Foundation under Grant ECS 01-34132.

References

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *IEEE Trans. of Pattern Analysis and Machine Intelligence*, 2003.

[2] P. Fiore. Efficient linear solution of exterior orientation. *IEEE Trans. of Pattern Analysis and Machine Intelligence*, 2001.

[3] R. Hartley. Minimizing algebraic error in geometric estimation problems. *IEEE Sixth International Conference on Computer Vision*, 1998.

[4] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 1986.

[5] B. Horn, H. Hilden, and S. Negahdariour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, 1988.

[6] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal of Optimization*, 2001.

[7] C. Lu, G. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Trans. of Pattern Analysis and Machine Intelligence*, 2000.

[8] F. Moreno-Noguer, P. Fua, and V. Lepetit. Accurate non-iterative o(n) solution to the pnp problem. *IEEE 11th International Conference on Computer Vision*, 2007.

[9] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Trans. of Pattern Analysis and Machine Intelligence*, 1999.

[10] J. Steinbis. A new vision & inertial pose estimation system for handheld augmented reality: Design, implementation, & testing. *CSM Masters Thesis*, 2008.

[11] J. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 1999.