A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations[1]

A. Lorusso
Robotics and Automation Department
Tecnopolis CSATA Novus Ortus
70010 Valenzano - Bari - Italy
adele@rob.csata.it

D. W. Eggert and R. B. Fisher
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland EH1 2QL
eggertd or rbf@aifh.ed.ac.uk

Abstract

A common need in machine vision is to compute the 3-D rigid transformation that exists between two sets of points for which corresponding pairs have been determined. In this paper a comparative analysis of four popular and efficient algorithms is given. Each computes the translational and rotational components of the transform in closed-form as the solution to a least squares formulation of the problem. They differ in terms of the representation of the transform and the method of solution, using respectively: singular value decomposition of a matrix, orthonormal matrices, unit quaternions and dual quaternions. This comparison presents results of several experiments designed to determine the (1) accuracy in the presence of noise, (2) stability with respect to degenerate data sets, and (3) relative computation time of each approach.

1 Introduction

Determining the relationship between two coordinate systems using sets of corresponded features is known as the *absolute orientation problem.* It has numerous applications in the areas of *photogrammetry*, robotics (*constructing world models*), *object motion analysis*, computing the *hand-eye transform* for cameras, as well as object *pose estimation* following recognition.

A recent survey by Sabata and Aggarwal [1] lists a large number of algorithms developed to compute the 3-D rigid transformation between two sets of corresponded features (e.g., surfaces, lines and points). However, it does not provide a quantitative comparison of these techniques. In this paper a commonly used subset of these approaches is analyzed; *closed-form* solutions using *corresponded points.*

The four popular closed-form solutions compared here differ with respect to the transformation representation and alternative ways of minimizing a criterion function. The first was developed by Arun, Huang and Blostein [2], and is based on computing the *singular value decomposition* (SVD) of a derived matrix. A similar approach based on *orthonormal matrices*, but computing the *eigensystem* of a derived matrix was presented by Horn, Hilden and Negahdaripour [3]. The third algorithm, also due to Horn [4], involves representing the rotational component

[1]This work was funded by EC H.C.M. Project ERB40500PL921003 through the SMART network, and UK EPSRC Grant GR/H/86905.

using a *unit quaternion*. The use of *dual quaternions* to represent both transform components is the basis of the fourth technique of Walker, Shao and Volz [5].

The comparison presented here consists of three parts. First, the *accuracy* of each algorithm is examined as the coordinates of corresponding points are corrupted with increasing amounts of noise. Second, the *stability* is determined as original 3-D point sets degenerate into such forms as a plane, line and single point. Lastly, the relative *efficiency*, in terms of actual execution time, is reported for the above situations. Conclusions based on these results should make the choice of an appropriate algorithm for a given application simpler and more reliable.

2 Description of Algorithms

Each of the four algorithms computes the solution to a similar problem which can be described as follows. Assume there exist two corresponded point sets $\{m_i\}$ and $\{d_i\}$, $i = 1 \ldots N$, such that they are related by $d_i = R\, m_i + T + V_i$, where R is a standard 3×3 rotation matrix, T is a 3-D translation vector and V_i a noise vector. Solving for the optimal transformation $[\hat{R}\ ,\ \hat{T}]$ that maps the set $\{m_i\}$ onto $\{d_i\}$ typically requires minimizing the *least squares error criterion*:

$$\Sigma^2 \ = \ \sum_{i=1}^{N} \| \ d_i - \hat{R}\ m_i - \hat{T} \ \|^2 \tag{1}$$

For the exact details of the four algorithms we must refer the reader to the original papers. However, in the following subsections the characteristics of each that are relevant to the comparison are described; namely, transformation representation, minimization method and special cases. All of the solutions are in closed form, which are typically more efficient than iterative techniques (e.g., the extended Kalman filter approach of Zhang [6]).

2.1 A solution involving the SVD of a matrix

This first method was developed by Arun, Huang and Blostein [2]. The transformation rotation is represented using a standard 3×3 orthonormal matrix, while translation is a 3-D vector, as in the equations above.

The solution method can be described as follows. By noticing that the point sets should have the same centroid at the correct solution, the rotation component is found first by analyzing point sets after their translation to the origin. A 3×3 correlation matrix given by

$$H \ = \ \sum_{i=1}^{N} m_{c,i}\ d_{c,i}^{T} \tag{2}$$

is computed based on these new centered point sets. Its singular value decomposition, $H = U \Lambda V^{\,T}$, is determined. The optimal rotation matrix is then $\hat{R} = V U^{\,T}$. (Note, this computation of $\hat{R}$ is also known as the *orthogonal Procrustes problem*, the SVD-based solution of which has been known for some time [7].) The optimal translation is found as that which aligns the centroid of the set $\{d_i\}$ with the centroid of the optimally rotated set $\{m_i\}$, that is $\hat{T} = \bar{d} - \hat{R}\, \bar{m}$.

A special case occurs for planar data sets or in the presence of large amounts of noise. If the determinant of $\hat{R}$ is -1, a reflection rather than a rotation has been found. This is corrected by setting $\hat{R} = V' U^T$, where $V' = [\ v_1, v_2, -v_3\ ]$ and v_3 is the third column of V corresponding to the zero singular value of H. This special case has also been handled in alternative derivations by Umeyama [8] and Kanatani [9]. The algorithm is not designed for linear or singular point data sets.

2.2 A solution involving orthonormal matrices

The second algorithm is similar in nature to the first, but was developed independently by Horn, Hilden and Negahdaripour [3]. The same representation for R and T is used. Again the point sets are translated to the origin, and the correlation matrix H in equation (2) calculated. However, rather than compute the SVD of this matrix, a *polar decomposition* [9] is used, such that $H = RS$, where $S = (HH^T)^{1/2}$. Here, $\hat{R} = H^T(\frac{1}{\sqrt{\lambda_1}}\ u_1\ u_1^T + \frac{1}{\sqrt{\lambda_2}}\ u_2\ u_2^T + \frac{1}{\sqrt{\lambda_3}}\ u_3\ u_3^T)$, where $\{\lambda_i\}$ and $\{u_i\}$ are the eigenvalues and corresponding eigenvectors of the matrix HH^T. The optimal translation $\hat{T}$ is computed as in the first algorithm, after calculating $\hat{R}$.

Planar point sets again give rise to a special case, since the equation for $\hat{R}$ does not hold as λ_3 gets small. Unfortunately, the proposed correction in [3] is not valid in all cases. An alternate solution is: $\hat{R} = H^T S^+ \pm \frac{X}{\sqrt{|\ Trace(X)\ |}}$, where the sign is chosen such that the determinant of $\hat{R}$ is positive, $S^+ = \frac{1}{\sqrt{\lambda_1}}\ u_1\ u_1^T + \frac{1}{\sqrt{\lambda_2}}\ u_2\ u_2^T$, $X = [\ (H^T\ S^+)\ (H^T\ S^+)^T - I\ ]\ u_3\ u_3^T$ and u_3 is the eigenvector associated with the smallest eigenvalue of S. Neither derivation can be applied to linear or singular point data sets.

2.3 A solution involving unit quaternions

The third method is also due to Horn [4], but in this case a different representation of the transformation is used. Rather than have the standard 3×3 orthonormal matrix represent rotation, a unit quaternion is employed. When a rotation is considered as a movement through an angle θ about an axis $a = [\ a_x, a_y, a_z\ ]$ going through the origin, the equivalent unit quaternion is defined as a 4-D vector $q = [\ cos(\theta/2),\ sin(\theta/2)\ a_x,\ sin(\theta/2)\ a_y,\ sin(\theta/2)\ a_z\ ]$. Translation is still represented using a 3-D vector.

By rewriting the minimization problem in the quaternion framework (see [4]) a new 4×4 matrix can be constructed from the correlation matrix H in (2) as:

$$\mathrm{P} = \begin{bmatrix} H_{00} + H_{11} + H_{22} & H_{12} - H_{21} & H_{20} - H_{02} & H_{01} - H_{10} \\ H_{12} - H_{21} & H_{00} - H_{11} - H_{22} & H_{01} + H_{10} & H_{20} + H_{02} \\ H_{20} - H_{02} & H_{01} + H_{10} & H_{11} - H_{00} - H_{22} & H_{12} + H_{21} \\ H_{01} - H_{10} & H_{20} + H_{02} & H_{12} + H_{21} & H_{22} - H_{11} - H_{00} \end{bmatrix}$$

The optimal rotation, represented in quaternion form, is the eigenvector corresponding to the largest positive eigenvalue of P. This rotation quaternion can then be converted into standard matrix form. Following this, the optimal translation can be computed as in the other two methods. The above technique does not need to be modified to handle planar point sets, but as with the others, will not work for linear or singular point data sets.

2.4 A solution involving dual quaternions

The fourth algorithm is due to Walker, Shao and Volz [5], and is the most significantly different of the group, originally designed to minimize the equation:

$$\Sigma^2 = \sum_{i=1}^{L} \alpha_i \parallel n_{1i} - \hat{R}\, n_{2i} \parallel^2 + \sum_{i=1}^{N} \beta_i \parallel d_i - \hat{R}\, m_i - \hat{T} \parallel^2 \tag{3}$$

where $\{n_{1i}\}$ and $\{n_{2i}\}$ are two sets of L corresponding unit normal vectors, and $\{\alpha_i\}$, $\{\beta_i\}$ are weighting factors reflecting data reliability. This equation can be simplified to that of equation (1) by setting $\alpha_i = 0$ and $\beta_i = 1$.

In this method the rotation and translation are represented together using a dual quaternion, $q_d = [r, s]$. Here, motion is modelled as a simultaneous rotation around and translation along a particular line, with direction $n = [\ n_x, n_y, n_z\ ]$, passing through a point $p = [\ p_x, p_y, p_z\ ]$. The amount of motion is given by an angle θ and a distance t, defining the components of q_d as:

$$r = \begin{bmatrix} \sin(\theta/2)\ n \\ \cos(\theta/2) \end{bmatrix} \qquad\qquad s = \begin{bmatrix} \frac{t}{2}\ \cos(\theta/2)\ n + \sin(\theta/2)\ (p \times n) \\ -\frac{t}{2}\ \sin(\theta/2) \end{bmatrix}$$

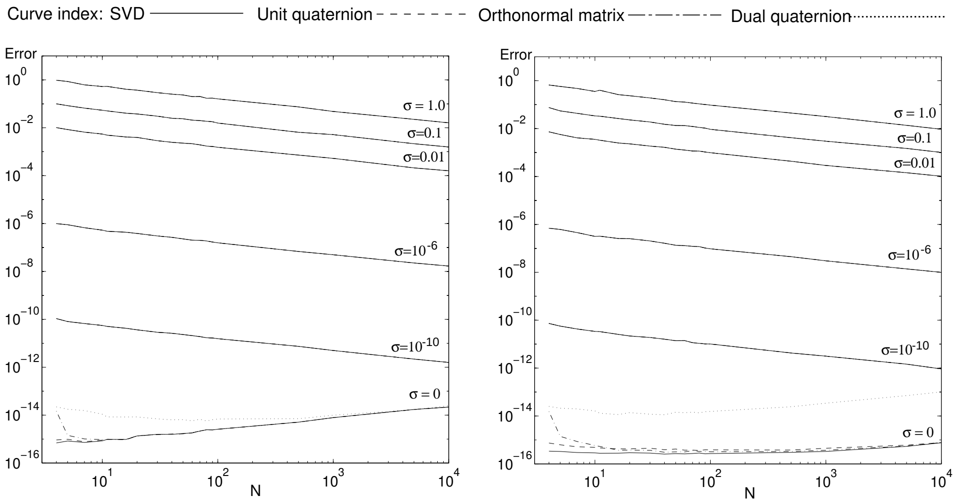
Again the minimization equation can be rewritten in this new framework (see [5]), resulting in an equation involving r and s. The optimal component $\hat{r}$ for this new equation is found as the eigenvector corresponding to the largest positive eigenvalue of a 4×4 matrix, $A = \frac{1}{4N}\ C_1^T\ C_1 - C_2$, where C_1 and C_2 are matrices whose elements are functions of the point coordinates. From $\hat{r}$ the standard rotation matrix can be computed as $\hat{R} = (\hat{r}_3^2 - \hat{r}_{0..2}^T\ \hat{r}_{0..2})\ I + 2\ \hat{r}_{0..2}\ \hat{r}_{0..2}^T + 2\ \hat{r}_3\ K(\hat{r})$, where $\hat{r}_{0..2}$ is the first three elements of $\hat{r}$ and K is a matrix that is a function of the components of $\hat{r}$. The optimal value of $\hat{s}$ can also be computed from $\hat{r}$ as $\hat{s} = -\frac{1}{2N}\ C_1\ \hat{r}$, from which the optimal translation follows as $\hat{T} = W(\hat{r})^T\ \hat{s}$, where W is another matrix of quaternion components. As in the unit quaternion approach planar sets are processed fine, but linear and singular point sets are not handled.

3 Experimental Comparison

Each of the four algorithms was implemented in C++ with efficiency in mind (compiled using gcc, version 2.6.3, using -O3 optimization). To provide a source of commonality, routines from the Eispack linear algebra package [10] (converted from Fortran to C) were used for all SVD (routine *svd*) and eigensystem (routines *tred2* and *tql2*) calculations. These are known for their stability and rapid convergence, and behaved properly during the series of experiments described below. More detailed versions of these experiments can be found in [11].

3.1 The accuracy experiment

In this experiment, the absolute accuracy of each of the algorithms was tested under varying levels of noise. Non-degenerately arranged 3-D point sets of size $N = 4$ to $N = 10,000$ were used. The points $\{m_i\}$ were chosen randomly from a



(a) Translation error: All algorithms vs. True (noisy)

(b) Rotation error: All algorithms vs. True (noisy)

Figure 1: Errors between computed translation and rotation from algorithms and known transformation. Graphs (log-log scale) are translation error, $||\hat{T}_{alg} - \hat{T}_{true}||$, and rotation error, $||\hat{q}_{alg} - \hat{q}_{true}||$, vs. data set size, N.

uniform distribution within a cube of size $2 \times 2 \times 2$ centered about the origin. The corresponding set $\{d_i\}$ was formed by first adding uncorrelated, isotropic Gaussian noise with zero mean and variable standard deviation to each point, and then transforming to a new location. The translation components of the transform were randomly selected from a uniform distribution in the range $[-10 .. 10]$. The rotation matrix was calculated via a random selection from a uniform distribution of unit quaternions representing valid rotations.

For each data set size and noise level one hundred trials were run and the average response recorded. A trial consisted of new point, transform and noise values. Two error statistics were computed for the calculated transforms. These are the norm of the difference between the true and estimated translation vectors, as well as the norm of the difference between true and estimated unit quaternions representing the rotation.

Figure 1 shows the accuracy of the four algorithms for changes in noise level with respect to the error in computed translation and rotation. As expected, as the number of points grows, the error in the computed transformation approaches a value dependent on the noise level. The exception is the zero noise case. Here the dual quaternion (DQ) algorithm appears less accurate than the others, which are all very similar except on small data sets. However, these differences are really minimal, the largest difference being 10^{-13}.

From these results, one can conclude that the *difference* in accuracy between the algorithms is many orders of magnitude below normal noise levels, almost at the level of machine precision ($2 * 10^{-16}$). The unit quaternion (UQ) and SVD results are most similar, deviating from that of the orthonormal matrix (OM) only for small data sets, where these three perform better than the DQ method.

3.2 The stability experiments

In this section the stability of the algorithms is tested, in terms of how well they compute a correct transformation for degenerate data forms (plane, line and point). Here a sequence of data sets is needed to monitor the breakdown of the algorithms. Successive sets in the sequence are taken from a volume whose dimensions are steadily reduced from that of an original $2 \times 2 \times 2$ cube to approach those of a plane, a line, or a single point.

Given this data sequence, two noise perturbations are applied to the points. The first is isotropic, as in the accuracy experiment. This models the case where data is gathered and no prior uncertainty model is known. The second is anisotropic noise, which models higher data accuracy in some dimensions, but less in others. The more accurate dimension here is the direction of degeneracy (for instance, perpendicular to a plane). Noise added in this direction is reduced proportional to the changing volume's dimensions.

The error criterion for these experiments is the root mean square (RMS) error of a control point set, which is necessary for measuring any error in the unconstrained degrees of freedom of the degeneracy. The points of this set (again within a $2 \times 2 \times 2$ cube) undergo both the true and computed motions, and then the distances between the resulting sets are measured. The error plots in Figure 2 are for the SVD algorithm. Except where noted below these responses are typical of the others.

3-D to 2-D degeneracy In this experiment the degeneracy ratio (ratio of original cube Z dimension to new reduced Z dimension) is increased until the data set closely resembles the $X - Y$ plane. RMS error responses were computed for several different data set sizes, noise levels and types. Typical values are basically independent of the degeneracy level, as shown in Figures 2.a and 2.b. Also, RMS values for data sets of increasing size converge to $\sqrt{3}\,\sigma$ for isotropic noise and $\sqrt{2}\,\sigma$ in the anisotropic case.

The individual responses of the algorithms are virtually identical, except in the zero noise case depicted in Figure 3.a. Here, the response from OM is less numerically stable than the others (perhaps suggesting the correction provided here is not the best alternative), and that of DQ is slightly less accurate than either the SVD or UQ results.

3-D to 1-D degeneracy In this experiment the data set degenerates into a line, namely the Z axis. The SVD error responses are shown in Figures 2.c and 2.d. Since the algorithms are not designed for linear data sets, the graphs indicate the degeneracy level necessary for breakdown. Under isotropic noise the error steadily rises as the diameter of the line decreases, completely breaking down in the best cases when the line diameter reaches the level of the noise, and in most other cases earlier than that. Under anisotropic noise the error remains at the original noise level until numeric instability occurs, thereafter it breaks down steadily.

The largest difference in breakdown position is observed in the zero noise case, as graphed in Figure 3.b. Results for the UQ algorithm are very similar to those of SVD, breaking down slightly earlier. The DQ algorithm breaks down first in all cases, while the OM algorithm's performance is highly dependent on the data set

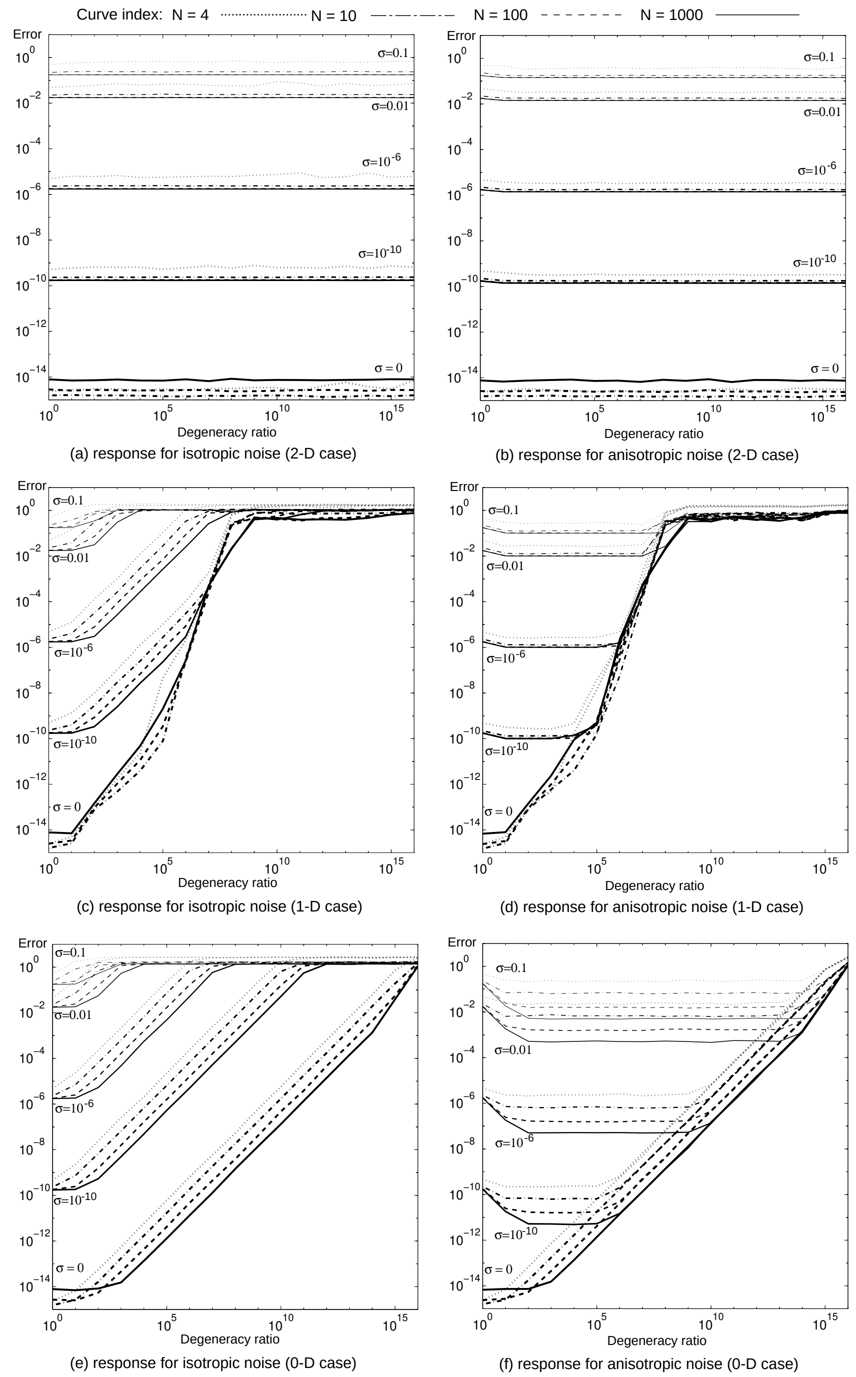


Figure 2: Graphs (log-log scale) of RMS errors of SVD algorithm for (a-b) 2-D, (c-d) 1-D, and (e-f) 0-D degenerate data sets of changing size under isotropic and anisotropic noise of varying levels.

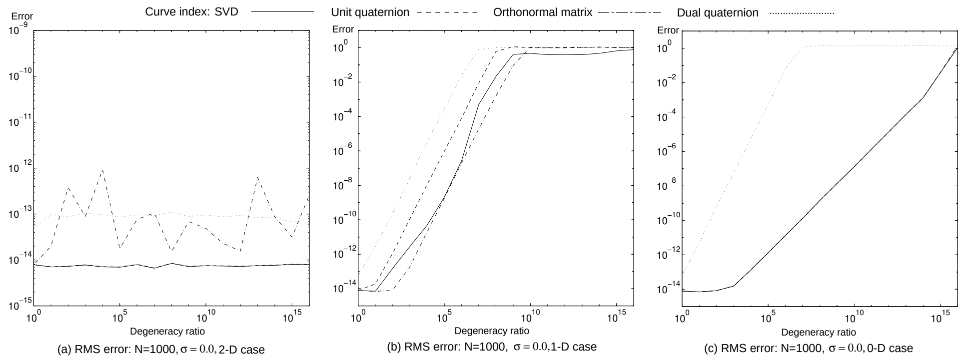


Figure 3: Graphs (log-log scale) of RMS errors of algorithms for $N = 1000$ and $\sigma = 0$. Cases are for degeneracy to (a) plane, (b) line, and (c) singular point.

size. It actually performs the best for large data sets as shown in Figure 3.b, but does worse than the SVD and UQ responses for small data sets.

3-D to 0-D degeneracy In this experiment the data set degenerates into a point (the origin). Figures 2.e and 2.f show typical response curves which are very similar for all but the DQ algorithm. Breakdown trends similar to those in the linear degeneracy case can be seen, with the point of breakdown generally occurring later. Once again, DQ always falls apart sooner (see Figure 3.c), even earlier than in the previous experiment.

3.3 Efficiency experiments

From a theoretical standpoint, each of the algorithms examined has a time complexity of $O(N)$. In this section the actual coefficients of the linear time functions are compared by measuring the execution times of the previous four experiments. All timings were performed on a 50 MHz Sun Sparc Station 10 with a 1MB cache memory. The graphs in Figure 4 represent the zero noise case for nondegenerate 3-D data sets. The changes in timings due to different levels of noise were minimal, but there was some increase in speed noticed for degenerate data sets.

First consider the coefficient of the linear term of the time equation. All but the DQ algorithm initially spend time computing the same set of sums for the correlation matrix. Therefore the linear terms, which are dependent on the number of data points, should be identical for these three. The different set of sums computed by the DQ algorithm is larger in number but requires fewer references to the data values. In looking at the curves in Figure 4 the relative slopes are different for small and large data sets. The change occurs when the data's memory requirements exceed the machine's cache size. Prior to the transition point, the greater number of computations of the DQ algorithm are evident, but as cache misses become more frequent the roles are reversed.

Next consider the constant terms of the time equations, which are unique for each algorithm. SVD computes the singular value decomposition of a 3×3 matrix, followed by matrix multiplication and determinant operations. The OM algorithm

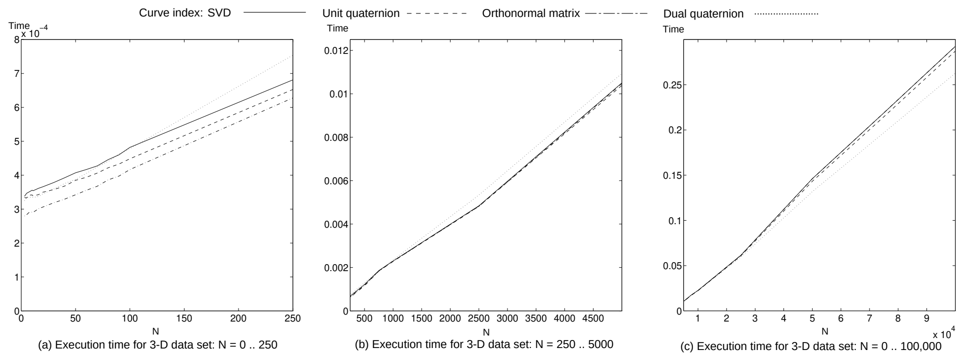


Figure 4: Execution times (in seconds) of algorithms on nondegenerate data sets.

determines the eigensystem of a 3×3 matrix, in addition to a greater number of matrix operations. The UQ algorithm computes the eigensystem of a 4×4 matrix, followed by a minimal amount of computation for representation conversion. Lastly, the DQ algorithm also computes the eigensystem of a 4×4, and does more matrix calculations. Similar work is done by all to compute the translation.

The relative times of these calculations are best seen in Figure 4.a. Here, the SVD computations seem least efficient, while the smaller eigensystem calculation of OM is quickest. But this is not true for all data set sizes, as shown in Figures 4.b and 4.c. Because the SVD and eigensystem computations are actually iterative, their relative timings can change somewhat based on data complexity, which causes the observed curve crossovers. With degenerate data sets (see [11]) the calculation times are reduced in general and similar crossovers still occur.

So, when $N < 100$, the OM algorithm is quickest, by as much as 15%. For large data sets the relative difference of the SVD, UQ and OM methods becomes negligible, about 1%. Prior to data size exceeding the cache ($N < 10,000$) the DQ technique is slower, but then it becomes more efficient as N increases.

4 Conclusions

A comparison has been made between four algorithms that compute a rigid 3-D transformation between two sets of corresponding points in closed form: SVD, UQ, OM and DQ. As with most comparisons, no one algorithm was found to be superior in all cases, but some general conclusions can be made.

The difference in accuracy on nondegenerate 3-D point sets, even for various noise levels, is almost insignificant. This is perhaps not unexpected since they were designed to solve the same problem. But, this conclusion is not in agreement with earlier findings of Walker, *et al.* which stated "the two algorithms produce the same rotation errors ... for the translation errors, the DQ algorithm exhibits better performance than the SVD algorithm ... " [5, pg. 364]. This conclusion is certainly not supported by the data here.

There does appear to be a greater separation in algorithm stability. Here, in most cases, the SVD and UQ methods were very similar and usually the most stable. The OM method was not as stable for planar data sets, but was superior for certain large degenerate data sets. The DQ algorithm was never the most stable, and usually broke down before the others.

In terms of efficiency, for small data set sizes the OM algorithm was quicker (this conclusion depends heavily on the numerical package used, see [11]). On larger data sets the memory configuration of the computer must be considered to determine if the relative speed of the DQ algorithm is superior.

So, in conclusion, it appears that the SVD algorithm provides the best overall accuracy and stability, but may not be as efficient as DQ for large data sets. Otherwise, on smaller data sets, the UQ technique can be chosen for slightly better speed performance than SVD with little loss in accuracy and stability.

References

[1] Sabata B, Aggarwal JK, "Estimation of Motion From a Pair of Range Images: A Review," CVGIP: Image Understanding, **54**, 3, 1991, pp 309-324.

[2] Arun KS, Huang TS, Blostein SD, "Least-Squares Fitting of Two 3-D Point Sets," IEEE Transactions on Pattern Analysis and Machine Intelligence, **9**, 5, 1987, pp 698-700.

[3] Horn BKP, Hilden HM, Negahdaripour S, "Closed-form Solution of Absolute Orientation Using Orthonormal Matrices," Journal of the Optical Society of America, Series A, **5**, 7, 1988, pp 1127-1135.

[4] Horn BKP, "Closed-form Solution of Absolute Orientation using Unit Quaternions," Journal of the Optical Society of America, Series A, **4**, 4, 1987, pp 629-642.

[5] Walker MW, Shao L, Volz RA, "Estimating 3-D Location Parameters Using Dual Number Quaternions," CVGIP: Image Understanding, **54**, 3, 1991, pp 358-367.

[6] Zhang Z, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," International Journal of Computer Vision, **13**, 2, 1994, pp 119-152.

[7] Schonemann P, "A Generalized Solution of the Orthogonal Procrustes Problem," Psychometrika, **31**, 1966, pp 1-10.

[8] Umeyama S, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**, 4, 1991, pp 376-380.

[9] Kanatani K, "Analysis of 3-D Rotation Fitting," IEEE Transactions on Pattern Analysis and Machine Intelligence, **16**, 5, 1994, pp 543-549.

[10] Smith BT, Boyle JM, Ikebe Y, Klema VC, Moler CB, *Matrix Eigensystem Routines: Eispack Guide*, 2nd Ed., Springer-Verlag, New York, 1970.

[11] Lorusso A, Eggert DW, Fisher RB, "A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations," *Technical Report #737*, Department of Artificial Intelligence, University of Edinburgh, 1995.