Machine Vision and Applications (1997) 9: 272–290

Machine Vision and Applications

Estimating 3-D rigid body transformations: a comparison of four major algorithms

D.W. Eggert[1], **A. Lorusso**[2], **R.B. Fisher**[3]

[1] Department of Computer Science, University of New Haven, West Haven, CT 06516, USA, e-mail: eggert@vision.newhaven.edu
[2] Department of Artificial Intelligence, University of Edinburgh, Edinburgh EH1 2QL, Scotland, UK, e-mail: adele@houdini.icb.ge.cnr.it
[3] Department of Artificial Intelligence, University of Edinburgh, Edinburgh EH1 2QL, Scotland, UK, e-mail: rbf@aifh.ed.ac.uk

Abstract. A common need in machine vision is to compute the 3-D rigid body transformation that aligns two sets of points for which correspondence is known. A comparative analysis is presented here of four popular and efficient algorithms, each of which 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 transformation representation used and the mathematical derivation of the solution, using respectively singular value decomposition or eigensystem computation based on the standard $[\mathbf{R}, \mathbf{T}]$ representation, and the eigensystem analysis of matrices derived from unit and dual quaternion forms of the transform. This comparison presents both qualitative and quantitative results of several experiments designed to determine (1) the accuracy and robustness of each algorithm in the presence of different levels of noise, (2) the stability with respect to degenerate data sets, and (3) relative computation time of each approach under different conditions. The results indicate that under "ideal" data conditions (no noise) certain distinctions in accuracy and stability can be seen. But for "typical, real-world" noise levels, there is no difference in the robustness of the final solutions (contrary to certain previously published results). Efficiency, in terms of execution time, is found to be highly dependent on the computer system setup.

Key words: Motion analysis – 3-D rigid transformations – Pose estimation

1 Introduction

Determining the relationship between two coordinate systems through the use of sets of corresponded feature measurements is known as the *absolute orientation problem*. It has numerous applications in the areas of photogrammetry, robotics (constructing world models), object motion analysis, relating a camera coordinate system to others (the hand-eye transform), as well as estimating the position and orientation of a recognized object (pose estimation).

Correspondence to: D.W. Eggert

A recent survey by Sabata and Aggarwal (1991) lists a large number of methods that have been developed to compute the 3-D rigid body transformation between two sets of corresponded features. These techniques are categorized based on feature type (surfaces, lines or points) and general solution method (iterative vs closed form). Point features are the most commonly used in practice. Closed form solutions are generally superior to iterative methods, in terms of efficiency and robustness, because the latter suffer from the problems of not guaranteeing convergence, becoming trapped in local minima of the error function and requiring a good starting estimate. For these reasons, this paper compares only a subset of the approaches mentioned in the indicated survey: *closed form* solutions involving corresponded *point* sets. The problem of determining point correspondence, itself an active research area, is not addressed here.

Four popular and efficient solutions are compared in this paper. These differ according to the representations used for the transformation components, and in the ways they minimize a criterion function. The first solution was developed by Arun et al. (1987) and is based on computing the *singular value decomposition* (SVD) of a matrix derived from the standard $[\mathbf{R}, \mathbf{T}]$ representation. A similar approach, but based on exploiting the *orthonormal* properties of the rotation matrix, computes the *eigensystem* of a different derived matrix, as presented by Horn et al. (1988). The third algorithm, also developed by Horn (1987), involves computing the eigensystem of a matrix related to representing the rotational component as a *unit quaternion*. Yet another eigensystem is analyzed when the translation and rotation components are represented using *dual quaternions*, as presented in the fourth technique by Walker et al. (1991).

The comparison conducted here consists of three parts. First, the absolute *accuracy* of each algorithm is determined using ground truth under noiseless conditions, and then *robustness* is measured as the coordinates of corresponding points are corrupted with increasing amounts of noise. Second, the *stability* of the algorithms is quantified by finding the breakdown limits as the original 3-D point sets degenerate into such forms as a plane, a line and a single point. Lastly, the relative *efficiency* of the algorithms, 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 easier and more reliable.

2 Previous comparisons

The review by Sabata and Aggarwal (1991), while quite broad in the number of techniques described, unfortunately only presents a qualitative summary and comparison of these techniques (also, the dual quaternion method was developed after this survey). Quantitative results which would help answer the question, "Which is the best?", were not given. However, in the defining papers of these approaches there appear certain quantitative results and qualitative assessments of accuracy and speed, including the following.

Arun et al. observed that "the computer time requirements for the SVD and (unit) quaternion algorithms are comparable" (1987, p. 700). In a paper not directly related to any of the methods, Zhang implemented both of the quaternion algorithms and found that "they yield exactly the same motion estimate" (1994, p. 124). Also, he found these two techniques to be more efficient than an iterative technique based on the extended Kalman filter that he developed. Finally, Walker et al. stated that "the two algorithms produce the same rotation errors ... for the translation errors, the DQ algorithm exhibits better performance than the SVD algorithm ... " (1991, p. 364). The thorough and unbiased comparison presented here will clarify, extend (and even refute) some of these previous findings.

3 Descriptions of the algorithms

Each of the four algorithms computes the solution to a similar problem, which can be described as follows. Assume that there exist two corresponded point sets $\{m_i\}$ and $\{d_i\}$, $i = 1 \,..\, N$, such that they are related by:

$$d_i \;=\; \mathbf{R}\, m_i + \mathbf{T} + \mathbf{V}_i \tag{1}$$

where $\mathbf{R}$ is a standard 3×3 rotation matrix, $\mathbf{T}$ is a 3-D translation vector and $\mathbf{V}_i$ a noise vector. Solving for the optimal transformation $[\hat{\mathbf{R}}, \hat{\mathbf{T}}]$ that maps the set $\{m_i\}$ onto $\{d_i\}$ typically requires minimizing a *least squares error criterion* given by:

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

It is true that if outliers exist in the data set (through incorrect correspondences), this least squares solution is not optimal and other techniques should be used (Meer et al. 1991). However, for the remainder of this paper we assume only correct correspondences exist.

In the following sections, the basic steps involved in minimizing Eq. 2 are listed for each method in a common framework. This includes exactly how the rotation and translation components of the transformation are represented and computed, as well as any special cases that must be considered for a complete solution. (For complete derivations of each technique, the reader is invited to consult the appropriate paper.)

3.1 The singular value decomposition of a matrix

This first method was developed by Arun et al. (1987), and was originally designed to explicitly minimize Eq. 2.

3.1.1 Transformation representation

For this approach, the rotation is represented using a standard 3×3 orthonormal matrix. Translation is a 3-D vector, as in Eq. 1.

3.1.2 Calculating rotation

As a consequence of the least-squares solution to Eq. 2, the point sets $\{d_i\}$ and $\{m_i\}$ should have the same centroid. Using this constraint a new equation can be generated. By defining:

$$\bar{d} \;=\; \frac{1}{N} \sum_{i=1}^{N} d_i \qquad\qquad d_{\mathrm{c}_i} \;=\; d_i - \bar{d}$$

$$\bar{m} \;=\; \frac{1}{N} \sum_{i=1}^{N} m_i \qquad\qquad m_{\mathrm{c}_i} \;=\; m_i - \bar{m} \tag{3}$$

Eq. 2 can be rewritten and reduced to:

$$\begin{aligned} \Sigma^2 &= \sum_{i=1}^{N} \| \, d_{\mathrm{c}_i} - \hat{\mathbf{R}}\, m_{\mathrm{c}_i} \, \|^2 \\ &= \sum_{i=1}^{N} (d_{\mathrm{c}_i}^{\mathrm{T}}\, d_{\mathrm{c}_i} + m_{\mathrm{c}_i}^{\mathrm{T}}\, m_{\mathrm{c}_i} - 2\, d_{\mathrm{c}_i}^{\mathrm{T}}\, \hat{\mathbf{R}}\, m_{\mathrm{c}_i}) \end{aligned} \tag{4}$$

This equation is minimized when the last term is maximized, which is equivalent to maximizing Trace($\hat{\mathbf{R}}\,\mathbf{H}$), where $\mathbf{H}$ is a correlation matrix defined by:

$$\mathbf{H} \;=\; \sum_{i=1}^{N} m_{\mathrm{c}_i}\, d_{\mathrm{c}_i}^{\mathrm{T}} \tag{5}$$

If the singular value decomposition of $\mathbf{H}$ is given by $\mathbf{H} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^{\mathrm{T}}$, then the optimal rotation matrix, $\hat{\mathbf{R}}$, that maximizes the desired trace is

$$\hat{\mathbf{R}} \;=\; \mathbf{V}\,\mathbf{U}^{\mathrm{T}} \tag{6}$$

Note: minimizing Eq. 4 is also known as the *orthogonal Procrustes problem*, the SVD-based solution of which has been known for some time (Schonemann 1966). [It has also been shown by Goryn and Hein (1995) that the above solution holds in the case where both the model and data points have been corrupted by noise.]

3.1.3 Calculating translation

The optimal translation aligns the centroid of the set $\{d_i\}$ with the rotated centroid of the set $\{m_i\}$ as mentioned earlier. That is

$$\hat{\mathbf{T}} \;=\; \bar{d} - \hat{\mathbf{R}}\, \bar{m} \tag{7}$$

3.1.4 Special cases

When the determinant of $\hat{\mathbf{R}}$ is +1, all is well. However, when the two point sets are planar, or large amounts of noise exist, the determinant of $\hat{\mathbf{R}}$ may become -1, indicating a reflection rather than a rotation has been computed. In this case the desired rotation can be found as $\hat{\mathbf{R}} = \mathbf{V}'\mathbf{U}^{\mathrm{T}}$, where the matrix $\mathbf{V}' = [v_1, v_2, -v_3]$ is formed from the columns of $\mathbf{V}$, with v_3 being the column that corresponds to the singular value of $\mathbf{H}$ that is zero.

This special case has also been treated in alternative derivations by Umeyama (1991) and Kanatani (1994). Here the optimal rotation is expressed as:

$$\hat{\mathbf{R}} = \mathbf{U} \begin{pmatrix} 1 & & \\ & 1 & \\ & & \det(\mathbf{U}\,\mathbf{V}^{\mathrm{T}}) \end{pmatrix} \mathbf{V}^{\mathrm{T}} \tag{8}$$

None of the algorithms are suitable for use on linear or singular point data sets.

3.2 A solution involving orthonormal matrices

The second algorithm is similar in nature to the first, but was developed independently by Horn et al. (1988). As presented in their paper, the error criterion function was slightly different:

$$\Sigma^2 = \sum_{i=1}^{N} \| d_i - \hat{s}\,\hat{\mathbf{R}}\,m_i - \hat{\mathbf{T}} \|^2 \tag{9}$$

including a scale factor, $\hat{s}$, in the transformation. Solving for $\hat{s}$ is done independently of solving for $\hat{\mathbf{R}}$ and $\hat{\mathbf{T}}$. Therefore, in the following summary of their approach it is assumed that $\hat{s} = 1$ to enable an easier comparison.

3.2.1 Transformation representation

As in the SVD approach, rotation is represented using a 3×3 orthonormal matrix and translation is given by a 3-D vector.

3.2.2 Calculating rotation

As in the previous algorithm, the constraint of overlaying the point set centroids is used to derive Eq. 4. Again, this equation is minimized when the last term is maximized, which is also equivalent to maximizing the Trace($\hat{\mathbf{R}}^{\mathrm{T}}\,\mathbf{M}$), where

$$\mathbf{M} = \mathbf{H}^{\mathrm{T}} = \sum_{i=1}^{N} d_{\mathrm{c}_i}\, m_{\mathrm{c}_i}^{\mathrm{T}} \tag{10}$$

Assuming $\mathbf{M}$ is non-singular, it can be decomposed as $\mathbf{M} = \mathbf{U}\,\mathbf{S}$, where

$$\mathbf{U} = \mathbf{M}\,\mathbf{S}^{-1} \qquad \mathbf{S} = (\mathbf{M}^{\mathrm{T}}\,\mathbf{M})^{1/2} \tag{11}$$

$\mathbf{U}$ being an orthonormal matrix. When Trace($\hat{\mathbf{R}}^{\mathrm{T}}\,\mathbf{M}$) is maximized, $\hat{\mathbf{R}} = \mathbf{U}$ and can be expressed as:

$$\hat{\mathbf{R}} = \mathbf{M} \left(\frac{u_1\,u_1^{\mathrm{T}}}{\sqrt{\lambda_1}} + \frac{u_2\,u_2^{\mathrm{T}}}{\sqrt{\lambda_2}} + \frac{u_3\,u_3^{\mathrm{T}}}{\sqrt{\lambda_3}} \right) \tag{12}$$

where $\{\lambda_j\}$ and $\{u_j\}$ are the eigenvalues and corresponding eigenvectors of the matrix $\mathbf{M}^{\mathrm{T}}\,\mathbf{M}$.

3.2.3 Calculating translation

The optimal translation $\hat{\mathbf{T}}$ is computed using Eq. 7 given $\hat{\mathbf{R}}$ as found in Eq. 12.

3.2.4 Special cases

If the two point sets are planar, then the matrix $\mathbf{M}$ is singular and of rank two. The solution above does not hold in this case, since λ_3 becomes zero. The alternative solution presented in (Horn et al. 1988) is:

$$\hat{\mathbf{R}} = \mathbf{M} \left(\frac{u_1\,u_1^{\mathrm{T}}}{\sqrt{\lambda_1}} + \frac{u_2\,u_2^{\mathrm{T}}}{\sqrt{\lambda_2}} \right) \pm u_3\,u_3^{\mathrm{T}} \tag{13}$$

where u_3 is the eigenvector corresponding to the zero eigenvalue of the matrix $\mathbf{M}^{\mathrm{T}}\,\mathbf{M}$. The proper sign is chosen so that the determinant of $\hat{\mathbf{R}}$ will be +1.

Unfortunately, this solution is not correct in all cases and must be modified. One such modification is:

$$\hat{\mathbf{R}} = \mathbf{M}\,\mathbf{S}^{+} \pm \frac{\mathbf{X}}{\sqrt{|\ \mathrm{Trace}(\mathbf{X})\ |}} \tag{14}$$

where

$$\begin{aligned} \mathbf{S}^{+} &= \left(\frac{u_1\,u_1^{\mathrm{T}}}{\sqrt{\lambda_1}} + \frac{u_2\,u_2^{\mathrm{T}}}{\sqrt{\lambda_2}} \right) \\ \mathbf{X} &= [\ (\mathbf{M}\,\mathbf{S}^{+})\,(\mathbf{M}\,\mathbf{S}^{+})^{\mathrm{T}} - \mathbf{I}\ ]\ u_3\,u_3^{\mathrm{T}} \end{aligned} \tag{15}$$

This formulation will also hold in the general case of non-planar point sets, but is more expensive to compute. When the point sets are linear or singular, the eigenvalues λ_2 and λ_1 also become zero, indicating neither solution remains valid.

3.3 A solution involving unit quaternions

The third method is also due to Horn (1987), and was also designed to solve Eq. 9. However, in this case a new technique was developed based on a different representation for the transformation. Again, we assume $\hat{s} = 1$ in the following.

3.3.1 Transformation representation

Rather than use the standard 3×3 orthonormal matrix to represent rotation, the unit quaternion is employed. If the overall rotation is represented by a movement through an angle θ about an axis $a = [a_x, a_y, a_z]$ that is passing through the origin (with $\|a\| = 1$), the equivalent unit quaternion is a 4-D vector defined as $q = [cos(\theta/2),\ sin(\theta/2)\ a_x,\ sin(\theta/2)\ a_y,\ sin(\theta/2)\ a_z]$. Translation is still represented using a 3-D vector in the following algorithm.

3.3.2 Calculating rotation

Unit quaternions have several useful properties (Horn 1987) which can be exploited to rewrite the final term of Eq. 4 in terms of quaternions and quaternion multiplication as:

$$\Sigma' = \sum_{i=1}^{N} d_{c_i}^{\mathrm{T}}\, \hat{\mathbf{R}}\, m_{c_i} = \sum_{i=1}^{N} (\hat{q}\, \dot{m}_{c_i}\, \hat{q}^{*}) \cdot \dot{d}_{c_i} \tag{16}$$

where $\hat{q}$ is the unit quaternion representing rotation $\hat{\mathbf{R}}$, $\hat{q}^{*} = [\hat{q}_0, -\hat{q}_1, -\hat{q}_2, -\hat{q}_3]$, and $\dot{m}_{c_i}$ and $\dot{d}_{c_i}$ are quaternions formed by setting their first component to zero and the remaining three components to the corresponding point's coordinates. The expression in the above equation can be further rewritten as $\Sigma' = \hat{q}^{\mathrm{T}}\, \mathbf{P}\, \hat{q}$, where $\mathbf{P}$ is a 4×4 matrix as given in Eq. 17. For each of the elements in this matrix the S terms are sums of products of point coordinates similar in nature to the elements of $\mathbf{H}$ in Eq. 5: $S_{ab} = \sum_{i=1}^{N} m_{c_{i_a}}\, d_{c_{i_b}}$. The quaternion $\hat{q}$ which maximizes Eq. 16 is the eigenvector associated with the largest positive eigenvalue of matrix $\mathbf{P}$. Given this $\hat{q}$, the equivalent rotation matrix $\hat{\mathbf{R}}$ can be found as in Eq. 18.

$$\mathbf{P} = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & S_{yy}-S_{xx}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & S_{zz}-S_{xx}-S_{yy} \end{bmatrix} \tag{17}$$

$$\hat{\mathbf{R}} = \begin{bmatrix} (\hat{q}_0^{\,2}+\hat{q}_1^{\,2}-\hat{q}_2^{\,2}-\hat{q}_3^{\,2}) & 2\,(\hat{q}_1\,\hat{q}_2-\hat{q}_0\,\hat{q}_3) & 2\,(\hat{q}_1\,\hat{q}_3+\hat{q}_0\,\hat{q}_2) \\ 2\,(\hat{q}_2\,\hat{q}_1+\hat{q}_0\,\hat{q}_3) & (\hat{q}_0^{\,2}-\hat{q}_1^{\,2}+\hat{q}_2^{\,2}-\hat{q}_3^{\,2}) & 2\,(\hat{q}_2\,\hat{q}_3-\hat{q}_0\,\hat{q}_1) \\ 2\,(\hat{q}_3\,\hat{q}_1-\hat{q}_0\,\hat{q}_2) & 2\,(\hat{q}_3\,\hat{q}_2+\hat{q}_0\,\hat{q}_1) & (\hat{q}_0^{\,2}-\hat{q}_1^{\,2}-\hat{q}_2^{\,2}+\hat{q}_3^{\,2}) \end{bmatrix} \tag{18}$$

3.3.3 Calculating translation

Once the matrix $\hat{\mathbf{R}}$ has been computed from the elements of $\hat{q}$, the optimal translation $\hat{\mathbf{T}}$ can again be found using Eq. 7.

3.3.4 Special cases

When the point sets are planar the above algorithm does not need to be modified, but depending on the method used to compute the eigensystem of $\mathbf{P}$, certain simplifications can be made. Horn (1987) presents such an alternative algorithm for planar point sets, but it will not be used in this comparison. Neither of Horn's methods are intended for linear or singular point data sets.

3.4 A solution involving dual quaternions

The fourth algorithm is due to Walker et al. (1991), and is the most significantly different of the four. The original error criterion function which they minimized is:

$$\Sigma^2 = \sum_{i=1}^{L} \alpha_i \parallel n_{1_i} - \hat{\mathbf{R}}\, n_{2_i} \parallel^2 + \sum_{i=1}^{N} \beta_i \parallel d_i - \hat{\mathbf{R}}\, m_i - \hat{\mathbf{T}} \parallel^2 \tag{19}$$

where $\{n_{1_i}\}$ and $\{n_{2_i}\}$ are two sets of L corresponding unit normal vectors, and $(\{\alpha_i\}, \{\beta_i\})$ are weighting factors reflecting data reliability. Thus this approach incorporates both positional and directional information in the minimization. In order to compare this approach with the others, it is assumed that $\alpha_i = 0$ and $\beta_i = 1$ so that only point positions are relevant. However, note that if independent direction information is available, this algorithm can integrate both data types, unlike the previous algorithms in their current form.

3.4.1 Transformation representation

In this method the rotation and translation are represented together using a dual quaternion. As the name suggests, it is a quaternion consisting of two parts, $q_d = [r, s]$. Rather than describing motion as a rotation about the origin and then a translation, it can be modelled as a simultaneous rotation around and translation along a particular line in 3-D space. Given that the line has direction $n = [n_x, n_y, n_z]$ ($\|n\| = 1$), passes through a point $p = [p_x, p_y, p_z]$, and the amount of motion is defined by an angle θ and a distance t, then the two components of q_d are 4-D vectors defined as:

$$r = \begin{bmatrix} \sin(\theta/2)\, n \\ \cos(\theta/2) \end{bmatrix}$$
$$s = \begin{bmatrix} \frac{t}{2}\, \cos(\theta/2)\, n + \sin(\theta/2)\, (p \times n) \\ -\frac{t}{2}\, \sin(\theta/2) \end{bmatrix} \tag{20}$$

These two components of the dual quaternion have the properties that $r^{\mathrm{T}} r = 1$ and $s^{\mathrm{T}} r = 0$.

3.4.2 Calculating rotation

The restricted version of Eq. 19 can be rewritten in terms of the components of the optimal dual quaternion $\hat{q}_d = [\hat{r}, \hat{s}]$ as:

$$\begin{aligned} \Sigma^2 = \sum_{i=1}^{N} & (\hat{s}^{\mathrm{T}}\, \hat{s} + 2\, \hat{s}^{\mathrm{T}}\, (\mathbf{W}(\tilde{m_i}) - \mathbf{Q}(\tilde{d_i}))\, \hat{r} - \\ & 2\, \hat{r}^{\mathrm{T}}\, \mathbf{Q}(\tilde{d_i})\, \mathbf{W}(\tilde{m_i})\, \hat{r} + (\tilde{m_i}^{\mathrm{T}}\, \tilde{m_i} + \tilde{d_i}^{\mathrm{T}}\, \tilde{d_i})) \\ = {} & \hat{r}^{\mathrm{T}}\, \mathbf{C}_1\, \hat{r} + \hat{s}^{\mathrm{T}}\, \mathbf{C}_2\, \hat{s} + \hat{s}^{\mathrm{T}}\, \mathbf{C}_3\, \hat{r} + \mathbf{C}_4 \end{aligned} \tag{21}$$

where $\tilde{m_i}$ and $\tilde{d_i}$ are 4-D vectors defined with the first three components set equal to half the values of the original point's coordinates and the fourth component set to zero. The 4×4 matrices $\mathbf{Q}(v)$ and $\mathbf{W}(v)$ are defined by:

$$\mathbf{Q}(v) = \begin{bmatrix} v_3\mathbf{I} + \mathbf{K}(v_{0..2}) & v_{0..2} \\ -v_{0..2}^{\mathrm{T}} & v_3 \end{bmatrix} \quad \mathbf{K}(v) = \begin{bmatrix} 0 & -v_2 & v_1 \\ v_2 & 0 & -v_0 \\ -v_1 & v_0 & 0 \end{bmatrix}$$
$$\mathbf{W}(v) = \begin{bmatrix} v_3\mathbf{I} - \mathbf{K}(v_{0..2}) & v_{0..2} \\ -v_{0..2}^{\mathrm{T}} & v_3 \end{bmatrix} \tag{22}$$

where $v_{0..2}$ represents the first three elements of v. The $\mathbf{C}$ terms are given as:

$$\mathbf{C}_1 = \sum_{i=1}^{N} \mathbf{Q}(\tilde{d_i})\, \mathbf{W}(\tilde{m_i}) \qquad \mathbf{C}_2 = N\, \mathbf{I}$$
$$\mathbf{C}_3 = 2 \sum_{i=1}^{N} (\mathbf{W}(\tilde{m_i}) - \mathbf{Q}(\tilde{d_i})) \tag{23}$$
$$\mathbf{C}_4 = \sum_{i=1}^{N} (\tilde{m_i}^{\mathrm{T}}\, \tilde{m_i} + \tilde{d_i}^{\mathrm{T}}\, \tilde{d_i})$$

Table 1. Summary of four motion algorithms. Included are the transformation representation form, method of solution, and the range of data configurations for which solutions were given

Cited Work	Abbreviation	Transform Representation	Solution Technique	Data point configurations handled			
				3-D	2-D	1-D	0-D
Arun et al. (1987)	SVD	**R**, **T**	use svd of derived matrix	yes	special case	no	no
Horn et al. (1988)	OM	**R**, **T**	use eigensystem of derived matrix	yes	special case (modified)	no	no
Horn (1987)	UQ	**q**, **T**	use largest eigenvector of derived matrix	yes	yes	no	no
Walker et al. (1991)	DQ	$\mathbf{q_d} = (\mathbf{r}, \mathbf{s})$	use largest eigenvector of derived matrix	yes	yes	no	no

In order to properly minimize Eq. 21 the two constraints $\hat{r}^{\,\mathrm{T}}\hat{r} = 1$ and $\hat{s}^{\,\mathrm{T}}\hat{r} = 0$ are incorporated using Lagrange multipliers. Then it can be derived that given the 4×4 matrix **A**:

$$\begin{aligned}\mathbf{A} &= \tfrac{1}{2}\,(\mathbf{C}_3^{\mathrm{T}}\,(\mathbf{C}_2 + \mathbf{C}_2^{\mathrm{T}})^{-1}\,\mathbf{C}_3 - \mathbf{C}_1 - \mathbf{C}_1^{\mathrm{T}}) \\ &= \tfrac{1}{4N}\,\mathbf{C}_3^{\mathrm{T}}\,\mathbf{C}_3 - \mathbf{C}_1\end{aligned} \tag{24}$$

the value of $\hat{r}$ is the eigenvector corresponding to the largest positive eigenvalue of **A**. The vector $\hat{r}$ can be used to calculate the equivalent rotation matrix $\hat{\mathbf{R}}$ as:

$$\hat{\mathbf{R}} = (\hat{r}_3^{\,2} - \hat{r}_{0..2}^{\mathrm{T}}\,\hat{r}_{0..2})\,\mathbf{I} + 2\,\hat{r}_{0..2}\,\hat{r}_{0..2}^{\mathrm{T}} + 2\,\hat{r}_3\,\mathbf{K}(\hat{r}_{0..2}) \tag{25}$$

Equation 25 is equivalent to the quaternion to matrix conversion of Eq. 18.

3.4.3 Calculating translation

The values of $\hat{s}$ and $\hat{\mathbf{T}}$ are calculated based on $\hat{r}$ as:

$$\begin{aligned}\hat{s} &= -(\mathbf{C}_2 + \mathbf{C}_2^{\mathrm{T}})^{-1}\,\mathbf{C}_3\,\hat{r} = -\tfrac{1}{2N}\,\mathbf{C}_3\,\hat{r} \\ \hat{\mathbf{T}} &= \mathbf{W}(\hat{r})^{\,\mathrm{T}}\,\hat{s}\end{aligned} \tag{26}$$

3.4.4 Special cases

As with the unit quaternion algorithm, there is no need for modification in the case where the points sets are planar, while linear and singular point sets are still not handled.

3.5 Summary

A summary of the four techniques mentioned in this section is given in Table 1. The listing shows the representation used for each motion transform, the general solution method, and which data configurations the algorithms were intended to handle. The indicated abbreviations are used in all of the following experimental descriptions.

4 Experimental comparison

Each of the four algorithms was implemented and a series of experiments performed to determine their accuracy, robustness, stability and speed. In the following the results of these experiments are detailed.

4.1 The implementations

All experiments were conducted on a 50-MHz Sun Sparcstation 10 with a 1-MB cache running Solaris 2.4. The implementations were coded in C++ and compiled using the GNU compiler[1]. While the majority of the coding was straightforward, a decision was necessary concerning which routines to use in computing the SVD and eigensystem of a matrix.

In Horn's work (1987, 1988), it is stated that the various eigensystems can be computed by analytically solving cubic and quartic characteristic equations to find the eigenvalues, and then using Gaussian elimination to compute the eigenvectors (the SVD of a matrix can also be calculated analytically using this eigensystem algorithm). The authors of the other papers used routines from various mathematical packages. Preliminary experiments showed the standard routines were in general more stable and accurate than Horn's analytic approach. Therefore, to provide another level of commonality, routines from the Eispack[2] linear algebra package (Smith et al. 1970) were used in all. Singular value decomposition was performed using the *svd* routine, while eigensystems were computed using a combination of the *tred2* and *tql2* functions. Since these standard functions are in fact iterative, the overall closed form nature of each solution was technically violated. However, these routines are well known for their stability and rapid convergence, and no problems were observed.

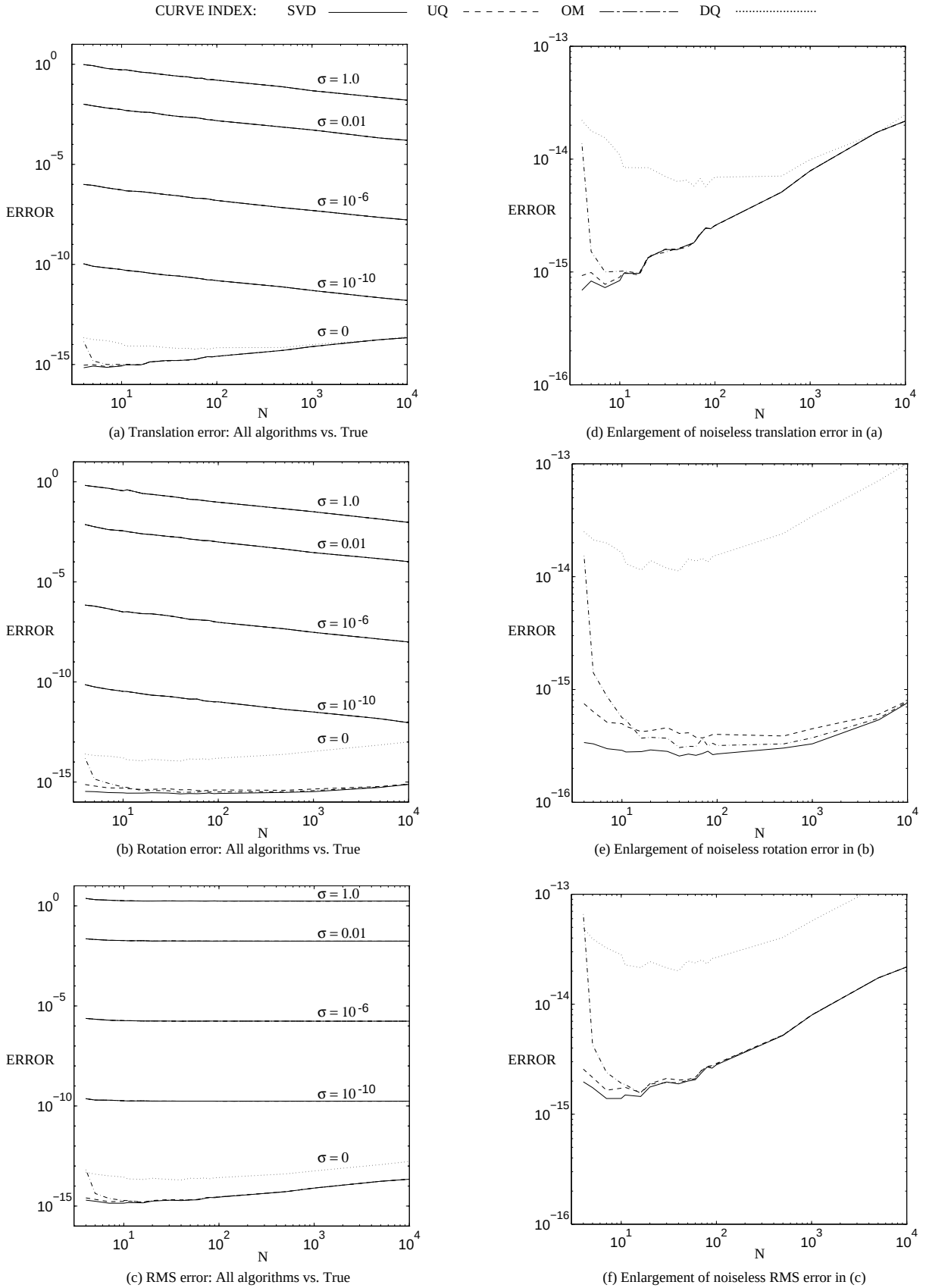
4.2 The accuracy/robustness experiment

In this experiment, the absolute accuracy of each of the algorithms was tested by comparing generated output motions to ground truth transformations under noiseless conditions. The robustness was measured by examining the error in output transformations under varying levels of noise in the input. The data for these tests were generated as follows:

(1) Several different model sets of non-degenerately arranged 3-D points were created, ranging from a minimum of $N = 4$ points (the smallest non-planar set) to a maximum of $N = 10,000$ points. The points in these sets, $\{m_i\}$, were chosen randomly from a uniform distribution within a cube of size $2 \times 2 \times 2$ centered about the origin. Each corresponding data set, $\{d_i\}$, was formed by adding noise to the

[1] Code was compiled using gcc, version 2.6.3, using -O3 level optimization.

[2] This package was converted from Fortran to C using standard utilities (f2c). Its routines generally have formed the basis of those in almost all other packages, including Numerical Recipes in C (Press et al. 1992).



(a) Translation error: All algorithms vs. True

(d) Enlargement of noiseless translation error in (a)

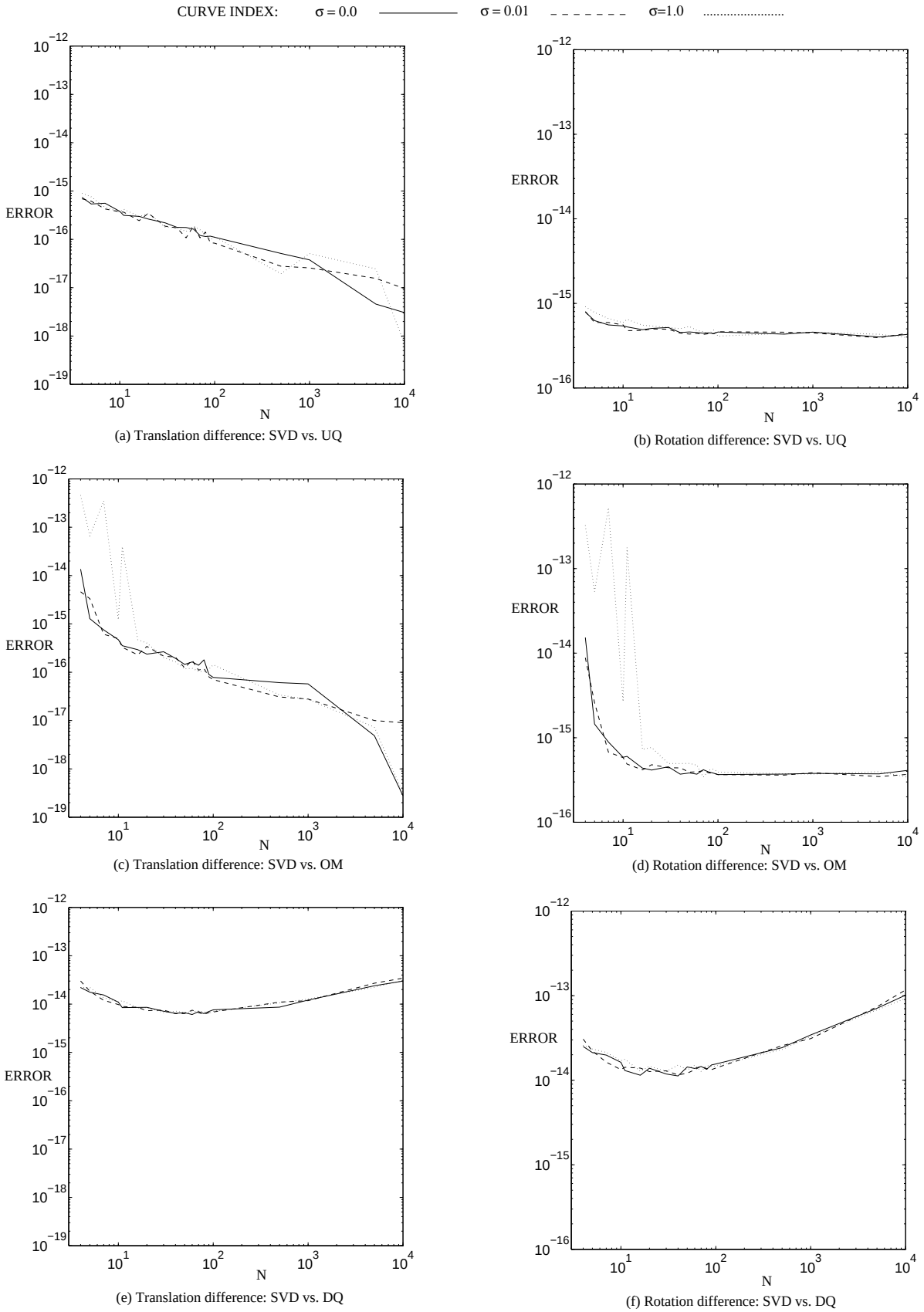
(b) Rotation error: All algorithms vs. True

(e) Enlargement of noiseless rotation error in (b)

(c) RMS error: All algorithms vs. True

(f) Enlargement of noiseless RMS error in (c)

Fig. 1. Differences between known transformations and those computed by algorithms. Graphs (log-log scale) are translation error, rotation error, and RMS error vs data set size, N. A range of noise levels, $\sigma = 0.0$ to $\sigma = 1.0$, are shown in the left column (**a–c**), while the noiseless case is enlarged in the right column (**d–f**)



(a) Translation difference: SVD vs. UQ

(b) Rotation difference: SVD vs. UQ

(c) Translation difference: SVD vs. OM

(d) Rotation difference: SVD vs. OM

(e) Translation difference: SVD vs. DQ

(f) Rotation difference: SVD vs. DQ

Fig. 2. Differences between transformations computed by the SVD algorithm and the UQ (**a, b**), the OM (**c, d**) and the DQ (**e, f**) algorithms. Graphs (log-log scale) are translation error, $\| \hat{\mathbf{T}}_{svd} - \hat{\mathbf{T}}_{other} \|$, and rotation error, $\| \hat{q}_{svd} - \hat{q}_{other} \|$, vs data set size, N, for various noise levels σ = 0.0, 0.01, 1.0

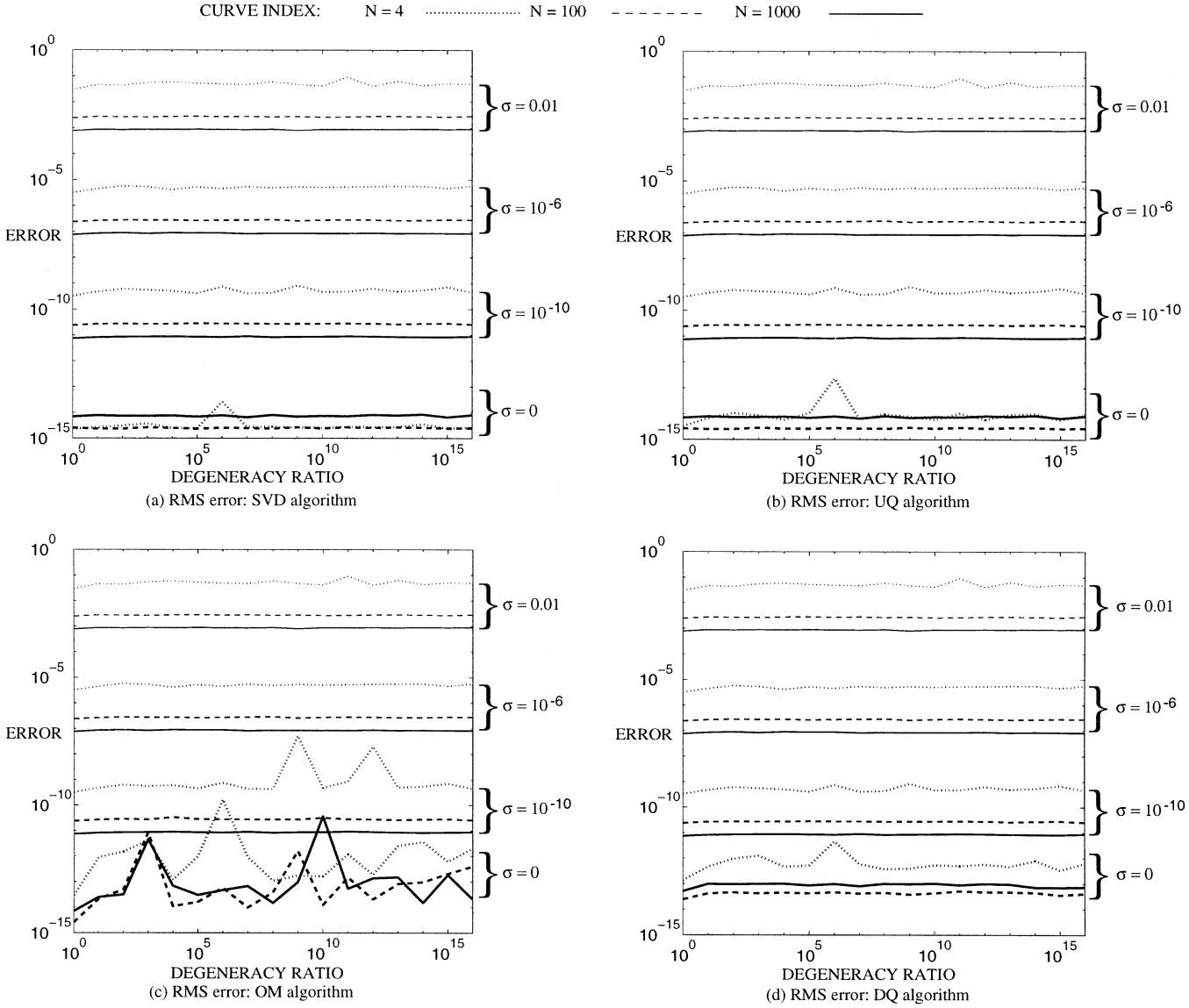


Fig. 3. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 2-D degenerate data sets under *isotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

individual points in $\{m_i\}$ and transforming them to a new location[3]. The noise added to each component was uncorrelated, isotropic and Gaussian in nature, with a zero mean and variable standard deviation ranging from a minimum of zero to a maximum of one unit.

(2) Transformations were generated in two parts. A 3-D translation was computed by randomly selecting components uniformly distributed in the range $[-10 \ \ldots \ 10]$. Each rotation was calculated from a unit quaternion that was randomly selected from a uniform distribution representing valid rotations (those having random components in the range $[-1 \ \ldots \ 1]$ subject to the magnitude of the quaternion originally being less than 1, and then eventually normalized).

(3) For each data set size/noise level combination one hundred trials were performed, a trial consisting of new points, noise and transformation values. The average response of each algorithm over these one hundred trials was used to compute three different error statistics for the calculated transforms. These included the norm of the difference between the true and estimated translation vectors, $T_{err} = \| \hat{\mathbf{T}}_{alg} - \hat{\mathbf{T}}_{true} \|$, as well as the norm of the difference between true and estimated unit quaternions representing the rotation, $q_{err} = \| \hat{q}_{alg} - \hat{q}_{true} \|$. Also the root mean square (RMS) error of the distance between corresponding points in the model and adjusted data sets under the found transformation was calculated:

$$RMS_{err} = \sqrt{\frac{\sum_{i=1}^{N} \| d_i - \hat{\mathbf{R}}_{alg}\, m_i - \hat{\mathbf{T}}_{alg} \|^2}{N-3}}$$

[3] Normally noise is added after a transformation. But, if the transform is Euclidean and the noise is isotropic, both techniques are equivalent. Later experiments will require special anisotropic noise to be added, which is most easily done before the transformation. Therefore a consistent method was used throughout.

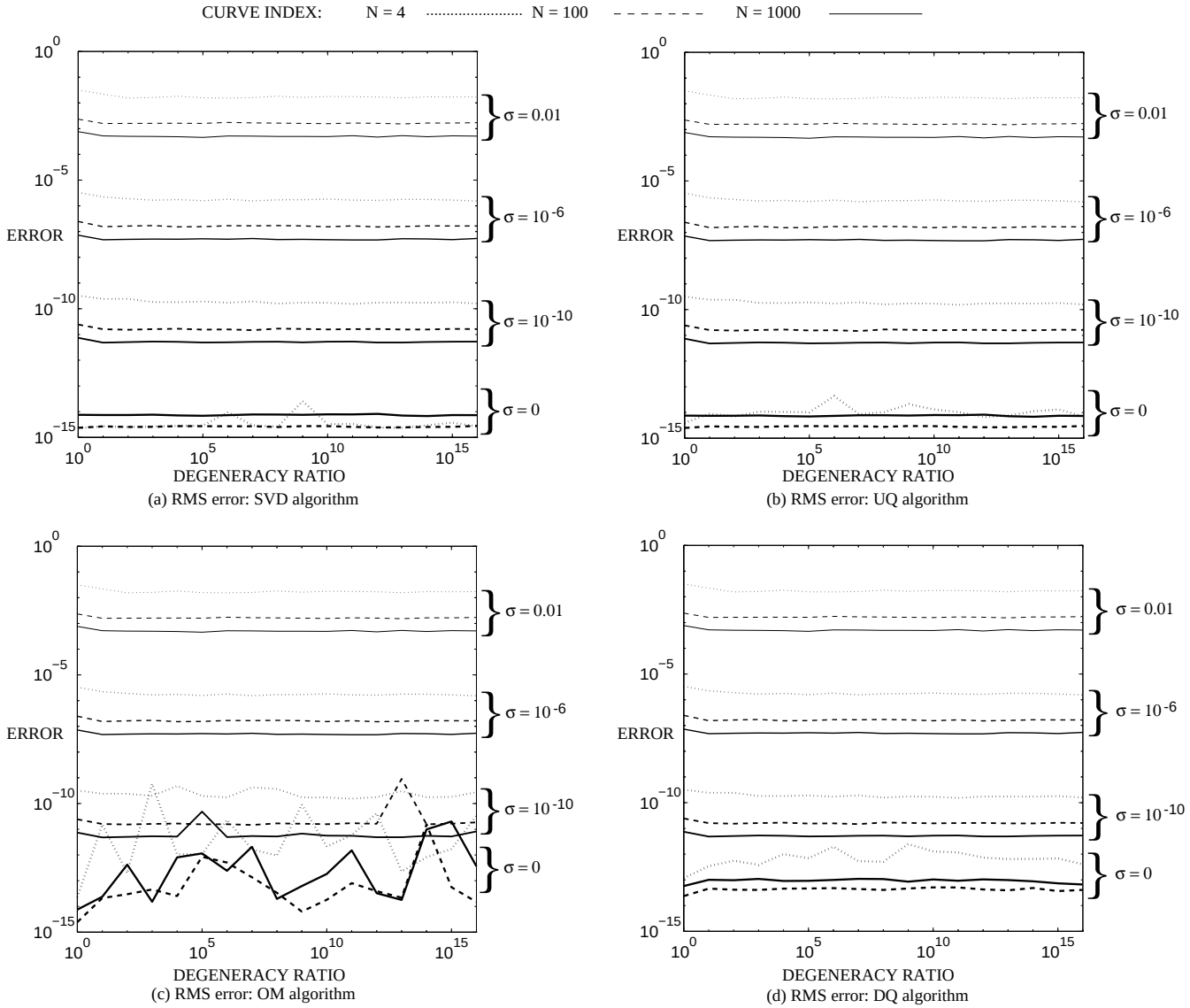


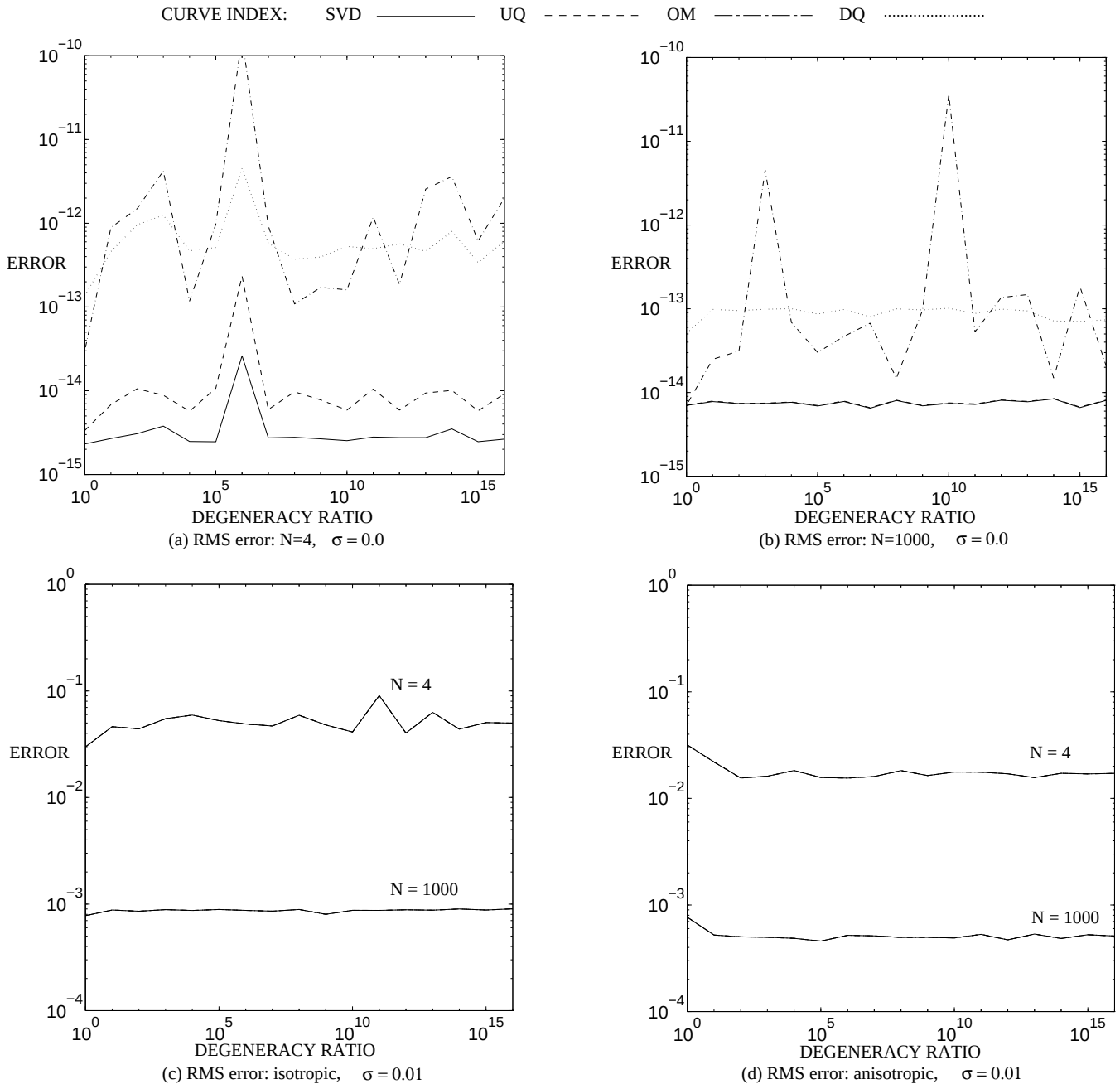
Fig. 4. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 2-D degenerate data sets under *anisotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

The first two of these error measures were also computed between pairs of algorithms to determine any levels of difference between them.

The left column of Fig. 1 shows the robustness of the four algorithms with regards to changes in noise level in terms of the translation, rotation and RMS errors, respectively. As might be expected, when the number of points grows, the error in the computed transformation approaches a value dependent on the noise level used to disturb the data (the RMS error actually approaches $\sqrt{3}\sigma$). The exception to this is the noiseless case, the graphs of which are enlarged in the right column of Fig. 1. Here it is seen that the absolute accuracy of the dual quaternion algorithm is less than that of the others, which are very similar to one another except on small data sets, where SVD seems the most accurate.

Figure 2 tries to emphasize any actual differences in translation and rotation solutions between algorithms. The comparisons shown are between the SVD algorithm and the UQ (Fig. 2a,b), OM (Fig. 2c,d) and DQ (Fig. 2e,f) algorithms. As one can see, the differences are really minimal (in most cases near the level of machine precision, 2×10^{-16}), and virtually independent of noise level. Even the largest differences, 10^{-12} to 10^{-13}, would not be considered noticeable. The overall shape of the curves also suggests that as the data set size increases, the individual algorithm differences lose their significance.

So, from the results shown here, one can conclude that the difference in solutions produced by the algorithms is many orders of magnitude below typical RMS noise levels in real data, almost at the level of machine precision. Only in the theoretically noiseless case can any real differences be observed. Then, the UQ and SVD results are most similar, deviating from that of OM for small data sets, while these three generally perform better than the DQ method.



(a) RMS error: N=4, $\sigma = 0.0$

(b) RMS error: N=1000, $\sigma = 0.0$

(c) RMS error: isotropic, $\sigma = 0.01$

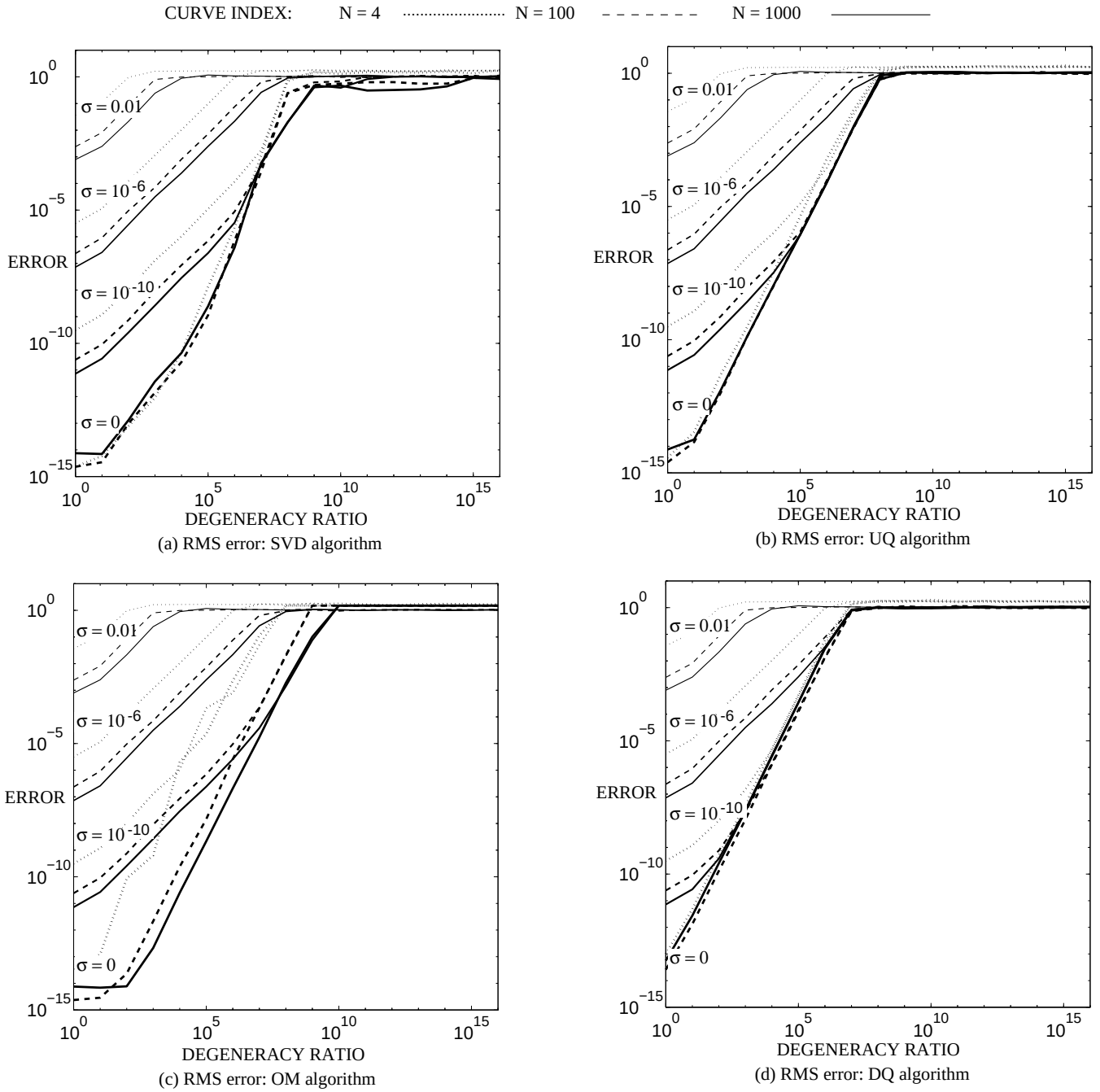
(d) RMS error: anisotropic, $\sigma = 0.01$

Fig. 5. Graphs (log-log scale) of RMS errors of algorithms on 2-D degenerate data sets with no noise (**a,b**), isotropic noise (**c**), and anisotropic noise (**d**)

4.3 Stability experiments

In this section the stability of the algorithms is tested, i.e. how well they compute a correct transformation under different data conditions. In most cases the algorithms were designed for non-degenerate 3-D data sets, perhaps with additional modifications for planar data. However, none were designed to handle data points arranged in a linear or singular point relation due to the unconstrained degrees of freedom in the solution. The following subsections explore the differences in algorithm performance as the data sets approach these degenerate forms. The relative stabilities of the algorithms, as well as the actual levels of degeneracy necessary to cause breakdown, are given.

A sequence of data sets is necessary to monitor the breakdown of the algorithms. Successive point sets in the sequence were taken from a volume whose dimensions were steadily reduced from that of an original $2 \times 2 \times 2$ cube to those of either a 2×2 square, a 2-unit-long line, or a single point. The level of degeneracy present in a particular point set in the sequence is measured as the $degeneracy\ ratio = 2/d$, where d is the current size of the cube in the shrinking dimension. In the experiments this degeneracy ratio varies from a value of one for the initial cube to an upper limit of 10^{16} for the most degenerate form. Each data set in the sequence was constructed using the process described in the



(a) RMS error: SVD algorithm

(b) RMS error: UQ algorithm

(c) RMS error: OM algorithm

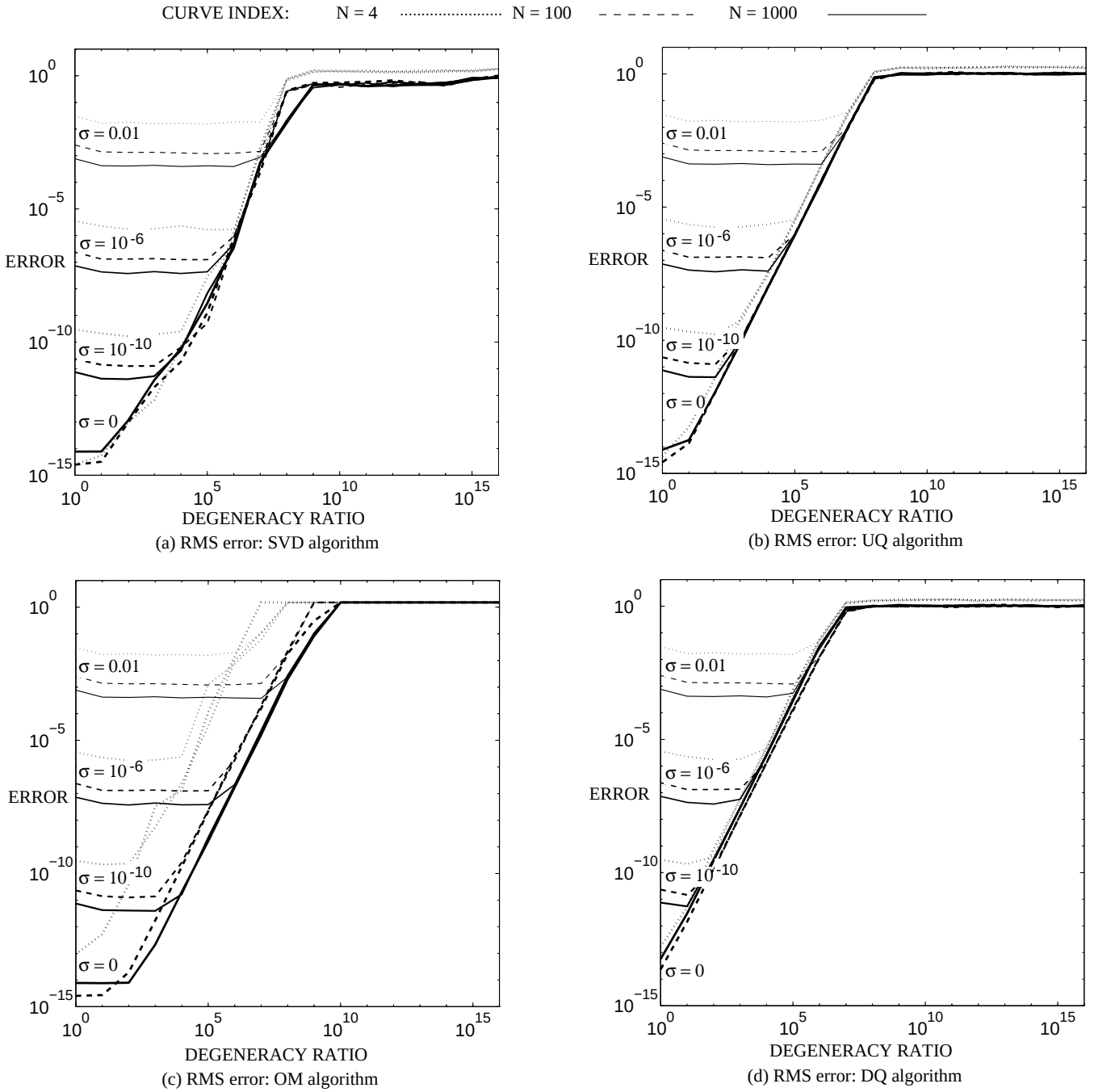
(d) RMS error: DQ algorithm

Fig. 6. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 1-D degenerate data sets under *isotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

previous section, with the exception that two different noise models were used.

The first model, isotropic noise, represents the assumption that no prior uncertainty information is known about the data. The second model, anisotropic noise, represents the belief that data uncertainty is limited to certain dimensions. Here, the more accurate dimensions are along the direction of degeneracy (for instance, perpendicular to a plane). The level of noise added in this direction is less than that in the other dimensions by an amount proportional to the volume's decreasing size. Adding this form of noise in any other direction would only lead to responses similar to those resulting from isotropic noise.

The error criterion used to measure algorithm stability is RMS error. However, it is not appropriate to simply compute this error between the transformed data points and the original model points. This is because several transforms can produce similar error sums on degenerate data sets. For instance, even for largely different rotations about the axis of a cylinder (representing a thick line), a data point does not move far from its original position when the cylinder dimensions approach that of a line. Therefore, in order to distinguish between such transforms, a control data set is used.



(a) RMS error: SVD algorithm

(b) RMS error: UQ algorithm

(c) RMS error: OM algorithm

(d) RMS error: DQ algorithm

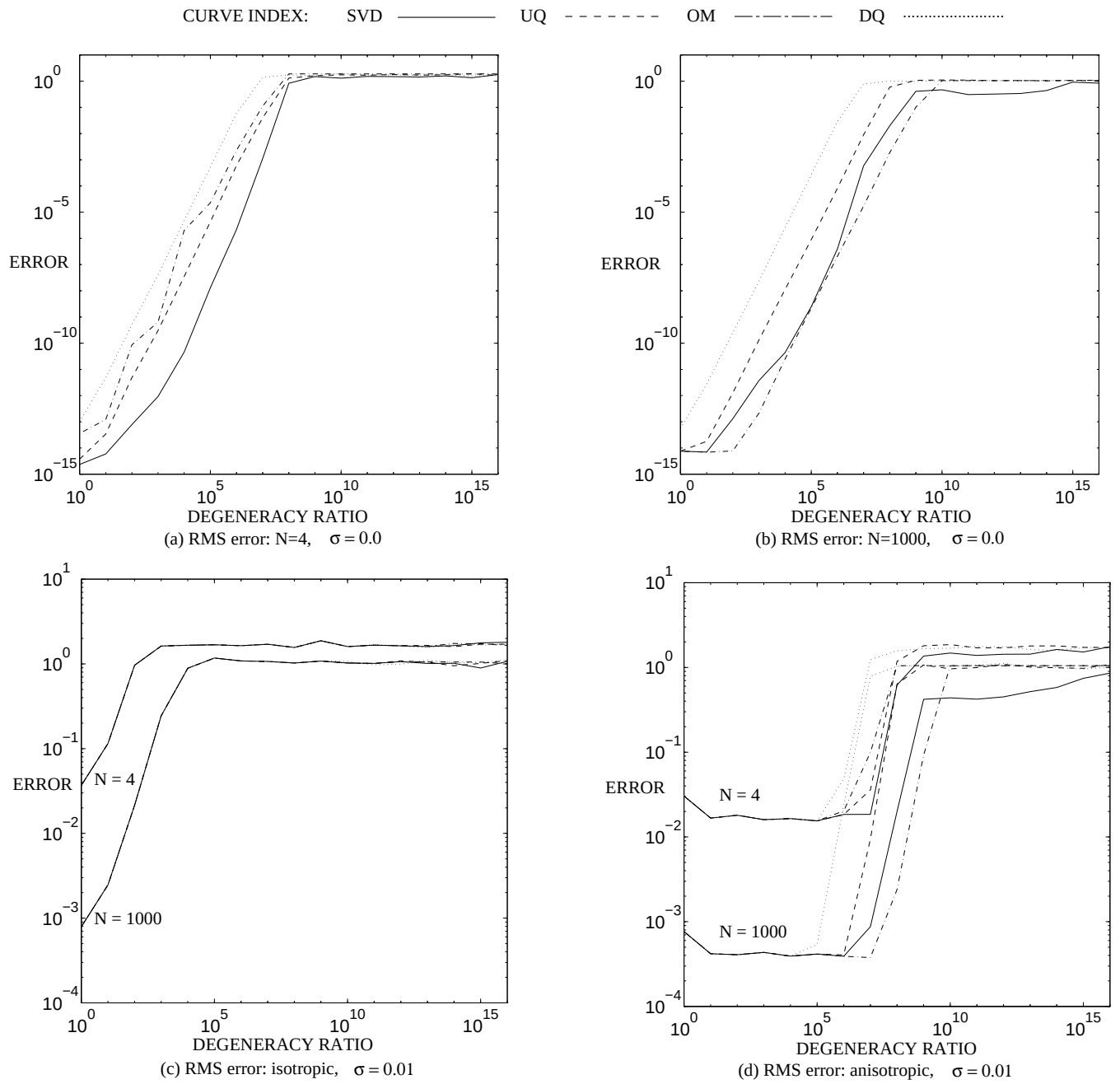
Fig. 7. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 1-D degenerate data sets under *anisotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

This consists of randomly chosen points equal in number to that used in the computation, but taken from the original $2 \times 2 \times 2$ cube. When this data set is transformed according to the true and estimated motions, the RMS calculation using these points provides a distinguishing error measure.

4.3.1 Stability under 3-D → 2-D degeneracy

In this experiment each data set degenerates into a plane. The Z dimension of the 3-D cube is steadily shrunk from its 2-unit size down to the level of machine precision, 2×10^{-16}. Figure 3 shows the error responses of the algorithms with respect to the isotropic noise model. As the degeneracy ratio is increased, the data set more closely resembles a plane. Curves are drawn for varying noise levels and data set sizes. Figure 4 shows an equivalent range of responses when anisotropic noise is used[4].

[4] Here, the noise level, σ, is the same in the X and Y dimensions as was used for the isotropic case, but the new value of σ in the Z dimension is computed by dividing the original value by the degeneracy ratio, thereby reducing it for the more planar-like sets.



(a) RMS error: N=4, $\sigma = 0.0$

(b) RMS error: N=1000, $\sigma = 0.0$

(c) RMS error: isotropic, $\sigma = 0.01$

(d) RMS error: anisotropic, $\sigma = 0.01$

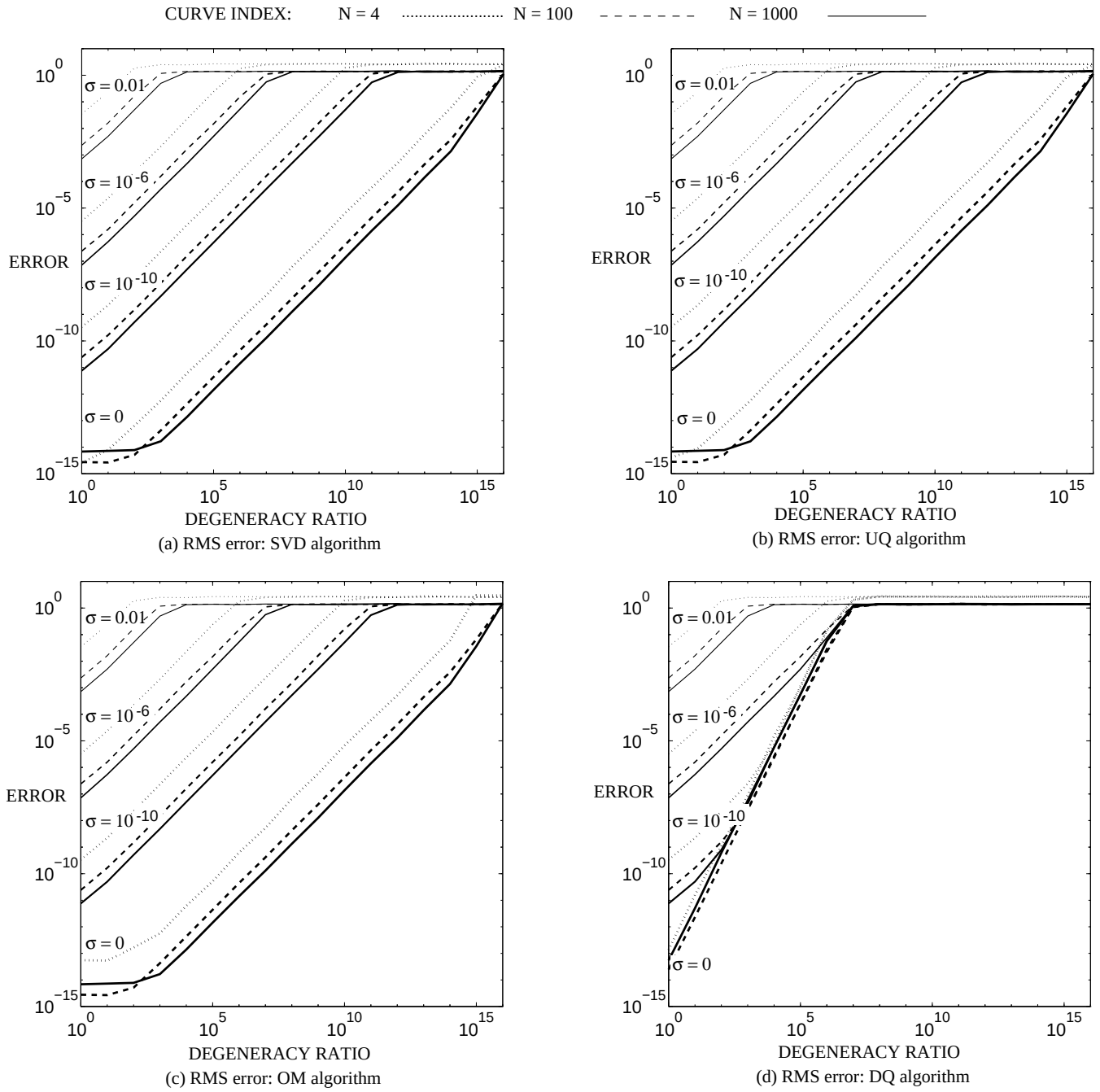
Fig. 8. Graphs (log-log scale) of RMS errors of algorithms on 1-D degenerate data sets with no noise (**a,b**), isotropic noise (**c**), and anisotropic noise (**d**)

The error values in both sets of graphs show, for the most part, an independence from the degeneracy value. This is to be expected, since the algorithms were designed to handle planar point sets. The only deviations from this trend occur in the noiseless case. Here, the response from the OM algorithm is less numerically stable than the others (perhaps suggesting the correction provided in Eq. 14 is not the best alternative). Figure 5 provides a more direct comparison of the algorithms at the extremes of the test sets. Figures 5c and 5d show that under noisy conditions the responses of the algorithms are the same. But when noise is not present (Fig. 5a,b), only the SVD and UQ approaches remain similar, while both the DQ and OM techniques produce less stable answers.

4.3.2 Stability under 3-D → 1-D degeneracy

In this experiment each data set degenerates into a line. The X and Y dimensions of the 3-D cube are steadily shrunk from their 2-unit size down to the level of machine precision. Figures 6 and 7 show the error responses of the algorithms under isotropic and anisotropic noise, again for four levels of noise and three data set sizes.

None of the algorithms were designed to handle linearly organized data sets. The graphs in Figs. 6 and 7 show the



(a) RMS error: SVD algorithm

(b) RMS error: UQ algorithm

(c) RMS error: OM algorithm

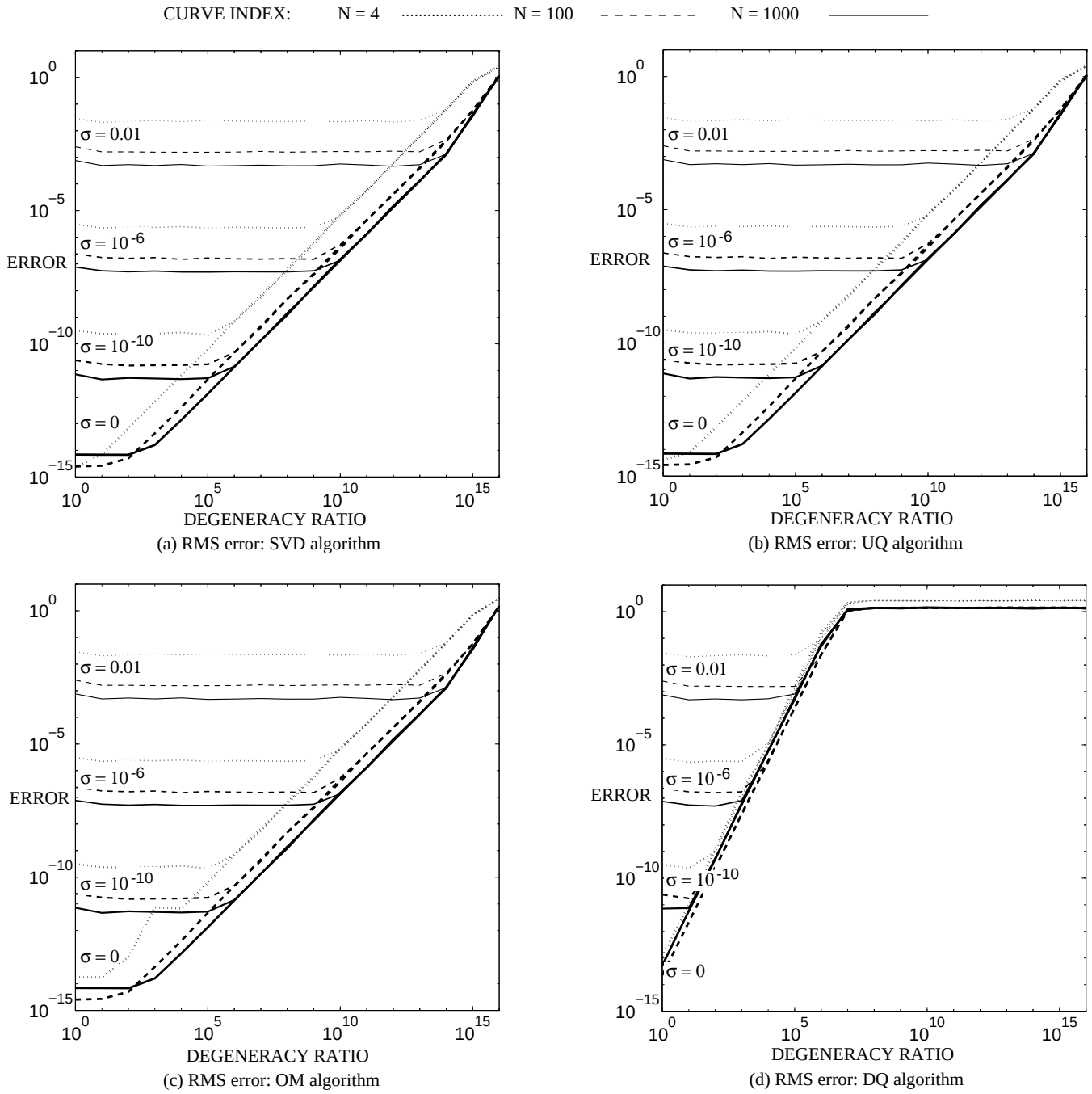
(d) RMS error: DQ algorithm

Fig. 9. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 0-D degenerate data sets under *isotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

different levels of degeneracy necessary to cause breakdown. Under isotropic noise the error steadily rises as the diameter of the tube representing the line decreases. Total breakdown in the best cases occurs only after the line diameter reaches the level of the noise, and under most other conditions earlier than that. For anisotropic noise[5] the trend is for the error to remain at the noise level until numeric instability occurs, and then continue to rise steadily as the line diameter decreases, finally reaching total breakdown.

Figure 8 shows a direct comparison of the relative stabilities of the methods under the extreme conditions. Under isotropic noise the algorithms behave similarly, independent of these conditions (Fig. 8c). Under either anisotropic noise (Fig. 8d) or no noise at all (Fig. 8a,b), the differences in numerical stability become apparent. The DQ algorithm breaks down first in all cases, while the UQ curves are always less stable than those of SVD. The OM algorithm's performance is highly dependent on the data set size, actually being the most stable for large data sets.

[5] In the anisotropic case the noise level in the Z dimension is the same as in the isotropic case, while that in the X and Y dimensions is reduced by the value of the degeneracy ratio.



(a) RMS error: SVD algorithm

(b) RMS error: UQ algorithm

(c) RMS error: OM algorithm

(d) RMS error: DQ algorithm

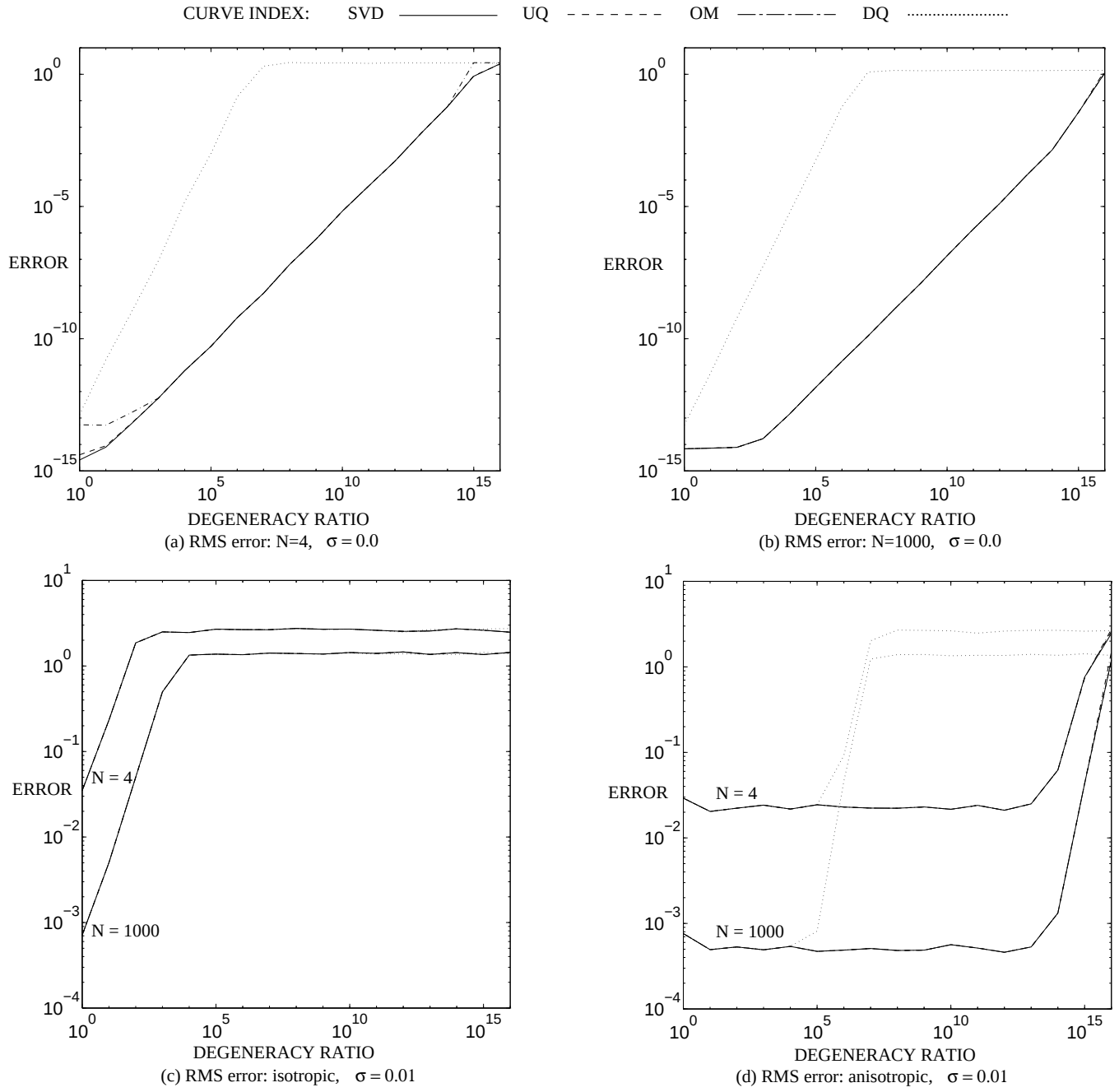
Fig. 10. RMS errors of **a** SVD, **b** UQ, **c** OM and **d** DQ algorithms for 0-D degenerate data sets under *anisotropic* noise. Curves on graphs (log-log scale) present errors for three data set sizes, N = 4, 100, 1000, and for four noise levels, $\sigma = 0.0, 10^{-10}, 10^{-6}, 0.01$

4.3.3 Stability under 3-D → 0-D degeneracy

In this experiment each data set degenerates into a point. All three dimensions of the 3-D cube are steadily shrunk from their 2-unit size down to the level of machine precision. Figures 9 and 10 give the response curves of the algorithms for the isotropic and anisotropic noise cases, respectively.

As with the linear data sets, the algorithms will break down because there are too many unconstrained degrees of freedom in the solution. However, the graphs in Figs. 9 and 10 are slightly different from those in the previous experiment. Under isotropic noise the same steady rise in error is seen as the cube size decreases. Here, total breakdown is not achieved by any of the algorithms, except DQ, until the size of the cube reaches the noise level. DQ's stability is apparently less than the others in this case, especially for small amounts of noise. In the second case[6], when the amount of anisotropic noise is reduced, the error level of the response does not change until a common breakdown curve is joined.

[6] In the anisotropic case, the noise is actually isotropic, but the level is reduced by the degeneracy ratio in all three dimensions from that in the regular case.



(a) RMS error: N=4, $\sigma = 0.0$

(b) RMS error: N=1000, $\sigma = 0.0$

(c) RMS error: isotropic, $\sigma = 0.01$

(d) RMS error: anisotropic, $\sigma = 0.01$

Fig. 11. Graphs (log-log scale) of RMS errors of algorithms on 0-D degenerate data sets with no noise (**a,b**), isotropic noise (**c**), and anisotropic noise (**d**)

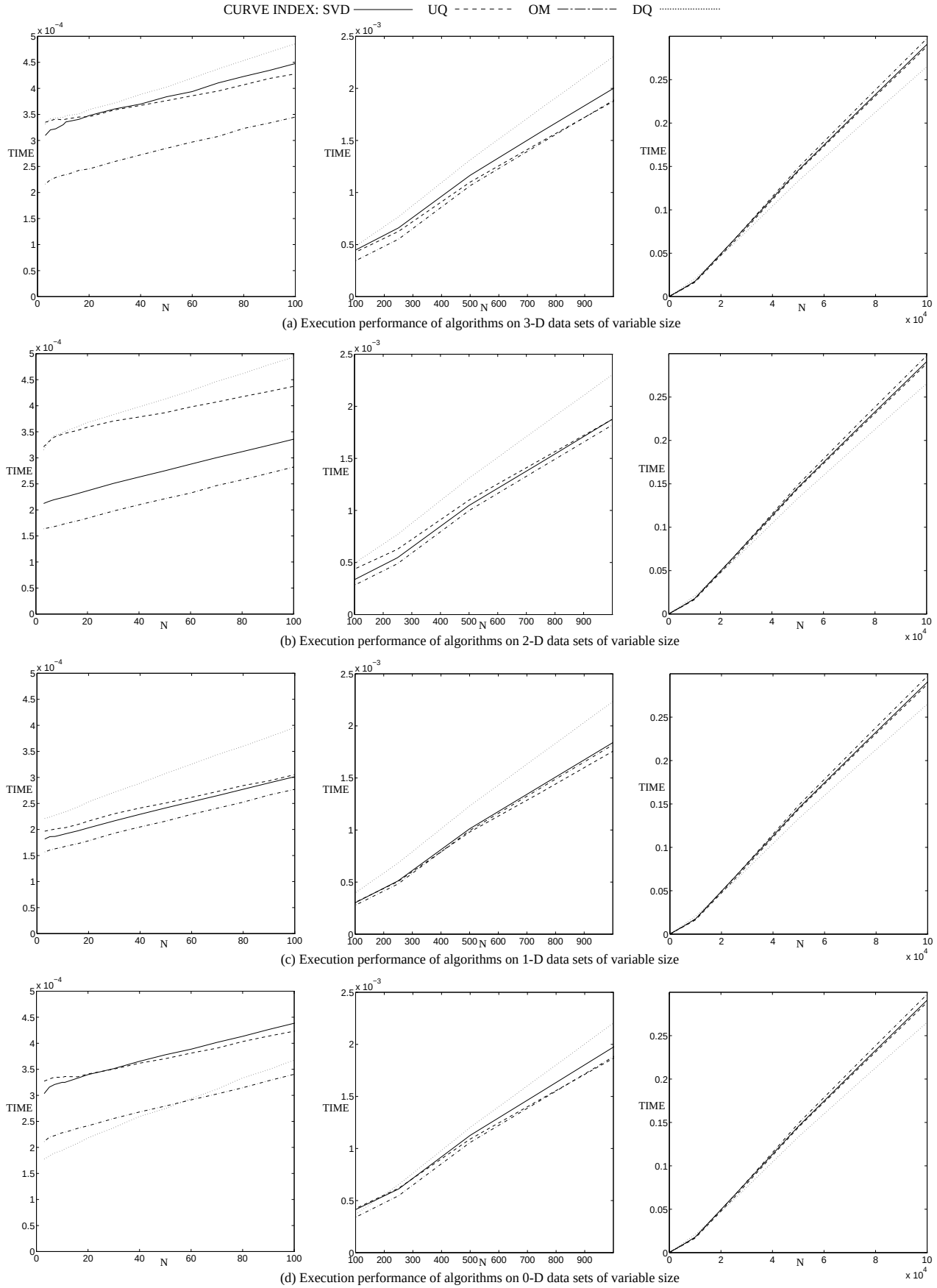
Again, the DQ algorithm does not respond as robustly as the others.

The magnitude of the differences in stability between the DQ algorithm and the others is made very apparent in the comparison graphs of Fig. 11. Here it can be seen that breakdown occurs much earlier, except for large levels of isotropic noise. The other responses are virtually identical, with the OM curves breaking down slightly sooner if a small data set size is used. However, even this apparently large level of difference is not necessarily overly significant, since the level of degeneracy required to cause any breakdown under typical noise conditions might not be reached.

4.4 The efficiency experiment

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 examined. Timings were performed on both small and large data sets under different conditions. The timing diagrams for the noiseless cases on different data types and sizes are shown in Fig. 12. Other timings observed under varying levels of noise were virtually identical to these, and so are not included here.

Let us first consider the coefficient of the linear term of the time equation. All of the algorithms, except DQ, initially spend time computing the same set of sums (the outer prod-



(a) Execution performance of algorithms on 3-D data sets of variable size

(b) Execution performance of algorithms on 2-D data sets of variable size

(c) Execution performance of algorithms on 1-D data sets of variable size

(d) Execution performance of algorithms on 0-D data sets of variable size

Fig. 12. Execution times (in seconds) of algorithms on non-degenerate and degenerate data sets of sizes $N = 4 \ldots 100$ (*left*), $N = 100 \ldots 1000$ (*center*), and $N = 4 \ldots 100,000$ (*right*)

ucts of the centered point coordinates in Eq. 5). Therefore the linear terms, which are dependent on this computation, should be identical for these three. The DQ algorithm computes a different set of sums. The relative values of these two terms are shown most clearly in the slopes of the graphs in the middle and right columns of Fig. 12.

The right column makes it apparent that two basic slopes exist for each algorithm's response. This is a side-effect of the size of the computer's memory cache. Although the computations of the DQ routine are greater in number, they make fewer memory accesses for the data coordinates. This means that on smaller data sets, which fit entirely in the cache, the DQ algorithm is less efficient. But as the data size increases, so do the cache misses, reversing the relative efficiencies of the calculations. For the 1-MB cache used here, the transition occurs at approximately $N = 10,000$. The smaller deviations in the slope between the three algorithms other than DQ are attributed to minor caching effects during the timing process.

Next, consider the constant terms of the time equations. Here each algorithm's execution varies and can be broken into two components: that necessary to compute an eigensystem or SVD of a matrix, and the accompanying calculations to generate the matrix. Measurements of the auxiliary calculations for the various routines were respectively: DQ ($6\,\mu$s), UQ ($5\,\mu$s), SVD (with reflection correction $12\,\mu$s, without $7\,\mu$s) and OM (with planar correction $41\,\mu$s, without $18\,\mu$s). These small amounts are only a fraction of the time necessary for the eigensystem and SVD computations. Approximate timings on non-degenerate data were: SVD of 3×3 matrix $350\,\mu$s, eigensystem of 4×4 matrix $275\,\mu$s and eigensystem of 3×3 matrix $150\,\mu$s. The combined effect of these times can be seen in the leftmost portion of Fig. 12a. Because the SVD and eigensystem routines are iterative, the approximate times just listed can vary based on changes in data set size and degeneracy (see the remainder of the leftmost column in Fig. 12). The variability of these routines also contributes to crossovers between certain curves, as can be seen in the center column.

So, overall, if the data set size is less than 100, the OM algorithm is the quickest, by as much as 30%[7]. For large data set sizes the relative difference between the SVD, UQ and OM methods becomes negligible, a few percent. Prior to the data size exceeding the cache's effectiveness ($N < 10,000$) the DQ technique is slower than the others, but eventually it becomes the most efficient as N increases.

5 Conclusions

A comparison has been made between four algorithms which 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. In fact, for any practical application there will be no discernible differences in accuracy or stability. Actual quantitative measures of accuracy and stability have been presented in the previous section. A qualitative summary of algorithm performance is given in Table 2. From this, some final general conclusions can be drawn.

As mentioned, the accuracy and robustness of the algorithm responses for non-degenerate 3-D point sets, even for various levels of noise, is virtually the same. This is perhaps not unexpected since they were designed to solve the same problem. This conclusion is in agreement with all but the previous findings of Walker et al. (1991) mentioned earlier in the paper. It is not known how they derived their conclusion of superior accuracy for the DQ algorithm, but it is certainly not supported by the data presented here.

A greater separation of algorithm performance was noticed in the stability testing. Here, in most cases, the SVD and UQ methods were very similar, with SVD marginally more stable. The OM method was not as stable for planar data sets, but was superior on certain degenerate data sets of large size. In none of the testing was the DQ algorithm more stable than the others, and usually it would break down before them. However, under realistic amounts of noise, the level of degeneracy necessary to expose these differences is likely greater than what will occur in practice.

In terms of efficiency, for small data set sizes the OM algorithm appears quickest, given the setup reported here. On larger data sets the differences in speed among all but the quicker DQ algorithm are not overly significant. The memory configuration of the computer is an important factor in determining when the relative speed of the DQ algorithm becomes superior.

So, in conclusion, it appears that it is only possible to show a difference in the accuracy and stability of the presented algorithms in certain "ideal" situations. In any real world application possessing even a low level of noise, one would not expect to notice any differences in the algorithm solutions. The only truly distinguishing factor is execution time. And to make an informed decision about which is fastest, one must carefully consider the data set size, the linear algebra package, the operating system and the computer hardware configuration (memory cache) being used.

Finally, mention should be made of another recent algorithm due to Wang and Jepson (1994). In their work they consider the weighted least squares equation:

$$\sum_{i=1}^{N} c_i \, d_i \;=\; \mathbf{R} \sum_{i=1}^{N} c_i \, m_i + \sum_{i=1}^{N} c_i \, \mathbf{T} \tag{27}$$

By appropriately choosing the values, $\{c_i\}$, the rotation term can be eliminated and the translation computed. Similarly the translation term can be eliminated, or the calculated value substituted, in order to compute the rotation using eigensystem methods. Preliminary results (Wang and Jepson 1994) indicate that superior performance may be obtained because translation is computed independent of rotation, thereby avoiding compounding any error in the rotation result, especially for data sets far from the origin. However, this performance depends on the robustness of the calculation of the set of coefficients, $\{c_i\}$. This calculation works

[7] However, it should be noted that the timings for small data sets are highly dependent on the particular linear algebra routines being used. For instance, execution times of corresponding functions in the Numerical Recipes in C implementation are between two and four times faster than those in the converted Eispack set. Under these conditions the relative ordering can change as the auxiliary calculations become more significant. See Lorusso et al. (1995) for alternate timings using these routines. The accuracy and stability results are not significantly different for the two math packages.

Table 2. Qualitative comparison of algorithm performance (1 = best, 4 = worst). Ratings are based on the overall responses to different noise levels and data set sizes. Comparisons are given for accuracy/robustness using 3-D data sets both with and without noise; stability of response on degenerate data sets corrupted with no noise, isotropic noise (i-noise) and anisotropic noise (a-noise); and the overall execution time for small and large data sets

Method	3-D accuracy		2-D stability			1-D stability			0-D stability			Execution time	
	ideal	noise	ideal	i-noise	a-noise	ideal	i-noise	a-noise	ideal	i-noise	a-noise	small N	large N
SVD	1	1	1	1	1	2	2	2	3	1	1	2	2
OM	3	1	4	4	4	1	1	1	1	1	1	1	4
UQ	2	1	2	1	1	3	3	3	1	1	1	2	3
DQ	4	1	3	1	1	4	4	4	4	4	4	4	1

best on non-degenerate 3-D data. Since Wang and Jepson are still examining this issue, at their request their method has not been included in the comparison here.

Acknowledgements. This work was funded by EC HCM Project ERB40500 PL921003 through the SMART network and by UK EPSRC Grant GR/H/86905. A shorter version of this work appears in the proceedings of the 6th British Machine Vision Conference. Many thanks to Andrew Fitzgibbon for helpful conversations and insights along the way.

References

Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-D point sets. IEEE Trans Pattern Anal Machine Intell 9:698–700

Goryn D, Hein S (1995) On the estimation of rigid body rotation from noisy data. IEEE Trans Pattern Anal Machine Intell 17:1219–1220

Horn BKP (1987) Closed-form solution of absolute orientation using unit quaternions. J Opt Soc Am Ser A 4:629–642

Horn BKP, Hilden HM, Negahdaripour S (1988) Closed-form solution of absolute orientation using orthonormal matrices. J Opt Soc Am Ser A 5:1127–1135

Kanatani K (1994) Analysis of 3-D rotation fitting. IEEE Trans Pattern Anal Machine Intell 16:543–549

Lorusso A, Eggert DW, Fisher RB (1995) A comparison of four algorithms for estimating 3-D rigid transformations. Technical Report 737, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland

Meer P, Mintz D, Rosenfeld A, Kim DY (1991) Robust regression methods for computer vision: a review. Int J Comput Vision 6:59–70

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in C: the art of scientific computing, 2nd edn., Cambridge University Press, Cambridge, UK

Sabata B, Aggarwal JK (1991) Estimation of motion from a pair of range images: a review. CVGIP: Image Understanding 54:309–324

Schonemann P (1966) A generalized solution of the orthogonal procrustes problem. Psychometrika 31:1–10

Smith BT, Boyle JM, Ikebe Y, Klema VC, Moler CB (1970) Matrix eigensystem routines: Eispack guide, 2nd edn., Springer Verlag, New York Berlin Heidelberg

Umeyama S (1991) Least-squares estimation of transformation parameters between two point patterns. IEEE Trans Pattern Anal Machine Intell 13:376–380

Walker MW, Shao L, Volz RA (1991) Estimating 3-D location parameters using dual number quaternions. CVGIP: Image Understanding 54:358–367

Wang Z, Jepson A (1994) A new closed-form solution for absolute orientation. Proc IEEE Conference on Computer Vision and Pattern Recognition, pp 129–134

Zhang Z (1994) Iterative point matching for registration of free-form curves and surfaces. Int J Comput Vision 13:119–152

David W. Eggert received the BSEE and BSCpE (1986), MSCS (1988), and PhD (1991) degrees from the University of South Florida. During his time at USF, he was named IEEE Student Engineer of the Year (1985 and 1986) and Sigma Xi's outstanding Master's (1988) and PhD (1991) Student of the Year. During the time of the work reported in this paper, he was serving a a postdoctoral researcher in the Department of Artificial Intelligence at the University of Edinburgh. He is currently an Assistant Professor in the Department of Computer Science at the University of New Haven. His research interests include object recognition using aspect graphs, reverse engineering of CAD models, computer graphics and robotics.

Adele Lorusso graduated (summa cum laude) in Computer Science from the Universita' di Bari (Italy) in 1991. After working as a software engineer for Olivetti, she held a 2-year position at Tecnopolis CSATA (Bari) as a researcher in computer vision and robotics. During the summer of 1994 she was a visiting researcher at the Department of Artificial Intelligence of the University of Edinburgh, after which she worked for 1 year as a researcher at the Cybernetic and Biophysics Institute of the National Council of Research in Genoa. Her main research interests are in computer vision and robotics.

Robert B. Fisher received a BS with honors (Mathematics) from California Institute of Technology (1974) and an MS (Computer Science) from Stanford University (1978). He received his PHD from the University of Edinburgh (awarded 1987), investigating computer vision in the Department of Artificial Intelligence. Dr. Fisher is a Senior Lecturer at the Department of Artificial Intelligence, University of Edinburgh. His research covers topics in high-level computer vision, and he directs research projects investigating three-dimensional model-based vision, automatic model acquisition and robot grasping. He teaches general and industrial vision courses for undergraduate, MSc and PHD level students.

This article was processed by the author using the LaTeX style file *pljour2* from Springer-Verlag.