On the Robustness of Absolute Orientation

Ross J. Micheals Terrance E. Boult

Abstract—**Ideally, absolute orientation (AO) algorithms should be both efficient and robust. Many researchers in need of a solution to the AO problem have used various least-squares approaches, such as those established and popularized by B. K. P. Horn, or O. Faugeras. Although least-squares approaches can perform well with additive Gaussian noise of fixed variance, mismatches and outliers have a more profound effect. In this paper, the authors introduce a new, closed-form solution to the AO problem. This new closed-form is then extended into new AO algorithms with specific provisions for handling outliers and mismatches. Similar extentions are also made to the traditional least-squares approach. Included is a comparitive analysis of the various strengths and weaknesses of both the previous least-squares approaches and the new techniques.**

Keywords—**robust control, robot control, mobile robots, pose estimation**

I. Introduction

Although the precise definition of the *absolute orientation* (AO) problem varies across the literature of robotics, computer vision, computer graphics, and photogrammetry, its essence remains the same: how can one determine the transformation between two different three-dimensional coordinate systems?

In the context of this paper, the *absolute orientation problem* is defined as: the determination of the translational, rotational, and uniform scalar correspondence between two different Cartesian coordinate systems given a collection of corresponding point pairs.

II. Previous Research

In 1959, E. H. Thompson published one of the first "solutions" to the absolute orientation problem [1]. Months later, Schut published his own extension of Thompson's work [2] and employed quaternions to significantly reduce its complexity. In 1975, Sansò [3] developed a solution that would later be independently rediscovered by Horn in 1987 [4]. Their derivations are fundamentally different, however, both reduce the rotational component of the sought-after transform to an Eigen-decomposition of the same matrix. Since Sansò mentions using Jacobi's method for the decomposition, he considers his method to be an "exact" solution, but not necessarily closed. Because of their "ready availability," [4] Horn also recommends the use of an iterative method such as Jacobi's for implementation. Regardless, Horn shows that his method is closed by discussing the potential application of Ferrari's method for solving the roots of a quartic. One of the strengths of Sansò's or Horn's methods is not necessarily that one is "exact" or "closed" — it is that neither require an initial approximation.

Meanwhile, the least-squares, high-order geometric primitives based approach by Faugeras and Hebert [5] published in 1983 became (and continues to be) a popular algorithm. Since then, there have been many approaches that incorporate higher-order geometric primitives, including [6], [7], [8], [9]. Researchers have also continued to expand the standard point-based least-squares method ([10], [11], [12], [13], [14]). The novel solution introduced in [15] uses Fourier transforms to bypass the need for explicit correspondences. Gradient-descent techniques have been explored by [16] and [17].

Unlike the vast majority of previous AO research (including most of the aforementioned references) this paper explicitly addresses two various noise phenomenon — *outliers* and *mismatches* — in both algorithm development and analysis. In Section III-A a formal definition of the AO problem is introduced. A new closed-form solution to the AO problem is introduced in Section III-B. In Section IV, two new AO algorithms are introduced; each on a different point along the robustness vs. computational effciency curve. The first new AO algorithm, R4 is based on the new closed-form and has specific provisions for handling outliers and mismatches. The second new AO algorithm, L4, is a marriage between traditional least-squares and some of the techniques developed for R4. Finally, in Section VII, traditional least-squares LS, R4 and L4 are compared.

III. Absolute Orienation Formulae

A. Formalization and Notation

Consider a set of n three dimensional column vectors $R = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n\}$ where $\mathbf{r}_i = [r_{i_x} r_{i_y} r_{i_z}]^T$ and $1 \leq i \leq n$. Let S represent the corresponding set of n points after undergoing a translation, rotation, and uniform scaling. Pre-transform vector $\mathbf{r}_i \in R$ and its corresponding post-transform vector $\mathbf{s}_i \in S$ are related by the equation

$$\mathbf{s}_i = kM(\theta, \hat{\mathbf{u}})\mathbf{r}_i + \mathbf{t}, \tag{1}$$

where $\mathbf{t}$ represents a translation, k represents a uniform scaling, and M represents the matrix form of the rotational component determined by the angle θ and axis $\hat{\mathbf{u}}$.

Often, it is convenient to use the representation

$$(\mathbf{s}_i - \mathbf{c}_s) = kM(\theta, \hat{\mathbf{u}})(\mathbf{r}_i - \mathbf{c}_r), \tag{2}$$

where $\mathbf{c}_r$ and $\mathbf{c}_s$ represent separate pre-transform and post-transform translational components (which are usually, but not necessarily, the centroids of R and S). $\mathbf{t}$, $\mathbf{r}_t$, and $\mathbf{s}_t$ may be related by solving equation (2) for $\mathbf{t}$:

$$\mathbf{t} = \mathbf{s}_t - kM(\theta, \hat{\mathbf{u}})\mathbf{r}_t. \tag{3}$$

The notation of equation (2) more closely reflects the methods typically used in multi-stage absolute orientation algorithms. In this paper, the terms *pre-transform translational component* and *post-transform translational component* will be used to describe the vectors $\mathbf{c}_r$ and $\mathbf{c}_s$ respectively. Note, if $\mathbf{c}_r = \mathbf{0}$, then $\mathbf{t} = \mathbf{c}_s$ and if $\mathbf{c}_s = \mathbf{0}$, then $\mathbf{t} = kM(\theta, \hat{\mathbf{u}})\mathbf{c}_r$.

Formally, the absolute orientation problem is: given R and S, determine k, $\mathbf{t}$, θ, and $\hat{\mathbf{u}}$. In this paper, like previous research, it is assumed that the uniform scalar component is unit (i.e. $k = 1$). Extending the new algorithms to encorporate scaling is straightfoward.

B. The New Closed Form

For the sake of deriving the new form, we assume that our input data (R and S) is noiseless, complete, and correctly matched (in later sections the effects of different types of noise and outliers are more closely examined). If $\mathring{q}$ is the unit quaternion

$\mathring{q} = (sin(\theta/2), cos(\theta/2)\hat{\mathbf{u}})$, then it can be shown that the post-rotation vector $\mathbf{s}_i$ is uniquely determined by the quaternion multiplication $(0, \mathbf{s}_i) = \mathring{q}(0, \mathbf{r}_i)\mathring{q}^*$ where $(0, \mathbf{r}_i)$ is a quaternion with a scalar component of zero and vector component $\mathbf{r}_i$, and $\mathring{q}^*$ is the conjugate of quaternion $\mathring{q}$. Equation (1) may be rewritten as $\mathbf{s}_i = \mathring{q}(0, \mathbf{r}_i)\mathring{q}^* + \mathbf{t}$.

Fully expanding (s_{i_x}, s_{i_y} and s_{i_z}) algebraically yields Equations (5)–(7) of Figure 1. From these equations, it follows that the components of the sought-after quaternion cannot be expressed as a direct linear combination of the input points. However, if the *product* of two of the quaternion components are considered instead, we can establish a linear relationship.

As shown in Equation (8) Figure 1, let $\mathbf{R}_i$ represent a 3×10 matrix built from the input vector $\mathbf{r}_i$. Also, let $\mathbf{Q}$ represent the matrix of unknown quaternion component products:

$$\mathbf{Q} = \begin{bmatrix} q_{sx} & q_{sy} & q_{sz} & q_{xy} & q_{yz} & q_{xz} & q_{s^2} & q_{x^2} & q_{y^2} & q_{z^2} \end{bmatrix} \quad (4)$$

where $q_{sx} = q_s q_x$, $q_{sy} = q_s q_y$, ..., $q_{z^2} = q_z q_z$. Using these simple constructs, any four distinct point pairs may be used to build a linear system such as the one shown in Equation (9). (Note $i \neq j \neq k \neq l$). The first twelve constraints follow directly from Equation (8). The 13^{th} constriant is the unit quaternion identity $\|\mathring{q}\| = 1$. If there is no translational component — i.e. $\mathbf{t} = 0$, then the system may be greatly simplified. This is equivalent to the case in which the pre-transform and post-transform translational components are known *a priori*, and the input data is normalized accordingly. In this special case, only three distinct point pairs are needed to build a system such as that shown in Equation (10). Again, the unit quaternion constraint is used to fully constrain the system.

An algebraic manipulation package such as **Maple**™may be used to obtain closed-form solutions for the unknowns of both Equation (9) and (10). In fact, there exists a unique (and somewhat surprising) relationship between the two closed-forms. Consider the following:

1. Let R and S represent corresponding sets of noiseless, perfectly matched input data points, where $|R| = |S| = 4$.
2. Let $\mathbf{Q}_4(R, S)$ and $\mathbf{t}_4(R, S)$, functions of R and S, represent closed-form solutions to $\mathbf{Q}$ and $\mathbf{t}$ as derived from Equation (9).
3. Let $\mathbf{Q}_3(R, S)$, a function of R and S, represent the closed-form solution to $\mathbf{Q}$ as derived from Equation (10).
4. Let $\mathring{q}_3$ represent a unit quaternion extracted from $\mathbf{Q}_3$.[1]
5. Let $\mathbf{c}_r$ and $\mathbf{c}_s$ represent the centroids of R and S respectively.
6. Let R_c and S_c represent the sets R and S normalized with respect to their centroids $\mathbf{c}_r$ and $\mathbf{c}_s$.

Note that in Step 3 above, $\mathbf{Q}_3(R, S)$ requires three input points, although $|R| = |S| = 4$. Any non-coplanar subset of R and S is sufficient as arguments to Q_3. The selection of this subset in the presence of noise is discussed later.

Given the above, it may be shown that $\mathbf{Q}_4(R, S)$ is *algebraically* equivalent to $\mathbf{Q}_3(R_c, S_c)$ (and therefore will be equivalent even in the presence of noise). However, $\mathbf{t}(R, S)$ and $\mathbf{t}_3$, where $\mathbf{t}_3 = \mathbf{c}_s - \mathring{q}_3 \mathring{\mathbf{c}}_r \mathring{q}_3^*$, are *not* algeraially equivalent, and therefore differ significantly in the presence of noise. Through extensive experimentation, the authors have discovered that $\mathbf{t}_3(R, S)$ is far more robust than $\mathbf{t}_4(R, S)$. This is most likely due to the significantly lower numer of computations required for calculating $\mathbf{t}_3$; in $\mathbf{t}_4$, there are more opportunities for errors to propagate.

This unique identity allows significant reduction in the number of calculations required to calculate $\mathbf{Q}$. It can be shown (the authors used the mathematical software package **Maple**™) that a common subexpression elimination optimized version of $\mathbf{Q}_4(R, S)$ and $\mathbf{t}_4(R, S)$ requires 618 floating-point additions and 472 floating-point multiplications (1090 FLOPS total). A similarly optimized $\mathbf{Q}_3(R, S)$ requires only 266 FLOPS, which does not include the 24 FLOPS required for determining $\mathbf{c}_r$ and $\mathbf{c}_s$.

Consider the rotational component of a least-squares solution such as Horn's [4]. The most computationally intensive component of Horn's solution is an Eigenvector/Eigenvalue decomposition. According to [18], a typical Jacobi rotation over an $m \times m$ matrix, requires $6m$ operations if both Eigenvectors and Eigenvalues are calculated. Since "typical" matricies require $3m^2$ to $5m^2$ Jacobi rotations, the total number of operations required for such a matrix is therefore within the range of $18m^3$ to $30m^4$. Since in Horn's method $m = 4$, we may estimate that the Eigen-decomposition requires $1,152$ to $1,920$ operations. Additionally, given n input points, $14 + 18n$ operations are required to construct the matrix to be Eigen-decomposed and $6n$ operations are required for the centroid calculations.

Significant algebraic manipulation of $\mathbf{Q}_3$ shows that all products of two components of a rotation can be expressed as a linear combination of triple product ratios. Equation (12) shows these ratios in an abbreviated form, where ξ and ζ are each elements of $\{s, x, y, z\}$. The matricies of Equation (11) should be substituted into (12) as appropriate.

The extraction of a quaternion from the matrix $\mathbf{Q}_3(R, S)$ has not yet been addressed. Obviously, not all ten components are required to calculate the sought-after quaternion, leaving many possible reconstruction techniques. The dual nature of quaternions[2] now becomes advantageous; any one component can be arbitrarily fixed as posistive, and the sign of the other components can be adjusted accordingly. The resulting rotation is not affected. Since the angle of rotation is a function of q_s (as opposed to some other quaternion component) it makes the most intuitive sense to fix q_s as positive.

Extensive experimentaiton by the authors revealed two interesting phenomena about $\mathbf{Q}_3(R, S)$. First: out of the 10 products, q_s^2, q_x^2, q_y^2, and q_z^2 are the least sensitive to noise. The proper signs of these components can be determined through the signs of the remaining six components. Second: it was found that in general, a significantly more accurate rotation may be obtained from $\mathbf{Q}_3(R, S)$ if $\langle \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle > \langle \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3 \rangle$, and vice versa. This is most likely due to numerical instability in Equation (12) occurs when $\langle \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle$ is very small. Therefore, when choosing a subset of three point pairs from four, one seeks the subset with the highest triple product.

The formulas, observations, and experimental evidence discussed in this section lead diretly to the following algorithm. This algorithm, presented informally, takes pre-transform and post-transform data sets R and S as inputs, and returns the rotation $\mathring{q}$ and translation $\mathbf{t}$ that relates them. It is assumed that $|R| = |S| = 4$.

1. Assign to $\mathbf{c}_r$ and $\mathbf{c}_s$, the centroids of R and S, respectively.

[1] Extraction methods are discussed later.

[2] A rotation through $\mathring{q}$ is equivalent to a rotation through -$\mathring{q}$.

2. Normalize R and S with respect to their centroids $\mathbf{c}_r$ and $\mathbf{c}_s$ and store the results in sets R_c and S_c.
3. Let R_3 represent an arbitrary subset of R_c of cardinality 3. Find the particular subset R_3 for which the magnitude of the triple products of its elements is maximum. Name this subset R_3^t. Store the triple product of R_3^t in t_r. Name the set of corresponding post-transform vectors S_3'.
4. Peform an analogous operation for S.
5. If $t_r > t_s$, then perform $\mathbf{Q}_3(R_3^t, S_3')$; otherwise, perform $\mathbf{Q}_3(S_3^t, R_3')$. Store the results of this calculation in $\mathbf{Q}$.
6. From $\mathbf{Q}$, use q_s^2, q_x^2, q_y^2, and q_z^2 to build a quaternion $\mathring{q}$.
7. Adjust the signs of q_x, q_y, and q_z as indicated to the signs of the six other components of $\mathbf{Q}$.
8. If $t_s > t_r$, then $\mathring{q}$ must be reversed; assign $-\mathring{q}$ to $\mathring{q}$.
9. Normalize $\mathring{q}$.
10. Using $\mathring{q}$, $\mathbf{c}_r$, and $\mathbf{c}_s$, calculate the translational component $\mathbf{t} = \mathbf{c}_s - \mathring{q}\mathbf{c}_r\mathring{q}^*$.
The above algorithm is referred to in later sections as simply EXTRACT4.

IV. ABSOLUTE ORIENTATION ALGORITHMS

As described previously, the new closed form EXTRACT4 is limited to an input set of 4 point pairs. In this section we extend EXTRACT4 to accommodate an arbitrary number of inputs. There are many ways to generalize the AO problem to n points. For instance, one may seek a rotation that:
- minimizes the sum of the squared distances between the pre-transform and post-transform points.
- maximizes the alignment of the data, treated as vectors, before and after the rotation.
- minimizes the sum of squared errors in any linear formulation that produces a solution.
- minimizes the median distance between the pre-transform and post-transform points.

Obviously, there exist many other possible constraints that may be used to calculate an accurate absolute orientation transform — it is likely to change based on the application. As disucussed in Section II, least-squares approaches are often considered, but are only optimal with Gaussian noise, and require outlier and mismatch-free input. It is the profound effect of outliers that motivated Fischler and Bolles to develop their landmark RANSAC (Random Sample Consensus) framework [19]. RANSAC instantiates not just one, but many solutions through the use of repeated subsampling. By seeking a majority of points consistent with their models, Fischler and Bolles use RANSAC to increase the robustness of a classic photogrammetric localization problem.

Our RANSAC-inspired approach, one in which a model is built through repeated subsampling, both extends EXTRACT4 to an arbitrary number of points, and increases its robustness. An outline of the algorithm is:
1. Select a subset of 4 point pairs from the input set.
2. Find $\mathring{q}$ and $\mathbf{t}$ via an aforementioned reconstruction technique, and add it to a set of potential solutions.
3. Repeat.
4. From the set of potential solutions, extract a solution.

The critical step that determines the efficacy of this approach is the last one; extracting a solution from the collection of potentials. Certainly, given a metric for evaluation, it would be possible to test the merit of *all* collected solutions, and simply choose the estimate with the "best" metric. However, testing all the estimates lacks both computational efficiency and the ability to fuse accurate solutions. A simplistic approach to solution extraction would be to use an average or median from the collected data sets. However, this would give equal weight to all estimates. Recognizing potentially accurate estimates, and appropriately weighting their contribution to the final estimate is one approach to robustness. If the effects of noise and outliers could be quantized, then it would be easier to identify desirable estimates. Because a noise measure may need to be calculated for several thousand estimates, efficiency becomes a vital constraint of this process.

A. Quantifying the Effects of Noise

Let $\mathring{q}$ and $\mathbf{t}$ represent the rotation and translation estimates obtained from an AO algorithm such as EXTRACT4 or a least-squares approach. Therefore, it follows from the definition of the AO problem, that if the input is noiseless and correctly matched, that the following identities would be perfectly satisfied [3]:

$$\sum_{i=1}^{|R|} |\mathbf{s}_i - (\mathring{q}\mathbf{r}_i\mathring{q}^* + \mathbf{r}_i)| = 0. \tag{13}$$

In the presence of noise, however, one could not expect these identities to remain valid. Therefore, examining the differences between the identity's ideal case and those values which are calculated, can give insight into the amount of noise inherent in the subsampled set. Let d (for "distance") represent a function which calculates this difference:

$$d(R,S) = \sum_{i=1}^{|R|} |\mathbf{s}_i - (\mathring{q}\mathbf{r}_i\mathring{q}^* + \mathbf{r}_i)|. \tag{14}$$

Beacause there is no limit to the size of R and S in the above equation, it would be impractical to test the AO estimates against the entire input set. It is assumed for practical purposes, that for quantifying the effect of error in a subset, that R and S are limited to the subsets themselves.

Similarly, in the case of EXTRACT4, it would be expected that the following identities that relate the various elements of $\mathbf{Q}$ are also true:

$$q_{s^2}q_{x^2} = (q_{sx})^2 \quad \ldots \quad q_{x^2}q_{z^2} = (q_{xz})^2 \tag{15}$$

Let p (for "performance") represent a function which calculates this difference:

$$\begin{aligned} p(\mathbf{Q}) = \; & |q_{s^2}q_{x^2} - (q_{sx})^2| + |q_{s^2}q_{y^2} - (q_{sy})^2| + \\ & |q_{s^2}q_{z^2} - (q_{sz})^2| + |q_{x^2}q_{y^2} - (q_{xy})^2| + \\ & |q_{y^2}q_{z^2} - (q_{yz})^2| + |q_{x^2}q_{z^2} - (q_{xz})^2|. \end{aligned} \tag{16}$$

There are various advantages and disadvantages to both p and d. First, calculating d is not limited to any one particular AO algorithm, since all AO algorithms output a rotation and translation estimate. Calculating p, however, is specific to EXTRACT4. Second, p is significantly less scale dependendant than d. In other words, p is independent to changes in the ratio of noise to point magnitude, while d is subject to scaling. As will be shown by the results of our simulations, in the general case, there is an

[3] with the exception of standard floating-point error.

$$s_{n_x} = t_x + r_x q_s^2 + r_x q_x^2 - r_x q_y^2 - r_x q_z^2 + 2r_y q_x q_y + 2r_z q_x q_z + 2r_z q_s q_y - 2r_y q_s q_z \tag{5}$$

$$s_{n_y} = t_y + r_y q_s^2 - r_y q_x^2 - r_y q_y^2 - r_y q_z^2 + 2r_x q_x q_y + 2r_y q_z q_z - 2r_z q_s q_x - 2r_x q_s q_z \tag{6}$$

$$s_{n_z} = t_z + r_z q_s^2 - r_z q_x^2 - r_z q_y^2 + r_x q_z^2 + 2r_y q_y q_z + 2r_x q_x q_z + 2r_y q_s q_x - 2r_x q_s q_y. \tag{7}$$

$$\mathbf{R}_i = \begin{bmatrix} 0 & 2r_{i_z} & -2r_{i_y} & 2r_{i_y} & 0 & 2r_{i_z} & r_{i_x} & r_{i_x} & -r_{i_x} & -r_{i_x} \\ -2r_{i_z} & 0 & 2r_{i_x} & 2r_{i_x} & 2r_{i_z} & 0 & r_{i_y} & -r_{i_y} & r_{i_y} & -r_{i_y} \\ 2r_{i_y} & -2r_{i_x} & 0 & 0 & 2r_{i_y} & 2r_{i_x} & r_{i_z} & -r_{i_z} & -r_{i_z} & -r_{i_z} \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \mathbf{R}_i & \mathbf{I} \\ \mathbf{R}_j & \mathbf{I} \\ \mathbf{R}_k & \mathbf{I} \\ \mathbf{R}_l & \mathbf{I} \\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 & 0\ 0\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{Q} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} \mathbf{s}_i \\ \mathbf{s}_j \\ \mathbf{s}_k \\ \mathbf{s}_l \\ 1 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} \mathbf{R}_i \\ \mathbf{R}_j \\ \mathbf{R}_k \\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \mathbf{s}_i \\ \mathbf{s}_j \\ \mathbf{s}_k \\ 1 \end{bmatrix} \tag{10}$$

$$P_{ss} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{xx} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{yy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{zz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad P_{sx} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$P_{sy} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad P_{sz} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad P_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad P_{yz} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad P_{xz} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{11}$$

$$q_\xi q_\zeta(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3) = \begin{cases} \left| \dfrac{\langle \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle - \langle P_{\xi\zeta}\mathbf{s}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle - \langle \mathbf{r}_1, P_{\xi\zeta}\mathbf{s}_2, \mathbf{r}_3 \rangle - \langle \mathbf{r}_1, \mathbf{r}_2, P_{\xi\zeta}\mathbf{s}_3 \rangle}{4\langle \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle} \right| & \text{if } \xi = \zeta \\ \dfrac{\langle P_{\xi\zeta}\mathbf{s}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle + \langle \mathbf{r}_1, P_{\xi\zeta}\mathbf{s}_2, \mathbf{r}_3 \rangle + \langle \mathbf{r}_1, \mathbf{r}_2, P_{\xi\zeta}\mathbf{s}_3 \rangle}{4\langle \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \rangle} & \text{otherwise} \end{cases} \tag{12}$$

Fig. 1. **Equations 5–7: Fully expanded expressions for the elements of $\mathbf{r}_n$ after a quaternion rotation about $\mathring{q}$. Equations 9–10: Systems of equations providing the bases for the new closed form. Equation 12: Solutions to the system as ratios of triple products. Triple products are denoted with angle brackets.**

Algorithm RECONSTRUCT4

Input: Two set of vecotrs n (where $n \geq 4$), R and S, respectively representing points before and after undergoing an unknown translation **t** and an unknown rotation $\mathring{q}$.
Output: Rotation $\mathring{q}$ and translation **t**.

for $i \leftarrow 1$ **to** $(n-3)$ **do**
  $(\mathring{q}, \mathbf{t}, p, d) \leftarrow$ EXTRACT4$(\{\mathbf{r}_i, \ldots, \mathbf{r}_{i+4}\}, \{\mathbf{s}_i, \ldots, \mathbf{s}_{i+4}\})$
  score $\leftarrow 1/(p^2 d^2)$
  WeightedQ$_s$*Sum* $\leftarrow q_s \cdot$ *score*
  WeightedQ$_x$*Sum* $\leftarrow q_x \cdot$ *score*
  WeightedQ$_y$*Sum* $\leftarrow q_y \cdot$ *score*
  WeightedQ$_z$*Sum* $\leftarrow q_z \cdot$ *score*
  t $\leftarrow$ **t** $\cdot$ *score*
  SumOfWeights $\leftarrow$ *SumOfWeights* + *score*
end for
$q_s \leftarrow$ *WeightedQ*$_s$*Sum* / *SumOfWeights*
$q_x \leftarrow$ *WeightedQ*$_x$*Sum* / *SumOfWeights*
$q_y \leftarrow$ *WeightedQ*$_y$*Sum* / *SumOfWeights*
$q_z \leftarrow$ *WeightedQ*$_z$*Sum* / *SumOfWeights*
t $\leftarrow$ **t**/*SumOfWeights*
$\mathring{q} \leftarrow \mathring{q}/||\mathring{q}||$
return $\{\mathring{q}, \mathbf{t}\}$

Fig. 2. **RECONSTRUCT4 pseudo-code**. Through repeated subsampling, collect a series of estimates. Using their distance and performance metrics, calculate the absolute orientation transform from weighted averages. Declarations and initializations have been omitted for brevity.

inherent correlation between quaternions with a low p value and the original rotation.

As mentioned previously, a straightfoward way of encorporating the ability to quickly estimate the amount of error in a subsample would be through weighted averages (our particular choice of weighting function will be discussed shortly). Figure 2 shows a pseudo-code implementation of RECONSTRUCT4 (also referred to as R4); a new AO algorithm based on the ideas presented thus far. It may appear that calculating p and d could be significant additions to our computational overhead, but the reader should be reminded that an optimized technique may be used, and that the relative additional expense is quite small. Regardless, the payoff is the ability to rapidly rate each estimate.

The authors tried many different weighting functions in their quest to maximize the robustness of an EXTRACT4 based AO algorithm. In our experience, out of all of the weighting functions tried, $1/(p^2 d^2)$ provided the most accurate results. Functions that used both p and d, generally provided more accurate results than one just using either p or d. This is most likely due to the redunancy introduced by encorporating both measures. This weighting function has two desirable qualities: rapidly increases as p and d become very small, and becoming almost negible for much larger values. The simulations discussed in the next section, demonstrate the efficacy of this particular weighting function.

B. Subsampled Least-Squares

The RANSAC-like framework of RECONSTRUCT4 is not limited to the EXTRACT4 algorithm. Any AO algorithm may be extended by being substituted into RECONSTRUCT4 with its own weighting function. For the sake of fair comparison, we now introduce L4, an extended form of a least-squares approach. Syntactically, L4 is nearly identical to R4 except for the following substitutions:

- Replace EXTRACT4 with Horn's least-squares algorithm. The function arguments are not changed.
- Since calculating p is depends on the use of EXTRACT4, replace the weighting function $1/(p^2 d^2)$ with $1/(d^2)$.

As discussed in Section III-B, Horn's least-squares approach requires significantly more computation than EXTRACT4. For example, if we assume (pessimistically) that 300 FLOPS are required for each call to EXTRACT4 and 1400 FLOPS for each call to an implementation of Horn's least-squares method, then if $n = 250$, then R4 requires approximately 75 KFLOPS and L4 approximately 350 KFLOPS. The least-squares approach is, however, more optimal (statistically speaking).

V. SIMULATION

To evaluate the efficacy of the new algorithm, a small-scale, but full-featured simulation system was developed. In this section, we will discuss the simulation software used to test and evaluate the performance of both the existing and new absolute orientation algorithms. The simulation software was written in an object-oriented nature with C++. All of the random number generators used in the simulation software are based on the source code provided in [18]. The free numerical library *newmat* [20] was used for many of the matrix applications needed for the implementation of Horn's algorithms. The simulation software's source code is freely available for the purposes of research and education. It may be obtained by e-mailing the authors.

A. Overall Process

The overall process of the simulation is straightforward. Generate two collections of vectors — one representing points before, and another representing the corresponding points after, a transform. In our simulation, we restrict the transform to a translation and rotation; i.e. we assume there is a unit uniform scalar correspondence, since this is true in many applications. The goals of the simulation are:

- to use an absolute orientation algorithm and the input point set data to extract the sought-after transform.
- compare and contrast a standard least-squares approach to the new approach.

The three algorithms analyzed in this paper are Horn's least-squares method, denoted as LS, L4, and R4. For a full description of LS, readers are referred to [4].

In outline form, the overall simulation process is as follows:

1. Generate a set of pre-rotation points R.
2. Generate a random rotation $\mathring{q}$.
3. Using the pre-rotation points R and the rotation $\mathring{q}$, generate a corresponding set of post-rotation points S.
4. Generate outliers and mismatches.
5. Run an absolute orientation algorithm.
6. Evaluate the performance of the algorithm with a set of metrics.

This overall structure is a natural extension of Equation (2).

Our simulation differs from previous research by including models of more realistic noise phenomena. In addition to the standard additive Gaussian noise, our simulation includes explicit steps to generate outliers and mismatches.

As an aside, in the context of our simulation, when we refer to a "set" it is not in the strict mathematical sense, but more like an array. Implicitly, a single index is used to refer to a pre-transform point and its corresponding post-transform (or a set with some other quality) point. For instance, it is assumed that the match of each pre-transform point $\mathbf{r}_i$ is $\mathbf{s}_i$. This allows the injection of invalid data by simply substituting the contents of an array element.

Some of the important parameters in our simulation are the following: m_p = mismatch probability, ω_p = outlier probability, n = number of input point pairs, N = additive noise model, and s = surface shape. The precise role of each parameter will be explained as they come into context. We now examine each major step of the simulation in greater detail.

B. Generate Pre-Transform Points

First, we create the pre-transform data set from one of four different shapes/surfaces — a sphere, a constrained subsection of the sphere (an octant specifically), a section of PVC pipe, and a wooden block. The sphere and octant data points are completely synthetic. The pipe and block data points are from the Michigan State University Range Image Database [21]. Points from the MSU database were collected using a Technical Arts 100X Range Scanner. The particular models used in the simulations are shown in Figure 4. The approximate ranges for the pipe data points are ($[-4.3, 4.3]$, $[-0.2, 7.2]$, $[-1.6, 2.5]$) and for the block data are ($[-3.2, 3.0]$, $[-0.9, 7.9]$, $[-3.3, 3.0]ve$). Both the sphere and octant have radius 5.0.

The simulation generates a collection of n random vectors R that could represent points on the surface of shape s with a center at the pre-rotational translational component $\mathbf{r}_t$ — a random vector of maximum magnitude 10.0. Noise is simulated with one of three simple additive models:

- **Gaussian**: a standard Gaussian model with fixed variance σ^2 in each of the axial directions.
- **Fractional Gaussian**: a standard Gaussian model with variance $\sigma^2 = km$ where m is the magnitude of the vector to which the noise shall be added and k is a fixed constant.
- **Stereo**: a simple model based on the projective geometry usually associated with stereo vision systems. The simulation model assumes two identical cameras with parallel focal axes, baseline b, focal length(s) f, and pixel pitch(es) δ. Then, added to each point (x, y, z), is $(b(x_l + \Delta_{x_l})/d,\ b(y_l + \Delta_y)/d,\ bf/d)$ where disparity $d = (x_l + \Delta_{x_l}) - (x_r + \Delta_{x_r})$, $x_l = fx/z$, $x_r = f(x - b)/z$, $y_l = fy/z$, and Δ_{x_l}, Δ_{x_r}, and Δ_y are random numbers from a uniform distribution spanning $-\delta/2$ to $\delta/2$. In this paper, the stereo parameters were selected so that the magnitude of the added noise was comperable to the Gaussian and fractional Gaussian models: $b = 0.5$ m, $f = 0.028$ m and $\delta = 8.7 \times 10^{-5}$ m.

Random vectors generated from noise model N are added to the original points to create the noisy pre-rotation data set R_n. In Figure 5, the Gaussian noise model with standard deviation σ is denoted at G/σ and the fractional noise model with constant k is denoted as F/k. The stereo model is denoted with an S.

C. Generate Post-Transform Points

First, we generate a random vector $\mathbf{s}_t$ to be used for the post-transformation translational component. Then, using the pre-rotation vectors R and a random rotation $\mathring{q}$, the simulation generates the set of noiseless counterparts S, adding in $\mathbf{s}_t$ (also a random vector of maximum magnitude 10.0) to each point. Next, an additional noise vector generated from model N is added to each element in S to create the set of noisy post-rotation vectors S_n.

R	=	set of noiseless pre-transform points
R_n	=	set of noisy pre-transform points
R_ω	=	set of noisy pre-transform points, including outliers
S	=	set of noiseless post-transform points
S_n	=	set of noisy post-transform points
S_ω	=	set of noisy post-transform points, including outliers
S_m	=	set of partially matched, noisy, post-transform points, including outliers

Fig. 3. Sets of data points generated by the simulation.

It is at this point that the simulation significantly departs from previous research by modeling two types of real-life error phenomena. First, it is possible that our correspondences themselves are imperfect, i.e. they include a number of *mismatches*. Second, our input points may include *outliers* — which we categorize as extremely noisy points. Sources of mismatches and outliers include bad stereo correspondences, numeric instabilities, and equipment or human error.

D. Simulating Outliers

To simulate the presence of outliers, we start by copying the noisy sets of R_n and S_n into new outlier sets R_ω and S_ω. Then, for each point $\mathbf{p}_i$ in R_ω and S_ω, we choose a random floating-point number k from a uniform distribution spanning 0.0 to 1.0. If k is less than the input parameter ω_p, then we generate a point of random direction and random magnitude (of at maximum 25.0) and replace $\mathbf{p}_i$ with the outlier. The simulation makes no guarantee that the magnitude of an outlier lies *outside* a specific range.

E. Simulating Mismatches

The partially matched set, S_m, is generated in a similar manner. Due to the symmetry of mismatches, there is no need to generate a pre-rotation mismatch set R_m. The mismatch set, S_m, starts as a duplicate of S_ω. For each point $\mathbf{s}_i$ in S_n, we choose a random floating-point number k', from a uniform distribution spanning 0.0 to 1.0. If k' is less than the input parameter m_p, then we choose an element of S_n (uniformly random), and replace $\mathbf{s}_i$ with its value. No special consideration is taken if $\mathbf{s}_i$ is already an outlier; i.e. an outlier could be replaced by a mismatch.

In summary, after this stage, the point sets described in Figure 3 have been generated. These point sets may be used directly as inputs to an AO algorithm. The inputs used for algorithms L4 and R4 are R_ω and S_m. Note that in the case of *least-squares*, sets R_ω and S_ω are used for the calculation of the translational component, but R_ω and S_m are used for the calculation of the rotational component. Because LS uses raw centroids for calculating $\mathbf{t}$, point correspondence information is not required during this stage.

VI. METRICS

A standard part of evaluating any algorithm is determining the metrics to be used when quantifying its performance. In this section, we will select our metrics, discuss how they should be used in our simulation, and the significance of their application.

A. Absolute Quaternion Distance: AQD

To the best of the authors' knowledge, a metric that has not yet been used to evaluate absolute orientation is a measure we will refer to as the **absolute quaternion distance**, or AQD. Essentially, the AQD is a measure of proximity between the quaternion used to generate the post rotation points and the quaternion estimated from an algorithm.[4] The AQD gives no information about the validity of an algorithm's translation estimate. Given the ground-truth rotation $\mathring{q}$ and an AO algorithm's estimate $\mathring{q}'$, the AQD is trivial to compute: $\text{AQD}(\mathring{q}, \mathring{q}') = \|\mathring{q} - \mathring{q}'\|$.

Since the AQD does not directly use point information in its calculation, the AQD metric is not directly affected by outliers and mismatches. With a distance-based metric, a single outlier can have a significant affect on the metric. Consequently, observing the distance metric alone may give the impression that a particular estimate is less accurate than it actually is. The coupling between metric and system robustness is complex — outliers and mismatches always affect the AO algorithm *itself*. A robust metric, however, can give specific insight into an algorithm's performance.

B. Absolute Translation Distance: ATD

Parallel in both construction and concept to the AQD, the **absolute translation distance**, or ATD is a measure of proximity between the translation used in the point set generation and the translation estimated from the algorithm. Naturally, the ATD gives no information about the validity of an algorithm's rotation estimate. Given the grouth-truth translation $\mathbf{t}$ and the AO algorithm's estimate $\mathbf{t}'$, $\text{ATD}(\mathbf{t}, \mathbf{t}') = \|\mathbf{t} - \mathbf{t}'\|$ can be used to compute the ATD.

C. Average Distance Metrics: ADMs

Traditionally, absolute orientation algorithms have sought the transform that minimizes the average pointwise Euclidean distance between the elements of R' and S, where R' represents the points in R after undergoing an AO algorithm's transform. At first glance, this informal definition may appear to provide enough specificity for building our metric. However, with access to full ground truth, and the fact that this simulation models multiple error phenomena, we must consider two important questions: one, what specific data sets (and combinations thereof) should be used to measure these distances; and two, how do these choices affect the qualities that the metrics evaluate?

Therefore, first consider a fully parameterized form of the ADM — one that is a function of: a pre-transform data set R, a post-transform data set S, a translation estimate $\mathbf{t}$, and a rotation estimate $\mathring{q}$. Then,

$$\text{ADM}(R, S, \mathbf{t}, \mathring{q}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{t} - \mathring{q}\mathring{\mathbf{r}}_i\mathring{q}^*\|. \qquad (17)$$

[4]Obviously, the AQD is only appropriate in the presence of ground truth.

This more general form of the ADM, provides a uniform way to describe different variations of the "traditional" distance metric. Selecting the proper ADM parameters is dependent on the algorithm being evaluated.

D. Experimental ADM: ADM–E

In typical experimentation, ground truth is not available. Therefore, the **experimental ADM** (or ADM–E), which is modeled after such situations, is defined as ADM–E = $\text{ADM}(R_\omega, S_m, \mathbf{t}, \mathring{q})$. In this case, choosing the proper data sets for this metric is trivial — in experiments without ground truth there is no choice but to use the noisy, partially matched, outlier-laden data sets (R_ω and S_m).

Notice that even in the presence of a single outlier, the ADM–E may become significantly larger, and give an impression that an estimate is less accurate than it actually is. Mismatches can have similar effects. This phenomenon is clearly illustrated later in Section VII.

E. Cleaned ADM: ADM–C

The goal of the **cleaned ADM**, or ADM–C, is to show how an algorithm performed with respect to the valid input data, in spite of the presence of outliers and mismatches. The ADM–C is generated in the same method as the ADM–E, except that points known to be either mismatches or outliers are explicitly ignored; i.e. they are included neither when determining distances nor during translational normalization.

Let E represent the expression

$$E = (\mathbf{r}_{n_i} = \mathbf{r}_{\omega_i}) \wedge (\mathbf{s}_{n_i} = \mathbf{s}_{\omega_i}) \wedge (\mathbf{s}_{n_i} = \mathbf{s}_{m_i}). \qquad (18)$$

Then, let R_C (C for *cleaned*) represent $R_C = \{\mathbf{r} \in R_n : E\}$ and $S_C = \{\mathbf{s} \in S_n : E\}$. Essentially, R_C and S_C are the sets containing only the non-outlier (but noisy) point pairs that "survived" the simulation's point generation process. Therefore, ADM–C, defined as ADM–C = $\text{ADM}(R_C, S_C, \mathbf{t}, \mathring{q})$ is a reflection of an estimate's performance with respect to valid data, although the estimates themselves were obtained with data that included outliers and mismatches.

In summary, we have presented four different metrics to provide a comprehensive framework by which absolute orientation algorithms may be analyzed. Two of the metrics, ATD and AQD, evaluate an absolute orientation algorithm's estimate with respect to the original, noise-free, synthetic data. ADM–E evaluates the estimate without ground truth, and ADM–C shows how the algorithm performs with respect to the valid, but noisy, input data.

F. On the Merits of Metrics

As mentioned previously, the majority of previous absolute orientation research attempts to find the transform that minimizes ADM–E. If our input data is free of outliers and mismatches, then, for non-degenerate cases, any method based on this criterion will achieve the same (minimal) ADM–E and ADM–C values. Therefore, in this "ideal" situation, it is theoretically impossible to outperform a least-squares approach on these ADM metrics.

The presence of mismatches and outliers significantly changes the very nature of the absolute orientation problem. For instance, Horn's least-squares algorithm is founded on the

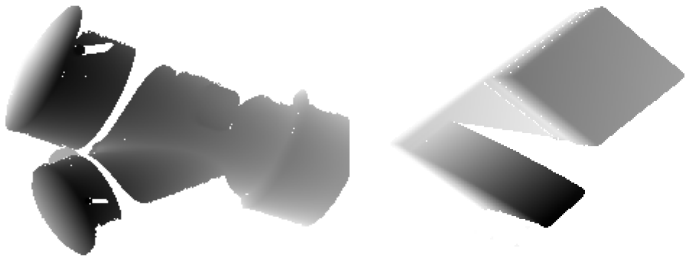

Fig. 4. Illustration of MSU range image of PVC pipe section and block used in simulation software. Darker pixels represent closer regions.

assumption that all of the input points are correctly matched, complete, and free of outliers. These very assumptions, however, allow least-squares approaches to achieve high-accuracy estimates, even in the face of Gaussian noise with large variance. When the original assumptions made by least-squares approaches are violated, one cannot expect these algorithms to consistently produce optimal results. Therefore, the fact that neither L4 and R4 assume mismatch and outlier free-data is both a strength and a weakness of the algorithms. Neither algorithm requires perfectly matched input data with purely Gaussian noise to perform well. However, it simply cannot match the performance of a full least-squares approaches when the input data is outlier and mismatch free — these assumptions are not an inherent part of the algorithm's model. In statistical terms, L4 and R4 trade statistical efficiency for robustness.

VII. Evaluation

In this section, we examine some of the trends of the algorithms' behavior in the simulations. This particular analysis is concerned primarily with how the algorithms react to changes in noise models, the number of input points, mismatch probability, and outlier probability.

The term *configuration* will be used to refer to one particular set of simulation parameter values. A *simulation run* refers to 250 independent runs of the simulation with a fixed configuration, but changing random seed. The full output of a simulation run consists of 45 values — the mean, median, and standard devation (across each iteration) for each metric-algorithm pair.

The data collection for this trend analysis proceeded as follows. First, for each simulation parameter of interest, a set of values was selected. In this case they were:

$N \in \{G/0.01, G/0.05, G/0.10, F/0.01, F/0.025, F/0.05, S\}$
$s \in \{\text{sphere}, \text{octant}, \text{block}, \text{pipe}\}$
$m \in \{0.01, 0.05, 0.10, 0.20\}$
$\omega \in \{0.01, 0.05, 0.10, 0.20\}$
$n \in \{20, 50, 100, 250\}$

All other simulation parameters were fixed as described in Section V. For each of the $1,792$ combinations of parameters, simulation runs were executed, and their output recorded. To summarize these thousands of experiments, summary graphs were produced. For each data point in the graphs of Figure 5, the named parameter is fixed, and the data point is obtained by averaging the data for a paricular algorithm-metric pair. In addition to the mean of the means we also show the standard deviation of the means, which shows how the results vary across the many different simulation parameters.

More formally, each data point shows the mean and one standard deviation of the set $S(A, M, p, v) =$

$$\bigcup_{\{x \in (m \times \omega \times n \times N \times s): p = v\}} \frac{1}{i} \sum_{j=1}^{i} M(A, S(m, \omega, n, N, s)) \quad (19)$$

where $p \in \{ m, \omega, n, N, S \}$ represents the set of parameters, v represents a particular parameter value, i represents the number of iterations (i.e. 250), $S(m, \omega, n, N, S)$ represents a single data set generated from the simulation with parameters (m, ω, n, N, S), $A \in \{LS, L4, R4\}$ represents an element of the set of algorithms mapping a data set to an AO, and $M \in \{$ ADM–E, ADM–C, AQD, ATD $\}$ represents an element of the set of metrics.

For instance, take the upper-left graph of Figure 5. The graph is divided into three regions — LS (plotted with squares), L4 (plotted with circles), and R4 (plotted with triangles) — one for each algorithm. The leftmost point indicates that for all experiments where the noise model was $G/0.01$ the average mean value of the ADM–E metric was approximately 3.1. The standard devation of those means was approximately 2.9. To the graph's right are the corresponding ADM–Cs.

Graphs for the other two metrics — AQD and ATD — are not shown. Except for a difference in scale and minor changes in their standard devations, upon visual inspection, the graphs for the other metrics are virtually indistinguishable from the ADM–C graphs. Most of the graphs indicate, that in the majority of cases, R4 slightly outperforms L4. Your mileage may vary.

In all cases, there is a significant difference between the *ADM–E* and *ADM–C* graphs, indicating that just observing trends in *ADM–E* does not give a complete description of an AO algorithm's behavior. The new metrics all require ground truth, but, unlike most previous research, the measure of the AO algorithm quality it not soley based on a limited metric such as the ADM–E.

Specifically, the *Points/ADM–E*, *Mismatches/ADM–E*, and *Outliers/ADM–E* graphs, do not clearly indicate that any one algorithm is significantly more robust than any other. This is especially true in the *Outliers/ADM–E* graph, which indicates that L4 performs no better than LS or R4. However, when measured against ground truth, L4 and R4 clearly show their increased robustness.

Likewise, the *Points/ADM–E* graph also does not strongly indicate that increasing the number of points has any significant effect on any algorithm's performance. The *Points/ADM–C* graph, however, does reinforce the intuitive notion that increasing the number of points gives more accurate results. Because of the weakness (i.e. lack of robustness) of the ADM–E measure, this trend tends to be masked by the amplification of noise in the metric itself.

The standard deviations of the LS algorithm in the *Outliers/ADM–E* graph are significantly smaller than in any other graph. This indicates that the standard devation of LS in the other graphs is primarily due to the presense of outliers.

There are two categories of ways in which one particular algorithm outperfomed another. First, there were many configurations for which one algorithm clearly was a better choice than another. For example:

- L4 performed particularly well in the configuration $s = \text{pipe}$, $n = 250$, $N = G/0.01$, $m = 0.2$, $\omega = 0.1$.

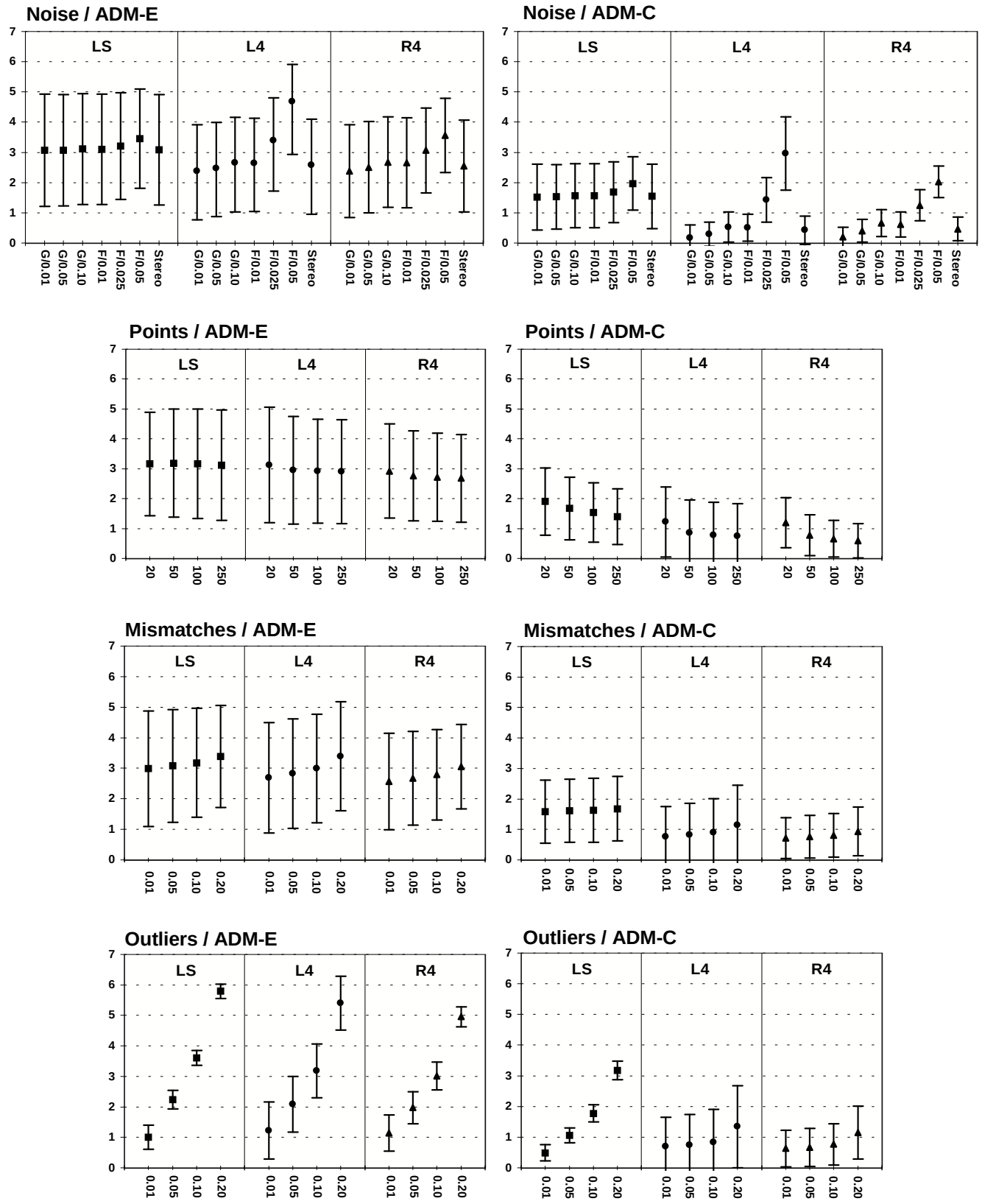

Fig. 5. Graphical summaries of the data collected in simulation. See the text for a full explanation.

• LS performed particularly well in the configuration $s =$ octant, $n = 50$, $N = F/0.05$, $m = 0.2$, $\omega = 0.1$.
• R4 performed particularly well in the configuration $s =$ pipe, $n = 50$, $N = S$, $m = 0.1$, $\omega = 0.1$.
• In an analysis of the data given a translational component of **0**, L4 and R4 were usually quite comparable, but L4 did perform slightly better than R4.

Second, within each simulation run, there were natural fluxuations in which algorithm had the "best" estimation according to a certain metric.

VIII. Conclusion

In this paper, a new, closed-form solution to the absolute orientation problem was introduced. This new closed-form was then extended into a new AO algorithm, R4 with specific provisions for handling outliers and mismatches. Similar extentions were made to Horn's least-squares approach, so that the new provisions also could be also used to enhance previous research. A large set of simulations were performed and evaluated so that the new algorithms could be analyzed.

In summary, the evaluation presented here suggests the following:

• **ADM–E is a weak metric.** Comparing two AO algorithms based on traditional measures may not give insight into the algorithm's robustness. This is primarily due to lack of robustness in the ADM–E itself.
• **Outliers may be more influential than mismatches.** In the simulations here, outliers held the most influence in determining the effectiveness of an algorithm. As indicated by Figure 5, simulation sets with fixed outlier probabilities showed results with significantly less variance than those in which the outlier probability varied.
• **Proper reconciliation is important.** The LS and L4 algorithms are based on statistical optimality. However, with the proper weighting, R4 can achieve not only comparable, but in many cases, more accurate results, even though no *a priori* noise model has been assumed. In analysis not shown here, we have considered many weighting functions (including uniform weighting) and it has a significant impact on the quality of the result.
• **Robustness requires a bottom-up, not a top down approach.** As demonstrated by Fishler and Bolles [19], removing the "worst" residual does not always correspond to removing outliers. Like RANSAC, the algorithms presented in this paper use an approach based on subsampling and reconsillation.

The analysis showed that with a significantly reduced number of computations, the algorithm R4 could achieve better results than L4 despite a significant reduction in the number of computations performed.

References

[1] E. H. Thompson, "An exact linear solution of the problem of absolute orientation," *Photogrammetria*, vol. 14, no. 8, pp. 20–28, August 1959.
[2] G. H. Schut, "On exact linear equations for the computation of the rotational elements of absolute orientation," *Photogrammetria*, vol. 17, no. 1, pp. 34–37, 1960.
[3] F. Sansò, "An exact solution of the roto-translation problem," *Photogrammetria*, vol. 29, pp. 203–216, 1973.
[4] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, April 1987.
[5] O. D. Faugeras and M. Hebert, "A 3d recognition and positioning algorithm using geometrical matching between primite surfaces," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Karlshure, West Germany, August 1983, pp. 996–1002.
[6] B. Sabata and J. K. Aggarwal, "Surface correspondence and motion computation for a pair of range images," *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 232–250, March 1996.
[7] G. Calafiore and B. Bona, "Constrained optimal fitting of three-dimensional vector patterns," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 838–844, October 1998.
[8] X. Pennec and J. Thirion, "A framework for uncertainty and validation of 3-d registration methods based on point and frames," *International Journal of Computer Vision*, vol. 25, no. 3, pp. 203–229, 1997.
[9] B. Kamgar-Parsi and B. Kamgar-Parsi, "Matching sets of 3d line segments with application to polygonal arc matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1090–1099, October 1997.
[10] B. K. P. Horn, H.M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, pp. 1127–1638, July 1988.
[11] M. Walker and L. Shao, "Estimating 3-d location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, November 1991.
[12] K. Kanatani, "Analysis of 3-d rotation fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 543–549, May 1994.
[13] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, September 1987.
[14] J. Lasenby, W. J. Fitzgerald, A. N. Lasenby, and C. J. L. Doran, "New geometric methods for computer vision: An application to structure and motion estimation," *International Journal of Computer Vision*, vol. 26, no. 3, pp. 191–213, 1998.
[15] Z. Lin, T. S. Huang, S. D. Blostein, H. Lee, and E. A. Margerum, "Motion estimation from 3d point sets with and without correspondences," in *Proceedings of the 1986 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1986, pp. 194–201.
[16] G. Barequet and M. Sharir, "Partial surface and volume matching in three dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 929–948, September 1997.
[17] M. D. Wheeler and K. Ikeuchi, "Iterative estimation of rotation and translation using the quaternion," Tech. Rep. CMU–CS–95–215, Carnegie-Mellon University, December 1995.
[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipies in C: The Art of Scientific Computing*, Cambridge University Press, 2nd edition, 1992.
[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.
[20] R. Davies, "newmat09," `http://nz.com/webnz/robert/nzc_nm09.html`, Software library.
[21] Michigan State University, "Michican state university range image database," Laser range-finder range database.