The Determinant Ratio Matrix Approach to Solving 3D Matching and 2D Orthographic Projection Alignment Tasks

Andrew J. Hanson
Indiana University
Bloomington, IN
hansona@iu.edu

Sonya M. Hanson
The Flatiron Institute
New York City, NY
shanson@flatironinstitute.org

***Abstract*—Pose estimation is a general problem in computer vision with applications ranging from self-driving cars to biomedical imaging. The relative orientation of a 3D reference object can be determined from a 3D rotated version of that object, or, as is often the case, from a projection of the rotated object to a 2D planar image. This projection can be a perspective projection, referred to as the PnP problem, or an orthographic projection, which we refer to as the OnP problem; in this work, we restrict our attention to the OnP problem and the full 3D pose estimation task, which we refer to for convenience as the EnP problem. Here we present a novel closed-form solution to both the EnP and OnP problems, derived by applying either the quaternion adjugate or the rotation matrix element framework to the corresponding least squares problem, and dubbed the determinant ratio matrix (DRaM) approach. While our solutions are strictly valid only for error-free data, we can adjust noise-distorted improper rotation matrix candidates to their nearest exact rotation matrices with a straightforward rotation correction scheme. While the SVD and optimal quaternion eigensystem methods for the EnP task, often known as the RMSD (root-mean-square-deviation) problem, determine the noisy 3D-3D alignment even more accurately than our method, the noisy 3D-2D orthographic (OnP) task has no known comparable closed form, and is solved quickly and to a high degree of accuracy by our scheme. We note while previous similar work has been presented in the literature exploiting both the QR decomposition and the Moore-Penrose pseudoinverse transformations, here we place these methods in a larger context that has not previously been fully recognized in the absence of the corresponding DRaM solution. Thus we term this class of solutions as the DRaM family, and conduct comparisons to the behavior of the RMSD family of solutions for the EnP and OnP rotation estimation problems. Overall, this work presents both a new solution to the 3D and 2D orthographic pose estimation problems and provides valuable insight into these classes of problems. With hindsight, we can even show that our DRaM solutions to the exact EnP and OnP problems possess derivations that could have been discovered in the time of Gauss, and generalize to all analogous N-dimensional Euclidean pose estimation problems.**

***Index Terms*—RMSD, OnP, Alignment, Pose Estimation, Orientation Matching, Attitude Extraction, Orthographic Projection.**

I. Introduction

Finding the best rotation aligning a reference object to an observed, rotated instance of the same object is a problem relevant to areas ranging from machine vision to astronautics to biochemistry. Analytical solutions determining the rotation that solves the least-squares problem for exact or noisy 3D matched point-cloud objects have existed for decades (see, e.g., Davenport (1968); Schönemann (1966); Green (1952)); however, no comparable solutions have been presented for the rotation aligning a 3D reference model to a corresponding 2D orthographic or perspective projected image. Here we present closed-form determinant ratio matrices (DRaM's), expressed transparently in terms of data cross-covariances, that solve the error-free 3D-3D alignment (Euclidean distance loss or "EnP") and 3D-2D orthographic projection alignment (Orthographic projection loss or "OnP") problems. We reveal in addition the unexpected correspondence of at least two other methods applicable to both EnP and OnP in the literature that are numerically identical to the DRaM's. The work we present here is a significant elaboration and contextualization of the authors' preliminary developments of this material, building on Hanson and Hanson (2022); Hanson (2024); in addition, an initial application-driven exploration, exploiting only earlier theoretical aspects, was published in Lin et al. (2025). The methods employed here were applied to solve the more challenging perspective n-point (PnP) problem in Hanson (2024), but a DRaM form of that solution evaded discovery; an explicit DRaM form for the error-free PnP task has recently been obtained, and the details will be elaborated elsewhere (Hanson, 2026).

Our results include several components with subtle relationships whose context we outline now to assist the reader. Our first step, well-known, for example, from (Horn, 1987), is to pose the EnP and OnP problems (for the 3D-3D and 3D-2D data, respectively) as least squares loss expressions to be minimized over the nine rotation matrix elements R_{ij} or the quaternion 4-vectors q_i, alternatively expressable in terms of the ten quadratic *adjugate quaternions* $q_{ij} = q_i q_j$. The latter convert the challenging least squares quaternion loss optimizations to an approachable quadratic form. The rotation matrix and adjugate quaternion matrix loss-minimizing expressions can each be solved, for error-free data, to give the DRaM solution for the pose. We remark that we came late to the realization of this equivalence, which does not appear in Hanson and Hanson (2022); Hanson (2024). The first remarkable side-result given at the end of Section IV is that the resulting DRaM rotation matrix solutions can be generalized to *any Euclidean dimension*, essentially robbing the 3D quaternion approach of any unique properties it might

have for EnP and OnP. What is therefore important is that Section IV establishes the existence, via multiple derivations, of a universal "DRaM class" of solutions to the EnP and OnP alignment problems in any dimension for error-free data.

The second important revelation that we present is that these solutions are in fact not all that new, but provide elegant and unexpected connections to entire fields of prior methods. Specifically, in Section VII, we show that the DRaM class has exactly the same numerical properties (and presumably hidden algebraic properties) as the QR-based matrices in the work of Steger (2018) and the distinct pseudoinverse reference data mapping methods of, e.g., Dementhon and Davis (1995) or Hajder (2017).

The introduction of measurement error in the data deforms the DRaM family of candidate rotation matrices so they are no longer pure rotations. Only the 3D EnP problem has a known algebraic solution for noisy data (the quaternion eigenvalue method and equivalents), but we propose exploiting *rotation-correction* methods that yield *the closest pure rotation matrix* to the deformed DRaM-class of solutions (see, e.g., Björck and Bowie (1971); Higham (1986); Bar-Itzhack (2000)). After this corrective procedure, the DRaM-related methods are only slightly less accurate than the gold standard `ArgMin` (e.g., Levenberg-Marquardt) numerical search methods and can be somewhat faster. The complete spectrum of properties for all known methods for attacking the EnP and OnP alignment tasks for perfect and noisy data are summarized in Table I; we know of no equivalent analysis in the literature. Appendix A outlines the original quaternion-based derivation of the DRaM formula that led over a period of years to the broader analysis we present here, while Appendix B supplies a family of applicable software algorithms in Mathematica, most of which are algebraic formulas easily (if less transparently) realized in other languages.

II. Previous Work

Orientation matching has been the subject of a wide spectrum of publications over the years. This task was not a major topic of investigation until fairly recently in human history, when technological requirements like satellite tracking along with automated data processing and analysis brought the pose estimation problem and methods of solving it into focus. The classic problem of aligning a model's orientation, represented as a set of 3D points in space, with a measured set of corresponding points in a different orientation is what we label as the EnP task; this alignment problem arose in the early days of astronautics, where it is known as "Wahba's Problem" (Wahba, 1965), also referred to as the Orthogonal Procrustes or RMSD (root mean square deviation) problem. As the digital age matured, applications in disciplines such as machine vision and robotics, as well as life sciences areas such as proteomics, molecular geometry, and electron microscopy, developed requirements in the pose matching problem domain. Three principal approaches were found to solve the orientation discovery or pose determination tasks. Several instances of a basic linear algebra method (the unitary-symmetric "polar decomposition" theorem) came to light independently, including Green (1952); Thompson (1958), followed later by Horn, Hilden, and Negahdaripour (1988); we will refer to it as "HHN" corresponding to the latter frequently-cited paper. Other studies employing this method include Giardina et al. (1975); Bar-Itzhack (1975); Sarabandi et al. (2020) and note that it can be proven to solve the Frobenius norm minimization, and can be discovered from an SVD formulation. Soon after Wahba defined the problem, Davenport (1968) in the aeronautics community proposed a quaternion eigensystem solution (refined in another form by Kabsch (1976, 1978), and as a higher performance quaternion system in Shuster and Oh (1981)).

The quaternion eigensystem approach was rediscovered repeatedly in the machine vision and photogrammetry literature (e.g., Sansò, 1973; Faugeras and Hebert, 1983, 1986; Horn, 1987), and in the bioinformatics literature (e.g., Kabsch, 1976, 1978; Diamond, 1988; Kearsley, 1989; Kneller, 1991; Theobald, 2005; Liu et al., 2010). There are two distinct variants, one a rigorous least-squares approach, usually credited to Faugeras and Hebert (1983), that we will label "QMIN", and a more common version based on maximizing the least-squares cross-term, "QMAX." In parallel, the singular-value-decomposition ("SVD") approach of Thompson (1958) reappeared, for example, in Schut (1960); Schönemann (1966); Arun et al. (1987); Markley (1988), where the standard mathematical source seems to be that given in Golub and van Loan (1983), Section 12.4, which cites Schönemann (1966), though a parallel treatment appears later in the same journal (Cliff, 1966).

The possible puzzle of the alternative quaternion eigensystem and SVD approaches was resolved, e.g., by Coutsias et al. (2004), who explicitly proved their equivalence. An alternative dynamical spring-based system method has been suggested by Yang et al. (2021). We observe that the quaternion eigensystem method is applicable only in 3D or 4D (see, e.g. Hanson, 2020), while the SVD and HHN methods can be used for data in any Euclidean dimension. All of these EnP pose discovery methods, QMIN, QMAX, SVD, and HHN, are equivalent and form a single class that will refer to as the "RMSD class", in deference to traditional literature; all recover the same "gold standard" numerical least squares optimal rotation (`ArgMin`) when applied to *either* exact or noisy data. Further discussion of these solutions to the ENP pose discovery problem can be found, e.g., in Flower (1999); Förstner (1999); Hanson (2020, 2024).

The task of pose determination when the only target data are an *orthographic projection* of the rotated reference data, the OnP problem, is more challenging than one might expect. While the exact-data case can be solved immediately by several methods, and the optimal rotation correction problem can be solved by rectangular SVD or the equivalent Bar-Itzhack quaternion eigenvalue method, solving the noisy-data OnP case exactly appears quite challenging. The basic requirements of the OnP problem appear in the context of various approaches to the PnP (Perspective n-Point) problem, using terminology such as scaled orthography or weak perspective. Various treatments have appeared in the literature, with a numerical OnP solution as an 11^{th}-degree polynomial given by

Hajder (2017, 2019), and an extensive evaluation of available techniques presented by Steger (2018), who introduces the QR decomposition method. The alternative Moore-Penrose pseudoinverse approach to the exact OnP problem is exploited in the POSIT work of Dementhon and Davis (1995) and is advocated in Hajder (2019). While the pseudoinverse system appears closely related to the QR method, it is distinct in its implementation. Nevertheless, to the best of our knowledge, while closed-form noisy-data solutions exist for the EnP problem, no analogous closed-form solutions have appeared in the literature for the noisy-data OnP problem.

Two other closely related problems are significant for our discussion. One is the *extraction of a quaternion from a rotation matrix*, which is addressed in Bar-Itzhack (2000) using an adaptation of the quaternion eigensystem EnP method, and the other is the extraction of the *nearest possible pure rotation* from a measurement-contaminated deformed rotation matrix candidate, which is given, for example, using an SVD method in Björck and Bowie (1971); Higham (1986) as well with an alternative quaternion approach in Bar-Itzhack (2000). The classic brute force approach to the quaternion-from-rotation task, checking carefully for the variety of possible singularities, is that of Shepperd (1978); the final sections of Sarabandi et al. (2018) include a succinct rephrasing of the principles of Shepperd's singularity-removal method, which is also interpretable in terms of the quaternion adjugate-matrix method (Lin et al. (2023); Hanson (2024)). Our preferred approach is that of Bar-Itzhack (2000), based on a reworking of the quaternion pose-extraction method (e.g., Horn (1987)), with the added observation that it also works with only a 2×3 projection matrix, which in fact is also true for the SVD approach. A contemporary contribution to the same problem using the polar-decomposition and pointing out its relation to the SVD is given by Sarabandi et al. (2020). Four-dimensional quaternion-pair extraction from an $\mathbf{SO}(4)$ rotation was solved by Hanson (2020), and also studied by Sarabandi and Thomas (2022).

III. Fundamental Background and Methods

Our main narrative takes advantage of concepts that may be unfamiliar to some readers. These elements are essential to establishing the terminology used in the text, and thus we present them immediately in this section rather than relegating them to appendices or supplementary material. The topics appearing next include quaternion methods, and particularly the linear algebra adjugate application and how quaternion adjugate variables emerge, along with the basic family of ways to define the EnP (3D:3D matching) and OnP (3D:2D-orthographic matching) pose discovery problems via loss minimization.

A. Rotations in Terms of Quaternions and Quaternion Adjugate Variables

A *quaternion* is a point on a 3-sphere representable as a unit 4-vector $q = [q_0, q_1, q_2, q_3]$ with $q \cdot q = 1$. Quaternions obey a particular algebra, whose details we can ignore here (see, e.g. Hanson, 2006, 2024, for more details), that allows us to encode any proper 3D rotation using the following quadratic quaternion form:

$$R(q) = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix}. \tag{1}$$

With the single constraint $q \cdot q = 1$, $R(q)$ can easily be shown to be orthonormal with unit determinant; furthermore, it is clear that any rotation optimization task can be formulated exclusively using the quaternion, sidestepping the six orthonormality constraints on the nine rotation matrix elements by merging them into the single quaternion constraint.

The *quaternion adjugate variables* are obtained from the 10 unique elements of the symmetric quadratic quaternion matrix with elements $q_{ij} = q_i q_j$:

$$\text{Adj}(q_{ij}) = \begin{bmatrix} q_{00} & q_{01} & q_{02} & q_{03} \\ q_{01} & q_{11} & q_{12} & q_{13} \\ q_{02} & q_{12} & q_{22} & q_{23} \\ q_{03} & q_{13} & q_{23} & q_{33} \end{bmatrix}. \tag{2}$$

As argued in Hanson and Hanson (2022); Lin et al. (2023); Hanson (2024), the form in Eq. (2) follows directly from solving the quaternion eigensystem of the EnP (also "Procrustes" or "RMSD") cloud-to-cloud matching problem and expressing the charts of the manifold covering the quaternion solution using the adjugate of the corresponding characteristic equation. Each row or column of the matrix $\text{Adj}(q_{ij})$ is equal to a full quaternion vector q multiplied by one of the other four quaternions. We emphasize the fact that there are 14 possible combinations of zeros of q_i rendering one or more columns of Eq. (2) unnormalizable, so employing the full Eq. (2) is essential for expressing quaternion output in machine learning tasks to avoid the 14 singular cases occurring inevitably if only a single quaternion 4-vector is used.

We see from Eq. (1) that an alternative expression for any rotation matrix is therefore

$$R(q_{ij}) = \begin{bmatrix} q_{00} + q_{11} - q_{22} - q_{33} & 2q_{12} - 2q_{03} & 2q_{13} + 2q_{02} \\ 2q_{12} + 2q_{03} & q_{00} - q_{11} + q_{22} - q_{33} & 2q_{23} - 2q_{01} \\ 2q_{13} - 2q_{02} & 2q_{23} + 2q_{01} & q_{00} - q_{11} - q_{22} + q_{33} \end{bmatrix}. \tag{3}$$

The standard unit-length constraint $q \cdot q = 1$ imposes a more complicated set of *seven* constraints on the adjugate variables, namely

$$\left.\begin{array}{c} q_{00} + q_{11} + q_{22} + q_{33} = 1 \\ q_{00}\, q_{11} = {q_{01}}^2 \quad q_{00}\, q_{22} = {q_{02}}^2 \quad q_{00}\, q_{33} = {q_{03}}^2 \\ q_{22}\, q_{33} = {q_{23}}^2 \quad q_{11}\, q_{33} = {q_{13}}^2 \quad q_{11}\, q_{22} = {q_{12}}^2 \end{array}\right\}. \tag{4}$$

In a moment, we will show how to exploit the quaternion adjugate variables to reduce the degree of quaternion-based pose-discovery loss functions by a factor of two, thus enabling us to obtain previously unknown explicit algebraic expressions for pose-rotation solutions directly from the least-squares loss functions. This is a significant step beyond the application of the quaternion adjugate variables to machine-learning protocols noted in Lin et al. (2023).

B. Elements of Quaternion Eigensystems

We will frequently refer to basic linear algebra methods that have been a mainstay of the quaternion-based EnP solutions, and which embed the origin of the quaternion adjugate variables. The essence starts with a 4×4 matrix M, one of whose four eigenvalues, say ϵ_{opt}, has an eigenvector, say q_{opt}, that we need. The eigenvalues are computed by taking the characteristic equation,

$$\chi = [M - \epsilon I_4] \,, \tag{5}$$

solving the quartic polynomial $\det \chi(\epsilon) = 0$, and selecting one of the roots as ϵ_{opt}. Classic linear algebra now leads us to an elegant calculation of the eigenvector corresponding to any single root ϵ_{opt}. Obviously $\chi(\epsilon_{\text{opt}})$ is singular since $\det \chi(\epsilon_{\text{opt}}) = 0$. However, the first step of computing the inverse of $\chi(\epsilon_{\text{opt}})$ is *still legal*: that is, the transpose cofactor or *adjugate* of the 4×4 matrix χ, $A_{\text{opt}} = \text{ Adjugate } \big[\chi(\epsilon_{\text{opt}})\big]$ is singularity-free. We next exploit the fact that, by definition, the adjugate construction yields the determinant:

$$\chi(\epsilon_{\text{opt}}) \cdot A_{\text{opt}} = \det \chi(\epsilon_{\text{opt}}) \ .$$

But since $\det \chi(\epsilon_{\text{opt}}) = 0$, we can expand χ in terms of its original components Eq. (5), resulting in

$$\begin{aligned} \chi \cdot A = \big[M - \epsilon_{\text{opt}} I_4\big] \cdot A = \det \chi(\epsilon_{\text{opt}}) I_4 = 0 \\ \Rightarrow \ M \cdot A_i = \epsilon_{\text{opt}} \cdot A_i \ . \end{aligned} \tag{6}$$

Each of the adjugate matrix A's 4-vector columns A_i is therefore proportional to the *same* quaternion eigenvector of M. Some of those may be proportional to zero, eliminating those columns from consideration; in fact there *fourteen* different possible unnormalizable cases, but since the eigenvector q_{opt} obeys the constraint $q \cdot q = 1$, there is always at least *one* normalizable candidate in the four columns of A, and we simply normalize one of those to always obtain a legal quaternion eigenvector. For quaternion-defining matrices M, this adjugate matrix is exactly proportional to Eq. (2), hence the origin of our "quaternion adjugate variable" terminology.

C. Least-Squares Loss Measures for EnP and OnP

For the 3D-3D (EnP) and 3D-2D:Orthographic (OnP) pose discovery problems, exact or noisy, we will define least squares loss expressions based upon sets of points in Euclidean space. Our fundamental data will be a $3 \times K$ matrix $\mathbf{X} = \{\mathbf{x}_k\}$ denoting our model point set, the *reference data*. The 3D points $\mathbf{x}_k$ are arranged in $\mathbf{X}$ so that an arbitrary orthonormal proper rotation R in $\mathbf{SO}(3)$ can rotate from the left to produce the (hypothetically experimentally measured) *target data*, $\mathbf{Y} = R \cdot \mathbf{X} = \{R \cdot \mathbf{x}_k\} = \{\mathbf{y}_k\}$. For orthographically projected data, we use only the 2×3 projection matrix $P = [R_1, R_2]^{\text{t}}$, and write the image 2-vector data as $\mathbf{U} = P \cdot \mathbf{X} = \{P \cdot \mathbf{x}_k\} = \{\mathbf{u}_k\}$. We will consider $\mathbf{Y}$ and $\mathbf{U}$ both with exact data and with Gaussian noise with standard deviation σ added to the elements of each target vector after rotation and possible projection.

Our basic EnP loss expression constructs a sum of 3D Euclidean squared differences between each reference point and its matching, possibly noisy, target point; 2D Euclidean squared distances between ideal projected reference points and the target projected image points are summed in the OnP variant. The sums of Euclidean differences vary smoothly with changes in the proposed quaternion-based rotation transformation $R(q)$, and the objective is to find the optimal values q_{opt} or $R_{\text{opt}} \equiv R(q_{\text{opt}})$ with the smallest possible loss. The EnP and OnP losses then are written

$$\mathbf{S}_{\text{3D-3D EnP}} = \sum_{k=1}^{K} \| R(q) \cdot \mathbf{x}_k - \mathbf{y}_k \|^2 \ , \tag{7}$$

$$\mathbf{S}_{\text{3D-2D Ortho OnP}} = \sum_{k=1}^{K} \| P(q) \cdot \mathbf{x}_k - \mathbf{u}_k \|^2 \ . \tag{8}$$

Closed form algebraic solutions are well known for rotations that minimize the EnP Eq. (7) with exact or noisy data (see, e.g., Hanson (2020) for a review), while only exact data, not noisy data, can be accommodated in closed form with solutions known at this time for the OnP Eq. (8) (see, e.g., Steger (2018); Hajder (2017)).

IV. The Determinant Ratio Matrix

We now present the DRaM formulas solving the exact-data EnP and OnP problems and outline their origins and properties. A synopsis of the authors' original derivation (Hanson and Hanson, 2022) that may be of interest to some readers is given in Appendix A. Appendix B supplies the basic algorithms for all of the following. We begin by noting that one can determine from Eq. (7) and Eq. (8) that, when the loss functions are expanded, there are no other variables upon which the rotation matrix can depend except the set of the cross-covariance elements that can be constructed from the data matrices $\mathbf{X} = \{[x_k, y_k, z_k]\}$, $\mathbf{Y} = \{[u_k, v_k, w_k]\}$, and $\mathbf{U} = \{[u_k, v_k]\}$. We denote the members of this set as $\{\text{xx}, \text{xy}, \text{xz}, \text{yy}, \text{yz}, \text{zz}\}$ and $\{\text{xu}, \text{yu}, \text{zu}, \text{xv}, \text{yv}, \text{zv}, \text{xw}, \text{yw}, \text{zw}\}$, where $\text{xx} = \sum_k x_k x_k$, $\text{xy} = \sum_k x_k y_k$, $\text{xz} = \sum_k x_k z_k$ and $\text{ux} = \sum_k u_k x_k$, $\text{uy} = \sum_k u_k y_k$, $\text{uz} = \sum_k u_k z_k$, and so on. The essence of the derivation is to reparameterize the rotation matrices in Eq. (7) and Eq. (8) as $R(q_{ij})$ using the adjugate quaternion variables as shown in Eq. (3). Expanding the resulting loss functions in terms of the q_{ij} and the cross-covariance sums $\{\text{xx}, \text{xy}, \text{xz}, \ldots\}$ results in a pair of complicated but tractable expressions quadratic in q_{ij}, much simpler than the original equations expressed quartically in q_i. In summary, after trying many combinations of the loss function derivatives with respect to the ten q_{ij} and the seven constraints Eq. (4) using the Mathematica solver, two distinct combinations were found to yield exact-data solutions for the EnP and the OnP system in 0.5 seconds and 7.0 seconds, respectively (details are given in Appendix A). Imposing *all* derivative equations and constraints together would yield a solution that would solve the least squares problem with or without error, but we could not find such a solution in DRaM form. The exact-data solutions in fact can be extended easily from the quaternion-restricted 3D space domain to the EnP or OnP problem in *any* ND Euclidean space, as we will show explicitly below.

The EnP Case. When we insert the ten solutions for $\{q_{ij}\}$, into the nine rotation matrix expressions in Eq. (3), we find that the rotation-matrix elements are composed of ratios of simple 3×3 determinants of the 15 potentially independent cross-covariances, each with the common denominator given by the self-covariance determinant

$$d_0 = \det \begin{bmatrix} \text{xx} & \text{yx} & \text{zx} \\ \text{xy} & \text{yy} & \text{zy} \\ \text{xz} & \text{yz} & \text{zz} \end{bmatrix} . \tag{9}$$

Each of the numerators of the rotation matrix elements R_{ij} takes the form of a 3×3 determinant d_{ij}, where, for example,

$$d_{11} = \det \begin{bmatrix} \text{yx} & \text{zx} & \text{ux} \\ \text{yy} & \text{zy} & \text{uy} \\ \text{yz} & \text{zz} & \text{uz} \end{bmatrix} . \tag{10}$$

The remaining elements of the set $\{d_{ij}\}$ are constructed from cyclic permutations, with $[u, v, w]$ in the columns $[d_{i1}, d_{i2}, d_{i3}]$, respectively, and the first two columns having cyclic ordering of the elements $[(y, z), (z, x), (x, y)]$. The full DRaM matrix for the exact-data EnP pose is then

$$R(x, y, z; u, v, w) = \begin{bmatrix} \frac{d_{11}}{d_0} & \frac{d_{12}}{d_0} & \frac{d_{13}}{d_0} \\ \frac{d_{21}}{d_0} & \frac{d_{22}}{d_0} & \frac{d_{23}}{d_0} \\ \frac{d_{31}}{d_0} & \frac{d_{32}}{d_0} & \frac{d_{33}}{d_0} \end{bmatrix} . \tag{11}$$

It is easy to verify for error-free data that $R(x, y, z; u, v, w)$ gives the exact same numerical answer as any other formulas, as tabulated in Table I. However, as one can immediately see, the DRaM formula has the remarkable advantage that one can actually see how the elements of the optimal pose-determining rotation matrix relate to the underlying cross-covariance structure and geometry. If this problem had come to the attention of Gauss and his contemporaries, this formula could have been part of the mathematical folklore for the last two hundred years.

The OnP Case. We originally considered the adjugate quaternion expansion of the OnP loss function Eq. (8) independently, and by a similar but distinct solution procedure (see Hanson and Hanson (2022) and Appendix A), we found solutions for the ten q_{ij} that were very complicated expressions of the cross-covariances, involving square roots with very long and seemingly intractable arguments. However, when substituted into the top two rows of Eq. (3), all the complications collapse, yielding a 2×3 projection matrix $P(x, y, z; u, v)$ solving the exact-data OnP problem that was *exactly* the top two rows of the EnP solution Eq. (11). This was what it had to be for exact data, since the bottom row of a 3×3 rotation matrix must be the cross-product of the first two rows for consistency. Nevertheless, the required consistency can be discovered independently for the OnP problem. The bottom row itself, given only the 2D OnP data $\mathbf{U}\{[u_k, v_k]\}$, happens to have its own elegant form computable from the cross-product as

$$\tilde{d}_{31} \to \begin{bmatrix} \text{xx} & \text{ux} & \text{vx} \\ \text{xy} & \text{uy} & \text{vy} \\ \text{xz} & \text{uz} & \text{vz} \end{bmatrix} . \tag{12}$$

If we write Eq. (12) as $\tilde{d}_{31}(x, u, v)$, then the rest of the bottom row of the OnP exact-data DRaM is $\tilde{d}_{32}(y, u, v)$ and $\tilde{d}_{33}(z, u, v)$, and we can write the OnP pose solution projection matrix, extended from 2×3 to 3×3, as

$$P(x, y, z; u, v) = \begin{bmatrix} \frac{d_{11}}{d_0} & \frac{d_{12}}{d_0} & \frac{d_{13}}{d_0} \\ \frac{d_{21}}{d_0} & \frac{d_{22}}{d_0} & \frac{d_{23}}{d_0} \\ \frac{\tilde{d}_{31}}{d_0} & \frac{\tilde{d}_{32}}{d_0} & \frac{\tilde{d}_{33}}{d_0} \end{bmatrix} . \tag{13}$$

V. Equivalent DRaM Derivation using Least Squares with Rotation Matrix Itself

The previous Section describes the quaternion-motivated research process that led to the DRaM solutions for EnP and OnP in the form of Eq. (11) and Eq. (13). However, *once that form was known*, we began to wonder if we had neglected other simpler approaches to these forms: in fact, we had missed an important alternative, made obvious by doing a reverse parameterization of Eq. (3), which takes the form

$$\Big\{ q_{00} \to \frac{1}{4}\left(r_{11} + r_{22} + r_{33} + 1\right), q_{11} \to \frac{1}{4}\left(r_{11} - r_{22} - r_{33} + 1\right), \\ q_{22} \to \frac{1}{4}\left(-r_{11} + r_{22} - r_{33} + 1\right), q_{33} \to \frac{1}{4}\left(-r_{11} - r_{22} + r_{33} + 1\right), \\ q_{01} \to \frac{1}{4}\left(r_{32} - r_{23}\right), q_{02} \to \frac{1}{4}\left(r_{13} - r_{31}\right), q_{03} \to \frac{1}{4}\left(r_{21} - r_{12}\right), \\ q_{23} \to \frac{1}{4}\left(r_{23} + r_{32}\right), q_{13} \to \frac{1}{4}\left(r_{13} + r_{31}\right), q_{12} \to \frac{1}{4}\left(r_{12} + r_{21}\right) \Big\} . \tag{14}$$

Therefore one would expect that simply substituting the matrix

$$R(r_{ij}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{15}$$

into the loss Eqs. (7) and (8) with some constraints could also yield the DRaM formulas. We had rejected this possibility originally because our hypothesis was that we could only solve the least-squares problem with the nine r_{ij} derivatives of the loss functions if we included constraints such as the six conditions $\sum_k r_{ik} r_{jk} = \delta_{ij}$ on the rows, or at the very least the constraint $\det R = 1$, analogous to the adjugate quaternion constraint $q_{00} + q_{11} + q_{22} + q_{33} = 1$. These symbolic optimization problems over the r_{ij} variables produced empty results with our available Mathematica solution software. However, we were aware that, for the perfect-data case, we had in some cases achieved correct quaternion-based solutions even with missing constraints. When we finally applied the solver to the loss functions with *only* the nine vanishing r_{ij} derivatives, and with *no* constraints, both the EnP and OnP systems immediately yielded the DRaM equations. Theoretically, once the system with full r_{ij} constraints failed, that would have removed that method from consideration; why eliminating the constraints, which would be necessary for a

noisy-data solution, gives an unsuspected exact-data solution is puzzling. But, in any case, we report that we can use either set of variables, $\{q_{ij}\}$ or $\{r_{ij}\}$, to produce exact-data DRaM pose solutions for EnP and OnP, provided one stumbles on the serendipitous combination of constraints or lack thereof to supply to the solver. In the next Section, we will exhibit our discovery of a *third* unexpected DRaM derivation that is entirely independent of any process involving the optimization of loss functions.

VI. Independent DRaM Derivation and Extension to All Dimensions.

Once Eq. (11) and Eq. (13) present themselves as the result of solving the EnP and OnP least squares problems, one recognizes that the structures of the DRaM numerator determinants have very suggestive linear algebraic properties. One is led to look at the transformation of the reference elements $\mathbf{x}_k = [x, y, z]_k$ to the elements of the target data $\mathbf{y}_k = [u, v, w]_k$ explicitly for each k, which take the form

$$\begin{aligned} u &= r_{11}\, x + r_{12}\, y + r_{13}\, z \\ v &= r_{21}\, x + r_{22}\, y + r_{23}\, z \\ w &= r_{31}\, x + r_{32}\, y + r_{33}\, z \ . \end{aligned} \tag{16}$$

Then, since the typical mixed cross-covariances have the form

$$\mathrm{ux} = \sum_k x_k u_k = \sum_k x_k \left(r_{11} x_k + r_{12} y_k + r_{13} z_k\right) \ , \tag{17}$$

we can look at a representative determinant with one column of mixed cross-covariances and write out the reduction of the expression

$$\begin{aligned} d_{11} &= \det \begin{bmatrix} \mathrm{xy} & \mathrm{xz} & \mathrm{ux} \\ \mathrm{yy} & \mathrm{yz} & \mathrm{uy} \\ \mathrm{zy} & \mathrm{zz} & \mathrm{uz} \end{bmatrix} \\ &= \det \begin{bmatrix} \mathrm{xy} & \mathrm{xz} & r_{11}\,\mathrm{xx} + r_{12}\,\mathrm{xy} + r_{13}\,\mathrm{xz} \\ \mathrm{yy} & \mathrm{yz} & r_{11}\,\mathrm{xy} + r_{12}\,\mathrm{yy} + r_{13}\,\mathrm{yz} \\ \mathrm{zy} & \mathrm{zz} & r_{11}\,\mathrm{xz} + r_{12}\,\mathrm{zy} + r_{13}\,\mathrm{zz} \end{bmatrix} \\ &= \det \begin{bmatrix} \mathrm{xy} & \mathrm{xz} & r_{11}\,\mathrm{xx} \\ \mathrm{yy} & \mathrm{yz} & r_{11}\,\mathrm{xy} \\ \mathrm{zy} & \mathrm{zz} & r_{11}\,\mathrm{xz} \end{bmatrix} = \ r_{11}\, d_0 \ . \end{aligned} \tag{18}$$

Strikingly, by the standard determinant column-subtraction equivalence rules, everything cancels out except the overall multiple of the rotation matrix element r_{11} by the self-covariance determinant d_0, which is exactly the DRaM denominator. The same argument holds for all the d_{ij} determinants, and so Eq. (11) is identical to the rotation matrix $R = [r_{ij}]$, and our proof is complete. We can easily see that for cloud distributions in N-dimensional Euclidean space, the exact same proof goes through, and, for error-free data, we can compute the $\mathbf{SO}(N)$ matrix for the EnP task, rotating one N-dimensional cloud into another using simple determinants of the cross-covariance elements. Similar arguments hold for the OnP case involving $(N-1)$-dimensional orthographic image data, with the additional exploitation of the N-dimensional generalization of the cross-product to obtain the last row of the $\mathbf{SO}(N)$ rotation. Therefore, the entire EnP and OnP DRaM representations solving the error-free pose-estimation problem follow from the most elementary rules of linear algebra, with no connection to the least-squares optimization problems that led us originally to the DRaM.

VII. Additional Members of the DRaM Class of Exact-Data Solutions: QR and Moore-Penrose

Once the DRaM solutions are known, further investigation reveals that the DRaM is not an isolated case, but is in fact the most intuitive form belonging to a *class* of (at least) three distinct algebraic methods with *identical* numerical behavior under all conditions of which we are aware. In particular, all three solve the exact-data cases of both EnP and OnP, and all produce rotation matrix candidates that deviate from orthonormality in exactly the same way for all reasonable values of noise and model data sets. As noted by Hajder (2017), at least four data points are generally required for this class. We now outline the properties of these alternative methods, assembling detailed comparisons of the assorted techniques below in Table I.

QR Decomposition Transformation of Loss Function to Trivial Frobenius Norm of matrices. Our first alternative solution of the DRaM class is based on an exploitation of the QR decomposition suggested in the extensive review of the OnP problem by Steger (2018). This method takes the reference data array $\mathbf{X}$ appearing in both Eqs. (7, 8) and rearranges its QR decomposition to convert the conventional sum of Euclidean squared differences in either 3D or 2D to a Frobenius difference between two small matrices, that is, to the sum of squared differences of all the elements of these matrices. We begin with the fundamental QR decomposition, which is

$$\mathbf{QR}\left(\mathbf{X}\right) = \{S, T\} \quad \text{where} \quad \mathbf{X} = S^{\mathrm{t}} \cdot T \ . \tag{19}$$

For our default data array dimension $\mathbf{X} \to 3 \times K$, the QR elements are $S \to 3 \times 3$ obeying $S^{\mathrm{t}} \cdot S = I_3$ and $T \to 3 \times N$ upper triangular in the left three columns. This enables us to multiply the $3 \times K$ form of the squared loss expression on the right by the Moore-Penrose pseudoinverse T^{+} of T to perform the transformation

$$\|R \cdot \mathbf{X} - \mathbf{Y}\|_2^2 \to \|R \cdot S^{\mathrm{t}} - \mathbf{Y} \cdot T^{+}\|_2^2 \ . \tag{20}$$

The exact same equations hold for OnP with $R \to P$ and the $3 \times K$ target data $\mathbf{Y}$ replaced by the $2 \times K$ projected data $\mathbf{U}$. Since for square matrices, the sum of columns-squared is the sum of squares of all elements, the transformed EnP and OnP losses can then be written not as a sum of K separate Euclidean vector differences but as much simpler Frobenius norms of 3×3 matrices for EnP or 2×2 matrices for OnP: (Steger (2018)):

$$\mathbf{S}_{\text{QR EnP}} = \|R(q) \cdot S^{\mathrm{t}} - \mathbf{Y} \cdot T^{+}\|_{\text{Frob}}^2 \tag{21}$$

$$\mathbf{S}_{\text{QR OnP}} = \|P(q) \cdot S^{\mathrm{t}} - \mathbf{U} \cdot T^{+}\|_{\text{Frob}}^2 \ . \tag{22}$$

However, we now observe that can go *one step further* than the transformation employed in Steger (2018), to obtain *explicit*

closed form solutions for the OnP pose rotation, provided the data are exact. The additional transformation simply inserts an identity that reduces each half of the squared loss to a multiplication by the *entire* inverse of the QR map of $\mathbf{X}$ as follows:

$$\|R \cdot \mathbf{X} - \mathbf{Y}\|_2^2 = \|R - \mathbf{Y} \cdot T^+ \cdot S\|_{\text{Frob}}^2 \ . \tag{23}$$

One can easily verify that

$$\mathbf{R}_{\text{QR EnP opt}} = \mathbf{Y} \cdot T^+ \cdot S \tag{24}$$

$$\mathbf{P}_{\text{QR OnP opt}} = \mathbf{U} \cdot T^+ \cdot S \tag{25}$$

immediately solve the exact-data pose discovery problem for EnP and OnP, while the noisy-data rotation matrices have corrupted orthonormality. Thus the QR method requires the same rotation correction that we have already noted for the DRaM.

Remark: The QR decomposition works slightly differently, avoiding the pseudoinverse, and corresponding more closely to the presentation of Steger, if one presents the data arrays as $\mathbf{X} \to K \times 3$, i.e., with loss element $(R \cdot \mathbf{X}^{\text{t}} - \mathbf{Y}^{\text{t}})$. In this case, $S \to 3 \times K$, still satisfying $S \cdot S^{\text{t}} = I_3$, but now $T \to 3 \times 3$ is *square* upper triangular and normally admits an ordinary inverse. The multiplications in the loss elements are therefore reversed, so

$$R_{\text{opt}} = \mathbf{Y}^{\text{t}} \cdot S^{\text{t}} \cdot \left(T^{\text{t}}\right)^{-1} \ .$$

Moore-Penrose PseudoInverse Map of Reference Array. Our second alternative solution in the DRaM class involves the Moore-Penrose pseudoinverse. We already exploited the pseudoinverse in the QR decomposition above, but in fact, as suggested, for example, in the POSIT treatment of Dementhon and Davis (1995) and in Hajder (2019), we can also carry that idea one step further, omitting the QR decomposition altogether. Instead of applying the pseudoinverse to the T matrix in $\mathbf{X} = S^t \cdot T$, we can eliminate $\mathbf{X}$ entirely by multiplying inside the loss expression by the pseudoinverse

$$\mathbf{X}^+ = (\mathbf{X}^{\text{t}} \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^{\text{t}} \tag{26}$$

of the data matrix $\mathbf{X}$ *itself*, in either $3 \times K$ or $K \times 3$ format. One can verify that, given exact data for both the EnP loss and the OnP loss, the correct rotation matrices are obtained from

$$\mathbf{R}_{\text{P.I. EnP opt}} = \mathbf{Y} \cdot \mathbf{X}^+ \tag{27}$$

$$\mathbf{P}_{\text{P.I. OnP opt}} = \mathbf{U} \cdot \mathbf{X}^+ \ . \tag{28}$$

Written out explicitly, these become

$$\mathbf{R} = \sum_k Y_{3k} \cdot \text{PseudoInverse}[X_{3k}]$$

$$\mathbf{P} = \sum_k U_{2k} \cdot \text{PseudoInverse}[X_{3k}] \ .$$

One can recognize the pseudoinverse solutions as closely related to the result of the transition from the initial QR decomposition in Eq. (21) and Eq. (22) to Eq. (24) and Eq. (25).

Noise Abatement. For noisy data, the candidate rotation matrices of all of the last three methods, the QR decomposition of $\mathbf{X}$ and the pseudoinverse of $\mathbf{X}$, as well as the DRaM of cross-covariances, deviate from orthonormality in exactly the same way. This defect can be partially healed using the rotation correction methods introduced in Section IX, which are themselves based on the algorithms in the next section that produce gold standard orthonormal rotation matrices solving the EnP problem for data with or without error. Even when corrected, none of the DRaM class of algorithms correspond exactly to the gold standard pose solutions, though they are very close.

VIII. The RMSD Class: Methods that Exactly Solve EnP with or without Noise

For the EnP 3D cloud alignment task, we know a family of algorithms that produce the optimal aligning rotation without regard to whether the data are exact or noisy, unlike the DRaM class we have just presented. We refer to this set of methods as the "RMSD Class" for convenience. Several of these methods supply our rotation correction needs in addition to their roles in the EnP problem, so it is important to list them all explicitly here. Remarkably, none of these approaches (except `ArgMin`) can solve the noisy OnP problem (Steger, 2018). The best we can do is to replace the target $\mathbf{Y}$ data in the EnP problem by a 3D constant-z extension of the 2D OnP $\mathbf{U}$ data. This does result in pure uncorrupted orthonormal rotation matrices, but their properties are very poor. We now examine these methods in turn; all give the same numerical answers, despite very distinct appearances, though some (for example the 3D SVD method and the maximal quaternion eigensystem method), have been proven identical (Coutsias et al., 2004). As for the DRaM class, the RMSD class methods have their properties given in Table I.

1. ArgMin: The "Gold Standard" for least squares problems such as EnP and OnP (as well as PnP) is numerical search in the space of rotations to find the, hopefully global, minimum value of the loss. The basic algorithm is typically represented in quaternion space, e.g.,

$$q_{\text{opt}} = \ \underset{q}{\text{argmin}} \left\|R(q) \cdot \mathbf{X} - \mathbf{Y}\right\|^2 , \tag{29}$$

subject to the constraint $q \cdot q = 1$, which is generally more stable than searching with the constraints of the full rotation matrix. Typically Levenberg-Marquardt algorithms handle this well, while the 11^{th}-degree polynomial Lagrange-multiplier method that Hajder (2017) proposed for OnP should also be sufficient. The solution to the pose problem is then $R_{\text{opt}} = R(q_{\text{opt}})$. It is this value against which all other methods must be compared for evaluation. *Using the data-simulation defining rotation R_{init} is simply wrong.* Once noise is introduced, R_{init} is completely uncomputable, and no possible single algebraic formula independent of the input data can ever recover its value. In contrast, there are actual algebraic formulas, with explicit examples in this section, that can extract the exact `ArgMin` value of the optimal pose from any data.

2. QMIN: The classic full least squares formula Eq. (7) was solved as a quaternion eigensystem problem requiring the computation of the *minimum* quaternion eigenvalue by Faugeras and Hebert (1983). By a clever insertion of an identity quaternion expansion, they reduced Eq. (7) to a 4×4 quaternion matrix system of the form

$$q \cdot [B(x, y, z; u, v, w)] \cdot q \ , \tag{30}$$

where the B-matrix is defined as follows: first we define, for each k, the matrix $A(\mathbf{x}_k, \mathbf{u}_k))$ as

$$A_k = \begin{bmatrix} 0 & -a_1 & -a_2 & -a_3 \\ a_1 & 0 & s_3 & -s_2 \\ a_2 & -s_3 & 0 & s_1 \\ a_3 & s_2 & -s_1 & 0 \end{bmatrix}_k \tag{31}$$

where, with "a" for "antisymmetric" and "s" for "symmetric,"

$$\left.\begin{aligned} a_{\{1,2,3\}} &= \{x - u,\ y - v,\ z - w\} \\ s_{\{1,2,3\}} &= \{x + u,\ y + v,\ z + w\} \end{aligned}\right\} \ . \tag{32}$$

Then, for each k, we convert the A matrix into a symmetric matrix with real eigenvalues,

$$B_k \ = \ {A_k}^{\mathrm{t}} \cdot A_k \ = \\ \begin{bmatrix} {a_1}^2 + {a_2}^2 + {a_3}^2 & a_3 s_2 - a_2 s_3 & a_1 s_3 - a_3 s_1 & a_2 s_1 - a_1 s_2 \\ a_3 s_2 - a_2 s_3 & {a_1}^2 + {s_2}^2 + {s_3}^2 & a_1 a_2 - s_1 s_2 & a_1 a_3 - s_1 s_3 \\ a_1 s_3 - a_3 s_1 & a_1 a_2 - s_1 s_2 & {a_2}^2 + {s_1}^2 + {s_3}^2 & a_2 a_3 - s_2 s_3 \\ a_2 s_1 - a_1 s_2 & a_1 a_3 - s_1 s_3 & a_2 a_3 - s_2 s_3 & {a_3}^2 + {s_1}^2 + {s_2}^2 \end{bmatrix}_k$$

and define our B matrix in Eq. (30) as the sum of these elements:

$$B = \sum_{k=1}^{K} B_k \ . \tag{33}$$

Using the full squared-difference minimization measure Eq. (7) requires the global minimal value, so the solution for the optimal quaternion in Eq. (30) is the eigenvector of the *minimal* eigenvalue of B in Eq. (33). If ϵ_{opt} is the minimal eigenvalue of B, the adjugate of the characteristic equation $\chi = \left[B - \epsilon_{\text{opt}} I_4\right]$ is proportional to $\text{Adj}(q)$ in Eq. (2), from which we can select a nonsingular version of the quaternion eigenvector q_{opt} determining $R_{\text{opt}} = R(q_{\text{opt}})$. This is the approach used by Faugeras and Hebert in the earliest application of the quaternion method to scene alignment of which we are aware. *Error-free Eigenvalue:* For exact data, $\epsilon_{\text{opt}} \equiv 0$, independent of any data, so q_{opt} can be calculated immediately from B alone (which remains data dependent).

3. QMAX: Perhaps the most common EnP solution is based on the quaternion eigensystem resulting from maximizing the negative cross-term of the loss function Eq. (7) (see, for example, Horn (1987), or Hanson (2020) for a review). The eigensystem then takes the form

$$\Delta(R(q); \mathbf{X}, \mathbf{Y}) = \text{tr}\, R(q) \cdot E \ = \ q \cdot M(E) \cdot q \ , \tag{34}$$

which achieves its optimum when the quaternion q is the eigenvector of the *maximal* eigenvalue of $M(E)$. Here E is the cross-covariance matrix

$$E(\mathbf{X}, \mathbf{Y}) = \mathbf{X} \cdot \mathbf{Y}^{\mathrm{t}} = \begin{bmatrix} \text{ux} & \text{vx} & \text{wx} \\ \text{uy} & \text{vy} & \text{wy} \\ \text{uz} & \text{vz} & \text{wz} \end{bmatrix} \ . \tag{35}$$

and $M(E)$ is the *profile matrix*, a traceless, symmetric 4×4 matrix following from inserting Eq. (1) into Eq. (34), and expanding the coefficients to give

$$M(E_{ab}) = \begin{bmatrix} E_{11} + E_{22} + E_{33} & E_{23} - E_{32} & E_{31} - E_{13} & E_{12} - E_{21} \\ E_{23} - E_{32} & E_{11} - E_{22} - E_{33} & E_{12} + E_{21} & E_{31} + E_{13} \\ E_{31} - E_{13} & E_{12} + E_{21} & -E_{11} + E_{22} - E_{33} & E_{23} + E_{32} \\ E_{12} - E_{21} & E_{31} + E_{13} & E_{23} + E_{32} & -E_{11} - E_{22} + E_{33} \end{bmatrix} . \tag{36}$$

If ϵ_{opt} is the maximal eigenvalue of $M(E)$, the adjugate of the characteristic equation $\chi = \left[M - \epsilon_{\text{opt}} I_4\right]$ is proportional to $\text{Adj}(q)_{ij}$ in Eq. (2), from which we can select a nonsingular version of the quaternion eigenvector q_{opt} determining $R_{\text{opt}} = R(q_{\text{opt}})$. *Error-free Eigenvalue:* For exact data, $\epsilon_{\text{opt}} = \text{tr}\, E_0 = (\text{xx} + \text{yy} + \text{zz})$ is rotation-independent and q_{opt} can be calculated immediately from M (which remains rotation dependent), omitting the eigenvalue computation step.

4. SVD: The singular value decomposition method (see, for example, Schönemann (1966) among a number of others), which is known (Coutsias et al., 2004) to be identical to the QMAX method, follows from examining the SVD of the cross-covariance matrix, $\{U, S, V\} = \text{SVD}\left(E(\mathbf{X}, \mathbf{Y})\right)$. One can then express the original matrix as $E(\mathbf{X}, \mathbf{Y}) = U \cdot S \cdot V^{\mathrm{t}}$, and rearrange the decomposition elements to form an orthonormal matrix solving the corresponding Frobenius norm problem, and taking the form

$$R_{\text{opt}} = V \cdot D \cdot U^{\mathrm{t}} \ , \tag{37}$$

where $D = \text{diag}[1, 1, \text{sign}(\det U \det V))$. This version of R_{opt} is the same as the QMIN and QMAX solutions, and can also be applied to a standalone matrix E^{t} (with no available $\mathbf{X}$ and $\mathbf{Y}$ data) to produce an orthonormal rotation matrix with the smallest possible Frobenius distance from the input matrix E^{t}.

5. HHN: Our final example of the error-insensitive RMSD class of solutions to the EnP optimal pose discovery problem we will label as "HHN" corresponding to the method of Horn et al. (1988), although the general approach was known some time earlier. While this method looks very similar to a pseudoinverse method (see Eq. (26)), it involves using a matrix square root to construct a perfect rotation matrix from the profile matrix $E(\mathbf{X}, \mathbf{Y})$, and has two alternative forms:

$$R^1_{\text{opt}} = \left(E^{\mathrm{t}} \cdot E\right)^{-1/2} \cdot E^{\mathrm{t}} \tag{38}$$

$$R^2_{\text{opt}} = \left(E^{\mathrm{t}} \cdot E\right)^{+1/2} \cdot E^{-1} \ . \tag{39}$$

Similar to the SVD, this method will also produce the closest possible orthonormal rotation to data for which only the matrix E^{t} is given.

Remark on OnP Adaptations. The QMIN and QMAX methods can be applied to OnP $\mathbf{U}$ data if it is extended to 3D by placing it in a plane at some fixed z value, and they guarantee orthonormal matrix results, but these disagree badly with the ideal `ArgMin` values. The SVD and HHN methods have natural extensions to rectangular matrices such as $\mathbf{X} \cdot \mathbf{U}^{\mathrm{t}}$, but those also have exactly the same disagreements with the `ArgMin` , as noted in Table I. If there is a way to reproduce the exact agreement with `ArgMin` for noisy OnP data, corresponding with the perfect EnP results, we have not yet discovered it.

IX. Rotation Correction for the DRaM Class of Matrices.

The family of pose estimation methods including DRaM, QR, and PseudoInverse produces perfect solutions and valid rotation matrices only for exact input data; with the introduction of noise, the rotation matrices become corrupted and are no longer orthonormal, though they may be fairly close to orthonormal. In order to achieve the best possible information from the DRaM family, we therefore are motivated to augment these formulas with a final step, the computation of the *closest possible pure rotation* to the output matrices corrupted by noise. This can be achieved by repurposing our basic EnP pose estimation methods to minimize the Frobenius norm relating a pure rotation matrix to our noisy matrices. We thus begin with the loss function

$$S_{\text{Frobenius}} = \|R(q) - S\|^2_{\text{Frob}} = \text{ [const]} + \text{[const]}\,\mathrm{tr}\left(R(q) \cdot S^{\mathrm{t}}\right) \quad \rightarrow \quad q \cdot M(s_{ab}) \cdot q \ . \tag{40}$$

Here the rearrangement of the rotation data to produce the 4×4 profile matrix $M(s_{ab})$ parallels the construction of Eq. (36) for the QMAX pose method, except the matrix indices on S are *transposed* with respect to those of E in Eq. (36). Solving this as a quaternion eigenvalue problem is the method proposed by Bar-Itzhack (2000). However, Bar-Itzhack's work included several more details: in his "Version 1" method, he observes that the technique works for 2×3 projection matrices if one simply omits the bottom row of s_{ab} in Eq. (40), while his "Version 2" is essentially the QMAX method modified with the intent of discovering the quaternion of a rotation matrix in a more elegant way than the traditional method of Shepperd (1978). In his "Version 3", he notes that, in addition to simply computing the quaternion for a 2×3 projection matrix or a full 3×3 rotation matrix, the technique is perfectly well-suited for computing rotation correction, since it minimizes the Frobenius distance between a noisy rotation and a perfect quaternion-defined symbolic rotation.

In addition to Bar-Itzhack's adaptation of the QMAX quaternion eigenvalue method for computing the nearest quaternion to perfect or noisy rotation candidates, each of the other members of the class except the QMIN accept a 3×3 or 2×3 matrix as direct input, and, properly arranged, can produce a correction from a noisy rotation matrix to an orthonormal rotation matrix. The SVD rotation correction approach is well known, appearing, for example, in Björck and Bowie (1971); Higham (1986); Schönemann (1966). The SVD methods work in any dimension, and are very simple to use, as simply supplying the transpose S^{t} of a candidate rotation S to the SVD method in place of the cross-covariance E will generate the corrected rotation as the result of Eq. (37). Alternatively, one can directly implement the correction algorithm using $\{U, S, V\} = \text{SVD}\left(R_{\text{candidate}}\right)$. One can then express the orthonormal matrix solving the corresponding Frobenius norm problem as the transpose of Eq. (37),

$$R_{\text{opt}} = U \cdot D \cdot V^{\mathrm{t}} \ . \tag{41}$$

A similar procedure works for the HNN formulation. Finally, we note that, just as Bar-Itzhack's method works for abbreviated 2×3 projection-matrix portions of a rotation matrix, the SVD and HNN methods are similarly entirely capable of handling rectangular matrices. For example, if we write $\{U, S, V\} = \text{SVD}\left(\text{Mat23}\right)$, we find that if we choose the 2-row matrix $D_{23} = [[100], [010]]$, then we have a solution for a partial ideal rotation matrix of the form

$$\text{Orthogonal 2 x 3 Matrix} = U \cdot D_{23} \cdot V^{\mathrm{t}} \ . \tag{42}$$

X. Experimental Results

We next present some simple illustrations of the overall quality of our method. Starting with a list of randomly generated 3D point clouds of unit radius and applying a rotation based on a uniform randomly selected quaternion, we produced a noise-free alignment problem either in the form of rotated 3D target cloud, or a corresponding 2D projection of the rotated 3D cloud to a plane. Optional noise with, e.g., $\sigma = 0.1$ was added to the 3D target point sets or the 2D target point sets, and each point set is recentered with center of mass at the origin to avoid anomalies. We began by considering three ways of computing a list of 3D-3D and 3D-2D losses Eq. (7) and Eq. (8) for zero noise and some typical random noise:

- R_{init}: Compute the loss with the bare rotation used to simulate each dataset via $\mathbf{u} = R_{\text{init}} \cdot \mathbf{x} + \mathbf{err}$, truncating to a 2D plane for the 3D-2D data.
- R_{DRaM}: Evaluate the loss with the DRaM rotations Eq. (11), which may be invalid rotations.
- $R_{\text{RotOpt:DRaM}}$: In the case with non-vanishing noise, correct the warped DRaM rotation candidates to the nearest pure rotation using a rotation-correction algorithm such as the Bar-Itzhack (denoted "BI:") or SVD methods.
- R_{opt}: Evaluate the loss with the exact optimal aligning rotation matrix results incorporating added error. Applying a standard library `ArgMin` numerical search algorithm to Eq. (7) and Eq. (8) is one option. Exactly the same results for R_{opt} are obtained from the (equivalent) SVD and optimum quaternion eigenvector methods for the 3D-3D EnP problem, with or without noise; only the `ArgMin` method obtains the true R_{opt} for the noisy OnP case.

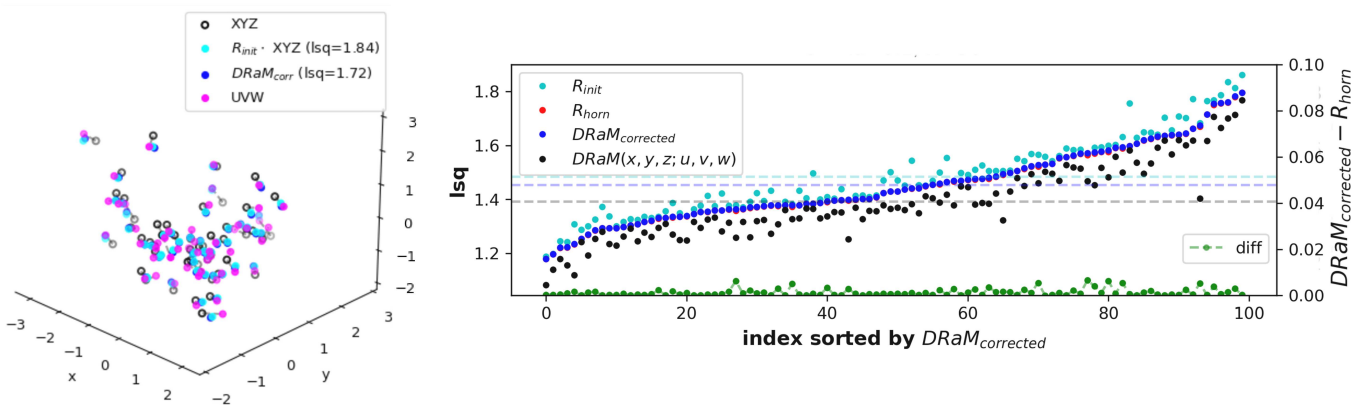

Fig. 1. Results for $\sigma = 0.1$, cloud data width ~ 2.0, using the DRaM analytical solution to the EnP 3D-to-3D alignment problem. The corrected EnP DRaM is not identical to the known-to-be-optimal SVD/Quaternion "Horn" solution, but it is very close, with 20X scaled differences plotted at the bottom.

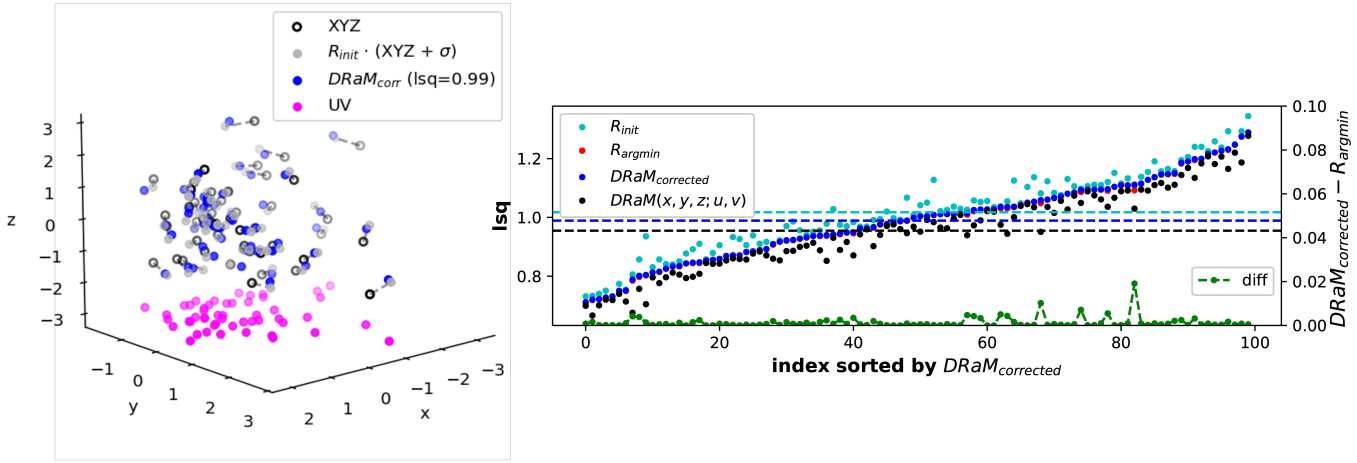

Fig. 2. Results for $\sigma = 0.1$, cloud data width ~ 2.0, using the DRaM analytical solution to the OnP 3D-to-2D orthographic projection alignment problem. The corrected OnP DRaM is not identical to the known-to-be-optimal `ArgMin` solution, but it is very close, with 20X scaled differences plotted at the bottom.

For error-free data, all four rotation choices are identical and when substituted into the loss functions are uniformly zero to machine accuracy, consistent with the DRaM matrices being exact least-squares loss minimizing solutions to the both pose estimation optimization problems.

In Fig. (1), we plot the values of the 3D-3D EnP least squares losses Eq. (7), for all four options we considered, sorting by the losses using optimal SVD/Quaternion rotation matrix values, which define the gold standard of comparison to other losses. A similar plot is shown in Fig. (2) for the 3D-2D OnP orthographic projection losses defined by Eq. (8) using numerical `ArgMin` rotation matrix values, which are the gold standard for this case. From top to bottom, the worst case is the R_{init} rotation, whose behavior is expected because, as we know from the SVD/Quaternion solution to the noisy 3D-3D alignment problem, there exists a closed form algebraic solution (the Cardano formula for the quaternion eigenvalues) for R_{opt}, while R_{init} cannot in any way be computed from noisy data. $R_{\text{BI:DRaM}}$ is next, plotted essentially on top of the R_{opt} losses because at this scale they are almost indistinguishable: the $R_{\text{BI:DRaM}}$ loss always exceeds or equals the R_{opt} loss, and this amount magnified by around 20X is plotted on the bottom axis. As we might expect, the unaltered R_{DRaM} matrix, which is *not a rotation matrix*, but remains a solution to the improper least-squares minimization problem, actually appears *below* the loss value plot for R_{opt}; this is meaningless because this DRaM matrix is not orthonormal — the smallest possible loss with an orthonormal matrix in the loss function is always the R_{opt} loss.

To address the sensitivity of each method to error in simulated data, we present in Fig. (3)(a,b) some simple studies of the properties of the various pose discovery methods, showing the sensitivity of the loss curve for a given pose rotation output to increasing error.

We can see that even when the DRaM class of pose extraction algorithms encounters significant errors, after rotation correction (noted as "BI" for Bar-Itzhack), they produce loss responses that are extremely close to the `ArgMin` gold standard pose estimates, though they are not exact, as are the members of the EnP class of solutions.

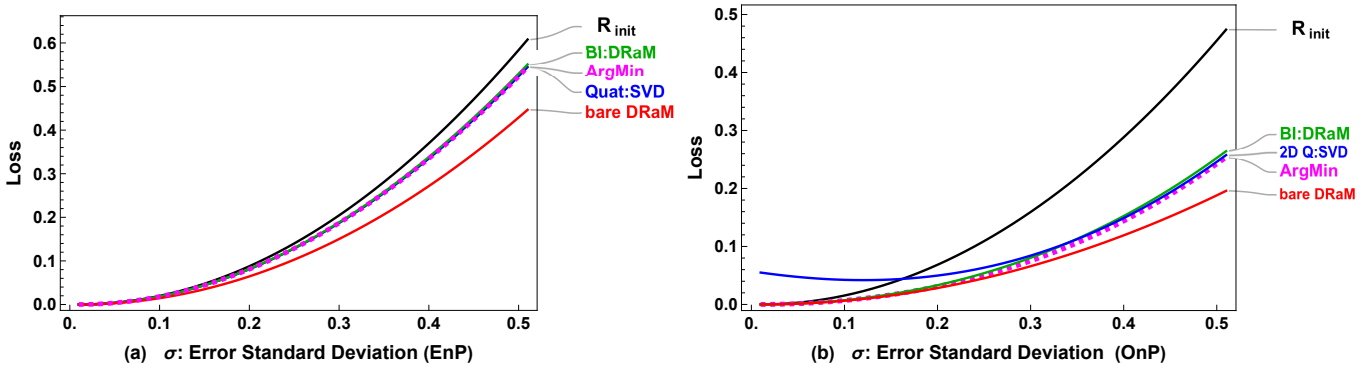

Fig. 3. (a) EnP with noise. Behavior of the EnP loss (Eq. (7)) as a function of standard deviation σ, data width ~ 2.0, for a selection of pose estimation methods. (b) OnP with noise. Behavior of the OnP loss (Eq. (8)) as a function of standard deviation σ for a selection of pose estimation methods. Here applying the SVD method with planar target data is highly inaccurate at $\sigma = 0$ (see Table I), but approaches the other methods for $sigma > 0.3$. For the EnP problem with or without noise, the quaternion-eigenvalue:SVD method is *exact*, and we see that it corresponds to machine accuracy with the `ArgMin` gold standard. Note that in both plots, the bare DRaM's loss is better than the gold standard `ArgMin` only because it is deformed and not actually a valid rotation matrix.

.

In summary, depending on the application, the corrected DRaM family candidates for the aligning rotation in 3D-3D and 3D-2D can be quite sufficient, and, while they are not necessarily superior to SVD/quaternion methods for the 3D-3D problem, they are much faster to compute for the 3D-2D problem than the available `ArgMin` method, which typically involves a Levenberg-Marquardt brute force numerical search of the quaternion rotation space. We have selected several properties of each method of the RMSD class and the DRaM class for comparison, namely speed, loss measured by the appropriate least squares formula, and total rotation difference relative to the `ArgMin` gold standard solution, and tabulate these all together in the comprehensive Table I for both exact and moderately noisy data. The results have many interesting features. For example, while typical times for 100,000 OnP pose discoveries are 2,000 seconds for `ArgMin` , for the same noisy-data OnP task, DRaM, QR, and PseudoInverse take about 6 seconds, 3 seconds, and 2 seconds, respectively. The numerical results for the three methods are identical, so it appears that the PseudoInverse would be the method of choice, in agreement with an observation of Hajder (2019). While the bare DRaM family encountering data with $\sigma = 0.1$ is quite inaccurate, the result after the Bar-Itzhack-style correction, with negligible expense, is typically within 3 degrees of the gold standard.

XI. Conclusion

The investigation we have described began with the unexpected discovery that the EnP (3D:3D cloud matching) and OnP (3D:2D orthographic cloud matching) tasks are solvable for exact data by determinant-ratio optimal rotation matrices, the DRaMs of Eqs. (11) and (13). Three distinct derivations of these simple algebraic equations were introduced, one based on solving the 3D least squares optimization of Eqs. (7) and (8) using the ten quaternion adjugate q_{ij} variables, the second achieving the same exact result using the nine orthogonal 3D rotation matrix variables r_{ij}, and the last completely independent of the least squares framework and valid not only for 3D space but any N-dimensional Euclidean space for both the EnP and OnP tasks. Further investigation revealed that there is an entire class of three algebraic methods with exactly the same properties as the DRaM function, with this "DRaM class" consisting of the DRaM, the QR decomposition mapping applied to the reference data array, and Moore-Penrose pseudoinverse applied to the reference data array. All members of the DRaM class produce perfect solutions to the EnP and OnP pose determination problems for perfect target data, and all members produce non-orthogonal warped rotation matrices for noisy target data, and yet all can be restored to highly accurate (but not exactly accurate) orthonormal matrices using rotation correction algorithms. We also enumerated and evaluated a complementary set of four algorithms, the minimal quaternion eigenvalue, the maximal quaternion eigenvalue, the SVD, and the HNN methods of the "RMSD class" that produce perfect solutions, matching the `ArgMin` "gold standard" value, for all EnP problems with either perfect or noisy data. However, no analogs to the RMSD class algorithms are known for the OnP problem, and the rotation-corrected DRaM class algorithms appear to be the best available closed-form methods known at this time.

While we have shown the power of the DRaM class of algorithms, a number of open questions remain. For example, we know that the 11^{th}-degree polynomial OnP solution of Hajder (2017) is an accurate numerical solution with results equivalent to the `ArgMin` method. Is it possible that the Hajder solution could reduce to the sought-for closed form OnP formula belonging to the RMSD class? Next, in addition to the DRaM-class of solutions we have presented for the orthogonal projection (OnP) problem, the application of the DRaM method to the six-degree-of-freedom PnP (perspective projection) problem, extending the incomplete PnP treatment in Hanson (2024), is known and will be described elsewhere. Finally, we can imagine that aspects of the framework we have presented here for the pose estimation problem might be effectively incorporated into the machine learning and deep neural network approaches to machine vision.

A	Time/10^5	loss	EnP (3D:3D) Exact Data $(\mathbf{X},\ \mathbf{Y} = R_{\text{init}} \cdot \mathbf{X})$	Time/10^5	loss, angle	OnP (3D:2D[Ortho]) Exact Data $(\mathbf{X},\ \mathbf{U} = P_{\text{init}} \cdot \mathbf{X})$
ArgMin Gold Std	2755.	0	$R_0 \Leftrightarrow \underset{q}{\text{argmin}} \lVert R(q)\cdot\mathbf{X} - \mathbf{Y}\rVert^2$	2054.	0, 0°	$R_0 \Leftrightarrow \underset{q}{\text{argmin}} \lVert P(q)\cdot\mathbf{X} - \mathbf{U}\rVert^2$
QMIN	13.46	0	$R_0 \Leftrightarrow q_{\text{Hebert}}\,(B(\mathbf{X},\mathbf{Y}))$	13.29	0.055, 37°	$R_* \Leftrightarrow q_{\text{Hebert Ortho}}\,(B(\mathbf{X},\mathbf{U}))$
QMAX	8.08	0	$R_0 \Leftrightarrow q_{\text{Horn}}\left(M(E = \mathbf{X}\cdot\mathbf{Y}^{\text{t}})\right)$	7.42	0.055, 37°	$R_* \Leftrightarrow q_{\text{Horn Ortho}}\,(M_{2\times3})$
SVD	3.06	0	$R_0 \Leftrightarrow \text{SVD}_{3\times3}\left(E = \mathbf{X}\cdot\mathbf{Y}^{\text{t}}\right)$	2.62	0.055, 37°	$R_* \Leftrightarrow \text{SVD}_{2\times3}\,(M_{2\times3})$
HHN	2.55	0	$R_0 \Leftrightarrow \text{HHN}\left(E = \mathbf{X}\cdot\mathbf{Y}^{\text{t}}\right)$	3.62	0.055, 37°	$R_* \Leftrightarrow \text{HHN}_{2\times3}\,(M_{2\times3})$
DRaM	5.81	0	$R_0 \Leftrightarrow \text{DRaM}\,(\mathbf{X},\ \mathbf{Y})$	4.84	0, 0°	$R_0 \Leftrightarrow \text{DRaM}\,(\mathbf{X},\ \mathbf{U})$
QR	1.70	0	$R_0 \Leftrightarrow \text{QR}\,(\mathbf{X},\ \mathbf{Y})$	1.34	0, 0°	$R_0 \Leftrightarrow \text{QR}\,(\mathbf{X},\ \mathbf{U})$
PINV	1.32	0	$R_0 \Leftrightarrow \text{PseudoInverse}\,(\mathbf{X})\cdot\mathbf{Y}$	1.00	0, 0°	$R_0 \Leftrightarrow \text{PseudoInverse}\,(\mathbf{X})\cdot\mathbf{U}$

B	Time/10^5	loss	angle	EnP (3D:3D) Errorful Data $(\mathbf{X},\ \widetilde{\mathbf{Y}} = R_{\text{init}}\cdot\mathbf{X} + \epsilon)$	Time/10^5, loss, angle	OnP (3D:2D[Ortho]) Errorful Data $(\mathbf{X},\ \widetilde{\mathbf{U}} = P_{\text{init}}\cdot\mathbf{X} + \epsilon)$
ArgMin Gold Std	2746.	0.0225	0°	$R_1 \Leftrightarrow \underset{q}{\text{argmin}} \left\lVert R(q)\cdot\mathbf{X} - \widetilde{\mathbf{Y}}\right\rVert^2$	2097., 0.0084, 0°	$R_1 \Leftrightarrow \underset{q}{\text{argmin}} \left\lVert P(q)\cdot\mathbf{X} - \widetilde{\mathbf{U}}\right\rVert^2$
QMIN	13.60	0.0225	0°	$R_1 \Leftrightarrow q_{\text{Hebert}}\left(B(\mathbf{X},\widetilde{\mathbf{Y}})\right)$	13.14, 0.042 , 31°	$R_2 \Leftrightarrow q_{\text{Hebert Ortho}}\left(B(\mathbf{X},\widetilde{\mathbf{U}})\right)$
QMAX	8.27	0.0225	0°	$R_1 \Leftrightarrow q_{\text{Horn}}\left(M(E = \mathbf{X}\cdot\widetilde{\mathbf{Y}^{\text{t}}})\right)$	7.38, 0.042, 31°	$R_2 \Leftrightarrow q_{\text{Horn Ortho}}\,(M_{2\times3})$
SVD	3.11	0.0225	0°	$R_1 \Leftrightarrow \text{SVD}_{3\times3}\left(E = \mathbf{X}\cdot\widetilde{\mathbf{Y}}^{\text{t}}\right)$	2.66, 0.042, 31°	$R_2 \Leftrightarrow \text{SVD}_{2\times3}\,(M_{2\times3})$
HHN	2.58	0.0225	0°	$R_1 \Leftrightarrow \text{HHN}\left(E = \mathbf{X}\cdot\widetilde{\mathbf{Y}^{\text{t}}}\right)$	3.65, 0.042, 31°	$R_2 \Leftrightarrow \text{HHN}_{2\times3}\,(M_{2\times3})$
DRaM	5.68	0.0227	1.42°	$R_3 \Leftrightarrow \text{RotOpt}[\text{DRaM}\left(\mathbf{X},\widetilde{\mathbf{Y}}\right)]$	6.80, 0.0089, 2.85°	$R_4 \Leftrightarrow \text{RotOpt}[\text{DRaM}\left(\mathbf{X},\widetilde{\mathbf{U}}\right)]$
QR	1.72	0.0227	1.42°	$R_3 \Leftrightarrow \text{RotOpt}[\text{QR}\left(\mathbf{X},\widetilde{\mathbf{Y}}\right)]$	3.16, 0.0089, 2.85°	$R_4 \Leftrightarrow \text{RotOpt}[\text{QR}\left(\mathbf{X},\widetilde{\mathbf{U}}\right)]$
PINV	1.33	0.0227	1.42°	$R_3 \Leftrightarrow \text{RotOpt}[\text{PseudoInv}\,(\mathbf{X})\cdot\widetilde{\mathbf{Y}}]$	2.79, 0.0089, 2.85°	$R_4 \Leftrightarrow \text{RotOpt}[\text{PseudoInv}\,(\mathbf{X})\cdot\widetilde{\mathbf{U}}]$

C	EnP (3D:3D) Rotation Correction (RotOpt)	OnP (3D:2D[Ortho]) Rotation Correction (RotOpt)
Bar-Itzhack	$= \text{QuatToRot}[\,q_{\text{Horn}}\left(M(E = R^{\text{t}}_{\text{approx}})\right)]$	$= \text{QuatToRot}[\,q_{\text{Horn}}\left(M_{2\times3}(E = P^{\text{t}}_{\text{approx}})\right)]$
SVD	$= \text{SVD}_{3\times3}\left(R^{\text{t}}_{\text{approx}}\right)$	$= \text{SVD}_{2\times3}\left(P^{\text{t}}_{\text{approx}}\right)$

TABLE I
SUMMARY OF EnP AND OnP POSE DISCOVERY ALGORITHM PROPERTIES FOR EXACT AND ERRORFUL DATA.

In these tables we present eight algorithms that have been used to produce rotation matrices aligning reference point clouds with observed rotated target data. The two major categories, EnP and OnP, are supplied with exact data in Table **A**, and errorful data with standard deviation $\sigma = 0.1$ in Table **B**. Table **C** lists typical rotation correction variants of the algorithms. We use a single reference data set consisting of an 8 point random cloud with average radius one unit, and a 21° rotation of the target data, orthographically projected for the OnP case, with the intent of showing a typical behavior rather than an extensive collection of cases. Times are in seconds for 100,000 repetitions of the calculation in Mathematica, on an Apple M4 Max processor, transforming the input reference and target point arrays to a rotation matrix pose estimation. Accuracy is measured by two redundant but useful measures, the actual loss function specified in the top `ArgMin` entry, and the deviation from the `ArgMin` gold standard angle of each rotation matrix's angle θ obtained from its quaternion $q_0 = \cos(\theta/2)$. The details of the algorithms and programs used are given in the main text and detailed in the Supplementary Material. One should note that the SVD-like algorithms naturally employ input that is transformed from the n-dimensional data arrays into compact 3×3 or 2×3 cross-covariance matrices, while such data is insufficient for the DRaM family of algorithms, which require the reference data ($\mathbf{X}$ in the table) to be supplied separately from the target data ($\mathbf{Y}$ and $\mathbf{U}$ in the table). For OnP, even with exact data, the RMSD class algorithms fail to be exact matches to the `ArgMin` gold standards because the corresponding cross-covariances, which contain all the needed data for EnP, are missing a component of the loss corresponding to non-canceling rotation matrix terms.

ACKNOWLEDGMENTS

This project has evolved over a period of several years, and we benefited from feedback and suggestions from many colleagues, whose input we thankfully acknowledge. We are particularly grateful to B.K.P. Horn, who has suggested readings, provided guidance, and has always been ready to return an answer to a question. AJH also thanks David Crandall for his collegiality and willingness to take the time to help in clarifying some issues. We are especially appreciative of several referees who, in rejecting an earlier version of this work, provided pointers to exactly the references we needed to go further and significantly improve our understanding of the problem as whole. SMH gratefully acknowledges the support of The Flatiron Institute, a division of the Simons Foundation.

REFERENCES

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3D point sets. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-9(5):698–700, 1987.

I. Bar-Itzhack. Iterative optimal orthogonalization of the strapdown matrix. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(1):30–37, 1975.

Itzhack Y. Bar-Itzhack. New method for extracting the quaternion from a rotation matrix. *Journal of Guidance, Control, and Dynamics*, 23(6):1085–1087, 2000.

Å. Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM J. Numer. Anal.*, 8(2):358–364, 1971.

N. Cliff. Orthogonal rotation to congruence. *Psychometrika*, 31:33–42, 1966.

E. A. Coutsias, C. Seok, and K. A. Dill. Using quaternions to calculate RMSD. *J Comput Chem.*, 25(15):1849–1857, 2004.

P.B. Davenport. A vector approach to the algebra of rotations with applications. Technical Report TN D-4696, NASA: Goddard Space Flight Center, Greenbelt, Maryland, 1968.

Daniel F. Dementhon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

R. Diamond. A note on the rotational superposition problem. *Acta Crystallogr.*, A44:211–216, 1988.

Olivier Faugeras and Martial Hebert. A 3D recognition and positioning algorithm using geometrical constraints between primitive surfaces. In *Proc. 8th Joint Conf. on Artificial Intell.*, pages 996–1002. Morgan Kaufmann, 1983.

Olivier Faugeras and Martial Hebert. The representation, recognition, and locating of 3D objects. *International Journal of Robotic Research (IJRR)*, 5:27–52, 1986.

D. R. Flower. Rotational superposition: a review of methods. *J. Mol. Graph. Model.*, 17:238–244, 1999.

Wolfgang Förstner. On estimating rotations. In *Festschrift für Prof. Dr.-Ing. Heinrich Ebner zum 60. Geburtstag*, page 12. Lehrstuhl für Photogrammetrie und Fernerkundung, Technische Universität München, 1999.

C. R. Giardina, R. Bronson, and L. Wallen. An optimal normalization scheme. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(4):443–446, 1975.

G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1st edition, 1983. Sec 12.4.

Bert F. Green. The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, 17:429–440, 1952.

Levente Hajder. W-pnp method: Optimal solution for the weak-perspective n-point problem and its application to structure from motion. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 6: VISAPP, (VISIGRAPP 2017)*, pages 265–276. INSTICC, SciTePress, 2017.

Levente Hajder. Weak-perspective and scaled-orthographic structure from motion with missing data. In *Computer Vision, Imaging and Computer Graphics – Theory and Applications*, pages 128–153, Cham, 2019. Springer International Publishing.

Andrew J. Hanson. *Visualizing Quaternions*. Morgan-Kaufmann/Elsevier, 2006.

Andrew J. Hanson. The quaternion-based spatial-coordinate and orientation-frame alignment problems. *Acta Crystallographica Section A*, 76(4):432–457, 2020.

Andrew J. Hanson. *Visualizing More Quaternions*. Morgan-Kaufmann/Elsevier, 2024.

Andrew J. Hanson. In preparation, 2026.

A. J. Hanson and S. M. Hanson. Exploring the adjugate matrix approach to quaternion pose extraction. https://arxiv.org/pdf/2205.09116.pdf, 2022.

N. J. Higham. Computing the polar decomposition – with applications. *SIAM J. Sci. Stat. Comput.*, 7(4):1160–1174, 1986.

B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4:629–642, 1987.

Berthold K. P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am. A*, 5(7):1127–1135, 1988.

W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr.*, A32:922–923, 1976.

W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallogr.*, A34:827–828, 1978.

S. K. Kearsley. On the orthogonal transformation used for structural comparisons. *Acta Crystallogr.*, A45(2):208–210, 1989.

Gerald R. Kneller. Superposition of molecular structures using quaternions. *Molecular Simulation*, 7(1–2):113–119, 1991.

Chen Lin, Andrew J. Hanson, and Sonya M. Hanson. Algebraically rigorous quaternion framework for the neural network pose estimation problem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14097–14106, 2023.

Chen Lin, Weizhi Du, Zhixiang Min, Baochen She, Enrique Dunn, and Sonya M. Hanson. DRaM-LHM: A quaternion framework for iterative camera pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages –, 2025.

P. Liu, D. K. Agrafiotis, and D. L. Theobald. Fast determination of the optimal rotational matrix for macromolecular superpositions. *J. Comput. Chem.*, 31:1561–1563, 2010.

F. L. Markley. Attitude determination using vector observations and the singular value decomposition. *Journal of the Astronautical Sciences*, 38(2):245–258, 1988.

Fernando Sansò. An exact solution of the roto-translation problem. *Photogrammetria*, 29(6):203–216, 1973.

Soheil Sarabandi and Federico Thomas. On closed-form solutions to the 4D nearest rotation matrix problem. *Mathematical Methods in the Applied Sciences*, "Special Issue": 1–9, 2022.

Soheil Sarabandi, Alba Perez-Gracia, and Federico Thomas. Singularity-free computation of quaternions from rotation matrices in $\mathbb{E}^4$ and $\mathbb{E}^3$. In *Conference on Applied Geometric*

Algebras in Computer Science and Engineering, pages 23–27, 2018.

Soheil Sarabandi, Arya Shabani, Josep M. Porta, and Federico Thomas. On closed-form formulas for the 3D nearest rotation matrix problem. *IEEE Transactions on Robotics*, 36(4):1333–1339, 2020.

P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31:1– 10, 1966.

G. H. Schut. On exact linear equations for the computation of the rotational elements of absolute orientation. *Photogrammetria*, 17(1):34–37, 1960.

S. W. Shepperd. Quaternion from rotation matrix. *Journal of Guidance and Control*, 1(3):223–224, 1978.

M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4(1):70–77, 1981.

Carsten Steger. Algorithms for the orthographic-n-point problem. *Journal of Mathematical Imaging and Vision*, 60(2): 246–266, 2018.

Douglas Theobald. Rapid calculation of RMSDs using a quaternion-based characteristic polynomial. *Acta Crystallogr.*, A61:478–480, 2005.

E. H. Thompson. An exact linear solution of the problem of absolute orientation. *Photogrammetria*, 15(4):163–179, 1958.

G. Wahba. Problem 65-1, A least squares estimate of spacecraft attitude. *SIAM Review*, 7(3):409, 1965.

Heng Yang, Chris Doran, and Jean-Jacques Slotine. Dynamical pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5906–5915, 2021.

APPENDIX A
DETAILS OF THE QUATERNION-BASED DRAM SOLUTION

In this Appendix, we fill in the details of the process of solving the EnP and OnP least squares problems by exploiting the quaternion adjugate form. In Eq. (2), we defined the 10 quaternion adjugate variables, $q_{ij} = q_i q_j$, whose advantages were introduced in Hanson and Hanson (2022); Lin et al. (2023); Hanson (2024); Lin et al. (2025). Here we review the essential elements, observing, as noted in Section IV, that the quaternion adjugate variables q_{ij} are equivalent to the rotation matrix variables $[R]_{ij} = r_{ij}$, but with very different constraint structure. In fact, the following derivations can be duplicated using the r_{ij} variables in Eq. (7) or Eq. (8) with no additionally imposed constraints, a fact we had not yet discovered when we developed the derivations in this Appendix, which require the use of one or more of the quaternion adjugate constraints. As also observed in Section IV, both the r_{ij} and q_{ij} derivations using explicit solutions of the loss-function minimization algebra are essentially irrelevant since, with hindsight, the DRaM matrix forms can be proven by hand on the back of an envelope to solve the EnP and OnP pose discovery problem for exact data.

We begin the algebraic loss function minimization process with the observation that if we replace the rotation in the 3D-3D least squares expression Eq. (7) by the quaternion adjugate expression $R(q_{ij})$ shown in Eq. (3), the optimization problem becomes quadratic in q_{ij} and therefore potentially more easily solvable in terms of the cross-covariances of the $\{\mathbf{x}_k\}$ and $\{\mathbf{y}_k\}$ appearing in Eq. (9) and Eq. (10) above. We remind ourselves that if we could constrain an algebraic solver to impose all seven adjugate constraints Eq. (4), we would have an optimal solution of the least squares problems Eq. (7) and Eq. (8) for both exact and errorful data. Unfortunately, the equations with all the constraints imposed appears to resist closed-form solution. However, if we write out the expression as a quadratic polynomial,

$$S_{\text{EnP}}(q_{ij}; xx, xy, xz, \ldots; ux, vx, wx, \ldots) = {q_{00}}^2 f(xx, \ldots) + \cdots + q_{00}\, g(xx, \ldots) + \cdots \quad ,$$

and try a variety of combinations of vanishing derivative conditions on $S_{\text{EnP}}(q_{ij})$ with selections of the constraints Eq. (4), we find one that actually works. Since the constraints imply that the derivatives are not independent, and since the only true guarantee of a solution is to require *all* the constraints to be enforced, the fact that we found a combination that is *perfect* for exact data is perhaps unexpected; if there is a deep reason for why this succeeded at all, we have not yet understood it. The successful choice is to treat vanishing of *all ten* derivatives of $S_{\text{EnP}}(q_{ij})$ with respect to q_{ij} as independent conditions, while imposing only the *first* constraint in Eq. (4), namely $q_{00} + q_{11} + q_{22} + q_{33} = 1$. Assembling this into the Mathematica constraint-list solver in the form

```
the3D3DAdjMatchSolns =
  Module[{eqn = S_EnP(q_ij)},
   Solve[ {
     D[eqn, q00] == 0, D[eqn, q11] == 0,
     D[eqn, q22] == 0, D[eqn, q33] == 0,
     D[eqn, q01] == 0, D[eqn, q02] == 0
     D[eqn, q03] == 0, D[eqn, q23] == 0,
     D[eqn, q13] == 0, D[eqn, q12] == 0
              q00 + q11 + q22 + q33 == 1},
    {q00, q11, q22, q33, q01, q02, q03,
                         q23, q13, q12}]].
```

returns a complete list of solutions for the $q_{ij}(\text{xx}, \ldots)$ in precisely the form of Eq. (9) and Eq. (10), combining into the EnP DRaM expression Eq. (11) when assembled to construct the rotation using the components in Eq. (3). The EnP solution takes 0.5 seconds to solve and 9 seconds to assemble the q_{ij} into a rotation and simplify to the DRaM form. As noted in the previous section, any set of error-free data that is regular with sufficient degrees of freedom produces *exactly* the correct rotation matrix applied to the reference data when the reference and target data are applied to the DRaM formula Eq. (11), but errorful data disrupts the orthonormality of R_{opt}. Applying the Bar-Itzhack or equivalent SVD rotation correction to the deformed rotation produces an improved, and orthonormal, rotation matrix that deviates from the gold standard minimizing rotation, but is very close, as we see in Table I.

The OnP exact-data solution, in hindsight, is just the the top two lines of the EnP solution, or equivalently, what is left when the data for the "w′′ components are set to zero. However, we did actually solve the OnP case *before* we looked at the simpler EnP case, and the details of that solution are amusing, if not particularly essential, so we add an outline of that process here for completeness. Starting with the $R(q_{ij})$ inserted into the OnP loss Eq. (8), and computing the corresponding algebraic function having only $(\text{ux}, \ldots)$ and $(\text{vx}, \ldots)$ components, but no $(\text{wx}, \ldots)$ components,

$$S_{\text{OnP}}(q_{ij}; xx, xy, xz, \ldots; ux, vx, , \ldots) \ ,$$

the question is what combination of derivatives of $S_{\text{OnP}}(q_{ij})$ and the constraints Eq. (4) is required to produce the DRaM solution equal to the top two rows of Eq. (11), The answer, again following from extensive experimentation with the possible candidates, was again to include all 10 derivatives with respect to q_{ij}, despite the issue of cross-dependencies, and to impose the *first four* of the constraints on the q_{ij} given in Eq. (4), that is, all the constraints containing q_{00}, and not just the $q_{00} + q_{11} + q_{22} + q_{33} = 1$ constraint that sufficed for EnP (any set containing any single q_{ii} would work as well). The resulting Mathematica solver code becomes

```
the3D2DAdjMatchSolns =
 Module[{eqn = S_OnP(q_ij)},
  Solve[ {
    D[eqn, q00] == 0, D[eqn, q11] == 0,
    D[eqn, q22] == 0, D[eqn, q33] == 0,
    D[eqn, q01] == 0, D[eqn, q02] == 0,
    D[eqn, q03] == 0, D[eqn, q23] == 0,
    D[eqn, q13] == 0, D[eqn, q12] == 0,
          q00 + q11 + q22 + q33   == 1,
             q00 q11 == q01 q01,
             q00 q22 == q02 q02,
             q00 q33 == q03 q03 },
  {q00, q11, q22, q33, q01, q02, q03,
                       q23, q13, q12}]] .
```

The solver produces a solution that is a single list of the 10 q_{ij} rules as ratios of the cross-covariances with square-root containing algebraic expressions up to 9MB in length. However, substituting the solution values into Eq. (3), we find that the first two rows, the only ones we *used* in the OnP loss function, simplify into the OnP DRaM solutions corresponding to the top two rows of Eq. (11). The huge expressions containing enormous square roots have large parts that simply cancel out. In the 3rd row of Eq. (3), the enormous parts *add up*, and that row is nonsense, as we might expect. However, as noted in Eq. (12), the (error-free) cross-product of the top two rows converts into an elegant DRaM expression for the third row, completing a full 3×3 orthonormal rotation matrix determining the pose from exact OnP data, while the usual issues are again present for noisy data. We observed that the OnP solution takes 7 seconds to solve and 33 seconds to assemble the q_{ij} into a 2×3 rotation in the DRaM orthographic form.

APPENDIX B
MATHEMATICA ALGORITHMS FOR THE DRAM METHOD

A. Quaternion Manipulation Utilities

```
  qqDiff[qtest_, qref_] := (2 ArcCos[Min[1.,Abs[qtest.qref]]])/Degree

  frobDiff[mat1_, mat2_] := Tr[(mat1-mat2).Transpose[mat1-mat2]]

  makeq0plus[q_]:=If[q[[1]]<0,-q,q,q]

  pick4DQAdj[adjugate_]:=
      Module[
         {kmax=Position[Abs/@Diagonal[adjugate],Max[Abs/@Diagonal[adjugate]]][[1,1]]},
            Normalize[adjugate[[kmax]]]]

qrotsym = { { q0^2+q1^2-q2^2-q3^2,  2 q1 q2-2 q0 q3,  2 q0 q2+2 q1 q3},
            { 2 q1 q2+2 q0 q3,  q0^2-q1^2+q2^2-q3^2,  -2 q0 q1+2 q2 q3},
            { -2 q0 q2+2 q1 q3,  2 q0 q1+2 q2 q3,  q0^2-q1^2-q2^2+q3^2} }

qqrotsym =  {  q00+q11-q22-q33,  -2 q03 + 2 q12,  2 q02 + 2 q13},
            {  2 q03 + 2 q12,  q00-q11+q22-q33,  -2 q01 + 2 q23},
            {  -2 q02 + 2 q13, 2 q01 + 2 q23,  q00-q11-q22+q33}}

QuatToRot[{q0_,q1_,q2_,q3_}] :=
    { {q0^2+q1^2-q2^2-q3^2, 2 q1 q2 - 2 q0 q3, 2 q0 q2 + 2 q1 q3},
      {2 q1 q2 + 2 q0 q3, q0^2-q1^2+q2^2-q3^2, -2 q0 q1+ 2 q2 q3},
      {-2 q0 q2 + 2 q1 q3, 2 q0 q1 + 2 q2 q3, q0^2-q1^2-q2^2+q3^2} }

RotToQuat[rot33_] :=   Module[{eig,adjugate,mMat44},
     mMat44=makeMxxmat[Transpose[rot33]];
     eig=Max[Eigenvalues[mMat44]];
     adjugate=Adjugate[mMat44-eig IdentityMatrix[4]];
     makeq0Plus[pick4DQAdj[adjugate]]]
```

B. Data Simulation

```
sigmaSNRFun[tz_,SNR_:70.,w_:2.]:=(Exp[-(SNR/20.)] w)/tz
   (*  For PnP, if tz = 6, SNR 70,  then sigma = 0.01. *)

SeedRandom[1357] ;
RandomReal[{-1, 1}] (*  = -0.75708 (EVALUATE THIS to check Random initialization.) *)

pquat1 = Module[{th = 21.5 Degree, nhat = Normalize[{1.,2.,4.}]},
               Join[{Cos[th/2]},Sin[th/2] nhat]]

prot1 = QuatToRot[pquat1] ;
(* %  Check that initial rotation is reproduceable:
%\begin{array}{ccc}
% 0.933731 & -0.313282 & 0.173208 \\
% 0.326535 & 0.943671 & -0.0534695 \\
% -0.1467 & 0.106485 & 0.983433 \\
%\end{array}
   *)

(*  Settable variables  s = std deviation, camera displacement {tx,ty,tz}  *)

{ pcloud1, protcloud1, protcloudErr1, protTcloud1,
            portho1, porthoErr1, portho3DErr1  pproj1, pprojErr1} =

Module[{npts = 8, rot = prot1, orthoZ=1.0, cloud, cm,
       rotcloud, protcloudErr, rotTcloud , ortho, orthoErr, ortho3DErr, proj, projErr},
cloud = RandomReal[{-1,1},{npts,3}];
cm = Mean[cloud];
cloud=(#1-cm&)/@cloud;
rotcloud = (rot.#)&/@cloud;
rotcloudErr  =(s*RandomVariate[NormalDistribution[0,1],{3}]+#)&/@rotcloud;
rotTcloud = Table[(rotcloud[[k]] + {tx,ty,tz}),{k,1,npts}] ;
ortho  = (#[[1;;2]]&)/@rotcloud;
orthoErr =(s*RandomVariate[NormalDistribution[0,1],{2}]+#)&/@ortho;
ortho3DErr ={#[[1]],#[[2]],orthoZ}&/@orthoErr;
proj  =Table[rotTcloud[[k]][[1;;2]]/rotTcloud[[k]][[3]],{k,1,npts}] ;
projErr = (s*RandomVariate[NormalDistribution[0,1],{2}]+#)&/@proj;
{ cloud, rotcloud, rotcloudErr, rotTcloud, ortho, orthoErr, ortho3DErr, proj, projErr}];
```

C. Profile matrices for quaternion eigensystem.

```
(* For the Max eigenvalue quaternion eigensystem characteristic equation. *)
    makeMxxmat[Exxmat_] :=
         Module[{ Exx = Exxmat[[1, 1]], Exy = Exxmat[[1, 2]],  Exz = Exxmat[[1, 3]],
                  Eyx = Exxmat[[2, 1]], Eyy = Exxmat[[2, 2]],  Eyz = Exxmat[[2, 3]],
                  Ezx = Exxmat[[3, 1]], Ezy = Exxmat[[3, 2]],  Ezz = Exxmat[[3, 3]]},
    {{Exx + Eyy + Ezz,       Eyz - Ezy,        Ezx - Exz,         Exy - Eyx},
    {    Eyz - Ezy,     Exx - Eyy - Ezz,      Exy + Eyx,         Ezx + Exz},
    {    Ezx - Exz,         Exy + Eyx,    -Exx + Eyy - Ezz,      Eyz + Ezy},
    {    Exy - Eyx,         Ezx + Exz,        Eyz + Ezy,     -Exx - Eyy + Ezz}}]

(* Bar-Itzhack Method I: partial rotation variation on the characterisic equation. *)
 makeMxx23mat[Exx23mat_]  :=
    Module[{ Exx=Exx23mat[[1,1]], Exy=Exx23mat[[1,2]],  Exz=Exx23mat[[1,3]],
                  Eyx=Exx23mat[[2,1]], Eyy=Exx23mat[[2,2]],  Eyz=Exx23mat[[2,3]]},
        {{Exx+Eyy,     Eyz,     -Exz,     Exy-Eyx},
         {  Eyz,     Exx-Eyy,   Exy+Eyx,    Exz},
         { -Exz,     Exy+Eyx,  -Exx+Eyy,    Eyz},
         {Exy-Eyx,     Exz,      Eyz,    -Exx-Eyy}}]

 (* For the Min eigenvalue quaternion eigensystem characteristic equation. *)
  (* (Atranspose . A)  summed over 'i' *)
buildFullEnPBmat[testVecs_,targetVecs_ ] :=
       Module[{npts = Length[testVecs],ithAMat,
         aMatFun = Function[{ x,y,z,xx,yy,zz},
                       {{0,-x +xx ,-y+yy,-z+zz},
                        {x-xx,0,z+zz,-y-yy},
                        {y-yy,-z-zz,0,x+xx},
                        {z-zz,y+yy,-x-xx,0}}] },
        Sum[ ithAMat =
                 aMatFun[ testVecs[[i,1]],testVecs[[i,2]],testVecs[[i,3]],
                          targetVecs[[i,1]],targetVecs[[i,2]],targetVecs[[i,3]]];
               Transpose[ithAMat] . ithAMat,
                 {i,1,npts}]]
```

D. The EnP, OnP, and PnP loss functions

```
  (* For ArgMin, standard  RMSD Loss  {EnP)  *)
loss3DRMSDFunction [rot_,cloud_, rotcloud_] :=
     Module[{npts= Length[cloud],x,y,z,u,v,w,term },
        1./npts Sum[ {x,y,z} = cloud[[k]];    {u,v,w} = rotcloud[[k]];
                              term =   rot . {x,y,z}  - {u,v,w};
                              term . term,
                                 {k,1,npts}]]

(* For ArgMin, standard Ortho Loss  {OnP) *)
loss3DOrthoFunction [rot_,cloud_,ortho_ ] :=
     Module[{npts= Length[cloud], p23rot = rot[[1;;2]], x,y,z,u,v,term},
            1./npts Sum[  x,y,z} = cloud[[k]];       {u,v} = ortho[[k]];
                                  term =  p23rot . {x,y,z}  - {u,v};
                                  term . term,
                                     {k,1,npts}]]

(* For ArgMin, standard z=1 projected image Loss (PnP), given rot and tx,ty,tz  *)
lossDiv3DwTFunction [rot_, t3_, cloud_, img_ ] :=
 Module[{npts = Length[cloud], p23rot = rot[[1 ;; 2]],   denrot = rot[[3]],
                       x, y, z, u, v, term, txy, tz },
  txy = t3[[1 ;; 2]]; tz = t3[[3]];
  1./npts Sum[{x, y, z} = cloud[[k]]; {u, v} = img[[k]];
                   term =  (p23rot . {x, y, z} + txy)/( denrot . {x, y, z} + tz ) - {u, v};
                   term . term,
                         {k, 1, npts}]]
```

E. The Gold Standards: ArgMin is the Basis of Comparison

```
 (* Supplied: the least squares loss functions: and quaternion form R(q) :
      qrotsym =
      { {q0^2+q1^2-q2^2-q3^2,2 q1 q2-2 q0 q3,2 q0 q2+2 q1 q3},
       {2 q1 q2+2 q0 q3,q0^2-q1^2+q2^2-q3^2,-2 q0 q1+2 q2 q3},
      {-2 q0 q2+2 q1 q3,2 q0 q1+2 q2 q3,q0^2-q1^2-q2^2+q3^2} } *)

  (* EnP  and OnP  and PnP  least squares ArgMin  pose estimates *)

 getRMSDArgMin[cloud_,rotcloud_ ]:= Module[{ ruleQ, quat, rot },
           ruleQ = FindMinimum[{loss3DRMSDFunction[qrotsym, cloud, rotcloud_],
                           q0^2+q1^2+q2^2+q3^2 ==1}, {{q0,1},{q1,0},{q2,0},{q3,0}},
                           AccuracyGoal -> 12]   [[2]];
           quat = {q0,q1,q2,q3}/.ruleQ;
           rot = QuatToRot[quat];
           {makeq0Plus[quat], rot} ]

 getOrthoArgMin[cloud_,ortho_ ] := Module[{ ruleQ, quat, rot },
           ruleQ = FindMinimum[{loss3DOrthoFunction[qrotsym, cloud, ortho ],
                           q0^2+q1^2+q2^2+q3^2 ==1},{{q0,1},{q1,0},{q2,0},{q3,0}},
                            AccuracyGoal -> 12]  [[2]];
           quat = {q0,q1,q2,q3}/.ruleQ;
           rot = QuatToRot[quat];
           {makeq0Plus[quat], rot} ]

(* Note: typical  projTS ->
           pprojErr1 /.{tx->0,ty->0,tz->6.0, s->sigma or n*lhmSigmaFun[tz,70.,w] *)

 getProjArgMin[cloud_,projTS_] :=   Module[{loss,ruleTR,trans,quat,rot},
      {loss, ruleTR }= FindMinimum[
                 {lossDiv3DwTFunction[qrotsym,{tx,ty,tz},cloud,projTS],
                      q0^2+q1^2+q2^2+q3^2 ==1},
                        {{q0, 1}, {q1, 0}, {q2, 0}, {q3, 0}, {tx, 2}, {ty, 3}, {tz, 7}},
                         AccuracyGoal -> 12];
           trans= {tx,ty,tz}/.ruleTR;
           quat = {q0,q1,q2,q3}/.ruleTR;
           rot = QuatToRot[quat];
           {makeq0Plus[quat], rot, trans}]
```

F. Rotation Correction

```
  (*  ROTATION EXTENSION:  2 x 3  -> 3 x 3.  Extend top two perfect rows of a
       rotation matrix to 3x3 orthonormal matrix. *)

  getRotFrom2x3[P2x3_] := Module[{cross = Normalize[ Cross[P2x3[[1]],P2x3[[2]]] ]},
              Append[P2x3, cross]]

 (* ROTATION REFINEMENT:  Methods that obtain a full orthonormal rotation matrix
       from any  3 x 3 or  2x3 matrix approximating a rotation.
       Bar-Itzhack eigensystem method 1,3 and equivalent square 3D SVD.
       Bar-Itzhack eigensystem method 2,3 and equivalent rectangular SVD(2x3)
     NOTE:  The algorithms match  A.Transpose[B],  so if a rotation matrix is
       input, its TRANSPOSE is processed in the algorithm, which means
       these functions are not the same as the corresponding Pose Estimators. *)

getBestQuatRot[Rapprox_] := Module[{Rmat, Mmat, eigopt, quat},
       Rmat = Transpose[Rapprox];
       Mmat = makeMxxmat[Rmat],
       eigopt = Max[Eigenvalues[Mmat]];
       quat = pick4DQAdj[Adjugate[Mmat - eigopt IdentityMatrix[4]]];
       QuatToRot[quat]]

getBestSVDRot[Rapprox] :=  Module[{ Rmat , uu, ss, vv, dd},
     Rmat = Transpose[Rapprox];
    { uu, ss, vv} = SingularValueDecomposition[Rmat];
     dd = DiagonalMatrix[
    Table[If[i == dim, Sign[Det[uu] Det[vv]], 1], {i, 1, dim}]];
     vv . dd . Transpose[uu]]

  (*  Best rotation for 2 x 3  Matrices: Bar-Itzhack Method I  and 2 x 3 singular values;
       Note that both output the TRANSPOSE of the result in a slightly different
       way than the transposed rotation is handled for 3x3 above. *)

getHorn23Rot[R23approx_] :=
 Module[{Mmat = makeMxx23mat[R23approx], eigopt, quat},
   eigopt = Max[Eigenvalues[Mmat]];
  quat = pick4DQAdj[Adjugate[Mmat - eigopt IdentityMatrix[4]]];
  Transpose[QuatToRot[quat]]]

getSVD23Rot[R23approx_] := Module[{P23=R23approx[[1;;2]],uu,vv,ss,dd,r1,r2,r3,rotOpt},
        {uu,ss,vv}=SingularValueDecomposition[P23];
        dd={{1,0,0},{0,1,0}};
        {r1,r2}=uu.dd.Transpose[vv];   (* Transpose of standard SVD output *)
        r3=Normalize[ Cross[r1, r2];
        rotOpt={r1,r2,r3}]
```

G. The List of RMSD-class EnP Pose Discovery Functions

These algorithms produce agreement with the `ArgMin` Gold Standard for EnP data *with or without noise*.

```
getRMSDArgMin[cloud_, rotcloud_ ] :=  Module[{loss,ruleQ, quat, rot},
  {loss, ruleQ} = FindMinimum[{loss3DRMSDFunction[qrotsym, cloud, rotcloud],
         q0^2 + q1^2 + q2^2 + q3^2 == 1}, {{q0, 1}, {q1, 0}, {q2, 0}, {q3, 0}},
        AccuracyGoal -> 12] ;
        quat = {q0, q1, q2, q3} /. ruleQ;
       {makeq0Plus[quat],  QuatToRot[quat]}]

getHebertRotXY[cloud_, target_] := Module[{ Bmat, eigopt, quat},
       Bmat = buildFullEnPBmat[cloud, target];
      eigopt = Chop[ Min[Eigenvalues[Bmat]]];
      quat = pick4DQAdj[Adjugate[Bmat - eigopt IdentityMatrix[4]]];
      QuatToRot[quat]]

getHornRotXY[ cloud_, target_] :=  Module[{eigopt, quat, adjugate, eMat33, mMat44},
       eMat33 = Transpose[cloud] . target;
       mMat44 = makeMxxmat[eMat33];
       eigopt = Max[Eigenvalues[mMat44]];
       adjugate = Adjugate[mMat44 - eigopt*IdentityMatrix[4]];
        (* The pick4DQAdj[ ]  normalizes *)
        quat = makeq0Plus[ pick4DQAdj[adjugate] ] ;
        QuatToRot[quat]]

getSVDRotXY[cloud_, target_] := Module[{uu, ss, vv, dd, eMat, dim},
        eMat = Transpose[cloud] . target;  dim = Dimensions[eMat][[1]];
        {uu, ss, vv} = SingularValueDecomposition[eMat];
       dd = DiagonalMatrix[Table[If[i == dim, Sign[Det[uu] Det[vv]], 1], {i, 1, dim}]];
       vv . dd . Transpose[uu]]

 (* HNN has two options: "Plus root" and "Minus root":
     This First version needs inverse of  eMat, which is a problem if Det[eMat]=0 *)
getHHNRotPXY[cloud_, target_] := Module[{eMat, rootEtE, rotInv, rotOpt},
         eMat = Transpose[cloud] . target ;
         rotInv = Inverse[eMat];
          rootEtE = MatrixPower[Transpose[eMat] . eMat, +1/2];
          rotOpt = rootEtE . rotInv]

(* Our Preferred version does not need inverse of eMat, just inverse of
        the S factor and a Transpose. *)
getHHNRotMXY[cloud_, target_] := Module[{invRootEtE, rotOpt, eMat},
         eMat = Transpose[cloud] . target ;
         invRootEtE = MatrixPower[Transpose[eMat] . eMat, -1/2];
         rotOpt = invRootEtE . Transpose[eMat]]
```

H. Alternative forms with single cross-covariance matrix argument.

```
(* Native form of quaternion eigenvalue system, corresponding to
    getHornRotXY[cloud, target]   above. *)

getHornRotEmat [Emat_] :=   Module[{Mmat, eigopt, quat},
       Mmat = makeMxxmat[Emat];
       eigopt = Max[Eigenvalues[Mmat]];
       quat = pick4DQAdj[Adjugate[Mmat - eigopt IdentityMatrix[4]]];
       QuatToRot[quat]]

(*  Native form of singular value decomposition system, corresponding to
       getSVDRotXY[cloud, target] above. *)

getSVDRotEmat[eMat_] := Module[{dim =Dimensions[eMat][[1]], uu,ss,vv,dd},
          { uu,ss,vv}=SingularValueDecomposition[eMat];
           dd = DiagonalMatrix[Table[If[i==dim,Sign[Det[uu]Det[vv]],1],{i,1,dim}]];
           vv.dd.Transpose[uu]]

(*  Native form of HNN matrix algorithm, corresponding to
     getHHNRotPXY[cloud_ target] and  getHHNRotMXY[cloud, target]  above.

 (* HNN  Emat  has two options: Plus 1/2 root and Minus 1/2 root *)

(*  HNN "Plus  root":This First version needs inverse of eMat,
which is a problem if Det[eMat]=0 *)

getHHNRotPEmat [eMat_] :=  Module[{rootEtE, rotInv, rotOpt},
       rotInv = Inverse[eMat];
       rootEtE = MatrixPower[Transpose[eMat] . eMat, +1/2];
       rotOpt = rootEtE . rotInv]

(*  HNN "Minus root" Our Preferred version does not need inverse of eMat,
     just inverse of the S factor and a Transpose.*)

getHHNRotMEmat [eMat_] :=  Module[{invRootEtE, rotOpt},
       invRootEtE = MatrixPower[Transpose[eMat] . eMat, -1/2];
       rotOpt = invRootEtE . Transpose[eMat]]
```

I. The List of DRaM class EnP Pose Discovery Functions

These functions produce perfect rotation results *only* for perfect, error-free inputs, and become non-orthonormal in the presence of error. Rotation Correction restores orthonormality, and their accuracy in the presence of error is then within a few degrees of the `ArgMin` , but never exact.

```
 getAnyQRDecRot[cloud_, target_] :=  Module[{theS, theT},
           {theS, theT} = QRDecomposition[cloud];
           Transpose[target] . Transpose[theS] . Inverse[Transpose[theT]]]

(* Reference:  Moore-Penrose PseudoInverse has this implementation *)
myPseudoInverse[matrix_] :=  Module[{MtMForm = Transpose[matrix] . matrix},
             Inverse[MtMForm] . Transpose[matrix]]

(* This  extracts the pose matrix using only the  cloud data matrix: *)
 getAnyPseudoInverseRot[cloud_, target_] :=  Transpose[PseudoInverse[cloud] . target]

 getEnPDRaM[cloud_, target_] :=
  Module[{the3DMatchDraM, d0, d11, d12, d13, d21, d22, d23, d31, d32, d33},
   Module[{xx, yy, zz, xy, xz, yz, ux, uy, uz, vx, vy, vz, wx, wy, wz},
    Module[{ xList = Transpose[cloud][[1]], yList = Transpose[cloud][[2]],
             zList = Transpose[cloud][[3]],
     uList = Transpose[target][[1]],  vList = Transpose[target][[2]],
           wList = Transpose[target][[3]]},
     {xx, yy, zz, xy, xz, yz, ux,  uy, uz, vx, vy, vz , wx, wy, wz} =
         {xList . xList, yList . yList, zList . zList,
            xList . yList,  xList . zList, yList . zList,
      uList . xList, uList . yList, uList . zList,
          vList . xList, vList . yList, vList . zList,
             wList . xList, wList . yList, wList . zList};
     d0 = Det[{{xx, xy, xz}, {xy, yy, yz}, {xz, yz, zz}}];
    d11 = Det[{{xy, yy, yz}, {xz, yz, zz}, {ux, uy, uz}}];
    d12 = Det[{{xz, yz, zz}, {xx, xy, xz}, {ux, uy, uz}}];
    d13 = Det[{{xx, xy, xz}, {xy, yy, yz}, {ux, uy, uz}}];
    d21 = Det[{{xy, yy, yz}, {xz, yz, zz}, {vx, vy, vz}}];
    d22 = Det[{{xz, yz, zz}, {xx, xy, xz}, {vx, vy, vz}}];
    d23 = Det[{{xx, xy, xz}, {xy, yy, yz}, {vx, vy, vz}}];
    d31 = Det[{{xy, yy, yz}, {xz, yz, zz}, {wx, wy, wz}}];
    d32 = Det[{{xz, yz, zz}, {xx, xy, xz}, {wx, wy, wz}}];
    d33 = Det[{{xx, xy, xz}, {xy, yy, yz}, {wx, wy, wz}}];
    the3DMatchDraM = {{d11/d0, d12/d0, d13/d0},
                      {d21/d0, d22/d0, d23/d0},
                      {d31/d0, d32/d0, d33/d0}}] ]]
```

Rotation Correction forms for the non-orthonormal DRaM class candidates for the EnP pose with errorful data. Our convention is to require the transpose when using `getSVDRot[mat]` or equivalent for rotation correction.

```
(* Using this form, which solves any EnP problem with the cross-covariance
      matrix as the argument, must be called with the transpose rotation
      matrix candidate for Rotation Correction. *)

getSVDRot[matrix_] :=  Module[{dim = Dimensions[matrix][[1]], uu, ss, vv, dd},
         {uu, ss, vv} =  SingularValueDecomposition[matrix];
         dd = DiagonalMatrix[
                  Table[If[i == dim, Sign[Det[uu] Det[vv]], 1], {i, 1, dim}]];
         vv . dd . Transpose[uu]]


(*  Corrected  DRaM changes angular error from 1.78 deg to 1.42 deg  *)
getEnPDRaMRC[cloud_, target_] :=
       getSVDRot[Transpose[getEnPDRaM[ cloud, target]]  ]

(*  Corrected  QR Decomposition  changes angular error from 1.78 deg to 1.42 deg  *)
getAnyQRDecRotRC[cloud_, target_] :=
     getSVDRot[Transpose[getAnyQRDecRot[cloud, target]]]

 (*  Corrected  PseudoInverse Map  changes angular error from 1.78 deg to 1.42 deg  *)
getAnyPseudoInverseRotRC[cloud_, target_] :=
    getSVDRot[Transpose[getAnyPseudoInverseRot[cloud, target]]]
```

J. The List of RMSD-class OnP Pose Discovery Functions

These are adapted from the EnP class to handle a planar 2D target, which always fails to give a reasonable pose estimate.

```
getOrthoArgMin[cloud_, ortho_ ] :=   Module[{loss,ruleQ, quat, rot},
   {loss, ruleQ } = FindMinimum[{loss3DOrthoFunction[qrotsym,  cloud,  ortho],
         q0^2 + q1^2 + q2^2 + q3^2 == 1}, {{q0, 1}, {q1, 0}, {q2,  0}, {q3, 0}},
          AccuracyGoal -> 12 ];
   quat = {q0, q1, q2, q3} /. ruleQ;
  {makeq0Plus[quat], QuatToRot[quat]}]

getHebert23Rot[cloud_, ortho_] :=  Module[{Bmat, eigopt, quat},
     Bmat = buildFullEnPBmat[cloud, ({#1[[1]], #1[[2]], 0} &) /@ ortho];
     eigopt = Chop[Min[Eigenvalues[Bmat]]];
     quat = pick4DQAdj[Adjugate[Bmat - eigopt IdentityMatrix[4]]];
     QuatToRot[quat]]

(* This has an obvious alternate  getHorn23Rot[Emat23] form *)
getHorn23RotXY[cloud_, ortho_] :=  Module[{Mmat, Emat23, eigopt, quat},
      Emat23 = Transpose[  Transpose[cloud] . ortho];
     Mmat = makeMxx23mat[Emat23];
    eigopt = Max[Eigenvalues[Mmat]];
     quat = pick4DQAdj[Adjugate[Mmat - eigopt IdentityMatrix[4]]];
    Transpose[QuatToRot[quat]] ]

(* This has an obvious alternate  getSVD23Rot[Emat23] form *)
 getSVD23RotXY[cloud_, ortho_] :=
 Module[{Emat23 = Transpose[Transpose[cloud] . ortho], uu, vv, ss, dd,
   r1, r2, r3, rotOpt}, {uu, ss, vv} = SingularValueDecomposition[Emat23];
  dd = {{1, 0, 0}, {0, 1, 0}}; {r1, r2} = uu . dd . Transpose[vv];
  r3 = Normalize[Cross[r1, r2]]; rotOpt = {r1, r2, r3}]

    (* This  HHN  2x3  version  exploits the PseudoInverse of the
                singular matrix  EtE and that works..  *)
 (* This has an obvious alternate  getHNN23Rot[Emat23] form *)
  getHHN23RotXY [cloud_, ortho_] :=
  Module[{Emat23, EtE, pseudoinv, rootEtEinv, p23Opt, rot3, rotOpt},
      Emat23 = Transpose[Transpose[cloud] . ortho];
      EtE = Transpose[Emat23] . Emat23;
      pseudoinv = PseudoInverse[EtE];
      rootEtEinv = MatrixPower[pseudoinv, +1/2];
      p23Opt = Emat23 . rootEtEinv;
      rot3 = Normalize[Cross[p23Opt[[1]], p23Opt[[2]]]];
      rotOpt = {p23Opt[[1]], p23Opt[[2]], rot3}]
```

K. The List of DRaM class OnP Pose Discovery Functions

These functions produce perfect rotation results *only* for perfect, error-free inputs, and become non-orthonormal in the presence of error. Rotation Correction restores orthonormality, and their accuracy in the presence of error is then within a few degrees of the `ArgMin` , but never exact.

```
(* These are universal, for 3x3 targets or 2x3 ortho data targets.  *)
 getAnyQRDecRot[cloud_, ortho_] :=  Module[{theS, theT},
           {theS, theT} = QRDecomposition[cloud];
           Transpose[ortho] . Transpose[theS] . Inverse[Transpose[theT]]]

(* These are universal, for 3x3 targets or 2x3 ortho data targets.  *)
 getAnyPseudoInverseRot[cloud_, ortho_] :=  Transpose[PseudoInverse[cloud] . ortho]


 getOnPDRaM[cloud_, ortho_] :=
 Module[{the3DposeInit, d0, d11, d12, d13, d21, d22, d23, dd31, dd32, dd33},
  Module[{xx, yy, zz, xy, xz, yz, ux, uy, uz, vx, vy, vz},
   Module[{
      xList = Transpose[cloud][[1]],
            yList = Transpose[cloud][[2]],
                 zList = Transpose[cloud][[3]],
      uList = Transpose[ortho][[1]],
            vList = Transpose[ortho][[2]]},
      {xx, yy, zz,   xy, xz, yz,   ux, uy, uz,    vx, vy, vz } =
          {xList . xList, yList . yList, zList . zList,
             xList . yList, xList . zList, yList . zList,
               uList . xList, uList . yList, uList . zList,
                   vList . xList, vList . yList, vList . zList};
    d0 = Det[{{xx, xy, xz}, {xy, yy, yz}, {xz, yz, zz}}];
    d11 = Det[{{xy, yy, yz}, {xz, yz, zz}, {ux, uy, uz}}];
    d12 = Det[{{xz, yz, zz}, {xx, xy, xz}, {ux, uy, uz}}];
    d13 = Det[{{xx, xy, xz}, {xy, yy, yz}, {ux, uy, uz}}];
    d21 = Det[{{xy, yy, yz}, {xz, yz, zz}, {vx, vy, vz}}];
    d22 = Det[{{xz, yz, zz}, {xx, xy, xz}, {vx, vy, vz}}];
    d23 = Det[{{xx, xy, xz}, {xy, yy, yz}, {vx, vy, vz}}];
    dd31 = Det[{{xx, xy, xz}, {ux, uy, uz}, {vx, vy, vz}}];
    dd32 = Det[{{xy, yy, yz}, {ux, uy, uz}, {vx, vy, vz}}];
    dd33 = Det[{{xz, yz, zz}, {ux, uy, uz}, {vx, vy, vz}}];
    the3DposeInit  =
                    {{d11/d0,  d12/d0,  d13/d0},
                     {d21/d0,  d22/d0,  d23/d0},
                     {dd31/d0, dd32/d0, dd33/d0}}] ]]
```

Rotation Correction forms for the non-orthonormal DRaM class candidates for the OnP pose with errorful data. Our convention is to require the transpose when using `getSVDRot[mat]` or equivalent for rotation correction.

```
(* This form, which solves any EnP problem with the cross-covariance
      matrix as the argument, must be called with the transpose of
      a 3x3 rotation  matrix candidate for Rotation Correction. *)

getSVDRot[matrix_] :=  Module[{dim = Dimensions[matrix][[1]], uu, ss, vv, dd},
         {uu, ss, vv} =  SingularValueDecomposition[matrix];
         dd = DiagonalMatrix[
                  Table[If[i == dim, Sign[Det[uu] Det[vv]], 1], {i, 1, dim}]];
         vv . dd . Transpose[uu]]

(* This form takes a 2x3 partial rotation matrix and correct it, no transpose.
    Note the inversion of the final SVD matrix order. *)
getSVD23Rot[Emat23_] :=  Module[{uu, vv, ss, dd, r1, r2, r3, rotOpt},
     {uu, ss, vv} =   SingularValueDecomposition[Emat23];
    dd = {{1, 0, 0}, {0, 1, 0}}; {r1, r2} = uu . dd . Transpose[vv];
     r3 = Normalize[Cross[ r1, r2];
     rotOpt = {r1, r2, r3}]

(*  Corrected  DRaM changes angular error from nonsense  to 2.85 deg  *)
getOnPDRaMRC[cloud_, ortho] :=
       getSVDRot[Transpose[getOnPDRaM[ cloud, ortho]]  ]

(*  Corrected  QR Decomposition  changes angular error from  nonsense  to 2.85 deg  *)
getOnPQRDecRotRC[cloud_, ortho] :=
     getSVD23Rot[getAnyQRDecRot[cloud, ortho]]

 (*  Corrected  PseudoInverse Map  changes angular error from  nonsense  to 2.85 deg  *)
getOnPPseudoInverseRotRC[cloud_, ortho_] :=
       getSVD23Rot[getAnyPseudoInverseRot[cloud, ortho]]
```

L. The 3D:3D EnP DRaM Solution

```
poseLSQEnP =
 Module[{ rotq = {{q00 + q11 - q22 - q33, -2 q03 + 2 q12,  2 q02 + 2 q13},
                           {2 q03 + 2 q12,  q00 - q11 + q22 - q33, -2 q01 + 2 q23},
                           {-2 q02 + 2 q13, 2 q01 + 2 q23, q00 - q11 - q22 + q33}}},
   Expand[(rotq . {x, y, z} - {u, v, w}) .(*dot*)(rotq . {x, y, z} - {u, v, w})  ] /.
               {x^2 -> xx,  x y -> xy, x z -> xz, y x -> xy, y^2 -> yy, y z -> yz,
               z x -> xz,  z y -> yz, z^2 -> zz,
                u x -> ux, u y -> uy, u z -> uz, v x -> vx, v y -> vy,  v z -> vz,
                x w -> wx,  y w -> wy,  z w -> wz,  u^2 -> uu, v^2 -> vv}] //
                Collect[#, {q00, q11, q22, q33, q01, q02, q03, q23, q13, q12}] &

 Out[] :=   uu+vv+q12 (-4 uy-4 vx)+w^2+q23 (-4 vz-4 wy+8 q13 xy+8 q12 xz)+
 q03^2 (4 xx+4 yy)+q12^2 (4 xx+4 yy)+q33 (2 ux+2 vy-2 wz-8 q12 xy-8 q02 xz+
 8 q01 yz)+q13 (-4 uz-4 wx+8 q12 yz)+q03 (4 uy-4 vx+8 q23 xz+q12 (8 xx-8 yy)-
 8 q13 yz)+q01 (4 vz-4 wy-8 q02 xy+8 q13 xy-8 q03 xz-8 q12 xz+q23 (8 yy-8 zz))+
 q22 (2 ux-2 vy+2 wz+8 q03 xy-8 q13 xz-8 q01 yz+q33 (2 xx-2 yy-2 zz))+
 q00^2 (xx+yy+zz)+q11^2 (xx+yy+zz)+q22^2 (xx+yy+zz)+q33^2 (xx+yy+zz)+
 q02^2 (4 xx+4 zz)+q13^2 (4 xx+4 zz)+q01^2 (4 yy+4 zz)+q23^2 (4 yy+4 zz)+
 q11 (-2 ux+2 vy+2 wz-8 q03 xy+8 q02 xz-8 q23 yz+q33 (-2 xx+2 yy-2 zz)+
 q22 (-2 xx-2 yy+2 zz))+q00 (-2 ux-2 vy-2 wz+8 q12 xy+8 q13 xz+8 q23 yz+
 q11 (2 xx-2 yy-2 zz)+q22 (-2 xx+2 yy-2 zz)+q33 (-2 xx-2 yy+2 zz))+
 q02 (-4 uz+4 wx-8 q23 xy-8 q03 yz+8 q12 yz+q13 (-8 xx+8 zz))

  (*   ~ 0.5  seconds to Solve,   ~ 9 seconds to Simplify *)
  theAdjSolnsEnP =
     Module[{eqn = poseLSQEnP},
         Solve[{D[eqn, q00] == 0, D[eqn, q11] == 0, D[eqn, q22] == 0,
                   D[eqn, q33] == 0, D[eqn, q01] == 0, D[eqn, q02] == 0,
                   D[eqn, q03] == 0, D[eqn, q23] == 0, D[eqn, q13] == 0,
                   D[eqn, q12] == 0, q00 + q11 + q22 + q33 == 1},
                   {q00, q11, q22,  q33, q01, q02, q03, q23, q13, q12}]][[1]] /Simplify ]

  qqrotsym /. theAdjSolnsEnP // Simplify

  Out[]:=  (* The 3 x 3 EnP DRaM rotation matrix *)
```

M. The 3D:2D Orthographic OnP DRaM Solution

```
 poseLSQOnP=Expand[({{q00+q11-q22-q33,-2 q03+2 q12,2 q02+2 q13},
            {2 q03+2 q12,q00-q11+q22-q33,-2 q01+2 q23}}.{x,y,z}-{u,v}).(*dot*)
    ({{q00+q11-q22-q33,-2 q03+2 q12,2 q02+2 q13},
        {2 q03+2 q12,q00-q11+q22-q33,-2 q01+2 q23}} . {x,y,z} - {u,v})]/.
           {x^2->xx,x y->xy,x z->xz,y x->xy,y^2->yy,y z->yz,z x->xz,z y->yz,z^2->zz,
             u x->ux,u y->uy,u z->uz,v x->vx,v y->vy,v z->vz,u^2->uu,v^2->vv}//
       Collect[#,{q00,q11,q22,q33,q01,q02,q03,q23,q13,q12}]&

Out[]:= uu+vv+q12 (-4 uy-4 vx)+q23 (-4 vz+8 q12 xz)+q00^2 (xx+yy)+q11^2 (xx+yy)+
q22^2 (xx+yy)+q33^2 (xx+yy)+q03^2 (4 xx+4 yy)+q12^2 (4 xx+4 yy)+q13 (-4 uz+
8 q12 yz)+q03 (4 uy-4 vx+8 q23 xz+q12 (8 xx-8 yy)-8 q13 yz)+q33 (2 ux+
2 vy-8 q12 xy-4 q02 xz-4 q13 xz+4 q01 yz-4 q23 yz)+q11 (-2 ux+2 vy-8 q03 xy+
4 q02 xz+4 q13 xz+q22 (-2 xx-2 yy)+q33 (-2 xx+2 yy)+4 q01 yz-4 q23 yz)+
q22 (2 ux-2 vy+8 q03 xy-4 q02 xz-4 q13 xz+q33 (2 xx-2 yy)-4 q01 yz+4 q23 yz)+
q00 (-2 ux-2 vy+8 q12 xy+4 q02 xz+4 q13 xz+q33 (-2 xx-2 yy)+q11 (2 xx-2 yy)+
q22 (-2 xx+2 yy)-4 q01 yz+4 q23 yz)+4 q01^2 zz+4 q02^2 zz+4 q13^2 zz+
4 q23^2 zz+q02 (-4 uz-8 q03 yz+8 q12 yz+8 q13 zz)+
q01 (4 vz-8 q03 xz-8 q12 xz-8 q23 zz)

 (* ~ 7 seconds to Solve,  ~ 33 seconds to Simplify *)
 the3D2DAdjSolns =
   Module[{eqn = poseLSQOnP},
     Solve[{D[eqn, q00] == 0, D[eqn, q11] == 0, D[eqn, q22] == 0,
       D[eqn, q33] == 0, D[eqn, q01] == 0, D[eqn, q02] == 0,
       D[eqn, q03] == 0, D[eqn, q23] == 0, D[eqn, q13] == 0,
       D[eqn, q12] == 0, q00 + q11 + q22 + q33 == 1,
       q00 q11 == q01 q01, q00 q22 == q02 q02, q00 q33 == q03 q03},
      {q00, q11, q22, q33, q01, q02, q03, q23, q13, q12}]][[1]]

(* Top row of OnP DRaM: *)
   {q00 + q11 - q22 - q33,  -2 q03 + 2 q12, 2 q02 + 2 q13} /. the3D2DAdjSolns // Simplify

(* Second row of OnP DRaM: *)
   {2 q03 + 2 q12,  q00 - q11 + q22 - q33, -2 q01 + 2 q23} /.  the3D2DAdjSolns  // Simplify

  Out[]:= (* The 2 x 3 OnP DRaM rotation matrix *)
```

N. The rij - based 3D:3D EnP DRaM Solution

```
  (* This least squares form works for EnP and also OnP if we drop 3rd line of rij *)

poseLSQEnPRR =
 Module[{ rot  = {{r11, r12, r13}, {r21, r22, r23}, {r31, r32, r33}}},
   Expand[(rot . {x, y, z} - {u, v, w}) .(*dot*)(rot . {x, y, z} - {u, v, w})] /.
    {x^2 -> xx, x y -> xy, x z -> xz, y x -> xy, y^2 -> yy,  y z -> yz, z x -> xz,
    z y -> yz, z^2 -> zz,  u x -> ux, u y -> uy, u z -> uz, v x -> vx, v y -> vy,
    v z -> vz,  x w -> wx,  y w -> wy,  z w -> wz, u^2 -> uu, v^2 -> vv, w^2 -> ww}] //
  Collect[#, {r11, r12, r13, r21, r22, r23, r31, r32, r33}] &

Out -> uu - 2 r13 uz + vv - 2 r23 vz + ww - 2 r33 wz +
  r11^2 xx + r21^2 xx +  r31^2 xx + r11 (-2 ux + 2 r12 xy + 2 r13 xz) +
  r21 (-2 vx + 2 r22 xy + 2 r23 xz) +  r31 (-2 wx + 2 r32 xy + 2 r33 xz) +
  r12^2 yy + r22^2 yy + r32^2 yy +  r12 (-2 uy + 2 r13 yz) + r22 (-2 vy + 2 r23 yz) +
  r32 (-2 wy + 2 r33 yz) +  r13^2 zz + r23^2 zz + r33^2 zz

(*  The solution of the LSQ symbolic form (for perfect data). *)
Timing[
  rijOfxyzuvwRule  = Module[{eqn = poseLSQEnPRR},
    Solve[{  D[eqn, r11] == 0, D[eqn, r12] == 0, D[eqn, r13] == 0,
             D[eqn, r21] == 0, D[eqn, r22] == 0, D[eqn, r23] == 0,
             D[eqn, r31] == 0, D[eqn, r32] == 0, D[eqn, r33] == 0},
        {r11, r12, r13, r21, r22, r23, r31, r32, r33}] ] [[1]] //  Simplify ]

 Out =->  0.011 sec.
 {r11 -> (uz xz yy - uz xy yz - uy xz yz + ux yz^2 + uy xy zz - ux yy zz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r12 -> (-uz xy xz + uy xz^2 + uz xx yz - ux xz yz - uy xx zz +  ux xy zz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r13 -> (uz xy^2 - uy xy xz - uz xx yy + ux xz yy + uy xx yz - ux xy yz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r21 -> (vz xz yy - vz xy yz - vy xz yz + vx yz^2 + vy xy zz - vx yy zz)/
                (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r22 -> (-vz xy xz + vy xz^2 + vz xx yz - vx xz yz - vy xx zz + vx xy zz)/
                (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r23 -> (vz xy^2 - vy xy xz - vz xx yy + vx xz yy + vy xx yz - vx xy yz)/
                 (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r31 -> (wz xz yy - wz xy yz - wy xz yz + wx yz^2 + wy xy zz - wx yy zz)/
                  (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r32 -> (-wz xy xz + wy xz^2 + wz xx yz - wx xz yz - wy xx zz + wx xy zz)/
                  (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r33 -> (wz xy^2 - wy xy xz - wz xx yy + wx xz yy + wy xx yz - wx xy yz)/
                  (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz) }
```

O. The rij - based 3D:2D Orthographic OnP DRaM Solution

```
 (* Same eqn as EnP can be used, omit r3k bottom line. *)
 Timing[
   rijOnPxyzuvRule  = Module[{eqn = poseLSQEnPRR},
    Solve[{  D[eqn, r11] == 0,   D[eqn, r12] == 0,   D[eqn, r13] == 0,
                D[eqn, r21] == 0,   D[eqn, r22] == 0,    D[eqn, r23] == 0
                   (*D[eqn,r31]==0, D[eqn,r32]==0,  D[eqn,r33]==0*)},
                {r11, r12, r13, r21, r22, r23}] ] [[1]] //   Simplify ]

Out ->  0.0082  sec
{r11 -> (uz xz yy - uz xy yz - uy xz yz + ux yz^2 + uy xy zz - ux yy zz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r12 -> (-uz xy xz + uy xz^2 + uz xx yz - ux xz yz - uy xx zz +  ux xy zz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r13 -> (uz xy^2 - uy xy xz - uz xx yy + ux xz yy + uy xx yz - ux xy yz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r21 -> (vz xz yy - vz xy yz - vy xz yz + vx yz^2 + vy xy zz - vx yy zz)/
               (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r22 -> (-vz xy xz + vy xz^2 + vz xx yz - vx xz yz - vy xx zz + vx xy zz)/
                (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz),
  r23 -> (vz xy^2 - vy xy xz - vz xx yy + vx xz yy + vy xx yz - vx xy yz)/
                (xz^2 yy - 2 xy xz yz + xx yz^2 + xy^2 zz - xx yy zz)}
```