2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

An Efficient Solution to Absolute Orientation

Manolis Lourakis
Institute of Computer Science, Foundation for Research and Technology - Hellas
Nikolaou Plastira 100, GR - 70013, Heraklion, Greece

***Abstract*—The absolute orientation problem arises often in vision and robotics. Despite that robust algorithmic solutions exist for quite some time, they all rely on matrix factorizations such as eigen or singular value decomposition. These factorizations are relatively expensive to compute, therefore might become a performance bottleneck when absolute orientation needs to be repeatedly computed on low-end hardware. The issue is exacerbated by implementations relying on standard numerical software libraries like LAPACK, since the linear algebra factorization routines they include are optimized for large matrices and thus are not the most efficient choice for small ones. Based on an attitude estimation algorithm originating from astronautics, this paper proposes a direct, factorization-free solution to the absolute orientation problem that is both computationally efficient and numerically accurate. Results from an experimental comparison with established methods demonstrate its superior performance.**

I. Introduction

Absolute orientation is the problem of determining the rigid transformation aligning two sets of corresponding 3D points. This problem arises often in computer vision, graphics, photogrammetry and robotics. For example, it is involved in the solution of several variants of the perspective-n-point (PnP) problem such as the P3P solvers in [1], [2], the P4P solver in [3] and the EPnP solver of [4]. In all these cases, the distances from the camera focal point to a set of n 3D reference points are estimated and then used to compute the coordinates of the reference points in the camera coordinate frame. Following this, absolute orientation is invoked to compute the camera pose as the transformation mapping the reference points from the world to the camera coordinate system. Another common occurrence of absolute orientation is during the alignment of two point sets using the iterative closest point (ICP) algorithm with the point-to-point error metric [5]. There, pairs of closest points are presumed to correspond and absolute orientation is used to find the transformation that best aligns the point sets. The estimated transformation is applied to one of the sets to bring it closer to the other and the process repeats until convergence. In a similar vein, the need for absolute orientation also arises in stereo visual odometry, where the relative camera motion between two successive stereo images is estimated via the registration of two 3D point sets reconstructed from image features that are matched in the stereo pairs [6].

To safeguard against mismatched points (i.e. outliers), calculations such as PnP and ICP are typically applied in a robust regression framework such as RANSAC [7]. In this setting, given hundreds of noisy points, small random point sets are repeatedly chosen, PnP or ICP is solved and the solution is ranked according to how well it agrees with the rest of the data. Eventually, the best solution found after a certain number of iterations is retained. As a result, absolute orientation needs to be estimated frequently, therefore completing this task fast is important to the efficiency of the overall algorithm. This becomes even more important when the estimation is carried out on hardware of limited capacity such as that found on mobile devices, unmanned aerial vehicles (UAVs), space rovers, underwater robots, etc.

Attitude (i.e. orientation) estimation is crucial for the guidance and navigation of modern spacecraft and a significant amount of research has focused on it [8]. Despite that such developments are directly relevant to the often-encountered absolute orientation problem, the computer vision and robotics communities seem to be unaware of this fact. This paper establishes a link between attitude and absolute orientation estimation and puts forward a fast algorithm to deal with the latter based on the FOAM algorithm of [9]. To the best of the author's knowledge, estimating absolute orientation with attitude determination techniques has not been attempted before. Compared to established techniques based on eigen decomposition or SVD, the proposed algorithm is much faster and yields estimates of practically identical accuracy.

The rest of the paper is organized as follows. Section II provides an overview of attitude and absolute orientation estimation. The proposed absolute orientation algorithm is presented in section III and is compared with established methods in section IV. The paper concludes in section V. Concerning notation, in the following matrices are typeset in bold uppercase whereas bold lowercase letters denote vectors.

II. Background

A. Attitude Estimation

Determining the orientation of a spacecraft is known as attitude estimation in astronautics. It is a topic that has developed significantly after Wahba published her famous attitude determination problem involving any number of vector observations [8]: Given two sets of corresponded unit vectors $\{\mathbf{b}_i\}$ and $\{\mathbf{r}_i\}$, Wahba's problem is to find the orthogonal matrix $\mathbf{A}$ with determinant $+1$ minimizing the loss function

$$L(\mathbf{A}) = \frac{1}{2}\sum_{i=1}^{N} ||\mathbf{b}_i - \mathbf{A}\mathbf{r}_i||^2. \quad (1)$$

Owing to the orthogonality of $\mathbf{A}$, the terms $||\mathbf{b}_i - \mathbf{A}\mathbf{r}_i||^2$ can be simplified as $||\mathbf{b}_i||^2 + ||\mathbf{A}\mathbf{r}_i||^2 - 2\ \mathbf{b}_i^T(\mathbf{A}\mathbf{r}_i) = 2 - 2\ trace(\mathbf{A}\mathbf{r}_i\mathbf{b}_i^T)$, hence Eq. (1) can be written as

$$L(\mathbf{A}) = \lambda_0 - trace(\mathbf{A}\mathbf{B}^T), \quad (2)$$

where $\lambda_0 = N$, $trace(.)$ denotes matrix trace (sum of diagonal elements) and $\mathbf{B}$ is the "attitude profile matrix" defined by

$$\mathbf{B} \equiv \sum_{i=1}^{N} \mathbf{b}_i \mathbf{r}_i^T. \tag{3}$$

Minimizing Eq. (2) is equivalent to maximizing $trace(\mathbf{A}\mathbf{B}^T)$.

Several solutions minimizing Eq. (1) have been developed since [8], some of them quite demanding for the limited capacity flight computers of their time. An early, direct solution for exactly two pairs of corresponding unit vectors is provided by the so-called TRIAD method [10]. Apart from not generalizing to arbitrary numbers of vectors, TRIAD is based on the assumption that one of the unit vectors, let $\mathbf{b}_1$, is more accurately determined than the other, so the estimate satisfies $\mathbf{A}\mathbf{r}_1 = \mathbf{b}_1$ exactly, but $\mathbf{A}\mathbf{r}_2 = \mathbf{b}_2$ only approximately. Solutions to Eq. (1) have been largely overlooked by the computer vision and robotics communities, despite being highly relevant to the absolute orientation problem, as will be explained next.

B. Absolute Orientation

Let $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_i\}$, $i = 1 \dots N$ be two sets of corresponding 3D points measured in two different coordinate systems. The absolute orientation problem consists in finding the Euclidean transformation $\mathbf{R}$, $\mathbf{t}$ that aligns the two sets. For three points in general position, the problem has a unique solution. Since points are corrupted by noise, for $N > 3$ a least squares solution minimizing the mean squared residual error is sought:

$$\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{q}_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})||^2. \tag{4}$$

It is well-known that this problem is simplified by moving the origin of the two coordinate systems to the point set centroids, defined by

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i, \qquad \bar{\mathbf{q}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{q}_i. \tag{5}$$

The new point coordinates are then given by

$$\mathbf{p}_i^{'} = \mathbf{p}_i - \bar{\mathbf{p}}, \qquad \mathbf{q}_i^{'} = \mathbf{q}_i - \bar{\mathbf{q}} \text{ for } i = 1 \dots N \tag{6}$$

and the mean squared error from (4) becomes

$$\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{q}_i^{'} + \bar{\mathbf{q}} - (\mathbf{R}(\mathbf{p}_i^{'} + \bar{\mathbf{p}}) + \mathbf{t})||^2. \tag{7}$$

Horn showed in [11] that (7) simplifies to

$$\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{q}_i^{'} - \mathbf{R}\mathbf{p}_i^{'}||^2 + ||\bar{\mathbf{q}} - \mathbf{R}\bar{\mathbf{p}} - \mathbf{t}||^2 \tag{8}$$

and observed that since the second term is non-negative, the mean squared error is minimized when it equals zero. Hence, the optimal translation $\hat{\mathbf{t}}$ and rotation $\hat{\mathbf{R}}$ are related with

$$\hat{\mathbf{t}} = \bar{\mathbf{q}} - \hat{\mathbf{R}}\bar{\mathbf{p}} \tag{9}$$

and the mean squared error to be minimized consists of just the first term of (8), i.e.

$$\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{q}_i^{'} - \mathbf{R}\mathbf{p}_i^{'}||^2. \tag{10}$$

The last two expressions are at the heart of absolute orientation estimation: the rotation is determined by minimizing (10) and translation follows by substitution of the estimated rotation in (9). The rotation minimizing (10) was shown in [11] to be represented by a unit quaternion which is the eigenvector associated with the largest eigenvalue of a symmetric 4×4 matrix. In a subsequent paper, Horn et al. [12] used orthonormal matrices to represent rotations and developed a solution based on the eigen decomposition of a symmetric 3×3 matrix. A solution for the best rotation based on the singular value decomposition (SVD) of a 3×3 covariance matrix was proposed by Arun et al. [13]. This solution was later modified by Umeyama, to ensure that it will always yield a rotation instead of occasionally computing a reflection when the input point sets are planar or very noisy [14]. The various solutions proposed in [11], [12], [13] were compared in [15].

Both the eigensystem and the SVD-based solutions share the drawback of involving rather expensive numerical matrix operations. High-quality, numerically robust SVD and eigensystem routines are nowadays included in standard linear algebra packages such as IMSL, MKL and LAPACK. However, as these are targeted to large matrices, they do not perform efficiently for small ones. Furthermore, these routines are iterative, thus their execution times can vary depending on input and degeneracy. As a result, using direct solutions instead of employing solvers invoking such canned linear algebra routines is highly desirable, especially when estimation is executed often on low capacity hardware, for example the slow, radiation-hardened CPUs on board planetary rovers [6].

The key observation of this work is that the centroid-centered points in Eq. (6) can be seen as defining direction vectors which are transformed covariantly by a rotation matrix (i.e., the new direction vectors are the rotated old ones), similarly to the vectors involved in Wahba's problem. Indeed, close inspection of (10) and (1) reveals that the two are very similar. Apart from the scale factors, their only difference is that (1) is defined for unit vectors whereas (10) it is not. However, as it will be explained in the next section, this is not crucial for the solution and the FOAM algorithm from [9] can be applied almost unmodified to obtain the optimal rotation.

III. Optimal Rotation with the FOAM Algorithm

In the case of non-unit corresponded vectors, term λ_0 in Eq. (2) equals $1/2$ $(\sum_{i=1}^{N} ||\mathbf{b}_i||^2 + \sum_{i=1}^{N} ||\mathbf{r}_i||^2)$ and is clearly still independent of the optimal rotation. In [9], Markley presented a direct solution for the optimal attitude. In the remainder of the section, a brief overview of this solution is provided and the interested reader is referred to the original publication for more details.

It is shown in [9] that the optimal rotation matrix is given for $\lambda = \lambda_{max}$ by the following equation:

$$\mathbf{R}(\lambda) = \frac{(\lambda^2 + ||\mathbf{B}||_F^2)\mathbf{B} + 2\,\lambda\,adj(\mathbf{B}^T) - 2\,\mathbf{B}\mathbf{B}^T\mathbf{B}}{\lambda(\lambda^2 - ||\mathbf{B}||_F^2) - 2\,det(\mathbf{B})}, \quad (11)$$

where $\mathbf{B}$ is given by Eq. (3), $adj(.)$ and $det(.)$ denote the adjoint and determinant of their matrix argument, and $||.||_F$ is the Frobenius norm, defined as $||\mathbf{B}||_F^2 = trace(\mathbf{B}\mathbf{B}^T)$. Scalar λ_{max} is the largest root of

$$p(\lambda) \equiv (\lambda^2 - ||\mathbf{B}||_F^2)^2 - 8\,\lambda\,det(\mathbf{B}) - 4\,||adj(\mathbf{B})||_F^2 = 0, \quad (12)$$

which originates from $\lambda - trace(\mathbf{R}(\lambda)\mathbf{B}^T) = 0$ (cf. Eq. (2)). The proof that $\mathbf{R}(\lambda_{max})$ from Eq. (11) corresponds to the optimal rotation is based on the representation of the Frobenius norm, determinant and adjoint of $\mathbf{B}$ in terms of its singular values. The latter, however, are only used for analysis and need not be extracted for computing the optimal rotation matrix. Orthogonality of the rotation matrices computed with Eq. (11) for roots of $p(\lambda)$ is guaranteed (cf. Eq. (28) in [9]).

Based on the findings presented above, Markley has proposed his Fast Optimal Matrix Algorithm (FOAM) [9]. FOAM is based on Eq. (11), which is a quartic polynomial in λ with four real roots [16]. Of these roots, the maximum is sought. The roots of Eq. (11) can be computed analytically, however this involves five transcendental function evaluations [9]. Instead, Markley suggests that it is more efficient to exploit Shuster's observation that the maximum root λ_{max} is close to λ_0 [16]. Thus, starting from λ_0, λ_{max} can be computed iteratively with the Newton-Raphson method, using the sequence of estimates defined by

$$\lambda_i = \lambda_{i-1} - p(\lambda_{i-1})/p^{'}(\lambda_{i-1}), \quad (13)$$

where the derivative $p^{'}(\lambda)$ equals $4\,(\lambda^2 - ||\mathbf{B}||_F^2)\lambda - 8\,det(\mathbf{B})$. The iteration converges very fast in practice, except when the maximum root of $p(\lambda)$ is not unique, which results in the derivative in the denominator being close to zero as the root is approached. In this case, the rotation is indeterminate. Once λ_{max} has been computed, the optimal rotation follows from Eq. (11) and translation from Eq. (9). The whole process is illustrated in pseudocode as Algorithm 1.

Algorithm 1 Calculate $\hat{\mathbf{R}}$, $\hat{\mathbf{t}}$ aligning point sets $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_i\}$

Require: $|\{\mathbf{p}_i\}| = |\{\mathbf{q}_i\}| \wedge |\{\mathbf{p}_i\}| \geq 3$
1: Compute $\{\mathbf{p}_i^{'}\}$ and $\{\mathbf{p}_i^{'}\}$ with Eq. (6)
2: Compute $\mathbf{B}$ with Eq. (3) and $\{\mathbf{p}_i^{'}\} = \{\mathbf{r}_i\}$, $\{\mathbf{q}_i^{'}\} = \{\mathbf{b}_i\}$
3: Compute $det(\mathbf{B})$, $||\mathbf{B}||_F^2$, $adj(\mathbf{B})$, $||adj(\mathbf{B})||_F^2$ and $\mathbf{B}\mathbf{B}^T\mathbf{B}$
4: $\lambda_0 \leftarrow 0.5 * (\sum_{i=1}^{N} ||\mathbf{p}_i^{'}||^2 + \sum_{i=1}^{N} ||\mathbf{q}_i^{'}||^2)$
5: $converged \leftarrow$ **false**
// Newton-Raphson, $p(\lambda_i)$ given by Eq. (12)
6: **while** $converged \neq$ **true do**
7: $\lambda_i \leftarrow \lambda_{i-1} - p(\lambda_{i-1})/p^{'}(\lambda_{i-1})$
8: $converged \leftarrow |\frac{\lambda_i - \lambda_{i-1}}{\lambda_i}| < \epsilon$ // $\epsilon > 0$ a small constant
9: **end while**
10: $\lambda_{max} \leftarrow \lambda_i$
11: Compute $\hat{\mathbf{R}}$ from λ_{max} and Eq. (11)
12: Compute $\hat{\mathbf{t}}$ from $\hat{\mathbf{R}}$ and Eq. (9)
13: **return** $\hat{\mathbf{R}}$, $\hat{\mathbf{t}}$

IV. Experiments

Algorithm 1 with a stopping threshold $\epsilon = 1\mathrm{e}{-12}$ as well as those in [11], [12] and [13] were implemented in C, compiled with GNU gcc 4.9.3 and tested on a desktop PC with an Intel Core i7-4790 CPU at 3.60 GHz, 32 Gb of RAM and 256 Kb L1 cache. For brevity, the four algorithms tested will be denoted as 'FOAM', 'Quat', 'Ortho' and 'SVD', respectively.

In order to systematically test the performance of the four algorithms under varying conditions, synthetic input was used. This input consisted of 3D point sets of different sizes that were corrupted by various amounts of noise, generated with a procedure similar to that employed in [15]: For a given size $N \geq 3$, a set of N 3D points were drawn from a uniform distribution in $[-1, 1] \times [-1, 1] \times [-1, 1]$. Then, a random rigid transformation was applied to these 3D points, comprised of a rotation by a unit quaternion and a translation. The quaternion and translation components were uniformly distributed in $[-1, 1]$ and $[-10, 10]$, respectively. Zero mean Gaussian noise with standard deviation σ was added to each component of the transformed points. The size N of the generated data sets ranged from 3 to 10, while the noise level σ varied from 0.0 to $1\mathrm{e}{-02}$ in increments of $1\mathrm{e}{-03}$. Paired with the original points, the noise-corrupted transformed points were used to estimate absolute orientation with all four algorithms. For each point set and algorithm, we measured the elapsed execution time and the root mean square (RMS) of the misalignment error, given by the square root of (4) for the estimated rigid motion. For all tested values for N and σ, 100 trials with different 3D points, rigid transformation and noise were performed and the results averaged.

The time taken by all four algorithms on a modern desktop CPU is very small, therefore accurately measuring it with time routines of limited resolution is problematic. Thus, instead of directly measuring time, in all experiments the elapsed CPU cycles were measured using the Time Stamp Counter (TSC) and the `RDTSC` x86 instruction, as suggested in [17]. As observed in [15], the timings for small data sets are highly dependent on the particular linear algebra routines being used. General-purpose SVD and eigen decomposition routines from LAPACK (namely `dgesvd` and `dsyev`), were found to require several tens of thousands of cycles to complete. Hence, to make the comparison more fair, custom 4×4 and 3×3 Jacobi eigensolvers for symmetric matrices were used for implementing [11] and [12], whereas a custom 3×3 SVD based on solving a cubic polynomial was employed for [13].

It is noted that we are mostly interested in small-sized inputs (in particular, minimal with $N = 3$ and near-minimal ones), as these correspond to practical cases where the absolute orientation problem needs to be frequently solved. Besides, beyond a certain size, the time needed to move points up in the memory hierarchy (cf. steps 1-2 in Alg. 1) becomes the

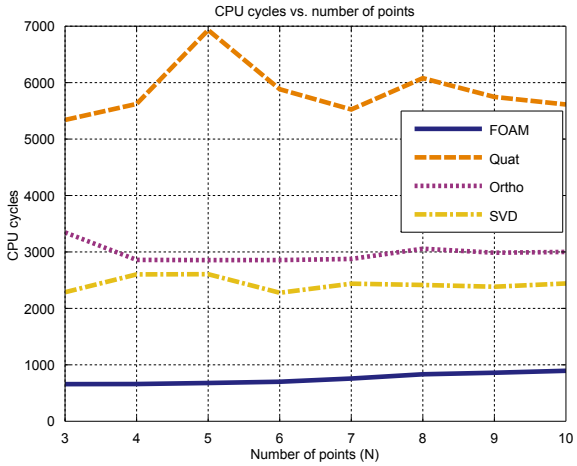

Fig. 1. Execution times (in CPU cycles) for the four algorithms on various sizes of input N and fixed noise $\sigma = 1\text{e}{-}02$. On the CPU tested, 1K cycles amount to 0.28 microsecond (usec).

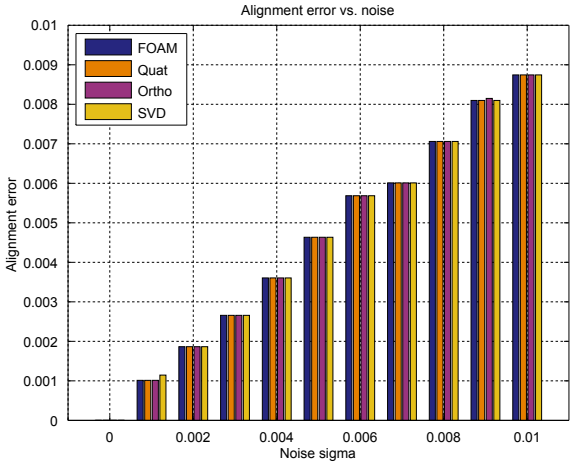

Fig. 2. RMS alignment errors for the four algorithms for $N = 3$ and varying noise. Corresponding errors are identical up to at least the fifth decimal place.

primary bottleneck, dominating the time needed for subsequent linear algebra operations. Figure 1 illustrates the average cycles measured over 100 trials for the four algorithms for various input sizes and the highest amount of noise ($\sigma = 1\text{e}{-}02$). 'FOAM' requires less than 1K cycles to complete and is evidently the fastest of all algorithms, being approximately between 2 to 7 times faster in all cases. It is noted here that if general-purpose linear algebra routines were used instead of the custom, fixed-size ones, these differences would be one to two orders of magnitude larger. 'Ortho' and 'SVD' perform similarly, while 'Quat' is the slowest, most probably due to the increased number of floating point operations necessary to decompose the 4×4 matrices it involves.

Figure 2 compares the average RMS alignment error over 100 trials among the four algorithms for $N = 3$ and various amounts of noise. Clearly, the alignment errors for the proposed method are indistinguishable from those induced by the the other three algorithms, which suggests that the former is a viable alternative to established solutions to absolute orientation. It is also noted that the experiments confirm the conclusions of [15], where 'Quat', 'Ortho' and 'SVD' were also compared against each other and found to have the same accuracy for realistic amounts of noise; 'Ortho' and 'SVD' were also found to be the two fastest algorithms for small N.

V. Conclusion

An efficient solution to the absolute orientation problem has been presented. Experimental results have demonstrated that it executes faster than established alternatives while being on par with them in terms of accuracy. These properties make it particularly attractive for use by real-time applications or on low-end hardware such as that found on mobile devices, UAVs and space robots. Although not shown, the presented solution can easily be extended to account for scale changes as in [11].

References

[1] J. Grunert, "Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in Geodäsie," *Grunerts Archiv für Mathematik und Physik*, vol. 1, pp. 238–248, 1841.

[2] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation," in *Proc. of CVPR '11*, 2011, pp. 2969–2976.

[3] M. Bujnak, Z. Kukelova, and T. Pajdla, "A General Solution to the P4P Problem for Camera With Unknown Focal Length," in *Proc. of CVPR '08*, 2008, pp. 1–8.

[4] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *Int. J. Comput. Vision*, vol. 81, no. 2, pp. 155–166, Feb 2009.

[5] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[6] G. Lentaris, I. Stamoulias, D. Soudris, and M. Lourakis, "HW/SW Co-design and FPGA Acceleration of Visual Odometry Algorithms for Rover Navigation on Mars," *IEEE Trans. Circuits Syst. Video Technol.*, vol. PP, no. 99, pp. 1–1, 2016.

[7] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[8] G. Wahba, "A Least Squares Estimate of Spacecraft Attitude," *SIAM Review*, vol. 7, no. 3, p. 409, Jul 1965.

[9] F. Markley, "Attitude Determination Using Vector Observations: A Fast Optimal Matrix Algorithm," *J. Astronaut. Sci.*, vol. 41, no. 2, pp. 261–280, 1993.

[10] H. Black, "A Passive System for Determining the Attitude of a Satellite," *AIAA Journal*, vol. 2, no. 7, pp. 1350–1351, 1964.

[11] B. K. P. Horn, "Closed-form Solution of Absolute Orientation Using Unit Quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987.

[12] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Am. A*, vol. 5, no. 7, pp. 1127–1135, Jul 1988.

[13] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, Sept 1987.

[14] S. Umeyama, "Least-squares Estimation of Transformation Parameters Between Two Point Patterns," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 4, pp. 376–380, Apr 1991.

[15] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-D Rigid Body Transformations: A Comparison of Four Major Algorithms," *Mach. Vision Appl.*, vol. 9, no. 5-6, pp. 272–290, Mar 1997.

[16] M. Shuster and S. Oh, "Three-axis Attitude Determination from Vector Observations," *J. Guid. Control Dynam.*, vol. 4, no. 1, pp. 70–77, 1981.

[17] G. Paoloni, "How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures," Intel Corp. White Paper, 2010.