

# An Efficient and Stable Algorithm for Learning Rotations

Raman Arora

Department of Electrical Engineering  
University of Washington  
Seattle, WA 98105

William A. Sethares

Dept. of Electrical & Comp. Engineering  
University of Wisconsin-Madison  
Madison, WI 53706

## Abstract

*This paper analyses the computational complexity and stability of an online algorithm recently proposed for learning rotations. The proposed algorithm involves multiplicative updates that are matrix exponentials of skew-symmetric matrices comprising the Lie algebra of the rotation group. The rank-deficiency of the skew-symmetric matrices involved in the updates is exploited to reduce the updates to a simple quadratic form. The Lyapunov stability of the algorithm is established and the application of the algorithm to registration of point-clouds in  $n$ -dimensional Euclidean space is discussed.*

## 1 Introduction

Learning rotations is an important aspect of data processing in many areas including computer vision, robotics, graphics, physics and quantum mechanics. The problem of learning rotations can be broadly classified into batch and online version. Whereas in the batch version two point-clouds are given that can be related with a change-of-basis, in the online version a new pair of corresponding points is obtained at each instance. The batch version is well understood as the problem of orthogonal Procrustes [11] or least-squares linear-fitting [5]. The online version, on the other hand, is challenging and was recently posed as an open problem [12].

Online learning of rotations is especially of interest when the rotations are changing continuously. The complexity of online learning of rotations stems from the fact that the set of rotation matrices is a curved space. The curved space associated with rotation matrices in  $n$ -dimensional Euclidean space is the unit sphere  $\mathbf{S}^{n-1}$  in  $\mathbb{R}^n$ . Consequently, changing rotations may not be tracked using a standard Kalman filter. The gradient of a loss function on  $\mathbf{S}^{n-1}$  gives the geodesic direction and velocity vector on  $\mathbf{S}^{n-1}$ . However, a naive steepest descent algorithms designed for  $\mathbb{R}^n$  takes a step in

direction of the gradient in  $\mathbb{R}^n$ , thereby steering off the manifold. This leads to updates that require repeated approximation and projection on to  $\mathbf{S}^{n-1}$ .

In [3], we proposed an algorithm for online learning of rotations and discussed its application to tracking rotations in  $n$ -dimensional Euclidean space. The key idea in the development of the updates in [3] was to utilize the parallel-transport mechanism along the geodesics on the unit sphere. This avoids the need for repeated approximation and projection on to  $\mathbf{S}^{n-1}$ . The resulting updates are similar to steepest descent algorithms on Riemannian manifolds for optimization under unitary matrix constraint [1, 7]. However, since the updates involve expensive matrix-exponentiation, they are comparable in computational complexity to repeated projection and approximation methods. One of the contributions in this paper is proving a reduction in computational complexity of the online algorithm presented in [3]. The second main contribution of the paper is a Lyapunov stability result for the simplified updates which states that the Frobenius norm of the difference of the true and the estimated rotation matrix is non-increasing.

The proposed algorithm offers another desirable feature, that of averaging over the rotation group. This finds application in registration of noisy point-clouds in  $n$ -dimensional Euclidean space. Note that the group of rotation matrices does not admit the structure of a vector space. Consequently, the observation noise in the point-clouds cannot be dealt with by averaging rotation matrices. However, with the proposed updates, the averaging actually takes place in the Lie algebra (i.e. the tangent vector-space at the identity rotation) associated with the group of rotation matrices. This application to the registration of noisy point-clouds is in many ways similar to iterative closest point (ICP) method restricted to pure rotations [5]. It should be remarked though that ICP acts only on three dimensional data whereas the proposed updates apply to rotations of  $n$ -dimensional data.

## 2 Online Algorithm

Let  $\mathbf{D}_1 = \{\mathbf{x}_i\}_{i=1}^M$  and  $\mathbf{D}_2 = \{\mathbf{y}_i\}_{i=1}^N$  be two point-clouds in  $\mathbb{R}^n$ . Without loss of generality, assume that  $\mathbf{x}_i$  and  $\mathbf{y}_i$  are unit vectors. Let  $\mathbf{R}_*$  be an unknown  $n \times n$  rotation matrix (or a change-of-basis transformation) that relates the two point-clouds, i.e. the matrix  $\mathbf{R}_*$  acts on  $\mathbf{x}_i \in \mathbf{D}_1$  to give the rotated vector  $\mathbf{y}_i = \mathbf{R}_* \mathbf{x}_i \in \mathbf{D}_2$ . Recall that a rotation matrix  $\mathbf{R}$  satisfies the properties that  $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ ,  $\mathbf{R} \mathbf{R}^T = \mathbf{I}$  and  $\det(\mathbf{R}) = 1$ . The rotation group acting on the unit sphere  $\mathbf{S}^{n-1}$  in  $\mathbb{R}^n$  is denoted as  $\mathbf{SO}(n)$ .

At each instance, we learn a new pair of corresponding points  $(\mathbf{x}_t, \mathbf{y}_t) \in \mathbf{D}_1 \times \mathbf{D}_2$  in the two point-clouds. Let  $\hat{\mathbf{R}}_t$  denote the estimate of  $\mathbf{R}_*$  at instance  $t$  and  $\hat{\mathbf{y}}_t = \hat{\mathbf{R}}_t \mathbf{x}_t$  represent the prediction for the rotated vector  $\mathbf{y}_t$ . Let  $L_t(\hat{\mathbf{R}}_t) = d(\mathbf{y}_t, \hat{\mathbf{R}}_t \mathbf{x}_t)$  denote the loss incurred due to error in prediction of rotation of the input vector  $\mathbf{x}_t$ . The estimate of the rotation needs to be updated based on the loss incurred at every instance and the objective is to develop an algorithm for learning  $\mathbf{R}_*$ . Consider the online updates proposed in [3],

$$\hat{\mathbf{R}}_{t+1} = \hat{\mathbf{R}}_t \exp\left(-\eta \mathbf{skew}\left(\hat{\mathbf{R}}_t^T \nabla_{\hat{\mathbf{R}}_t} L_t(\hat{\mathbf{R}}_t)\right)\right),$$

where  $\mathbf{skew}(\cdot)$  is the skew-symmetrization operator on the matrices,  $\mathbf{skew}(\mathbf{A}) = \mathbf{A} - \mathbf{A}^T$ . With the squared-error loss-function ( $L_t(\hat{\mathbf{R}}_t) = \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|^2$ ), the resulting updates are

$$\hat{\mathbf{R}}_{t+1} = \hat{\mathbf{R}}_t \exp\left(-2\eta \mathbf{skew}\left(\hat{\mathbf{R}}_t^T (\hat{\mathbf{y}}_t - \mathbf{y}_t) \mathbf{x}_t^T\right)\right). \quad (1)$$

It is easy to check that if  $\hat{\mathbf{R}}_t$  is a rotation matrix then  $\hat{\mathbf{R}}_{t+1}$  given by the updates in (1) is also a rotation matrix [3].

## 3 Computational complexity of updates

The updates presented in [3] (see eqn. (1) above) ensure that the estimates for the rotation matrix stay on the manifold associated with the rotation group at each iteration. However, with the matrix exponentiation at each step, the updates are computationally intensive and in fact the computational complexity of the updates is comparable to other approaches that would require repeated approximation and projection on to the manifold. The next result discusses a complexity reduction result to establish a simpler update by exploiting the eigen-structure of the update matrix [2].

**Theorem 3.1.** *The update proposed in eqn. (1) reduce to the following update,*

$$\hat{\mathbf{R}}_{t+1} = \hat{\mathbf{R}}_t \left( \mathbf{I} + \frac{\sin(\lambda)}{\lambda} \mathbf{S} + \frac{1 - \cos(\lambda)}{\lambda^2} \mathbf{S}^2 \right), \quad (2)$$

where  $\mathbf{S} = -2\eta \mathbf{skew}\left(\hat{\mathbf{R}}_t^T (\hat{\mathbf{y}}_t - \mathbf{y}_t) \mathbf{x}_t^T\right)$  is the skew-symmetric matrix in eqn. (1) with eigenvalues  $\pm j\lambda$ , for  $\lambda = 2\eta \sqrt{1 - (\hat{\mathbf{y}}_t^T \mathbf{y}_t)^2}$ .

*Proof.* First observe that the matrix  $\mathbf{S}$  can be written as

$$\begin{aligned} \mathbf{S} &= -2\eta \mathbf{skew}\left(\hat{\mathbf{R}}_t^T (\hat{\mathbf{y}}_t - \mathbf{y}_t) \mathbf{x}_t^T\right), \\ &= -2\eta \mathbf{skew}\left(\hat{\mathbf{R}}_t^T (\hat{\mathbf{R}}_t \mathbf{x}_t - \mathbf{R}_* \mathbf{x}_t) \mathbf{x}_t^T\right), \\ &= -2\eta \mathbf{skew}\left(\mathbf{x}_t \mathbf{x}_t^T - \hat{\mathbf{R}}_t^T \mathbf{R}_* \mathbf{x}_t \mathbf{x}_t^T\right), \\ &= 2\eta \left(\hat{\mathbf{R}}_t^T \mathbf{R}_* \mathbf{x}_t \mathbf{x}_t^T - \mathbf{x}_t \mathbf{x}_t^T \mathbf{R}_*^T \hat{\mathbf{R}}_t\right), \\ &= 2\eta \left(\hat{\mathbf{R}}_t^T \mathbf{y}_t \mathbf{x}_t^T - \mathbf{x}_t \mathbf{y}_t^T \hat{\mathbf{R}}_t\right), \end{aligned}$$

where  $\mathbf{y}_t = \mathbf{R}_* \mathbf{x}_t$ . Each term in the matrix  $\mathbf{S}$  is a rank-one matrix. Thus  $\mathbf{S}$  is at most rank-two. Since  $\mathbf{S}$  is skew-symmetric, it has (at most) two eigenvalues in a complex conjugate pair  $\pm j\lambda$  (and  $n - 2$  zero eigenvalues) [6]. Furthermore, Butler shows in (2.1) of [6] that the nonzero eigenvalues of the sum of two rank-one matrices  $\mathbf{u}_1 \mathbf{v}_1^T + \mathbf{u}_2 \mathbf{v}_2^T$  can be expressed in closed form as

$$\tilde{\lambda} = \frac{1}{2} \left( \mathbf{v}_1^T \mathbf{u}_1 + \mathbf{v}_2^T \mathbf{u}_2 \pm \sqrt{(\mathbf{v}_1^T \mathbf{u}_1 - \mathbf{v}_2^T \mathbf{u}_2)^2 + 4(\mathbf{v}_1^T \mathbf{u}_2)(\mathbf{v}_2^T \mathbf{u}_1)} \right). \quad (3)$$

For matrix  $\mathbf{S}$ , write  $\mathbf{u}_1 = 2\eta \hat{\mathbf{R}}_t^T \mathbf{y}_t$ ,  $\mathbf{v}_1 = \mathbf{x}_t$ ,  $\mathbf{u}_2 = \mathbf{x}_t$ , and  $\mathbf{v}_2 = -2\eta \hat{\mathbf{R}}_t^T \mathbf{y}_t$ . Then the first term in parentheses in eqn. (3) can be written as

$$\begin{aligned} \mathbf{v}_1^T \mathbf{u}_1 + \mathbf{v}_2^T \mathbf{u}_2 &= 2\eta (\mathbf{x}_t^T \hat{\mathbf{R}}_t^T \mathbf{y}_t - \mathbf{y}_t^T \hat{\mathbf{R}}_t \mathbf{x}_t) \\ &= 2\eta (\hat{\mathbf{y}}_t^T \mathbf{y}_t - \mathbf{y}_t^T \hat{\mathbf{y}}_t) \\ &= 0 \end{aligned}$$

And the term  $4(\mathbf{v}_1^T \mathbf{u}_2)(\mathbf{v}_2^T \mathbf{u}_1)$  in (3) can be simplified as

$$\begin{aligned} 4(\mathbf{v}_1^T \mathbf{u}_2)(\mathbf{v}_2^T \mathbf{u}_1) &= 4(\mathbf{x}_t^T \mathbf{x}_t)(-4\eta^2 \mathbf{y}_t^T \hat{\mathbf{R}}_t \hat{\mathbf{R}}_t^T \mathbf{y}_t) \\ &= -16\eta^2 \mathbf{y}_t^T \mathbf{y}_t \\ &= -16\eta^2 \end{aligned}$$

Then the eigenvalues given by eqn. (3) are

$$\begin{aligned}
\tilde{\lambda} &= \pm \frac{1}{2} \sqrt{\left(2\eta(\mathbf{x}_t^T \hat{\mathbf{R}}_t^T \mathbf{y}_t + \mathbf{y}_t^T \hat{\mathbf{R}}_t \mathbf{x}_t)\right)^2 - 16\eta^2} \\
&= \pm \frac{1}{2} \sqrt{4\eta^2 (\hat{\mathbf{y}}_t^T \mathbf{y}_t + \mathbf{y}_t^T \hat{\mathbf{y}}_t)^2 - 16\eta^2} \\
&= \pm \eta \sqrt{(2 \hat{\mathbf{y}}_t^T \mathbf{y}_t)^2 - 4} \\
&= \pm 2\eta \sqrt{(\hat{\mathbf{y}}_t^T \mathbf{y}_t)^2 - 1} \\
&= \pm j 2\eta \sqrt{1 - (\hat{\mathbf{y}}_t^T \mathbf{y}_t)^2}
\end{aligned}$$

In order to simplify the exponential of  $\mathbf{S}$ , a generalization of the Rodrigues' formula from [8] is useful. When  $\mathbf{S}$  is skew-symmetric and rank 2

$$e^{\mathbf{S}} = \mathbf{I} + \frac{\sin(\lambda)}{\lambda} \mathbf{S} + \frac{1 - \cos(\lambda)}{\lambda^2} \mathbf{S}^2$$

where  $\lambda = 2\eta \sqrt{1 - (\hat{\mathbf{y}}_t^T \mathbf{y}_t)^2}$ .  $\square$

Owing to the result above the matrix exponential reduces to a simple quadratic form involving an element from the Lie algebra of the rotation group. The simplified update at instance  $t$  is given in Algorithm 1.

---

**Input:** Old estimate  $\hat{\mathbf{R}}_t$ , step size  $\eta$ , data  $\mathbf{x}_t, \mathbf{y}_t \in \mathbb{R}^n$   
**Output:** Revised estimate  $\hat{\mathbf{R}}_{t+1}$  of the rotation matrix  
Prediction:  $\hat{\mathbf{y}}_t = \hat{\mathbf{R}}_t \mathbf{x}_t$ ;  
Loss:  $L_t(\hat{\mathbf{R}}_t) = \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|^2$ ;  
Compute matrix:  $\mathbf{S} = 2\eta \left( \hat{\mathbf{R}}_t^T \mathbf{y}_t \mathbf{x}_t^T - \mathbf{x}_t \mathbf{y}_t^T \hat{\mathbf{R}}_t \right)$ ;  
Compute eigenvalues:  $\lambda = 2\eta \sqrt{1 - (\mathbf{y}_t^T \hat{\mathbf{y}}_t)^2}$ ;  
New estimate:  $\hat{\mathbf{R}}_{t+1} = \hat{\mathbf{R}}_t \left( \mathbf{I} + \frac{\sin(\lambda)}{\lambda} \mathbf{S} + \frac{1 - \cos(\lambda)}{\lambda^2} \mathbf{S}^2 \right)$

**Algorithm 1:** Update at instance  $t$

---

## 4 Stability of updates

We now establish the Lyapunov stability of the proposed updates and show that Frobenius norm is a Lyapunov function. First note that the iteration (1) has an equilibrium at  $\mathbf{R}_*$  since

$$\begin{aligned}
\mathbf{S} &= 2\eta \left( \hat{\mathbf{R}}_t^T \mathbf{y}_t \mathbf{x}_t^T - \mathbf{x}_t \mathbf{y}_t^T \hat{\mathbf{R}}_t \right) \\
&= 2\eta \left( \hat{\mathbf{R}}_t^T \mathbf{R}_* \mathbf{x}_t \mathbf{x}_t^T - \mathbf{x}_t \mathbf{x}_t^T \mathbf{R}_*^T \hat{\mathbf{R}}_t \right) \quad (4)
\end{aligned}$$

is equal to  $\mathbf{0}_n$  (the  $n \times n$  matrix of all zeroes) when  $\hat{\mathbf{R}}_t = \mathbf{R}_*$ . Since  $\mathbf{exp}(\mathbf{0}_n) = \mathbf{I}$ ,  $\hat{\mathbf{R}}_{t+1} = \hat{\mathbf{R}}_t$ , and the iteration remains fixed.

A Lyapunov function [9] for a discrete iteration (such as (1)) about its equilibrium  $\mathbf{R}_*$  is a function  $V(\cdot)$  with the properties: (a)  $V(\mathbf{R}_* - \hat{\mathbf{R}}_t)$  is positive definite, (b)  $V(0) = 0$ , and (c)  $V(\mathbf{R}_* - \hat{\mathbf{R}}_{t+1}) \leq V(\mathbf{R}_* - \hat{\mathbf{R}}_t)$  for all  $t$ . The equilibrium is Lyapunov stable if there exists such a  $V$ .

**Theorem 4.1.** *The Frobenius norm  $V(\mathbf{R}) = \|\mathbf{R}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |\mathbf{R}_{ij}|^2} = \sqrt{\text{tr}(\mathbf{R}^T \mathbf{R})}$  is a Lyapunov function for the algorithm (1) about its equilibrium  $\mathbf{R}_*$ .*

Since Theorem 4.1 holds for all initial states  $\mathbf{R}_0$  and all possible input (unit) vectors  $\mathbf{x}_t$ , the result is a global Lyapunov stability. The Lyapunov stability is also evident from the simulation results for the noise-less setting in [3, 2].

## 5 Experimental results

This section discusses estimation performance of the proposed algorithm. We first focus on learning 3D rotations with noisy observations. Given, an input vector  $\mathbf{x}_t \in \mathbb{R}^3$ , the rotated vector  $\mathbf{y}_t$  is sampled from a Kent distribution [10] with mean equal to the true rotated vector  $\mathbf{R}_* \mathbf{x}_t$ ,

$$\mathbf{y}_t \sim \text{Kent}(\mathbf{R}_* \mathbf{x}_t, \gamma_2, \gamma_3, \kappa, \beta). \quad (5)$$

The parameter  $\kappa$  controls the concentration of the distribution about the mean (larger the  $\kappa$ , more concentrated the distribution) and  $\beta$  controls the ellipticity of equal probability contour ( $\gamma_2, \gamma_3 \in \mathbb{R}^3$  determine the major and minor axes of the elliptical equal probability contours). We fix  $\beta = 0$  in our simulations which results in circular equal probability contours.

Figure 1 shows the estimation error (averaged over 100 realizations) in terms of the Frobenius norm of the difference of the true 3D rotation matrix and the estimated rotation matrix using the updates in Algorithm (1) for various step sizes and  $\kappa = (10, 20, 50)$ . The simulated noise models coarse initial registration for 3D objects which is typical in applications like face recognition where it is easier to identify facial landmarks (like nose or mouth regions) in a pair of images than to establish exact registration (of nose tips, for instance). It is evident from the plots that for fixed step sizes, the noisy-observations can cause the estimation error to level-out rather than decrease at every step. For smaller step sizes a lower noise floor is achieved but requires a much larger number of instances to learn. Choosing step-size  $\eta_t = 1/t$ , so as to give a relatively proportional weight to each new observation leads to faster convergence and lower noise floor. This is intuitive since we are averaging over more and more data.

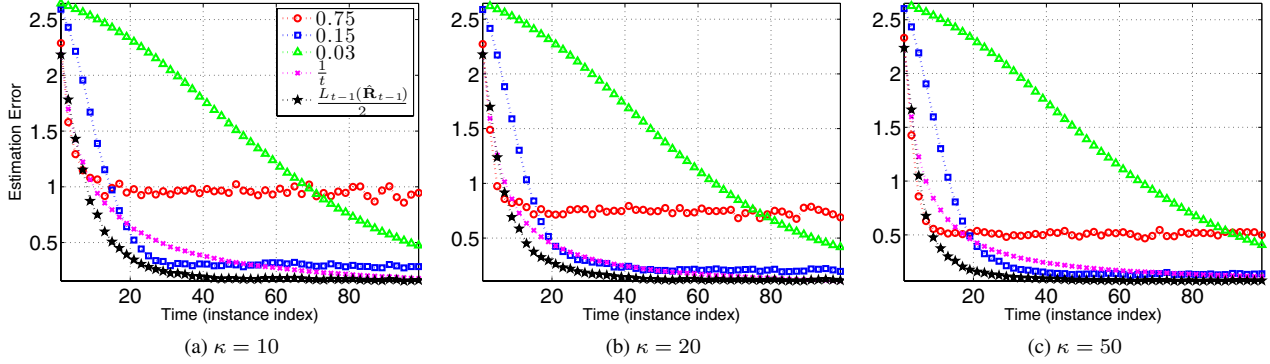


Figure 1: Estimation performance for constant step sizes (0.75, 0.15, 0.03) and variable step sizes.

Another ‘good’ choice for the sequence of step-sizes is one that adapts to the confidence in estimates of the rotation matrix; the adaptive step size at each instance is chosen proportional to the loss incurred at the previous instance ( $\eta_t = L_{t-1}(\hat{\mathbf{R}}_{t-1})/2$ ) in early part of the learning and a fixed small step-size is chosen when the observed loss drops below a threshold. Details of the experimental setup, including simulation code, can be found at [4].

Simulations in higher dimension are done with projected Gaussian noise. Consider the spherical projection map  $\pi : \mathbb{R}^n \rightarrow \mathbf{S}^{n-1}$  that projects a vector  $\mathbf{x} \in \mathbb{R}^n$  to  $\pi(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|} \in \mathbf{S}^{n-1}$ . Given, an input vector  $\mathbf{x}_t \in \mathbb{R}^n$ , the noisy rotated vector  $\mathbf{y}_t$  is given as  $\mathbf{y}_t = \pi(\mathbf{R}_* \mathbf{x}_t + \mathbf{w}_t)$ , where  $\mathbf{w}_t \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ . Figure 2 plots estimation error (averaged over 100 realizations) when learning rotation matrices  $\mathbf{R}_* \in \mathbb{R}^{10 \times 10}$  with the noise level corresponding to  $\sigma = 0.08$ .

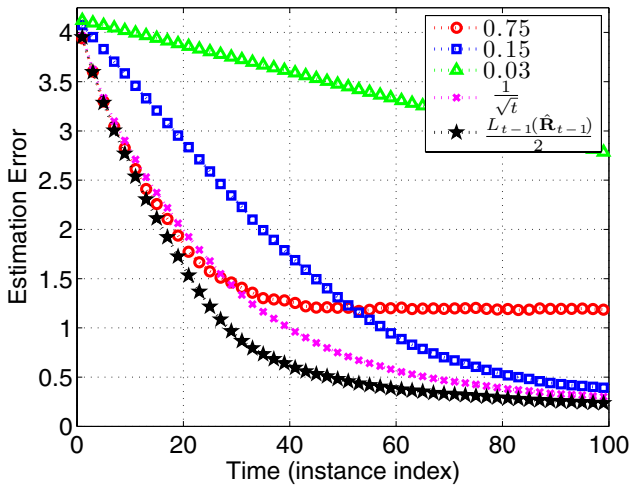


Figure 2: Learning  $\mathbf{R}_* \in \mathbf{SO}(10)$ ,  $\sigma = 0.08$ .

## 6 Conclusion

This paper proved a complexity reduction result for online learning of rotations and also established Lyapunov stability of the simplified updates. The application of the algorithm to tracking rotations and registration of point-clouds in  $n$ -dimensional Euclidean space is discussed with emphasis on the choice of step-size for the gradient updates.

## References

- [1] T. E. Abruđan, J. Eriksson, and V. Koivunen. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56:1134–1147, March 2008.
- [2] R. Arora. *Group theoretical methods in signal processing: learning similarities, transformations and invariants*. PhD thesis, Univ. of Wisconsin-Madison, 2009.
- [3] R. Arora. On learning rotations. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.
- [4] R. Arora. Online Supplementary Material. <http://faculty.washington.edu/rmnarora/ICPR/>, 2010.
- [5] P. Besl and N. McKay. A method for registration of 3D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.
- [6] L. J. Butler. *Applications of Matrix Theory to Approximation Theory*. MS Thesis, Texas Tech Univ., 1999.
- [7] S. Fiori. Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial. *Journal of Machine Learning Research*, 6:743–781, 2005.
- [8] J. Gallier and D. Xu. Computing exponentials of skew-symmetric matrices and logarithms of orthogonal matrices. *Int. J. Robotics Automation*, 17(4), 2002.
- [9] W. Hahn. *Stability of Motion*. Springer-Verlag, 1967.
- [10] J. T. Kent. The Fisher-Bingham distribution on the sphere. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(1):71–80, 1982.
- [11] P. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, Mar 1966.
- [12] A. Smith and M. Warmuth. Learning rotations. In *Conference on Learning Theory*, Jun 2008.