Probabilistic Permutation Synchronization using the Riemannian Structure of the Birkhoff Polytope - Supplementary Material

Tolga Birdal [1, 2] Umut Şimşekli [3]
[1] Computer Science Department, Stanford University, CA 94305 Stanford, US
[2] Fakultät für Informatik, Technische Universität München, 85748 München, Germany
[3] LTCI, Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France

1. Riemannian Limited Memory BFGS

We now explain the R-LBFGS optimizer used in our work. To begin with, we recall the foundational BFGS [9], that is a quasi Newton method operating in the Euclidean space. We then review the simple Riemannian descent algorithms that employ line-search. Finally, we explain the R-BFGS used to solve our synchronization problem. R-BFGS can be modified slightly to arrive at the desired limited memory Riemannian BFGS solver.

1.1. Euclidean BFGS

The idea is to approximate the true hessian $\mathbf{H}$ with $\boldsymbol{\mathcal{B}}$, using updates specified by the gradient evaluations[1]. We will then transition from this Euclidean space line search method to Riemannian space optimizers. For clarity, in Alg. 1 we summarize the Euclidean BFGS algorithm, that computes $\boldsymbol{\mathcal{B}}$ by using the most recent values of the past iterates. Note that many strategies exist to initialize $\boldsymbol{\mathcal{B}}_0$, while a common choice is the scaled identity matrix $\boldsymbol{\mathcal{B}}_0 = \gamma\mathbf{I}$. Eq. 1 corresponds to the particular BFGS-update rule. In the limited-memory successor of BFGS, the L-BFGS, the Hessian matrix $\mathbf{H}$ is instead approximated up to a pre-specified rank in order to achieve linear time and space-complexity in the dimension of the problem.

Algorithm 1: Euclidean BFGS

input: A real-valued, differentiable potential energy U, initial iterate $\mathbf{X}_0$ and initial Hessian approximation $\boldsymbol{\mathcal{B}}_0$.
$k \leftarrow 0$
while $\mathbf{x}_k$ *does not sufficiently minimize* f **do**
  Compute the direction $\boldsymbol{\eta}_k$ by solving $\boldsymbol{\mathcal{B}}_k\boldsymbol{\eta}_k = -\nabla U(\mathbf{x}_k)$ for $\boldsymbol{\eta}_k$.
  Define the new iterate $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \boldsymbol{\eta}_k$.
  Set $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k \leftarrow \nabla U(\mathbf{x}_{k+1}) - \nabla U(\mathbf{x}_k)$.
  Compute the new Hessian approximation:

$$\boldsymbol{\mathcal{B}}_{k+1} = \boldsymbol{\mathcal{B}}_k + \frac{\mathbf{y}_k\mathbf{y}_k^\top}{\mathbf{y}_k^\top\mathbf{s}_k} - \frac{\boldsymbol{\mathcal{B}}_k\mathbf{s}_k\mathbf{s}_k^\top\boldsymbol{\mathcal{B}}_k}{\mathbf{s}_k^\top\boldsymbol{\mathcal{B}}_k\mathbf{s}_k}. \tag{1}$$

  $k \leftarrow k + 1$.

1.2. Riemannian Descent and Line Search

The standard descent minimizers can be extended to operate on Riemannian manifolds $\mathcal{M}$ using the geometry of the parameters. A typical Riemannian update can be characterized as:

$$\mathbf{x}_{k+1} = R_{\mathbf{x}_k}(\tau_k\boldsymbol{\eta}_k) \tag{2}$$

[1] Note that certain implementations can instead opt to approximate the inverse Hessian for computational reasons.

where R is the retraction operator, i.e. the smooth map from the tangent bundle $\mathcal{TM}$ to $\mathcal{M}$ and $R_{\mathbf{x}_k}$ is the restriction of R to $\mathcal{T}_{\mathbf{x}_k}$, the tangent space of the current iterate $\mathbf{x}_k$, $R_{\mathbf{x}_k} : \mathcal{T}_{\mathbf{x}_k} \to \mathcal{M}$. The descent direction is defined to be on the tangent space $\boldsymbol{\eta}_k \in \mathcal{T}_{\mathbf{x}_k}\mathcal{M}$. When manifolds are rather simple shapes, the size of the step τ_k can be a fixed value. However, for most matrix manifolds, some form of a line search is preferred to compute τ_k. The retraction operator is used to take steps on the manifold and is usually derived analytically. When this analytic map is length-preserving, it is called *true* exponential map. However, such exactness is not a requirement for the optimizer and as it happens in the case of doubly stochastic matrices, $R_{\mathbf{x}_k}$ only needs to be an approximate *retraction*, e.g. first or second order. In fact, for a map to be valid retraction, it is sufficient to satisfy the following conditions:

1. R is continuously differentiable.
2. $R_{\mathbf{x}}(\mathbf{0}_{\mathbf{x}}) = \mathbf{x}$, where $\mathbf{0}_{\mathbf{x}}$ denotes the zero element of $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. This is called the *centering* property.
3. The curve $\gamma_{\boldsymbol{\eta}_{\mathbf{x}}}(\tau) = R_{\mathbf{x}}(\tau\boldsymbol{\eta}_{\mathbf{x}})$ satisfies:

$$\frac{d\gamma_{\boldsymbol{\eta}_{\mathbf{x}}}(\tau)}{d\tau}\Big|_{\tau=0} = \boldsymbol{\eta}_{\mathbf{x}},\ \forall \boldsymbol{\eta}_{\mathbf{x}} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}. \tag{3}$$

This is called the *local rigidity* property.

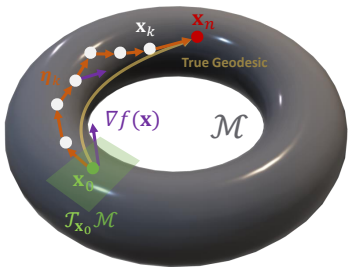


Figure 1. Visualization of the entities used on a sample toroidal manifold.

Considering all these, we provide, in Alg. 2, a general Riemannian descent algorithm, which can be customized by the choice of the direction, retraction and the means to compute the step size. Such a concept of minimizing by walking on the manifold is visualized in Fig. 1.

Algorithm 2: General Riemannian Line Search Minimizer

input: A Riemannian manifold $\mathcal{M}$, a retraction operator R and initial iterate $\mathbf{x}_k \in \mathcal{M}$ where $k = 0$.
while $\mathbf{x}_k$ *does not sufficiently minimize* f **do**
  Pick a gradient related descent direction $\boldsymbol{\eta}_k \in \mathcal{T}_{\mathbf{x}_k}\mathcal{M}$.
  Choose a retraction $R_{\mathbf{x}_k} : \mathcal{T}_{\mathbf{x}_k}\mathcal{M} \to \mathcal{M}$.
  Choose a step length $\tau_k \in \mathbb{R}$.
  Set $\mathbf{x}_{k+1} \leftarrow R_{\mathbf{x}_k}(\tau_k\boldsymbol{\eta}_k)$.
  $k \leftarrow k + 1$.

It is possible to develop new minimizers by making particular choices for the Riemannian operators in Alg 2. We now review the Armijo variant of the Riemannian gradient descent [10], that is a common and probably the simplest choice for an accelerated optimizer on the manifold. Though, many other line-search conditions such as Barzilai-Borwein [14] or strong Wolfe [20] can be used. The pseudocode for this backtracking version is given in Alg. 3. Note that the Riemannian gradient $\text{grad} f(\mathbf{x})$ is simply obtained by projecting the Euclidean gradient $\nabla f(\mathbf{x})$ onto the manifold $\mathcal{M}$ and the next iterate is obtained through a line search so as to satisfy the Armijo condition [3], tweaked to use the retraction R for taking steps. For some predefined Armijo step size, this algorithm is guaranteed to converge for all retractions [1].

1.3. LR-BFGS for Minimization on $\mathcal{DP}_n$

Based upon the ideas developed up to this point, we present the Riemannian variant of the L-BFGS. The algorithm is mainly based on Huang *et al*. and we refer the reader to their seminal work for more details [13]. It is also worth noting [18]. Similar to the Euclidean case, we begin by summarizing a BFGS algorithm with the difference that it is suited to solving the synchronization task. This time, $\boldsymbol{\mathcal{B}}$ will approximate the action of the Hessian on the tangent space $\mathcal{T}_{\mathbf{x}_k}\mathcal{M}$. Generalizing any (quasi-)Newton method to the Riemannian setting thus requires computing the Riemannian Hessian operator, or its approximation. This necessitates taking a some sort of a directional derivative of a vector field. As the vector fields belonging to different tangent spaces cannot be compared immediately, one needs the notion of connection Γ generalizing the directional derivative of a vector field. This connection is closely related to the concept of vector transport $T : \mathcal{TM} \otimes \mathcal{TM} \to \mathcal{TM}$ which allows moving from one tangent space to the other $T_{\boldsymbol{\eta}}(\boldsymbol{\zeta}) : (\boldsymbol{\eta}, \boldsymbol{\zeta}) \in \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{T}_{R_{\mathbf{x}}(\boldsymbol{\eta})}\mathcal{M}$. For the first order treatment of the Birkhoff Polytope, this vector transport takes a simple form:

$$T_{\boldsymbol{\eta}}(\zeta) \in \mathcal{T}_{R_{\mathbf{x}}(\boldsymbol{\eta})}\mathcal{DP}_n \triangleq \Pi_{R_{\mathbf{x}}(\boldsymbol{\eta})}(\boldsymbol{\zeta}). \tag{5}$$

Algorithm 3: General Riemannian Steepest Descent with Armijo Line Search

input: A Riemannian manifold $\mathcal{M}$, a retraction operator R, the projection operator onto the tangent space $\Pi_{\mathbf{x}_k} : \mathbb{R}^n \rightarrow \mathcal{T}_{\mathbf{x}_k}\mathcal{M}$, a real-valued, differentiable potential energy f, initial iterate $\mathbf{x}_0 \in \mathcal{M}$ and the Armijo line search scalars including c.

while $\mathbf{x}_k$ *does not sufficiently minimize* f **do**

// Euclidean gradient to Riemannian direction

$\boldsymbol{\eta}_k \leftarrow -\text{grad} f(\mathbf{x}_k) \triangleq \Pi_{\mathbf{x}_k}(-\nabla f(\mathbf{x}_k))$.

Select $\mathbf{x}_{k+1}$ such that:

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq c\big(f(\mathbf{x}_k) - f(R_{\mathbf{x}}(\tau_k \boldsymbol{\eta}_k))\big), \tag{4}$$

where τ_k is the Armijo step size.

$k \leftarrow k + 1$.

where $\Pi_{\mathbf{X}}(\cdot)$ is the projection onto the tangent space of $\mathbf{X}$ as defined in the main paper. We give the pseudocode of the R-BFGS algorithm in Alg. 4 below. To ensure Riemannian L-BFGS always produces a descent direction, it is necessary to adapt a line-search algorithm which satisfies strong Wolfe (SW) conditions [22, 23]. Roptlib [12] does implement the SW while ManOpt [4] uses the simpler Armijo conditions, even for the LR-BFGS. Another simple possibility is to take the steps on the Manifold using the retraction, while performing the line search in the Euclidean space. This results in a projected, approximate line search, but can terminate quite quickly, making it possible to exploit existing Euclidean space SW LBFGS solvers such as Ceres [2]. Without delving into rigorous proofs, we provide one such implementation at `github.com/tolgabirdal/MatrixManifoldsInCeres` where several matrix manifolds are considered. We leave the analysis of such approximations for a future study and note that the results in the paper are generated by limited memory form of the R-BFGS procedure summarized under Alg. 4. While many modifications do exist, LR-BFGS, in essence, is an approximation to R-BFGS, where $O(MN^2)$ storage of the full Hessian is avoided by unrolling the RBFGS descent computation and using the last m input and gradient differences where $m << MN^2$. This allows us to handle large data regimes.

2. Proof of Proposition 1 and Further Analysis

Proof. Consider the ray cast outwards from the origin: $\mathbf{r} = \mathbb{R}_{\geq 0}\mathbb{1}$. $\mathbf{r}$ exits $\mathcal{DP}_n$ at $\frac{1}{n}\mathbb{1}$ but exits the sphere $\mathcal{S}^{n-1}$ at $\frac{1}{\sqrt{n}}\mathbb{1}$. This creates a gap between the two manifolds whose ratio grows to ∞ as n grows. For a perspective of optimization, consider the linear function $\lambda(\mathbf{P}) = \sum_{i,j} P_{ij}$. $\lambda(\mathbf{P})$ is minimized on $\mathcal{DP}_n$ at $\frac{1}{n}\mathbb{1}$ and on the sphere at $\frac{1}{\sqrt{n}}\mathbb{1}$, □

We now look at the restricted orthogonal matrices and seek to find the difference between optimizing a linear functional on them and $\mathcal{DP}_n$. Note that minimizing functionals is ultimately what we are interested in as the problems we consider here are formulated as optimization. Let $\mathcal{A}$ be the affine linear space of $(n-1)\times(n-1)$ matrices whose rows and columns sum to 1, *i.e.* doubly stochastic but with no condition on the signs or *a generalized doubly stochastic matrix*. The Birkhoff Polytope $\mathcal{DP}_n$ is contained in $\mathcal{A}$, whereas the orthogonal group $\mathcal{O}_n$ is not. So, there exists an affine functional λ that is minimized at a point on $\mathcal{DP}_n$, $\lambda = 0$ but not on $\mathcal{O}$. Let $\mathcal{AO}_n = \mathcal{DP}_n \cap \mathcal{O}_n$, a further restricted manifold. This time unlike the case of $\mathcal{DP}_n$, $\mathcal{AO}_n$ would not coincide the permutations $\mathcal{P}$ due to the negative elements. In fact $\mathcal{P} \subset \mathcal{AO}_n$.

Proposition 3. *The ratio between the time a ray leaves $\mathcal{DP}_n$ and the same ray leaves $\mathcal{AO}_n$ can be as large as $n-1$.*

Proof. Consider the line through $\frac{1}{n}\mathbb{1}$ and $\mathbf{I} : l(x) = \frac{1+x}{n}\mathbb{1} - x\mathbf{I}$. Such a ray leaves $\mathcal{DP}_n$ at $x = \frac{1}{n-1}$ and for all $x \in [-1, 1]$ is in the convex hull of $\mathcal{AO}_n$. When $x = 1$, it is an orthogonal matrix, *i.e.* on $\mathcal{O}_n$. Hence, the ray can be on $\mathcal{AO}_n$ for $n-1$ times as long as in $\mathcal{DP}_n$. This quantity also grows to infinity as $n \rightarrow \infty$. If same analysis is done for the case of the sphere, the ratio is found to be $(n-1)^{3/2}$, grows quicker to infinity and is a larger quantity. □

Algorithm 4: Riemannian BFGS for Synchronization on $\mathcal{DP}_n$

input: Birkhoff Polytope $\mathcal{DP}$ with Riemannian (Fisher information) metric g, first order retraction R, the parallel transport T, a real-valued, differentiable potential energy function of synchronization U, initial iterate $\mathbf{X}_0$ and initial Hessian approximation $\boldsymbol{\mathcal{B}}_0$.

while $\mathbf{x}_k$ *does not sufficiently minimize* f **do**

Compute the Euclidean gradients using:

$$\nabla_{\mathbf{X}_i} U(\mathbf{X}) = \sum_{(i,j)\in E} -2(\mathbf{P}_{ij} - \mathbf{X}_i\mathbf{X}_j^\top)\mathbf{X}_j \qquad \nabla_{\mathbf{X}_j} U(\mathbf{X}) = \sum_{(i,j)\in E} -2(\mathbf{P}_{ij} - \mathbf{X}_i\mathbf{X}_j^\top)\mathbf{X}_i \tag{6}$$

Compute the Riemannian gradient $\operatorname{grad} U(\mathbf{X}_k) \leftarrow^{\triangle} \Pi_{\mathbf{X}_k}(-\nabla U(\mathbf{X}_k))$.

Compute the direction $\boldsymbol{\eta}_k \in \mathcal{T}_{\mathbf{X}_k}\mathcal{DP}_n$ by solving $\boldsymbol{\mathcal{B}}_k\boldsymbol{\eta}_k = -\operatorname{grad} U(\mathbf{X}_k)$.

Given $\boldsymbol{\eta}_k$, execute a line search satisfying the strong Wolfe conditions [22, 23, 13]. Set step size τ_k.

Set $\mathbf{x}_{k+1} = R_{\mathbf{x}_k}(\tau_k\boldsymbol{\eta}_k)$.

Use vector transport to define:

$$\mathbf{S}_k = T_{\tau_k\boldsymbol{\eta}_k}(\tau_k\boldsymbol{\eta}_k)\,, \qquad \mathbf{Y}_k = \operatorname{grad} U(\mathbf{X}_{k+1}) - T_{\tau_k\boldsymbol{\eta}_k}(\operatorname{grad} U(\mathbf{x}_k)). \tag{7}$$

Compute $\tilde{\boldsymbol{\mathcal{B}}}_k = T_{\tau_k\boldsymbol{\eta}_k} \circ \boldsymbol{\mathcal{B}}_k \circ T^{+}_{\tau_k\boldsymbol{\eta}_k}$ where T^+ is denotes (pseudo-)inverse of the transport.

Compute the linear operator $\boldsymbol{\mathcal{B}}_{k+1} : \mathcal{T}_{\mathbf{x}_{k+1}}\mathcal{M} \to \mathcal{T}_{\mathbf{x}_{k+1}}\mathcal{M}$:

$$\tilde{\boldsymbol{\mathcal{B}}}_{k+1}\mathbf{Z} = \tilde{\boldsymbol{\mathcal{B}}}_k\mathbf{Z} + \frac{g(\mathbf{Y}_k, \mathbf{Z})}{g(\mathbf{y}_k, \mathbf{S}_k)}\mathbf{Y}_k - \frac{g(\mathbf{S}_k, \tilde{\boldsymbol{\mathcal{B}}}\mathbf{Z})}{g(\mathbf{S}_k, \tilde{\boldsymbol{\mathcal{B}}}\mathbf{S}_k)} \qquad \forall \mathbf{Z} \in \mathcal{T}_{\mathbf{x}_{k+1}}\mathcal{DP}_n. \tag{8}$$

$k \leftarrow k + 1$.

3. Proof of Proposition 2

Proof. The conclusion of the proposition states that $p(\mathbf{X})$ and $p(\mathbf{P}|\mathbf{X})$ have the following form:

$$p(\mathbf{X}) = \frac{1}{C}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \|\mathbf{X}_{ij}\|_{\mathrm{F}}^2\Big) \prod_{(i,j)\in\mathcal{E}} Z_{ij} \tag{9}$$

$$p(\mathbf{P}|\mathbf{X}) = \prod_{(i,j)\in\mathcal{E}} p(\mathbf{P}_{ij}|\mathbf{X}_i, \mathbf{X}_j) \tag{10}$$

$$= \prod_{(i,j)\in\mathcal{E}} \exp\Big(2\beta\,\mathrm{tr}(\mathbf{P}_{ij}^\top\mathbf{X}_{ij})\Big)\frac{1}{Z_{ij}} \tag{11}$$

$$= \exp\Big(\beta \sum_{(i,j)\in\mathcal{E}} 2\mathrm{tr}(\mathbf{P}_{ij}^\top\mathbf{X}_{ij})\Big) \prod_{(i,j)\in\mathcal{E}} \frac{1}{Z_{ij}} \tag{12}$$

where

$$Z_{ij} := \prod_{b=1}^{B_{ij}} Z_{\mathbf{X}}(\beta, \theta_{ij,b}). \tag{13}$$

Our goal is to verify that the following holds with the above definitions:

$$p(\mathbf{P}, \mathbf{X}) = \frac{1}{Z}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \|\mathbf{P}_{ij} - \mathbf{X}_i\mathbf{X}_j^\top\|_{\mathrm{F}}^2\Big) = \frac{1}{Z}\prod_{(i,j)\in\mathcal{E}} \psi(\mathbf{P}_{ij}, \mathbf{X}_i, \mathbf{X}_j), \tag{14}$$

where Z is a positive constant (cf. Section 4 in the main paper). Then, we can easily verify this equality as follows:

$$
\begin{aligned}
p(\mathbf{P}, \mathbf{X}) &= p(\mathbf{P}|\mathbf{X})p(\mathbf{X}) && (15)\\
&= \frac{1}{C}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \|\mathbf{X}_{ij}\|_{\mathrm{F}}^2\Big)\exp\Big(\beta \sum_{(i,j)\in\mathcal{E}} 2\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij})\Big) && (16)\\
&= \frac{1}{C}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \big(\|\mathbf{X}_{ij}\|_{\mathrm{F}}^2 - 2\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij})\big)\Big) && (17)\\
&= \frac{1}{C}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \big(\|\mathbf{X}_{ij}\|_{\mathrm{F}}^2 - 2\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij}) - n + n\big)\Big) && (18)\\
&= \frac{1}{C}\exp(\beta n|\mathcal{E}|)\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \big(\|\mathbf{X}_{ij}\|_{\mathrm{F}}^2 - 2\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij}) + \|\mathbf{P}_{ij}\|_{\mathrm{F}}^2\big)\Big) && (19)\\
&= \frac{1}{Z}\exp\Big(-\beta \sum_{(i,j)\in\mathcal{E}} \|\mathbf{P}_{ij} - \mathbf{X}_i\mathbf{X}_j^\top\|_{\mathrm{F}}^2\Big) && (20)
\end{aligned}
$$

where we used the fact that $\|\mathbf{P}_{ij}\|_{\mathrm{F}}^2 = n$ in Equation 19 since $\mathbf{P}_{ij} \in \mathcal{P}_n$ and $Z = C\exp(-\beta n|\mathcal{E}|)$.

In the rest of the proof, we will characterize the normalizing constant $Z_{ij} = \int_{\mathcal{P}_n} \exp(2\beta \mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij})\mathrm{d}\mathbf{P}_{ij}$. Here, we denote the the counting measure on $\mathcal{P}_n$ as $\mathrm{d}\mathbf{P}_{ij}$.

We start by decomposing $\mathbf{X}_{ij}$ via Birkhoff-von Neumann theorem:

$$
\mathbf{X}_{ij} = \sum_{b=1}^{B_{ij}} \theta_{ij,b}\mathbf{M}_{ij,b}, \quad \sum_{b=1}^{B_{ij}} \theta_{ij,b} = 1, \quad B_{ij} \in \mathbb{Z}_+, \quad \theta_{ij,b} \geq 0,\ \mathbf{M}_{ij,b} \in \mathcal{P}_n, \quad \forall b = 1, \ldots, B_{ij}. \tag{21}
$$

Then, we have:

$$
\begin{aligned}
Z_{ij} &= \int_{\mathcal{P}_n} \exp\Big(2\beta \mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{X}_{ij}\Big)\mathrm{d}\mathbf{P}_{ij} && (22)\\
&= \int_{\mathcal{P}_n} \exp\Big(2\beta \mathrm{tr}(\mathbf{P}_{ij}^\top \sum_{b=1}^{B_{ij}} \theta_{ij,b}\mathbf{M}_{ij,b}\Big)\mathrm{d}\mathbf{P}_{ij} && (23)\\
&= \int_{\mathcal{P}_n} \exp\Big(2\beta \sum_{b=1}^{B_{ij}} \theta_{ij,b}\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b})\Big)\mathrm{d}\mathbf{P}_{ij} && (24)\\
&\geq \exp\Big(\int_{\mathcal{P}_n} 2\beta \sum_{b=1}^{B_{ij}} \theta_{ij,b}\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b})\mathrm{d}\mathbf{P}_{ij}\Big) && (25)\\
&= \exp\Big(\sum_{b=1}^{B_{ij}} \int_{\mathcal{P}_n} 2\beta\theta_{ij,b}\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b})\mathrm{d}\mathbf{P}_{ij}\Big) && (26)
\end{aligned}
$$

where we used Jensen's inequality in (25). Here, we observe that $\int_{\mathcal{P}_n} 2\beta\theta_{ij,b}\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b})\mathrm{d}\mathbf{P}_{ij}$ is similar to the normalization constant of a Mallows model [8], and it only depends on β and $\theta_{ij,b}$. Hence, we conclude that

$$
\begin{aligned}
Z_{ij} &\geq \prod_{b=1}^{B_{ij}} \exp\Big(\int_{\mathcal{P}_n} 2\beta\theta_{ij,b}\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b})\mathrm{d}\mathbf{P}_{ij}\Big) && (27)\\
&:= \prod_{b=1}^{B_{ij}} f(\beta, \theta_{ij,b}) && (28)
\end{aligned}
$$

where f is an increasing function of $\beta, \theta_{ij,b}$ since $\mathrm{tr}(\mathbf{P}_{ij}^\top \mathbf{M}_{ij,b}) \geq 0$. This completes the proof. □

4. Derivation of the Retraction Euler Integrator

We start by recalling the SDE

$$\mathrm{d}\tilde{\mathbf{X}}_t = (-\mathbf{G}^{-1}\nabla_{\tilde{\mathbf{X}}} U_\lambda(\tilde{\mathbf{X}}_t) + \mathbf{\Gamma}_t)\mathrm{d}t + \sqrt{2/\beta\mathbf{G}^{-1}}\mathrm{dB}_t. \tag{29}$$

By [24], we know that

$$\mathbf{\Gamma}_t = \frac{1}{2}\mathbf{G}^{-1}\nabla_{\tilde{\mathbf{X}}} \log|\mathbf{G}|. \tag{30}$$

By using this identity in Equation 29, we obtain:

$$\mathrm{d}\tilde{\mathbf{X}}_t = -\mathbf{G}^{-1}(\nabla_{\tilde{\mathbf{X}}} U_\lambda(\tilde{\mathbf{X}}_t) + \frac{1}{2}\nabla_{\tilde{\mathbf{X}}} \log|\mathbf{G}|)\mathrm{d}t + \sqrt{2/\beta\mathbf{G}^{-1}}\mathrm{dB}_t. \tag{31}$$

By using a similar notation to [15], we rewrite the above equation as follows:

$$\mathrm{d}\tilde{\mathbf{X}}_t = -\mathbf{G}^{-1}(\nabla_{\tilde{\mathbf{X}}} U_\lambda(\tilde{\mathbf{X}}_t) + \frac{1}{2}\nabla_{\tilde{\mathbf{X}}} \log|\mathbf{G}|)\mathrm{d}t + \mathbf{G}^{-1}\mathbf{M}^\top\mathcal{N}(0, 2\beta\mathbf{I}) \tag{32}$$

where $\mathcal{N}$ denotes the Gaussian distribution and $[\mathbf{M}]_{ij} = \partial[\mathbf{X}]_{ij}/\partial[\tilde{\mathbf{X}}]_{ij}$. We multiply each side of the above equation by $\mathbf{M}$ and use the property $\nabla_{\tilde{\mathbf{X}}} = \mathbf{M}^\top\nabla_{\mathbf{X}}$ [15], which yields:

$$\mathrm{d}\mathbf{X}_t = -\mathbf{M}\mathbf{G}^{-1}\mathbf{M}^\top\nabla_{\mathbf{X}} U(\mathbf{X}_t)\mathrm{d}t + \mathbf{M}\mathbf{G}^{-1}\mathbf{M}^\top\mathcal{N}(0, 2\beta\mathbf{I}) \tag{33}$$

$$= \mathbf{M}(\mathbf{M}^\top\mathbf{M})^{-1}\mathbf{M}^\top\Big(-\nabla_{\mathbf{X}} U(\mathbf{X}_t)\mathrm{d}t + \mathcal{N}(0, 2\beta\mathbf{I})\Big). \tag{34}$$

Here we used the area formula (Equation 11 in the main paper) and the fact that $\mathbf{G} = \mathbf{M}^\top\mathbf{M}$.

The term $\mathbf{M}(\mathbf{M}^\top\mathbf{M})^{-1}\mathbf{M}^\top$ turns out the be the projection operator to the tangent space of $\mathbf{X}$ [6]. Therefore, the usual geodesic integrator would consist of the following steps at iteration k:

- Set a small step size h
- Compute the term $-h\nabla_{\mathbf{X}} U(\mathbf{X}_t) + \sqrt{2h/\beta}\mathbf{Z}$, with $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{I})$
- Obtain the 'direction' by projecting the result of the previous step on the tangent space
- Move the current iterate on the geodesic determined by the direction that was obtained in the previous step.

Unfortunately, the last step of the integrator above cannot be computed in the case of $\mathcal{DP}_n$. Therefore, we replace it with moving the current iterate by using the retraction operator, which yields the update equations given in the main paper.

5. Details on the Synthetic Dataset

We now give further details on the synthetic data used in the paper. We synthesize our data by displacing a 4×4 2D grid to simulate 5 different images and scrambling the order of correspondence. Shown in Fig. 2a, the ground truth corresponds to an identity ordering where the i^{th} element of each grid is mapped to i^{th} node of the other. Note that the graph is fully connected but we show only consecutive connections. To simulate noisy data, we introduce 16 random swaps into the ground truth as shown in Fig. 2b. We also randomize the initialization in a similar way. On this data we first run a baseline method where instead of restricting ourselves to the Birkhoff Polytope, we use the paths of the orthogonal group $O(N)$. Our second baseline is the prosperous method of Maset *et al*. [17]. Note that, regardless of the initialization (Fig 2e,f) our method is capable of arriving at visually more satisfactory local minimum. This validates that for complex problems such as the one at hand, respecting the geometry of the constraints is crucial. Our algorithm is successful at that and hence is able to find high quality solutions. In the figure the more *parallel* the lines are, the better, depicting closeness to the ground truth.

6. Examples on the Willow Dataset

Finding the Optimum Solution We now show matches visualized after running our synchronization on the Willow dataset [7]. For the sake of space, we have omitted some of those results from the paper. Fig 3 plots our findings, where our algorithm is able to converge from challenging initial matches. Note that these results only show our the MAP estimates explained in the main paper. It is possible to extend our approach with some form of geometric constraints as in Wang *et al*. [21]. We leave this as a future work.

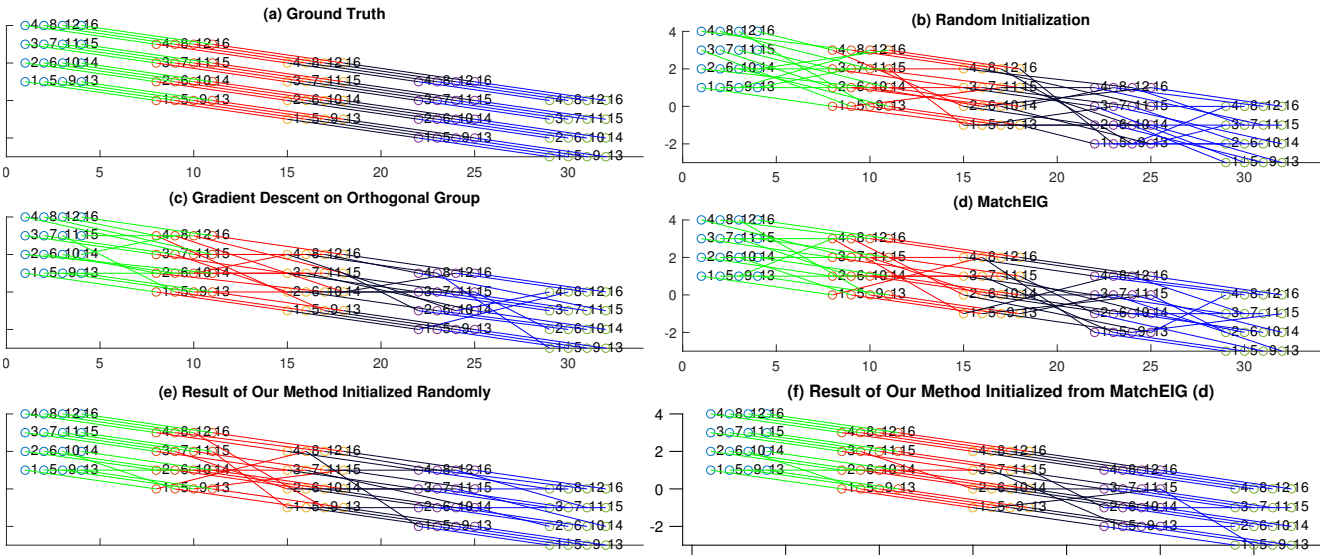


Figure 2. Results from running our method on the synthetic dataset as well as some baseline methods such as Maset *et al*. [17] and minimizing our energy on the orthogonal group. Note that while using the same energy function, considering the Birkhoff Polytope (ours) rather than the orthogonal group leads to much more visually appealing results with higher recall.

Uncertainty Estimation There are not many well accepted standards in evaluating the uncertainty. Hence in the main paper, we resorted what we refer as the *top-K* error. We will now briefly supply more details on this. The purpose of the experiment is to measure the quality of the sample proposals generated by our Birkhoff-RLMC. To this end, we introduce a random sampler that generates, K arbitrary solutions that are highly likely to be irrelevant (bad proposals). These solutions alone do not solve the problem. However if we were to consider the result correct whenever either the found optimum or the random proposal contains the correct match i.e. append the additional $K-1$ samples to the solution set, then, even with a random sampler we are guaranteed to increase the recall. Similarly, samples from Birkhoff-RLMC will also yield higher recall. The question then is: Can Birkhoff-RLMC do better than a random sampler? To give an answer, we basically record the relative improvement in recall both for the random sampler and Birkhoff-RLMC. The *top-K error* for different choices of K is what we presented in Table 2 of the main paper. We further visualize these solutions in the columns (c) to (f) of the same table. To do so, we simply retain the assignments with the K-highest scores in the solutions $\mathbf{X}$. Note that the entire $\mathbf{X}$ acts as a confidence map itself (Column d). We then use the MATLAB command *imagesc* on the $\mathbf{X}$ with the *jet* colormap.

Next, we provide insights into how the sampler works in practice. Fig. 4 plots the objective value attained as the iterations of the Birkhoff-RLMC sampler proceeds. For different values of β these plots look different. Here we use $\beta = 0.08$ and show both on Motorbike and Winebottle samples (used above) and for 1000 iterations, the behaviour of sample selection. In the paper, we have accumulated these samples and estimated the confidence maps. Note that occasionally, the sampler can discover better solutions than the one being provided. This is due to two reasons: 1) rarely, we could jump over local minima, 2) the initial solution is a discrete one and it is often plausible to have a doubly stochastic (relaxed) solution matrices that have lower cost.

7. Application to Establishing 3D Correspondences

We now apply our algorithm to solve correspondence problems on isolated 3D shapes provided by the synthetic Tosca dataset [5] that is composed of non-rigidly deforming 3D meshes of various objects such as cat, human, Centaur, dog, wolf and horse. The *cat* object from this dataset along with ten of its selected ground truth correspondences is shown in Fig. 5. This is of great concern among the 3D vision community and the availability of the complete shapes plays well with the assumptions of our algorithm, i.e. permutations are *total*. It is important to mention that when initial correspondences are good ($\sim 80\%$) all methods, including ours can obtain 100% accuracy [11]. Therefore, to be able to put our method to test, we will intentionally degrade the initial correspondence estimation algorithm we use.

Initialization Analogous to the 2D scenario, we obtain the initial correspondences by running a siamese Point-Net [19] like network regressing the assignments between two shapes. Unlike 2D, we do not need to take care of occlusions, visibility

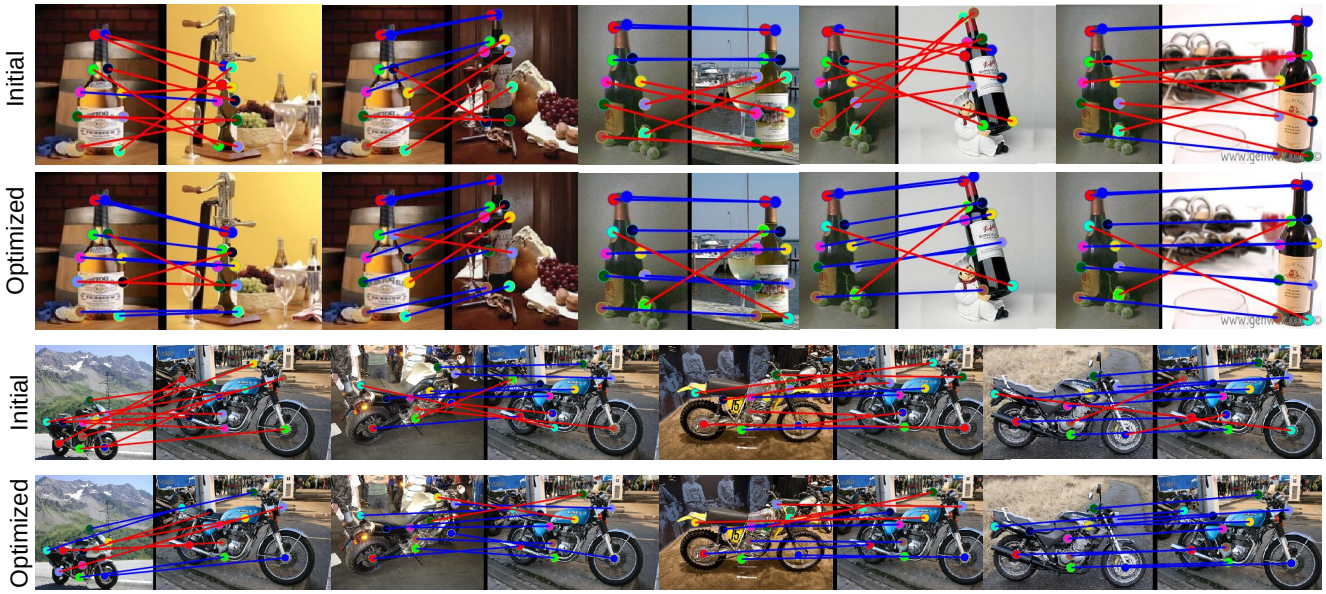
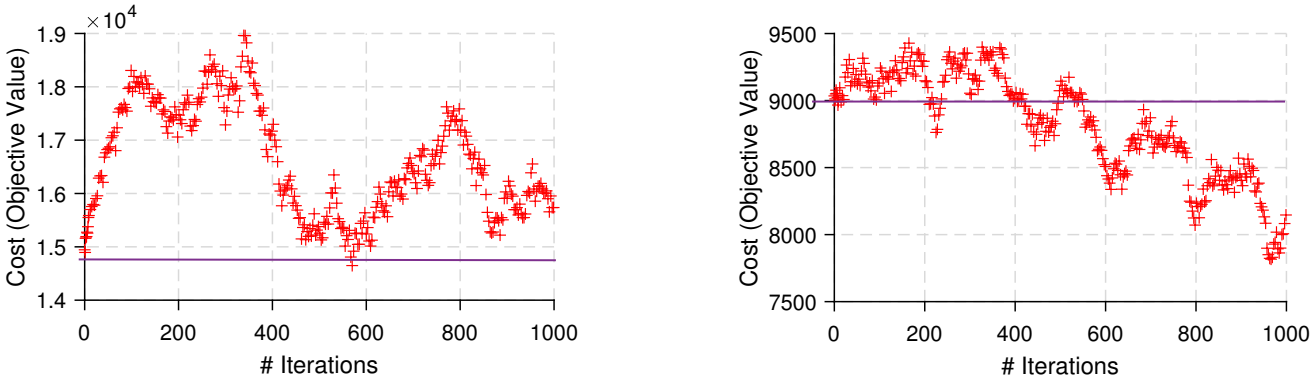


Figure 3. Matching results on sample images from the *Winebottle* (**top-2-rows**) and *Motorbike* (**bottom-2-rows**) datasets in the form of correspondences. The first sub-row of each row shows the initial matches computed via running a pairwise Hungarian algorithm, whereas the second row (tagged *Optimized*) shows our solution. Colored dots are the corresponding points. A line segment is shown in red if it is found to be wrong when checked against the ground truth. Likewise, a blue indicates a correct match. For both of the datasets we use the joint information by optimizing for the cycle consistency, whereas the pairwise solutions make no use of such multiview correspondences. Note that thanks to the cycle consistency, once a match is correctly identified, its associated point has the tendency to be correctly matched across the entire dataset.



(a) Sampling on the Winebottle

(b) Sampling on the Motorbike

Figure 4. Iterates of our sampling algorithm. With $\beta = 0.085$, our Birkhoff-RLMC sampler wanders around the local mode and draws samples on the Birkhoff Polytope. It can also happen that better solutions are found and returned. Purple line indicates the starting point, that is only a slight perturbation of the found solution.

or missing keypoint locations as 3D shapes can be full and clean as in this dataset. On the average, we split half of the datasets for training and the other half for testing. Note that such amount of data is really insufficient to train this network, resulting in suboptimal predictions. We gradually downsample the point sets uniformly to sizes of $N_s = \{200, 50, 20\}$ to get the keypoints. At this point, it is possible to use sophisticated keypoint prediction strategies to establish the optimum cardinality and the set of points under correspondence. Note that it is sufficient to compute the keypoints per shape as we do not assume the presence of a particular order. Each keypoint is matched to another by the network prediction. The final stage of the prediction results in a soft assignment matrix on which we apply Hungarian algorithm to give rise to initial matches.

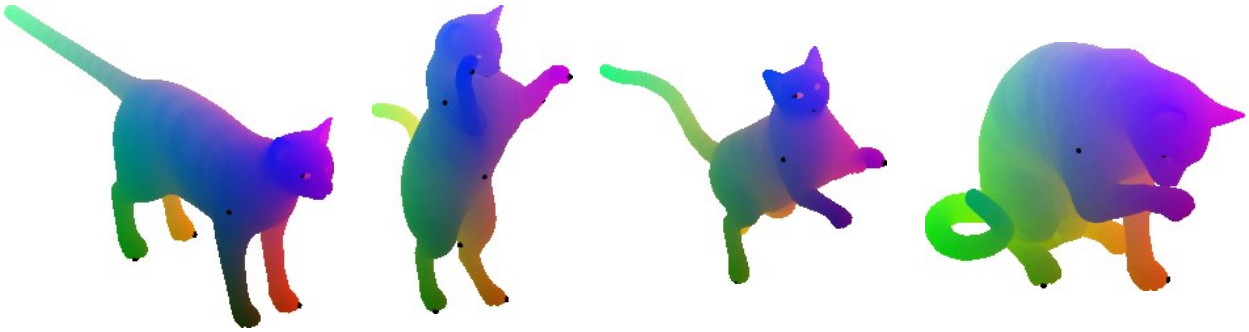


Figure 5. Visualizations of the *cat* object from the Tosca dataset with the ground truth correspondences depicted.

Quantitative Results We run, on these initial sets of correspondences, our algorithm and the state of the art methods as described in the main paper. We count and report the percentage of correctly matched correspondences (recall) in Tab. 1 for only two objects, *Cat* and *Michael*. Running on the full dataset is a future work. It is seen that our algorithm consistently outperforms the competing approaches. The difference is less visible when the number of samples start dominating the number of edges. This is because none of the algorithms are able to find enough consistency votes to correct for the errors.

Table 1. Correspondence refinement on 3D inputs. We show our results on the meshes of Tosca [5] dataset. The cells show the percentage of correctly detected matches (*recall)*.

	N=200				N=50				N=20			
	Initial	MatchEIG	Wang et al.	Ours	Initial	MatchEIG	Wang et al.	Ours	Initial	MatchEIG	Wang et al.	Ours
Cat	38.42%	32.92%	37.83%	**39.08%**	53.38%	55.42%	56.13%	**56.93%**	35.56%	37.33%	38.33%	**40.33%**
Michael	30.64%	34.39%	35.92%	**36.12%**	46.40%	52.23%	**54.93%**	54.62%	45.76%	51.05%	51.63%	**55.95%**

8. Final Words and Future Work for Extension to Partial Matches

We mention in our paper that we leave the synchronization of partial matching as a future work. Nevertheless, here we would like to present certain directions for managing this within our existing framework.

In the case of partiality, we are given an $N \times M$ partial permutation matrix with $M > N$ (or vice versa). It is known that the partial doubly stochastic projection can be converted to doubly stochastic projection by introducing slack variables, a standard technique from linear programming [16]. This is similar to soft-assigning the non-assigned features to virtual ones. The slacked $M \times M$ matrix would now live on the Birkhoff Polytope. Convergence of Sinkhorn operator is still guaranteed [16] and the new variables can be treated within our framework. The desired end result is a sub-space of this lifted, higher dimensional space, on which the optimization is carried out. Detailed analysis, application, evaluation, computational aspects and convergence proofs for such an approach (especially with inexact retractions) are still to be investigated and we leave them for future research.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[3] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[4] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15, 2014.

[5] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[6] Simon Byrne and Mark Girolami. Geodesic monte carlo on embedded manifolds. *Scandinavian Journal of Statistics*, 40(4):825–845, 2013.

[7] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 25–32, 2013.

[8] Stéphan Clémençon and Jérémie Jakubowicz. Kantorovich distances between rankings with applications to rank aggregation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010.

[9] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[10] Seyedehsomayeh Hosseini, Wen Huang, and Rohollah Yousefpour. Line search algorithms for locally lipschitz functions on riemannian manifolds. *SIAM Journal on Optimization*, 28(1):596–619, 2018.

[11] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. Eurographics Association, 2013.

[12] Wen Huang, P.-A. Absil, K. A. Gallivan, and Paul Hand. Roptlib: an object-oriented c++ library for optimization on riemannian manifolds. Technical Report FSU16-14.v2, Florida State University, 2016.

[13] Wen Huang, Kyle A Gallivan, and P-A Absil. A broyden class of quasi-newton methods for riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[14] Bruno Iannazzo and Margherita Porcelli. The riemannian barzilai–borwein method with nonmonotone line search and the matrix geometric mean computation. *IMA Journal of Numerical Analysis*, 38(1):495–517, 2017.

[15] Chang Liu, Jun Zhu, and Yang Song. Stochastic gradient geodesic mcmc methods. In *Advances in Neural Information Processing Systems*, pages 3009–3017, 2016.

[16] Yao Lu, Kaizhu Huang, and Cheng-Lin Liu. A fast projected fixed-point algorithm for large graph matching. *Pattern Recognition*, 60:971–982, 2016.

[17] E. Maset, F. Arrigoni, and A. Fusiello. Practical and efficient multi-view matching. In *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[18] Chunhong Qi, Kyle A Gallivan, and P-A Absil. Riemannian bfgs algorithm with applications. In *Recent advances in optimization and its applications in engineering*, pages 183–192. Springer, 2010.

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[20] Wolfgang Ring and Benedikt Wirth. Optimization methods on riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, 2012.

[21] Qianqian Wang, Xiaowei Zhou, and Kostas Daniilidis. Multi-image semantic matching by mining consistent features. In *CVPR*, 2018.

[22] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[23] Philip Wolfe. Convergence conditions for ascent methods. ii: Some corrections. *SIAM review*, 13(2):185–188, 1971.

[24] Tatiana Xifara, Chris Sherlock, Samuel Livingstone, Simon Byrne, and Mark Girolami. Langevin diffusions and the Metropolis-adjusted Langevin algorithm. *Statistics & Probability Letters*, 91:14–19, 2014.