Noname manuscript No.
(will be inserted by the editor)

Ground Metric Learning on Graphs

Matthieu Heitz · Nicolas Bonneel · David Coeurjolly · Marco Cuturi · Gabriel Peyré

the date of receipt and acceptance should be inserted later

Abstract Optimal transport (OT) distances between probability distributions are parameterized by the ground metric they use between observations. Their relevance for real-life applications strongly hinges on whether that ground metric parameter is suitably chosen. The challenge of selecting it adaptively and algorithmically from prior knowledge, the so-called ground metric learning (GML) problem, has therefore appeared in various settings. In this paper, we consider the GML problem when the learned metric is constrained to be a *geodesic distance* on a *graph* that supports the measures of interest. This imposes a rich structure for candidate metrics, but also enables far more efficient learning procedures when compared to a direct optimization over the space of all metric matrices. We use this setting to tackle an inverse problem stemming from the observation of a density evolving with time; we seek a graph ground metric such that the OT interpolation between the starting and ending densities that result from that ground metric agrees with the observed evolution. This OT dynamic framework is relevant to model natural phenomena exhibiting displacements of mass, such as the evolution of the color palette induced by the modification of lighting and materials.

Matthieu Heitz
Université de Lyon, CNRS/LIRIS, France
E-mail: matthieu.heitz@univ-lyon1.fr

Nicolas Bonneel
Université de Lyon, CNRS/LIRIS, France

David Coeurjolly
Université de Lyon, CNRS/LIRIS, France

Marco Cuturi
Google Research, Brain team, France

Gabriel Peyré
CNRS and ENS, PSL University, France

Keywords Optimal transport · Metric learning · Displacement interpolation

1 Introduction

Optimal transport (OT) is a powerful tool to compare probability measures supported on geometric domains (such as Euclidean spaces, surfaces or graphs). The value provided by OT lies in its ability to leverage prior knowledge on the proximity of two isolated observations to quantify the discrepancy between two probability distributions of such observations. This prior knowledge is usually encoded as a “ground metric” [39], which defines the cost of moving mass between points.

The Wasserstein distance between histograms, densities or point clouds, all seen here as particular instances of probability measures, is defined as the smallest cost required to transport one measure to another. Because this distance is geodesic when the ground metric is geodesic, OT can also be used to compute interpolations between two probability measures, namely a path in the probability simplex that connects these two measures as end-points. This interpolation is usually referred to as a *displacement* interpolation [33], describing a series of intermediate measures during the transport process.

When two discrete probability distributions are supported on a Euclidean space, and the ground metric is itself the Euclidean distance (the most widely used setting in applications), theory tells us that the displacement interpolation between these two measures only involves particles moving along straight lines, from one point in the starting measure to another in the end measure. Imagine that, on the contrary, we observe a time series of measures in which mass displacements

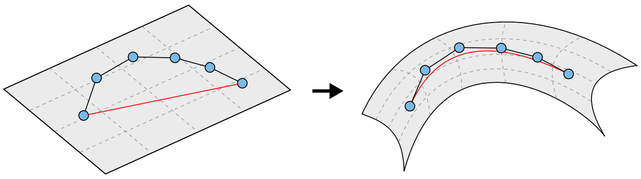

Fig. 1: Left: before metric learning, the sequence of observed histograms (blue points) lies in the Wasserstein space of probability distributions with a Euclidean ground metric. The observed sequence does not match the Wasserstein geodesic (red line) between the first and last element. Right: after modifying the ground metric iteratively, the Wasserstein space is now deformed in such a way that the geodesic between the first and last element in this new geodesic space (red curve) is as close as possible to the sequence.

do not seem to match that hypothesis. In that case the ground metric inducing such mass displacements must be of a different nature. We cast in that case the following inverse problem: under which ground metric could this observed mass displacement be considered optimal? The goal of our approach here is precisely to answer that question. We give an illustrative example in Figure 1, where we show that we search for a ground metric that deforms the space such that the sequence of mass displacements that is observed is close to a Wasserstein geodesic with that ground metric.

The main choice in our approach relies on looking at (anisotropic) diffusion-based geodesic distances [55] as the space of candidate ground metrics. We then minimize the reconstruction error between measures that are observed at intermediary time stamps and interpolated histograms with that ground metric. The problem we tackle is challenging in terms of time and memory complexity, due to repeated calls to solve Wasserstein barycenter problems with a non-Euclidean metric. We address these issues using a sparse resolution of an anisotropic diffusion equation, yielding a tractable algorithm. The optimization is performed using a quasi-Newton solver and automatic differentiation to compute the Jacobians of Wasserstein barycenters, here computed with entropic regularization and through a direct differentiation of Sinkhorn iterations [8, 23]. Because an automatic differentation of this entire pipeline would suffer from a prohibitive memory footprint, we also propose closed-form gradient formulas for the diffusion process. We validate our algorithm on synthetic datasets, and on the learning of color variations in image sequences. Finally, a Python implementation of our method as well as some datasets used in this paper are available online [1].

Contributions

- We introduce a new framework to learn the ground metric of optimal transport, where it is restricted to be a geodesic distance on a graph. The metric is parameterized as weights on the graph's edges, and geodesic distances are computed through anisotropic diffusion.
- We estimate this metric by fitting measures that are intermediate snapshots of a dynamical evolution of mass as Wasserstein barycenters.
- We provide a tractable algorithm based on the sparse discretization of the diffusion equation and efficient automatic differentiation.

2 Related Works

2.1 Computational Optimal transport

Solving OT problems has remained intractable for many years because doing so relies on solving a bipartite minimum-cost flow problem, with a number of variables that is quadratic with regard to the histograms' size. Fortunately, in the past decade, methods to approximate OT distances using various types of regularizations have become widespread. Cuturi [16] introduced an entropic regularization of the problem, which allows the efficient approximation of the Wasserstein distance, using an iterative scaling method called the Sinkhorn algorithm. This algorithm is very simple as it only performs point-wise operations on vectors, and matrix-vector multiplications that involve a kernel, defined as the exponential of minus the ground metric, inversely scaled by the regularization strength. Cuturi and Doucet [18] then extended this method to compute Wasserstein barycenters, a concept which was previously introduced in [1]. Benamou et al. [6] later linked this iterative scheme to Bregman projections, and showed that it can be adapted to solve various OT related problems such as partial, multi-marginal, or capacity-constrained OT. This regularization allows the computation of OT for large problems, such as those arising in machine learning [14, 22, 23] or computer graphics [8, 43, 45].

Recently, Altschuler et al. [2] introduced a method to accelerate the Sinkhorn algorithm via low-rank (Nyström) approximations of the kernel [2]. Simultaneously, there have been considerable efforts to study

[1] https://github.com/matthieuheitz/2020-JMIV-ground-metric-learning-graphs

the convergence and approximation properties of the Sinkhorn algorithm [3] and its variants [21].

Other families of numerical methods are based on variational or PDE formulations of the problem [35, 4], or semi-discrete formulations [30]. We refer to [38] and [41] for extensive surveys on computational optimal transport.

The entropic regularization scheme has helped to tackle inverse problems that involve OT, since it converts the original Wasserstein distance into a fast, smooth, differentiable, and more robust loss. Although differentiating the Wasserstein loss has been extensively covered, differentiating quantities that build upon it, such as smooth Wasserstein barycenters, is less common. A few examples are Wasserstein barycentric coordinates [8], Wasserstein dictionary learning on images [43] and graphs [44], and model ensembling [19].

2.2 Metric Learning

In machine learning, metric learning is the task of inferring a metric on a domain using side information, such as examplar points that should be close or far away from each other. The assumption behind such methods is that metrics are chosen within parameterized families, and tailored for a task and data at hand, rather than selected among a few handpicked candidates. Metric learning algorithms are supervised, often learning from similarity and dissimilarity constraints between pairs of samples (x_i should be close to x_j), or triplets (x_i is closer to x_j than to x_k). Metric learning has applications in different tasks, such as classification, image retrieval, or clustering. For instance, for classification purposes, the learned metric brings closer samples of the same class and drives away samples of different classes [53].

Metric learning methods are either linear or non-linear, depending on the formulation of the metric with respect to its inputs. We will briefly recall various metric learning approaches, but refer the reader to existing surveys [29, 5]. A widely-used linear metric function is the squared Mahalanobis distance, which is employed in the popular Large Margin Nearest Neighbors algorithm (LMNN) [51] along with a k-NN approach. Other linear methods [11, 37] choose not to satisfy all distance axioms (unlike the Mahalanobis distance) for more flexibility and because they are not essential to agree with human perception of similarities [5]. Non-linear methods include the prior embedding of the data (kernel trick) before performing a linear method [47, 50], or other non-linear metric functions [13, 28]. Facing problems where the data samples are histograms, researchers have developed metric learning methods based on distances that are better suited for histograms such as χ^2 [28, 56] or the Wasserstein distance, which we describe in more detail.

2.3 Ground Metric Learning

The Wasserstein distance relies heavily—one could almost say exclusively—on the ground metric to define a geometry on probability distributions. Setting that parameter is therefore crucial, and being able to rely on an adaptive, data-based procedure to select it is attractive from an applied perspective. The ground metric learning (GML) problem, following the terminology set forth by Cuturi and Avis [17], considers the generic case in which a ground cost that is a true metric (definite, symmetric and satisfying triangle inequalities) is learned using supervised information from a set of histograms. This method requires projecting matrices onto the cone of metric matrices, which is known to require a cubic effort in the size of these matrices [9]. Wang and Guibas [49] follow GML's approach but drop the requirement that the learned cost must be a metric. Zen et al. [57] use GML to enhance previous results on Non-negative Matrix Decomposition with a Wasserstein loss (EMD-NMF) [40], by alternatively learning the matrix decomposition and the ground metric. Learning a metric from the observation of a matching is a well-studied problem. Dupuy et al. [20] learn a similarity matrix from the observation of a fixed transport plan, and use this to propose factors explaining weddings across groups in populations. Stuart and Wolfram [46] infer graph-based cost functions similar to ours, but learn from noisy observations of transport plans, in a Bayesian framework. Finally, Li et al. [31] extend the method of Dupuy et al. [20] for noisy and incomplete matchings, by relaxing the marginal constraint using the regularized Wasserstein distance itself (instead of the KL divergence commonly used in unbalanced optimal transport [12]). Huang et al. [27] consider a non-discrete GML problem that involves point-clouds, and propose to learn a Mahalanobis metric between word embeddings that agree with labels between texts, seen here as bags of words. Both of these approaches use the entropic regularization of Wasserstein distances (see next section). More recently, Xu et al. [54] combined several previous ideas to create a new metric learning algorithm. It is a regularized Wasserstein distance-flavored LMNN scheme, with a Mahalanobis distance as ground metric, parameterized by multiple local metrics [52] and a global one.

Similarly to the above works, our method aims to learn the ground metric of OT distances, but differs in the formulation of the ground metric. We search for metrics that are geodesic distances on graphs, via a diffusion equation. Our method also differs in the data we learn from: the observations that are fed to our algorithm are snapshots of a mass movement, and not pair or triplet constraints. We use displacement interpolations to reconstruct that movement, hence our objective function contains multiple inverse problems involving OT distances. This contrasts with simpler formulations where the objective function or the constraints are weighted sums of OT distances [17, 49, 54]. Furthermore, the interest in these previous works was generally to perform supervised classification, which is something we do not aim to do in this paper. Nevertheless, our learning algorithm is supervised, since we provide the exact timestamps of each sample in the sequence.

Our method is similar to the work of Zen et al. [57], in the sense that we both aim to reconstruct the input data given a model. However they only use OT distances as loss functions to compare inputs with linear reconstructions, while we use OT distances to synthesize the reconstructions themselves. The difference between our method and those of Dupuy et al. [20], Stuart and Wolfram [46], Li et al. [31] lies in the available observations: they learn from a fixed matching (transport plan) whereas we learn from a sequence of mass displacements from which we infer both the metric and an optimal transport plan. Our method thus does not require identifying information on traveling masses. The fact that this identification is not required ranks among the most important and beneficial contributions of the OT geometry to data sciences, notably biology [42].

Our approach to metric learning corresponds to setting up an optimization problem over the space of geodesic distances on graphs, which is closely related to the continuous problem of optimizing Riemannian metrics. Optimizing metrics from functionals involving geodesic distances has been considered in [7]. This has recently been improved in [34] using automatic differentiation to compute the gradient of the functional involved, which is also the approach we take in our work. This type of metric optimization problem has also been studied within the OT framework (see for instance [10]), but these works are only concerned with convex problems (typically maximization of geodesic or OT distances), while our metric learning problem is highly non-convex.

3 Context

Optimal transport defines geometric distances between probability distributions. In this paper, we consider discrete measures on graphs. These measures are sums of weighted Dirac distributions supported on the graph's vertices: $\mu = \sum_{i=1}^{N} u_i \delta_{x_i}$, with the weight vector $\mathbf{u} = (u_i)$ in the probability simplex $\Sigma_N \overset{\text{def.}}{=} \left\{\mathbf{u} \in \mathbb{R}_+^N \,\middle|\, \sum_{i=1}^N \mathbf{u}_i = 1\right\}$, and x_i the position of vertex i in an abstract space. In the following, we will refer to the weight vector of these measures as "histograms".

A transport plan between two histograms $\mathbf{a}, \mathbf{b} \in \Sigma_N$ is a matrix $\mathbf{P} \in \mathbb{R}_+^{N \times N}$, where $\mathbf{P}_{i,j}$ gives the amount of mass to be transported from vertex i of $\mathbf{a}$ to vertex j of $\mathbf{b}$. We define the transport polytope of $\mathbf{a}$ and $\mathbf{b}$ as

$$U(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \left\{\mathbf{P} \in \mathbb{R}_+^{N\times N} \;\middle|\; \mathbf{P}\mathbb{1}_N = \mathbf{a} \;\text{ and }\; \mathbf{P}^T\mathbb{1}_N = \mathbf{b}\right\}.$$

The Kantorovich problem. Optimal transport aims to find the transport plan $\mathbf{P}$ that minimizes a total cost, which is the mass transported multiplied by its cost of transportation. This is called the Kantorovich problem and it is written

$$W_{\mathbf{C}}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \min_{\mathbf{P}\in U(\mathbf{a},\mathbf{b})} \sum_{i,j} \mathbf{C}_{i,j}\mathbf{P}_{i,j}. \tag{1}$$

The cost matrix $\mathbf{C} \in \mathbb{R}_+^{N\times N}$ defines the cost $\mathbf{C}_{i,j}$ of transporting one unit of mass from vertex i to j. If the cost matrix is $\mathbf{C}_{i,j} = d(x_i, x_j)^p$, with d a distance on the domain, then $W_{\mathbf{C}}^{1/p}$ is a distance between probability distributions, called the p-Wasserstein distance [38, Proposition 2.2].

Entropy regularization. This optimization problem can be regularized, and a computationally efficient way to do so is to balance the transportation cost with the entropy H of the transport plan [16]. The resulting entropy-regularized problem is written as

$$W_{\mathbf{C}}^{\varepsilon}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \min_{\mathbf{P}\in U(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P}\rangle - \varepsilon H(\mathbf{P}), \tag{2}$$

where $H(\mathbf{P}) \overset{\text{def.}}{=} -\sum_{i,j} \mathbf{P}_{i,j}(\log(\mathbf{P}_{i,j}) - 1)$ and $\varepsilon > 0$. The addition of this regularization term modifies how the Kantorovich problem can be addressed. Without regularization, the problem must be solved with network flow solvers, whereas with regularization it can be conveniently solved using the Sinkhorn algorithm, which is less costly than minimum cost network flow algorithms for sufficiently large values of ε (the smaller the regularization, the slower the convergence). The obtained value $W_{\mathbf{C}}^{\varepsilon}$ is an approximation of the exact Wasserstein distance, and the approximation error can be controlled with ε. Another chief advantage of this regularization is that $W_{\mathbf{C}}^{\varepsilon}(\mathbf{a},\mathbf{b})$ defines a smooth function of both its inputs $(\mathbf{a},\mathbf{b})$ and the metric C. This property is important to be able to derive efficient and stable metric learning schemes, as we seek in this paper.

Displacement interpolation. Given two histograms $\mathbf{r}_0$ and $\mathbf{r}_1$, their barycentric interpolation is defined as the curve parameterized for $t \in [0,1]$ as

$$\gamma_{\mathbf{C}}(\mathbf{r}_0,\mathbf{r}_1,t) \overset{\text{def.}}{=} \underset{\mathbf{r}\in\Sigma_N}{\operatorname{argmin}} \ (1-t)W_{\mathbf{C}}(\mathbf{r}_0,\mathbf{r}) + tW_{\mathbf{C}}(\mathbf{r},\mathbf{r}_1). \tag{3}$$

This class of problem was introduced and studied by [1]. In the case where d is a geodesic distance and $p = 2$, then $W_{\mathbf{C}}(\mathbf{r}_0,\mathbf{r}_1)$ not only represents the total cost of transporting $\mathbf{r}_0$ to $\mathbf{r}_1$, but also the square of the length of the shortest path (a geodesic) between them in the Wasserstein space. In this case, (3) defines the so-called *displacement interpolation* [33], which is also the geodesic from $\mathbf{r}_0$ to $\mathbf{r}_1$.

With a slight abuse of notation, in the following we call $\gamma_{\mathbf{C}}$ the displacement interpolation path, for any generic cost C. In practice, we approximate this interpolation using the regularized Wasserstein distance $W^{\varepsilon}_{\mathbf{C}}$, which means we can compute it as a special case of regularized Wasserstein barycenter [6] between two histograms. We denote $\gamma^{\varepsilon}_{\mathbf{C}}$ the resulting smoothed approximation.

4 Method

4.1 Metric parametrization

Since histograms are supported on a graph, we parameterize the ground metric by a positive weight $w_{i,j}$ associated to each edge connecting vertices i and j. This should be understood as being inversely proportional to the length of the edge, and conveys how easily mass can travel through it. Additionally, we set $w_{i,j} = 0$ when vertices i and j are not connected.

We aim to carry out metric learning using OT where the ground cost is the square of the geodesic distance associated to the weighted graph. Instead of optimizing a full adjacency matrix $\mathbf{W} = (w_{i,j})_{i,j}$, which has many zero entries that we do not wish to optimize, we define the vector $\mathbf{w} \in \mathbb{R}^K$ as the concatenation of all metric parameters $w_{i,j} > 0$, that is, those for which vertices i and j are connected. This imposes a fixed connectivity on the graph.

4.2 Problem statement

Let $(\mathbf{h}_i)_{i=1}^P \in \Sigma_N$ be observations at P consecutive time steps of a movement of mass. We aim to retrieve the metric weights $\mathbf{w}$ for which an OT displacement interpolation approximates best this mass evolution. This corresponds to an OT regression scheme parameterized by the metric, and leads to the following optimization problem

$$\min_{\mathbf{w}} \ \sum_{i=1}^{P} \mathcal{L}\left(\gamma^{\varepsilon}_{\mathbf{C}_\mathbf{w}}\left(\mathbf{h}_1,\mathbf{h}_P,t_i\right),\mathbf{h}_i\right) + f(\mathbf{w}), \tag{4}$$

where:

- t_i are the timestamps: $t_i = (i-1)/(P-1)$,
- $\mathcal{L}$ is a loss function between histograms,
- $\mathbf{C}_\mathbf{w}$ is the ground metric matrix associated to the graph weights $\mathbf{w}$, detailed in section 4.4,
- and $f(\mathbf{w})$ is a regularization term detailed in section 4.5.

Note that we chose equally spaced timestamps for the sake of simplicity, but the method is applicable to any sequence of timestamps.

In the following sections, we detail the different components of our algorithm. Our objective function (4) is non-convex, and we minimize it with an L-BFGS quasi-Newton algorithm, to compute a local minimum of the non-convex energy. The L-BFGS algorithm requires the evaluation of the energy function, as well as its gradient with respect to the inputs. In our case, evaluating the energy function (4) requires reconstructing the sequence of input histograms using a displacement interpolation between the first and last histogram, and assessing the quality of the reconstructions. The gradient is calculated through automatic differentiation, which provides high flexibility when adjusting the framework.

In our numerical examples, we consider 2-D and 3-D datasets discretized on uniform square grids, so that the graph is simply the graph of 4 or 6 nearest neighbors on this grid.

4.3 Kernel application

As mentioned previously, we use entropy-regularized OT to compute displacement interpolations (3). These interpolations can be computed via the Sinkhorn barycenter algorithm 1, and for which the main computational burden is to apply the kernel matrix $\mathbf{K}$ on $\mathbb{R}^N$ vectors. When the domain is a grid and the metric is Euclidean, applying that kernel boils down to a simple convolution with a Gaussian kernel. For an arbitrary metric as in our case, computing the kernel $\mathbf{K}$ requires all-pairs geodesic distances on the graph. This can be achieved using e.g. Dijkstra's algorithm or the Floyd-Warshall algorithm. However, this kernel is a non-smooth operator, which is quite difficult to differentiate with respect to the metric weights (see for instance [7, 34] for works in this direction). In sharp contrast, our approach leverages Varadhan's formula [48]:

we approximate the geodesic kernel $\mathbf{K}$ with the heat kernel. This kernel is itself approximated by solving the diffusion equation using S sub-steps of an implicit Euler scheme. This has the consequence of 1) having faster evaluations of the kernel and 2) smoothing the dependency between the distance kernel and the metric. This last property is necessary to define a differentiable functional, which is key for an efficient solver. This has been applied to OT computation by Solomon et al. [45], inspired from the work of Crane et al. [15].

Algorithm 1 Sinkhorn barycenter [6]

```
Input: histograms $(\mathbf{a}_r) \in (\Sigma_N)^R$, weights $\lambda \in \Sigma_R$,
kernel $\mathbf{K}$, number of iterations $L$
Ouput: barycenter $\mathbf{b}$

  $\forall r, \mathbf{v}_r = \mathbb{1}_N$
  for $l = 1$ to $L$ do
      $\forall r, \mathbf{u}_r = \frac{\mathbf{a}_r}{\mathbf{K}\mathbf{v}_r}$
      $\mathbf{b} = \prod_r \left(\mathbf{K}^T \mathbf{u}_r\right)^{\lambda_r}$
      $\forall r, \mathbf{v}_r = \frac{\mathbf{b}}{\mathbf{K}^T \mathbf{u}_r}$
  end for
```

4.4 Computing geodesic distances

While Solomon et al. [45] discretize the diffusion equation using a cotangent Laplacian because they deal with triangular meshes, we prefer a weighted graph Laplacian parameterized by the metric weights $\mathbf{w}$, which we detail hereafter.

As mentioned in section 4.1, the weighted adjacency matrix $\mathbf{W}$ is defined as $\mathbf{W}_{i,j} = \mathbf{W}_{j,i} = w_{i,j}$ where $w_{i,j}$ are the (undirected) edge weights parameterizing the metric. It is symmetric and usually sparse, since $w_{i,j}$ is non-zero only for vertices that are connected, and 0 otherwise. The diagonal weighted degree matrix sums the weights of each row on the diagonal: $\mathbf{\Lambda} \overset{\text{def.}}{=} \operatorname{diag}(\mathbf{d})$, with $\mathbf{d}_i \overset{\text{def.}}{=} \sum_{j=1}^{N} w_{i,j}$. The negative semi-definite weighted graph Laplacian matrix is then defined as $\mathbf{L}_\mathbf{w} = \mathbf{W} - \mathbf{\Lambda}$.

We discretize the heat equation in time using an implicit Euler scheme and perform S sub-steps. It is crucial to rely on an implicit stepping scheme to obtain approximated kernels supported on the full domain, in order for Sinkhorn iterations to be well conditioned (as opposed to using an explicit Euler scheme, which would break Sinkhorn's convergence). Denoting $\mathbf{v}$ the initial condition of the heat diffusion, $\mathbf{u}$ the final solution after a time $\varepsilon/4$, and $\mathbf{L}_\mathbf{w}$ our discrete Laplacian operator, we solve

$$\left(\mathbf{Id} - \frac{\varepsilon}{4S}\mathbf{L}_\mathbf{w}\right)^S \mathbf{u} = \mathbf{v}. \tag{5}$$

We denote by $\mathbf{M}$ the symmetric matrix $\mathbf{Id} - \frac{\varepsilon}{4S}\mathbf{L}_\mathbf{w}$. Applying the kernel $\mathbf{K} \overset{\text{def.}}{=} \mathbf{M}^{-S}$ to a vector $\mathbf{v}$ is then simply achieved by solving S linear systems: $\mathbf{u} = \mathbf{K}\mathbf{v} = \mathbf{M}^{-S}\mathbf{v}$. We never compute the full kernel matrix $\mathbf{K}$ because it is of size N^2, which quickly becomes prohibitive in time and memory as histograms grow ($\approx$ 12GB for histograms of size $N = 200^2$, and $\approx$ 30GB for histograms of size $N = 40^3$).

The intuition behind this scheme is that, $(\mathbf{Id} - \frac{\varepsilon}{4S}\mathbf{L}_\mathbf{w})^{-S}$ approximates the heat kernel for large S, which itself for small ε approximates the geodesic exponential kernel, which is of the form $\exp(-d^2(x,y)/\varepsilon)$ for a small ε with d the geodesic distance on a manifold. Note however that this link is not valid on graphs or triangulations, although it has been reported to be very effective when choosing ε in proportion to the discretization grid size (see [45]). Our method can thus be seen as choosing a cost of the form

$$\mathbf{C}_\mathbf{w} \overset{\text{def.}}{=} -\varepsilon \log\left((\mathbf{Id} - \frac{\varepsilon}{4S}\mathbf{L}_\mathbf{w})^{-S}\right), \tag{6}$$

even though we never compute this cost matrix explicitly.

The chief advantage of the formula (5) to approximate a kernel evaluation is that the same matrix is repeatedly used S times, which is itself repeated at each iteration of Sinkhorn's algorithm 1 to evaluate barycenters. Following Solomon et al. [45], a dramatic speed-up is thus obtained by pre-computing a sparse Cholesky decomposition of $\mathbf{M}$. For instance, on a 2-D domain, the number of non-zero elements of such a factorization is of the order of N, so that each linear system resolution has linear complexity.

4.5 Inverse problem regularization

The metric learning problem is severely ill-posed and this difficulty is further increased by the fact that the corresponding optimization problem (4) is non-convex. These issues can be mitigated by introducing a regularization term $f(\mathbf{w}) \overset{\text{def.}}{=} \lambda_c f_c(\mathbf{w}) + \lambda_s f_s(\mathbf{w})$. Note also that since a global variation of scale in the metric does not affect the solution of optimal transport, the problem needs to be constrained, otherwise metric weights tend to infinity when they are optimized.

We introduce two different regularizations: f_c forces the weights to be close to 1 (this controls how much the space becomes inhomogeneous and anisotropic), and f_s

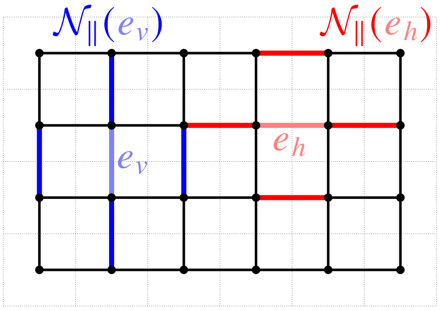

Fig. 2: Smooth prior: for each vertical (e_v) and horizontal (e_h) edge, we minimize the squared sum of weight differences with its respective neighbors of the same orientation.

constrains the weights to be spatially smooth. Since we carry out the numerical examples on graphs that are 2-D and 3-D grids, we use a smoothing regularization f_s that is specific to that case. This term must be adapted when dealing with general graphs.

The first regularization is imposed by adding the following term to our energy functional, multiplied by a control coefficient λ_c:

$$f_c(\mathbf{w}) \overset{\text{def.}}{=} ||\mathbf{w} - \mathbb{1}||_2^2 . \tag{7}$$

To enforce the second prior, we add the following term to our functional, multiplied by a control coefficient λ_s:

$$f_s(\mathbf{w}) \overset{\text{def.}}{=} \sum_{e \in E} \left(\sum_{e' \in \mathcal{N}_{\|}(e)} (w_e - w_{e'}) \right)^2 , \tag{8}$$

with E the set of undirected edges, and $\mathcal{N}_{\|}$ the set of neighbor edges of the same orientation, as illustrated in Figure 2 for the 2-D case.

We regularize separately horizontal and vertical edges to ensure that we recover an anisotropic metric. This is important for various applications, for example when dealing with color histograms, as MacAdam's ellipses reveal [32].

The selection of the regularization parameters (λ_c, λ_s) and their impact on the recovered metric is discussed in section 6.

4.6 Implementation

In order to ensure positivity of the metric weights, problem (4) is solved after a log-domain change of variable $\mathbf{w} = e^{\mathbf{w}'}$ and the optimization on $\mathbf{w}'$ is achieved using the L-BFGS algorithm.

Our method is implemented in Python with the Pytorch framework, which supports automatic differentiation (AD) [36]. The gradient is evaluated using reverse mode automatic differentiation [25, 26], which has a numerical complexity of the same order as that of evaluating the minimized functional (which corresponds to the evaluation of P barycenters). Reverse mode automatic differentiation computes the gradient of the energy (4) by back-propagating the gradient of the loss function $\mathcal{L}$ through the computational graph defined by the Sinkhorn barycenter algorithm. This back-propagation operates by applying the adjoint Jacobian of each operation involved in the algorithm. These adjoints Jacobians are applied in a backward pass, in an ordering that is the reverse of the one used to perform the forward pass of the algorithm. Most of these operations are elementary functions, such as matrix product and pointwise operations (e.g. division or exponentiation). These functions are built-in for most standard automatic differentiation libraries, so their adjoint Jacobians are already implemented. The only non-trivial case, which is usually not built-in, and can also be optimized for memory usage, is the application of the heat kernel $\mathbf{M}^{-S}$ to some vector $\mathbf{v}$.

The following proposition gives the expression of the adjoint Jacobian $[\partial \Phi(\mathbf{w})]^T : \mathbb{R}^N \to \mathbb{R}^K$ of the map $\mathbf{\Phi} : \mathbb{R}^K \to \mathbb{R}^N$

$$\mathbf{\Phi} : \mathbf{w} \in \mathbb{R}^K \mapsto (\mathbf{M})^{-S}\mathbf{v} \in \mathbb{R}^N \text{ where } \mathbf{M} \overset{\text{def.}}{=} \mathbf{Id} - \frac{\varepsilon}{4S}\mathbf{L}_{\mathbf{w}}, \tag{9}$$

from the graph weights $\mathbf{w}$ to the approximate solution of the heat diffusion, obtained by applying S implicit Euler steps starting from $\mathbf{v} \in \mathbb{R}^N$.

Proposition 1 *For $\mathbf{w} \in \mathbb{R}^K$ and $\mathbf{g} \in \mathbb{R}^N$, one has for each of the K edge indices (i, j)*

$$[\partial \Phi(\mathbf{w})]^T(\mathbf{g})_{i,j} = \frac{\varepsilon}{4S} \sum_{\ell=0}^{S-1} \left(\mathbf{g}_i^\ell - \mathbf{g}_j^\ell\right)\left(\mathbf{v}_i^\ell - \mathbf{v}_j^\ell\right) \tag{10}$$

$$\textit{where} \quad \begin{cases} \mathbf{g}^\ell \overset{\text{def.}}{=} \mathbf{M}^{\ell - S}\mathbf{g} \\ \mathbf{v}^\ell \overset{\text{def.}}{=} \mathbf{M}^{-\ell-1}\mathbf{v} \end{cases} \tag{11}$$

Proof The mapping $\mathbf{\Phi}$ is the composition of $\phi_1 : \mathbf{w} \mapsto \mathbf{M}$, $\phi_2 : \mathbf{M} \mapsto \mathbf{U} = \mathbf{M}^{-1}$, $\phi_3 : \mathbf{U} \mapsto \mathbf{V} = \mathbf{U}^S$ and $\phi_4 : \mathbf{V} \mapsto \mathbf{y} = \mathbf{V}\mathbf{v}$. The formula follows by composing the adjoint Jacobians of each of these operations, which are detailed in Appendix A. □

Note that the vectors $\mathbf{v}_\ell$ are already computed during the forward pass of the algorithm, so they need to be stored. Only the vectors $\mathbf{g}_\ell$ need to be computed by iteratively solving linear systems associated to $\mathbf{M}$.

It is important to set a fixed number of iterations L for the Sinkhorn barycenter algorithm when using automatic differentiation, because a stopping criteria based on convergence might be problematic to differentiate (it is a discontinuous function). Moreover, it would render memory consumption (and speed) unpredictable, which may be problematic due to the limited amount of memory available.

5 Experiments

We first show a few synthetic examples, in which the input sequence of measures has been generated as a Wasserstein geodesic using a ground metric known beforehand. This ground truth metric is compared with the output of our algorithm. We then present an application to a task of learning color variations in image sequences.

In the following, an "interpolation" refers to a displacement interpolation, unless stated otherwise.

5.1 Synthetic experiments

As mentioned in 4.2, our algorithm solves an inverse problem: given a sequence of histograms representing a movement of mass, we aim at fitting a metric for which that sequence can be sufficiently well approached by a displacement interpolation between the first and last frame.

5.1.1 Retrieving a ground truth metric

In Figure 3, Figure 4 and Figure 5, we test our algorithm by applying it on different sequences of measures that are themselves geodesics generated using handcrafted metrics, and verify that the learned metric is close to the original one, which constitutes a ground truth. In general, it is impossible to recover with high precision the exact same metric, because such an inverse problem is too ill-posed (many different metrics can generate the same interpolation sequence) and the energy is non-convex. Moreover, regularization introduces a bias while helping to fight against this non-convexity. Hence, we attempt to find a metric that shares the same large scale features as the original one.

We run three experiments with different handcrafted metrics, on 2-D histograms defined on an n by n Cartesian grid. The parameters for these experiments are: a grid of size $n = 50$, $L = 50$ Sinkhorn iterations, an entropic regularization factor ε=1.2e-2, $S = 100$ substeps for the diffusion equation, 1000 L-BFGS iterations, and the metric regularization factor $\lambda_c = 0$. The other regularization factor is $\lambda_s = 0.03$ for the first two experiments and $\lambda_s = 1.0$ for the third one. Finally, each of the three experiments is tested with three different loss functions, and we display the result that is closest to the ground truth. The different loss functions are the L^1 norm, the squared L^2 norm and the Kullback-Leibler divergence:

$$\mathcal{L}_1(\mathbf{p},\mathbf{q}) \overset{\text{def.}}{=} ||\mathbf{p}-\mathbf{q}||_1\,, \tag{12}$$

$$\mathcal{L}_2(\mathbf{p},\mathbf{q}) \overset{\text{def.}}{=} ||\mathbf{p}-\mathbf{q}||_2^2\,, \tag{13}$$

$$\mathcal{L}_{KL}(\mathbf{p},\mathbf{q}) \overset{\text{def.}}{=} \mathbb{1}^T\left(\mathbf{p}\odot\log(\mathbf{p}\oslash\mathbf{q})-\mathbf{p}+\mathbf{q}\right), \tag{14}$$

with $\odot$ and $\oslash$ being respectively the element-wise multiplication and division. We will see that the best loss function varies depending on the data.

The metric $\mathbf{w}$ is located along either vertical or horizontal edges. We thus display two images each time: one for the horizontal and one for the vertical edges.

In Figure 3, we are able to reconstruct the input sequence, and retrieve the main zones of low diffusion (in blue), that deviate the mass from a straight trajectory. The $\mathcal{L}_1$ loss gave the best result.

In Figure 4, the original horizontal and vertical metric weights are different and this experiment shows that we are able to recover the distinct features of each metric i.e. the dark blue and dark red areas. The $\mathcal{L}_{KL}$ loss gave the best result.

In Figure 5, the original metric is composed of two obstacles, but only one of them is in the mass' trajectory. We can observe that obstacles that are not approached by any mass are not recovered, which is expected, because the algorithm cannot find information in these areas. The $\mathcal{L}_2$ loss gave the best result.

5.1.2 Hand-crafted interpolations

We now test our algorithm on a dataset that has not been generated by the forward model, as in the previous paragraph. We reproduced in Figure 6 the trajectory of Figure 5 using a moving Gaussian distribution. By comparing them, we can see that reconstructions are close to the input, but not identical, since the metric induces an inhomogeneous diffusion, resulting in an apparent motion blur. The horizontal metric obtained on this dataset is similar to the one obtained in Figure 5, with a blue obstacle on the upper middle part that prevents mass from going straight. However, it differs in that the blue region extends towards the center of the support. This can be explained by the fact that the algorithm tries to fit the input densities, which are less diffuse horizontally. Therefore, it will decrease the metric so that diffusion is less strong in that zone. Parameters for this experiment were $n = 50$, $L = 50$, ε=1.2e-2, $S = 100$, $\lambda_c = 0$, $\lambda_s = 10$, and an $\mathcal{L}_2$ loss.

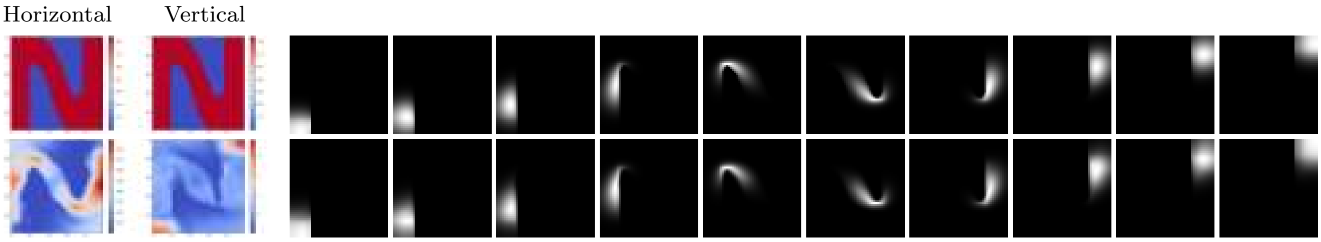

Fig. 3: First row: an initial metric (two leftmost images: weights on horizontal and vertical edges) is used to generate a histogram sequence. Second row: we apply our algorithm on that sequence, to recover the initial metric. The algorithm is able to recover the blue zones avoided by the mass, and red zones on the path it is taking.

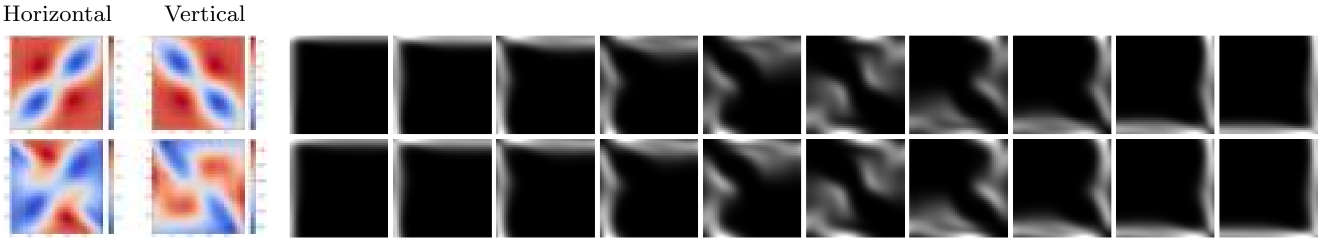

Fig. 4: First row: an initial metric (two leftmost images: weights on horizontal and vertical edges) is used to generate a histogram sequence. Second row: we apply our algorithm on that sequence, to recover the initial metric. The algorithm recovers the high (red) and low (blue) diffusion areas horizontally, as well as vertically.

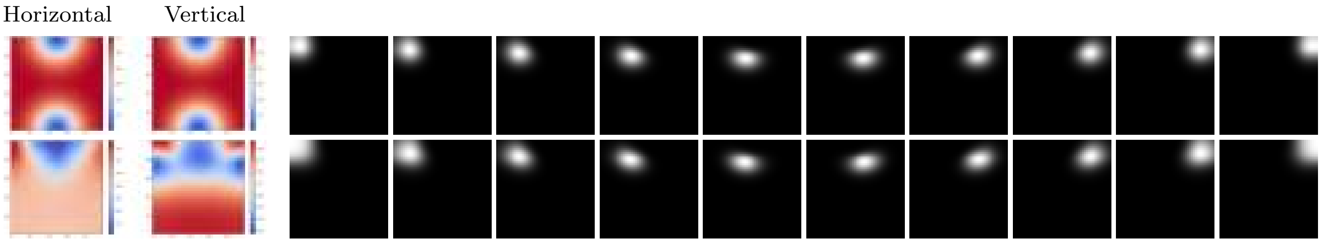

Fig. 5: First row: an initial metric (two leftmost images: weights on horizontal and vertical edges) is used to generate a histogram sequence. Second row: we apply our algorithm on that sequence, to recover the initial metric. This figure shows an example of a metric detail not being recovered because mass is not traveling in that region.

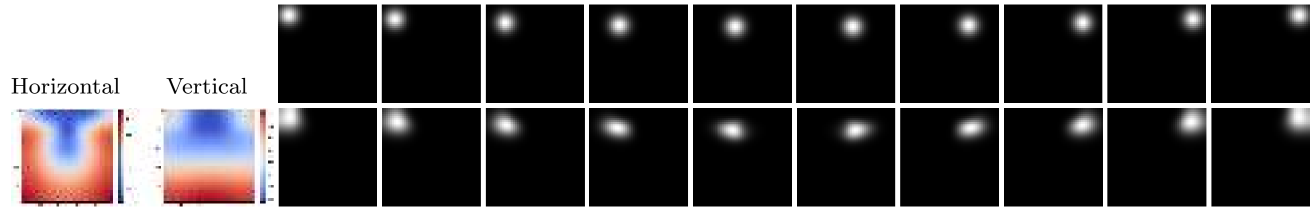

Fig. 6: Example of the resulting metric on a dataset that is not generated by the forward model. First row: synthetic dataset of a Gaussian distribution following the same trajectory as in Figure 5. Second row: we apply our algorithm to this dataset to recover a metric (two leftmost images), and the reconstructions. We recover a metric that shares the main features of the one in Figure 5.

5.1.3 Multiple sequences

We further attempted to recover metrics from multiple observed paths by summing our functional (4) over multiple input sequences. In Figure 7, we show a ground truth metric, and four sequences generated with that metric. They are all interpolations of two Gaussian distributions placed on either side of the support: horizontally, vertically, or diagonally. We then present in that figure three different metrics: the first is learned on the first sequence, the second is learned on the third sequence, and the last one is learned on all sequences. We can see that the metric learned on all sequences is closer to the ground truth. By providing more learning data, we reduce the space of solutions and prevent overfitting, effectively making the problem better posed. This comes at the cost of computational overhead, but that only increases linearly with the number of sequences. Parameters for these experiments were $n = 50$, $L = 50$, ε=1.2e-2, $S = 100$, $\lambda_c = 0$, $\lambda_s = 3$, and an $\mathcal{L}_2$ loss.

5.2 Evaluation

5.2.1 Diffusion equation

The parameters ε and S need to be carefully set for solving the diffusion equation. Indeed, depending on their value, the formula (5) yields a kernel that is a better or worse approximation of the heat kernel, which directly impacts the accuracy of the displacement interpolations computed with it. We demonstrate these effects in 2-D, by interpolating between two Dirac masses across a 50x50 image. We plot the middle slice of the 2-D image as a 1-D function, for easier visualization. In Figure 8, we plot 10 steps of an interpolation in each subplot, for different values of ε and S, with a Euclidean metric (all metric weights equal to 1).

We observe a trade-off between having sharp interpolations, and having evenly spaced interpolants, which means a constant-speed interpolation. It is important to note that memory footprint grows almost linearly with S (see next paragraph), since every intermediate vector in (5) is stored for the backward pass. In practice, we use either ε=4e-2 and $S = 20$, or ε=1.2e-3 and $S = 50$. With this level of smoothing, we set the number of Sinkhorn iterations to 50, which is generally enough for the Sinkhorn algorithm to converge.

5.2.2 Regularization

In order to evaluate the influence of the regularization, we compare the same experiment (the one conducted in Figure 4), with one of the two regularizers (f_c and f_s).

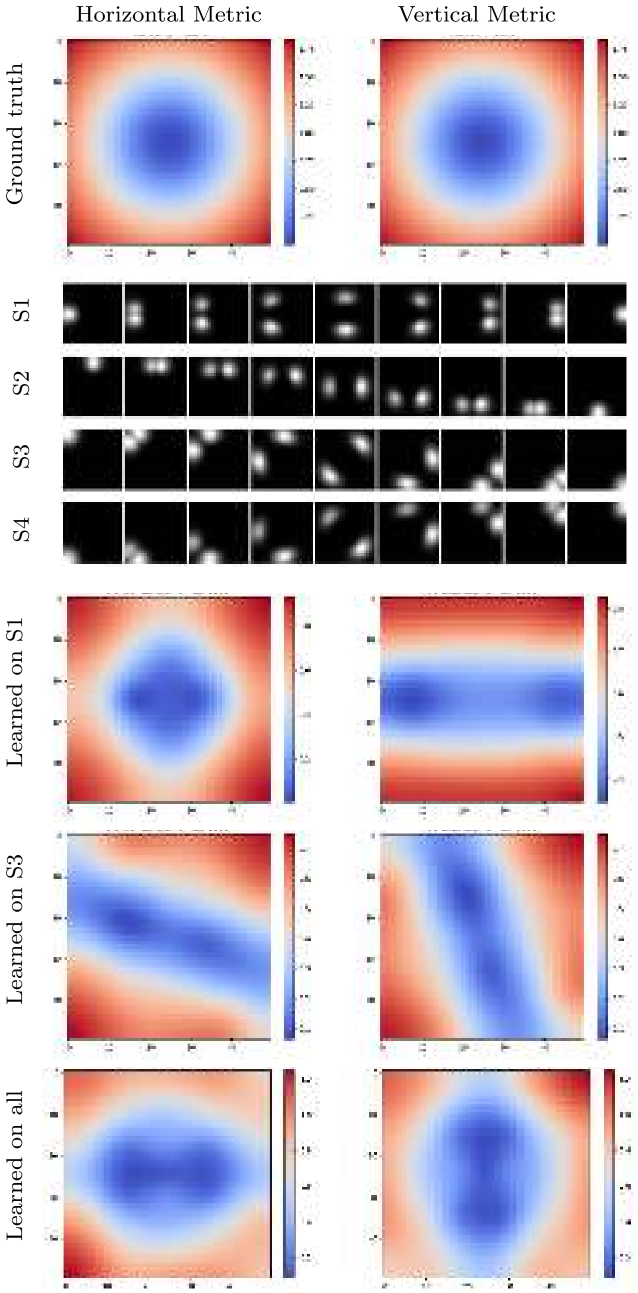

Fig. 7: Preliminary experiment of learning on multiple sequences. Using the ground truth metric in the first row, we generate 4 sequences of moving Gaussian distributions. In the last 3 rows, we compare the metric learned: on the first sequence, on the third sequence and on all sequences. Learning on multiple sequences makes the problem better posed, and we see that the recovered metric is closer to the ground truth than when learning with only one sequence.

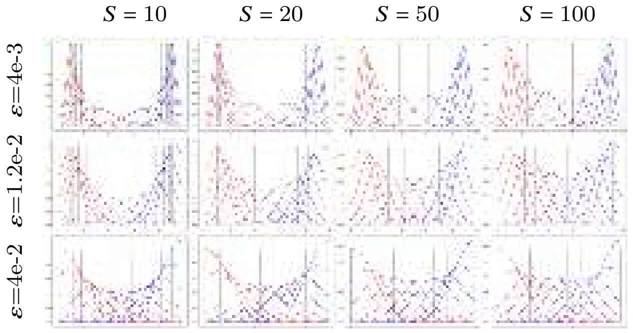

Fig. 8: Influence of parameters ε (diffusion time) and S (number of time discretization sub-steps) on displacement interpolation between 2 Dirac masses, computed with 50 Sinkhorn iterations. Each plotted line is the 1-D middle slice of a 2-D image. Vertical dotted line are also drawn at the maximum of each interpolant, to better visualize their spacing. We notice that there is a trade-off between the smoothness of interpolation, and the spacing equality between interpolants. An equal spacing translates a constant speed interpolation.

The first regularizer f_c effectively stabilizes the values around 1, but the recovered metric is noisy, with patterns that reflect over-fitting. The second regularizer f_s effectively produces a smooth metric, but we note that the metric values have drawn away from their initial value of 1. After experimenting with each one, we observed that while reconstruction errors are smaller with f_c (which is another sign of overfitting), the regularizer f_s produces more interpretable results, and allows the global metric scale to shift in order to adapt to the input sequence. Moreover, combining both generally does not significantly change the result compared to having only f_s. Finally, tuning the λ_s parameter allows the user to specify the desired smoothing scale (max spatial frequency) in the final metric.

5.2.3 Initialization

Since the problem we are addressing is non-convex, the initialization of the metric weights is expected to have non-negligible effects on the final result. In Figure 10, we present the end metric of the experiment in Figure 4 with $\lambda_s = 0.3$, and for three different initializations: (1) constant initialization to 1, (2) random initialization in [0.3,3] uniformly in log scale, and (3) random initialization in [0.1,10] uniformly in log scale. We observe that the level of noise in (2) does not change the result significantly, but the one in (3) does. In (2), the initial noise did not impact the final result, because it has been smoothed out by the regularization. We conclude that the algorithm allows for some noise in the initialization, but a noise level that is too high cannot be smoothed out by the regularizer, and impacts the reconstruction and the final metric significantly.

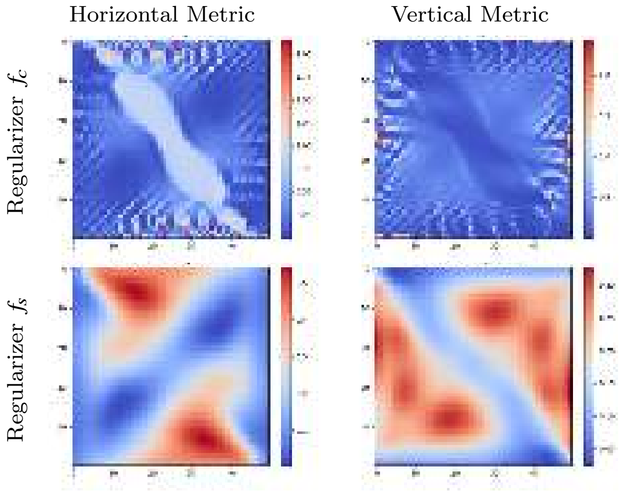

Fig. 9: In this experiment, we show the effects of each regularizer (f_c and f_s) on the final metric, using the experiment presented in Figure 4. First row: metric obtained with $\lambda_c = 0.03, \lambda_s = 0$. Second row: metric obtained with $\lambda_c = 0, \lambda_s = 0.03$. f_c constrains the weights to be close to 1, while f_s constrains them to be spatially smooth.

5.2.4 Loss function

The choice of the loss function $\mathcal{L}$ is left to the user, depending on what works best with their application. In Figure 11, we show three 2-D metrics learned on the synthetic experiment described in Figure 4, using the different loss functions (12),(13) and (14)

5.2.5 Timing and memory

In Table 1, we give the time and memory requirements of our algorithm, depending on the problem parameters. We use the same type of measures as previously described, that is, those defined on graphs that are d-dimensional Cartesian grids. The problem size is therefore $N = n^d$ and S is the number of sub-steps to solve the heat equation. The entropic regularization factor ε (which is used here as a diffusion time) does not affect the runtime. We give the timings for 500 L-BFGS iterations, which in our use cases, was generally sufficient for the algorithm to converge.

This algorithm is difficult to parallelize because we need to solve a very large number of medium-size linear systems, which individually do not benefit from multithreading. Giving more than one thread to the algo-

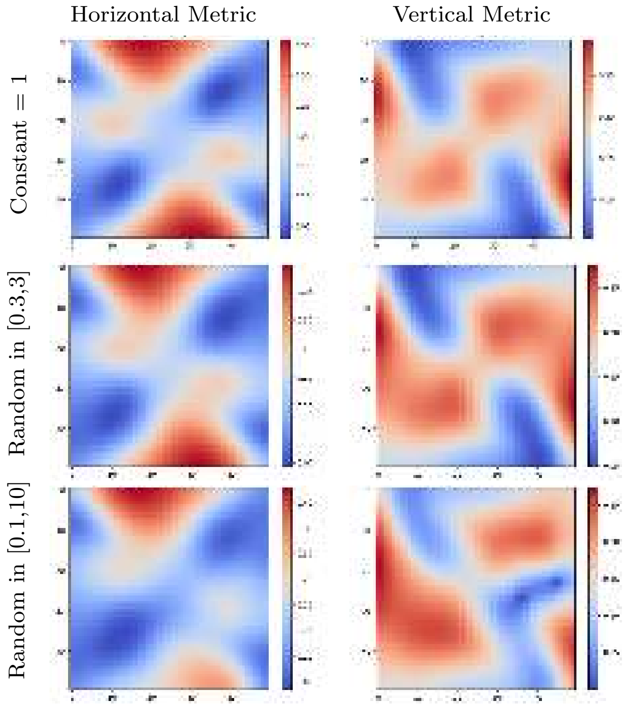

Fig. 10: We present the final metric of the experiment in Figure 4 for 3 different initializations: constant with weights equal to 1, random in log space in [0.3,3], and random in log space in [0.1,10]. The algorithm is robust to a half order of magnitude in the metric weights, but not to a full one.

d	n	N	S	$t_{500}(h)$	Mem.(GB)	Threads
2	50	2500	20	1.6	1.3	1
2	50	2500	100	7	4.7	1
2	100	10000	20	13	4	1
2	100	10000	100	60	16	1
3	16	4096	20	9	1.7	1
3	16	4096	50	25	3.3	1
3	16	4096	100	46	6.2	1
3	32	32768	20	110	10.9	8

Table 1: Time and memory requirements of our algorithm, with regard to problem size $N = n^d$ and S the number of sub-steps for solving the heat equation. "t_{500}" is the time it takes to run 500 iterations of L-BFGS, expressed in hours (h). "Mem." is the maximum resident memory that the algorithm requires, and "Threads" is the number of threads it runs on.

rithm was only faster for $N = 32^3$. If instead we parallelize over input images (we generally have around 10), the memory footprint grows 10 times, because the implementation is in Python, which duplicates memory for multi-processing.

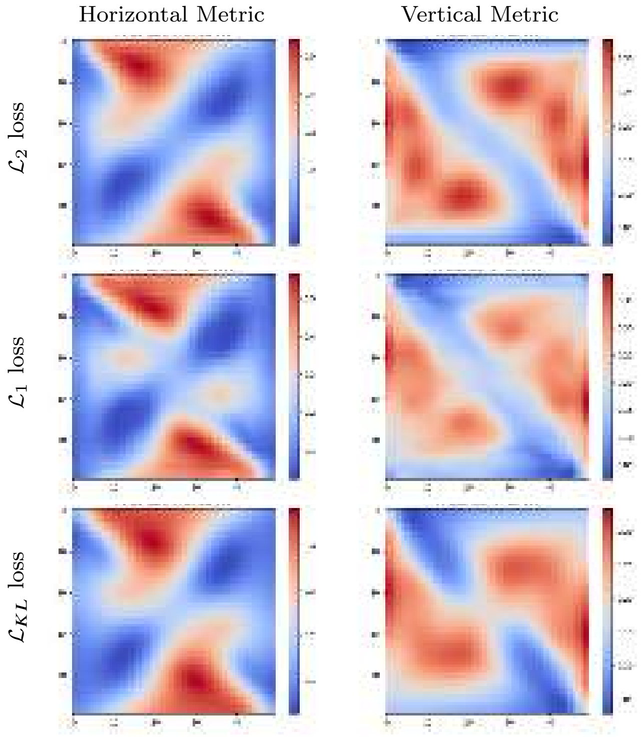

Fig. 11: The loss function $\mathcal{L}$ influences the resulting metric. We present the metric learned during three experiments using the same parameters, but three different loss functions $\mathcal{L}_2$, $\mathcal{L}_1$ and $\mathcal{L}_{KL}$. The experiment is the one described in Figure 4. One must choose the loss function depending on the application.

5.3 Learning color evolutions

We now demonstrate an application of our algorithm that deals with 3-D color histograms in the RGB color space. An important question in imaging and learning is which color space to use. The RGB space is simple to use, but variations in that space do not reflect variations of color perceived by the human eye. Other spaces, such as L*a*b* or L*u*v* have been designed to counteract this, and match variations in perception and space. Learning a ground color metric is a way to automatically fit the color space to the problem under consideration. Note that the problem of color metric learning in psychophysics has a long history, starting with the idea of MacAdam's ellipses [32], which introduces a Riemannian metric (corresponding to ellipses) to fit perceptual thresholds.

Given an input sequence of sunset images (for example Figure 12), we compute each input's color histogram, and use our algorithm to learn the metric for which the histogram sequence resembles an optimal transport of mass. All sequences presented hereafter contain around ten frames, but we only show five of them for brevity. Our final goal is to create a new sunset sequence from a new pair of day/night images, by interpolating between them using the learned metric, and transferring the interpolated color histograms onto the day image. Transferring the colors of an image (or of a color histogram) onto another image can be done via regularized OT, since the transport plan defines a mapping between the two histograms. This transfer is performed using the *barycentric projection map* followed by a bilateral filtering of the resulting image. The reader can refer to Peyré and Cuturi [38] and Solomon et al. [45] for more details.

5.3.1 Validating a learned metric

Before we apply a learned metric to a new dataset, we verify if it performs well on the training dataset. We first learn a metric from the *meteora2* sequence (Figure 12), and visualize the reconstruction of the input histogram sequence, at the end of the metric learning process (Figure 13).

We then check if the histograms reconstructed using the learned metric (Figure 13 second row) yield an image sequence that is close to the original one. Therefore, instead of transferring these histograms on a new day image, we transfer them on the first frame of the same sequence. In Figure 14, we compare this image sequence (fourth row) with two other methods of interpolation between the first and last frame, and a direct color transfer (no interpolation). The first row is the *meteora2* dataset (which is the ground truth), the second row is computed with a linear interpolation, and the third row is computed through a displacement interpolation with a Euclidean metric. Finally, the last row shows a direct color transfer of each frame of *meteora2* on its first frame.

Even though the differences are subtle, we can see that with our method of interpolation (using the learned metric), the colors (especially the red/orange tone) are closer to the ground truth and the direct transfer, than with a linear or Euclidean OT interpolation.

5.3.2 Reusing the learned metric

We now create a new sunset sequence from a pair of new day/night images, as described earlier. The image pair is extracted from the *country1* dataset (Figure 15), where we take the first and the last frame. We first learn a metric on the *seldovia2* dataset (Figure 16), with histograms of size 16^3, the $\mathcal{L}_2$ loss, 50 Sinkhorn iterations, 500 L-BFGS iterations, an entropic regularization of $\varepsilon = 4e-3$, $S = 20$ sub-steps, and a metric regularizer parameter $\lambda_s = 1$. Next, we interpolate between the day and night histograms of the *country1* dataset, using the learned metric, which is upsampled to 31^3 in order to decrease color quantization errors. Finally, we transfer each interpolated color histograms on the day frame to reproduce a sunset sequence.

In Figure 17, we compare the histogram sequence interpolated using the learned metric (3rd row), with a linear interpolation (1st row), and a displacement interpolation with a Euclidean metric (2nd row). In rows 2-4 of Figure 18, we show the color transfer of each interpolated histogram of Figure 17 onto the day frame. In the first row, we show the *country1* dataset, which constitutes a ground truth, and in row 5, we compare the results with a direct transfer of the *seldovia2* dataset on the day image of the *country1* dataset.

With linear interpolation, mass does not move. This results in color artifacts in our examples, such as clouds remaining bright and white until the very end of the sequence.

With the displacement interpolation with a Euclidean metric, mass travels in straight line between the first and last frame. We can see on the histogram sequence that mass travels close to the diagonal of the histogram, which represents the grey levels. Therefore, the resulting image sequence shows no red/orange tones that are typical of a sunset.

With our method, mass does not travel in straight line, by virtue of the learned metric. We can see on the histogram sequence that mass bypasses the diagonal of greys, and therefore travels in the red/orange tones, which results in an image sequence that is much closer to the ground truth. However, we also notice that the mass is split into two packages, one going through the red region, and the other one through the blue region, resulting in the sky getting bluer in the image sequence. This is an artifact that can be explained by the relatively high diffusion level required to have equally spaced interpolations, as explained in Figure 8.

Finally, a direct transfer also gives a plausible sunset sequence, however, the original colors of the target dataset (*country1*) are not preserved. Moreover, our method allows interpolation with an arbitrary number of frames, whereas the direct transfer can only produce the number of frames available in the source dataset.

6 Discussion

The problem we tackle is ill-posed and in general there is no way to find information where mass does not travel. Nevertheless, our regularization of the problem reduces the number of local minima and reduces

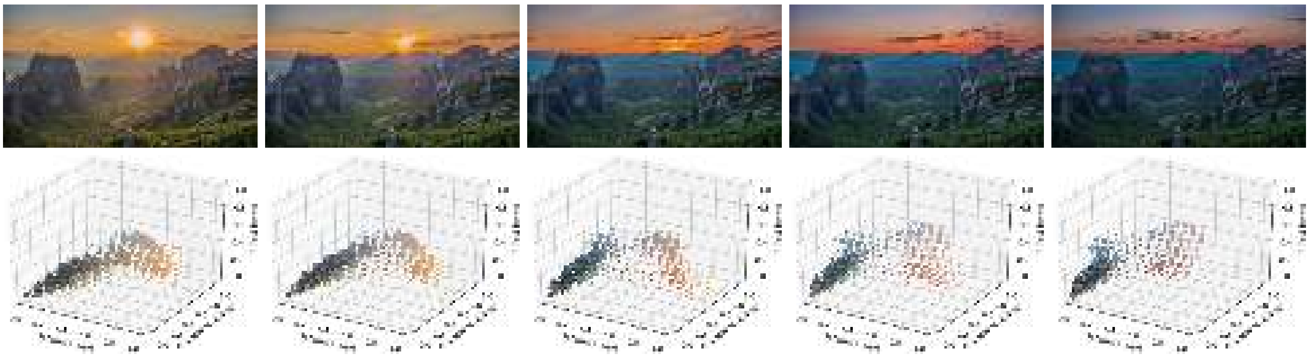

Fig. 12: The *meteora2* dataset: images (first row) and color histograms (second row). Video courtesy of PG ViSUAL

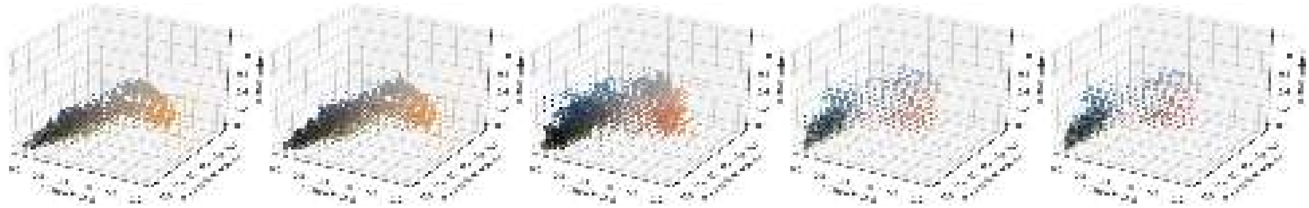

Fig. 13: Reconstruction of the color histogram sequence of the *meteora2* dataset (Figure 12), using the metric learned from it, after 500 iterations of our algorithm. The ground truth is in the second row of Figure 12. We notice that the reconstructions are more diffuse than the inputs, due to the entropic regularization.

the non-convexity by imposing spatially smooth metric weights, which also avoids over-fitting.

The parameterization we chose for the metric is limited in the sense that it only includes diffusion along the grid axes, which leads to low-precision approximations of the heat kernel on small domains. This approximation affects the quality of the displacement interpolations, as seen in Figure 8. It leads to a trade-off when choosing ε, between the smoothness of the interpolations, the regularity with which they are spread out spatially, and the computational limits it involves (as S increases). Moreover, as pointed out in [15, Appendix A], low values for the parameter ε yield a distance that is closer to the graph distance (number of edges), than to the geodesic distance. This means that as ε decreases, the edge weights have diminishing influence.

Although we managed to develop a tractable framework, as compared for instance to using a dense storage of the cost matrix, this algorithm remains computationally expensive for histograms with more than 10 000 points (see Table 1).

7 Conclusion

We have proposed a new method to learn the ground metric of optimal transport, as a geodesic distance on the graph supporting the data. We learn from observations of a mass displacement and aim to reconstruct them using displacement interpolations. We were able to turn a challenging task in terms of time and memory complexity into a tractable framework, using diffusion-based distance computations, regularized Wasserstein barycenters, and automatic differentiation. We demonstrated our method on toy examples, as well as on a color transfer application, where we learn the evolution of colors during a sunset, and use it to create a new sunset sequence. We finally discussed the limitations of the proposed method: our parametrization of the metric might be too simple, which limits the precision of the geodesic distances approximation. In turn, this impacts the interpolation, and adds trade-offs between having sharp interpolants, equally-spaced interpolants and the computational effort required for these.

Future work. For regular domains such as images and surfaces, it is possible to use a more precise approximation of a Riemannian metric as a field of tensors instead of a graph, as done for instance in [34], which in turn can be combined with triangulated meshes. Multi-resolution strategies can also be integrated into our pipeline to accelerate the linear system resolution and Sinkhorn algorithm (as proposed by [24]). In this paper, we conducted only preliminary experiments for learning from multiple sequences. The results seem promising, and we could imagine replacing our quasi-Newton optimizer by a stochastic gradient descent to potentially accelerate the learning. Unbalanced optimal

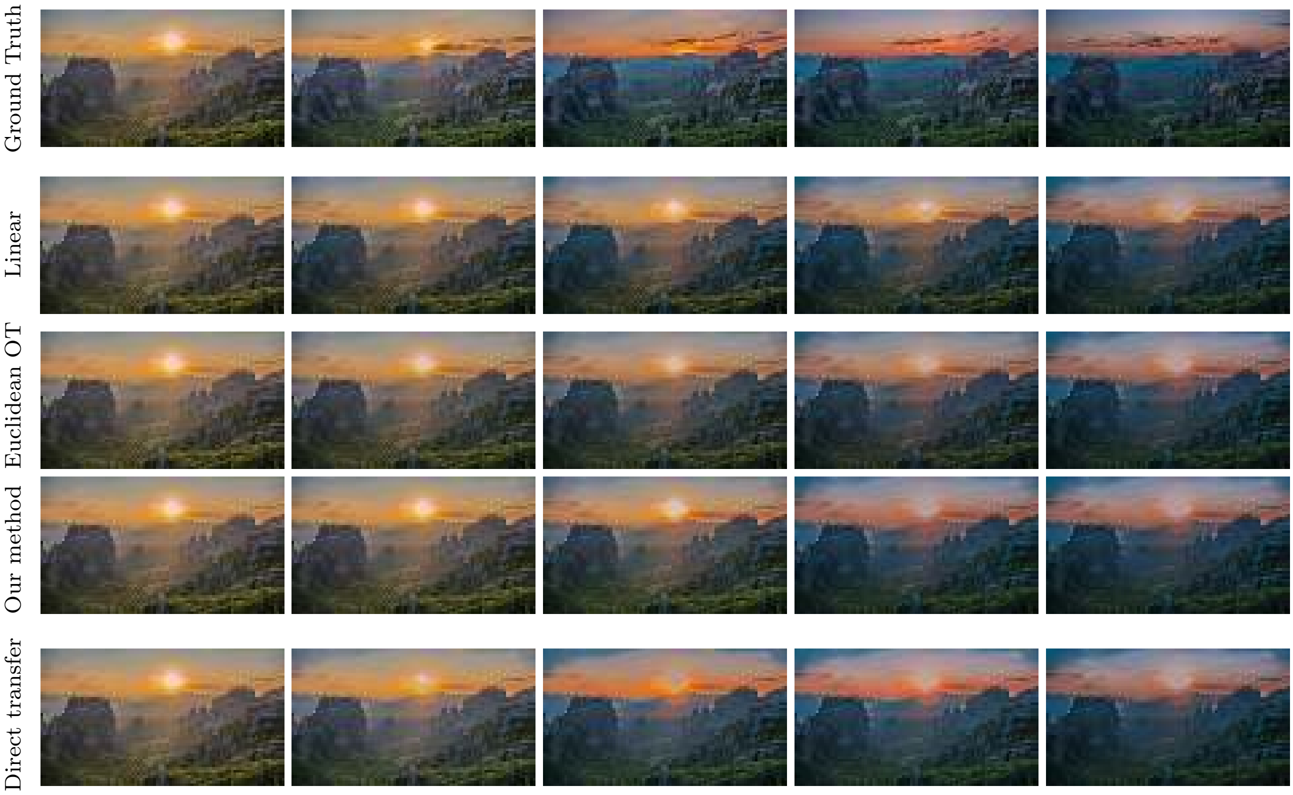

Fig. 14: Preliminary experiment: we learn a metric on the *meteora2* dataset (top row), then reinterpolate color histograms between the first and last frames using different methods, and transfer each interpolated histogram unto the first frame. The interpolation methods are: linear interpolation (second row), displacement interpolation with a Euclidean metric (third row), displacement interpolation with the learned metric (fourth row). The last row does not involve interpolation and is simply the color transfer of each frame of the ground truth onto the first frame.

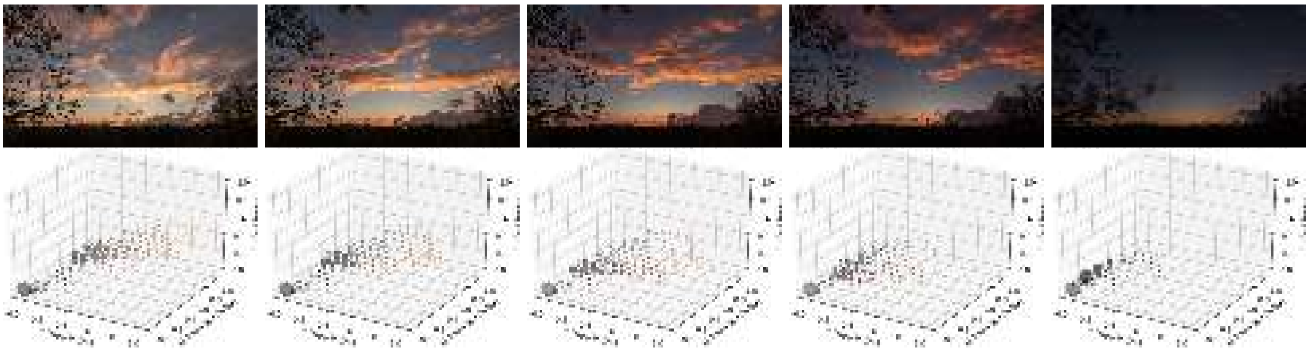

Fig. 15: The *country1* dataset: images (first row) and color histograms (second row). Video courtesy of Quincy van den Boom

transport [12] could also be valuable to account for mass creation and elimination during transport, which is crucial for some applications in chemistry or biology.

Acknowledgements This work was supported by the French National Research Agency (ANR) through the ROOT research grant (ANR-16-CE23-0009). The work of G. Peyré was supported by the European Research Council (ERC project NORIA) and by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

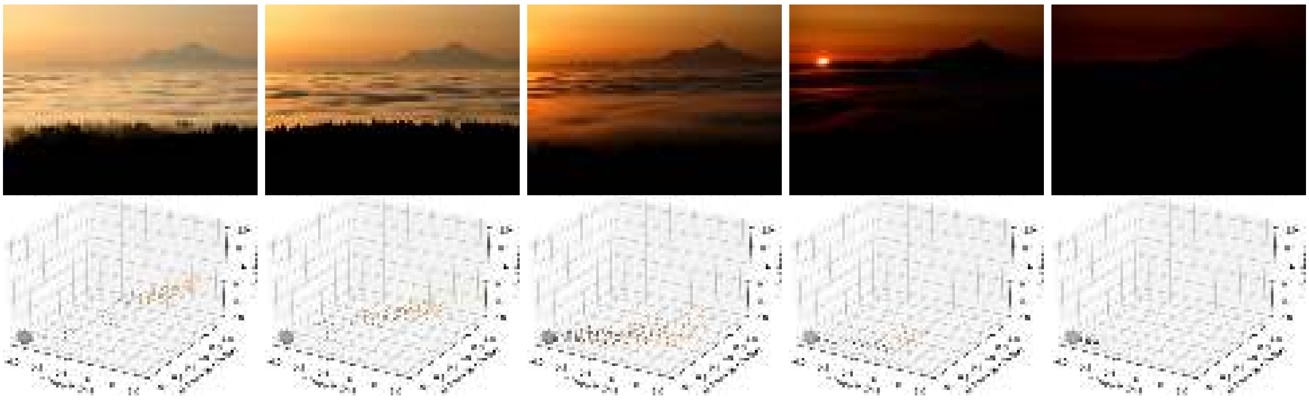

Fig. 16: The *seldovia2* dataset: images (first row) and color histograms (second row). Video courtesy of Bretwood Higman

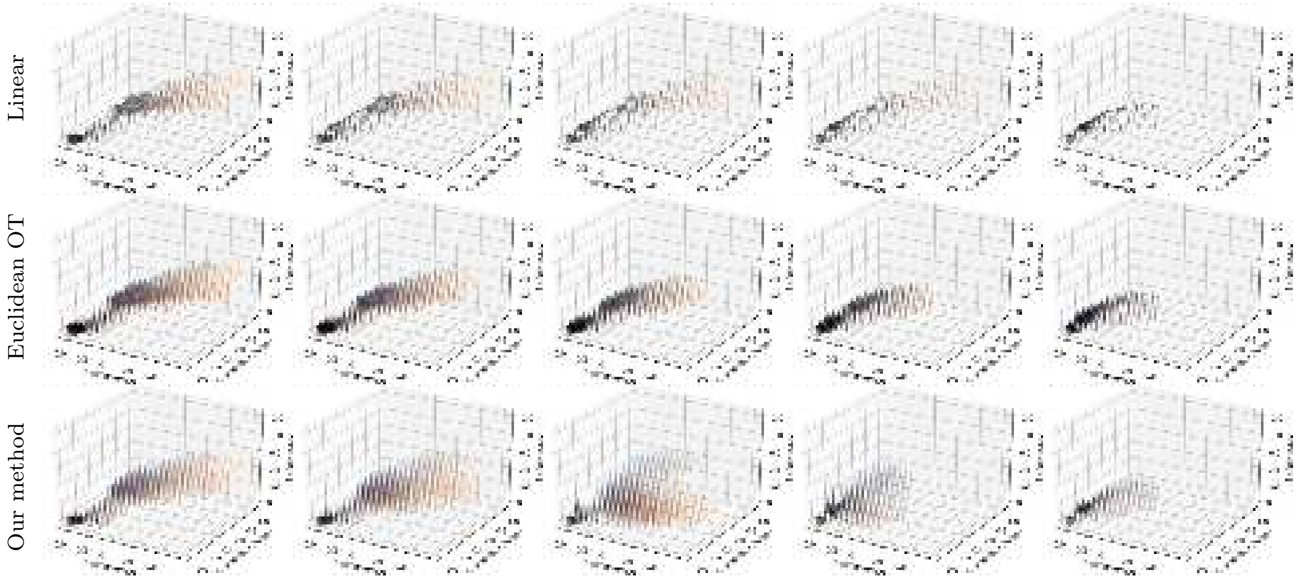

Fig. 17: Interpolation between day and night histograms of the *country1* dataset (Figure 15) using: linear interpolation (1st row), OT with Euclidean metric (2nd row) and OT with the metric learned on the *seldovia2* (Figure 16) dataset (3rd row). Color transfers using these histograms are presented in Figure 18.

References

1. Agueh, M., Carlier, G.: Barycenters in the Wasserstein Space. SIAM Journal on Mathematical Analysis **43**(2), 904–924 (2011). DOI 10.1137/100805741
2. Altschuler, J., Bach, F., Rudi, A., Weed, J.: Massively scalable Sinkhorn distances via the Nyström method. arXiv:1812.05189 [cs, math, stat] (2018)
3. Altschuler, J., Weed, J., Rigollet, P.: Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. arXiv preprint arXiv:1705.09634 (2017)
4. Angenent, S., Haker, S., Tannenbaum, A.: Minimizing Flows for the Monge-Kantorovich Problem. SIAM Journal on Mathematical Analysis **35**(1), 61–97 (2003). DOI 10.1137/S0036141002410927
5. Bellet, A., Habrard, A., Sebban, M.: Metric Learning. Synthesis Digital Library of Engineering and Computer Science. San Rafael, California (1537 Fourth Street, San Rafael, CA 94901 USA): Morgan & Claypool (2015)
6. Benamou, J.D., Carlier, G., Cuturi, M., Nenna, L., Peyré, G.: Iterative Bregman projections for regularized transportation problems. SIAM Journal on Scientific Computing **37**(2), A1111–A1138 (2015)
7. Benmansour, F., Carlier, G., Peyré, G., Santambrogio, F.: Derivatives with Respect to Metrics and Applications: Subgradient Marching Algorithm. Numerische Mathematik **116**(3), 357–381 (2010). DOI 10.1007/s00211-010-0305-8

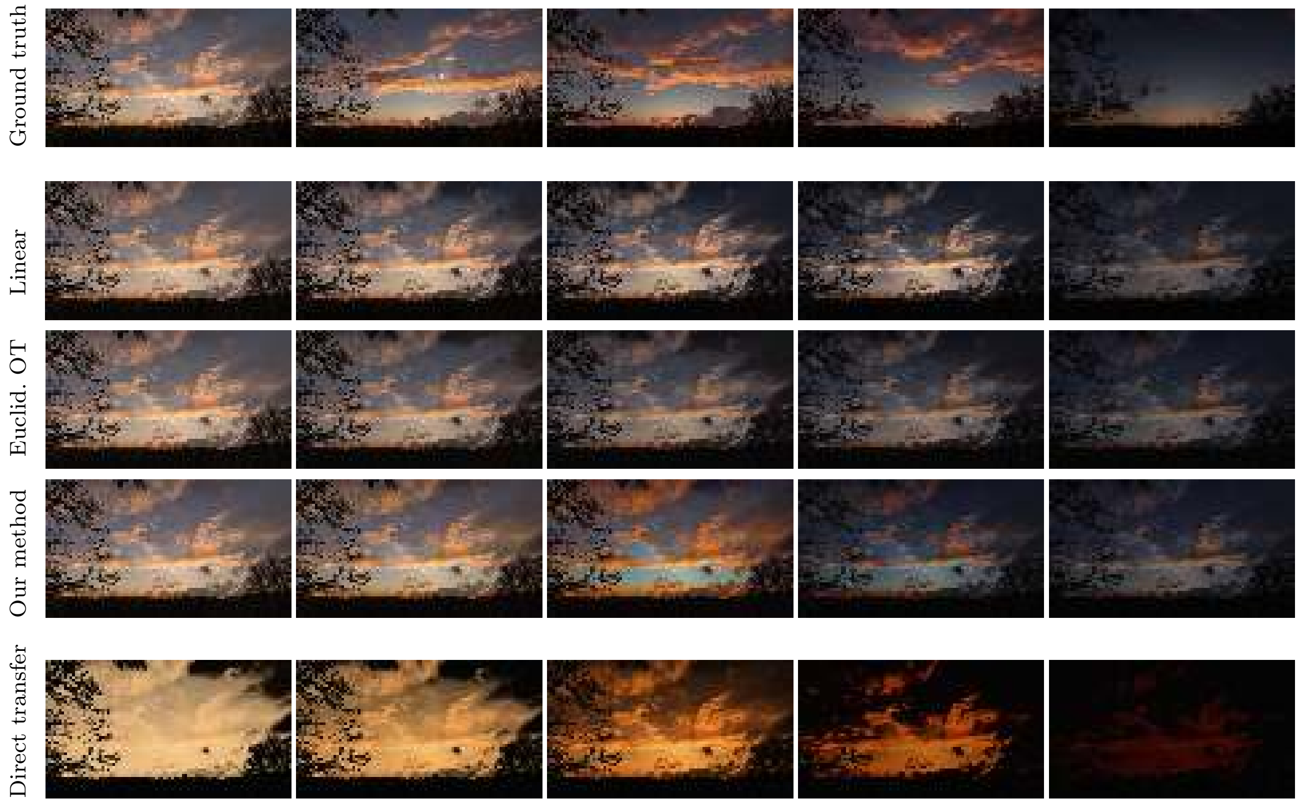

Fig. 18: 1st row: ground truth, the *country1* dataset. Rows 2-4: color transfer of each interpolated histograms in Figure 17. When comparing with the ground truth, we see that our method recreates sunset-like colors, as opposed to the two other methods. Row 5: direct transfer of each frame of the *seldovia2* dataset on the first frame of the *country1* dataset. Our method is able to preserve the original colors of the day image, contrary to a direct transfer.

8. Bonneel, N., Peyré, G., Cuturi, M.: Wasserstein barycentric coordinates: Histogram regression using optimal transport. ACM Transactions on Graphics **35**(4), 1–10 (2016). DOI 10.1145/2897824.2925918
9. Brickell, J., Dhillon, I.S., Sra, S., Tropp, J.A.: The Metric Nearness Problem. SIAM Journal on Matrix Analysis and Applications **30**(1), 375–396 (2008). DOI 10.1137/060653391
10. Buttazzo, G., Davini, A., Fragalà, I., Macià, F.: Optimal Riemannian distances preventing mass transfer. Journal für die reine und angewandte Mathematik (Crelles Journal) **2004**(575) (2004). DOI 10.1515/crll.2004.077
11. Chechik, G., Shalit, U., Sharma, V., Bengio, S.: An Online Algorithm for Large Scale Image Similarity Learning. Advances in Neural Information Processing Systems p. 9 (2009)
12. Chizat, L., Peyré, G., Schmitzer, B., Vialard, F.X.: Scaling algorithms for unbalanced transport problems. arXiv preprint arXiv:1607.05816 (2016)
13. Chopra, S., Hadsell, R., LeCun, Y.: Learning a Similarity Metric Discriminatively, with Application to Face Verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 539–546. IEEE, San Diego, CA, USA (2005). DOI 10.1109/CVPR.2005.202
14. Courty, N., Flamary, R., Tuia, D., Rakotomamonjy, A.: Optimal Transport for Domain Adaptation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(9), 1853–1865 (2016). DOI 10.1109/TPAMI.2016.2615921
15. Crane, K., Weischedel, C., Wardetzky, M.: Geodesics in heat: A new approach to computing distance based on heat flow. ACM Transactions on Graphics (TOG) **32**(5), 152 (2013)
16. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Advances in Neural Information Processing Systems, pp. 2292–2300 (2013)
17. Cuturi, M., Avis, D.: Ground metric learning. Journal of Machine Learning Research **15**(1), 533–564 (2014)
18. Cuturi, M., Doucet, A.: Fast computation of Wasserstein barycenters. In: International Confer-

ence on Machine Learning, pp. 685–693 (2014)
19. Dognin, P., Melnyk, I., Mroueh, Y., Ross, J., Santos, C.D., Sercu, T.: Wasserstein Barycenter Model Ensembling. arXiv:1902.04999 [cs, stat] (2019)
20. Dupuy, A., Galichon, A., Sun, Y.: Estimating matching affinity matrix under low-rank constraints. arXiv:1612.09585 [stat] (2016)
21. Dvurechensky, P., Gasnikov, A., Kroshnin, A.: Computational Optimal Transport: Complexity by Accelerated Gradient Descent Is Better Than by Sinkhorn's Algorithm. arXiv:1802.04367 [cs, math] (2018)
22. Frogner, C., Zhang, C., Mobahi, H., Araya, M., Poggio, T.A.: Learning with a Wasserstein Loss. Advances in Neural Information Processing Systems p. 9 (2015)
23. Genevay, A., Peyré, G., Cuturi, M.: Learning Generative Models with Sinkhorn Divergences. arXiv:1706.00292 [stat] (2017)
24. Gerber, S., Maggioni, M.: Multiscale strategies for computing optimal transport. arXiv preprint arXiv:1708.02469 (2017)
25. Griewank, A.: Who Invented the Reverse Mode of Differentiation? Documenta Mathematica p. 12 (2012)
26. Griewank, A., Walther, A.: Evaluating Derivatives. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (2008). DOI 10.1137/1.9780898717761
27. Huang, G., Guo, C., Kusner, M.J., Sun, Y., Sha, F., Weinberger, K.Q.: Supervised word mover's distance. In: Advances in Neural Information Processing Systems, pp. 4862–4870 (2016)
28. Kedem, D., Tyree, S., Sha, F., Lanckriet, G.R., Weinberger, K.Q.: Non-linear Metric Learning. Neural Information Processing Systems (NIPS) p. 9 (2012)
29. Kulis, B.: Metric Learning: A Survey. Foundations and Trends® in Machine Learning **5**(4), 287–364 (2013). DOI 10.1561/2200000019
30. Lévy, B.: A Numerical Algorithm for L2 Semi-Discrete Optimal Transport in 3D. ESAIM: Mathematical Modelling and Numerical Analysis **49**(6), 1693–1715 (2015). DOI 10.1051/m2an/2015055
31. Li, R., Ye, X., Zhou, H., Zha, H.: Learning to Match via Inverse Optimal Transport. Journal of Machine Learning Research p. 37 (2019)
32. MacAdam, D.L.: Visual Sensitivities to Color Differences in Daylight. Journal of the Optical Society of America **32**(5), 247 (1942). DOI 10.1364/JOSA.32.000247
33. McCann, R.J.: A Convexity Principle for Interacting Gases. Advances in Mathematics **128**(1), 153–179 (1997). DOI 10.1006/aima.1997.1634
34. Mirebeau, J.M., Dreo, J.: Automatic differentiation of non-holonomic fast marching for computing most threatening trajectories under sensors surveillance. arXiv:1704.03782 [math] (2017)
35. Papadakis, N., Peyré, G., Oudet, E.: Optimal Transport with Proximal Splitting. SIAM Journal on Imaging Sciences **7**(1), 212–238 (2014). DOI 10.1137/130920058
36. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch p. 4 (2017)
37. Pele, O., Ben-Aliz, Y.: Interpolated Discretized Embedding of Single Vectors and Vector Pairs for Classification, Metric Learning and Distance Approximation. arXiv:1608.02484 [cs] (2016)
38. Peyré, G., Cuturi, M.: Computational Optimal Transport. Now Publishers, Inc. (2018)
39. Rubner, Y., Tomasi, C., Guibas, L.J.: The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision p. 23 (2000)
40. Sandler, R., Lindenbaum, M.: Nonnegative Matrix Factorization with Earth Mover's Distance Metric for Image Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(8), 1590–1602 (2011). DOI 10.1109/TPAMI.2011.18
41. Santambrogio, F.: Optimal Transport for Applied Mathematicians. Springer (2015)
42. Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., Lee, L., Chen, J., Brumbaugh, J., Rigollet, P., Hochedlinger, K., Jaenisch, R., Regev, A., Lander, E.S.: Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming. Cell **176**(4), 928–943.e22 (2019). DOI 10.1016/j.cell.2019.01.006
43. Schmitz, M.A., Heitz, M., Bonneel, N., Ngolè Mboula, F.M., Coeurjolly, D., Cuturi, M., Peyré, G., Starck, J.L.: Wasserstein Dictionary Learning: Optimal Transport-based unsupervised non-linear dictionary learning. SIAM Journal on Imaging Sciences (2018)
44. Simou, E., Frossard, P.: Graph Signal Representation with Wasserstein Barycenters. arXiv:1812.05517 [eess] (2018)
45. Solomon, J., de Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., Guibas, L.: Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. ACM Trans. Graph. **34**(4), 66:1–66:11 (2015). DOI

10.1145/2766963

46. Stuart, A.M., Wolfram, M.T.: Inverse optimal transport. arXiv:1905.03950 [math, stat] (2019)
47. Torresani, L., Lee, K.c.: Large Margin Component Analysis. Advances in neural information processing systems p. 8 (2007)
48. Varadhan, S.R.S.: On the behavior of the fundamental solution of the heat equation with variable coefficients. Communications on Pure and Applied Mathematics **20**(2), 431–455 (1967)
49. Wang, F., Guibas, L.J.: Supervised Earth Mover's Distance Learning and Its Computer Vision Applications. In: Computer Vision – ECCV 2012, vol. 7572, pp. 442–455. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). DOI 10.1007/978-3-642-33718-5_32
50. Wang, J., Do, H.T., Woznica, A., Kalousis, A.: Metric Learning with Multiple Kernels. Advances in Neural Information Processing Systems p. 9 (2011)
51. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. Advances in neural information processing systems p. 8 (2006)
52. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: Proceedings of the 25th International Conference on Machine Learning - ICML '08, pp. 1160–1167. ACM Press, Helsinki, Finland (2008). DOI 10.1145/1390156.1390302
53. Xing, E.P., Jordan, M.I., Russell, S.J., Ng, A.Y.: Distance Metric Learning with Application to Clustering with Side-Information. In: Advances in Neural Information Processing Systems, p. 8 (2003)
54. Xu, J., Luo, L., Deng, C., Huang, H.: Multi-Level Metric Learning via Smoothed Wasserstein Distance. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, pp. 2919–2925. International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden (2018). DOI 10.24963/ijcai.2018/405
55. Yang, F., Cohen, L.D.: Geodesic Distance and Curves Through Isotropic and Anisotropic Heat Equations on Images and Surfaces. Journal of Mathematical Imaging and Vision **55**(2), 210–228 (2016). DOI 10.1007/s10851-015-0621-9
56. Yang, W., Xu, L., Chen, X., Zheng, F., Liu, Y.: Chi-Squared Distance Metric Learning for Histogram Data. Mathematical Problems in Engineering **2015**, 1–12 (2015). DOI 10.1155/2015/352849
57. Zen, G., Ricci, E., Sebe, N.: Simultaneous Ground Metric Learning and Matrix Factorization with Earth Mover's Distance. In: 2014 22nd International Conference on Pattern Recognition, pp. 3690–3695 (2014). DOI 10.1109/ICPR.2014.634

A Elements of proof for Proposition 1

The mapping $\phi_1 : \mathbf{w} \in \mathbb{R}^K \to \mathbf{M} \in \mathbb{R}^{N^2}$ admits as adjoint Jacobian:

$$[\partial\phi_1(\mathbf{w})]^T (\mathbf{X})_{i,j} = -\frac{\varepsilon}{4S}\left(\mathbf{X}_{i,j} + \mathbf{X}_{j,i} - \mathbf{X}_{i,i} - \mathbf{X}_{j,j}\right). \quad (15)$$

The mapping $\phi_2 : \mathbf{M} \in \mathbb{R}^{N^2} \to \mathbf{U} = \mathbf{M}^{-1} \in \mathbb{R}^{N^2}$ admits as adjoint Jacobian:

$$[\partial\phi_2(\mathbf{M})]^T (\mathbf{X}) = -\mathbf{M}^{-1}\mathbf{X}\mathbf{M}^{-1}. \quad (16)$$

The mapping $\phi_3 : \mathbf{U} \in \mathbb{R}^{N^2} \to \mathbf{V} = \mathbf{U}^S \in \mathbb{R}^{N^2}$ admits as adjoint Jacobian:

$$[\partial\phi_3(\mathbf{U})]^T (\mathbf{X}) = \sum_{l=0}^{S-1} \mathbf{U}^l\mathbf{X}\mathbf{U}^{S-l-1}. \quad (17)$$

The mapping $\phi_4 : \mathbf{V} \in \mathbb{R}^{N^2} \to \mathbf{y} = \mathbf{V}\mathbf{v} \in \mathbb{R}^N$ admits as adjoint Jacobian:

$$[\partial\phi_4(\mathbf{V})]^T (\mathbf{x}) = \mathbf{x}\mathbf{v}^T. \quad (18)$$

Since $\Phi = \phi_4 \circ \phi_3 \circ \phi_2 \circ \phi_1$, we compose the adjoint Jacobians in the reverse order as follows:

$$[\partial\Phi(\mathbf{w})]^T (\mathbf{g})_{i,j} = [\partial\phi_1]^T [\partial\phi_2]^T [\partial\phi_3]^T [\partial\phi_4]^T (\mathbf{g})_{i,j}, \quad (19)$$

to finally obtain:

$$[\partial\Phi(\mathbf{w})]^T (\mathbf{g})_{i,j} = \frac{\varepsilon}{4S}\sum_{\ell=0}^{S-1}\left(\mathbf{g}_i^\ell - \mathbf{g}_j^\ell\right)\left(\mathbf{v}_i^\ell - \mathbf{v}_j^\ell\right), \quad (20)$$

$$\text{where} \quad \begin{cases} \mathbf{g}^\ell \overset{\text{def.}}{=} \mathbf{M}^{\ell-S}\mathbf{g} \\ \mathbf{v}^\ell \overset{\text{def.}}{=} \mathbf{M}^{-\ell-1}\mathbf{v}. \end{cases} \quad (21)$$