BSP-OT: Sparse transport plans between discrete measures in loglinear time

Baptiste Genest, Nicolas Bonneel, Vincent Nivoliers, David Coeurjolly

BSP-OT: Sparse transport plans between discrete measures in loglinear time

BAPTISTE GENEST, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, France
NICOLAS BONNEEL, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France
VINCENT NIVOLIERS, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, France
DAVID COEURJOLLY, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France

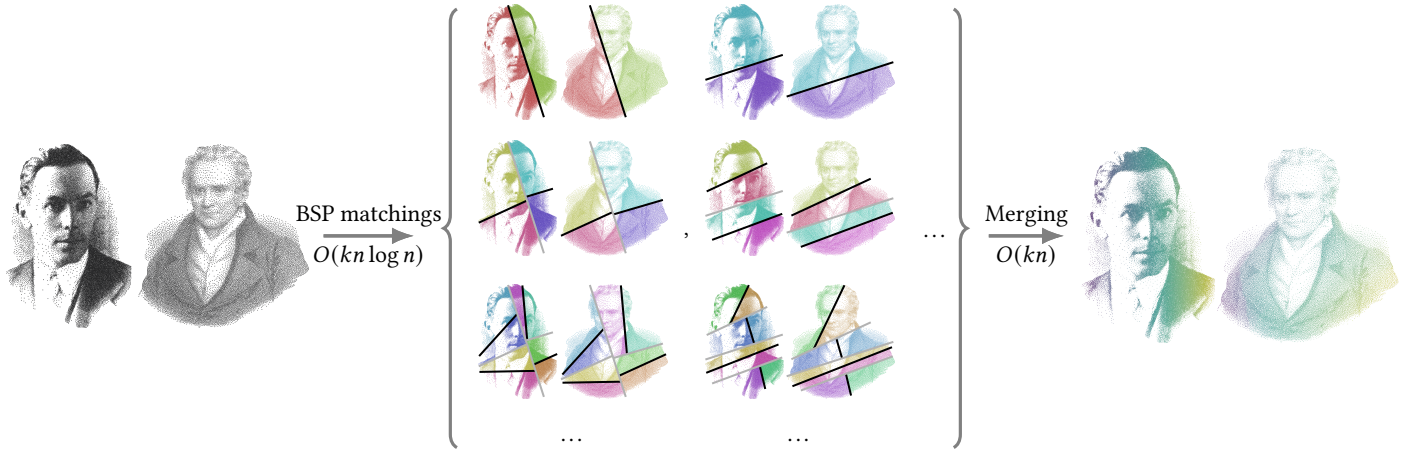


Fig. 1. To build a bijection between two point sets (here, stippled images of L. Kantorovich and G. Monge, left), we simultaneously build a pair of BSP-tree on both sets, by recursively splitting the points along the same random direction. We then match the corresponding leaves of the trees to obtain an assignation. After building such k assignations in parallel, we combine them into a single one, of low transport cost (color-coded on the right). This allows for numerous applications in computer graphics.

To solve the optimal transport problem between two uniform discrete measures of the same size, one seeks a bijective assignment that minimizes some matching cost. For this task, exact algorithms are intractable for large problems, while approximate ones may lose the bijectivity of the assignment. We address this issue and the more general cases of non-uniform discrete measures with different total masses, where partial transport may be desirable. The core of our algorithm is a variant of the Quicksort algorithm that provides an efficient strategy to randomly explore many relevant and easy-to-compute couplings, by matching BSP trees in loglinear time. The couplings we obtain are as sparse as possible, in the sense that they provide bijections, injective partial matchings or sparse couplings depending on the nature of the matched measures. To improve the transport cost, we propose efficient strategies to merge k sparse couplings into a higher quality one. For $k = 64$, we obtain transport plans with typically less than 1% of relative error in a matter of seconds between hundreds of thousands of points in 3D on the CPU. We demonstrate how these high-quality approximations can drastically speed-up usual pipelines involving optimal transport, such as shape interpolation, intrinsic manifold sampling, color transfer, topological data analysis, rigid partial registration of point clouds and image stippling.

CCS Concepts: • **Computing methodologies** → *Image manipulation*; *Shape modeling*.

Additional Key Words and Phrases: optimal transport, binary space partitioning, bijections, coupling, partial optimal transport

ACM Reference Format:
Baptiste Genest, Nicolas Bonneel, Vincent Nivoliers, and David Coeurjolly. 2025. BSP-OT: Sparse transport plans between discrete measures in loglinear time . *ACM Trans. Graph.* 44, 6 (December 2025), 15 pages. https://doi.org/10.1145/3763281

1 Introduction

Optimal transport naturally arises from a resource allocation problem. Given n sites each producing a unit of resource and n sites each consuming a unit of that resource, how should one assign each production site to a consumption site to minimize the total cost of transporting resources? This problem has seen many applications in computer graphics, where particles, Diracs, vertices, or colors have been interpreted as sites to be matched. It has thus been considered for image color matching, stippling, shape interpolation, fluid simulation and many other uses [Bonneel and Digne 2023].

In its simplest form, this problem can be formulated as finding a bijection $T^\star$ from the set of production sites $X = \{x_i\}$ to the

Authors' Contact Information: Baptiste Genest, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, France, baptiste.genest@liris.cnrs.fr; Nicolas Bonneel, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France, nicolas.bonneel@liris.cnrs.fr; Vincent Nivoliers, Université Claude Bernard Lyon 1, CNRS, INSA Lyon, France, vincent.nivoliers@liris.cnrs.fr; David Coeurjolly, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France, david.coeurjolly@cnrs.fr.

ACM 1557-7368/2025/12-ART
https://doi.org/10.1145/3763281

consumption sites $Y = \{y_i\}$ such that:

$$T^\star = \underset{T:X \xrightarrow{\text{bij}} Y}{\arg\min} \sum_{i=1}^{n} c(x_i, T(x_i)), \tag{1}$$

where c is typically a (power of a) distance. The combinatorial nature of the set of bijections makes this problem difficult to solve. Fortunately, it can be made much easier when formulated as a Linear Programming (LP) problem. Instead of looking for a map $T : X \to Y$, one looks for a coupling matrix $\pi = [\pi_{i,j} \in \mathbb{R}^+]_{i,j}$, where $\pi_{i,j}$ indicates the amount of mass sent from $x_i \in X$ to $y_j \in Y$. Mass preservation is expressed in terms of π's marginals: $\Sigma_n := \{\pi \in M_n(\mathbb{R}^+) \mid \sum_{i=1}^n \pi_{i,j} = 1 \text{ and } \sum_{j=1}^n \pi_{i,j} = 1\}$, and the assignment problem rewritten:

$$\pi^\star = \underset{\pi \in \Sigma_n}{\arg\min} \sum_{i,j} \pi_{i,j} c(x_i, y_j). \tag{2}$$

While this relaxes bijectivity, allowing mass to split, in practice, when considering that each production and consumption site carries a single unit of mass, the optimal coupling $\pi^\star$ remains a bijection. To our knowledge, no efficient and scalable algorithm (i.e., handling hundreds of thousands of points in seconds) produces bijections, approximating the optimal solution well. In addition, many problems of interest may involve more production sites n than consumption sites m ($m < n$) resulting in a search for injective maps (a so-called *partial transport problem*), or involve non-uniform mass distribution where couplings are sparse, but no transport map may exist.

In this paper, we introduce a way to produce and explore many sparse couplings in quasi-linear time and linear space complexity, based on Binary Space Partitioning (BSP). While each coupling is not optimal, we show that an efficient (linear-time in the bijective and partial cases) merging scheme exploiting their sparsity allows to sufficiently reduce the transportation cost and, in practice, yields approximations close to the optimal transport solution. We demonstrate important speedups on problems encountered in computer graphics such as shape interpolation and barycenters, blue noise sampling of surfaces and stippling of images, color transfer between images, partial point cloud registration, and persistence diagram matching in topological data analysis. In practice, we propose three variants of our BSP-based approach, depending on the settings, that either produce a bijective transport map in the simplest setting of matching point sets of the same cardinality (Sec. 3), an injective transport map when matching point sets of different cardinalities (Sec. 4) or a sparse coupling matrix when transporting weighted point sets, i.e., more general discrete measures, of equal total mass (Sec. 5).

2 Related works

General reviews of numerical methods for optimal transport can be found in the books of Peyré et al. [2019] and Santambrogio [2015]. A survey of Bonneel and Digne [2023] describes its use in computer graphics, and Khamis et al. [2024] reviews scalable optimal transport techniques in machine learning. We focus here on work related to providing discrete-to-discrete optimal transport couplings.

Exact solutions. The LP problem can be solved using a general LP solver such as the simplex algorithm [Nabli 2009] with an $O(n^3)$ complexity. The *network*-simplex algorithm [Orlin 1997], of the same complexity class, benefits from the bipartite structure of the optimal transport problem and runs faster. For bijective assignment problems, the *Hungarian method* [Kuhn 1955] computes the exact solution in $O(n^3)$ time, and variants in the geometric setting (where the cost is the Euclidean distance) in $O(n^2)$ [Gattani et al. 2023]. For data structured on grids and when the cost c is separable, complexity is reduced [Auricchio et al. 2018]. Approximations to the LP problem may be obtained, up to ε relative error, via the auction algorithm [Bertsekas 1990], or in $O(n \,\text{poly}(\log(n), \frac{1}{\varepsilon})$ using acceleration structures [Sharathkumar and Agarwal 2012]. ε-optimality is costly to ensure, making such a scheme unfit for large-scale problems.

Entropic regularization. To make the problem tractable, adding a regularization penalizing the entropy of the coupling matrix has proven very effective, but blurs the optimal coupling. The regularized problem is solved orders of magnitude faster using the Sinkhorn algorithm, possibly on the GPU [Cuturi 2013; Solomon et al. 2015]. Altschuler et al. [2019] uses an extremely fast low-rank approximation of π but at the price of higher entropy. Our approach instead relies on sparsity to enable efficient combinatorial optimization.

Sliced transportation. Sliced Optimal Transport (SOT) [Bonneel et al. 2015; Rabin et al. 2011] exploits the simplicity of discrete optimal transport in 1D, which boils down to sorting 1D points and assigning the i^{th} source to the i^{th} target. The sliced transportation distance between d-dimensional points is the average of 1-dimensional transport costs obtained by projections onto random 1D lines. This leads to an $O(n \log n)$ algorithm, efficiently implementable on the GPU, also benefiting machine learning [Deshpande et al. 2018; Heitz et al. 2021; Liutkus et al. 2019; Wu et al. 2019]. However, this simplicity comes at the cost of lacking an explicit coupling between measures. It is possible to perform a gradient flow of the SOT energy until convergence to recover a coupling, but this requires many gradient descent iterations that undermine its computational efficiency. Instead, a single well-chosen direction can be used to obtain an assignment from a single 1D problem [Mahey et al. 2023], but this results in significant approximation errors.

Multiscale approaches. Multiscale solvers can improve scalability, typically by building a partition of the two point clouds (e.g,, using k-means), and recursively matching and combining clusters [Blumberg et al. 2020; Oberman and Ruan 2015; Schmitzer 2016, 2019]. To our knowledge, the fastest such method for discrete optimal transportis the multiscale Sinkhorn algorithm of Feydy et al. [2019], implemented in the *Geomloss* library. Running on the GPU, it allows to compute registrations between hundreds of thousands of points in around 10s on high-end GPUs. Similar to sliced transport, they do not produce explicit couplings but register the source to the target measure via a gradient flow. Our approach runs faster, on the CPU, with comparable quality and directly provides couplings.

Particular optimal transport distances over (graph) trees, such as trees obtained from Bounding Volume Hierarchies, have a closed-form expression for specific costs $c(x, y)$. This allows to compute the optimal transport distance in linear time [Backurs et al. 2020; Le et al. 2019; Tran et al. 2025; Yamada et al. 2022], but does not provide explicit couplings. Nurbekyan et al. [2020] and Negrini and

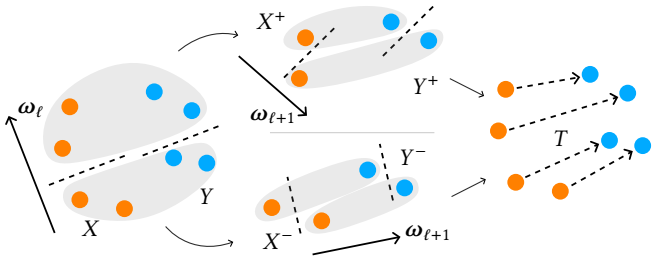


Fig. 2. We call BSP matchings between two point sets X and Y the following process: using linear slices we first split X into X^-, X^+ and Y into Y^-, Y^+, where X^- (resp. X^+) and Y^- (resp. Y^+) have the same number of points. Then we recursively match X^- to Y^- and X^+ to Y^+, until we reach a trivial case (here matching a single point of X to another of Y. When $n = m$ with uniform mass the result is a bijection, but we show how to generalize the process while keeping a sparse output assignment.

Nurbekyan [2024] obtain a *collision-free* transport map by simultaneously partitioning two measures into two BSP trees (kd-trees in practice) and matching their leaves. While they mainly study the properties of the continuous map obtained after infinite subdivisions of a single kd-tree, and numerically evaluate the discrete bijective case, we take inspiration from these works for more general discrete measures. We instead consider multiple BSP trees and extend BSP matchings to injective and non-uniform settings.

3 BSP Matching: the bijective case

We first describe BSP Matchings in the uniform balanced case, i.e., matching point sets X and Y of the same cardinality n. Sections 4 and 5 generalize our algorithm to different cardinalities and non-uniformly weighted points.

3.1 Randomized BSPs

By projecting onto 1D lines, sliced optimal transport loses significant information. The motivation behind BSP Matching is to keep SOT algorithmic simplicity and runtime complexity while reducing as much as possible information loss and benefiting from a multiscale strategy.

BSP matching proceeds by constructing BSP trees simultaneously on point sets X and Y. A BSP recursively splits a point set into two subsets, until each region only contains a single point (see Fig. 1). The key idea is to split simultaneously both X and Y: at each node of the tree, the two point sets are each divided into two parts. In the first (resp. second) part of X, there is the same number of points as in the first (resp. second) part of Y, and this number is expressed as a fraction ρ of the number of points in that level of the tree. While the work of Nurbekyan et al. [2020] considers splitting axis-aligned hyper-planes into two half-spaces containing the same number of points ($\rho = 0.5$) – resulting in a kd-tree – we instead generalize to unbalanced splits of arbitrary ρ with arbitrary hyper-plane orientations. For each node ℓ of the tree, a direction $\boldsymbol{\omega}_\ell$ is chosen, and point sets are split based on their dot product with $\boldsymbol{\omega}_\ell$, ensuring that both point sets contain the same number of points in the left (resp. right) side of the split.

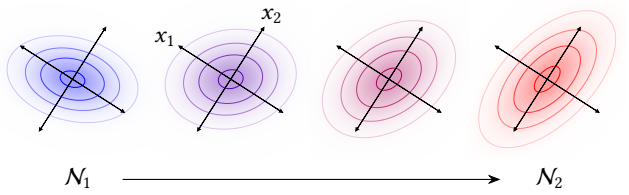


Fig. 3. Displacement interpolation with respect to the $\mathcal{W}_2$ optimal transport between two gaussians $\mathcal{N}_1(x_1, \mathrm{C}_1), \mathcal{N}_2(x_2, \mathrm{C}_2)$, expressed in closed form by an affine map $T_{\mathcal{N}_1,\mathcal{N}_2}(x) = A_{\mathrm{C}_1,\mathrm{C}_2}(x - x_1) + x_2$. One can observe that a point that starts its trajectory on any side of the eigenvectors (here noted x_1,x_2) of $A_{\mathrm{C}_1,\mathrm{C}_2}$, remains on the same side after transport. Hence we use such eigenvectors to define the slicing direction to build BSP matchings.

This process is illustrated in Fig. 2 and a pseudo-code implementation can be found in Alg. 1. The `QuickSelect` procedure (`std::nth_element` from the standard C++ library) rearranges an array in linear time based on the dot product with $\boldsymbol{\omega}_\ell$ such that the elements before the pivot index have smaller dot products than the ones after. The resulting algorithm runs in $O(n \log n)$.

Algorithm 1: `BSPMatching`

Data: $X = [x_1 \dots x_n], Y = [y_1 \dots y_n]$,
start $\in [\![1, n]\!]$, end $\in [\![1, n+1]\!]$, $T \in X \to Y$

```
if end − start = 1 then
    T(x_start) ← y_start;
    return
ω_ℓ ← GenerateSliceDirection();    // See Sec. 3.2
piv ← PickPivot(start,end); // Random int. or median
// reorder X and Y according to ω_ℓ and piv
QuickSelect(X, ω_ℓ, start, piv, end);
QuickSelect(Y, ω_ℓ, start, piv, end);
BSPMatching(X, Y, start, piv, T);
BSPMatching(X, Y, piv, end, T);
```

Interestingly, if $\boldsymbol{\omega}_\ell = \boldsymbol{\omega}$ is kept constant over the entire tree, then Alg. 1 boils down to a *QuickSort* algorithm where the sorting predicate is chosen to be the dot products with $\boldsymbol{\omega}_\ell$, i.e., exactly recovering the optimal 1D assignment along the $\boldsymbol{\omega}$ used in SOT. Remarkably, Feydy [2020] describe empirically the behavior of their multiscale Sinkhorn algorithm as a higher-dimensional generalization of the Quicksort algorithm, hinting at connections between numerical discrete OT and sorting.

3.2 Gaussian slicing for low dimensions

In the kd-tree approaches of Nurbekyan et al. [2020] and Negrini and Nurbekyan [2024], hyperplane normals $\boldsymbol{\omega}_\ell$ are chosen axis-aligned, and alternate between dimensions. While this strategy is simple, general and relatively efficient, it easily admits cases where the final matching is far from optimal. We propose a stochastic heuristic to efficiently split the point sets in a way that improves the quality of the matchings.

Randomly sampling the slicing direction at each node and choosing the quantity of mass that is sent left and right of the split maximizes the range of couplings explored. They may however not all be highly relevant, hence undermining the quality of the final merged matching. In lower-dimensional settings, we propose an alternative strategy that restrains the exploration to higher-quality ones.

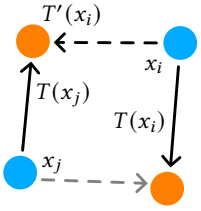
Our solution resides in a choice of directions that would actually separate the problem when matching (continuous) Gaussian distributions, for the quadratic cost. When transporting $\mathcal{N}_1 = \mathcal{N}(x_1, \mathrm{C}_1)$ towards $\mathcal{N}_2 = \mathcal{N}(x_2, \mathrm{C}_2)$, the optimal map is provided by $T_{\mathcal{N}_1,\mathcal{N}_2}(x) = A_{\mathrm{C}_1,\mathrm{C}_2}(x - x_1) + x_2$ where $A_{\mathrm{C}_1,\mathrm{C}_2} = \mathrm{C}_1^{-\frac{1}{2}}(\mathrm{C}_1^{\frac{1}{2}}\mathrm{C}_2\mathrm{C}_1^{\frac{1}{2}})^{\frac{1}{2}}\mathrm{C}_1^{-\frac{1}{2}}$. When separating the problem in the BSP construction, the points are partitioned along parallel hyperplanes. Since $T_{\mathcal{N}_1,\mathcal{N}_2}$ is affine, and $A_{\mathrm{C}_1,\mathrm{C}_2}$ symmetric positive definite, a half space orthogonal to a direction $\boldsymbol{\omega}_\ell$ is mapped to the corresponding parallel half space in $\mathcal{N}_2$ only if $\boldsymbol{\omega}_\ell$ is an eigenvector of $A_{\mathrm{C}_1,\mathrm{C}_2}$. Hence, we choose these eigenvectors since they linearly separate the optimal transport in the Gaussian case. We define the *Gaussian slicing strategy* between two point sets X, Y as randomly picking an eigenvector of $A_{\mathrm{C}(X),\mathrm{C}(Y)}$, where $\mathrm{C}(X)$ is the covariance matrix of X. We found that sampling eigenvectors uniformly resulted in a better exploration of couplings rather than importance sampling them based on their magnitude, since the slicing policy should be symmetric in X and Y while the eigenvalues of $A_{\mathrm{C}(X),\mathrm{C}(Y)}$ are the inverse of those of $A_{\mathrm{C}(Y),\mathrm{C}(X)}$. Note that a classical heuristic to build the BSP of a single point set consists in slicing along the largest eigenvector of its covariance matrix [Gottschalk et al. 1996] – our method can be seen as a generalization of this process when two sets of points must be considered at the same time.

A drawback of this approach is that it strongly depends on dimensionality, since computing the eigenvectors of the $d \times d$ transport matrix has a $O(d^3)$ complexity. Hence, this heuristic is mainly usable in the low-dimensional setting. Since the transport map between Gaussians first centers the distribution before applying the linear map $A_{\mathrm{C}(X),\mathrm{C}(Y)}$, the pivot point must be as close as possible to the mean of the distributions. In this context, setting $\rho = 0.5$ provided best results (see Sec. 6 and 7). In practice, since empirical covariance matrices are not relevant when estimated from too few samples, we only perform the Gaussian slicing when $n \geq 50$ in any given tree node, and $d \leq 20$. We also deduce the covariance matrix of the right child from that of the left child and its parent to halve covariance matrix computational time (see Appendix A).

3.3 Bijection merging

While the assignments produced by Alg. 1 have higher quality with respect to Eq. (1) than the ones obtained by sorting along 1D slices, with the same time complexity, they are still far from optimal. Our second key idea is to exploit their sparsity to further improve their quality by combining multiple assignments into one.

Assignment swapping. Here we consider that we have a current bijection T and another T' that we want to merge together. For a given point x_i assigned to $T(x_i)$, re-assigning it to a new proposal $T'(x_i)$ would break bijectivity, unless the point x_j such that $T(x_j) = T'(x_i)$ is assigned to $T(x_i)$. Hence, we perform the swap only if it is a global improvement of the cost, i.e., whenever $c(x_i, T'(x_i)) + c(x_j, T(x_i)) < c(x_i, T(x_i)) + c(x_j, T(x_j))$. The assignment of x_j to $T(x_i)$ becomes a new proposal that might be present neither in T nor T', which breaks linear separation artifacts, present in BSP matchings.



Monotone merging. An assignment swap only considers pairs of points x_i and x_j, but more global swaps may be interesting to explore. By clustering the (unoriented) bipartite graph whose nodes are $\{x_i\}_{i=1..n}$ and $\{y_i\}_{i=1..n}$ and arcs connect each x_i to both $T(x_i)$ and $T'(x_i)$ into connected components, disjoint sets of nodes are obtained. This clustering is performed in linear time using a depth-first traversal. For each component, we select the assignment of lower cost, either T or T' entirely for all x_i of that connected component.

Combining both. While monotone merging operates at larger scales and can be executed in parallel, assignment swapping is better at breaking slicing artifacts observed in individual BSP couplings. In practice, for each connected component of the monotone merge, we either start from T and try to improve this map by assignment swapping each node with T' sequentially, or start with T' and try to improve it with T, depending on which map offers the lowest total cost. Furthermore, since the disjoint components are independent, the entire process can be performed in parallel on each set. The resulting map is guaranteed to be no worse than the better of T and T' in transportation cost. Algorithm 2 describes this strategy and is illustrated in Fig. 4.

Accounting for the construction of k BSPs of n points in dimension d and the merging of these BSPs, the complexity of our entire algorithm is $O(kn \log(n)d)$, with an additional d^3 factor when using Gaussian slicing (Sec. 3.2).

This construction is independent of the considered transport cost. While we focus on the $\mathcal{W}_2$ case ($c(x, y) = \|x - y\|^2$), we show in Section B of the supplementary materials that our approach is efficient for more general L_p metrics.

Algorithm 2: Bijection merging

```
Data: X = [x_1 ... x_n], Y = [y_1 ... y_n], T, T' ∈ X → Y
// Find disjoint cycles in the conflict graph
CC ← ConnectedComponents(T ∪ T') ;
T* ← T;
for in parallel C ∈ CC do
    c_T ← Σ_{i∈C} c(x_i, T(x_i)) ;          // cost using T
    c_T' ← Σ_{i∈C} c(x_i, T'(x_i)) ;        // cost using T'
    if c_T' < c_T then  // use the best assignment on C
        for i < N ∈ C do
            T*(x_i) ← T'(x_i);
            T'(x_i) ← T(x_i) ;      // for the assignment
             swaps
    for i ∈ C do          // greedy local improvement
        AssignmentSwap(T*, T', i);
return T*;
```

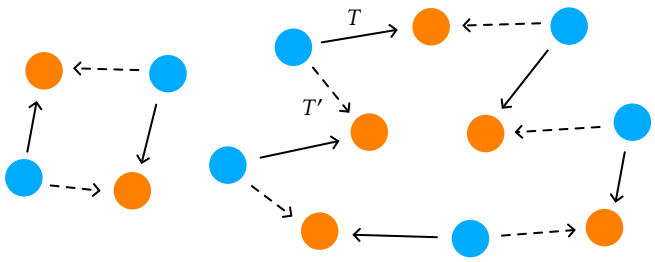


Fig. 4. When merging two bijections T and T', identifying connected components in the (unsigned) graph induced by $T \cup T'$ decomposes the merging on disjoint sets. On each set, in parallel, we first identify which bijection gives the lower cost, T or T', then perform assignation swapping.

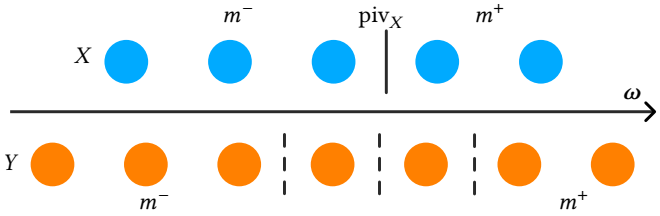


Fig. 5. In the partial setting, once a pivot for X is chosen along a slice $\boldsymbol{\omega}$, sending m_1 atoms to the left and m_2 atoms to the right, the pivot of Y must be chosen so that at least m_1 points are sent left and m_2 to the right (here piv_Y can be chosen as any of the three dashed line).

4 BSP Matching: the partial case

We now consider the case where the target point set $Y = \{y_i\}_{i=1..n}$ has more points than the source point set $X = \{x_i\}_{i=1..m}$, i.e. $n > m$, also known as *partial transport*. This results in a search over injective, rather than bijective maps.

4.1 BSP construction

When choosing a pivot in the balanced case, one would make sure that the same number of points was present on any given side for both X and Y, to enforce bijectivity at the leaf level. In the partial setting, after sending m^- points of X to the left child of the current BSP node, and m^+ points to the right (with $m^- + m^+ = m$), we need to split Y such that the left child has *at least* m^- points and the right one *at least* m^+ points. Since $m^- + m^+ < n$, the location of this split can be chosen anywhere in the middle part (see Fig. 6).

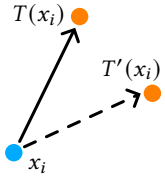
In practice, since partial OT is not translation invariant, among possible choices, we set the pivot of Y to be as close as possible, for the cost c, to the pivot of X. This ensures that transportation cost is overall reduced.

Since the partial sorting routine has a linear complexity, we still maintain an $O(n \log(m))$ complexity.

4.2 Merging partial matchings

We extend our merging procedure to the partial, injective, case.

Assignment swapping. Given two injective maps T and T' and a point x_i, either the point to which x_i is mapped by T' already receives mass from T, then it boils down to the bijective case, or $T'(x_i)$ does not receive mass by T and so x_i can just accept it directly if it reduces the cost, as displayed in the inset.



Algorithm 3: `PartialBSPMatching`

Data: $X = [x_1 \dots x_m], Y = [y_1 \dots y_n], m < n,$
$\text{start}_X, \text{end}_X, \text{start}_Y, \text{end}_Y, T$
if $end_X - start_X = 1$ **then**
  `// Assign atom to closest point of` Y
  $T(x_{\text{start}_X}) \leftarrow \underset{i \in [\text{start}_Y, \text{end}_Y[}{\arg\min} \; c(x_{\text{start}_X}, y_i)$;
  return
$\boldsymbol{\omega}_\ell \leftarrow$ `GenerateSliceDirection()`;
$\text{piv}_X \leftarrow$ `PickPivot(`$start_X$,end_X`)`;
`// Minimum number of points to send on both sides`
$m^- \leftarrow \text{piv}_X - \text{start}_X$;
$m^+ \leftarrow \text{end}_X - \text{piv}_X$;
`QuickSelect` $(X, \boldsymbol{\omega}_\ell, \text{start}_X, \text{piv}_X, \text{end}_X)$;
`QuickSelect` $(Y, \boldsymbol{\omega}_\ell, \text{start}_Y, \text{start}_Y + m^-, \text{end}_Y)$;
`QuickSelect` $(Y, \boldsymbol{\omega}_\ell, \text{start}_Y + m^-, \text{end}_Y - m^+, \text{end}_Y)$;
`// Set` piv_Y `as closest point to` piv_X `while respecting the mass constraint`
$\text{piv}_Y \leftarrow \underset{i \in [\text{start}_Y + m^-, \text{end}_Y - m^+[}{\arg\min} \; c(x_{\text{piv}_X}, y_i)$;
`PartialBSPMatching`$(X, Y, start_X, piv_X, start_Y, piv_Y, T)$;
`PartialBSPMatching`$(X, Y, piv_X, end_X, piv_Y, end_Y, T)$;

Algorithm 4: Injective Assignment swapping

Data: $X = [x_1 \dots x_m], Y = [y_1 \dots y_n], T, T', x_i$
$y_i \leftarrow T(x_i)$;
$y'_i \leftarrow T'(x_i)$;
if $T^{-1}(y'_i) \neq \emptyset$ **then**
  $x'_i \leftarrow T^{-1}(y'_i)$;
  if $c(x_i, y'_i) + c(x'_i, y_i) < c(x_i, y_i) + c(x'_i, y'_i)$ **then**
    $T(x_i) \leftarrow y'_i$;
    $T(x'_i) \leftarrow y_i$;
else if $c(x_i, y'_i) < c(x_i, y_i)$ **then**
  $T(x_i) \leftarrow y'_i$;

Monotone merging. Alg. 2 still holds, using this modified assignment swap.

5 BSP Matching: non-uniform balanced distributions

BSP Matchings can also be extended to the case two non-uniform measures $\mu = \sum_{i=1}^m \mu_i \delta_{x_i}$ and $\nu = \sum_{i=1}^n \nu_i \delta_{y_i}$ with the same total mass.

5.1 BSP construction

While balanced ($m = n$) uniform discrete distributions result in bijective assignments, the non-uniform case results in mass splitting, i.e., a point may be assigned to multiple points in the target point cloud. We here formulate the problem in terms of measures, matching a measure μ supported on point set X with a measure ν supported

on point set Y, by finding a coupling π rather than a map T. While the principle remains the same, considering previous cases where $\mu_i = 1$ and $\nu_i = 1$ for all i, the splitting of mass makes the search for a pivot more difficult. In the uniform case, since each term of the measure carries the same weight, splitting the array can be done based on array indices, and results in an integer number of samples on each side, obtained via `QuickSelect`. In the non-uniform case, we adapt the splitting procedure so that the subtrees of the BSP for μ and ν have matching masses. While it is generally not possible to group the terms of a measure so that they match any target amount of mass, a term can always be split in two to respect this constraint. At any level of recursion, for the measure μ, we simply partition the set of points of μ based on the dot product $\langle x_{\text{pivot}}, \boldsymbol{\omega}\rangle$ (`PartitionCDFByDot`, Alg. 5), and record their corresponding mass μ^- on the left side of the pivot. For the measure ν, we extend the `QuickSelect` to a `QuickCDF` algorithm that, instead of partitioning based on the number of elements, partitions the atoms $\{(y_i, \nu_i)\}_i$ to match the amount of mass on each side – see Alg. 6. This algorithm produces a pivot atom (y_p, ν_p) and the total mass ν^- of the atoms before the pivot. The atoms are partitioned so that $\nu^- < \mu^-$, $\nu^- + \nu_p \geq \mu^-$ and for any y_i before the pivot and y_j after the pivot, $\langle y_i, \boldsymbol{\omega}\rangle \leq \langle y_j, \boldsymbol{\omega}\rangle$. Intuitively, as illustrated in Fig. 6, y_p is the atom responsible for making the amount of mass of ν, along the slice, go from below μ^- to above. To split ν and respect the prescribed masses, the pivot atom has to be split into two atoms $(y_p, \nu_{p,-})$, and $(y_p, \nu_{p,+})$, one for each side of the BSP with $\nu_{p,+} + \nu_{p,-} = \nu_p$ and $\nu^- + \nu_{p,-} = \mu^-$. In general, at each level of the BSP, an atom of ν is split. This process is applied recursively, until one of the measures is made of a single atom. In that case, this atom is assigned to all the atoms remaining for the other measure. The overall algorithm for non-uniform distributions is summarized in Alg. 7.

We prove in Appendix B that a coupling π produced by Alg. 7 is an extremal point of the coupling simplex $\Pi(\mu, \nu)$, i.e., π is an acyclic graph. As such, the number of edges in π is bounded by $n + m - 1$ similarly to optimal couplings.

Invariance by similarity transform . Denoting $\mu_{s,t}$ the measure $\mu_{s,t} = \sum_i \mu_i \delta_{sx_i+t}$, with $s > 0, t \in \mathbb{R}^n$ (similarly for ν), then:

$$\pi\left(\mu_{s,t}, \nu_{s',t'}\right) = \pi\left(\mu, \nu\right),$$

where π denote the coupling resulting from `GeneralBSPMatching` or `BSPMatching`. Indeed, since the BSP only relies on the ordering of dot products, applying a similarity transform does not change their order and hence the output is the same. Since, we also have that $\pi(\mu, \mu \circ \sigma) = \sigma$, where $\mu \circ \sigma$ is a permutation of μ by σ, π is optimal (in the sense of optimal transport) when computing transport maps between measures only differing by a similarity transform.

5.2 BSP merging

While we can merge couplings produced by Alg. 7 in a way similar to the previous settings, it is algorithmically much more involved and the complexity of the merge depends on the degree of each vertex, which leads to significantly worse run times. These algorithms are implemented in the code we provide in supplementary materials.

Algorithm 5: `PartitionCDFByDot`

Data: $\mu = \sum_{i=1}^m \mu_i \delta_{x_i}, \boldsymbol{\omega} \in \mathbb{R}^d, I \subset [\![1, m]\!]$, start, piv, end
$i^- \leftarrow$ start; $i^+ \leftarrow$ end; $\mu^- \leftarrow 0$; $t \leftarrow \langle x_{I[\text{piv}]}, \boldsymbol{\omega}\rangle$;
while $i^- \leq i^+$ **do**
  while $i^- <$ end **and** $\langle x_{I[i^-]}, \boldsymbol{\omega}\rangle < t$ **do**
    `// Accumulate the mass before the pivot`
    $\mu^- \leftarrow \mu^- + \mu_{I[i^-]}$;
    $i^- \leftarrow i^- + 1$;
  while $i^+ \geq$ start **and** $\langle x_{I[i^-]}, \boldsymbol{\omega}\rangle \geq t$ **do**
    $i^+ \leftarrow i^+ - 1$;
  if $i^+ > i^-$ **then**
    swap$(I[i^-], I[i^+])$;
`// Return the new index of the pivot and its CDF`
return i^-, μ^-;

Algorithm 6: `QuickCDF`

Data: $\nu = \sum_{i=1}^n \nu_i \delta_{y_i}, \boldsymbol{\omega} \in \mathbb{R}^d, I \subset [\![1, n]\!]$, start, end, μ^-, ν^-
if end − start = 1 **then**
  `// start is the pivot index`
  `// ν⁻ is the total mass before the pivot`
  return start,ν^- ;
piv ←`UniformlyDraw(`$[\![start, end[\![$`)` ;
$i^-, \mathrm{d}\nu^- \leftarrow$`PartitionCDFByDot(`$\nu, \boldsymbol{\omega}, I, start, piv, end$`)`;
if $\nu^- + \mathrm{d}\nu^- \geq \mu^-$ **then**
  `// The desired split is before the pivot`
  return `QuickCDF(`$\nu, \boldsymbol{\omega}, I, start, i^-, \mu^-, \nu^-$`)`;
else
  `// The desired split is after the pivot`
  return `QuickCDF(`$\nu, \boldsymbol{\omega}, I, i^-, end, \mu^-, \nu^- + \mathrm{d}\nu^-$`)`;

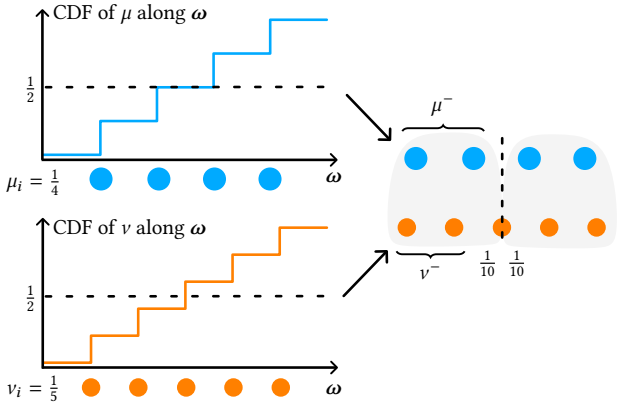


Fig. 6. Unlike in the bijective case, when the atoms of the two measures do not have the same masses, there is generally no offset along the slice separating the measure atoms into subsets of matching masses. To ensure that the mass constraint is preserved, an atom of ν is split. Here when separating μ leaving two atoms on each side, we get $\mu^- = \frac{1}{2}$. Splitting ν using Alg. 6 (`QuickCDF`) yields $\nu^- = \frac{2}{5}$ and the third atom of ν is split with masses $\nu_{p,-} = \nu_{p,+} = \frac{1}{10}$.

Algorithm 7: `GeneralBSPMatching`

Data: $\mu = \sum_{i=1}^{m} \mu_i \delta_{x_i}$, $\nu = \sum_{i=1}^{n} \nu_i \delta_{y_i}$, $I_\mu \subset [\![1, m]\!]$, $I_\nu \subset [\![1, n]\!]$, start_μ, end_μ, start_ν, end_ν

if $\text{end}_\mu - \text{start}_\mu = 1$ **then**
  for $j \in [\![\text{start}_\nu, \text{end}_\nu[\![$ **do**
    $\pi_{I_\mu[\text{start}_\mu], I_\nu[j]} \leftarrow \nu_{I_\nu[j]}$;
  return;

if $\text{end}_\nu - \text{start}_\nu = 1$ **then**
  for $i \in [\![\text{start}_\mu, \text{end}_\mu[\![$ **do**
    $\pi_{I_\mu[i], I_\nu[\text{start}_\nu]} \leftarrow \mu_{I_\mu[i]}$;
  return;

$\boldsymbol{\omega} \leftarrow$ `GenerateSliceDirection()`;

$p \leftarrow$ `PickPivot(`$start_\mu$,end_μ`)`;

`// Split` μ

$\text{piv}_\mu, \mu^- \leftarrow$ `PartitionCDFByDot(`$\mu, \boldsymbol{\omega}, I_\mu, start_\mu, p, end_\mu$`)`;

`// Split` ν `and find split atom`

$\text{piv}_\nu, \nu^- \leftarrow$ `QuickCDF(`$\nu, \boldsymbol{\omega}, I_\nu, start_\nu, end_\nu, \mu^-, 0$`)`;

$p \leftarrow I_\nu[\text{piv}_\nu] \quad \nu_{p,-} \leftarrow \mu^- - \nu^-; \quad \nu_{p,+} \leftarrow \nu_p - \nu_{p,-}$;

`// Execute left branch`

$\nu_p \leftarrow \nu_{p,-}$;

`GeneralBSPMatching(`$\mu, \nu, I_\mu, I_\nu, start_\mu, piv_\mu, start_\nu, piv_\nu + 1$`)`;

`// Execute right branch`

$I_\nu[\text{piv}_\nu] \leftarrow p; \quad \nu_p \leftarrow \nu_{p,+}$;

`GeneralBSPMatching(`$\mu, \nu, I_\mu, I_\nu, piv_\mu, end_\mu, piv_\nu, end_\nu$`)`;

$\nu_p \leftarrow \nu_{p,-} + \nu_{p,+}$;

6 Numerical study

All experiments were performed on an Intel(R) Xeon(R) W-2245 3.90Hz, with 8 cores, and a A5000 GPU. We provide our code in supplementary materials.

Number of trees. We evaluate the quality of the transport cost and speed of our algorithm with respect to the number of points, when matching two point clouds in 2D and 3D with the same number of points. We compute the mapping obtained from our bijective algorithm and compare it to a ground-truth given by the network-simplex implementation of Bonneel et al. [2011]. We use the `Monge` to `Kantorovitch` stippled images (see Fig. 1), and a single disk to two disks densities in 2D, an `armadillo` to a ball (3D, volumetric), and a `dragon` to a `bunny` (3D, surfacic) (see Fig. 7-d). In Fig. 7-a and 7-b, we plot the relative error in transport distance as a function of the number of trees, k, that are merged (10k random subsamples of each point cloud, averaged on 100 realizations). We also assess our random slicing (Fig. 7-a) and Gaussian slicing (Fig. 7-b) strategies. Both graphs show a fast decay in the first iterations then a slower convergence to a plateau. While this tends to indicate that we do not recover the ground truth optimal transport matching, the relative error remains small. The error quickly drops below 1% with Gaussian slicing, while it remains between 1 and 3% for an equivalent number of random slices. Note that it performs particularly well on the sphere to double sphere because the Gaussian slicing can identify the exact cut to split the sphere in two. In Figures 7-a and 7-b, we also compare with a single, axis-aligned, kd-tree construction as suggested by Nurbekyan et al. [2020]. Best results are obtained using Gaussian slicing as seen in Fig. 7-b. In Section C of the supplementary materials, we detail additional results where our merging strategy is combined with tree constructions with orthogonal directions.

In the remaining experiments, unless specified otherwise, we have fixed $k = 64$ and the Gaussian slicing strategy is used. In Fig. 7-c, we evaluate the speed for $k = 16$ as a function of the number of samples (random samples on the dragon and the bunny models). We experimentally confirm the loglinear behavior of our variations of the QuickSort algorithm for both slicing strategies.

Number of connected components. In Section A of the supplementary materials, we provide statistics of connected component sizes across merging iterations.

Synthetic examples. Methods that compute sliced transport often only provide distances but no actual coupling. In the bijective case, to compare to these approaches, we compare our coupling to a gradient flow which advects particles X towards Y by minimizing a (sliced) transportation cost via gradient descent, resulting in the advected point set X^* We evaluate the quality of the transport by computing both the quadratic distance between the initial point set X and its advection X^*, $||X^* - X||^2$ [Rabin et al. 2011], and the transport distance between the advected point set X^* and the target Y, $W_2(X^*, Y)$ [Tran et al. 2025]. While $||X^* - X||^2$ should ideally approximate $W_2(X, Y)$, $W_2(X^*, Y)$ should be as close as possible to 0. Table 1 shows comparisons on several synthetic datasets used in this literature: `Swiss-Roll` (2D), `25-G-2D` (mixture of 25 random 2d Gaussians), `G-20D` (random Gaussian in 20D). Our method uses $k = 64$ trees. In all cases, our approach better approximates optimal transport and perfectly match the target measure.

7 Applications

7.1 Shape interpolation and barycenters

Interpolation. We assess shape interpolation in the bijective case. We interpolate between two 3D shapes randomly sampled within their volume by computing a bijective map, and partly advecting samples in a straight line, producing a displacement interpolation [Bonneel et al. 2011; McCann 1997]. In Fig. 8 we show that we obtain an interpolation between two shapes comparable to the result of the multiscale Sinkhorn implementation in Geomloss [Feydy et al. 2019] but significantly faster, without requiring a gradient flow and we provide an explicit bijective matching between the shapes.

Barycenters. The Wasserstein barycenter problem allows to interpolate between multiple shapes, but it involves minimizing a weighted sum of optimal transport distances. Instead, we compute the *linearized* barycenter of S point sets Y^s, which approximates the Wasserstein barycenter, by computing S transport maps T^s between a pivot point set X (here, we sample a random uniform distribution) and each Y^s, and transform X by the weighted sum

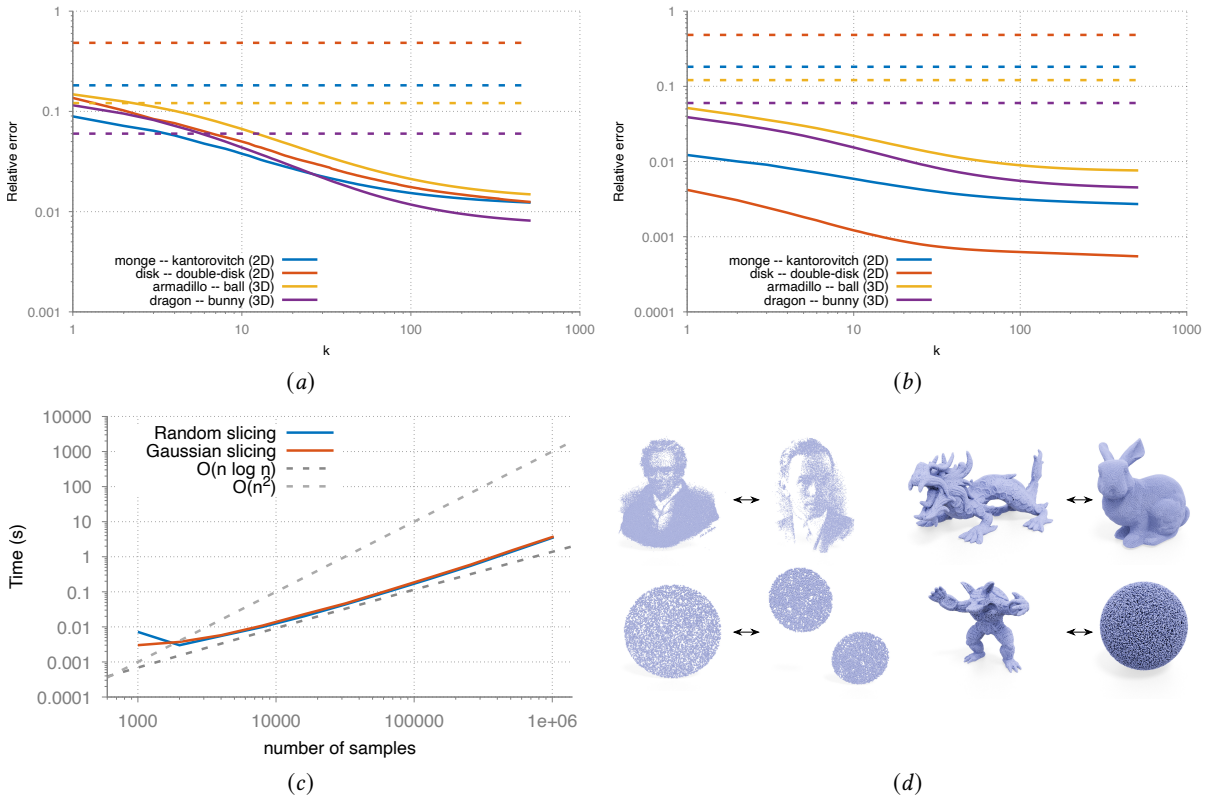


Fig. 7. For 2D and 3D matching problems (d), we compute the relative error of our BSPOT cost to W_2 as a function of the number of merged trees (k) using a random slicing (a) and with a Gaussian slicing (b). Dashed lines correspond to the relative error of a single kd-tree (axis aligned) as suggested by Nurbekyan et al. [2020] In (c) we compare timings for such sampling strategies for increasing number of random samples on the bunny to the dragon model ($k = 16$). *Armadillo, Bunny and Dragon from the Stanford Computer Graphics Laboratory 3D scanning repository.*

$T(X) = \sum_s \lambda_s T^s(X)$, given interpolation weights $\{\lambda_s\}$. We illustrate these linearized barycenters in the bijective setting in Fig. 9.

7.2 Intrinsic blue-noise manifold sampling

To obtain a blue-noise sampling of a curved surface $\mathcal{M}$, one can optimize a point set such that its optimal transport cost to a given measure (uniform or any probability density function) is minimal [Genest et al. 2024; Qin et al. 2017]. Optimizing point location directly on the surface rather than in ambient space results in better samplings in difficult cases, such as thin features, and near object boundaries [Genest et al. 2024; Zong et al. 2023]. Solving optimal transport problems on manifolds is challenging [Solomon et al. 2015, 2014], and in our settings, BSP matching is defined in Euclidean space. We hence rely on spectral embeddings. Let $\{(\lambda_i, u_i)\}$ be the (sorted) eigenpairs of the Laplace-Beltrami operator of $\mathcal{M}$. Using the first Λ eigenpairs, the embedding S_Λ from $\mathcal{M}$ to $\mathbb{R}^\Lambda$ is defined as:

$$x \mapsto \left(\frac{u_1(x)}{\sqrt{\lambda_1}}, \ldots, \frac{u_\Lambda(x)}{\sqrt{\lambda_\Lambda}} \right),$$

and is called *global point signature* [Rustamov et al. 2007]. Hence, we approximate the intrinsic matching between two point sets on $\mathcal{M}$ by an Euclidean mapping in $\mathbb{R}^\Lambda$ on the embedded point sets, where Euclidean distances in $\mathbb{R}^\Lambda$ approximate geodesic distances on $\mathcal{M}$. In practice, we use $\Lambda = 64$ eigen vectors. In our method, we further benefit from the fully bijective transport map between uniform measures, even when sampling non-uniform densities.

To compute blue-noise samples $X = \{x_i\}_{i=1..n}$ following a density ν on $\mathcal{M}$, we draw p sets $\{Y^j = \{y_i^j\}_{i=1..n}\}_{j=1..p}$ of n random samples from ν on $\mathcal{M}$, and compute the embedding of each point by S_Λ, denoted $\tilde{y}_i^j = S_\Lambda(y_i^j)$. We denote $Y = \bigcup_{j=1..p} Y_j$. We also compute the embedding of points x_i denoted $\tilde{x}_i = S_\Lambda(x_i)$ and set $\tilde{X} = \{\tilde{x}_i\}$ and $\tilde{Y}^j = \{\tilde{y}_i^j\}$. We then proceed iteratively to refine an estimate $X^{(\ell)}$ of blue noise samples, which we initialize by setting $X_i^{(1)} = Y^1$, and denote the embedded point set $\tilde{X}^{(\ell)}$. At each step ℓ, we compute the BSP-OT matching $T_j^{(\ell)}$ between $\tilde{X}^{(\ell)}$ and each $\tilde{Y}^j$. For each sample point of $\tilde{X}^{(\ell)}$, we compute the arithmetic mean $\chi(\tilde{x}_i^{(\ell)}) = \frac{1}{p} \sum_{j=1..p} T_j^{(\ell)}(\tilde{x}_i^{(\ell)})$ of its p assignments by all $T_j^{(\ell)}$ in $\mathbb{R}^\Lambda$. We then obtain the sample $x_i^{(\ell+1)}$ on $\mathcal{M}$ by reassigning $x_i^{(\ell)}$ to the sample y_i^j of Y on $\mathcal{M}$ whose embedding $\tilde{y}_i^j$ is closest to $\chi(\tilde{x}_i^{(\ell)})$. This reassignment process allows the points $\tilde{X}^{(\ell)}$ to remain within the fixed set of points Y and thus to only compute the embedding S_Λ once and for all, without having to invert the spectral embedding

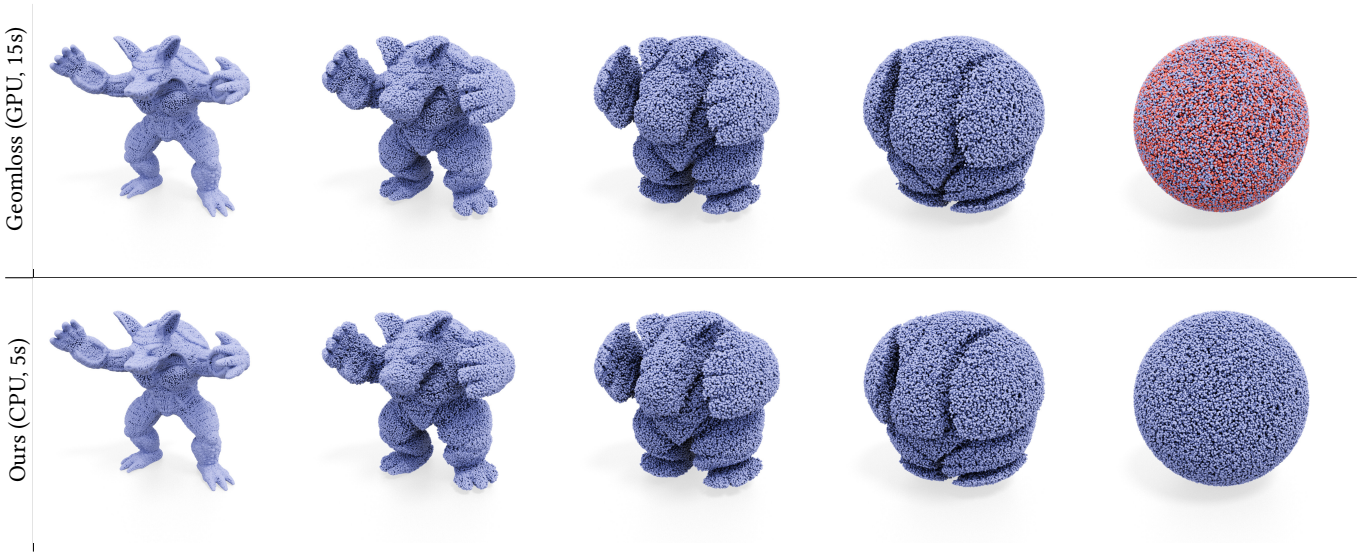


Fig. 8. Comparison with Geomloss: Result of the shape interpolation between two discrete uniform measures of 350k points from armadillo (left) to a ball (right). We used $k = 64$ plans for our method, with gaussian slicing, and the default parameters of Geomloss (10 iterations with $\sigma = 0.01$). In the rightmost column the points in red correspond to the target measure (the ball). It is not visible on the bottom row since, because of the bijectivity of our assignation, our points exactly match both measures, while flow based approaches hardly do. Computations performed on a A5000 GPU (for the record, on an older computer, with a GTX 1060, geomloss takes 1 minute for the same task, whereas our running time remains below 8s).

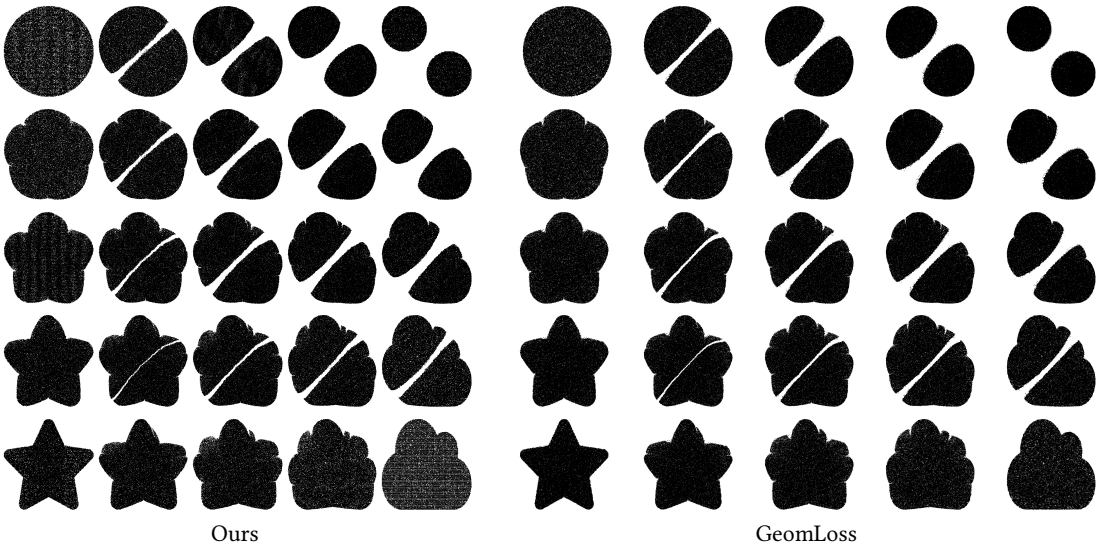


Fig. 9. We compare linearized barycenters computed using our matchings (left) with linearized barycenters computed with entropy-regularized transport on the GPU with GeomLoss (right), Our result is computed in 0.68s, with $k = 64$, and 1.98s with GeomLoss with 10 iterations and $\sigma = 0.01$, of 65k points.

back to $\mathcal{M}$. To stabilize and improve convergence, we initialize the BSP merging process to obtain $T_j^{(\ell+1)}$ by the resulting assignment obtained at the previous iteration $T_j^{(\ell)}$.

In Fig. 10, we show intrinsic sampling results on uniform and non-uniform densities. We compare our approach to the (non-intrinsic) restricted centroidal Voronoi tessellation sampling (CVT) of Liu et al. [2009] as a reference in the uniform case (from Levy [2025]), and the intrinsic sliced-based OT minimization NESOTS [Genest et al. 2024], on challenging, self-intersecting geometries. In the uniform case, the CVT fails because of their purely ambient approach. CVT in spectral coordinates (dimension 64) produces good results but is significantly slower (17x) than our approach, mostly due to an expensive re-projection to the mesh, a step that we completely avoid using bijectivity. Compared to NESOTS, we are 2.8 times faster with

Table 1. We use the same parameters as Tran et al. [2025], with $L = 25, k = 4$, $\delta = 10$ for the Db-TSW [2025], $L = 100$ slices for SW [2015] min-SWGG [2023],LCVSW [2023] with a learning rate of 5e-3 (5e-2 for Gaussian20D), with $n = 2000$ particles and 10000 iterations with the ADAM optimizer). For our method, we used Gaussian slicing in 2D and random slicing in 20D.

	method	relative error	$W_2^2(X^*, Y)$
Swiss-roll	SW	0.011	0.85
	min-SWGG	0.093	16.12
	Db-TSW	0.015	0.019
	LCVSW	0.004	0.008
	BSP-OT	0.002	0
25-G-2D	SW	0.025	7.01
	min-SWGG	0.121	1.17
	Db-TSW	0.005	0.64
	LCVSW	0.004	2.85
	BSP-OT	0.001	0
G-20D	SW	0.14	120
	min-SWGG	0.61	177.3
	Db-TSW	0.13	91.8
	LCVSW	0.16	111.2
	BSP-OT	0.027	0

a slightly better point distribution. For a non-uniform target where CVT cannot be used anymore, our results are similar to NESOTS but obtained 4 times faster.

7.3 Color transfer

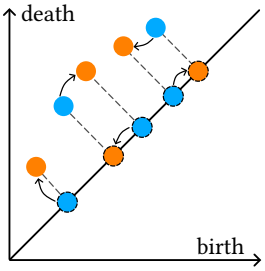
Optimal transport is routinely used to transfer colors from one image to another, by matching their color histograms. Sliced optimal transport advects each individual pixel to its target by gradient flow, but this requires repeated (sliced) optimal transport computations [Bonneel et al. 2015]. Alternatively, one may directly match color clusters, but at the cost of quantization artifacts [Morovic and Sun 2003]. Our approach allows to directly use the bijection between the source and target image to reorder pixels of the target image to compose an image as close as possible to the source image using only pixels from the target image, without requiring gradient flow. We use our bijective framework and assume that the number of pixels in both images match (up to stretching images, if necessary). Fig. 11 illustrates our results compared to sliced transport [Bonneel et al. 2015], or the bijection obtained by min-SWGG [Mahey et al. 2023]. Our color transfer exactly matches the target color distribution by construction which is not the case for sliced color transfer at the same computation cost. It also runs at a fraction of the cost of a fully-converged sliced color transfer.

7.4 Matching of persistence diagrams

In Topological Data Analysis, one can visualize properties of a scalar field using persistence diagrams. Given a function $f : X \to \mathbb{R}$, we compute the sublevel set g which assigns to a given threshold α the set of points of X where $f(x) \leq \alpha$, i.e., $g : \alpha \to \{x | f(x) \leq \alpha\}$. As α increases, the sublevel set grows and its topology evolves: new connected components may appear, existing ones merge, and higher-dimensional features such as loops or voids can emerge and disappear.

recording the birth and death of each feature.

A persistence diagram captures this evolving topology by Persistence diagrams are typically encoded as point sets on the 2D plane, each point encoding α_{death} as a function of α_{birth} for a connected component appearing and disappearing (see Fig. 12). In this context, optimal transport has been used to compare persistence diagrams [Kerber et al. 2017].



When comparing persistence diagrams with optimal transport, since each point is labeled, mass is not allowed to split, even if they have different number of points. Matching two persistence diagrams D_1, D_2 is then typically performed by comparing $\tilde{D}_1 = D_1 \cup \Delta(D_2)$ and $\tilde{D}_2 = D_2 \cup \Delta(D_1)$, where Δ is the projection on the diagonal. $\tilde{D}_1$ and $\tilde{D}_2$ thus have the same number of points, and can be compared in the *bijective* transport setting. Optimal transport is then computed using a modified quadratic cost $c(x, y) = \|x - y\|^2$ if x and y are not on the diagonal, and $c(x, y) = 0$ otherwise. We integrate this modified cost in our bijection merging procedure. Intuitively, a point matched to a diagonal point is considered as topological noise.

We show in Fig. 12 that the matching we provide offers a x52 speedup compared to a state-of-the-art auction solver, implemented in the hera library [2017], with less than 1% of relative error.

7.5 Point cloud registration

Bonneel and Coeurjolly [2019] use partial optimal transport to replace the nearest neighbor assignation in the Iterative Closest Point (ICP) algorithm to recover a similarity transformation to align two point clouds of different cardinality. Bai et al. [2023] follow this idea with another solution to the partial optimal transport problem in 1D. At each ICP iteration, Bonneel and Coeurjolly [2019] replaces nearest neighbor assignment by a gradient flow, while Bai et al. [2023] uses the assignment obtained by a projection on a single direction, and both then compute the transformation (rotation, translation and scaling) with Kabsch's formula. Both remain of quadratic complexity in the worst case since they solve the partial problem exactly in 1D. In Table 2 and Figure13 we show that the injective matchings produced by Alg. 3 in the partial transport settings, provides similar or better results than that of Bonneel and Coeurjolly [2019], and an order of magnitude faster. Registering point sets with the implementation of Bai et al. [2023] takes about 10 minutes for 10k samples on our machine, for comparable results, while our code runs within a second in that case.

7.6 Image stippling

We use our non-uniform transport matching for image stippling [Ahmed et al. 2022; de Goes et al. 2012; Do et al. 2025; Salaün et al. 2022]. We encode the image as a measure ν supported on a point set Y where each point $\{y_i\}$ is the center of a pixel, with mass inversely proportional to the pixel brightness. We also set a

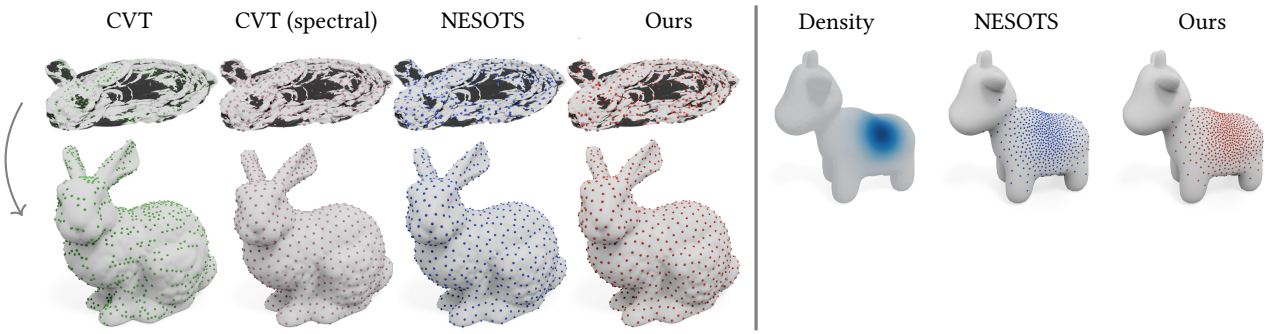


Fig. 10. Intrinsic blue-noise sampling of manifolds: to emphasize the intrinsic nature of our approach, we uniformly sample a *flat* mesh (top) with 1024 samples and highlight the sample distribution on isometrically *unflatten* one (middle). We compare the (non-intrinsic) CVT approach [Liu et al. 2009] (0.27s), a spectral (intrinsic) CVT approach (37.2s), the NESOTS sampling [Genest et al. 2024] (6s), our approach ($\Lambda = 64$, $p = 256$, 5 optimization steps for the barycenter, and 16 assignments merged per bijection, in 2.15 seconds). On a non-uniform density with 512 samples (bottom), NESOTS runs in 5.2s while our method runs in 1.3s. *Spot model from [Crane et al. 2013].*

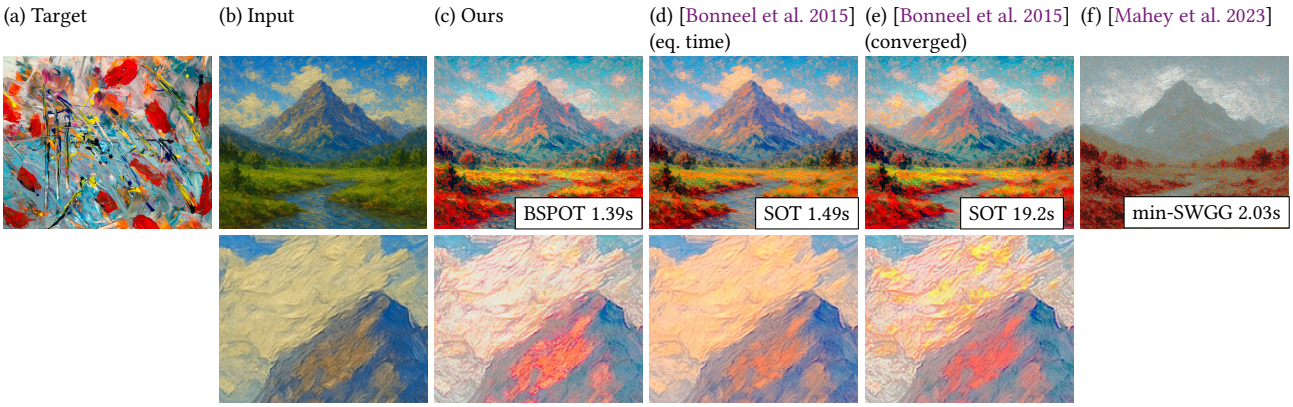


Fig. 11. We reorder the 850k pixels of the color image (a) to transfer its colors to the source image (b) using our bijection with $k = 16$ (c). This process hence exactly matches color distributions. Alternatively, we produce a gradient flow of the sliced optimal transport functional, and show results at equal time (d, after 8 steps of 8 slices in parallel) and at higher convergence (e, with 99 steps of 8 slices). Our approach exactly matches the target colors, with lower transport cost. This better preserves the texture of the input painting, as shown in the inset. We also show the result of the min-SWGG assignation (f). *The target image is from Steve Johnson (@steve_j).*

Table 2. We consider two sets X and Y, $X \subset Y$, with respectively n and m points. X is either a random sub-sampling of Y, or given by removing a part of Y. We consider $\tilde{X} = RXs + t$, where R using a random rotation of angle between $]-\frac{\pi}{3}, \frac{\pi}{3}[$, $s \in]0, 2[$ and $t \in [-\sigma, \sigma]^d$, with σ is the standard variation of X. We also concatenate to both point clouds 1k points of uniform noise. We perform the registration between $\tilde{X}$ and Y, with 100 iterations of ICP, with $k = 16$ for BSPOT and 40 slices for SPOT. We display the median error over 20 tries between the obtained registration and the template as $\frac{1}{n}||\tilde{X} - X||^2$.

name (n to m)	SPOT		BSP-OT	
	error	timing (s)	error	timing (s)
`Mumble` (80k to 90k)	0.009	70.3	0.002	8.81
`Castle` (150k to 200k)	0.003	175.2	0.004	17.37
`Cat-2D` (5k to 10k)	0.053	5	0.0002	0.77
`Car` (50k to 70k)	0.074	50	0.052	7.75

uniform measure μ supported on a uniformly random point set X. We set $\mu^{(0)} = \mu$ supported on $X^{(0)} = X$ and then proceed iteratively. At each iteration ℓ we compute k BSP matchings to match $\mu^{(\ell)}$ to the non-uniform ν using Alg. 7. The resulting k couplings $\pi^{k,(\ell)}$ allow to each determine an expected transport direction: $\mathbb{E}_{(X^{(\ell)},Y)\sim\pi^{k,(\ell)}}\left[Y - X^{(\ell)} | X^{(\ell)} = x_i\right] = \sum_j \pi^{k,(\ell)}_{i,j}(y_j - x_i)/\sum_j \pi^{k,(\ell)}_{i,j}$. We then obtain $X^{(\ell+1)}$ by advecting points of $X^{(\ell)}$ in the direction of the geometric median of these k direction vectors, computed with Wietzfield's algorithm, as in Genest et al. [2024].

In Figure 14, we compare our method with baseline algorithms: BNOT and GBN for high-quality but slower image stippling, and the recent rectified flow approach of Do et al. [2025] for fast OT-like approximation of the transport. Since our matching depends on the source image size, we present results for both 1024×1204 and 256×256 image sizes. Note that our results and timings have been obtained using the 256×256 image (see Fig.15 for a discussion). Our

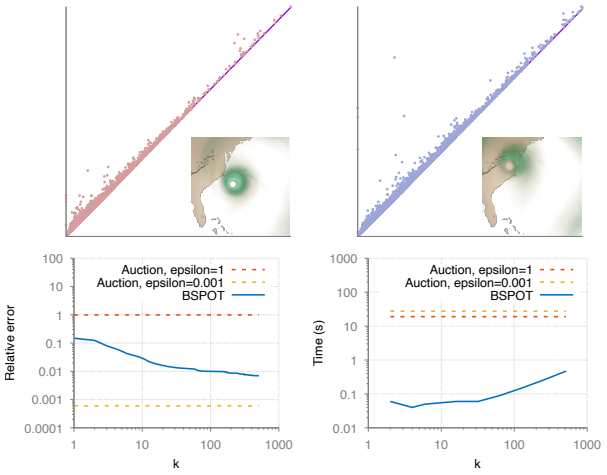


Fig. 12. On the `isabel` dataset of Vidal et al. [2019] (two recordings of the magnitude of the velocity of a storm), we compute the matching between two persistence diagrams (of 8406 and 8263 points) where each point is a saddle point in the levelset filtration (top), and compare our approach for various k with the $\epsilon-$auction algorithm, both in terms of relative error to the W_2 and timings.

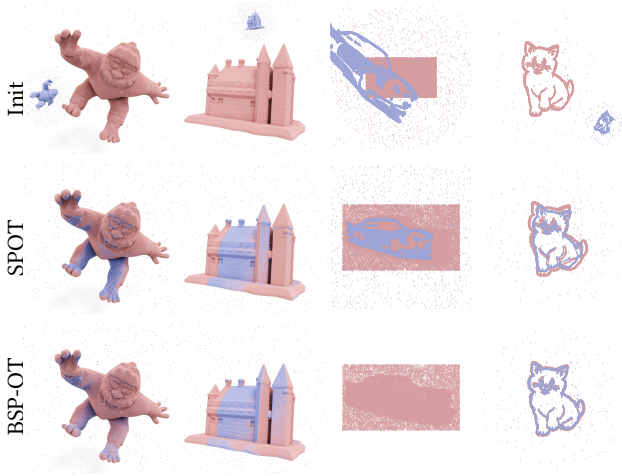


Fig. 13. Partial matching of two noisy point clouds, with translation, rotation and scaling. Using the assignment provided by the merging of 16 partial plans per iteration, we iteratively estimate the transformation to align the two point clouds (see Table. 2)

approach yields higher quality stippling compared to rectified flows, and comparable quality to BNOT or GBN but with a significant increase in speed.

8 Failure cases

As our merging procedure performs local optimizations over a combinatorial set (e.g. the set of bijections), the convergence toward the global minimum is unlikely. While the local minima we reach are typically of low relative error, it is not always the case. The worst case we identified is the computation of a bijection between two near identical point-sets, i.e., $\{X_i\}$ and a very small perturbation $\{X_i + \sigma\epsilon_i\}$, where $\epsilon_i \sim \mathcal{N}(0, I_d)$. As illustrated in Fig. 16, when σ is below a given threshold, the noise is small enough to be ignored by the BSP construction and it produces the optimal matching. However, when σ is just above this threshold, the points x_i and $x_i + \sigma\epsilon_i$ are sent on possibly very different branches of a BSP which produces a close but not exact matching that is then not corrected by merging since most proposed matchings are also incorrect. This results in an arbitrary high relative error since the ground truth has near zero cost. Finally, when σ keeps growing we recover the level of quality we usually obtain.

Note that this limitation is common to projection based assignation techniques as Min-SWGG [2023] and Kd-tree matchings [2020]. We however perform much better, the relative error on the example in Fig. 16 is twice higher for a kdtree matching and 42 times higher for the best 1D assignation of Min-SWGG.

9 Conclusion

We show that matching BSPs appears to be an efficient, simple, and scalable way to explore and parameterize the set of sparse couplings that are relevant for the optimal transport problem, that generalizes transport over slices. This combinatorial optimization for discrete measures yields many applications in computer graphics.

While Nurbekyan et al. [2020] analyze the properties of the continuous limit of the kd-tree subdivision for a single map and Bonnotte [2013] provides bounds for sliced transportation, we do not have such guarantees, and further analysis of our algorithm would be an interesting avenue for future research. The recursive nature of our BSPs makes them less GPU friendly, making integration into modern Machine Learning pipelines more difficult. The sequential merging procedure further complicates efficient GPU usage. Nonetheless, when slice directions are fixed, reformulating BSP Matchings similarly to Radix sort could offer a viable alternative. Injective matchings offer the advantage of extreme sparsity, enabling a merging procedure with linear time complexity. Yet, in non-uniform settings, varying point degrees and masses often result in quadratic complexity in the worst case. Designing a scalable merging procedure for general sparse couplings remains an open question. Finally, a major appeal of BSP matchings is the ease with which they can be generalized. Extending this approach to other settings of optimal transport like Gromov-Wasserstein, unbalanced optimal transport, or even non-Euclidean domains, is also an interesting axis of research.

Acknowledgments

This work is partially supported by the French National Research Agency within the StableProxies project (ANR-22-CE46-0006), by the ERC AdG 101054420 EYAWKAJKOS project, and donations from Adobe Inc. We thank Mattéo Clémot for his help on Topological Data Analysis experiments.

References

Abdalla GM Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian blue noise. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–15.

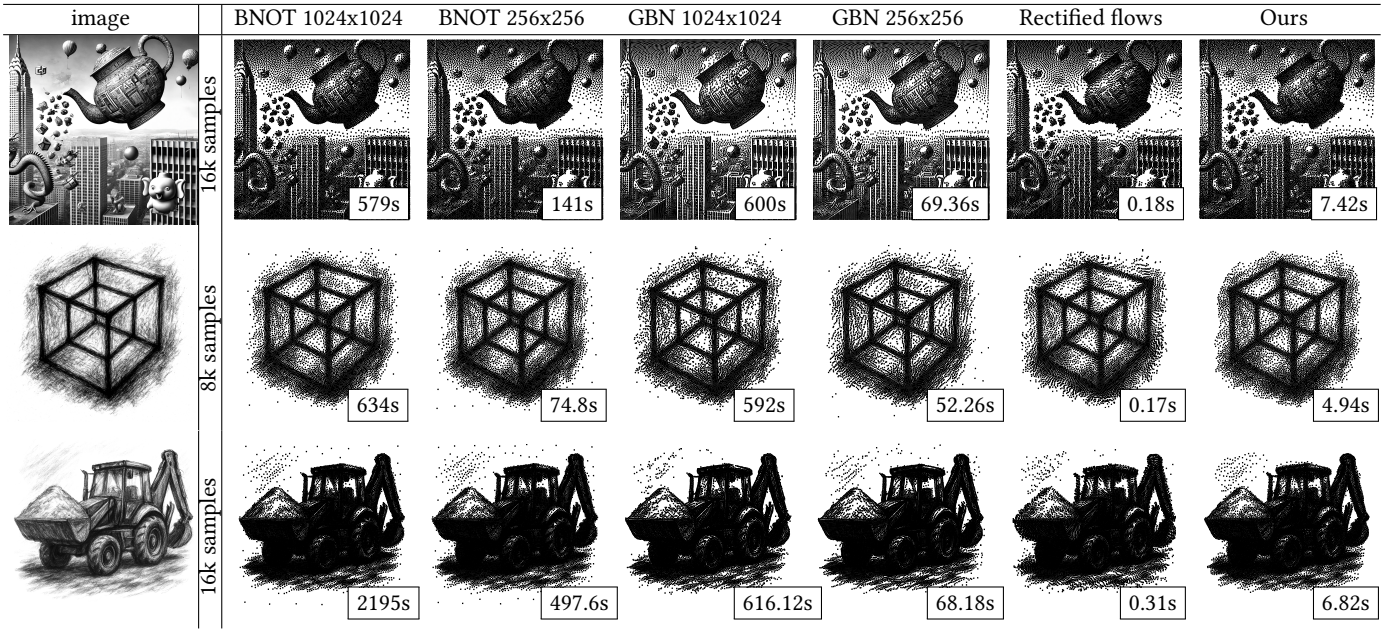


Fig. 14. Stippling results: on three different images, we compare BNOT [de Goes et al. 2012], GBN (10k iterations as suggested by Ahmed et al. [2022]) and Rectified Flows [Do et al. 2025], with our stippling approach (with $k = 32$ couplings per iteration, 100 iterations) on 256×256 images (see Fig.15 for discussion), either for 16k or 8k samples. Our approach provides high quality results (comparable to BNOT and GBN) but several orders of magnitude faster. Rectified flows produce extremely fast stippling but with noticeable sample alignments.

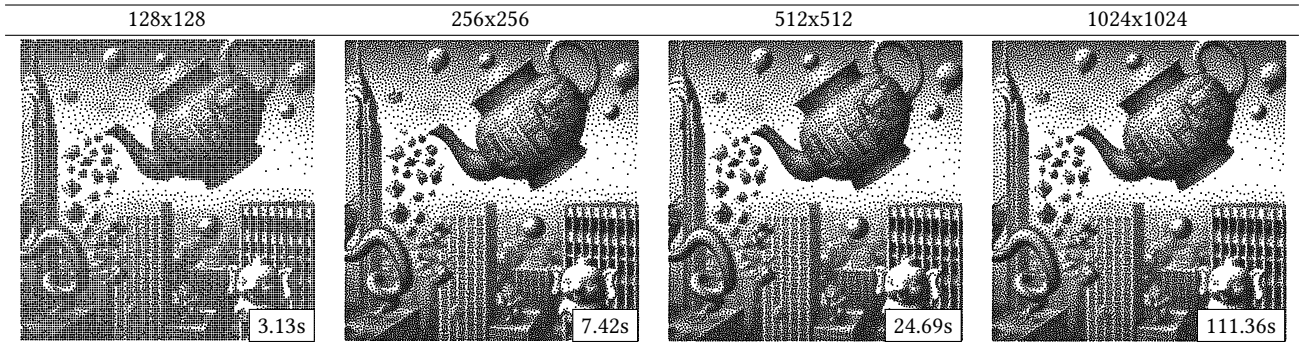


Fig. 15. Additional stippling results obtained by our approach when varying the target image resolution. In Fig. 14, we settled for 256×256 as a good compromise.

Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. 2019. Massively scalable Sinkhorn distances via the Nyström method. *Advances in neural information processing systems* 32 (2019).

Gennaro Auricchio, Federico Bassetti, Stefano Gualandi, and Marco Veneroni. 2018. Computing Kantorovich-Wasserstein Distances on d-dimensional histograms using $(d+1)$-partite graphs. *Advances in Neural Information Processing Systems* 31 (2018).

Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. 2020. Scalable nearest neighbor search for optimal transport. In *International Conference on machine learning*. PMLR, 497–506.

Yikun Bai, Bernhard Schmitzer, Matthew Thorpe, and Soheil Kolouri. 2023. Sliced optimal partial transport. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13681–13690.

Dimitri P Bertsekas. 1990. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces* 20, 4 (1990), 133–149.

Andrew J Blumberg, Mathieu Carriere, Michael A Mandell, Raul Rabadan, and Soledad Villar. 2020. MREC: a fast and versatile framework for aligning and matching point clouds with applications to single cell molecular data. *arXiv preprint arXiv:2001.01666* (2020).

Nicolas Bonneel and David Coeurjolly. 2019. Spot: sliced partial optimal transport. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.

Nicolas Bonneel and Julie Digne. 2023. A survey of optimal transport for computer graphics and computer vision. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, 439–460.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision* 51 (2015), 22–45.

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*. 1–12.

Nicolas Bonnotte. 2013. *Unidimensional and evolution methods for optimal transportation*. Ph. D. Dissertation. Université Paris Sud-Paris XI; Scuola normale superiore (Pise, Italie).

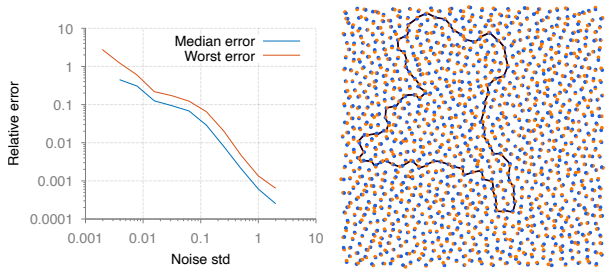


Fig. 16. (left) Median and worst relative error of our produced bijection between near identical point sets, i.e., between $\{X_i\}$ and $\{X_i+\sigma\epsilon_i\}$ where ϵ_i are standard Gaussian noise, with increasing σ. Using $k = 64$ and Gaussian slicing, results over 100 batches. Note that the relative error is exactly zero when σ is small enough, which results in the curves not displaying below $\sigma \lessapprox 0.002$. (right) Worst example with 267% of relative error. Assignation displayed in black, most of the points are correctly assigned except for a chain of points where each point is assigned to the optimal match of the previous one in the chain.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32 (2013), 1–10.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems* 26 (2013).

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.

Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. 2018. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3483–3491.

Khoa Do, David Coeurjolly, Pooran Memari, and Nicolas Bonneel. 2025. Linear-time Transport with Rectified Flows. *ACM Transactions on Graphics (TOG)* 44, 4 (2025).

Jean Feydy. 2020. Geometric data analysis, beyond convolutions. *Applied Mathematics* 3 (2020).

Jean Feydy, Pierre Roussillon, Alain Trouvé, and Pietro Gori. 2019. Fast and scalable optimal transport for brain tractograms. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part III 22*. Springer, 636–644.

Akshaykumar Gattani, Sharath Raghvendra, and Pouyan Shirzadian. 2023. A robust exact algorithm for the euclidean bipartite matching problem. *Advances in Neural Information Processing Systems* 36 (2023), 51706–51718.

Baptiste Genest, Nicolas Courty, and David Coeurjolly. 2024. Non-Euclidean Sliced Optimal Transport Sampling. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15020.

Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. 1996. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 171–180.

Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. 2021. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9412–9420.

Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. 2017. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)* 22 (2017), 1–20.

Abdelwahed Khamis, Russell Tsuchida, Mohamed Tarek, Vivien Rolland, and Lars Petersson. 2024. Scalable optimal transport methods in machine learning: A contemporary survey. *IEEE transactions on pattern analysis and machine intelligence* (2024).

Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. 2019. Tree-sliced variants of Wasserstein distances. *Advances in neural information processing systems* 32 (2019).

Bruno Levy. 2025. geogram. https://github.com/BrunoLevy/geogram

Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. 2009. On centroidal Voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)* 28, 4 (2009), 1–17.

Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. 2019. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on machine learning*. PMLR, 4104–4113.

Guillaume Mahey, Laetitia Chapel, Gilles Gasso, Clément Bonet, and Nicolas Courty. 2023. Fast optimal transport through sliced generalized Wasserstein geodesics. *Advances in Neural Information Processing Systems* 36 (2023), 35350–35385.

Robert J McCann. 1997. A convexity principle for interacting gases. *Advances in mathematics* 128, 1 (1997), 153–179.

Ján Morovic and Pei-Li Sun. 2003. Accurate 3d image colour histogram transformation. *Pattern Recognition Letters* 24, 11 (2003), 1725–1735.

Hédi Nabli. 2009. An overview on the simplex algorithm. *Appl. Math. Comput.* 210, 2 (2009), 479–489.

Elisa Negrini and Levon Nurbekyan. 2024. Applications of no-collision transportation maps in manifold learning. *SIAM Journal on Mathematics of Data Science* 6, 1 (2024), 97–126.

Khai Nguyen and Nhat Ho. 2023. Sliced wasserstein estimation with control variates. *arXiv preprint arXiv:2305.00402* (2023).

Levon Nurbekyan, Alexander Iannantuono, and Adam M Oberman. 2020. No-collision transportation maps. *Journal of Scientific Computing* 82, 2 (2020), 45.

Adam M Oberman and Yuanlong Ruan. 2015. An efficient linear programming method for optimal transportation. *arXiv preprint arXiv:1509.03668* (2015).

James B Orlin. 1997. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming* 78 (1997), 109–129.

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning* 11, 5-6 (2019), 355–607.

Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. 2017. Wasserstein blue noise sampling. *ACM Transactions on Graphics (TOG)* 36, 5 (2017), 1–13.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2011. Wasserstein barycenter and its application to texture mixing. In *International conference on scale space and variational methods in computer vision*. Springer, 435–446.

Raif M Rustamov et al. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on geometry processing*, Vol. 257. 225–233.

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable multi-class sampling via filtered sliced optimal transport. *arXiv preprint arXiv:2211.04314* (2022).

Filippo Santambrogio. 2015. *Optimal transport for applied mathematicians*. Vol. 87. Springer.

Bernhard Schmitzer. 2016. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision* 56 (2016), 238–259.

Bernhard Schmitzer. 2019. Stabilized sparse scaling algorithms for entropy regularized transport problems. *SIAM Journal on Scientific Computing* 41, 3 (2019), A1443–A1481.

Raghvendra Sharathkumar and Pankaj K Agarwal. 2012. A near-linear time ε-approximation algorithm for geometric bipartite matching. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 385–394.

Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–11.

Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. 2014. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (ToG)* 33, 4 (2014), 1–12.

Hoang V Tran, Khoi NM Nguyen, Trang Pham, Thanh T Chu, Tam Le, and Tan M Nguyen. 2025. Distance-based tree-sliced Wasserstein distance. *arXiv preprint arXiv:2503.11050* (2025).

Jules Vidal, Joseph Budin, and Julien Tierny. 2019. Progressive wasserstein barycenters of persistence diagrams. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 151–161.

Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. 2019. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3713–3722.

Makoto Yamada, Yuki Takezawa, Ryoma Sato, Han Bao, Zornitsa Kozareva, and Sujith Ravi. 2022. Approximating 1-wasserstein distance with trees. *arXiv preprint arXiv:2206.12116* (2022).

Chen Zong, Pengfei Wang, Dong-Ming Yan, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Qiang Hu. 2023. Parallel post-processing of restricted voronoi diagram on thin sheet models. *Computer-Aided Design* 159 (2023), 103511.

A Second order moment decomposition

From the mean $\mathbb{E}(\mu)$ and covariance $\mathrm{C}(\mu)$ matrices computed from n points, we aim at computing the same quantities for the disjoint sub-measures μ^-,μ^+ of n^- and n^+ points respectively (with $n = n^- + n^+$). We still have to compute $\mathbb{E}(\mu^-)$ and $\mathrm{C}(\mu^-)$ using the linear time standard algorithms, but one can compute $\mathbb{E}(\mu^+)$ and $\mathrm{C}(\mu^+)$ in

constant time by:

$$\begin{aligned}
\mathbb{E}(\mu^+) &= \frac{\mathbb{E}(\mu) - \alpha\mathbb{E}(\mu^-)}{\beta} \\
\Delta^- &= (\mathbb{E}(\mu) - \mathbb{E}(\mu^-)) \otimes (\mathbb{E}(\mu) - \mathbb{E}(\mu^-)) \\
\Delta^+ &= (\mathbb{E}(\mu) - \mathbb{E}(\mu^+)) \otimes (\mathbb{E}(\mu) - \mathbb{E}(\mu^+)) \\
\mathrm{C}(\mu^+) &= \frac{\mathrm{C}(\mu) - \alpha(\mathrm{C}(\mu^-) + \Delta^-)}{\beta} - \Delta^+
\end{aligned}$$

Where $\alpha = \frac{n^-}{n}$ and $\beta = 1 - \alpha$.

B Proof that Alg. 7 produces acyclic couplings

We prove by induction that the output of the general BSP matching does not contain cycles. The recursive construction of the BSP stops when one of the measures contains a single atom. In that situation, the produced assignation assigns this atom to all the atoms of the other measure. The corresponding graph is therefore a star and contains no cycle. In the general case of the recursion, by the induction hypothesis, the results of the recursive calls are acyclic couplings for both sides of the tree. By construction, the intersection of the atoms of μ on both sides is empty. For ν, a single pivot atom (y_p, ν_p) is used on both sides. The resulting coupling is therefore the union of two acyclic graphs with a single common vertex. There can be no cycle in such a graph, since there is no cycle on each side, and a cycle across both sides would require at least two common vertices to go from one side to the other. Alg. 7 therefore produces a coupling without cycles.