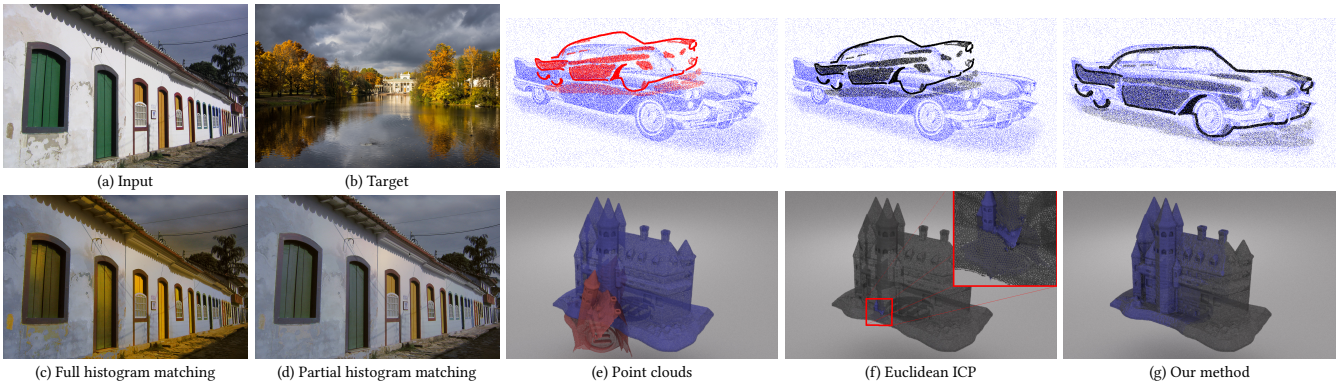
SPOT: Sliced Partial Optimal Transport

Nicolas Bonneel, David Coeurjolly

SPOT: Sliced Partial Optimal Transport

NICOLAS BONNEEL, Univ. Lyon, CNRS
DAVID COEURJOLLY, Univ. Lyon, CNRS



(a) Input (b) Target (c) Full histogram matching (d) Partial histogram matching (e) Point clouds (f) Euclidean ICP (g) Our method

Fig. 1. Our technique allows to partially match point sets within an optimal transport framework. **Left.** In color matching applications such as that of Pitié et al. [Pitié et al. 2005], matching all pixels of an input image (a) seen as 3D points in an RGB space to all pixels of a target image (b) can lead to erroneous transfers (c) due to mismatched content (here, trees of the target are not present in the input, and distort colors in the output). Instead, we match the input image (a) to a subset of an enlarged target image (b), thus effectively preventing spurious colors (d). **Right.** Given a source 2-d or 3-d point cloud (in red) and a target one (in blue) of different sizes (e), we match them with a similarity transform. While classical iterative closest point (ICP) fails (f), our approach, called *fast iterative sliced transport*, achieves robust registrations (g).

Optimal transport research has surged in the last decade with wide applications in computer graphics. In most cases, however, it has focused on the special case of the so-called "balanced" optimal transport problem, that is, the problem of optimally matching positive measures of equal total mass. While this approach is suitable for handling probability distributions as their total mass is always equal to one, it precludes other applications manipulating disparate measures. Our paper proposes a fast approach to the optimal transport of constant distributions supported on point sets of different cardinality via one-dimensional slices. This leads to one-dimensional partial assignment problems akin to alignment problems encountered in genomics or text comparison. Contrary to one-dimensional balanced optimal transport that leads to a trivial linear-time algorithm, such partial optimal transport, even in 1-d, has not seen any closed-form solution nor very efficient algorithms to date. We provide the first efficient 1-d partial optimal transport solver. Along with a quasilinear time problem decomposition algorithm, it solves 1-d assignment problems consisting of up to millions of Dirac distributions within fractions of a second in parallel. We handle higher dimensional problems via a slicing approach, and further extend the popular iterative closest point algorithm using optimal transport – an algorithm we call *Fast Iterative Sliced Transport*. We illustrate our method on computer graphics applications such a color transfer and point cloud registration.

Authors' addresses: Nicolas Bonneel, Univ. Lyon, CNRS, nicolas.bonneel@liris.cnrs.fr; David Coeurjolly, Univ. Lyon, CNRS, david.coeurjolly@liris.cnrs.fr.

0730-0301/2019/7-ART4 $15.00
https://doi.org/10.1145/3306346.3323021

CCS Concepts: • **Computing methodologies** → **Image manipulation**; **Point-based models**.

Additional Key Words and Phrases: Optimal transport, sequence alignment

ACM Reference Format:
Nicolas Bonneel and David Coeurjolly. 2019. SPOT: Sliced Partial Optimal Transport. *ACM Trans. Graph.* 38, 4, Article 4 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323021

1 INTRODUCTION

Optimal transport is a popular mathematical framework for manipulating positive measures, and in particular, in most cases studied so far, *probability measures*. It has become widespread in computer graphics [Bonneel et al. 2011; Nader and Guennebaud 2018; Solomon et al. 2015] and machine learning [Arjovsky et al. 2017; Deshpande et al. 2018; Kolouri et al. 2018] for its ability to compare histograms or to produce compelling interpolations between probability distributions. Within this framework, transporting a histogram f towards another g is often seen as moving a pile of sand shaped by the graph of f towards a hole shaped by g at minimal cost. Stopping the motion of the sand at some intermediate time defines a measure in-between f and g, a concept called *displacement interpolation*, or, when dealing with more than two input measures, *Wasserstein barycenter*. This computationally difficult problem has received recent attention, with fast solutions for various specific cases when the size of the hole matches that of the pile of sand, the so-called *balanced* optimal transport problem [Bonneel et al. 2015; Kitagawa et al. 2016; Nader and Guennebaud 2018; Solomon et al. 2015].

In our paper, we focus on discrete distributions with uniform weights, that is, measures of the form $\sum_{x_i} \delta_{x_i}$, i.e., a uniformly

weighted point cloud. In that form, the optimal transport problem between these two measures is also known as a *linear assignment problem*. However, in contrast to most approaches, we do not assume equal total masses, thus allowing for the hole to be larger than the mass it receives. This scenario for instance occurs when one tries to match a 3-d point cloud within a larger point set, or when one tries to transfer colors between images of different sizes, in which case the assignment is only partial.

Among existing solutions, for large datasets possibly involving millions of points scattered in high dimension, the only tractable (balanced) optimal transport algorithm to date relies on a sliced approximation [Bonneel et al. 2015; Pitié et al. 2005; Rabin et al. 2011]. Slicing consists in projecting the point cloud onto random one-dimensional lines, and solving one-dimensional transport problems. The speed of this approach relies on the triviality of solving balanced one-dimensional transport problems. But surprisingly, even in 1-d, the partial optimal transport problem has received little attention, making this sliced approach unusable for point clouds of different sizes.

Drawing connections between partial 1-d optimal transport and sequence alignment problems, we design a first solution of quadratic time complexity and linear space complexity outperforming state-of-the-art dynamic programming solvers such as those used in genomics. We further propose a new provably correct decomposition technique based on the number of non-injective nearest neighbor matches within quasilinear time complexity. This makes our algorithm efficient, even for problems involving millions of point masses, and with each independent sub-problems solvable in parallel. We integrate this 1-d solver within a sliced optimal transport framework to handle higher dimensional point sets. We further adapt the popular Iterative Closest Point (ICP) algorithm to replace nearest neighbor's matches by our sliced partial transport matches. We finally demonstrate applications such as proportion aware color transfer between images, and point set registration. We summarize our contributions as follows:

- We introduce a fast 1-d exact partial optimal transport algorithm that solves problems involving millions on points within fractions of a second. Its speed enables its use within a sliced transport framework to manipulate higher dimensional distributions.
- We develop a variant of the popular *iterative closest point* point set registration algorithm using sliced partial optimal transport that is more reliable on poorly initialized point clouds.

2 PRIOR WORK

Optimal transport is a computationally complex problem, often expressed as a linear program due to Kantorovich (Table 1, (a)), and we refer the reader to recent books on the subject for details on this theory [Peyré et al. 2017; Santambrogio 2015; Villani 2003]. The variables being optimized form what is known as a transport plan to describe the amount of mass from each bin of an input (possibly high-dimensional) histogram that needs to travel to each bin of a target histogram. In this definition, it is assumed that both histograms are normalized. Directly solving this linear program can be very costly, and has only been typically done for histograms of up to a few tens of thousands of bins [Bonneel et al. 2011]. Faster alternate approaches have tried to tackle particular cases, such as semi-discrete formulations [Gu et al. 2013; Kitagawa et al. 2016; Lévy 2015; Mérigot 2011] for matching weighted point sets to continuous densities via geometric constructions, or formulations to match 2-d continuous distributions to uniform continuous distributions [Nader and Guennebaud 2018]. Other approaches approximate the problem, such as entropy-regularized optimal transport, which results in blurred transport plan but is extremely fast to compute, especially when data lie on a grid [Solomon et al. 2015]. However, when data do not lie on a grid, this method may require explicitly storing cost matrices between all pairs of bins thus making it intractable for more than a few tens of thousands of bins. As a last resort, sliced optimal transport [Bonneel et al. 2015; Pitié et al. 2005; Rabin et al. 2011] relies on random 1-d projections for which optimal transport is trivial, but comes at the cost of being optimal only on these projections and not necessarily on the higher dimensional space. The bulk of the computations lies within the sorting of projected points on the line, which comes at an $O(n \log n)$ complexity.

When histograms are not normalized, the transport problem needs to be relaxed. Notably, the "unbalanced" optimal transport of Benamou et al. [Benamou 2003] replaces hard constraints by soft constraints in a Monge formulation to accommodate for densities of unequal masses within a PDE framework. Chizat et al. proposed a fast numerical solution of the unbalanced problem [2016] in the case of entropy-regularized optimal transport with soft constraints on marginals, but suffering from the same space complexity as the balanced problem for scattered data. Figalli introduces *partial optimal transport* that keeps all constraints of the linear program hard [Figalli 2010], but replaces equality with inequality constraints (Table 1, (b)). Our formulation is a special case of this linear program for the problem of matching point clouds (Table 1, (c)).

Our particular case specializes the above problem by Figalli in four ways. First, we consider a simpler 1-d problem and only move to higher dimensions via a slicing scheme. Second, we do not consider general functions or histograms, but only unweighted sets of Diracs; in the notations of Table 1, this amounts to $f_0(x_i) = 1$ for $x_i \in X$ and $f_1(y_j) = 1$ for $y_j \in Y$ where X and Y are two one-dimensional point sets respectively of size m and n. Third, we typically consider the cost of moving a point from one place to another as being a quadratic cost, that is, $c(x, y) = (x - y)^2$ (note that we could consider any convex function of the distance without changing the assignment [Villani 2003]). Finally, we require *all* the mass from the first, smaller, point cloud to be matched to some parts of the target point cloud – in the notations of Table 1, $\eta = \min(\sum f_0, \sum f_1) = m$.

While at the surface this problem seems overly specific, it simply amounts to matching point clouds of different cardinality, a common problem in computer graphics as we shall see in Sec. 6. Stated as an assignment problem, this question has been tackled as the linear assignment problem when point clouds have the same size [Lu and Boutilier 2015]. In a form related to the Monge optimal transport formulation, this formulation amounts to finding an injective map $a : \{1, m\} \hookrightarrow \{1, n\}$ minimizing

$$\min_{a \text{ injective}} \sum (x_i - y_{a(i)})^2 .$$

(a) Kantorovich discrete OT	(b) Partial OT [Figalli 2010]	(c) Linear Assignment (ours)
$W(f_0,f_1)=\min_{\mathbf{P}}\sum_{i,j}c(x_i,y_j)P_{i,j}$ (1)	$W_p(f_0,f_1)=\min_{\mathbf{P}}\sum_{i,j}c(x_i,y_j)P_{i,j}$ (5)	$W_s(f_0,f_1)=\min_{\mathbf{P}}\sum_{i,j}(x_i-y_j)^2P_{i,j}$ (10)
s.t. $P_{i,j}\geq 0\,,\forall i,j$ (2)	s.t. $P_{i,j}\geq 0\,,\forall i,j$ (6)	s.t. $P_{i,j}\geq 0\,,\forall i,j$ (11)
$\sum_{j=1}^{m}P_{i,j}=f_0(x_i)\,,\forall i$ (3)	$\sum_{j=1}^{n}P_{i,j}\leq f_0(x_i)\,,\forall i$ (7)	$\sum_{j=1}^{m}P_{i,j}=1\,,\forall i$ (12)
$\sum_{i=1}^{n}P_{i,j}=f_1(x_j)\,,\forall j$ (4)	$\sum_{i=1}^{m}P_{i,j}\leq f_1(x_j)\,,\forall j$ (8)	$\sum_{i=1}^{n}P_{i,j}\leq 1\,,\forall j$ (13)
	$\sum_{i=1}^{m}\sum_{j=1}^{n}P_{i,j}=\eta$ (9)	

Table 1. We position our formulation (c) with respect to other optimal transport formulations. Formulation (a) assumes histograms are normalized and formalizes the classical optimal transport problem. Formulation (b) deals with histograms of different masses, only transporting a fraction $\eta\leq\min(\sum f_0,\sum f_1)$. Our formulation (c) is a linear assignment problem, and is a special case of (b) when $\eta=\min(\sum f_0,\sum f_1)$ and when dealing with Dirac masses.

However, despite the simplicity of the problem formulation, such assignment problems remain extremely difficult to solve, with even approximate state-of-the-art solutions taking hours to solve for large scale instances (millions of points) [Lu and Boutilier 2015] but using more general cost functions.

When restricting to 1-d domains and the class of cost functions we target, the problem can be trivially solved if $m=n$. In this case, it can be shown that the trivial solution ends up pairing points in order from left to right [Rabin et al. 2011]. Still in 1-d, when $m\neq n$, the problem can be solved by using a dynamic time warping algorithm, but this procedure is much more costly and, to our knowledge, has not been investigated for optimal transport. Such an algorithm makes use of dynamic programming to find an alignment between the two point sequences. This approach has been used in genomics to align DNA sequences [Charter et al. 2000], for aligning text documents to display their "diff" [Gale and Church 1991], to compute the Levenshtein distance between two strings [Levenshtein 1966], or to align audio sequences [Kaprykowsky and Rodet 2006]. A classical exact algorithm is the Hirschberg's algorithm [Hirschberg 1975], that, while only requiring $O(m+n)$ storage (as opposed to $O(mn)$ for standard dynamic programming), still performs in $O(mn)$ time complexity and remains too slow in practice for large problems.

In contrast, while remaining in $O(mn)$ in the worst case, we provide an exact solution of large scale instances of millions of points in fractions of a second in 1-d, and approximately in seconds or minutes in n-d.

3 PARTIAL TRANSPORT IN 1-D

Given a set of points $X=\{x_i\in\mathbb{R}\}_{i=1..m}$ and $Y=\{y_j\in\mathbb{R}\}_{i=1..n}$ on the real line, $m<n$, the goal is to find an injective assignment $a:\{1,m\}\hookrightarrow\{1,n\}$ minimizing $\sum(x_i-y_{a(i)})^2$.

The core of our algorithm lies within an efficient 1-d assignment procedure. It consists of a worst-case quadratic time complexity matching that performs in near linear time in practice (Sec. 3.1), and a problem decomposition that performs in quasilinear time complexity and allows each sub-problem to be solved independently in parallel (Sec. 3.3). Both rely on the same principles, and start with a (non injective) nearest neighbor assignment t.

3.1 Quadratic time algorithm

We first compute a nearest neighbor assignment $t:\{1,m\}\rightarrow\{1,m\}$ between X and Y. This can be performed in linear time by sorting both sequences and scanning them at the same time from left to right since $t(i+1)\geq t(i)$. The nearest neighbor match already minimizes $\sum(x_i-y_{t(i)})^2$ by definition but may not be injective. In fact, if it is injective, the problem is already solved by taking $a=t$. Our algorithm thus consists in resolving issues at or around places where $t(i)=t(j)$ for $i\neq j$. To do so, we scan X from left to right and consider the sub-problem of matching $X'=\{x_i\}_{i=1..m'}$ with Y, progressively increasing m' from 1 to m.

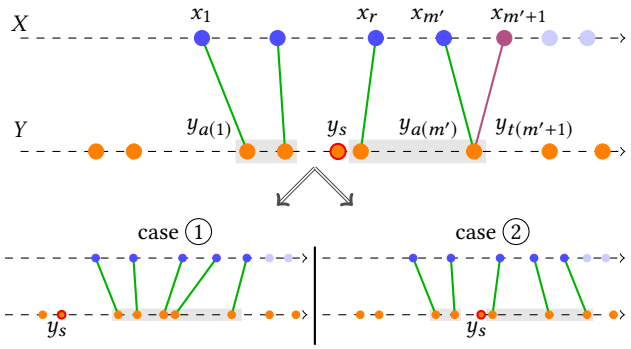


Fig. 2. X is shown as blue dots, Y as orange dots. The assignment for the current sub-problem involving $\{x_i\}_{i=1..m'}$ and Y is displayed as green lines. Intervals of bijective assignments are boxed in gray. When treating $x_{m'+1}$, its nearest neighbor, shown in magenta, collides with the existing assignment. Two cases arise: shifting all green arrows by one to the left (case ①), or preserving all green arrows (case ②), whichever costs less. In this example, case ① merges the two intervals of bijective assignments and s is thus updated.

Notations. Let us assume that we have solved the optimal assignment problem of X' towards Y, denoted by a. The assignment thus associates X' to the range of points $\{y_j\}_{j=a(1)\ldots a(m')}\in Y$ (see Fig. 2). Such a range can be decomposed into intervals of consecutive points of Y which are each assigned to consecutive points of

X'. We denote by s the rightmost point of $\{y_{a(1)} \dots y_{a(m')}\}$ which is not assigned to any point of X'. If the range has no such free spot s (in that case, a is bijective between X' and the range), we set $s \leftarrow a(1) - 1$. We denote by r the index of the point in X' such that $a(r) \leftarrow s + 1$[1]. By construction, $a([r, m'])$ is a range of consecutive integer values $[s + 1, a(m')]$.

Algorithm 1: Quadratic partial optimal assignment.

input : Points X and Y.
output: Optimal assignment of X to Y.

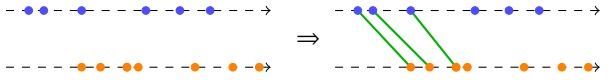
Let t be the nearest neighbor assignment of X to Y;
$a(1) \leftarrow t(1)$;
for m' *from 2 to* m **do**
  if $t(m'+1) > a(m')$ **then**
    $a(m'+1) \leftarrow t(m'+1)$;
  else
    Retrieve s and r ;
    $w_1 \leftarrow \sum_{i=r}^{m'} (x_i - y_{a(i)-1})^2 + (x_{m'+1} - y_{a(m')})^2$;
    $w_2 \leftarrow \sum_{i=r}^{m'} (x_i - y_{a(i)})^2 + (x_{m'+1} - y_{a(m')+1})^2$;
    if $w_1 < w_2$ **then**
      // case (1)
      $a(m'+1) \leftarrow a(m') + 1$;
      $a([r, m']) \leftarrow [s, a(m') - 1]$;
    else
      // case (2)
      $a(m'+1) \leftarrow a(m') + 1$;
return $\{a(i)\}_{i=1\dots m}$.

Update steps. We now consider the point $x_{m'+1}$. If $t(m'+1) > a(m')$, we set $a(m'+1) \leftarrow t(m'+1)$ – this corresponds to the nearest neighbor match $t(m')$ not colliding with any existing assignment. If $t(m'+1) \leq a(m')$, we analyze two cases: case (1) offsets the last subsequence of consecutive values in a to the left to leave room for a new assignment by setting $a([r, m']) \leftarrow [s, a(m') - 1]$ and $a(m'+1) \leftarrow a(m') + 1$. Case (2) directly pushes the new assignment on the right, that is, directly setting $a(m'+1) \leftarrow a(m')+1$. To choose between these two cases, we compare their costs (w_1 and w_2 in Algorithm 1) and proceed with the case with smaller cost. Algorithm 1 describes this process. Its correctness is given in appendix. As we increase m' to m during the update step, we end up with the optimal assignment of X to Y in $O(\max(n + m, m^2))$: the nearest neighbor assignment t is obtained in $O(n + m)$. Then, for (possibly) each m' between 1 and m we evaluate two sums in $O(m' - r)$. As we shall see in Sec. 3.4, we can considerably improve the efficiency of the algorithm making it tractable for large scale problems (millions of points) by recursively updating these sums, only re-evaluating them occasionally.

3.2 Simplifying the problem

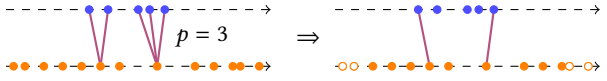
In many cases, parts of the initial problem can be solved in linear time. For instance, we can detect if t is injective in linear time, in which case the problem is solved by setting $a \leftarrow t$. Also, if $m = n$, the trivial assignment $a([1, m]) \leftarrow [1, m]$ is optimal. We explain below three other cases that simplify or even solve the initial problem in linear time.

First, when there exists some k such that $x_i < y_1, \forall i < k$, i.e., X starts "before" Y, we can assign $a(i) \leftarrow i, \forall i < k$. This can be made even stronger: while $x_i < y_i$, we set $a(i) \leftarrow i$ as illustrated below.



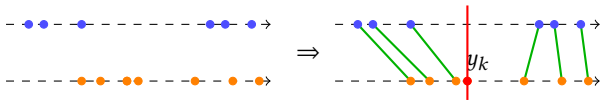
This holds symmetrically at the end of the sequence and often allows to significantly reduce the problem range.

Second, we can further reduce the range of Y based on the number of non-injective values in the nearest neighbor map t. Specifically, let $p = \text{card}\{t(i) = t(i+1), \forall i < m\}$. Then, it is enough to consider the sub-problem of matching X to $Y' = \{y_j\}_{j=\max(1,t(1)-p)\dots \min(t(m)+p,n)}$:



A less tight bound, considering only the interval $[\max(1, t(1) - m), \min(t(m) + m, n)]$ in Y, could be constructed in $O(\log n)$ without requiring any nearest neighbor computation. A proof of both statements is provided in supplementary material.

Finally, when $m = n - 1$, a linear time solution can be obtained, inspired by Hirschberg's algorithm [1975]. In this case, there exists some k to be determined such that $a([1, k-1]) = [1, k-1]$ and $a([k, m]) = [k+1, n]$. Such k minimizes $\min_k \sum_{i=1}^{k-1}(x_i - y_i)^2 + \sum_{i=k}^{m}(x_i - y_{i+1})^2$. Denoting $C = \sum_{i=1}^{m}(x_i - y_{i+1})^2$, the above minimization problem can be written $\min_k \sum_{i=1}^{k}(x_i - y_i)^2 + C - (x_i - y_{j+1})^2$ which can be obtained within a single linear search, as C is a constant and need not be computed.



All these simplification steps have a computational cost in $O(m + n)$ and can be used as preprocessing to reduce X and Y before using the quadratic algorithm of Sec. 3.1.

3.3 Quasilinear time problem decomposition

A key component of our algorithm is a quasilinear time decomposition, often allowing to decompose the initial problem into many small independent sub-problems that can be solved in parallel, and that all benefit from simplifications exposed in Sec. 3.2.

The bottleneck of our quadratic time solution is the need to re-evaluate a possibly large sum many times during the run of the algorithm to determine the better of two cases (1) or (2). The key intuition is that both cases can be kept as candidates without performing any summation. We maintain intervals of points in Y that contain both cases – these intervals may thus contain gaps, which ultimately will not be assigned by an point in X. The main advantage is that such intervals can be maintained in near linear time and can be used to split the problem into shorter independent sub-problems. Each sub-problem A_k is characterized by two points sets $X_k \subset X$

[1] When $a(1) = 1$, s is undefined. In that case, we set $r \leftarrow 1$ and proceed with only case (2) steps of our algorithm described next.

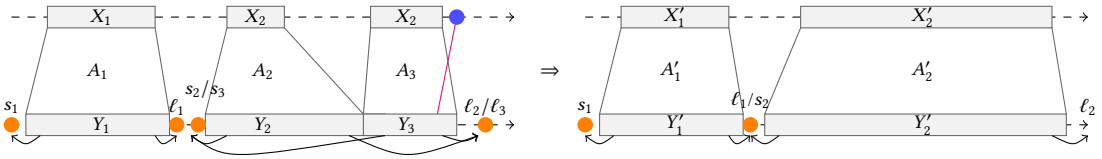


Fig. 3. When adding the blue point in to the linear time problem decomposition, we extend Y_3 on both sides and thus merge the sub-problems A_2 and A_3 (while updating the sub-problem extremities).

and $Y_k \subset Y$, and the index of the first free spot s_k on the left of Y_k and ℓ_k on the right in Y (see Fig.3).

Specifically, we again consider an increasing sub-problem involving $\{x_i\}_{i=1..m'}$ and similarly keep track of the last available free spot s as well as the currently last value ℓ considered. This sub-problem thus involves $\{y_j\}_{j=s+1..\ell}$. We also store a flag $\{f_j\}_{j=1..n}$ where $f_j = \texttt{false}$ indicates that y_j has been considered by either cases ① or ② at some point in the algorithm, or $f_j = \texttt{true}$ otherwise.

Update. We now consider the point $x_{m'+1}$. If $f_{t(m'+1)} = \texttt{true}$, we mark $t(m'+1)$ as occupied by a candidate, and we hence set $f_{t(m')} \leftarrow \texttt{false}$. If $t(m'+1) = t(m')$, we extend the range of the current sub-problem by two candidates, one at the left of the current range and one at its right. This reads $f_s \leftarrow \texttt{false}$ and $f_{\ell+1} \leftarrow \texttt{false}$. If $t(m'+1) \neq t(m')$ but $f_{t(m'+1)} = \texttt{false}$, the sub-problem is only extended on the right, that is, $f_{\ell+1} \leftarrow \texttt{false}$. The proof is detailed in supplementary materials. Our decomposition algorithm is explicited in Algorithm 2.

The amortized cost of retrieving s or merging intervals via a linked-list solution akin to Union-Find makes the expected complexity of each operation to scale as the inverse of the Ackermann function (nearly a constant) [Tarjan and Van Leeuwen 1984].

3.4 Efficient Implementation

A naive implementation of Algorithm 1 would require two sums to be re-evaluated each time $t(m'+1) \leq a(m')$, making it unnecessarily slow for large problems. In practice, we can considerably improve the efficiency of the algorithm. The key idea is to realize that in most cases, these sums can be progressively updated, by storing two sums S_1 and S_2, respectively corresponding to the case ① of offsetting a or ② to keep it unchanged.

These sums can be respectively updated as long as case ② is selected, by using the update formula $S_1 \leftarrow S_1 + (x_{m'+1} - y_{a(m')})^2$ and $S_2 \leftarrow S_2 + (x_{m'+1} - y_{a(m')+1})^2$. However, as soon as case ① is selected, the previously computed sum of the costs of moving the current subsequence to the left becomes the new sum of the costs of leaving the assignment unchanged, and the new sum of the costs of moving the current assignment to the left is invalidated and would need to be recomputed if needed in the future. As such, S_2 should take the value of S_1 and S_1 should be invalidated (it would need to be recomputed by summing values from r to m' if it is needed again, thus making the algorithm quadratic in the worst case). In our implementation, we make use of a linked-list structure akin to *Union-find* to merge intervals of bijective assignments ; we use a similar structure for the quasilinear time decomposition algorithm (Alg. 2). Source code is available in supplementary materials.

Algorithm 2: Decomposition of the assignment problem.

input : Points X and Y.
output: A decomposition of the assignment problem into independent ones.

Let t be the nearest neighbor assignment of X to Y;
Initialize Boolean flags $\{f_j\}_{j=1\ldots n}$ to true;
for m' *from 1 to* m **do**
  if $f_{t(m')}$ *is* true **then**
    $f_{t(m')} \leftarrow$ false;
    Compute the extremities s_k and ℓ_k (if $f_{t(m)-1}$ or $f_{t(m)+1}$ are false we may traverse sub-problems to locate the first free spots);
    Create a new sub-problem $A_k \leftarrow (\{x_{m'}\}, \{y_{t(m')}\}, s_k, \ell_k)$;
  else
    Let $A_{k'}$ be the sub-problem containing $y_{t(m')}$;
    if $f_{s_{k'}}$ (resp $f_{\ell_{k'}}$) is false, merge sub-problem $A_{k'}$ with the one associated to $y_{s_{k'}}$ (resp. $y_{\ell_{k'}}$) ;
    Update the extremities $s_{k'}$ and $\ell_{k'}$;
    if $t(m') \neq t(m-1)$ **then**
      Update the extremity $\ell_{k'}$;
      $A_{k'} \leftarrow (X_{k'} \cup \{x_{m'}\}, Y_{k'} \cup \{y_{\ell_{k'}}\}, s_{k'}, \ell_{k'})$;
    else
      Update both $s_{k'}$ and $\ell_{k'}$;
      $A_{k'} \leftarrow (X_{k'} \cup \{x_{m'}\}, Y_{k'} \cup \{y_{s_{k'}}, y_{\ell_{k'}}\}, s_{k'}, \ell_{k'})$;
return *disjoint sub-problems* $\{A_k\}$.

4 SLICED PARTIAL TRANSPORT

While 1-d optimal transport has limited interest per se, it is the main ingredient of sliced optimal transport. Sliced optimal transport is a very fast algorithm that shares similar properties with optimal transport [Bonneel et al. 2015] and works in n dimensions. It has notably been used for deep learning applications [Deshpande et al. 2018; Kolouri et al. 2018] and other machine learning tools [Kolouri et al. 2016], as well as for computer graphics [Bonneel et al. 2015; Pitié et al. 2005; Rabin et al. 2011]. Its speed comes from the fact that balanced 1-d optimal transport has a trivial linear complexity solution. It further allows to compute sliced Wasserstein barycenters, the equivalent of Wasserstein barycenters in the sliced framework. However, the balanced condition makes it difficult to use for problems that are intrinsically unbalanced.

Our partial 1-d optimal transport solution, while of quadratic complexity in the worst case, is fast (see experiments in Sec. 6.4) and is thus amenable to the computation of sliced optimal transport. Also,

formulas for gradients and Hessians computed for the balanced case by Bonneel et al. [2015] remain unchanged in the partial transport case. The algorithm for sliced (partial) Wasserstein barycenters is summarized below.

The sliced Wasserstein barycenter of a set of d-dimensional point sets $\{X_k\}_{k=1..K}$ weighted by $\{\lambda_k\}_{k=1..K}$ is defined as the minimizer

$$\underset{\widetilde{X}}{\operatorname{argmin}}\ E(\widetilde{X}) = \underset{\widetilde{X}}{\operatorname{argmin}}\ \sum_k \lambda_k \int_{S^{d-1}} W_S(\mathrm{Proj}_\omega(\widetilde{X}), \mathrm{Proj}_\omega(X_k))d\omega\,,$$

where S^{d-1} is the $(d-1)$-dimensional sphere of directions, usually discretized over a finite set of directions Θ (uniformly or randomly), $\mathrm{Proj}_\omega(X)$ projects the point set X onto the line of direction ω as $\mathrm{Proj}_\omega(X) = \{\langle x_i, \omega\rangle\}_i$. W is the transport cost as computed in Sec. 3. Minimizing this energy can be performed via a gradient descent or Newton steps [Bonneel et al. 2015]. In fact, the gradient can be easily expressed as:

$$\nabla E(\widetilde{X}) = \sum_k \lambda_k \int_{S^d} (\mathrm{Proj}_\omega(\widetilde{X}) - \mathrm{Proj}_\omega(X_k \circ a_k))\,,$$

where a_k is the assignment function of the (partial) optimal transport between $\widetilde{X}$ and X_k. The Hessian is also easily computed in closed form as

$$H(\widetilde{X}) = \frac{1}{|\Theta|}\sum_{\theta\in\Theta}\theta\theta^t \approx \frac{1}{d}\mathrm{Id}_{d\times d}\,,$$

where θ^t denotes the transpose of the direction θ. With these values, one can perform Newton's iterations via the update $\widetilde{X} \leftarrow \widetilde{X} - H^{-1}\nabla E(\widetilde{X})$. Transporting an entire point cloud X_0 to X_1 corresponds to a gradient descent of $W^S(X_0, X_1)$. In this case, we resort to a stochastic gradient descent strategy and perform descent steps with a single randomly determined direction at a time.

5 ITERATIVE TRANSPORT ALGORITHM

A common problem in point cloud processing is that of registering points under a given transformation model. For instance, one tries to match a point cloud with another by supposing the transformation between them is rigid, or constrained to a similarity, or affine, or homographic, etc. A well-known algorithm to solve this problem is the Iterative Closest Point (ICP) algorithm. Given a point cloud X_0 to be matched against X_1, this algorithm proceeds by first matching points of X_0 to their nearest neighbors in X_1, and, given this assignment, the best transformation in the class of allowed transformations is found by minimizing an energy (typically, for rigid or similarity transformations, the transformation can be found via an orthogonal Procrustes problem [Schönemann 1966] using Singular Value Decompositions or Kabsch algorithm). The algorithm then transforms the initial point cloud using the computed transformation. The process is repeated until convergence.

However, this algorithm requires points clouds to be relatively close to start with and suffers from local minima. While this has been addressed by Yang et al. [2013] for estimating rigid motion leading to a globally optimal solution, this does not easily extend to other classes of transformations. In practice, we observe that extremely bad behaviors may arise when considering similarity transforms (rotation, translation and scaling). In that case, the lack of injectivity of the nearest neighbor map tends to estimate progressively smaller scaling factors as iterations increase, occasionally leading to a trivial zero scale solution: this solution is in fact globally optimal for the ICP problem, leading to a zero cost of matching the entire input point cloud to a single nearest neighbor point in the target point cloud (see Fig. 9). This motivates the use of a metric which accounts for an injective mapping, such as the sliced metric we are proposing.

We thus propose to replace the nearest neighbor matching by a partial sliced optimal transport matching. We call our algorithm *Fast Iterative Sliced Transport* (FIST). We illustrate our results on 3d point cloud registration using a similarity transform in Sec. 6.2.

6 APPLICATIONS AND RESULTS

This section details two applications of our partial sliced optimal transport framework. The first matches colors in a photograph by possibly using superpixels, and the second uses our FIST algorithm to register point clouds. We then analyze our algorithm in term of performance.

6.1 Color Matching

Transferring colors between images has become a classical image processing problem. The goal is to distort the color distribution of an input image, without changing its content, to match the style of a target image. This matching problem has seen numerous optimal transport solutions [Bonneel et al. 2015, 2013; Pitié et al. 2005; Pitié et al. 2007; Rabin et al. 2010]. In addition to their use of optimal transport, a common point to these approaches is that they consider the problem of matching *normalized* histograms. A consequence of that is often acknowledged as a limitation: their content should not differ. For instance, matching an image with 80% trees and 20% sky to an image containing the opposite ratio will inevitably lead to trees becoming blue or sky becoming green. This is illustrated in our teaser, Fig. 1, in which the input image does not contain the colorful trees of the target image, thus leading to unatural colors in the transferred example using the formulation of Pitié et al. [2005]. Several approaches also require the images to have exactly the same number of pixels – this is precisely the case for sliced transportation [Bonneel et al. 2015; Rabin et al. 2010].

To address this issue we propose two solutions within our SPOT framework. Our first solution simply enlarges the target images to give more freedom to each input pixel to be matched to target pixel values. Our second solution, inspired by that of Rabin et al. [2014] in the context of relaxed optimal transport, segments the input image into a number of superpixels using the SLIC algorithm [Achanta et al. 2010] and consider the problem of matching each superpixel color distribution independently in the target image. To prevent discontinuities between adjacent superpixels cares need to be taken. In particular, all superpixels should use the same set of projection directions, and we further apply the color transfer regularization technique of Rabin et al. [Rabin et al. 2010] (altered to use a cross bilateral filter for simplicity [Paris and Durand 2009]). While the second solution is much faster since all superpixels are smaller and can further be processed in parallel, it sometimes lead to a loss of style as the number of superpixels increases (see Fig. 4). Results for the first solution were often better, albeit slower (see Fig. 1 and 5).

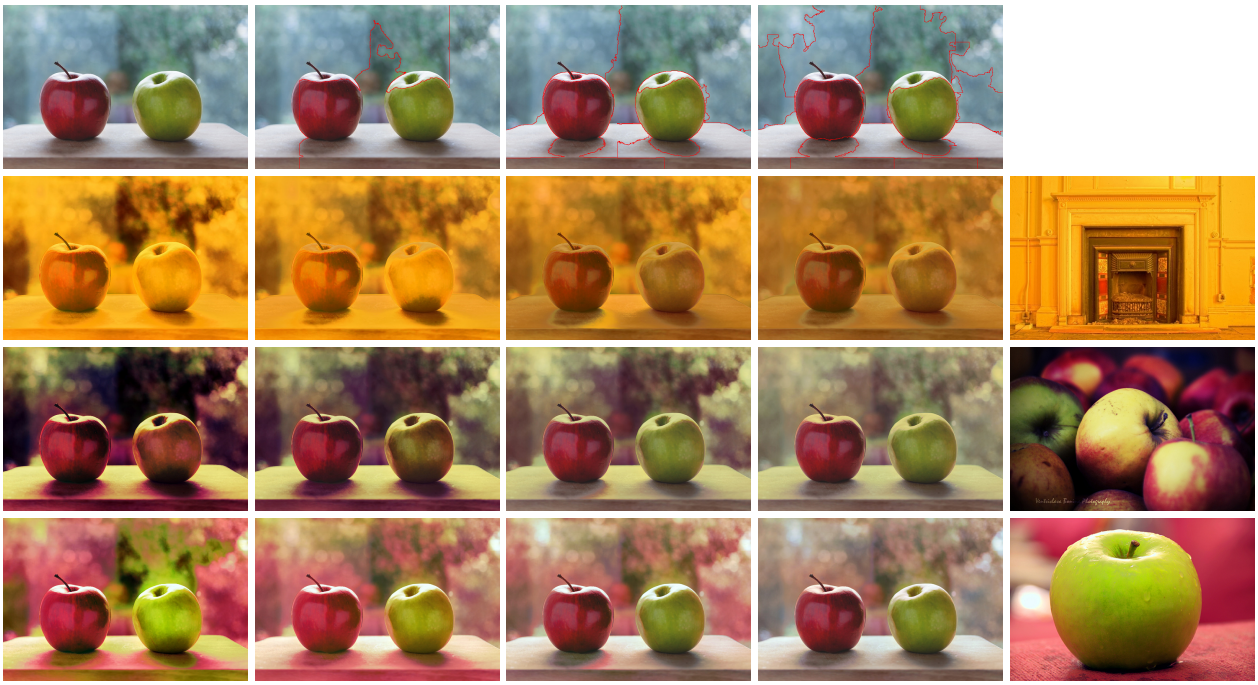


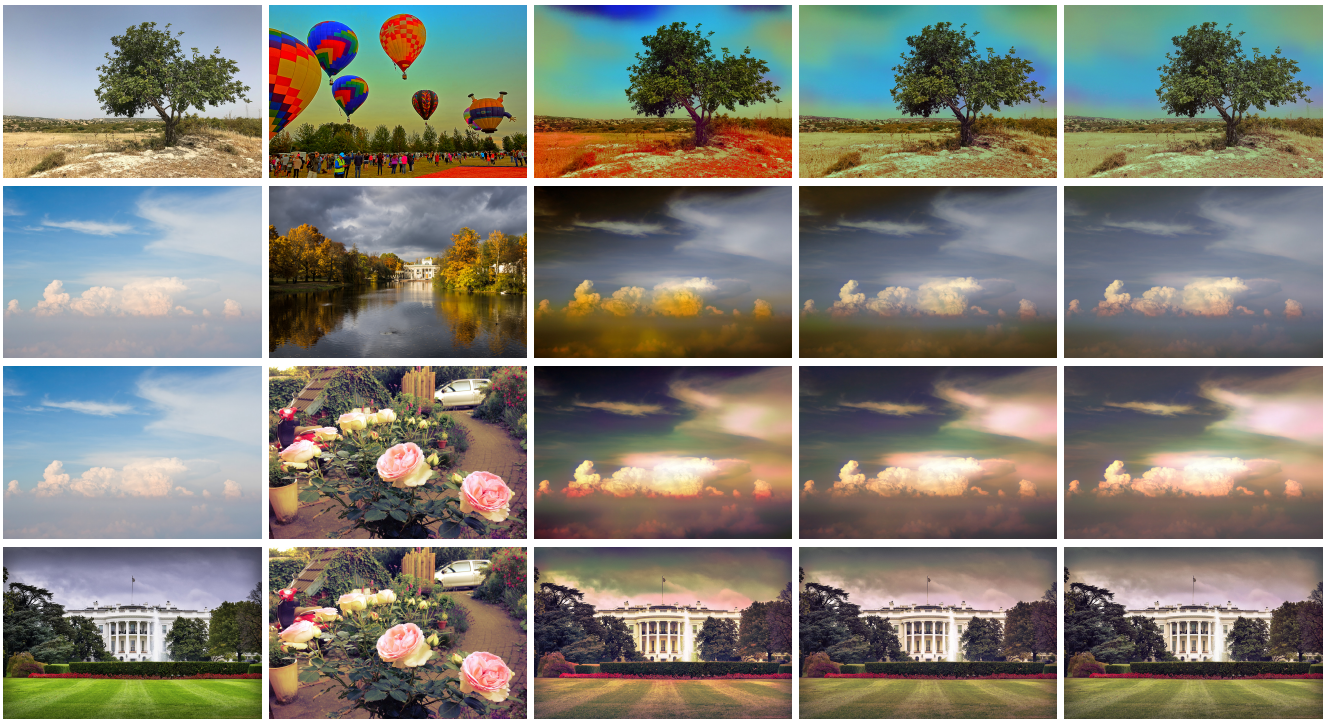
Fig. 4. We perform color transfer between two images of identical sizes (640×427 pixels) and regularize the result. We segment the source images in a number of SLIC superpixels [Achanta et al. 2010] (shown in the first row) and use the target image shown in the last column. A single superpixel (first column) corresponds to the balanced case, with the same energy being minimized as the method presented by Pitié et al. [Pitié et al. 2005]. Despite the color transfer being regularized, large distortions can occur in the balanced sliced color transfer ; increasing the number of superpixels tends to regularize the color mapping as there is more freedom to search for similar-looking pixel subsets in the target image. All color transfers were computed in 5 to 8 seconds for 100 iterations. Increasing to 1000 iterations did not visually impact results.

We compare our partial optimal transport solution to the unbalanced framework of Chizat et al. [2016] in Fig. 6. Their method, based on repeated convolutions [Solomon et al. 2015] requires two Gaussian filterings of so-called *scalings* per iteration. In our experiment, the algorithm typically converged in 200-300 iterations, which results in runnning times of 7-10 minutes when working on 256^3 RGB voxel grids.

6.2 Shape Registration

We illustrate our FIST algorithm on 2-d and 3-d registration examples, representing sampled 2-d images and 3-d shapes. We initialize the input point cloud far from the target point cloud, and we estimate a similarity transform between them. We compared to traditional ICP and an optimal transport solution. ICP systematically fails, estimating a too small scaling factor due to the lack of injectivity. Full fledged optimal transport performed best, but was intractable (see Sec. 6.4). Using our solution, we observed that registering point sets of very different cardinality produces many local minima (see Fig. 8, and failure case in Fig. 14). However, matching point sets of (roughly) similar cardinality performed well in most cases (Fig. 7 in 2-d and 9 in 3-d – see also teaser image 1).

In Table 2, we compare our method against ICP variants that account for scaling factors, proposed by Du et al. [2007], Horn et al. [1988], Umeyama et al. [1991] and Zinsser et al. [2003], all provided in Maass' library [2016]. We also compare with classical ICP and the full-fledged optimal transport solution based on a network simplex solver. To do so, we registered a sampled source Stanford bunny to the same bunny sampled differently, and applied 200 random similarity transforms to the source point cloud. These transforms were sampled uniformly in angle and translation, and using a Fisher-Snedecor distribution F(30,30) for scaling factors. We vary the size of the source point cloud (5k, 8k, 9k and 10k samples), and keep the target point cloud at 10k samples. We compute average and median relative error in Frobenius norm between the ground truth and estimated matrices as homogeneous transformations. The network simplex performs well, but it is also by far the slowest, making it unpractical for larger problems. However, this indicates that optimal transport-based solutions are useful in this context. Then, our method works best when point clouds have similar cardinalities, but is outperformed by that of Zinsser et al. [2003] and Umeyama et al. [Umeyama 1991] when the source point cloud only has half the number of points of the target point cloud. We did not observe significant differences between Umeyama et al. and Zinsser et al. In our experiment, classical ICP systematically resulted in infinite average and median relative errors since in most cases, the



(a) Input (b) Target (c) Full Transfer [Bonneel et al. 2015] (d) 20% Larger (Our method) (e) 40% Larger (Our method)

Fig. 5. We transfer the colors of the target image (b) to the input (a). We either resize the images so that the problem is balanced and leads to the same formulation as that of Pitié [Pitié et al. 2005] (c) or so that the problem becomes unbalanced using a 20% larger target image (both in width and height, column d) or 40% (e).

(a) Full Transfer [Solomon et al. 2015] (b) Unbalanced ($z = 0.99$) [Chizat et al. 2016] (c) Unbalanced ($z = 0.95$) [Chizat et al. 2016]

Fig. 6. For comparison, we repeat the same experiment as the first two rows of Fig. 5, but instead using the (balanced) entropy-regularized optimal transport of Solomon et al. [2015] (a), and the unbalanced variant of Chizat et al. [2016] with two different KL constraints on marginals (using their notation, $z_1 = z_2 = 0.99$ (b) or 0.95 (c)). We post-processed results with the same filtering [Rabin et al. 2010] to reduce quantization artifacts.

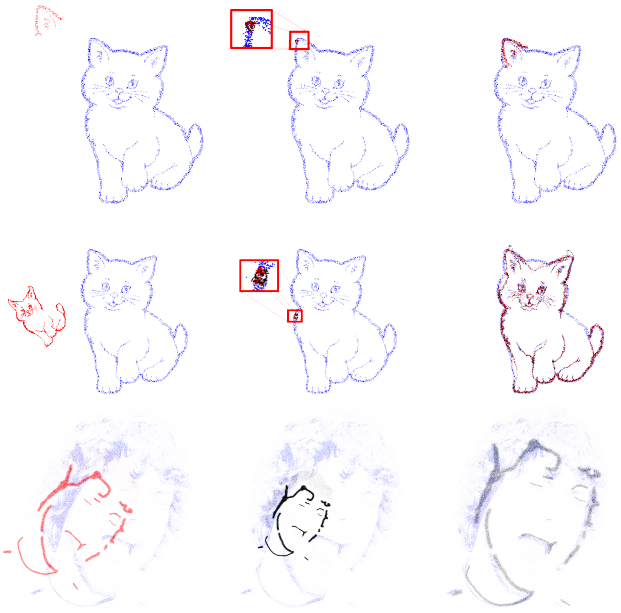
estimated scaling factor was 0 – these values are hence not reported in the table.

Method \# pts	5k	8k	9k	10k
Ours	31.65 (25.53)	3.06 (1.38)	1.73 (0.15)	1.74 (0.16)
Network Simplex	26.70 (13.58)	7.85 (1.85)	3.62 (0.89)	2.14 (0.07)
Du	34.04 (7.26)	33.97 (7.24)	34.02 (7.22)	34.25 (7.34)
Horn	216.45 (77.53)	332.86 (82.27)	INF (86.96)	168.82 (83.73)
Umeyama	21.59 (1.49)	21.58 (1.47)	21.53 (1.48)	21.69 (1.53)
Zinsser	21.59 (1.49)	21.58 (1.47)	21.53 (1.48)	21.69 (1.53)

Table 2. We compare our method against variants of ICP due to Du et al. [2007], Horn et al. [1988], Umeyama et al. [1991] and Zinsser at al. [2003], as well as a (slow) in-house network simplex based optimal transport solution [2011]. We show average (and median, in parenthesis) percentage relative errors in Frobenius norm between ground truth and estimated transformations matrices. We vary the size of the source point cloud from 5k to 10k samples and keep the target point cloud at 10k samples.

6.3 Sliced Partial Barycenters

We illustrate preliminary results on sliced partial barycenters. In Fig. 10, we compute a barycenter between a cat and a dog, both sampled with 100k points. This barycenter is a distribution of points in-between these two input distributions in term of optimal transport distances. It requires computing gradient descent steps that may lead to different local minimas depending on the initialization and lack of convexity of the energy landscape. To study this behaviour, we perform this gradient descent by either initializing the



(a) Point clouds (b) Euclidean ICP (c) Our method

Fig. 7. We register the red to the blue point cloud in (a) using a similarity transform (translation, rotation and scaling). Samples on both point clouds are different. We used a classical Iterative Closest Point algorithm in (b) that uses nearest neighbors and our sliced partial transport matching in (c).

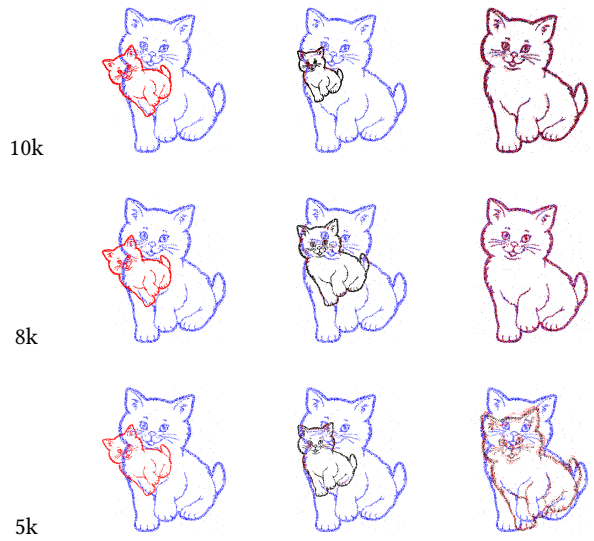


Fig. 8. We compare classical ICP with our method when reducing the number of points being matched. The target distribution always has 10k samples, but we vary the source distribution from 5k to 10k samples. We observe that matching point clouds of similar cardinality behave better: convergence speed increases, and local minimas are less frequently encountered.

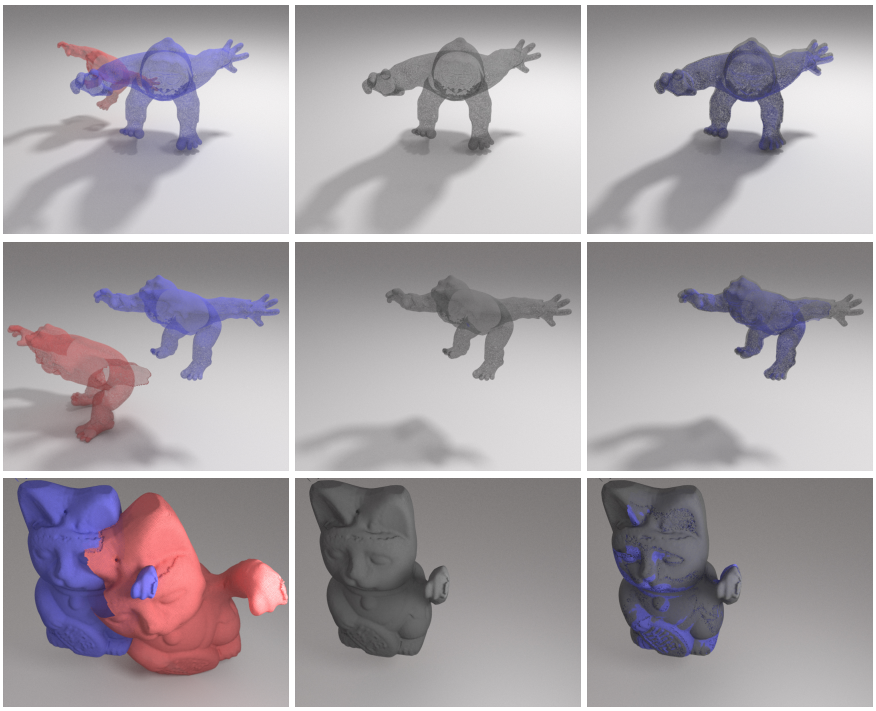
barycenter point cloud with the cat samples, or with a uniformly random point cloud. We also vary the number of samples within this barycenter. We observe that as the number of samples in the barycenter approaches the number of samples in the input distributions (100k), the influence of the initialization is reduced: the energy has less local minimas as the problem is more balanced.

6.4 Performance Analysis

We first analyze the ability of our decomposition algorithm to produce a number of sub-problems on a typical FIST application. We match two sampled cat images, shown in Fig. 7, second row. The input distribution has 8k samples and the target has 10k samples. We use 40 FIST iterations and 20 slices, ultimately producing 800 linear assignment problems of size $8k \times 10k$ to solve. Among those, 70 problems could not be decomposed any further and had to be solved entirely with the quadratic time algorithm, and 13 problems were decomposed in only two smaller sub-problems. Among these smaller sub-problems, a number of them were trivially solved with the special cases handled in Sec. 3.2. The proportion of these sub-problems is shown in orange. As the initial problem is decomposed into more sub-problems, their size also become smaller on average, and the proportion of them trivially solved hence increases.

We further analyze the performance of our algorithm on the same example when varying the number of samples in the source and target distributions. Results are shown in Fig. 12 – computation times do not include the sorting cost, but only the 1-d matching. In most cases, the algorithm runs in fractions of a second, even when matching millions of samples. All runs were performed on a 16-core machine. Interestingly, slower problems arise for $\frac{m}{n} \approx 0.5$: matching point clouds of either nearly equal or very different cardinality produces many trivial sub-problems. In Fig. 13, we illustrate the convergence rate of a sliced distance computation between two sampled 3-d characters seen in Fig. 9.

We finally provide total running times per FIST iteration, sorting time included, on the 2-d and 3-d examples shown in the paper in Table 3. We compare our performance with three competing strategies. First, we compare to a nearest neighbor based ICP implemented using the fast nanoflann nearest neighbor search library. While this method is about 10× faster per iteration, it completely fails in our registration experiments with poorly initialized matches, and also tends to require more iterations. Second, we compare to the strategy of matching samples via an exact optimal transport solver (as opposed to optimal per slice) that uses a state-of-the-art linear programming solver based on a network simplex algorithm [Bonneel et al. 2011]. We observed that this strategy leads to better quality results (see Fig. 14) and often converges in less iterations than our sliced approximation, but it remains between 1000 and 4000 times slower per iteration on smaller problems, and cannot be run on problems of more than a few tens of thousands samples due to memory limits. We finally compared to a house-made variant of Hirschberg's algorithm [Hirschberg 1975] that first decomposes the initial problem into sub-problems using Algorithm 2, and recursively solves linear-space and quadratic-time complexity dynamic programming, invoking the simplification of sub-problems detailed in Sec. 3.2 as



(a) Point clouds (b) Euclidean ICP (c) Our method

Fig. 9. We register the red 3d point cloud to the blue point cloud shown in (a) using a similarity transform estimated with a classical ICP and our transport based method, similarly to the setup in Fig. 7. In (b) and (c), the registered point set is shown in blue. In practice, for classical ICP, this point set is barely visible, if at all: the registration often leads to degenerate configurations with extremely small estimated scaling factors.

termination criteria for this recursive approach. This was initially considered to be part of our solution, and has been well optimized (parallelized and vectorized, partly on the GPU). Still, this method performs between 5 to 150 times slower than the proposed solution.

7 LIMITATIONS, CONCLUSIONS AND FUTURE WORK

We have proposed the first fast approach for computing sliced optimal transport of point sets of different cardinalities, and used it for color matching and within a similarity transform estimation algorithm. However, difficult fundamental questions remain regarding its behavior on energy landscapes when cardinalities differ significantly. For instance, our preliminary experiments with sliced partial Wasserstein barycenters show that a barycenter with much less samples than the input dataset heavily depends on the initialization, while this is not the case as the number of points in the barycenter increases. This would indicate that, as the number of points increases, the energies involved seem empirically more convex. We observe a similar behavior within our FIST algorithm. In addition, measuring optimal transport from slices may give surprising solutions, which are optimal per slice but far from globally optimal (Fig. 14, second row). Unfortunately, computing exact optimal transport remains intractable to date on large problems. We believe a mixed strategy may provide interesting solutions to these issues, by initially performing FIST steps and occasionnally switching to ICP.

Given the speed of our sequence alignment procedure, it could also be worth investigating its generalization to other alignment tasks such as DNA alignment. Still, we have shown that SPOT can be useful in practice for computer graphics on color and point set processing. We expect it could be further used for other applications such as part-based geometry retrieval, or machine learning applications similarly to traditional sliced transport.

Acknowledgments We thank the anonymous reviewers for their detailed feedback to improve the paper. This project was supported in part by French ANR ROOT (ANR-16-CE23-0009) and CoMeDiC (ANR-15-CE40-0006) grants. Image credits to Flickr users Phil Whitehouse, Kirt Edblom, Steve Parker, Tim Wang, Dominic Alves, Eneko Castresana Vara, Felix Schaumburg, Neil Williamson, Ventsislava Bonina, Diego Cambiaso and freestocks.org.

A CORRECTNESS OF ALGORITHM 1.

Let $a([1, m'])$ be the optimal assignment of X' to Y and let us consider a new point $x_{m'+1}$. First, we consider the case $t(m'+1) > a(m')$. This implies that $x_{m'+1}$ should be assigned to $y_{t(m'+1)}$ (lines 4–5) to minimize $\sum(x_i - y_{a(i)})^2$ as any other assignment of $x_{m'+1}$ would have a higher cost.

We now suppose that $t(m' + 1) \leq a(m')$. We claim that $x_{m'+1}$ is assigned either to $y_{a(m')+1}$ (case ②), or to $y_{a(m')}$ with a shift of the assignments of $\{x_r, \ldots, x_{m'}\}$ by one to the left until the first

Name	Source	Target	Slices	Time per iter.	# iter	Time per iter Network Simplex	Time per iter Nearest Neighbor	Time per iter Hirschberg
2d Cat	5k	10k	20	0.03	200	31	0.003	0.15
2d Cat	8k	10k	20	0.04	130	40	0.005	0.56
2d Cat	10k	10k	20	0.04	14	155	0.006	0.04
2d Car	90k	100k	20	0.66	16	X	0.05	83
2d Face	120k	200k	40	1.72	100	X	0.07	188
3d Cat	150k	200k	100	4.71	200	X	0.10	745
3d Castle	150k	200k	40	2.18	55	X	0.09	221
3d Character (cut)	80k	100	100	2.29	100	X	0.05	274
3d Character (full)	90k	100	20	0.69	100	X	0.05	40

Table 3. This table reports data for our FIST registration experiments: the number of samples for the source and target distribution, the number of slices used, the computation time per FIST iteration, and the number of iterations it took to converge. We also provide when available running times for competing approaches: matching points via full-fledged optimal transport correspondences computed with a network simplex solver [Bonneel et al. 2011], a traditional ICP using a fast nearest neighbor search, and a custom slice-based approach using Hirschberg's dynamic programming algorithm for 1-d matching. While we found the network simplex makes registration converge in less iterations, it is simply intractable for problems larger than a few thousand samples, mostly due to its memory requirements that grow in $O(mn)$.

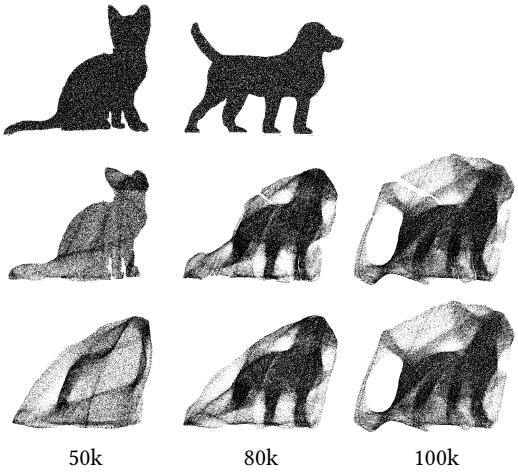


Fig. 10. Sliced partial Wasserstein barycenter ($\lambda_0 = \lambda_1 = 0.5$) of two $100k$ points sets (first row) representing a cat and a dog, using a varying number of samples. The minimization is initialized with the cat point set (second row) or with uniform random samples (third row). With $100k$ samples, this corresponds to the balanced case, solved in Bonneel et al. [2015]. While even in this case the energy is non-convex, both initializations result in similar centroids. However, when reducing the number of samples in the barycenter, the two initializations produce different results, illustrating the lack of convexity of the problem.

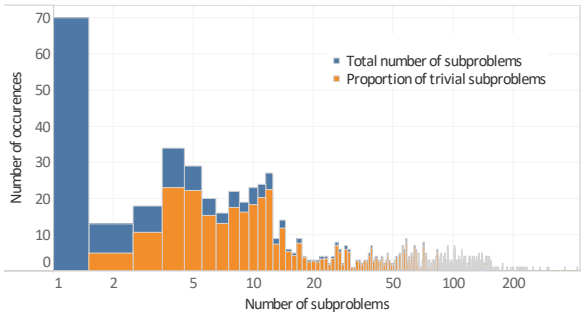


Fig. 11. Performing a FIST on an 8k to 10k problem using 20 slices and 40 FIST iterations, we analyze how the quasilinear time problem decomposition helps reduce the compexity. This graph shows how the 800 1-d problems were split into sub-problems of various sizes, as shown in blue, a fraction of which were trivial to completely solve via linear time procedures explained in Sec. 3.2. In particular, only 70 problems were impossible to decompose into simpler sub-problems, 13 problems were split into only 2 smaller sub-problems 5 of which being trivial to solve.

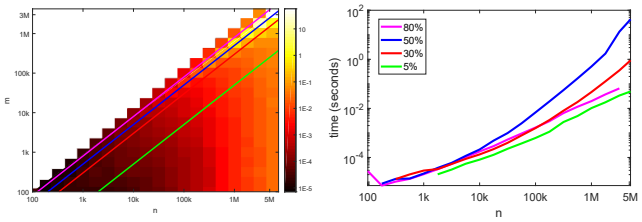


Fig. 12. **Left.** Performance of our 1-d assignment problem as a function of m and n when matching two point clouds (time in seconds averaged over 100 slices). **Right.** For clarity, we extract 4 lines of constant $\alpha = \frac{m}{n}$ ratio (also shown on the left) and express the time of matching αn 1-d points with n points as a function of n.

empty spot y_s (case ①). First, $x_{m'+1}$ cannot be optimally assigned to any y_l with $l > m' + 1$. Indeed, as the nearest neighbor of $x_{m'+1}$ is (strictly) before $y_{a(m')+1}$, any other assignment of $x_{m'+1}$ would have a higher cost. By definition, the assignment a is bijective between $\{x_r, \dots, x_{m'}\}$ and $\{y_{a(r)}, \dots, x_{a(m')}\}$) and is the optimal assignment between X' and Y. If $x_{m'+1}$ is assigned to $y_{a(m')}$, then the optimal assignment is obtained by only shifting the assignments of $\{x_r, \dots, x_{m'}\}$ by one to the left. Indeed, shifting those assignments by more than one would create a free spot $y_l \in \{y_{a(r)}, \dots, y_{a(m')}\}$ not assigned to any point in $X' \cup \{x_{m'+1}\}$ (let m'' be the index of the point in X' associated with y_{l-1}), leading to two independent sub-problems. This would have been detected at step m'' of Algorithm 1,

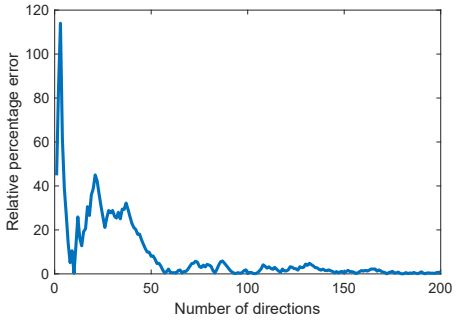


Fig. 13. Rate of convergence of a sliced optimal transport computation between two 3-d points set of 80k and 100k samples illustrated in Fig. 9 (first row) shown as percentage of relative error. Although not monotonously decreasing, the error falls below 1% after 200 slices are used.

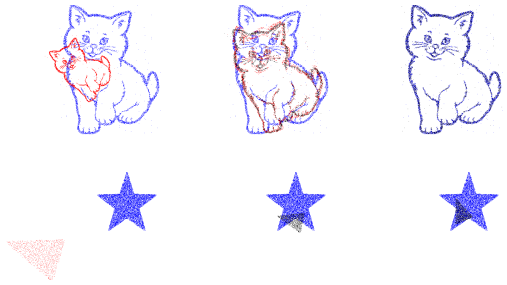


Fig. 14. We show the occasional behavior of sliced optimal transport (second column) when cardinalities differ too much. In the first row, we encounter a local optimum, when matching 5k to 10k samples. The second row shows slicing issues when matching 1k to 10k samples: while the transport is optimal per slice, it is not optimal in 2-d as samples of the triangle are outside of the star. In both cases, full-fledged optimal transport (third column) gives the expected result, but is intractable for more complex problems.

which is a contradiction to the fact that a is the optimal assignment of X' to Y. Similarly, if $x_{m'+1}$ is assigned to any point before $y_{a(m')}$ the assignment would have a higher cost. Algorithm 1 compares the costs of the two alternatives, ① and ②, to obtain the optimal partial assignment $a : \{1, m'+1\}$ minimizing $\sum(x_i - y_{a(i)})^2$.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. 2010. *Slic superpixels.* Technical Report.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. (2017). arXiv:1701.07875.

Jean-David Benamou. 2003. Numerical resolution of an "unbalanced" mass transport problem. *ESAIM: Mathematical Modelling and Numerical Analysis* 37, 5 (2003).

Nicolas Bonneel, Julien Rabin, G. Peyré, and Hanspeter Pfister. 2015. Sliced and Radon Wasserstein Barycenters of Measures. *J. of Math. Imaging and Vision* 51, 1 (2015).

Nicolas Bonneel, Kalyan Sunkavalli, Sylvain Paris, and Hanspeter Pfister. 2013. Example-Based Video Color Grading. *ACM Trans. on Graphics (SIGGRAPH)* 32, 4 (2013).

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *ACM Trans. on Graphics (SIGGRAPH Asia)*, Vol. 30.

Kevin Charter, Jonathan Schaeffer, and Duane Szafron. 2000. Sequence alignment using FastLSA. In *International conference on mathematics and engineering techniques in medicine and biological sciences*. 239–245.

Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. 2016. Scaling Algorithms for Unbalanced Transport Problems. *Math. Comp.* 87 (07 2016).

Ishan Deshpande, Ziyu Zhang, and Alexander Schwing. 2018. Generative Modeling using the Sliced Wasserstein Distance. In *IEEE Conf. on Comp. Vis. and Patt. Recognition (CVPR)*.

Shaoyi Du, Nanning Zheng, Shihui Ying, Qubo You, and Yang Wu. 2007. An extension of the ICP algorithm considering scale factor. In *IEEE Int. Conf. on Image Processing (ICIP)*, Vol. 5.

Alessio Figalli. 2010. The optimal partial transport problem. *Archive for rational mechanics and analysis* 195, 2 (2010), 533–560.

William A Gale and Kenneth W Church. 1991. Identifying word correspondences in parallel texts. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Xianfeng Gu, Feng Luo, Jian Sun, and S-T Yau. 2013. Variational principles for Minkowski type problems, discrete optimal transport, and discrete Monge-Ampere equations. *arXiv:1302.5472* (2013).

Daniel S. Hirschberg. 1975. A linear space algorithm for computing maximal common subsequences. *Commun. ACM* 18, 6 (1975), 341–343.

Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. 1988. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A* 5, 7 (1988).

Hagen Kaprykowsky and Xavier Rodet. 2006. Globally optimal short-time dynamic time warping, application to score to audio alignment. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 5.

Jun Kitagawa, Quentin Mérigot, and Boris Thibert. 2016. Convergence of a Newton algorithm for semi-discrete optimal transport. *arXiv:1603.05579* (2016).

Soheil Kolouri, Charles E Martin, and Gustavo K Rohde. 2018. Sliced-Wasserstein Autoencoder: An Embarrassingly Simple Generative Model. *arXiv:1804.01947* (2018).

Soheil Kolouri, Yang Zou, and Gustavo K Rohde. 2016. Sliced wasserstein kernels for probability distributions. In *IEEE Conf. on Comp. Vis. and Patt. Recognition (CVPR)*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

Bruno Lévy. 2015. A Numerical Algorithm for L2 semi-discrete optimal transport in 3D. *ESAIM M2AN (Mathematical Modeling and Numerical Analysis)* (2015).

Tyler Lu and Craig Boutilier. 2015. Value-Directed Compression of Large-Scale Assignment Problems. In *AAAI Conf. on Artificial Intelligence*.

Edgar Maass. 2016. *Point cloud alignment: ICP methods compared.* Technical Report.

Quentin Mérigot. 2011. A Multiscale Approach to Optimal Transport. *Computer Graphics Forum* (2011).

Georges Nader and Gael Guennebaud. 2018. Instant Transport Maps on 2D Grids. *ACM Trans. on Graphics (SIGGRAPH Asia)* 249 (2018).

Sylvain Paris and Frédo Durand. 2009. A fast approximation of the bilateral filter using a signal processing approach. *International journal of computer vision* 81, 1 (2009).

Gabriel Peyré, Marco Cuturi, et al. 2017. Computational optimal transport. *arXiv:1803.00567* (2017).

Francois Pitié, Anil C. Kokaram, and Rozenn Dahyot. 2005. N-Dimensional Probablility Density Function Transfer and Its Application to Colour Transfer. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

François Pitié, Anil C Kokaram, and Rozenn Dahyot. 2007. Automated colour grading using colour distribution transfer. *Comp. Vis. and Im. Understanding* 107, 1-2 (2007).

Julien Rabin, Julie Delon, and Yann Gousseau. 2010. Regularization of transportation maps for color and contrast transfer. In *IEEE Int. Conf. on Image Processing (ICIP)*.

Julien Rabin, Sira Ferradans, and Nicolas Papadakis. 2014. Adaptive color transfer with relaxed optimal transport. In *IEEE Int. Conf. on Image Processing (ICIP)*.

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.

Filippo Santambrogio. 2015. Optimal transport for applied mathematicians. *Birkäuser, NY* (2015).

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (1966), 1–10.

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 Trans. on Graphics (SIGGRAPH)* 34, 4 (2015).

Robert E Tarjan and Jan Van Leeuwen. 1984. Worst-case analysis of set union algorithms. *Journal of the ACM (JACM)* 31, 2 (1984), 245–281.

Shinji Umeyama. 1991. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Patt. Analysis & Machine Intelligence* 4 (1991).

Cédric Villani. 2003. *Topics in optimal transportation.* American Mathematical Soc.

Jiaolong Yang, Hongdong Li, and Yunde Jia. 2013. Go-icp: Solving 3d registration efficiently and globally optimally. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. 2003. A refined ICP algorithm for robust 3-D correspondence estimation. In *IEEE Int. Conf. on Image Processing*.