Linear-Time Transport with Rectified Flows

KHOA DO, University of Michigan, USA
DAVID COEURJOLLY, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France
POORAN MEMARI, CNRS, LIX, École Polytechnique, Inria, IP-Paris, France
NICOLAS BONNEEL, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France

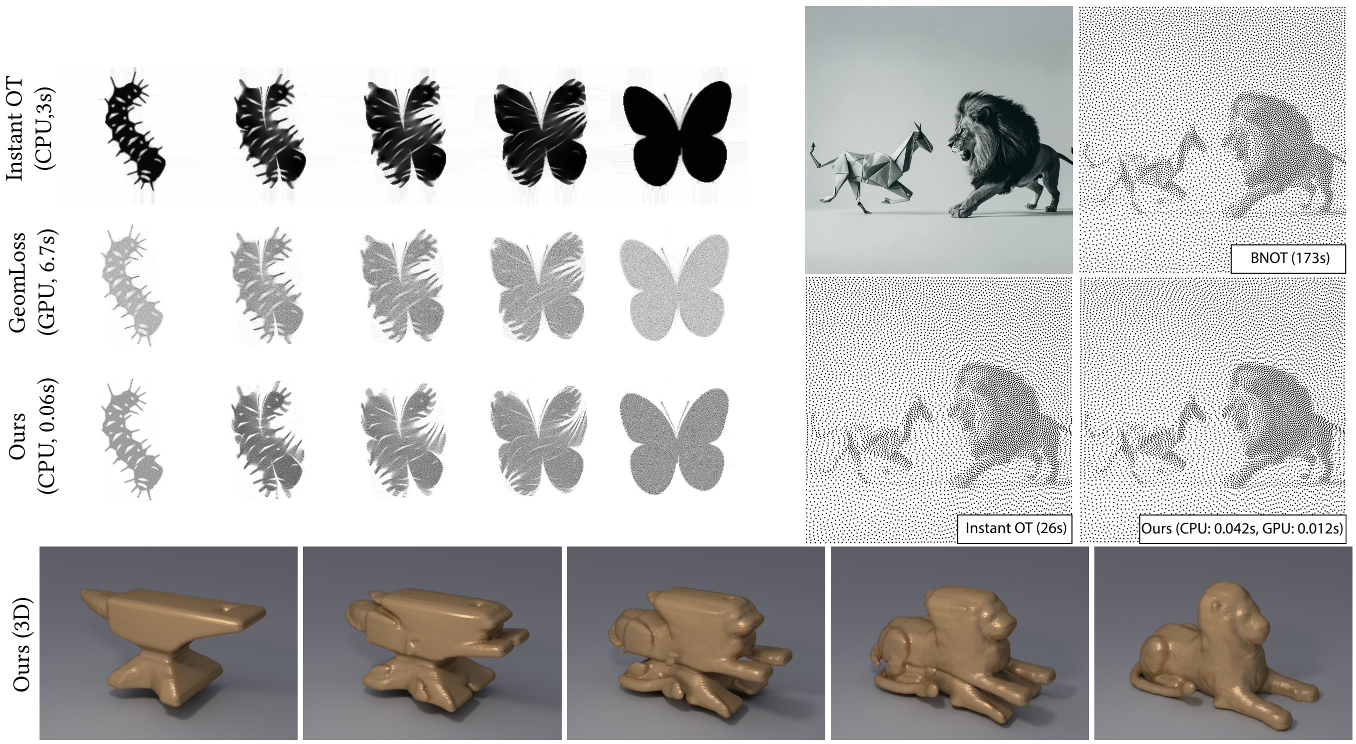

Fig. 1. Our transport algorithm computes shape interpolations and image stipplings in linear time, with results approaching those of optimal transport. In 2D, the interpolation is performed on a 256x256 grid with 65,536 samples and timings correspond to computing 5 interpolation steps; stippling is performed on a 1024x1024 grid with 8,192 samples. We compare our results to a GPU-optimized linearized transport with Sinkhorn divergences [Feydy et al. 2019b], the Instant Transport of Nader and Guennebaud [2018], the high-quality stippling of BNOT [de Goes et al. 2012], and that obtained using Instant Transport (timings for Instant OT stippling include 23.5s of precomputations that only depend on grid resolution). In our 3D example, our process takes 815s to compute transport maps on a 256^3 resolution with 16.8 millions of particles, and each interpolation step then requires 3s (splatting particles and mesh reconstruction). *Anvil model by TurboSquid user PotatoOrgyLt and Lion by CGTrader user tudorfat.*

Matching probability distributions allows to compare or interpolate them, or model their manifold. Optimal transport is a tool that solves this matching problem. However, despite the development of numerous exact and approximate algorithms, these approaches remain too slow for large datasets due to the inherent challenge of optimizing transport plans. Taking intuitions from recent advances in rectified flows we propose an algorithm that, while not resulting in optimal transport plans, produces transport plans from uniform densities to densities stored on grids that resemble the optimal ones in practice. Our algorithm has linear-time complexity with respect to the problem size and is embarrassingly parallel. It is also trivial to implement, essentially computing three summed-area tables and advecting particles with velocities easily computed from these tables using simple arithmetic. This already allows for applications such as stippling and area-preserving mesh parameterization. Combined with linearized transport ideas, we further extend our approach to match two non-uniform distributions. This allows for wider applications such as shape interpolation or barycenters, matching the quality of more complex optimal or approximate transport solvers while resulting in orders of magnitude speedups. We illustrate our applications in 2D and 3D.

Authors' Contact Information: Khoa Do, University of Michigan, USA, khoadb@umich.edu; David Coeurjolly, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France, david.coeurjolly@cnrs.fr; Pooran Memari, CNRS, LIX, École Polytechnique, Inria, IP-Paris, France, memari@lix.polytechnique.fr; Nicolas Bonneel, CNRS, Université Claude Bernard Lyon 1, INSA Lyon, France, nicolas.bonneel@liris.cnrs.fr.

ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731147

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

Additional Key Words and Phrases: optimal transport, interpolation, rectified flows

ACM Reference Format:
Khoa Do, David Coeurjolly, Pooran Memari, and Nicolas Bonneel. 2025. Linear-Time Transport with Rectified Flows. *ACM Trans. Graph.* 44, 4 (August 2025), 13 pages. https://doi.org/10.1145/3731147

1 Introduction

Optimal transport is a mathematical theory that allows one to compare probability distributions by interpreting them as piles of sand transported at minimal cost. The cost of moving an entire pile of sand towards another becomes a distance between these probability distributions, and stopping the sand motion in-between results in an intermediate distribution that can be used as an interpolation. Many algorithms have been proposed to compute the optimal transport between distributions. Because obtaining fast algorithms on this problem is difficult, much effort has been devoted either to solving particular instances or devising approximate solutions. While the resulting algorithms have been reported to be fast [Cuturi 2013; Feydy et al. 2019a,a; Jacobs and Léger 2020; Lévy et al. 2021; Nader and Guennebaud 2018; Solomon et al. 2015], their time complexity has remained ranging from $O(n \log n)$ for rough approximations to $O(n^3 \log n)$ for exact solutions. This is however hard to improve for optimal solution in general, based on the known theoretical complexities.

Among particular instances of interest, the problem of computing the optimal transport map between a uniform density towards a given non-uniform density has received much attention. This can be explained by the numerous applications that involve this setting, such as recovering incompressibility in fluid simulation [Gallouët and Mérigot 2018; Levy 2018], enforcing area preservation in conformal mapping [Dominitz and Tannenbaum 2009; Zhao et al. 2013], modeling the early universe from a uniform mass distribution at time 0 [Lévy et al. 2021], redirecting an input uniform light distribution to produce caustics [Meyron et al. 2018], stippling images by warping uniform patterns [Nader and Guennebaud 2018], or generating uniform Monte Carlo samples by optimizing the locations of points [Paulin et al. 2020]. Further, an approximate transport between two non-uniform distributions can be obtained by considering an intermediate – for instance, uniform – reference distribution. This process is known as linearized optimal transport [Nader and Guennebaud 2018], and is able to produce reasonable approximations of optimal transport in many cases.

In our work, we primarily tackle the problem of transporting a uniform distribution towards a non-uniform distribution stored on a grid, and use linearized optimal transport to address the more general case of matching two non-uniform distributions. Our work is inspired by the concept of rectified flows [Liu 2022; Liu et al. 2022]; our intuition is that when one of the distributions is uniform, one iteration of rectified flow can be achieved in closed form, which makes it extremely fast. Our algorithm takes as input a grid containing a non-uniform density (e.g., a grayscale image in 2D or a voxelized shape in 3D) and a set of particles uniformly sampling the domain. It then computes three summed-area tables and readily uses them to advect these particles. After advection, the particles exactly follow the input non-uniform density, and the matching between their initial position and their final advected position is similar to an optimal transport map. In our settings, the velocity field is entirely determined by solving the continuous problem of transporting a uniform density to a piecewise-constant density stored on a grid, but the transport map is evaluated at discrete locations by advecting particles. The algorithm is trivial to implement and is fully described in less than a single column of our article using standard arithmetic, is embarrassingly parallel, can be implemented on the GPU, results in linear-time complexity in the number of particles and input density grid discretization (though exponential in the dimension), and allows for many low-dimensional applications where optimal transport is routinely used. We demonstrate these applications and show that they result in comparable quality to that reported in the literature, but orders of magnitude faster. Typical runtimes are 42ms for transporting 8k points towards a 1024x1024 grid on the CPU or 12ms on the GPU, all precomputations and computations included. Combined with linearized transport, it becomes easy to extend our method to the problem of transporting two arbitrary discretized densities. We demonstrate applications such as 3D shape interpolation.

Our contribution consists in reformulating rectified flows in the case where one distribution is uniform to achieve linear-time complexity and realtime results in computer graphics applications requiring optimal transport. Code is available in supplementary materials and at https://github.com/DoBachKhoa/RectFlowOT.

2 Background and related work

2.1 Optimal transport

We review the main principles and algorithms of optimal transport and refer to the survey of Bonneel and Digne [2023] for a more exhaustive review and applications to computer graphics, and to the survey of Peyré and Cuturi [Peyré et al. 2019] for details on numerical methods. We further allow simplified notations for clarity.

Problem statement. Optimal transport provides a distance between distributions $\mu(x)$ $(x \in X)$ and $\nu(y)$ $(y \in Y)$ by optimizing a transport plan $\pi(x, y)$ with respect to some cost, such that both marginals match respectively μ and ν. This defines an energy $W_p^q(\mu, \nu)$ that corresponds to the so-called Wasserstein distance between μ and ν:

$$W_p^q(\mu, \nu) = \min_{\pi} \int_{X \times Y} \|x - y\|_p^q d\pi(x, y) \tag{1}$$

$$s.t. \int_Y d\pi(x, y) = \mu(x) \ \forall y \tag{2}$$

$$\int_X d\pi(x, y) = \nu(y) \ \forall x \tag{3}$$

$$\pi(x, y) \geq 0 \ \forall x, y, \tag{4}$$

where $\|x - y\|_p^q$ is the L^p distance at the power q between two points of coordinates $x = (x_i)_{i=1..d}$ and $y = (y_i)_{i=1..d}$ defined as $\|x - y\|_p^q = \left(\sum_{i=1}^d |x_i - y_i|^p\right)^{q/p}$. This amounts to solving a linear program. In our context, $X = Y = [0, S_0] \times [0, S_1] \times \ldots \times [0, S_{d-1}] = B$, a d-dimensional box. The quadratic cost ($p = q = 2$) is often used as the cost of moving a particle from location x to y – the so-called *ground distance* – but other costs are occasionally used in the literature.

Exact solutions. For discrete distributions supported on n points, a network simplex provides solutions in $O(n^3 \log n)$, and can be efficiently implemented in practice [Bonneel et al. 2011; Kelly and O'Niell 1991]. The setting of uniform discrete distributions with the same number of samples amounts to a linear assignment problem between samples.

In a semidsicrete setting where μ is a continuous distribution, ν is discrete, and considering W_2^2, the structure of the problem results in a power diagram – i.e., a Voronoï diagram of controllable cell sizes – which cells can be optimized with convex optimization methods [Lévy 2015]. This optimization iteratively computes power diagrams, each in $O(n \log n + n^{\lceil d/2 \rceil})$ in d dimensions. In practice, this results in very fast implementations that, with engineering efforts, can even be used to support problem sizes of tens of millions of 3D samples in hours on a personal computer [Lévy et al. 2021]. Similarly, the case W_2^1 results in Apollonius diagrams, which can also be obtained efficiently [Hartmann 2017].

Closer to our work, the Instant Transport of Nader and Guennebaud [2018] transports a uniform distribution towards a continuous distribution stored on a 2D grid by repeatedly solving Poisson equations. Since the Poisson matrix can be prefactored once and only depends on the grid resolution, solving multiple times remains efficient. They obtain approximate transport maps between two non-uniform densities using linearized optimal transport (see next). In contrast, while our approach is much faster, our maps between the uniform distribution and a density is not optimal.

Approximations. Adding an entropic term to W_2^2 regularizes the problem, allowing for the use of fast iterative Bregman projections implemented as matrix-vector multiplications [Cuturi 2013] – an algorithm called Sinkhorn. When distributions are further discretized on a grid, these projections can be implemented via fast convolutions [Solomon et al. 2015]. Using this formulation to approximate optimal transport requires $O(1/\varepsilon^2)$ iterations, where ε characterizes the ℓ^1 approximation error in the marginals μ and ν of π [Altschuler et al. 2017]. It tends to be numerically unstable as the regularization factor is reduced, and the regularized distance of a measure with itself is non-zero [Feydy et al. 2019b]. Greenkorn [Altschuler et al. 2017] is a greedy variant in near linear time that only updates a single row or column of π when the data are unstructured, but still requires $O(1/\varepsilon^2)$ iterations. To alleviate stability issues, Sinkhorn divergences debias the entropic energy [Feydy et al. 2019b]. While it does not directly result in a single transport plan π anymore, a transport map can still be recovered using a gradient flow. A fast GPU-based implementation in the GeomLoss library roughly allows for matching $10k$ samples in $1s$ with Sinkhorn divergences or $50k$ samples in $1s$ with the Sinkhorn algorithm.

Regarding very fast solutions, the sliced transport between discrete distributions computes a series of 1-d transportation problems on projections, which each can be carried out in $O(n \log n)$ via simple sorting operations [Bonneel et al. 2015; Rabin et al. 2012] or even $O(n)$ using a cumulative distribution function [Pitie et al. 2005]. Transporting a uniform (or more generally continuous) distribution towards a discrete distribution can also be achieved with such 1-d projections [Paulin et al. 2020], and continuous-to-continuous solutions can be achieved with the help of the Radon transform [Bonneel et al. 2015]. Although these approaches scale very well in practice and generalizes to the partial transport setting [Bonneel and Coeurjolly 2019], in particular in the high-dimensional discrete setting, the use of 1D projections incurs significant errors compared to d-dimensional optimal transport.

Linearized transport. In certain applications, for computing transport plans between μ and ν, it is more convenient to rely on a pivot measure τ, computing $\pi_{\mu\to\tau}$ and $\pi_{\tau\to\nu}$ and composing both transport plans. This has arisen in the context of generalized geodesics [Ambrosio et al. 2008]. Since it requires the computations of optimal transports to a pivot measure, it does not in itself reduce computation times for the transport problem, but allows for faster computations of approximate Wasserstein barycenters or indirectly accelerate computation times when fast solvers are available for a specific pivot measure (e.g., uniform) [Nader and Guennebaud 2018]. The pivot measure has been taken as uniform [Mérigot et al. 2020; Nader and Guennebaud 2018], Gaussian [Moosmüller and Cloninger 2023] or the average of the input measures μ and ν [Seguy and Cuturi 2015; Wang et al. 2013]. The concept has been called "linear optimal transport" [Moosmüller and Cloninger 2023; Wang et al. 2013] since it allows comparing distributions using the L^2 norm of their Monge maps with respect to the pivot measure. We instead call it here "linearized" to avoid confusion with our "linear-time" complexity. Similarly to the work of Nader and Guennebaud [2018], we use linearized transport to allow the calculation of transport maps between two arbitrary measures, benefiting from our fast transport solver to a pivot uniform measure.

2.2 Rectified flows

Rectified flows were introduced in the context of generative models [Liu et al. 2022], with connections to optimal transport [Liu 2022; Liu et al. 2022]. Given coupled observations $(X_0 \sim \mu, X_1 \sim \nu)$ in $\mathbb{R}^d$ $(d \geq 1)$, the goal is to advect X_0 towards X_1 by a velocity field v, that needs to be found based on the following least-square regression problem:

$$\min_v \int_0^1 \mathbb{E}\left[\|(X_1 - X_0) - v(X_t, t)\|^2\right] dt\,, \tag{5}$$

where $X_t = tX_1 + (1-t)X_0$. Intuitively, we look for a velocity field that is as straight as possible (following the direction $X_1 - X_0$) while joining X_0 and X_1 – hence the name *rectified flow*.

Link to Optimal Transport. As shown in [Liu 2022], the coupling (Z_0, Z_1) obtained by taking $Z_0 = X_0$ and Z_1 by advecting Z_0 with the resulting velocity field reduces the transportation cost between the two corresponding distributions, compared to the original coupling (X_0, X_1), while Z_1 also follows ν. However, it does not target a specific ground distance such as $c(x, y) = \|x - y\|^2$, but reduces the transportation cost for *all* convex ground distances (in expectation).

The original approach [Liu et al. 2022] models the velocity field v with a deep neural network, and optionally iterates the rectified flow procedure on the resulting coupling to iteratively reduce the transportation cost, further straightening the velocity field.

A follow-up approach aims to minimize the transportation cost for a given ground distance (as opposed to all convex ground distances simultaneously) by introducing the ground cost in Eq. 5 [Liu 2022]. Liu et al [2022] also note that Eq. 5 has an exact minimum given by

$$v(x,t) = \mathbb{E}_{X_0\sim\mu, X_1\sim\nu}\left[X_1 - X_0 | X_t = tX_1 + (1-t)X_0 = x\right] \ . \quad (6)$$

While their algorithm does not directly benefit from this formula, we build on it in the case where μ is a uniform probability distribution, making it highly efficient for large-scale problems, as we will explain in the next Section.

3 Rectified flows involving a uniform distribution

We aim to compute the conditional expectation of Eq. 6:

$$v(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x], \quad \text{where } X_t = tX_1 + (1-t)X_0 \, .$$

when μ has a uniform density.

3.1 Problem reformulation

Substituting $X_1 - X_0 = \frac{X_1 - X_t}{1-t}$ into the expectation, we find:

$$\mathbb{E}[X_1 - X_0 \mid X_t = x] = \frac{\mathbb{E}[X_1 \mid X_t = x] - x}{1-t}. \quad (7)$$

Thus, the problem reduces to computing $\mathbb{E}[X_1 \mid X_t = x]$. We further use $X_t = tX_1 + (1-t)X_0$ to express X_0 as $X_0 = \frac{X_t - tX_1}{1-t}$. The joint density of (X_1, X_t) can be expressed in terms of (X_1, X_0). The Jacobian of the transformation $(X_1, X_t) \to (X_1, X_0)$ is:

$$J = \begin{pmatrix} 1 & 0 \\ -\frac{t}{1-t} & \frac{1}{1-t} \end{pmatrix}, \quad \det(J) = \frac{1}{1-t} \, .$$

The joint density thus becomes:

$$f_{X_1,X_t}(x_1, x) = \frac{1}{1-t} f_1(x_1) f_0\left(\frac{x - tx_1}{1-t}\right),$$

where f_0 is the density of X_0, and f_1 is the density of X_1.

Using the general formula for conditional expectations, we obtain:

$$\mathbb{E}[X_1 \mid X_t = x] = \frac{\int x_1 f_{X_1,X_t}(x_1, x)\, dx_1}{\int f_{X_1,X_t}(x_1, x)\, dx_1} \quad (8)$$

$$= \frac{\int x_1 f_1(x_1) f_0\left(\frac{x - tx_1}{1-t}\right) dx_1}{\int f_1(x_1) f_0\left(\frac{x - tx_1}{1-t}\right) dx_1}. \quad (9)$$

3.2 Simplification for uniform X_0

Our **first core idea** is to take advantage of the results obtained in the context of rectified flows in the case where the following conditions are met. First the source density μ is a uniform probability distribution on a box domain B in $\mathbb{R}^d$. Second, the target measure ν has a piecewise constant density on a d-dimensional grid.

When $X_0 \sim \text{Uniform}(B)$, the density f_0 is constant over the domain B. To simplify the conditional expectation $\mathbb{E}[X_1 \mid X_t = x]$, we define the region for X_1 such that X_0 remains within its support B:

$$B'_{x,t} = \left\{ x_1 \mid \frac{x - tx_1}{1-t} \in B \right\}. \quad (10)$$

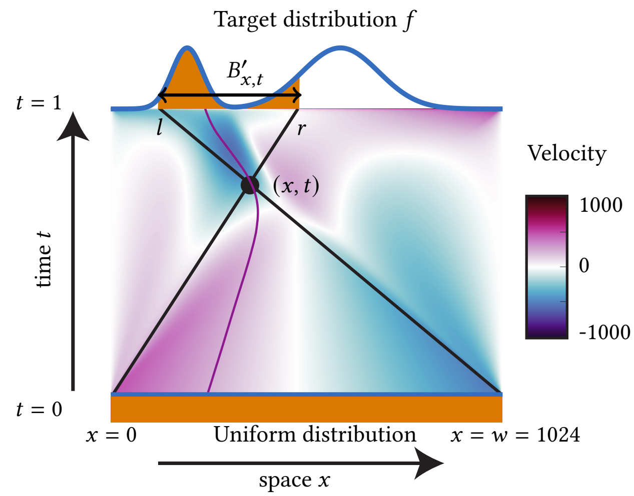

Fig. 2. 1-d example. We compute a velocity field for each space-time point (x,t) by relying on the integral of the target distribution f (here, a sum of two Gaussians) and of $x * f$ over the interval $B'_{x,t} = [\ell, r]$. These bounds are obtained by considering all straight lines passing through (x,t) and $(x', 0)$ for $x' \in [0, w]$. Here, we show a particle starting at $(x,t) = (256, 0)$ following this velocity field (purple curve). For uniformly distributed particles at time $t = 0$, this results in particles distributed according to f at $t = 1$.

Substituting into the formula for conditional expectation, we obtain:

$$\mathbb{E}[X_1 \mid X_t = x] = \frac{\int_{B'_{x,t}} x_1 f_1(x_1)\, dx_1}{\int_{B'_{x,t}} f_1(x_1)\, dx_1}. \quad (11)$$

This represents the weighted mean of X_1, where the weights are given by the target density f_1, restricted to the region $B'_{x,t}$.

Velocity field. By substituting the value of $\mathbb{E}[X_1 \mid X_t = x]$ into Equations 6 and 7, we obtain the velocity formula:

$$v(x,t) = \left(\frac{\int_{B'_{x,t}} x_1 f_1(x_1)\, dx_1}{\int_{B'_{x,t}} f_1(x_1)\, dx_1} - x \right) / (1-t) \, . \quad (12)$$

This highlights how the velocity field depends on the mean destination X_1 (conditioned on $X_t = x$) and the scaling factor $1/(1-t)$, which adjusts for the interpolation parameter t.

Integration bounds. $B'_{x,t}$ is a box defined by two extremal points $a(x,t)$ and $b(x,t) \in \mathbb{R}^d$. These are obtained by considering all lines passing through space-time point (x,t) and the uniform density's bounded support $B = [0, S_0] \times \cdots \times [0, S_{d-1}]$ at $t = 0$ (see Fig. 2 for an illustration in 1D). The line passing through space-time coordinates (x,t) and $(0,0)$ also passes through point $(x/t, 1)$. The line passing through (x,t) and $((S_0, \ldots, S_{d-1}), 0)$ also passes through point $(((S_0, \ldots, S_{d-1})(t-1) + x)/t, 1)$. We thus get the bounds for $B'_{x,t}$ as:

$$a_i(x,t) = \max\left(0, S_i - \frac{S_i - x}{t}\right),$$

$$b_i(x,t) = \min\left(S_i, \frac{x}{t}\right).$$

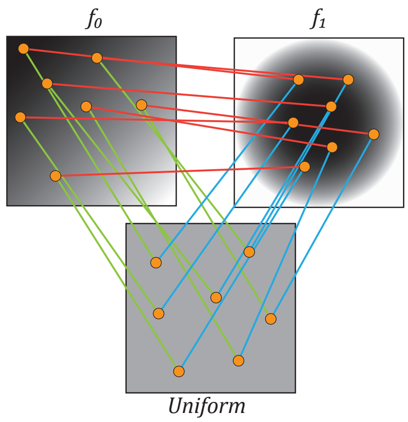

Fig. 3. We illustrate the linearized transport between distributions of two densities f_0 and f_1. We advect uniformly distributed particles towards f_0 and f_1 and use the resulting correspondences.

3.3 Fast interpolated summed-area tables

Our **second core idea** is that integrating piecewise constant functions and piecewise linear functions stored on grids over axis-aligned box-shaped domains is possible in constant time assuming that summed-area tables are precomputed in linear time (with respect to the number of grid cells). This makes our approach highly efficient for large-scale problems. A summed-area table precomputes $\sum_{i_0=0}^{k_0} \sum_{i_1=0}^{k_0} \cdots \sum_{i_{d-1}=0}^{k_{d-1}} f_{i_0,i_1,\ldots,i_{d-1}}$ in a table by reusing previously computed values (see Alg. 2).

However, during advection, particles fall within grid cells and interpolation is needed, especially when the grid resolution is coarse (see Fig. 4). While integrating piecewise constant density f over non-integer bounds simply amounts to accessing the summed-area table using standard bilinear interpolation, integrating piecewise linear functions of the form $xf(x)$ requires quadratic interpolation (see Alg. 1).

Advection. Our algorithm starts with a set of uniformly distributed particles at time $t = 0$ and progressively advects them with velocity field $v(x, t)$ until $t = 1$. To illustrate transport plans, we use particles regularly located on a grid, while for stippling applications we use blue noise point patterns. Advecting particles could be achieved most easily by the order 1 Euler integration scheme: $x_{t+1} = x_t + \mathrm{d}t \; v(x_t, t)$ where $\mathrm{d}t = 1/T$ and T is the number of time steps. In practice, we achieve higher accuracy with a 4th order Runge-Kutta integration scheme (RK4), fully detailed in Alg. 2.

Algorithm. A significant benefit of our approach is the conciseness of the resulting algorithm – comparable to sliced or entropy-regularized transportation.

We provide in Alg. 2 the computation of all summed-area table computations and particle advection in 2D. These methods are trivially extendable to higher dimensions though storing dense higher-dimensional voxels becomes impractical even for moderate dimensions. We provide in Alg. 1 the computation of the velocity field for a particle at position x and time t in 2D. The quadratic interpolation formulas become more involved and computationally costly as the dimension increases.

Algorithm 1 Velocity(E^π, E^x, E^y, I, X, t)

Require: $X = (x, y) \in [0,1]^2$ input point, $t \in [0,1]$ time

Require: E^π, E^x, E^y summed area tables and image. I of size $w \times h$; E^π, E^x, E^y of size $(w+1) \times (h+1)$

▷ *Interpolation helpers ; $\{x\} = frac(x)$, $\lfloor x \rfloor = floor(x)$* ◁

function $interp_x(E^x, I, \mathrm{x}, \mathrm{y})$

  return $E^x_{\lfloor x \rfloor, \lfloor y \rfloor} + \left(E^x_{\lfloor x \rfloor, \lfloor y \rfloor+1} - E^x_{\lfloor x \rfloor, \lfloor y \rfloor}\right)\{y\} + \left(\left(E^x_{\lfloor x \rfloor+1, \lfloor y \rfloor} - E^x_{\lfloor x \rfloor, \lfloor y \rfloor}\right)\{x\} + I_{\lfloor x \rfloor, \lfloor y \rfloor}\{x\}\{y\}\right)\frac{x+\lfloor x \rfloor}{2}$

function $interp_y(E^y, I, \mathrm{x}, \mathrm{y})$

  return $E^y_{\lfloor x \rfloor, \lfloor y \rfloor} + \left(E^y_{\lfloor x \rfloor+1, \lfloor y \rfloor} - E^y_{\lfloor x \rfloor, \lfloor y \rfloor}\right)\{x\} + \left(\left(E^y_{\lfloor x \rfloor, \lfloor y \rfloor+1} - E^y_{\lfloor x \rfloor, \lfloor y \rfloor}\right)\{y\} + I_{\lfloor x \rfloor, \lfloor y \rfloor}\{x\}\{y\}\right)\frac{y+\lfloor y \rfloor}{2}$

▷ *Determine integral bounds (ℓ, t) and (r, b)* ◁

$\ell \leftarrow \max(0, w - (w - x)/t)$

$r \leftarrow \min(w, x/t)$

$u \leftarrow \max(0, h - (h - y)/t)$

$b \leftarrow \min(h, y/t)$

▷ *Look up summed area tables and interpolate; Lerp is standard bilinear interpolation* ◁

$\mathcal{I}_\pi \leftarrow Lerp(E^\pi, r, b) - Lerp(E^\pi, \ell, u) + Lerp(E^\pi, \ell, b) + Lerp(E^\pi, r, u)$

$\mathcal{I}_x \leftarrow interp_x(E^x, r, b) - interp_x(E^x, \ell, u) + interp_x(E^x, \ell, b) + interp_x(E^x, r, u)$

$\mathcal{I}_y \leftarrow interp_y(E^y, r, b) - interp_y(E^y, \ell, u) + interp_y(E^y, \ell, b) + interp_y(E^y, r, u)$

return $((\mathcal{I}_x, \mathcal{I}_y)/\mathcal{I}_\pi - X)/(1 - t)$

Algorithm 2 Full 2D transport pseudo-code

Require: $X = \left\{X_i \in [0,1]^2\right\}_{i=1..n}$ uniform input points

Require: $I = \left\{I_{ij}\right\}_{i=0..h-1, j=0..w-1}$ grayscale input $w \times h$ image

Require: T number of times steps

Ensure: upon terminating, X follows the image density

▷ *Precompute summed-area tables* ◁

$(E^\pi, E^x, E^y) \leftarrow 0_{(w+1)\times(h+1)}$

for $(i, j) \in \{0, \ldots, h-1\} \times \{0, \ldots, w-1\}$ **do**

  $E^\pi_{i+1,j+1} \leftarrow E^\pi_{i+1,j} + E^\pi_{i,j+1} - E^\pi_{i,j} + I_{i,j}$

  $E^x_{i+1,j+1} \leftarrow E^x_{i+1,j} + E^x_{i,j+1} - E^x_{i,j} + I_{i,j}\,(j + \frac{1}{2})$

  $E^y_{i+1,j+1} \leftarrow E^y_{i+1,j} + E^y_{i,j+1} - E^y_{i,j} + I_{i,j}\,(i + \frac{1}{2})$

▷ *Advect particles with RK4* ◁

for $i \in \{0, \ldots, n\}$ **do**

  for $t \in \{0, \ldots, T\}$ **do**

    $V_1 \leftarrow Velocity(E^\pi, E^x, E^y, I, X_i, t/T)$

    $V_2 \leftarrow Velocity(E^\pi, E^x, E^y, I, X_i + \frac{V_1}{2T}, (t + \frac{1}{2})/T)$

    $V_3 \leftarrow Velocity(E^\pi, E^x, E^y, I, X_i + \frac{V_2}{2T}, (t + \frac{1}{2})/T)$

    $V_4 \leftarrow Velocity(E^\pi, E^x, E^y, I, X_i + \frac{V_3}{T}, (t + 1)/T)$

    $X_i \leftarrow X_i + (V_1 + 2\,(V_2 + V_3) + V_4)\,\frac{1}{6\,T}$

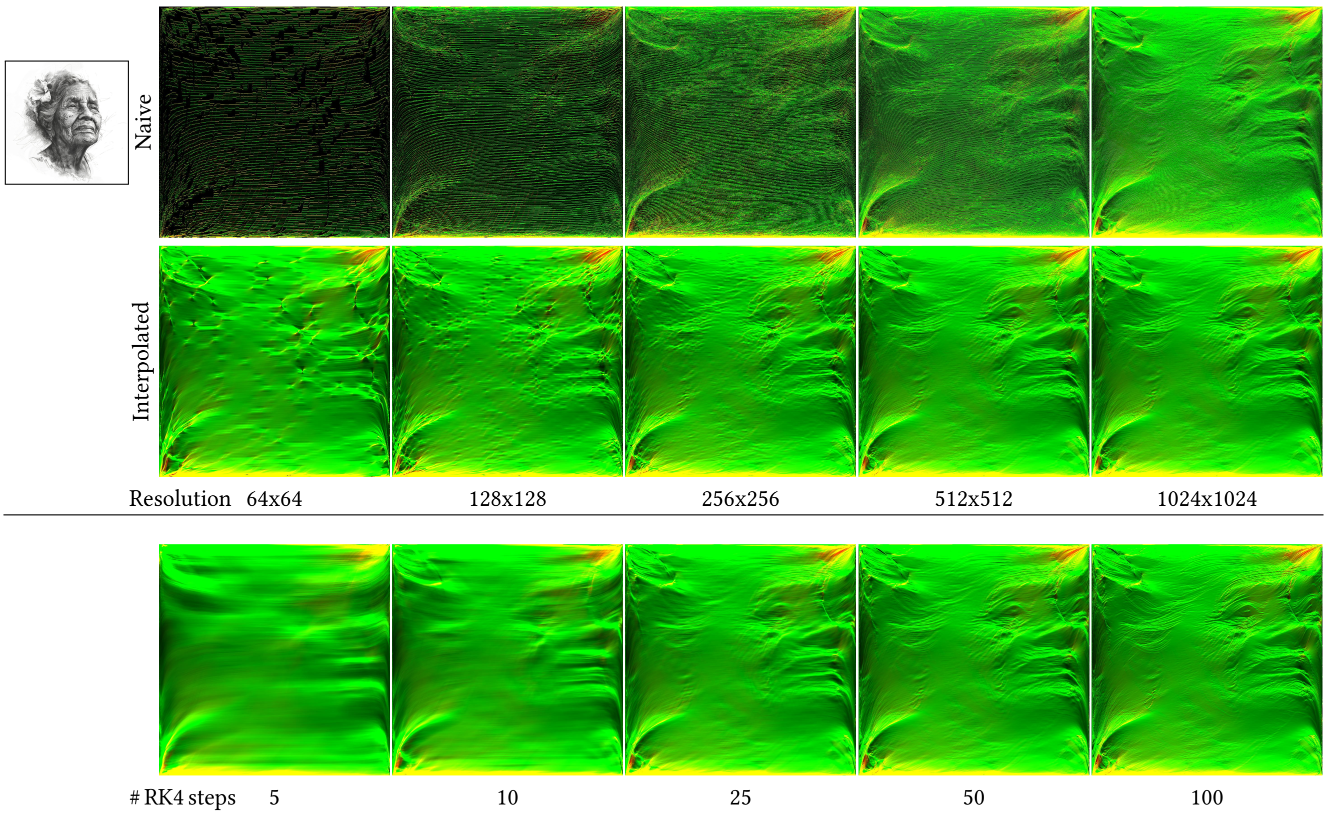

Fig. 4. Top. We assess the impact of our interpolation vs. naive access to the summed-area tables depending on grid resolution. Here, we compute the transport map using 1024^2 particles on a grid, but we vary the resolution of the underlying image and summed-area tables. We only plot $\Delta\psi$ (see Fig. 6). Naive access to the summed-area tables (first row) leads to many NaN values at low resolution, displayed as black pixels. Bottom. We assess the impact of the number of RK4 time steps on a 1024x1024 grid in the transport map ($\Delta\psi$).

3.4 Non-uniform f_0

To support the case of transporting a non-uniform distribution with density f_0 towards another (non-uniform) distribution of density f_1, we benefit from linearized optimal transport ideas. Given a set of particles Y_0 following a uniform distribution, we advect them towards f_0 and f_1 independently, and the two sets of advected particles are in one-to-one correspondence (see Fig. 3). This correspondence can be used to produce (approximate) displacement interpolation, or be further generalized to (approximate) Wasserstein barycenters when more than two densities are involved.

4 Numerical Evaluation

4.1 Time complexity and speed

The time complexity of our entire algorithm is easily shown to be $O(m + nT2^d)$, where m is the number of pixels, n the number of samples, T the number of iterations in our advection scheme, and d the dimension. $O(m)$ is due to image-dependent preprocessing, by computing summed-area tables; $O(nT2^d)$ is due to the advection itself, which requires T iterations, each accessing 2^d values for interpolating. In practice, we fix T to $T = 50$ (see Fig. 7 and Sec. 4.3).

Our timings are obtained using an AMD Ryzen Threadripper 2990WX with 32 cores at 3 GHz, from 2018[1] and an NVIDIA GeForce RTX 2080. Unless stated otherwise, timings refer to our CPU implementation. Our implementations are compared to CPU and GPU implementations of other methods (see Sec. 4.2). Our GPU CUDA implementation was readily produced by ChatGPT o4-mini from our CPU C++ implementation, and resulted in about 4x speedup. On the GPU, summed-area tables were parallelized per row and columns, successively performing horizontal and then vertical scans.

4.2 Comparison to optimal transport

Semi-discrete optimal transport. When the target measure is discrete (i.e., a sum of Diracs), optimal transport results in a decomposition of the domain into connected cells, where each cell sends mass to its associated target Dirac. When the ground distance is the squared Euclidean distance between two points, these cells are convex polytopes and form a power diagram. More complex structures arise from other costs.

[1]This desktop machine has relatively poor performances; we obtained twice faster results in the 3D interpolation experiment using an Apple laptop with M2 8-core processor.

Since our transport is not optimal specifically for a given ground distance but reduces all transport costs for all convex ground distances, we numerically assess the cells produced when transporting a uniform measure to a discrete measure.

We compare the shape of our cells with that of optimal transport using various ground distances in Fig. 5 for 1024 uniformly random 2D points and 1024 points sampled on a disk. In practice, measures were discretized on a 1024x1024 pixel grid and thus, do not exactly consist of Diracs. Ground truth optimal transport results were obtained using the network simplex of Bonneel et al. [2011] with this discretization. Our cells were obtained by uniformly sampling 256x256 particles inside each of these 1024 random pixels, and advecting them backwards to the uniform measure with our rectified flow. For this experiment only, we used a very large value $T = 3000$ of RK4 steps to assess the quality of the model rather than the quality of its numerical approximation. In supplementary materials, we provide more extensive comparisons to optimal transport W_p^q with various p and q.

Continuous W_2^2 transport. In the continuous 2D grid setting, the optimal transport map to a uniform distribution, for the quadratic cost, can be computed in acceptable time using the Instant OT approach of Nader and Guennebaud [2018]. This can provide a point of comparison for our nonoptimal solution. Their approach produces the optimal transport map T using the gradient of a convex potential: $T(x) = x + \nabla\psi(x)$ by iteratively solving Poisson equations. We can also produce a transport map by distributing particles on a grid and advecting them with our approach. In Fig. 6, we show $\nabla\psi = T - \mathrm{Id}$ and $\Delta\psi = \nabla.T$ for their approach and ours on two examples. These two examples were chosen because the Lion image has a relatively smooth density covering the entire image plane (higher density of ink is darker) that should result in a relatively smooth transport map, and the Still Life image shows a centered subject with a 0 density outside and higher contrast, resulting in strong discontinuities in the transport map. Our results show similar values for both $\nabla\psi$ and $\Delta\psi$ compared to Instant OT. For the Lion image (rows 1-2), the relative error in ℓ^2 norm for the transport map T is 1.5%, it is 14.6% for $\nabla\psi$, and 21.8% for $\Delta\psi$. For the Still Life image (rows 3-4), the relative error in ℓ^2 norm for the transport map T is 2.6%, it is 13.2% for $\nabla\psi$, though the error is 125% for $\Delta\psi$ due to highly penalized localized important errors near the discontinuities of T.

4.3 Ablation and hyper-parameters

We proposed a summed-area table interpolation, which we assess in Fig. 4. While in 2D, directly accessing the summed-area tables to the nearest value with no interpolation is approximately twice faster, it also produces NaN values, in particular at coarse resolution, when $B'_{x,t}$ becomes smaller than one pixel. When $B'_{x,t}$ is smaller than one pixel, the estimated integral from the alternating sum of nearest values in the summed-area table results becomes zero, which degenerates the denominator of Eq. 11. In Fig. 7, we evaluate the impact of the number of RK4 iterations on a stippling application (Sec. 5.1). We settle with $T = 50$, which offers a reasonable balance between speed and quality. Fig. 8 shows the impact of the number of samples and grid resolution on stippling quality and speed. It is easy to demonstrate that the magnitude of the velocity field is always bounded by the domain size. First, in Eq. 11, we obtain that $a_i(x,t) \le \mathbb{E}[X_1 \mid X_t = x] \le b_i(x,t)$ as $\mathbb{E}[X_1 \mid X_t = x]$ is obtained as a convex combination of points in $B'_{x,t}$. By substituting a_i and b_i with their expression, and using the fact that $\min(\frac{a}{b}, \frac{c}{d}) \le \frac{a+b}{c+d}$, we get $\min\left(S_i, \frac{x}{t}\right) \le S_i$ and thus the desired velocity bound. The velocity field is also Lipschitz continuous when the density and time parameters are positive, which ensures convergence [Ascher and Petzold 1998] (p.83).

5 Applications

The scope of our framework's applications is nearly as extensive as that of optimal transport. We discuss four key applications: Stippling, 2D Wasserstein barycenters, 3D shape interpolation, and Area-preserving parametrization, as partially illustrated in Figure 1.

5.1 Stippling

Direct stippling. Generating point distributions is motivated by diverse applications in Computer Graphics, such as halftoning, art stippling, and economical imprinting. A simple and efficient way to stipple grayscale images consists in using our time-continuous process to map an initial uniform blue-noise point set [de Goes et al. 2012] to the target distribution computed from the input image.

We offer a comparison to recent methods that target blue noise stippling or that use optimal transport in Fig. 1 and 9, using 8,192 samples in a 1024x1024 image – we refer to the survey of Martin et al. [Martín et al. 2017] for more general approaches. Probably the best results for stippling are obtained by Gaussian Blue Noise (GBN) [Ahmed et al. 2022] and BNOT [de Goes et al. 2012]. On this example, BNOT takes 173 seconds while GBN takes 152 seconds. We show that these iterative methods benefit from being initialized with our result to accelerate them (see next).

The scalable SOT approach of Salaün et al. [2022] performs well, but the result took 395 seconds to produce with their available parallel CPU implementation. Our method, that took 0.042s to run on the CPU or 0.012s on the GPU, has arguably the same quality as Instant OT [Nader and Guennebaud 2018] that took 26 seconds. Instant OT also advects an input point set, and we used the same uniform precomputed BNOT sampling pattern as input to both our and their methods. Instant OT's time includes an initialization step that took 23.5 seconds that only depends on the image resolution and not the image content, a solving step that took 1.5 seconds that depends on the image content but not the stippling pattern, and a map inversion that took 0.93 seconds that effectively advects samples. Other approaches produced arguably lower quality results, including Sliced OT [Paulin et al. 2020] (16 seconds, including 8 seconds for precomputing a Radon transform of the input image, and 8 seconds to optimize stipples) and Polyhex [Wachtel et al. 2014] (0.31s). Note that while Polyhex is the fastest competing approach, it also comes with 1.42 GB of precomputed lookup tables, that, in practice, took about 10 seconds to load on our machine before generating samples. In Table 1, we summarize all timings for our direct stippling experiments. Additional stippling results and comparison with Penrose tilling [Ostromoukhov et al. 2004] and Wang tiles [Kopf et al. 2006] are provided in supplementary material.

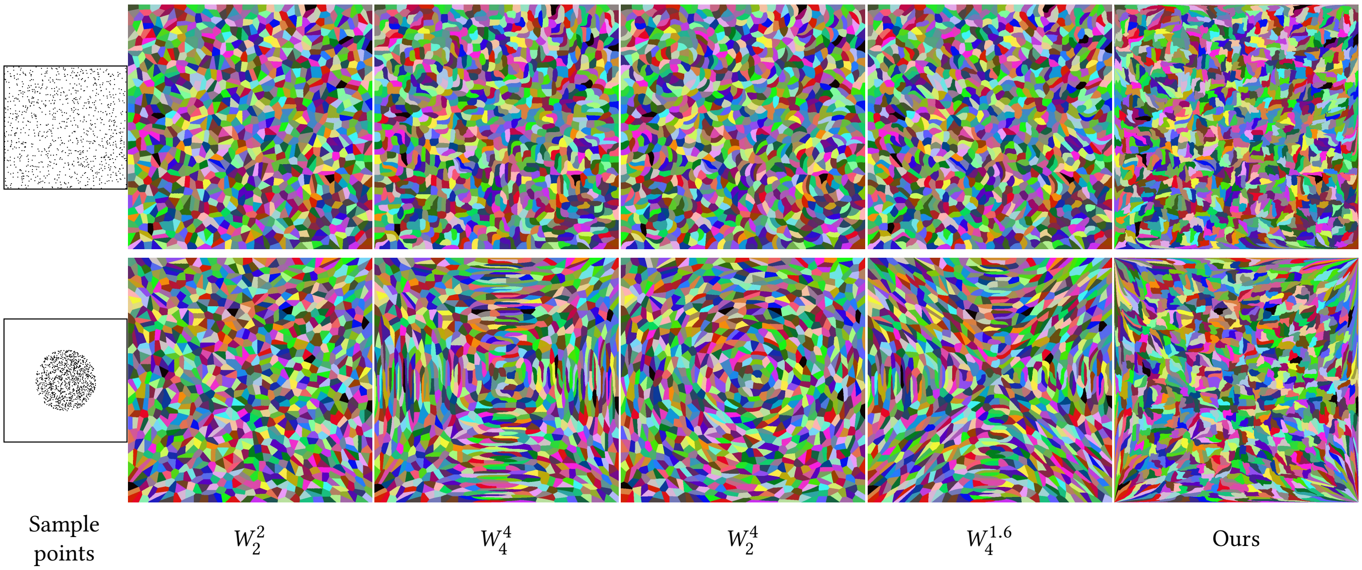

Fig. 5. Optimally transporting a uniform measure towards a sum of 1024 Diracs (left) results in a power diagram when the ground distance is quadratic ("W_2^2") or more complex, possibly non-convex, cells in other cases (here, computed with W_4^4, W_2^4, $W_4^{1.6}$ using linear programming). Our transport reduces the transport cost for all convex ground distances and results in non-convex cells of equal areas ("Ours"). We illustrate on uniformly drawn sample points (top row) and points sampled on a disk (bottom row).

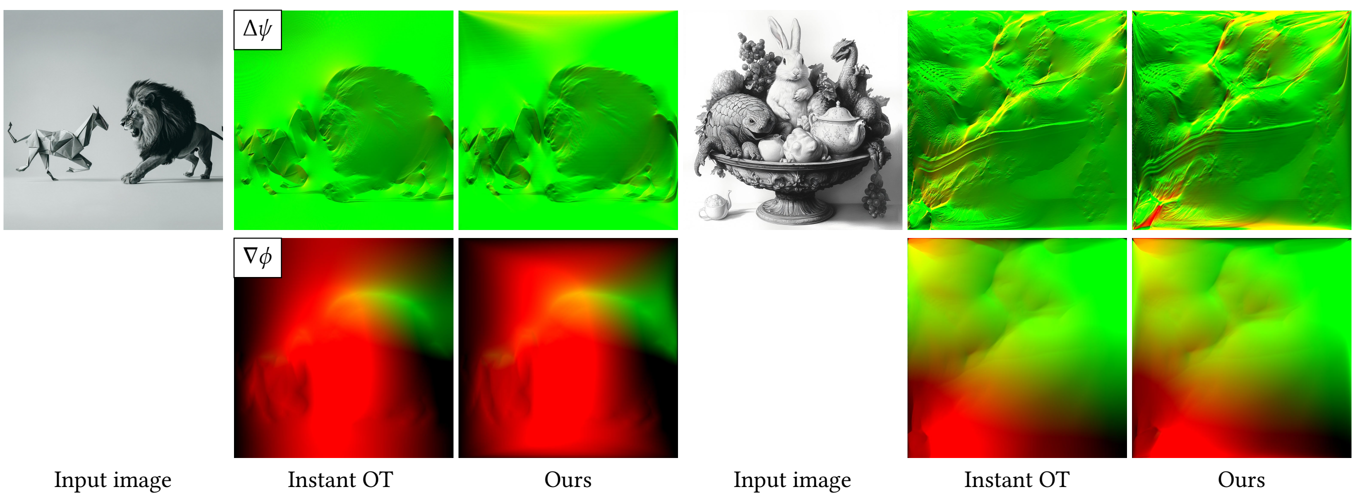

Fig. 6. The Instant OT approach of Nader and Guennebaud [2018] computes the optimal transport map T as $T(x) = x + \nabla\psi(x)$ by solving Poisson equations iteratively. We compare their optimal $\nabla\phi = T - \text{Id}$ and $\Delta\psi$ to our result that does not claim optimality for the quadratic cost (nor any specific cost). Here, darker input image values represent higher (ink) density. Maps are encoded with their (x, y) coordinates as red and green values.

Accelerating existing methods with fast initialization. Methods producing the highest quality stippling also tend to be the slowest. In our tests, BNOT [de Goes et al. 2012] and GBN [Ahmed et al. 2022] performed best. These methods are iterative and can benefit from a good initialization to accelerate convergence. We show that initializing these methods with our realtime stippling allows to recover the same quality at significant speedup. Figure 10 illustrates convergence results for GBN and BNOT, initialized with uniform random sampling and the result of our method. For BNOT, using our result as initialization, we achieve in a few seconds a convergence comparable to the result obtained in minutes using uniform random sampling as initialization.

Realtime stipple editing. Our linear-time method enables real-time editing, significantly improving usability for technical and artistic applications over similar tools that require minutes, such as Patternshop [Huang et al. 2023]. For this application, we use a uniform point set generated by LDBN [Ahmed et al. 2016] that gives qualitatively

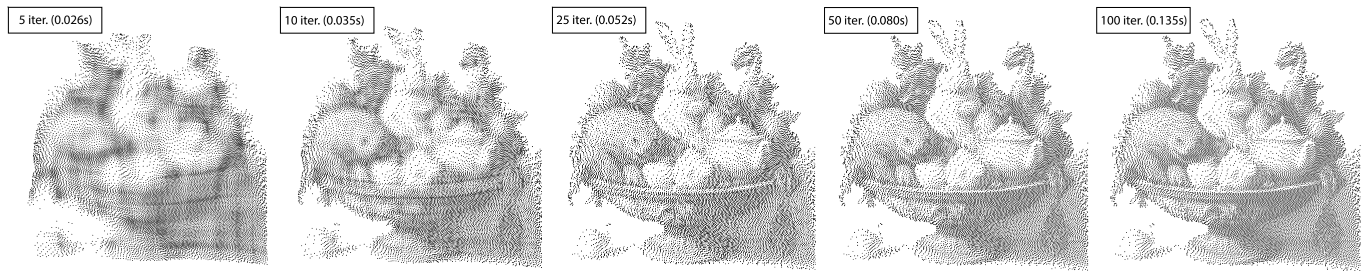

Fig. 7. Using the RK4 integrator, we show the iterations of the advected samples, here, on a 1024x1024 image with 16k samples. We settle for 50 iterations. Similar results can be obtained with more iterations of integrators with lower convergence (e.g., Euler).

Fig. 8. We vary the space discretization – sample count (first row, on a 2048 × 2048 image) and image resolution (second row, using $32k$ samples) – of our algorithm and evaluate speed and qualitative results.

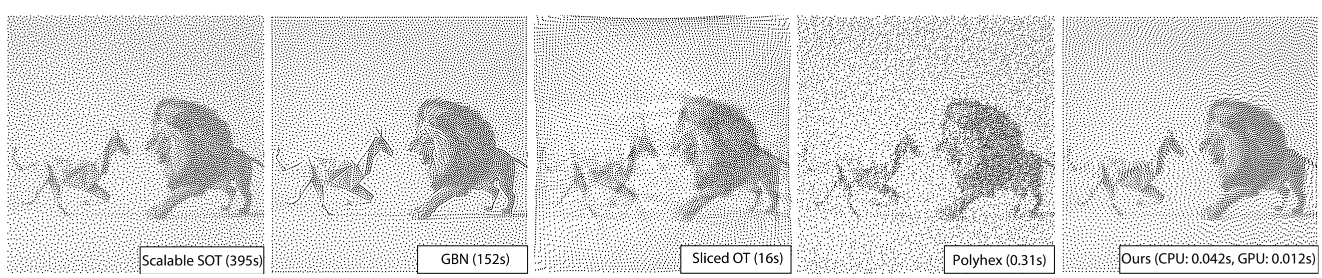

Fig. 9. From a 2D distribution and an initial uniform sampling pattern, we advect samples using our rectified flow. We compare our results with the state-of-the-art methods of Salaün et al. [2022], Gaussian Blue Noise [Ahmed et al. 2022], Sliced OT [Paulin et al. 2020] and Polyhex [Wachtel et al. 2014] (see also Fig. 1 for other comparisons).

similar results to BNOT [de Goes et al. 2012] but in realtime. Other fast point patterns can be used as input [Doignies et al. 2023]. Since our method does not incur significant precomputations, the density as well as the number of samples may be interactively adjusted – impacting the overall pattern. We showcase in supplementary

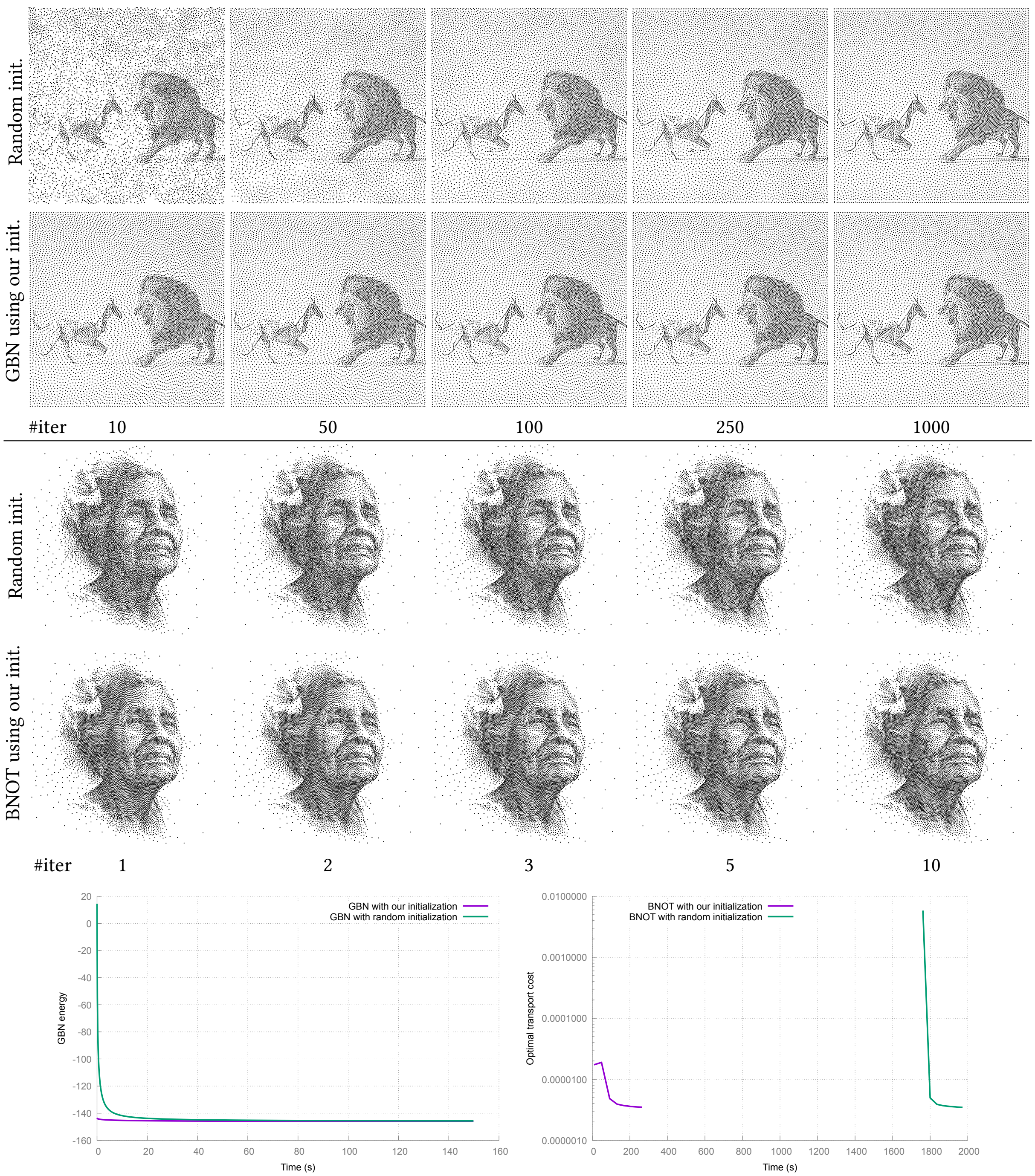

Fig. 10. We present two representative experiments to demonstrate the impact of our method as a fast initialization on the convergence of stippling methods. First, using the high-quality Gaussian Blue Noise iterative sampler [Ahmed et al. 2022] (first set of figures) and then, the BNOT sampler [de Goes et al. 2012] (second set), we compare two initialization strategies: uniform random sampling (top row in each experiment) and the result from our method (bottom row). The number of iterations of the GBN solver are indicated for each column, with our method improving convergence at negligible cost. For BNOT, we also report the number of weight optimization iterations, each of which involves several (and varying numbers of) Newton and line search steps. With our initialization, BNOT converges in 215 seconds, compared to 1972 seconds for uniform random initialization on this challenging example. Lastly, we illustrate the convergence of the GBN or BNOT energy as a function of the number of iterations and time, via two graphs. We observe that for BNOT on this example, most of the time is spent in the first iteration, while for GBN, each iteration takes the same time (*eg.* 150s for 1000 steps).

Table 1. We provide timing comparisons with other stippling methods (in seconds). Additional results and comparisons can be seen in supplementary materials.

Image	details	BNOT	GBN	Instant OT	Ours (CPU)	Ours (GPU)
	8k, 1024x1024	173	152	26	0.042	0.012
	8k, 1024x1024	1612	1205	30	0.048	0.012
	16k, 1024x1024	5653	852	56	0.080	0.018
	16k, 2048x2048	1056	2400	21	0.12	0.030

materials a video of this application, as well as a realtime webcam stippling video demo.

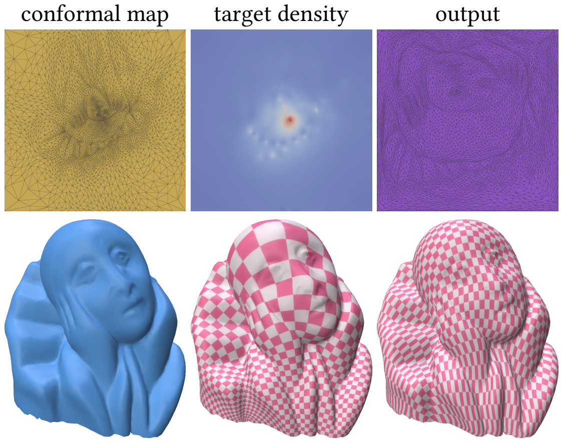

Fig. 11. Area preserving parametrization. First row: conformal parameterization, conformal factor used as density, our area-preserving result. Second row: input mesh, conformal parameterization, our area-preserving result. *Pierrot model by Frank ter Haar from the Aim@Shape repository.*

5.2 2D and 3D shape interpolation

Shape interpolation is an immediate application of our Linear-Time Transport method, enabling smooth transitions between two input shapes and producing plausible intermediate results. In 2D experiments, we compare our approach with both the instant transport method and a state-of-the-art GPU-based implementation (GeomLoss library), demonstrating that our method is two orders of magnitude faster than both, see Figure 1 top left. We demonstrate transitions from one connected component to two, along with more complex cases involving multiple holes, such as the caterpillar example featured in Fig. 1. Additionally, showcase 3D interpolations, surpassing the limitations of instant transport, which is restricted to 2D grids, see Figure 1 bottom. In 3D, the density is constructed by voxelizing 3D meshes (here, using 256^3 voxels). We then transport stratified uniformly distributed particles with one particle per voxel towards these densities. We then interpolate these particles, splat them onto the voxel grid using a small Gaussian centered at each particle, and use a marching cubes algorithm [Lorensen and Cline 1998] to reconstruct the interpolated mesh.

5.3 Linearized Wasserstein Barycenters

We further extend our approach to interpolating between more than two input shapes, akin to Wasserstein Barycenters, within our linearized transport framework. In the 2D and 3D examples of Figures 12 and 13 illustrate smooth interpolation from one connected component to two, showcasing the robustness of our approach in addressing complex topology (additional results in supplementary material).

5.4 Area preserving parametrization

Area-preservation flattening is another application of our fast computation of transport plans between densities. This method maps a 2D surface onto a flat plane while locally preserving areas. We proceed using the approach of Zhao et al. [2013] for meshes homeomorphic to a disk. We first compute a conformal map from the mesh to a square planar domain, and then transport the conformal factor towards the uniform measure. We use the transport map to advect vertices in parameter space – this results in an area-preserving map by construction, as shown in Figure 11.

6 Conclusions

Drawing on the concept of rectified flows, our approach efficiently computes transport in closed form. It has linear time complexity and is embarrassingly parallel, significantly improving speed. It works in arbitrary, but moderately low, dimensions, is trivial to implement, and is easily ported on the GPU. While it requires both a grid discretization and particles, which can be a challenge in some applications, this also makes it powerful, as it allows obtaining the transport map for a very sparse set of particles. Our framework is applicable to a range of problems, including stippling, Wasserstein barycenters, shape interpolation, and area-preserving parametrization. The simplicity, scalability, and flexibility of our method make it a powerful tool for further research in fast, transport-based context.

Acknowledgments

This work was partially funded by the ERC AdG 101054420 EYAWKAJKOS project, ANR-22-CE46-000 (StableProxies), and donations from Adobe Inc.

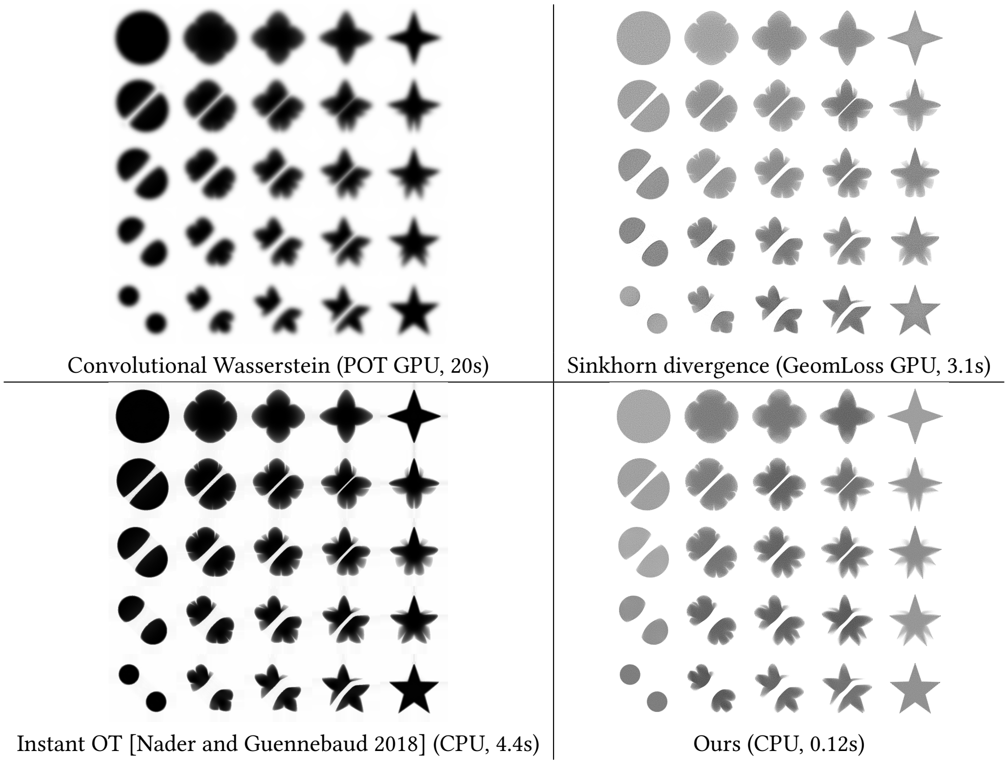

Fig. 12. We compute the interpolation of 4 shapes using: (a) direct Convolutional Wasserstein barycenter [Solomon et al. 2015] via Sinkhorn-like iterations on the GPU using POT [Flamary et al. 2021] (entropy set to 0.0019 to ensure both sharpness and numerical stability), (b) a Sinkhorn divergence iteratively minimized via GeomLoss on the GPU to debias Sinkhorn [Feydy et al. 2019b] (c), Instant OT [Nader and Guennebaud 2018] (that includes 0.4s of Laplacian prefactorization), and our method (d). Aside from Convolutional Sinkhorn that minimizes the optimal transport cost (up to an entropy-regularizing term that blurs results), all other approaches use a linearized transport approach to produce barycenters. For all methods, timings correspond to producing all 25 barycenters at 256x256 resolution (or 65,536 sample points).

References

Abdalla G.M. Ahmed, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dongming Yan, Hui Huang, and Oliver Deussen. 2016. Low-Discrepancy Blue Noise Sampling. *ACM Transactions on Graphics* 35, 6 (2016). https://doi.org/10.1145/2980179.2980218

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

Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. *Advances in neural information processing systems* 30 (2017).

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. 2008. *Gradient flows: in metric spaces and in the space of probability measures.* Springer Science & Business Media.

Uri M Ascher and Linda R Petzold. 1998. *Computer methods for ordinary differential equations and differential-algebraic equations.* SIAM.

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. *Computer Graphics Forum (Eurographics State of the Art Reports 2023)* 43, 2 (2023). https://doi.org/10.1111/cgf.14778

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.

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 Trans. Graph. (SIGGRAPH Asia)* 31 (2012). Issue 6.

Bastien Doignies, Nicolas Bonneel, David Coeurjolly, Julie Digne, Loïs Paulin, Jean-Claude Iehl, and Victor Ostromoukhov. 2023. Example-based sampling with diffusion models. In *SIGGRAPH Asia 2023 Conference Papers.* 1–11.

Ayelet Dominitz and Allen Tannenbaum. 2009. Texture mapping via optimal mass transport. *IEEE transactions on visualization and computer graphics* 16, 3 (2009), 419–433.

Jean Feydy, Pierre Roussillon, Alain Trouvé, and Pietro Gori. 2019a. 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.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. 2019b. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics.* PMLR, 2681–2690.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. 2021. Pot: Python optimal transport. *Journal of Machine Learning Research* 22, 78 (2021), 1–8.

Thomas O Gallouët and Quentin Mérigot. 2018. A Lagrangian scheme à la Brenier for the incompressible Euler equations. *Foundations of Computational Mathematics* 18, 4 (2018), 835–865.

Valentin Hartmann. 2017. A geometry-based approach for solving the transportation problem with Euclidean cost. *arXiv preprint arXiv:1706.07403* (2017).

Xingchang Huang, Tobias Ritschel, Hans-Peter Seidel, Pooran Memari, and Gurprit Singh. 2023. Patternshop: Editing Point Patterns by Image Manipulation. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Matt Jacobs and Flavien Léger. 2020. A fast approach to optimal transport: The back-and-forth method. *Numer. Math.* 146, 3 (2020), 513–544.

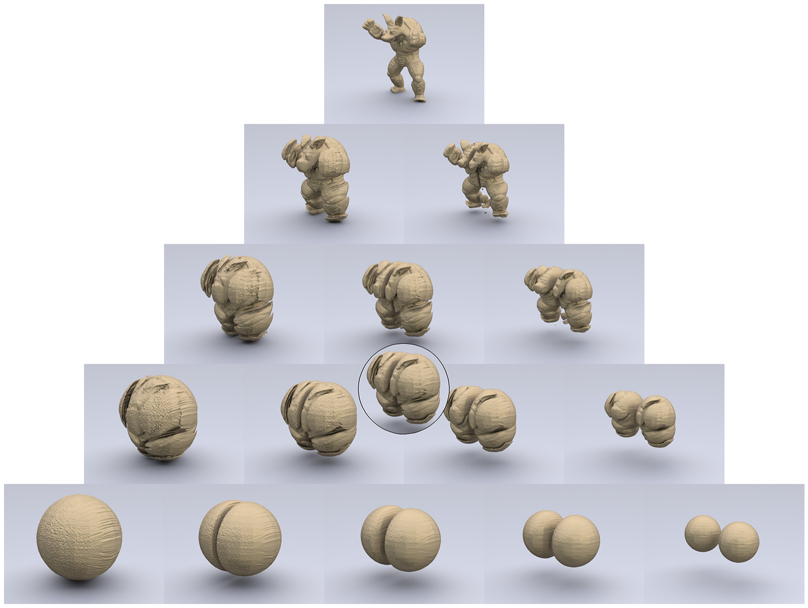

Fig. 13. We compute barycenters for voxelized 3d meshes. Computations required 1394s for 3 transport maps for 256^3 voxels, plus approx 4s per interpolation step (advecting 256^3 particles, splatting them on voxels, and marching cubes reconstruction). *Armadillo from the Stanford Computer Graphics Laboratory 3D scanning repository.*

Damian Kelly and Garrett O'Niell. 1991. The minimum cost flow problem and the network simplex method. *Master's thesis* (1991).

Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006. Recursive Wang tiles for real-time blue noise. In *ACM SIGGRAPH 2006 Papers*. 509–518.

Bruno Lévy. 2015. A Numerical Algorithm for $L\_\{2\}$ Semi-Discrete Optimal Transport in 3D. *ESAIM: Mathematical Modelling and Numerical Analysis* 49, 6 (2015), 1693–1715.

Bruno Levy. 2018. Simulating fluids with a computer: Introduction and recent advances. *arXiv preprint arXiv:1811.05636* (2018).

Bruno Lévy, Roya Mohayaee, and Sebastian von Hausegger. 2021. A fast semidiscrete optimal transport algorithm for a unique reconstruction of the early Universe. *Monthly Notices of the Royal Astronomical Society* 506, 1 (2021), 1165–1185.

Qiang Liu. 2022. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577* (2022).

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003* (2022).

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.

Domingo Martín, Germán Arroyo, Alejandro Rodríguez, and Tobias Isenberg. 2017. A survey of digital stippling. *Computers & Graphics* 67 (2017), 24–44.

Quentin Mérigot, Alex Delalande, and Frederic Chazal. 2020. Quantitative stability of optimal transport maps and linearization of the 2-Wasserstein space. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3186–3196.

Jocelyn Meyron, Quentin Mérigot, and Boris Thibert. 2018. Light in power: a general and parameter-free algorithm for caustic design. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–13.

Caroline Moosmüller and Alexander Cloninger. 2023. Linear optimal transport embedding: provable Wasserstein classification for certain rigid transformations and perturbations. *Information and Inference: A Journal of the IMA* 12, 1 (2023), 363–389.

Georges Nader and Gael Guennebaud. 2018. Instant transport maps on 2D grids. *ACM Transactions on Graphics* 37, 6 (2018), 13.

Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 488–495.

Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced optimal transport sampling. *ACM Trans. Graph.* 39, 4 (2020), 99.

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.

Francois Pitie, Anil C Kokaram, and Rozenn Dahyot. 2005. N-dimensional probability density function transfer and its application to color transfer. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. IEEE, 1434–1439.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2012. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision: Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29–June 2, 2011, Revised Selected Papers 3*. Springer, 435–446.

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable multi-class sampling via filtered sliced optimal transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 41, 6 (2022). https://doi.org/10.1145/3550454.3555484

Vivien Seguy and Marco Cuturi. 2015. Principal geodesic analysis for probability measures under the optimal transport metric. *Advances in Neural Information Processing Systems* 28 (2015).

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.

Florent Wachtel, Adrien Pilleboue, David Coeurjolly, Katherine Breeden, Gurprit Singh, Gaël Cathelin, Fernando de Goes, Mathieu Desbrun, and Victor Ostromoukhov. 2014. Fast Tile-Based Adaptive Sampling with User-Specified Fourier Spectra. *ACM Trans. Graph.* 33, 4 (2014).

Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. 2013. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision* 101 (2013), 254–269.

Xin Zhao, Zhengyu Su, Xianfeng David Gu, Arie Kaufman, Jian Sun, Jie Gao, and Feng Luo. 2013. Area-preservation mapping using optimal mass transport. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2838–2847.