Instant Transport Maps on 2D Grids

GEORGES NADER, Inria - Bordeaux University - LaBRI
GAEL GUENNEBAUD, Inria - Bordeaux University - LaBRI, France

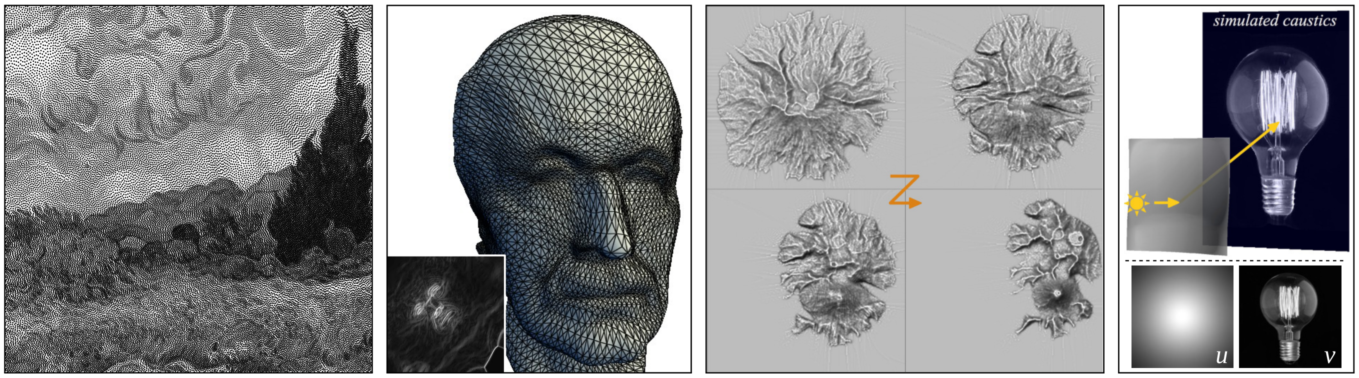


Fig. 1. Our fast mass-transport solver enables many applications such as adaptive sampling, surface remeshing, heightfield morphing and caustic design with interactive performance. From left to right: a painting of Van Gogh (*A Wheatfield with Cypresses*), Max-Planck 3D model courtesy of Max-Planck Institut für Informatik, and volcano heightmaps courtesy of University of Otago.

In this paper, we introduce a novel and extremely fast algorithm to compute continuous transport maps between 2D probability densities discretized on uniform grids. The core of our method is a novel iterative solver computing the L^2 optimal transport map from a grid to the uniform density in the 2D Euclidean plane. A transport map between arbitrary densities is then recovered through numerical inversion and composition. In this case, the resulting map is only approximately optimal, but it is continuous and density preserving. Our solver is derivative-free, and it converges in a few cheap iterations. We demonstrate interactive performance in various applications such as adaptive sampling, feature sensitive remeshing, and caustic design.

CCS Concepts: • **Computing methodologies** → *Image processing*;

Additional Key Words and Phrases: Optimal Transport, Nonlinear solver, Re-meshing, Sampling

ACM Reference Format:
Georges Nader and Gael Guennebaud. 2018. Instant Transport Maps on 2D Grids. *ACM Trans. Graph.* 37, 6, Article 249 (November 2018), 13 pages. https://doi.org/10.1145/3272127.3275091

1 INTRODUCTION

The notion of optimal transport (OT) was first suggested in 1781 by Monge as an engineering task to move a pile of sand into a hole with minimal effort. This problem has been formalized as the problem of finding a transport map T between two probability measures μ and ν that minimizes a certain cost c [Kantorovich 1942]. Since then, optimal transport has appeared in various form in numerous application domains [Villani 2008]. In the context of computer graphics, typical examples include color transfer [Morovic and Sun 2003], mesh generation [Delzanno et al. 2008], BRDF design [Bonneel et al. 2011], sampling [de Goes et al. 2012; Qin et al. 2017], shape morphing [Lévy 2015], inter surface mapping [Mandad et al. 2017], area preserving maps [Zhao et al. 2013], caustic design [Schwartzburg et al. 2014], centroidal convex decomposition [Xin et al. 2016], etc.

All these applications thus rely on the availability of efficient solvers of the underlying OT problem. Its choice and design heavily depend on the domain (e.g., Euclidean space versus an arbitrary manifold), the cost function, and the specificities of the application. The L^2 norm is by far the most used and well understood cost function. This choice provides welcome properties such as smoothness and unicity of the solution.

Previous Work on OT Solvers

In this context, several very different solving strategies have been developed, and we refer to recent surveys [Lévy and Schwindt 2018; Peyré and Cuturi 2018; Solomon 2018] for a more detailed overview.

The original formulation, as proposed by Kantorovich [1942], consists in first relaxing the problem of computing a bijective map $T : \Omega \to \Omega$ by rather computing a *transport plan* $\pi : \Omega \times \Omega \to \mathbb{R}$. Then, by discretizing both domains as Dirac masses, the problem boils down to a standard linear programming (LP) task. Since the number of unknowns is the square of the number of Dirac masses, such an approach has for a long time been limited to very small problems with up to a few thousands of Dirac points. More recently, through the use of continuous kernels as entropic regularizers and sparse representation of the transport plan, costly LP solver can be advantageously replaced by faster iterative scaling procedures [Cuturi 2013; Solomon et al. 2015]. Such approaches do not produce a bijective map but a more general transport plan, which might

Authors' addresses: Georges Nader, Inria - Bordeaux University - LaBRI, georges.nader@inria.fr; Gael Guennebaud, Inria - Bordeaux University - LaBRI, Talence, 33405, France, gael.guennebaud@inria.fr.

be a strength for some applications, but also a limitation for all applications seeking for a bijective map.

Another popular class of solvers computes the transport map between continuous and point-wise measures through a convex optimization of the weights of a power-diagram [Aurenhammer et al. 1998]. The convergence of such *semi-discrete* solvers can be greatly improved through multiresolution [Lévy 2015; Mérigot 2011] or Newton iterations [de Goes et al. 2012]. Similar to the previous *discrete* setting, one difficulty is that when applied to a pair of continuous source-target, one has to be discretized as Dirac masses and a continuous transport map has to be approximated using some kind of interpolation [Schwartzburg et al. 2014].

A third line of work proposes to directly compute the transport map between continuous densities. In this *continuous* setting, the OT problem boils down to solving a Monge-Ampère equation which is a nonlinear elliptic PDE. The non-linearity can be relaxed using computational fluid dynamics at the expense of an additional virtual time [Benamou et al. 2002; Papadakis et al. 2014]. More recently, a finite-difference scheme with accurate boundary condition and Newton iterations has been proposed by Benamou *et al.* [2014].

Whereas all the solvers of each category relies on very different discretizations, formulations, and numerical tools, it is surprising to see that the fastest solvers of each category exhibit about the same performance in terms of speed: processing typical images requires order of minutes. Achieving such a level of performance made several novel applications of OT possible. However, without significantly faster OT solvers, those novel tools can hardly be integrated within end user applications where the parameters of the given tool often need to be tuned in an interactive manner to explore the space of solutions. For the same reason, such tools can hardly be used as building blocks of higher level processing that would involve the repeated computation of OT maps.

Contribution

In order to achieve the 2 orders of magnitude speedup required to enable interactive applications, we first observe that for some graphics applications such as sampling, area-preserving mapping and re-meshing, one of the densities can be assumed to be uniform, thus opening the door for severe simplification and optimization opportunities. Moreover, in the more general case of a pair of non-uniform densities, we argue than many applications do not require a truly optimal solution regarding the transport cost as long as: 1) the densities are preserved and 2) that the transport map remains as smooth as the optimal L^2 norm solution. These two observations lead to the first central idea of this paper, which consists in focusing on computing as quickly as possible optimal L^2 transport maps between a non-uniform source and a uniform target. A more general density preserving and smooth map between arbitrary densities can then be obtained through numerical inversion and composition.

Under this hypothesis, we propose a very fast numerical method for computing the L^2 optimal transport map between a 2D Euclidean grid (i.e., an image) toward a uniform density. Our approach lies in the third category of solvers working in the continuous setting that has the main advantage of directly computing a transport map suitable for composition and fast solution space exploration for interpolation problems (e.g., color transfer, shape morphing, etc.). To achieve very high performance we make several technical contributions.

Firstly, we show that the resulting nonlinear Monge-Ampère equation can be advantageously decomposed to leverage a fixed Laplacian-like linear operator whose contribution dominates the equation to be solved. This decoupling will allow us to design an iterative solver based on a single prefactorizaton and very cheap iterations.

Secondly, we show how to adequately discretize our PDE and differential operators so that:

(1) The input density grid is properly interpreted as a piecewise constant function without implicit smoothing.
(2) The discrete equations are consistent with the initial energy preserving constraints.
(3) The discrete problem boils down to a problem of simultaneous nonlinear equations instead of a more classical minimization of a nonlinear least-square energy.

The first two properties will lead us to differential operators that depart from standard finite difference/element discretizations on regular grids. The third property might look insignificant at first, but as we will show it plays a fundamental role in the efficiency of our approach.

Last but not least, we propose a novel acceleration heuristic for solving nonlinear simultaneous equations that takes inspiration from nonlinear conjugate-gradient methods for nonlinear energy minimization.

Overall, each iteration of our solver requires only a few evaluations of residual vectors (about 3 or 4) only, and does not even require the computation of the Jacobian matrix nor gradient vectors. As a result each iteration is extremely fast, and as we will show our novel solver outperforms standard Newton iterations while avoiding the numerical difficulties of Newton iteration in non-convex area or in case of badly conditioned Jacobians. This also makes our approach very simple both conceptually and in terms of implementation.

As illustrated in Figure 1, we demonstrate our solver on various applications involving either one uniform density (remeshing, sampling) or a pair of non-uniform densities (shape morphing, goal based caustics). For all these applications, the solution has been computed in about 1s on a single core CPU.

In this paper we will focus on 2D uniform grids, leaving the extension of our work to 3D and arbitrary domains for future work.

2 BACKGROUND

2.1 Solving Simultaneous Nonlinear Equations

As sketched in the introduction, our discretization will eventually lead us to the problem of solving a set of nonlinear simultaneous equations of the form:

$$\mathbf{r}(\mathbf{x}) = \mathbf{0}\,, \tag{1}$$

with as many unknowns $\mathbf{x}$ as equations $\mathbf{r}$. The literature on this class of problems is much more scarce than on the optimization of nonlinear energies which receive of a lot of attention even in the computer graphics community [Kovalsky et al. 2016; Rabinovich et al. 2017].

Of course, such a problem can easily be cast as an optimization problem by considering the L^2 norm of the residual vector $\mathbf{r}$:

$$e(\mathbf{x}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x})\|^2 , \tag{2}$$

which could then be blindly minimized using a classical solver such as Newton, quasi-Newton, or nonlinear conjugate-gradient methods [Nocedal and Wright 2006]. This would, however, be a lose idea because we would loose information in the process, and the underlying squaring would make the problem much more anisotropic and thus difficult to solve [Broyden 1965].

Gauss-Newton solvers can exploit the specific structure of e, but the most classical approach to solve (1) is through Newton root finding algorithm, which is directly derived from a first-order approximation of $\mathbf{r}$ [Householder 1953]. In practice it is often coupled with a line-search algorithm leading to the following iterative algorithm:

$$\mathbf{d}_k = -\mathrm{J}_\mathbf{r}(\mathbf{x}_k)^{-1}\mathbf{r}_k \tag{3}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{4}$$

where k denotes the iteration number, $\mathbf{d}_k$ is the search direction, $\mathrm{J}_\mathbf{r}$ is the Jacobian of $\mathbf{r}$, $\mathbf{r_k}$ is a shortcut for $\mathbf{r}(\mathbf{x}_k)$, and $\alpha_k \in [0, 1]$ is found to prevent divergence in case of a poor local approximation. The line-search algorithm finding α_k requires an error function as e in equation (2). It is classically implemented using some variants of the Wolfe conditions [Nocedal and Wright 2006].

This Newton method suffers from two severe difficulties. The first one is the requirement of solving a new, non-symmetric, linear system of equations from scratch at each iteration, which is extremely costly especially with a large number of unknowns. This goes without saying that computing the Jacobian $\mathrm{J}_\mathbf{r}$ itself might be difficult and costly. The second one is that if the initial guess is not close enough to the solution, then the search direction is not necessarily a descent direction and the iterations will either stall or diverge. Even though the conditions for convergence are well known for decades [Householder 1953], modifying the Jacobian to ensure convergence remains a tricky and costly task.

The most well known alternative to this Newton method are the two quasi-Newton rules proposed by Broyden [Broyden 1965]. This approach follows the same algorithm than the above Newton iterations but using an approximation M of the inverse of the Jacobian that is updated at each iteration such that 1) it satisfies the secant equation between two successive solutions $\mathrm{M}_k(\mathbf{r}_k - \mathbf{r}_{k-1}) = \alpha_{k-1}\mathbf{d}_{k-1}$, and 2) the new approximate Jacobian (or its inverse) is the closest to the previous one. In practice, this boils down to rank-one updates, and just like the famous L-BFGS method [Nocedal and Wright 2006], this strategy can easily be implemented with a bounded number of recurrence vectors. This approach makes only use of derivative information from the past. On the one hand this a strength because no additional information from the problem at hand has to be computed, but on the other hand all the differential information at hand might be obsolete when working on high-dimensional and highly anisotropic problems.

In Sections 4.1 and 4.2 we will devise a novel approach to this class of problem, which can be understood as either a conjugate-gradient like approach for simultaneous linear equation, or a novel quasi-Newton update with only one recurrence term. Our approach exhibits much higher convergence rate with only one additional evaluation.

2.2 Optimal Transport & the Monge-Ampère equation

When working with continuous domains for both the source and target densities, the OT problem can be described as follows. Let u and v be two probability densities supported on U and V respectively, where U and V are compact subsets of $\mathbb{R}^d$. Let $\mathbb{M}$ be the set of maps transporting the source u into the target v:

$$\mathbb{M} = \{T : U \rightarrow V \; ; u = v(T)\det(\mathrm{J}_T)\} , \tag{5}$$

where J_T is the Jacobian matrix of T. In this paper, we seek for the map T that minimize the L^2 cost of transportation:

$$c(T) = \frac{1}{2}\int_U \|\mathbf{x} - T(\mathbf{x})\|^2 \, u(\mathbf{x}) \, d\mathbf{x} . \tag{6}$$

As shown by Brenier [1991], this problem is well posed and it admits a unique minimizer which is the gradient of a convex scalar potential:

$$\varphi : U \rightarrow \mathbb{R} , \qquad T(\mathbf{x}) = \nabla\varphi(\mathbf{x}) . \tag{7}$$

Plugging equation (7) within the density preserving equation (5) yields the elliptic Monge-Ampère PDE:

$$\det(\mathrm{H}_\varphi(\mathbf{x})) = \frac{u(\mathbf{x})}{v(\nabla\varphi(\mathbf{x}))} , \; \forall \mathbf{x} \in U , \tag{8}$$

where H_φ is a Hessian matrix of φ. Solving directly this equation in the general case is a challenging task, not only because it is highly nonlinear but also because we have to consider additional constraints such as the convexity of φ and the so called OT boundary condition $\nabla\varphi(U) = V$ which is also highly nonlinear. To the best of our knowledge, the most advanced numerical solver for this equation has been proposed by Benamou *et al.* [2014]. The boundary condition is approximated through a signed distance field leading to an Hamilton-Jacobi equation. Both PDE are then discretized through an adequate finite-difference scheme enforcing for convexity of the solution and Newton iterations. Despite its high accuracy, handling typical size images requires order of minutes of computations.

3 OUR SIMPLIFIED OPTIMAL TRANSPORT MODEL

In this section, we show how to derive from the general Monge-Ampère optimal transport formalism a simplified problem by introducing adequate restrictions. The non-linearity of equation 8 comes from three terms:

(1) the determinant,
(2) the quotient between u and v,
(3) the composition of v with the unknown map T.

Based on this observation, our first and major simplification consists in imposing that the target density v is constant. Without lack of generality, we can further assume that the area of the domain Y of v is unit, in which case $v = 1$, and the PDE to solve reduces to:

$$\det(\mathrm{H}_\varphi(\mathbf{x})) = u(\mathbf{x}) , \; \forall \mathbf{x} \in U , \tag{9}$$

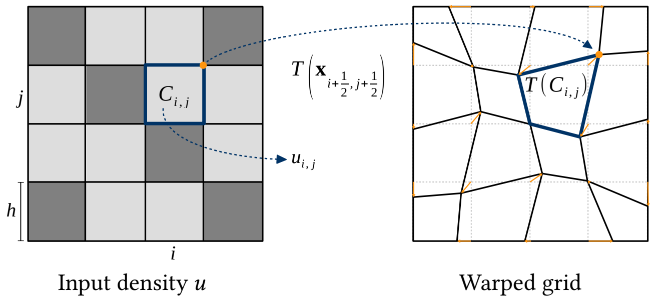


Fig. 2. In the discrete setting, the continous L^2 OT problem consists in displacing the vertices of the regular grid so that the area of each cell corresponds to the target density.

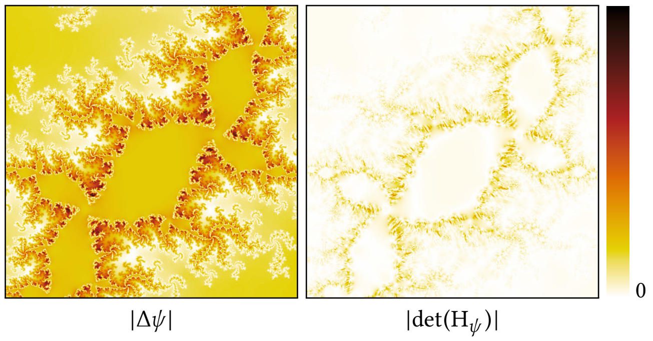


Fig. 3. Magnitude of the Laplacian and Hessian determinant of the scalar potential ψ as obtained using our L^2 optimal transport solver from the *Julia set* image to a constant density.

which is considerably simpler to solve, provided that the remaining nonlinear determinant can be efficiently handled as we will show in Section 4.

At a first glance this seems to be a severe restriction, but as we will demonstrate in the result section, many applications naturally deal with a uniform density as either the source or target. For instance, to adapt a uniform distribution to a non-uniform density u, we compute the transport map $T_{u\to 1}$ using our simplified problem and then apply its inverse $T^{-1}_{u\to 1}$ numerically (Section 6). For applications requiring a non-uniform source u and target v, we can compute a density preserving map $T_{u\to v}$ by 1) computing the maps $T_{u\to 1}$ and $T_{v\to 1}$ using our simplified problem, and 2) composing them numerically:

$$T_{u\to v} = T^{-1}_{v\to 1} \circ T_{u\to 1} \, . \tag{10}$$

Of course, this only gives an approximation of the optimal solution as the composed map $T_{u\to v}$ is generally not a minimizer of the L^2 cost function (6), but it is straightforward to verify that it does provide a valid transport map (i.e. $T_{u\to v} \in \mathbb{M}$), which we show is sufficient for many performance-demanding applications.

Our second simplification consists in imposing that both the source and target share the same support (i.e., $U = V$), and U must be a rectangle so that it admits a simple discretization through a regular grid. This restriction is naturally achieved when working on images, otherwise it needs to be enforced by padding the bounding rectangle with zeros. This choice considerably simplifies the handling of the non standard *OT boundary condition* since it reduces to linear Neumann boundary conditions [Delzanno et al. 2008; Haker et al. 2004].

More precisely, we assume that the input density u is given as a piecewise constant function defined over a 2D uniform grid consisting in n cells of size $h \times h$ with values $\mathbf{u} = (u_1, \dots, u_n)^T$. With such a discretization, it is natural to assume that T is piecewise bilinear over each cell such that the image of u by T is a warped regular grid with quadrilateral cells (Figure 2). Integrating our density preserving equation $\det(\mathrm{J}_T) = u$ over each cell leads to a system of n equations:

$$\mathrm{area}(T(C_{i,j})) - h^2 u_{i,j} = 0 \, , \quad \forall \text{ cell } C_{i,j} \, , \tag{11}$$

which simply states that the area of the warped cells must be proportional to the source density u.

4 DESIGN OF OUR EFFICIENT SOLVER

In this section we devise an efficient solving strategy for our simplified PDE eq. (9). In order to efficiently handle the remaining nonlinear term (i.e., the determinant), we propose to write the transport map as a displacement by the gradient of a different scalar potential ψ:

$$T(\mathbf{x}) = \mathbf{x} + \nabla\psi(\mathbf{x}) \, . \tag{12}$$

Such a decomposition is rather common in the literature where $-\psi$ is called the *Kantorovich potential*. It is related to the initial convex potential φ by $\varphi(\mathbf{x}) = 1/2\|\mathbf{x}\|^2 + \psi(\mathbf{x})$, and their Hessian are related by: $\mathrm{H}_\varphi = \mathrm{I} + \mathrm{H}_\psi$. When working in $\mathbb{R}^2$, plugging this identity in our simplified PDE (9) leads, after some trivial refactorizations, to the following equation:

$$\Delta\psi + \det(\mathrm{H}_\psi) + 1 = u \, . \tag{13}$$

This rewriting is one of the central ingredient of our fast solver. Indeed, the initial nonlinear determinant has now been decomposed as the sum of a linear Laplacian operator and another nonlinear determinant. Since the latter measures some kind of distortions of the displacement and that the displacement is expected to be locally smooth in most area, this expression is expected to be greatly dominated by the Laplacian term. This assumption is verified in Figure 3 on the *Julia set* example. For this rather complicated example, the integral of the magnitude of the Laplacian is about ×8 times larger than the one of the determinant.

Such an insight immediately suggests a *global/local alternate* solving approach. To this end, let us introduce a dual variable ζ for the determinants that is initialized to zero. Let the index k denotes the iteration number. Then the general idea would be to iterate through the following two steps:

(1) Solve for ψ^k assuming ζ_{k-1} is constant using the following Poisson equation:

$$\Delta\psi^k = u - \zeta_{k-1} - 1 \, . \tag{14}$$

(2) Update the determinants $\zeta_k := \det(\psi^k)$.

This strategy is appealing because the Laplacian operator can be prefactored only once so that the repeated linear problems (14) can be efficiently carried out during the iterations. However, since the problem to solve is not convex, this naive approach is unlikely to converge even after re-writing the steps as proper proximal operators. Moreover, the proximal operators associated to such steps involve significant computation overhead.

Before presenting our actual solver, let us precise the discretization of equation (13). As we will detail in Section 5, in order to yield a consistent discretization with as many unknowns as equations, we discretize the unknown function ψ on the dual grid of the input image with nodal values defined at the cell centers. Then, following the discretization made in equation (11) with one integrated equation per cell, we end up with a set of equations of the following form:

$$\boxed{\mathbf{r}(\psi) = \mathrm{L}\psi + h^2(\mathbf{q}(\psi) + 1 - \mathbf{u}) = 0}\,, \tag{15}$$

where $\mathbf{r}$ is the vector of residual, L is a sparse symmetric matrix corresponding to the linear Laplacian term, ψ is the vector of the unknowns, and $\mathbf{q}$ is the vector holding the determinants of the Hessian of ψ per cell. The actual form of L and $\mathbf{q}$ will be detailed in Section 5, whereas solving as efficiently as possible (15) is the subject of the next two subsections.

4.1 Our Fast Linearized Solver

Since we took care at building a proper system of simultaneous equations, our problem (15) could directly be solved using the Newton method and its variants described in Section 2.1. As we will show in the result section, in our case those methods are rather slow and even fails to converge for several inputs.

We overcome these issues by taking advantage of the insights we developed at the beginning of this section, while casting the naive alternate strategy into a more robust line-search based algorithm similar to Newton iterations described in Section 2.1. To this end, the idea is to use the solution of equation (15) as a target (instead of directly picking it as the next solution ψ^{k+1}), and then to seek for a *good enough* solution ψ^{k+1} between the current solution ψ^k and this target. After some algebraic manipulations, this search direction $\mathbf{d}$ is more directly obtained as the solution of:

$$\begin{aligned}\mathbf{d}_k &= -\mathrm{L}^{-1}h^2(\mathbf{q}_k + 1 - u) - \psi_k \\ &= -\mathrm{L}^{-1}(\mathrm{L}\psi_k + h^2(\mathbf{q}_k + 1 - u)) \\ &= -\mathrm{L}^{-1}\mathbf{r}_k\,. \end{aligned} \tag{16}$$

The next solution ψ^{k+1} is obtained through a line-search using equation (4) and the same error function e defined in equation (2). Line-search algorithms are classically implemented using some variants of the Wolfe conditions [Nocedal and Wright 2006]. However, those conditions involve the computation of the gradient of the error function e only to check whether the given approximate solution is acceptable or not. In our context, this is a waste of computation effort because the gradient of e is not needed for other purposes. We thus implemented an accurate line-search algorithm combining quadratic interpolations of the error function within the current search range, and a fallback to a golden-section search in case the interpolating quadratic is concave or its minimum leads to an invalid guess. We stop bracketing when the error is reduced and that the distance between two successive estimates of α is below some threshold t_α (e.g., $t_\alpha = 10^{-2}$).

Algorithm 1: Our iterative solver with conjugate directions.

```
Initialization: build and factorize the matrix $-\mathrm{L}$
$\psi^0 := \mathbf{0}$,    $\mathbf{r}_1 := \mathbf{0}$
do
    1: Solve $\widehat{\mathbf{d}}_k = -\mathrm{L}^{-1}\mathbf{r}_k$                          ▷ sec. (4.1)
    2: Improve search direction,                          ▷ sec. (4.2)
           $\mathbf{d}_k = \widehat{\mathbf{d}}_k + \beta\mathbf{d}_{k-1}$                         ▷ eq. (21)
    3: Minimize $\|\mathbf{r}(\psi^{k-1} + \alpha_k\mathbf{d}_k)\|^2$ using line-search,
       return new solution $\psi^{k+1} = \psi^k + \alpha_k\mathbf{d}_k$ and residual
       $\mathbf{r}_{k+1}$.
while $\|\mathbf{r}_{k+1}\|^2 < t_e/h^2$
```

Our solver iterations are summarized in Algorithm 1 where we can for now ignore the step 2 (i.e., $\beta = 0$) that will be described later. We stop our iterations when the norm of the residual is below a given user defined tolerance t_e that is scaled by the cell area h^2 to be agnostic from the grid resolution.

Let us emphasize that even though equation (16) resembles to Newton iterations with some arbitrary approximation of the Jacobian, it is actually stronger than that because it gives us a direction of projection towards the hyperplane of solutions defined by the Poisson equation (14).

4.2 Acceleration Through Conjugate Directions

The current version of our solver is particularly fast during the first iterations where L is a good approximation of the true Jacobian. However, for difficult inputs for which the determinants of ψ start to have a significant influence, the convergence is expected to slow down. In order to maintain a higher speed of convergence, we propose to take inspiration from nonlinear conjugate-gradient (NLCG) methods by updating the search direction with respect to the previous one as follows:

$$\mathbf{d}_k = \widehat{\mathbf{d}}_k + \beta_k\mathbf{d}_{k-1}\,, \tag{17}$$

with $\widehat{\mathbf{d}}_k = -\mathrm{L}^{-1}\mathbf{r}_k$ the initial search direction guess, and β a scalar coefficient appropriately chosen to yield a *better* search direction (Figure 4). However, NLCG methods have been developed to minimize nonlinear energies and none of the numerous heuristics that have been proposed so far makes sense in our context [Hager and Zhang 2006]. We thus designed a novel heuristic dedicated to the problem of solving simultaneous equations. The general idea consists in enforcing two successive directions to be conjugate with

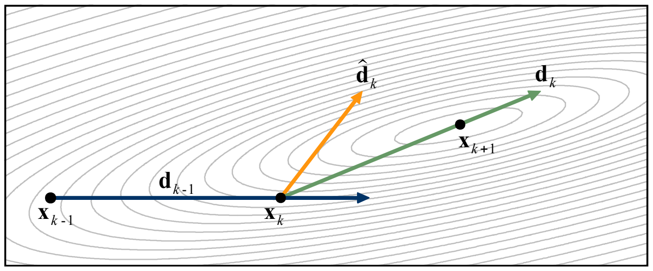


Fig. 4. Illustration of one iteration of our solver accelerated with conjugate directions.

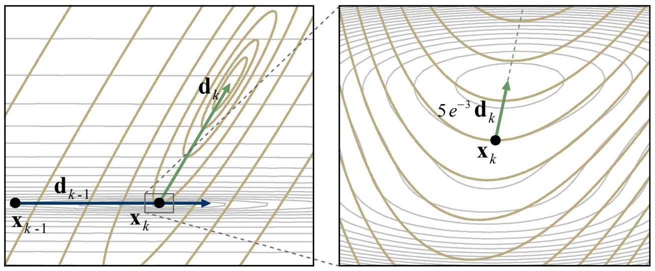


Fig. 5. Newton step for the second iteration on the *inverse Julia* problem. Iso-curves of the error e and the local quadratic approximation are shown in gray and brown respectively.

respect to the metric induced by $\mathbf{r}$. In other words, we would like that the following relation hold: $\mathbf{d}_k^T \mathrm{J}_k^T \mathrm{J}_k \mathbf{d}_{k-1} = 0$, where we used the shortcut $\mathrm{J}_k = \mathrm{J}_\mathbf{r}(\psi^k)$ for brevity. This is easily accomplished by using the following formula for β:

$$\beta = -\frac{\mathbf{d}_{k-1}^T \mathrm{J}_k^T \mathrm{J}_k \widehat{\mathbf{d}}_k}{\mathbf{d}_{k-1}^T \mathrm{J}_k^T \mathrm{J}_k \mathbf{d}_{k-1}} \,. \tag{18}$$

At a first glance this formula is not very appealing as it involves the full Jacobian matrix at the current iteration. However, let us observe that a product of the form $\mathrm{J}_k \mathbf{d}$ is nothing else but the scaled derivatives of $\mathbf{r}$ in the direction $\mathbf{d}$. We can thus efficiently compute those quantities using finite difference approximations:

$$\mathrm{J}_k \widehat{\mathbf{d}}_k \approx \frac{\mathbf{r}(\psi^k + \varepsilon \widehat{\mathbf{d}}_k) - \mathbf{r}_k}{\varepsilon} = \frac{\mathbf{r}_{k+\varepsilon} - \mathbf{r}_k}{\varepsilon} \,. \tag{19}$$

The derivatives along the previous search direction can directly be estimated for free from the previous residual as follows:

$$\mathrm{J}_k \mathbf{d}_{k-1} \approx \frac{\mathbf{r}_k - \mathbf{r}_{k-1}}{\alpha_{k-1}} \,. \tag{20}$$

This leads to the following final formula for β:

$$\boxed{\beta = -\frac{\alpha_{k-1} (\mathbf{r}_k - \mathbf{r}_{k-1})^T (\mathbf{r}_{k+\varepsilon} - \mathbf{r}_k)}{\varepsilon \|\mathbf{r}_k - \mathbf{r}_{k-1}\|^2}} \,, \tag{21}$$

which, overall, requires only one additional evaluation of the residual that is cheap to compute. According to our experiments, we found that taking $\varepsilon = \alpha_{k-1}$ to estimate $\mathrm{J}_k \mathbf{d}_{k-1}$ as in eq. (20) led to slightly faster convergence than computing a more accurate estimate, likely because of the implicit smoothing of the error function it provides. A similar observation has been made by Broyden [1965] with his quasi-Newton method. On the other hand, when estimating $\mathrm{J}_k \widehat{\mathbf{d}}_k$, since our error function e is extremely anisotropic, ε must not be chosen too large, and in practice we found that taking $\varepsilon \in [10^{-6}, 10^{-2}]$ has little impact on the convergence rate. We thus fixed it to 10^{-3} in our implementation.

Figure 4 depicts a typical step of this algorithm, which according to our experiments is almost always successful in finding the local minima in the given 2D subspace (see also supplemental material). In contrast, as depicted in Figure 5, the true Jacobian often yields very poor search directions from which only very small progress can be achieved.

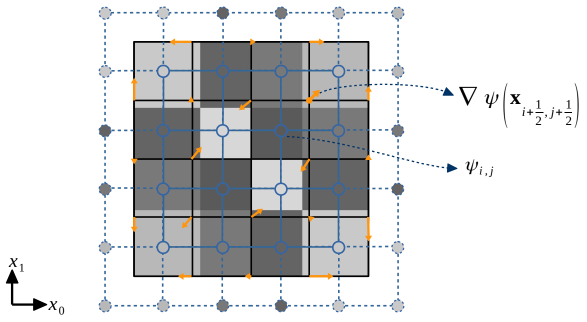

$\nabla\psi\left(\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}\right)$
$\psi_{i,j}$
x_1
x_0

Fig. 6. Illustration of our dual grid discretization. Nodal values of ψ are defined at cell centers $\mathbf{x}_{i,j}$, and gradients are computed at cell's corners $\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}$ from the 4 neighbor nodal values.

5 DISCRETIZATION

This section details how we derived our discrete equation (15) from its continuous form (13), how to treat boundaries, and some related implementation details.

Let us start by showing how to properly construct the pseudo-Laplacian matrix L and vector of Hessian determinants $\mathbf{q}$ so that eq. (15) is rigorously *equivalent* to our initial area-based density preserving eq. (11). In other words, we must make sure that the following identity holds:

$$\begin{bmatrix} \vdots \\ \mathrm{area}(T(C_{i,j})) \\ \vdots \end{bmatrix} = \mathrm{L}\psi + h^2(\mathbf{q}(\psi) + 1) \,. \tag{22}$$

With such a goal in mind, it is pretty clear that using a standard first-order finite difference/element scheme with nodes defined either on cell corners or on the dual grid to discretize our PDE (13) is doomed to fail. The main challenge to ensure a consistent discretization is to make sure that the discrete gradient and Laplacian differential operators are consistent to each other, that is, the Laplacian discretization must be based on the discrete gradient operator used to compute the displacements of the cell corners from ψ. To achieve highest performance as possible, those operators must further be based on as compact as possible stencils.

As already outlined, we achieve these goals using a finite-difference scheme based on the discretization of the potential ψ on the dual grid, that is, with one nodal value per cell center, as depicted in Figure 6.

5.1 Gradient & Laplacian

To compute the warped cell $T(C_{i,j})$ and its area we need to compute the image of each corner from the gradient of ψ:

$$T\left(\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}\right) = \mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}} + \nabla\psi\left(\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}\right) , \tag{23}$$

that we compute through averaged second-order central differences:

$$\nabla\psi\left(\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}\right) := \begin{bmatrix} \frac{\psi_{i+1,j-1}+\psi_{i+1,j}-\psi_{i,j}-\psi_{i,j-1}}{2h} \\ \frac{\psi_{i,j}+\psi_{i+1,j}-\psi_{i,j-1}-\psi_{i+1,j-1}}{2h} \end{bmatrix} . \tag{24}$$

We then define our consistent discrete Laplacian operator by composing the previous discrete gradient with an analogously defined divergence operator leading to the following 5-point stencil mask at cell i, j:

$$h^2 \Delta \psi(\mathbf{x_{i,j}}) := \frac{\psi_{i-1,j-1} + \psi_{i+1,j-1} + \psi_{i+1,j+1} + \psi_{i-1,j+1} - 4\psi_{i,j}}{2} , \tag{25}$$

which is different than the usual 5-point Laplacian stencil. Those operators are summarized below as stencil masks:

$\frac{\partial \cdot}{\partial x_0}$	$\frac{\partial \cdot}{\partial x_1}$	$h^2 \Delta \cdot$
$\frac{1}{2h}\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}$	$\frac{1}{2h}\begin{bmatrix}1 & 1\\ -1 & -1\end{bmatrix}$	$\frac{1}{2}\begin{bmatrix}1 & 0 & 1\\ 0 & -4 & 0\\ 1 & 0 & 1\end{bmatrix}$

In practice, it is easy to verify that the exact same formula for the Laplacian can be derived by extracting the linear part of the determinant of J_T at the cell center, which is nothing else than the relative signed area of the displaced cell: $\det(\mathrm{J}_T) = \mathrm{area}(T(C_{i,j}))/h^2$, thus guaranteeing a consistent discretization.

5.2 Hessian determinants

As for the Laplacian, the vector $\mathbf{q}(\psi)$ of determinants of the Hessian of ψ is equivalently obtained either as the quadratic part of $\det(\mathrm{J}_T)$ or as the signed area of the quadrilateral defined by the gradient vectors of ψ at the four respective cell corners. For each cell of index c, it as the form:

$$q_c = \det(\mathrm{J}_T(\mathbf{x}_c)) = \psi_{\bar{c}}^T \, \mathrm{Q} \, \psi_{\bar{c}} , \tag{26}$$

where Q is a 9×9 matrix and $\psi_{\bar{c}}$ denotes the 9 stacked values of ψ for the cell c and its 8 neighbors.

5.3 Boundary Conditions

Since we are working on a rectangular domain, our boundary condition $T(\delta U) = \delta U$ boils down to linear Neumann conditions:

$$\nabla \psi(\mathbf{x})^T N(\mathbf{x}) = 0 , \ \forall \mathbf{x} \in \partial U , \tag{27}$$

where N denotes the normal of the boundary ∂U of the domain U. In our discrete settings, this is easily accomplished by considering ghost nodes (Figure 6) along the boundaries with enforced mirrored values, e.g., $\psi_{-1,j} = \psi_{0,j}$. With such a boundary condition, our Poisson problem (14) needs to satisfy the *compatibility* condition $\int_U u = 1$ to be well posed, which is indeed the case.

5.4 Implementation details

Our solver requires only the gradient and Laplacian operators from which we can assemble the Laplacian matrix L and efficiently compute the residual vector $\mathbf{r}$ at each iteration through the area of the cells displaced by the the gradient of ψ (eq. 11).

The determinants of the Hessians given in equation (26) are required only to assemble the full Jacobian $\mathrm{J}_\mathbf{r}$ for Newton or Gauss-Newton iterations.

It is worth noting that the matrix L is symmetric, but negative semidefinite with the constant vector $\mathbf{1} = [1, 1, \ldots, 1]$ as singular vector. In order to enable a fast Cholesky factorization, we fix one arbitrary node to 0 prior to factorizing $-$L, and project the result in the subspace orthogonal to $\mathbf{1}$.

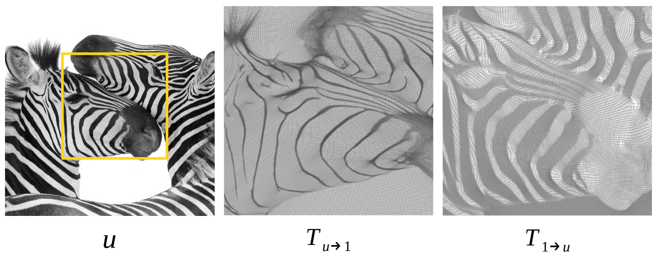


Fig. 7. Visualization of the forward map computed by our solver, and the inverse map obtained after numerical inversion of a uniform grid. Original image courtesy of Fredo Durand.

6 MAP INVERSION AND COMPOSITION

Numerical Inversion. Algorithm 1 is designed to efficiently compute the optimal map $T_{u\to 1}$ from a non-uniform distribution u towards a uniform one. This map is represented as a set of displaced bi-linear quads. To compute the inverse map $T_{1\to u}(\mathbf{y})$, we need to search the point $\mathbf{x}$ such that $T_{u\to 1}(\mathbf{x}) = \mathbf{y}$. To this end, we first search for the displaced quad containing $\mathbf{y}$ using a Bounding Volume Hierarchy (BVH), and then analytically compute its bi-linear barycentric coordinates to recover $\mathbf{x}$. Figure 7 shows the forward map $T_{u\to 1}$ computed using Algorithm 1, and the inverse map $T_{1\to u}$ obtained by numerically inverting a regular grid. As expected, after inverting a uniform grid, the resulting density is proportional to the density of the input.

Numerical Composition. This procedure can also be used to compute a transport map $T_{u\to v}(\mathbf{x})$ between a pair of non-uniform densities u and v by composing simpler maps as explained in eq. (10). To this end, we first pre-compute the two forward maps $T_{u\to 1}$ and $T_{v\to 1}$, and then perform a direct bi-linear interpolation to compute $\mathbf{y} = T_{u\to 1}(\mathbf{x})$ followed by the above inversion procedure to compute $T_{v\to 1}^{-1}(\mathbf{y})$.

Numerical Interpolation. Interpolating between two densities u and v according to a parameter α is then easily accomplished by:

$$T_{\alpha, u\to v}(\mathbf{x}) = (1-\alpha)\mathbf{x} + \alpha T_{u\to v}(\mathbf{x}) . \tag{28}$$

This map permits to compute intermediate densities when starting from the density u, but in some cases, it is desirable to compute intermediate densities from a uniform one, which is accomplished as follows:

$$T_{\alpha, 1\to uv} = T_{\alpha, u\to v} \circ T_{u\to 1}^{-1} = (1-\alpha) T_{u\to 1}^{-1} + \alpha T_{v\to 1}^{-1} . \tag{29}$$

Figure 1 includes a direct use of it to interpolate between two height-fields interpreted as densities. This equation generalizes to n-way barycenters as in Figure 12.

7 RESULTS

7.1 Raw Performance

Our tests were conducted on an Intel i7-7820X CPU at 3.6GHz with 16GB of memory. Our prototype uses *Cholmod* with nested dissection as fill-in reordering to pre-factorize the Laplacian matrix and solve the linear systems, and *Eigen* for other matrix and vector operations. In our current implementation, only the pre-factorization step is multi-threaded, while the rest runs on a single core.

Table 1. Detailed performance of our OT solver for various input densities.

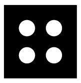
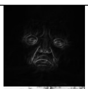
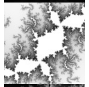
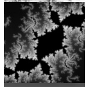
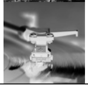
	Grid Resolution ($1/h$)	Factorization of $-L$ (s)	Optimization: Linear solve (step 1) (ms)	Optimization: Conj. Dir. (step 2) (ms)	Optimization: Line-search (step 3) (ms)	Optimization: # iterations	Optimization: **Total opt. time (s)**	$T_{u\to 1}^{-1}$ map (s)	**Total Time (s)**
	256	0.16	5.9	0.6	1.1	26	**0.20**	0.043	**0.43**
	512	0.64	25.7	2.9	5.2	25	**0.84**	0.18	**1.66**
	1024	2.66	115	13.5	27.1	25	**3.90**	0.78	**7.34**
	2048	11.2	518	68.9	105	25	**17.3**	3.60	**32.1**
	256	0.16	6.1	0.58	0.97	15	**0.11**	0.04	**0.31**
	450	0.50	20	1.8	6	4	**0.11**	0.12	**0.73**
	450	0.50	20	2.2	4	25	**0.64**	0.14	**1.38**
	512	0.65	25.9	2.6	5.7	5	**0.17**	0.16	**0.98**

Table 1 reports the computation cost of each part of our algorithm for different target difficulties and grid resolutions. The prefactorization step of the matrix L depends only on the grid resolution. When working on multiple problems having the same sizes, this step has to be done only once and can thus be neglected. In some applications, the Cholesky factors could even be cached on disk to completely save this step across multiple sessions.

The optimization itself depends on two factors: the cost of one iteration, and the number of required iterations that will be analyzed later. Each iteration mostly involves two sparse triangular solves to compute the search direction candidate $\widehat{\mathbf{d}}_k$, a few residual evaluations for the line-search, and one more to estimate β. Since triangular solves are about $\times 5$ times more expensive than computing one residual vector, it makes sense to spend a little more time in the line-search to find a pretty accurate 1D minima, hence reducing the overall number of iterations. In practice, we found that setting a tolerance $t_\alpha = 10^{-2}$ for the line-search to be a good tradeoff. The cost of one iteration ranges from 10ms to 700ms for grids ranging from about 65k to 4M cells.

The second to last column reports the time to apply the inverse map to a uniform grid having the same resolution as the input grid, including the construction of the BVH. It represents less that 10% of the overall computation time. Moreover, the inverse map can be applied to various grids and samplings without having to recompute the forward map nor to assemble the BVH.

From this table, we see that our prototype implementation is already able to maintain interactive performance for working grids up to about 512^2 cells.

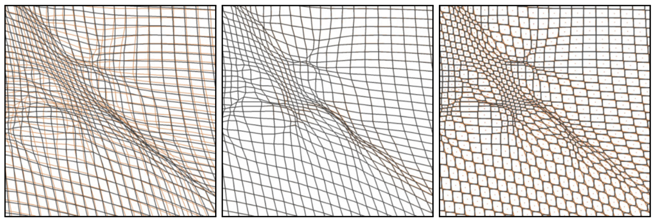


Fig. 8. Closeups of maps obtained for the *Julia* example with superimpositions of our converged map in orange (24 iterations, $e = 4 \cdot 10^{-13}$) on top of, from left-to-right: our maps obtained after 1 and 4 iterations, and the Laguerre cells obtained by a reference semi-discrete solvers (together with the Laguerre centroids and the centroids of our quads).

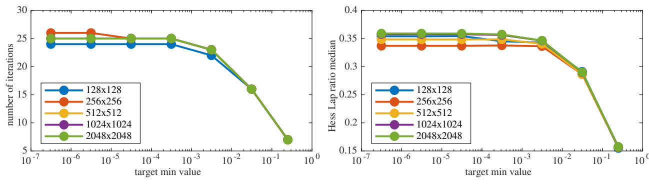


Fig. 9. Plots of the number of iterations (left) and the median of the Hessian determinant and Laplacian ratios (right) versus the minimal value of the *four disks* input density and various grid resolutions.

7.2 Convergence rate

Beside raw performance, the speed of our algorithm is also conditioned by the number of iterations, which itself depends on the stopping criterion and some notion of *difficulty* of the target density.

As explain in Algorithm 1, we stop our iteration when a certain level t_e of relative accuracy has been reached. Whereas its actual choice likely depends on the application, we found that taking $t_e = 10^{-5}$ to be a good, rather conservative, starting point as the differences of the maps become hardly visible afterwards (Figure 8). We used this setting for all the results of this paper.

As discussed in Section 4, the rate of convergence of our solver is expected to be correlated with the ratio between the magnitudes of the Laplacians and Hessian determinants. This behavior is depicted Figure 9 on the *four disks* binary example where we changed the value of the dark area to indirectly adjust this ratio. We can also observe that the working grid resolution has no impact on the convergence rate. Though this experiment might suggest that the convergence rate is somehow correlated with the contrast of the input density, this is not always the case. For instance, if the input density is constant along one axis, then our solver will find the exact solution at the first iteration regardless of the actual values. Moreover, as can be seen on the previous plots, the number of required iterations quickly stalls as the contrast increases.

Table 2 shows convergence rate for increasing grid resolution against a density with known analytic solution. On this example, our solver achieves the same accuracy as the more general and sophisticated solver of Benamou *et al.* [2014] while being orders of magnitude faster.

Table 2. Convergence rate and timings (in seconds) for increasing grid resolutions on the problem given in section 6.1 of [Benamou et al. 2014] with L^2 errors computed against the known analysis solution. This table also reports the performance of Benamou *et al.* solver and the semi-discrete solver of Geogram with the maximal and average Euclidean-distance between their Laguerre centroids and our displaced quad centers (section 7.5).

grid	Our solver		Benamou 2014		Geogram		
res.	facto.+solve	L^2 err.	time	L^2 err.	time	max	avg
32	0.006+0.001	0.0130	0.3	0.0127	0.01	2.4e-4	8.5e-5
64	0.013+0.002	0.0064	1.4	0.0064	0.06	7.0e-5	2.2e-5
128	0.041+0.009	0.0031	6.3	0.0032	0.32	1.7e-5	5.6e-6
256	0.160+0.040	0.0016	41.6	0.0016	1.62	4.5e-6	1.4e-6
362	0.320+0.086	0.0011	101.9	0.0011	4.05	2.2e-6	7.0e-7
1024	3.080+0.840	0.0004			104.82	2.8e-7	8.8e-8
2048	15.020+4.180	0.0002			1096.94	8.6e-8	2.2e-8

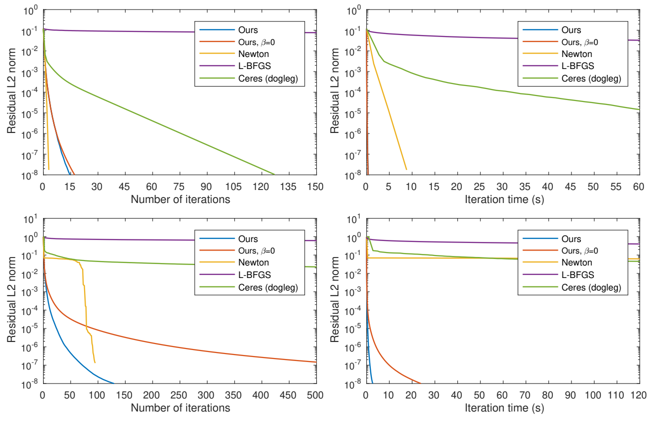


Fig. 10. Evolution of the L^2 norm residual with respect to the number of iterations and execution time for the *Julia* (top row) and *inverse Julia* (bottom row) targets (respectively, rows 3 and 4 in Table 1).

7.3 Comparison with classical nonlinear solvers

Figure 10 compares our original solver (with and without our conjugate-direction acceleration technique) against the damped Newton algorithm described in Section 2.1 combined with our line-search procedure, L-BFGS with 3 recurrence terms, and the Dogleg solver implemented in Ceres.

As can be seen, our solvers completely outperform both L-BFGS and Dogleg in terms of number of iterations, and since our iterations are much cheaper, the comparisons are even more impressive in terms of computation times. For the *Julia set* example, as expected, the Newton method performs remarkably well in terms of number of iterations, but since each Newton iteration requires the assembly of the Jacobian matrix and to solve a new linear and non-symmetric problem from scratch, each iteration is considerably more expensive and our solvers outperform Newton iterations. Moreover, for more difficult densities as the *inverse Julia* one, the true Jacobian often leads to poor search directions, as depicted in Figure 5, making the solver nearly stall until it finally find a good direction. Finally, Figure 10 also shows that for *difficult* densities, our conjugate-direction acceleration technique yields huge improvements, about ×3 for $t_e = 10^{-5}$ on this example, and more for higher accuracy.

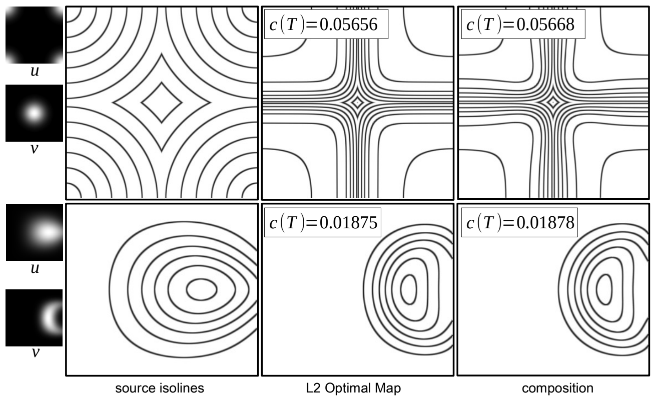
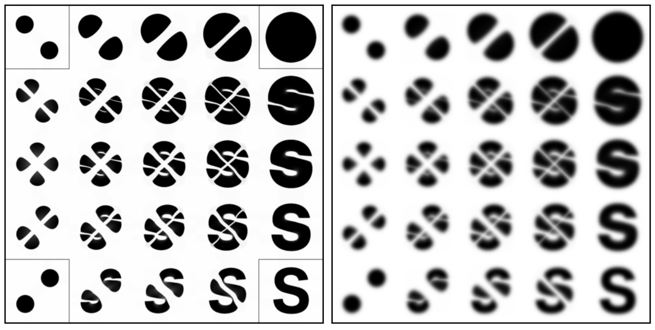


Fig. 11. Comparison of the $T_{u\to v}$ maps computed by solving the full Monge Ampère equation (eq. 8) versus the one obtained via composition (eq. 10). Iso-curves in the input domains and their images under the $T_{u\to v}$ maps, and the respective transport costs $c(T)$ are shown for two pairs of distributions. Top row: Gaussian example of Benamou *et al.* [2014]. Bottom row: a pair of anisotropic BRDF lobes.



Ours, 3.5s [Solomon et al. 2015], 1200s

Fig. 12. Bilinear barycenters from the four image corners of the left panel as inputs. Left: using our solver and a generalization of eq. (29). Right: convolutional Wasserstein barycenters [Solomon et al. 2015] (using the Matlab code provided by the authors). In both cases, transport maps/plans have computed on 299 × 299 images after inversion.

7.4 Composition & Interpolation

Figure 11 compares the transport map between pairs of non-uniform densities obtained using our solver and numerical composition, to the L^2 optimal maps. The later has been obtained by modifying our solver to directly solve the full Monge-Ampère equation. If v is non null on the domain, this is easily accomplished by replacing the right-hand-side of equation (11) by the integral of $u/(v \circ T)$ over each cell when computing the residual vector. This is of course much more expensive than computing an approximately optimal map through composition because 1) the solver has no knowledge on the variations of v, and 2) each iteration now involves costly integral computations. Our composition approximation yields very similar Wasserstein distances but tends to exhibits some undulations.

The same observation can be made in Figure 12 comparing our fast interpolation method to convolutional Wasserstein barycenters [Solomon et al. 2015]. This effect is especially visible when the "S" splits in half. On the other hand, entropic regularization yields over blurred results.

7.5 Comparison with semi-discrete solvers

Figure 8(c) compares the forward map obtained using our solver with the one obtained by the semi-discrete Newton solver implemented in Geogram [Lévy 2017]. To match our settings, we generate one weighted Dirac mass per grid center, and define the continuous target as a single quad with a uniform density. This configuration by-passes the very costly integrals over the Laguerre cells which are required in the case of a non-uniform target density v. It can be seen that our displaced quads overlap very well the Laguerre cells of the proved semi-discrete solution. This observation is confirmed in Table 2: both the maximal and average Euclidean-distances between the Laguerre centroids and our displaced quad centers are much smaller than the discretization step.

Such a result was expected since, as shown by Gu *et al.* [2016], semi-discrete solvers can be directly derived from the Monge-Ampère equation (8) through a discretization of the input density u as a set of Dirac masses. Furthermore, it is easy to show that the scalar potential ψ optimized by our solver is related to the power-diagram weights w optimized by the semi-discrete solver by: $\psi = -2w$ [Lévy 2015]. Since the discretizations are different, such an equality is only expected to be reached for an infinitely fine grid.

Semi-discrete solvers optimize a convex energy (e.g., eq. (2) in [Mérigot 2011]) whose gradient is analogue to our residual vector $\mathbf{r}$ comming from equation 11, but with the major difference that the target cells are polygonal Laguerre cells instead of simple quads. This difference justifies why the semi-discrete energy is not applicable in our case and yields to Jacobians of the residual exhibiting very different structures and properties. In the semi-discrete case, the Jacobian is symmetric positive semi-definite and similar to a *weighted Laplacian* [de Goes et al. 2012; Gu et al. 2016]. It has a standard Laplacian structure that might change during the optimization. A staggered triangulation has been suggested by de Goes *et al.* [2014] but edge flips, and thus structure changes, might still be required to reach the true optimal transport solution. In our case of a regular grid, it is nonetheless interesting to observe that edge flips can only occur within each pair of diagonals of each quad, and the edges of the Laguerre cells can only takes 4 directions $(0, \pi/4, \pi/2, -\pi/4)$. In contrast, our quadrilateral cell edges can take arbitrary orientations. In our case, the Jacobian is not symmetric but it has a fixed structure. We split this Jacobian as a fixed *Laplacian-like* matrix plus the Jacobian of a nonlinear term (which is never computed by our solver). Therefore, our Laplacian matrix should not be confused with the *weighted Laplacian* that appears in the semi-discrete settings. Even at the first iteration, for which our Laplacian is exactly our Jacobian and the Laguerre cells degenerate to the same uniform grid as us, the semi-discrete Laplacian boils down to a standard Laplacian mask whereas ours is aligned on the diagonal (section 5.1).

Finally, in terms of performance, Table 2 reveals that for comparable settings, our solver is orders of magnitude faster. We further measured that in the Geogram implementation, half of the time is spent in updating the Laguerre diagram, whereas the other half is spent in solving the underlying sparse linear problems. Of course, this high performance gain comes at a loss of generality, in particular semi-discrete solvers can work on arbitrary target domains with arbitrarily distributed Diracs as inputs.

8 APPLICATIONS

This section aims to evaluate our approach in the context of various computer-graphics applications that make a direct use of our OT solver.

8.1 2D Adaptive Sampling

The generation of random samples matching a given density is an essential ingredient in many applications such as Monté-Carlo integration or stippling. This is still an ongoing research area where the main challenge is to find the best tradeoff between speed and sampling quality. The algorithm we implemented is straightforward. Given a target density u, we pre-compute the forward map $T_{u\to 1}$ using our solver, build a BVH on the displaced quads, and then apply the inverse map $T^{-1}_{u\to 1}$ to a stream of samples uniformly distributed within the $[0, 1]^2$ domain. The cost of generating one adapted sample is thus the sum of generating the random sample itself plus the cost of one BVH search that has a $O(\log(n))$ complexity, with n the number of grid cells. For the actual generation, we implemented a simple tiling strategy of a pre-computed uniform pattern of 1024 samples, but for practical Monté-Carlo integration some recent work can stream high-quality uniform samples with very high efficiency [Perrier et al. 2018]. This procedure is analogue to the classical inverse transform sampling method based on conditional CDFs [Pharr et al. 2016].

This procedure is illustrated in Figure 13 in the context of stippling, in which case the input density corresponds to the inverse of the input image. Compared to discontinuous conditional CDFs, our L^2-optimal map is continuous and smooth (for a continuous density) and thus, it much better preserves the intrinsic quality of the input uniform distribution, thus yielding to much lower reconstruction error. Compared to a sophisticated blue noise generator such as the one of de-Goes *et al.* [2012], our result exhibits some structures in areas of strong anisotropic stretching, but our approach is orders of magnitude faster, it can handle a stream of samples, and by construction, it does converge to the target density. In applications for which the generated samples can be globally processed, we found that very few Lloyd relaxation steps (typically 3 as in fig. 13(b)) are enough to break those structures and reach a very high quality with a very small overhead. On this particular example, the Lloyd steps improve the reconstruction in sparse areas (highlights), but tends to blur the sharpest details.. As detailed in Table 3, our approach can generate 1M samples in half a second on a single CPU core, though we believe that our current BVH implementation has plenty of room for speed improvements.

Table 3. Processing time for generating density-adapted samples.

		50k	**250k**	**500k**	**1M**
ours	tile gen. (s)	0.001	0.003	0.005	0.009
	OT inversion (s)	0.051	0.170	0.266	0.436
	3 Lloyd steps (s)	0.011	0.056	0.091	0.152
	Total (s)	**0.063**	**0.229**	**0.362**	**0.597**
	CDFs (s)	0.012	0.046	0.092	0.170
	[de Goes et al. 2012] (s)	72	663	1600	-

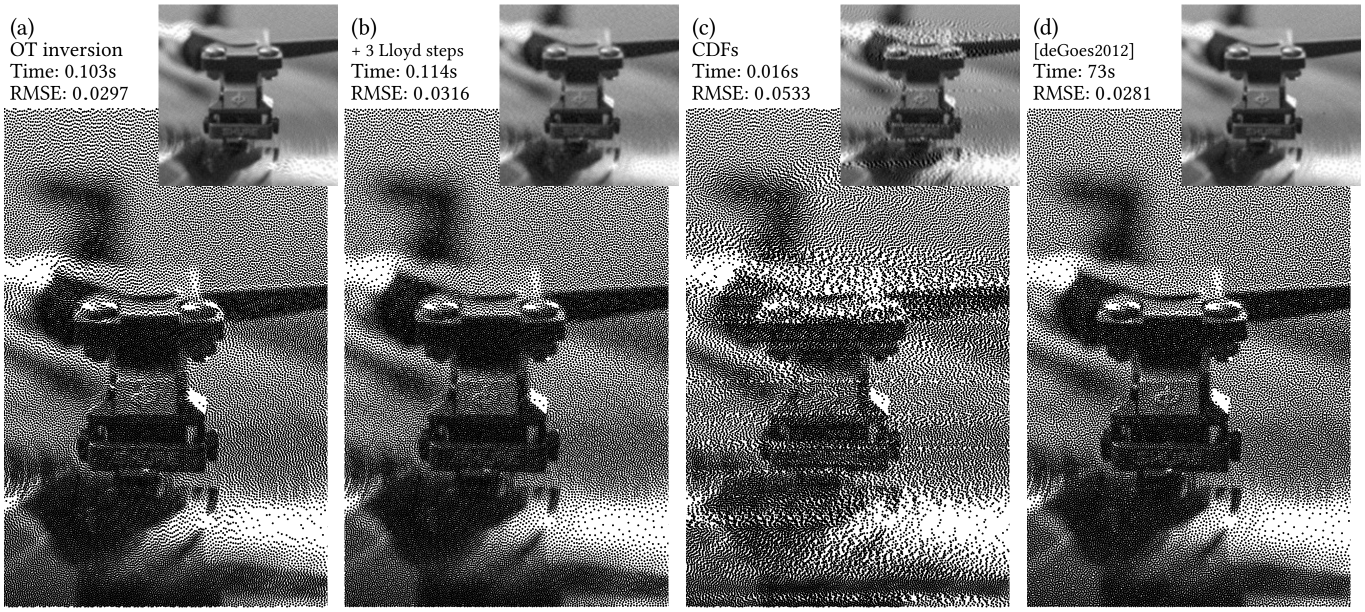


Fig. 13. (a) 50k samples generated by numerically inverting a set of tiled uniform blue noise samples with respect to the forward $T_{u\to 1}$ map. (b) 3 Lloyd steps applied to the numerical inversion output. (c) Sampling through numerical inversion of conditional CDFs of the same uniform pattern as in (a). (d) 50k blue noise samples generated with [de Goes et al. 2012]. Top-right insets show 256×256 images reconstructed through density estimation from which we have computed RMSE from the original image. Original photograph courtesy of Anton Hooijdonk.

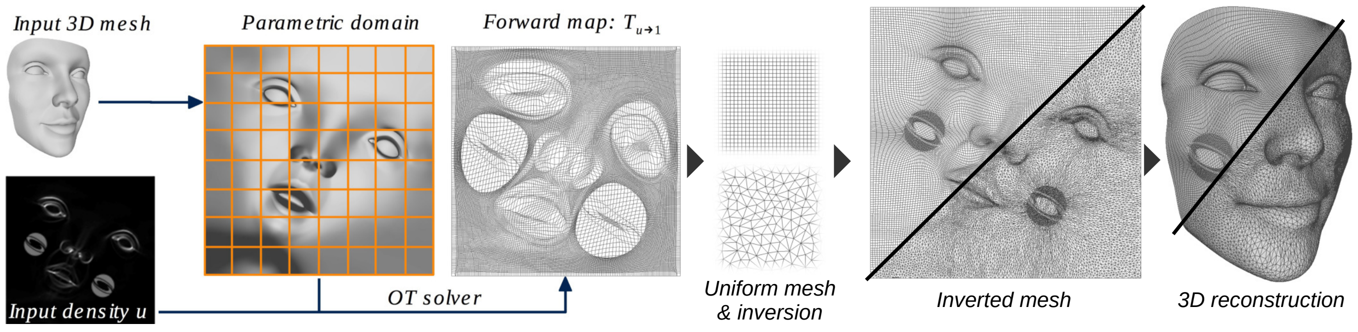


Fig. 14. Remeshing pipeline. From an input density defined on the parametric domain, we first compute the map $T_{u\to 1}$, then invert a uniform mesh to obtain the desired vertex density. The topology of the uniform mesh is preserved.

8.2 Surface remeshing with constrained connectivity

Feature adaptive surface remeshing is a well studied problem, but none of the methods proposed so far permits to constrain the connectivity of the output mesh. This is however required by some applications such as the generation of meshes suitable for hardware tessellation [Lambert et al. 2016], or to make the mesh connectivity implicit for streaming or compression purposes. To this end, our general approach consists in working in the 2D parametric space [Botsch et al. 2010], and take advantage that our transport maps are bijective (i.e., foldover free) to deform a given uniform mesh according to some density metric.

Our proposed remeshing pipeline is presented in Figure 14. Starting from a 3D mesh with a rectangular 2D parametrization, we generate a regular grid on which we define a target density u. We then compute the map $T_{u\to 1}$ using our solver, and apply its inverse to the vertices of a given uniform mesh. Finally, we recover the 3D positions of the inverted vertices through linear interpolation.

The density u should reflect surface curvatures while taking into account parametric distortions. To this end, we follow the *normal lifting* technique [Nivoliers et al. 2015] that consists in embedding the surface mesh in a 6D space composed of the 3D vertex coordinates $\mathbf{p}_i$ and scaled normal coordinates $\gamma \mathbf{n}_i$. The parameter γ controls the relative importance given to the normals. The target distribution is then obtained by computing the area of each cell of the working grid in this 6D space. The 3D position and normal coordinates of the cell corners are obtained from the corresponding surface point in parametric space trough linear interpolation.

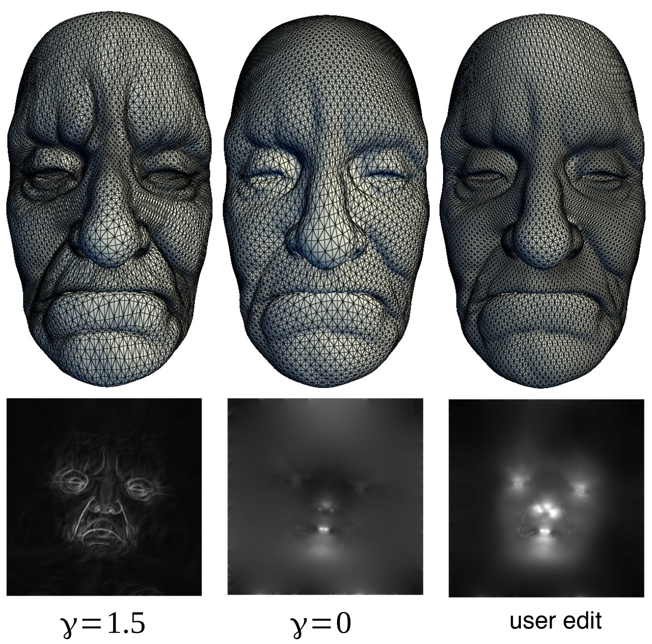


Fig. 15. Left: Feature sensitive remeshing obtained by setting the normal lifting parameter γ to 1.5. Middle: setting γ to 0 will yield in an isotropic remeshing. Right: remeshing using a user defined input density. 3D model courtesy of cvbtruong from Turbosquid.

Some results are given in Figure 15 for different values of γ, and a user edited target. Thanks to the high performance of our solver, it is possible to adjust the value γ and paint the target density with instant feedback (see the accompanying video). Finally, once the transport map has been computed we can generate several level of details instantly by simply applying the inverse map to meshes with different resolutions.

8.3 Freeform optic design

The problem of caustic design consists in finding a height-field surface producing through refraction of a given light source a desired image on a given receiver (Figure 1). The state-of-the-art approach to this problem [Schwartzburg et al. 2014] consists in first computing an OT map between the incoming light source and target densities. This gives an initial solution that is then iteratively improved until the map yields a normal field that is truly integrable. They reported that the initial step requires from 15mn to 4h for 1.5M samples. In this context, computing a strictly L^2 optimal map is again a waste of effort as the resulting normal field wont be integrable anyway. Our solver permits to cut down this computation time to the order of 1s, thus enabling interactive feedback during prototyping (see accompanying video). Results using our OT solver to establish the mapping followed by a direct Poisson integration of the resulting normal field are given in Figure 16.

9 DISCUSSIONS AND FUTURE WORK

This paper shows that computing optimal transport maps from a continuous probability density to the uniform density is a useful

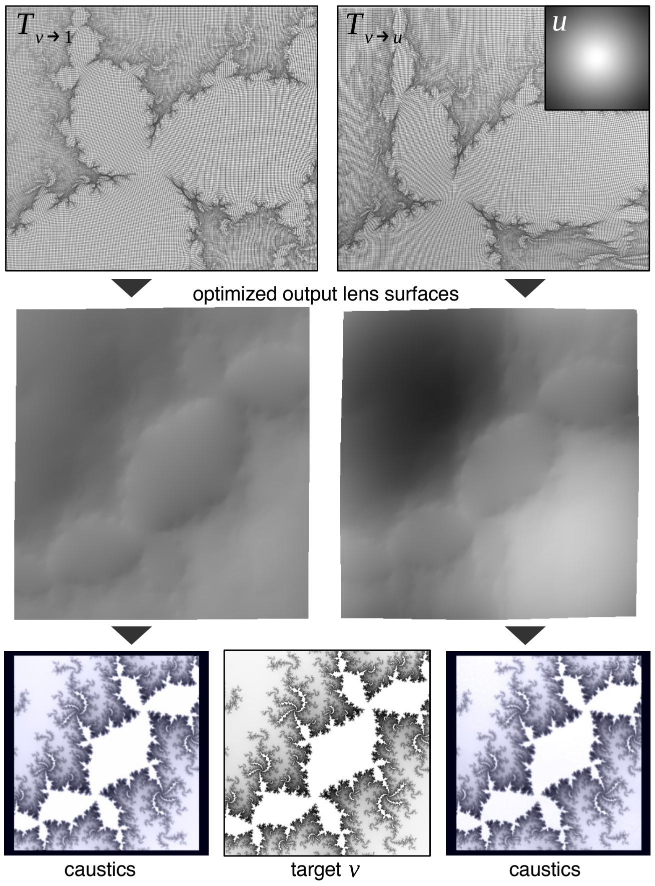


Fig. 16. Surface lense generation for the target v and a uniform (left) and non-uniform (right) incoming light sources. Top row shows close ups of the forward map $T_{v\to 1}$ and composite map $T_{v\to u}$. In both cases, simulated caustics (bottom row) closely ressemble the initial target v.

problem for various applications, including some involving pairs of non-uniform densities. By focusing on this simplified OT problem, we devised a dedicated solver based on 1) an original finite-difference discretization of the resulting Monge-Ampère equation, and 2) a new solver for nonlinear simultaneous equations that, for our problem at hand, outperforms classical Newton, quasi-Newton, and Gauss-Newton solvers by orders of magnitude. Even though we designed this new nonlinear solver in our OT context, it would be interesting to investigate how it would perform on other problems involving nonlinear simultaneous equations or even nonlinear least-square problems for which our conjugate-direction acceleration technique can directly be applied.

As with any numerical solvers, our approach comes with its sources of errors. The discretization of the input density on a uniform grid is exact for an input image but requires a lossy conversion for other kind of inputs (e.g., an analytic functions or a piecewise linear triangular mesh). It would thus be very interesting to generalize our approach to be able to work on adaptive grids or even directly on triangular meshes via the use of higher-order FEM discretizations.

Our solver assumes that the image of a quad cell remains a quad which is only an approximation classically admitted when solving OT through the Monge-Ampère equation [Benamou et al. 2014]. This source of error can be reduced by subdividing the grid, or by investigating the use of higher-order elements.

Regarding performance, our experiments revealed that our specialized solver is already orders of magnitude faster that concurrent, but more general, alternatives when applied on similar settings. Our prototype implementation is currently mono-threaded whereas the computation of the residual vectors can trivially be parallelized on either CPU or GPU, and the triangular solves could also benefits from GPU acceleration [Li 2017]. For *difficult* densities, it would also be interesting to experiment with a coarse-to-fine multi-resolution strategy, which, if successful, could drastically improve the overall performance of our solver.

In terms of generalization, it would be highly desirable to extend our approach to work on non-uniform grids while retaining the same level of accuracy and speed. This is especially the case for our remeshing application for which the input density is initially defined on a non uniform triangular mesh.

Another interesting question is whether our approach could be extended to 3D. Compared to the 2D setting, a first difference is that the decomposition we made to leverage a Laplacian will led to additional quadratic terms, but those wont be a problem as long as the Laplacian dominate. The most notable difference is that direct sparse solvers, as required to make our approach extremely fast, do not behave as well on 3D grids because of a much larger fill-in. It is thus unclear whether our approach will still outperform Newton iterations by orders of magnitude.

ACKNOWLEDGMENTS

This work has been supported by the ANR RichShape project (ANR-14-CE24-0004). We would like to thank the authors of [Solomon et al. 2015] and [de Goes et al. 2012] for making their code publicly available, Bruno Lévy for pointing us to his public implementation of a semi-discrete solver in Geogram, and the anonymous reviewers for their insightful comments. The 3D models and images are provided cour Additionally, we would like to acknowledge the support from the SUTD Digital Manufacturing and Design (DManD) Centre, supported by the Singapore National Research Foundation.

REFERENCES

F. Aurenhammer, F. Hoffmann, and B. Aronov. 1998. Minkowski-Type Theorems and Least-Squares Clustering. *Algorithmica* 20, 1 (1998), 61–76.

J.-D. Benamou, Y. Brenier, and K. Guittet. 2002. The Monge-Kantorovitch mass transfer and its computational fluid mechanics formulation. *International Journal for Numerical Methods in Fluids* 40, 1-2 (2002), 21–30.

Jean-David Benamou, Brittany D. Froese, and Adam M. Oberman. 2014. Numerical Solution of the Optimal Transportation Problem Using the Monge-Ampère Equation. *J. Comput. Phys.* 260 (2014), 107–126.

Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. *ACM Transactions on Graphics* 30, 6 (2011), 1.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. 2010. *Polygon Mesh Processing.*

Yann Brenier. 1991. Polar factorization and monotone rearrangement of vector-valued functions. *Comm. Pure Appl. Math.* 44, 4 (1991), 375–417.

C. G. Broyden. 1965. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comp.* (1965).

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NIPS*, Vol. 26. 2292–2300.

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise Through Optimal Transport. *ACM Transactions on Graphics* 31, 6 (2012).

G.L. Delzanno, L. Chacón, J.M. Finn, Y. Chung, and G. Lapenta. 2008. An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge–Kantorovich optimization. *J. Comput. Phys.* 227, 23 (2008), 9841 – 9864.

Fernando de Goes, Pooran Memari, Patrick Mullen, and Mathieu Desbrun. 2014. Weighted Triangulations for Geometry Processing. *ACM Transactions on Graphics* 33, 3, Article 28 (2014), 13 pages.

Xianfeng Gu, Feng Luo, Jian Sun, and Shing Tung Yau. 2016. Variational principles for Minkowski type problems, discrete optimal transport, and discrete Monge-Ampere equations. *Asian Journal of Mathematics* 20, 2 (2016), 383–398.

W. W. Hager and H. Zhang. 2006. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization* 2, 1 (2006), 35–58.

Steven Haker, Lei Zhu, Allen Tannenbaum, and Sigurd Angenent. 2004. Optimal Mass Transport for Registration and Warping. *International Journal of Computer Vision* 60, 3 (2004), 225–240.

A.S. Householder. 1953. *Principles of Numerical Analysis.* McGraw-Hill. 135–138 pages.

L. Kantorovich. 1942. On the transfer of masses (in Russian). *Doklady Akademii Nauk* 37, 2 (1942), 227–229.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Transactions on Graphics* 35, 4 (2016), 134:1–134:11.

Thibaud Lambert, Pierre Bénard, and Gael Guennebaud. 2016. Multi-Resolution Meshes for Feature-Aware Hardware Tessellation. *Computer Graphics Forum* (2016).

Bruno Lévy. 2015. A Numerical Algorithm for L2 Semi-Discrete Optimal Transport in 3D. *ESAIM: Mathematical Modelling and Numerical Analysis* 49, 6 (2015), 1693–1715.

Bruno Lévy. 2017. Geogram. (2017). http://alice.loria.fr/software/geogram

Bruno Lévy and Erica L. Schwindt. 2018. Notions of optimal transport theory and how to implement them on a computer. *Computers & Graphics* 72 (2018), 135 – 148.

Ruipeng Li. 2017. On Parallel Solution of Sparse Triangular Linear Systems in CUDA. *CoRR* (2017).

Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. 2017. Variance-Minimizing Transport Plans for Inter-surface Mapping. *ACM Transactions on Graphics* 36 (2017), 14.

Quentin Mérigot. 2011. A Multiscale Approach to Optimal Transport. *Computer Graphics Forum* 30, 5 (2011), 1583–1592.

Ján Morovic and Pei-Li Sun. 2003. Accurate 3D Image Colour Histogram Transformation. *Pattern Recogn. Lett.* 24, 11 (2003), 1725–1735.

Vincent Nivoliers, Bruno Lévy, and Christophe Geuzaine. 2015. Anisotropic and feature sensitive triangular remeshing using normal lifting. *J. Comput. Appl. Math.* 289 (2015).

J. Nocedal and S. Wright. 2006. *Numerical Optimization.* Springer New York.

Nicolas Papadakis, Gabriel Peyré, and Edouard Oudet. 2014. Optimal Transport with Proximal Splitting. *SIAM Journal on Imaging Sciences* 7, 1 (2014), 212–238.

Hélène Perrier, David Coeurjolly, Feng Xie, Matt Pharr, Pat Hanrahan, and Victor Ostromoukhov. 2018. Sequences with Low-Discrepancy Blue-Noise 2-D Projections. *Computer Graphics Forum* 37 (2018).

Gabriel Peyré and Marco Cuturi. 2018. Computational Optimal Transport. (2018). arXiv:arXiv:1803.00567

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc.

Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. 2017. Wasserstein Blue Noise Sampling. *ACM Transactions on Graphics* 36, 5 (2017).

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Transactions on Graphics* 36, 2 (2017), 16:1–16:16.

Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. 2014. High-contrast Computational Caustic Design. *ACM Transactions on Graphics* 33, 4 (2014), 74:1–74:11.

Justin Solomon. 2018. Optimal Transport on Discrete Domains. (2018). arXiv:arXiv:1801.07745

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* 34, 4 (2015), 66:1–66:11.

C. Villani. 2008. *Optimal Transport: Old and New.* Springer Berlin Heidelberg.

Shi-Qing Xin, Bruno Lévy, Zhonggui Chen, Lei Chu, Yaohui Yu, Changhe Tu, and Wenping Wang. 2016. Centroidal Power Diagrams with Capacity Constraints: Computation, Applications, and Extension. *ACM Transactions on Graphics* 35, 6 (2016), 244:1–244:12.

Xin Zhao, Zhengyu Su, X Gu, Arie Kaufman, Jian Sun, Jie Gao, and Feng Luo. 2013. Area-Preservation Mapping using Optimal Mass Transport. *Visualization and Computer Graphics, IEEE Transactions on* 19, 12 (2013), 2838–2847.