Numerical Optimal Transport, sliced, semi-discrete and regularized

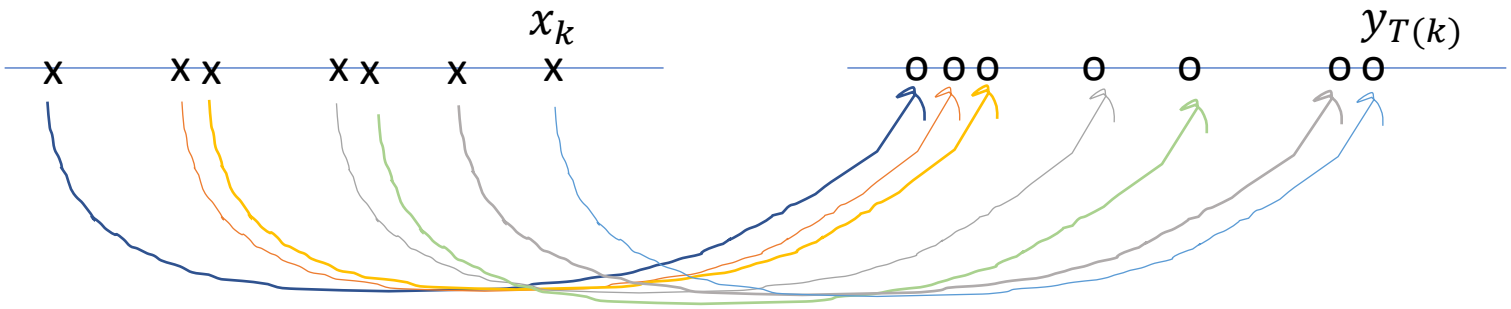
Nicolas Bonneel

Sliced and Radon Wasserstein

Optimal transport is simple in 1D

In the following, we assume $c(x, y) = f(x - y)$ with f non-negative and convex (e.g. quadratic cost). Then transport map T preserves order.

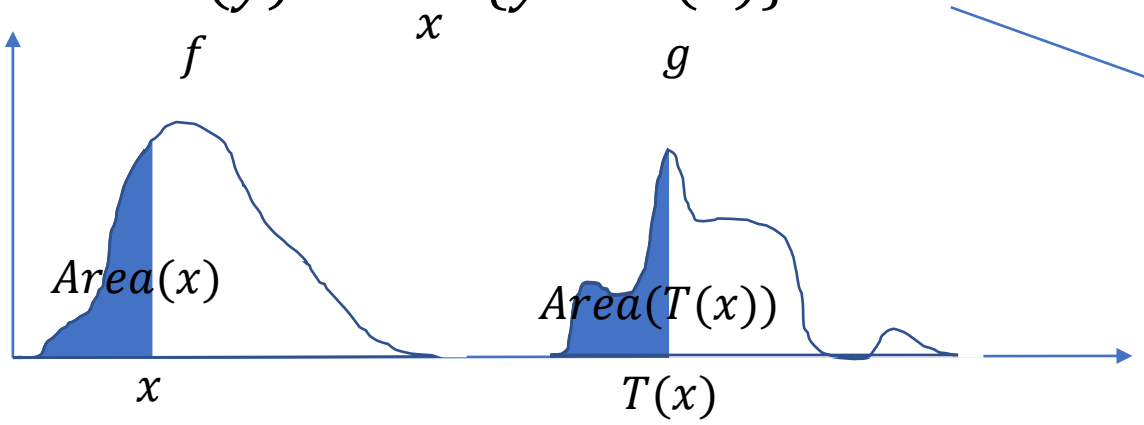
- Discrete case, $\mu = \sum_{k=1}^{n} \delta_{x_k}$, $\nu = \sum_{k=1}^{n} \delta_{y_k}$



Implementation: sort $\{x_k\}$ and $\{y_k\}$, T assigns in order.

Optimal transport is simple in 1D

- Continuous case with density, $\mu = f\ dx, \nu = g\ dy$ (or similarly without)
  - $\int_{-\infty}^{x} f(u)du = \int_{-\infty}^{T(x)} g(u)du$
  - $\Rightarrow T = G^{-1} \circ F$ with $F(x) = \int_{-\infty}^{x} f(u)du$ and $G(x) = \int_{-\infty}^{x} g(u)du$
  - Generalize G^{-1} : $G^{-1}(y) = \min_{x}\{y = G(x)\}$

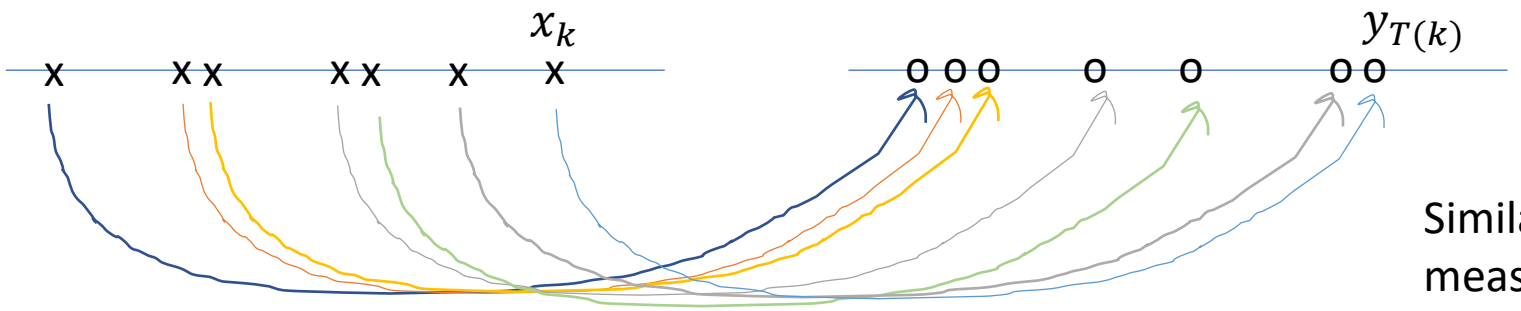


Quantile function: e.g.: “what salary corresponds to the first percentile”

  - OT cost: $\int_0^1 c\left(F^{-1}(t) - G^{-1}(t)\right)dt$

Wasserstein barycenters are simple in 1D

- Discrete case, $\mu = \sum_{k=1}^{n} \delta_{x_k}$, $\nu = \sum_{k=1}^{n} \delta_{y_k}$: advect partway



Similar approach for more than 2 measure (take a reference measure, and advect it by $\sum \alpha_i T_i$)

- Continuous case: $F_{bary}^{-1}(x) = \sum_i \alpha_i \, F_i^{-1}(x)$

Proof: Given $\{(\mu_i, \lambda_i)\}_i$ and a reference measure μ, $bary = (\sum_i \lambda_i T_{\mu \to \mu_i}) \# \mu$

With μ Lebesgues, $T_{\mu \to \mu_i} = F_i^{-1}$

Projecting optimal transport on lines

- In higher dimension d, consider energy:

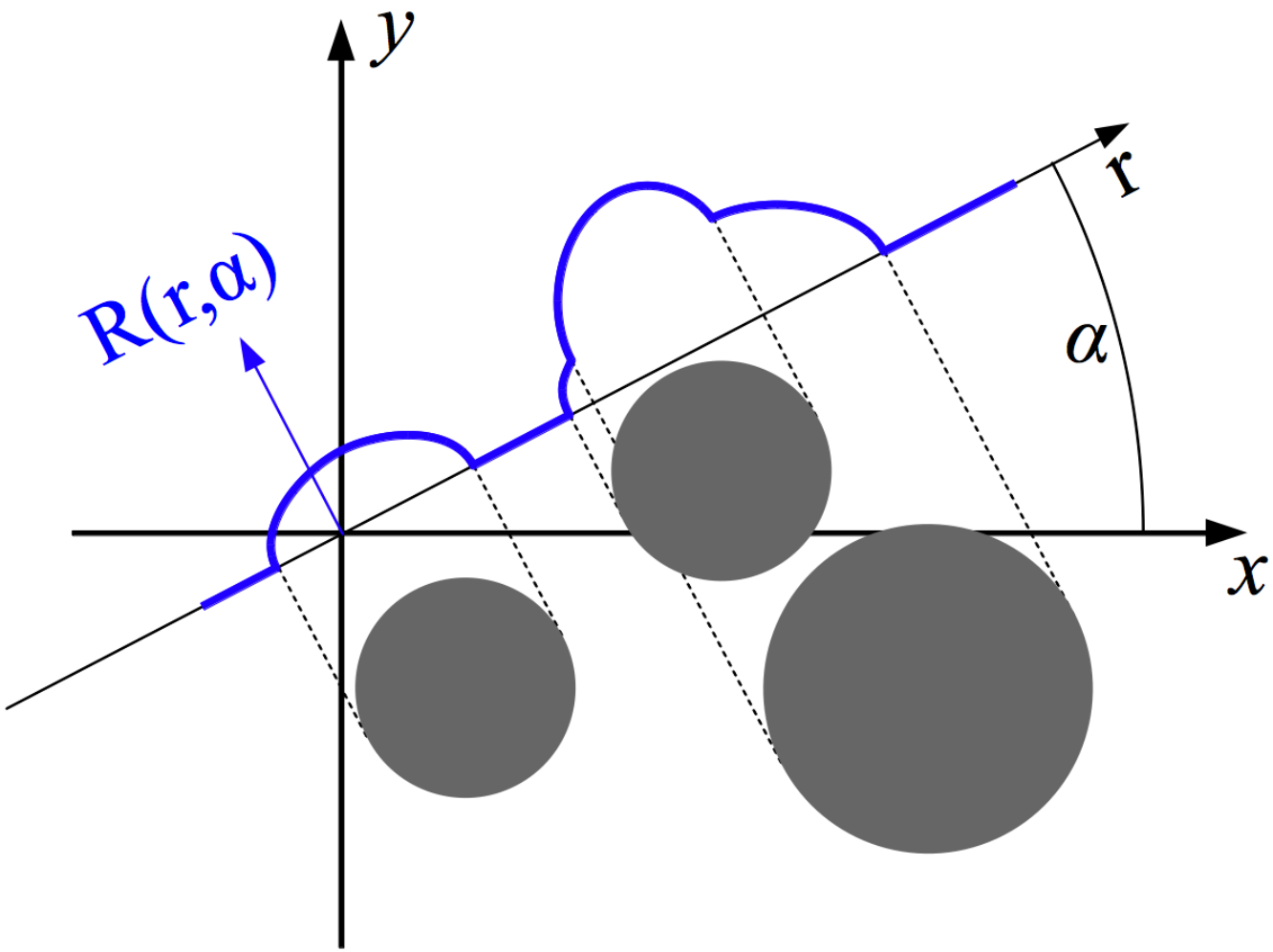
$$W_{proj}(\mu,\nu)^2 = \int_{S^{d-1}} W\left(\mu^\theta,\nu^\theta\right)^2 d\theta$$

μ^θ such that, $\forall f$ continuous, tending to 0 at ∞

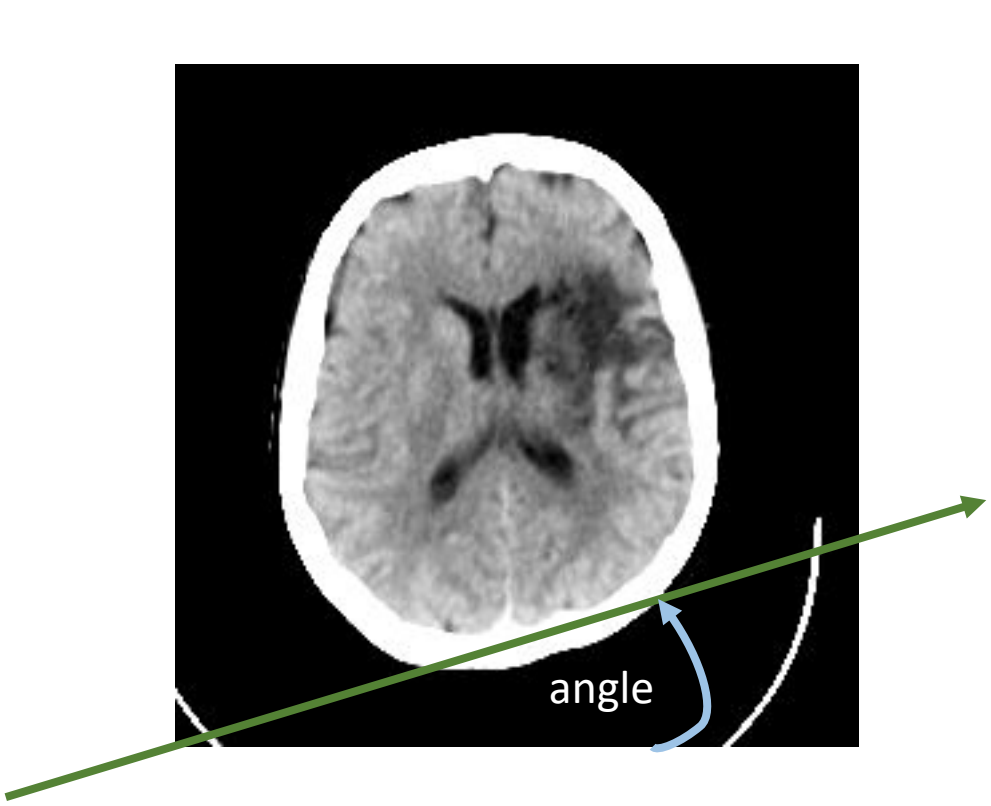
$$\int_{\Omega^{\mathrm{d}}} f(t,\theta)d\mu(t,\theta) = \int_{S^{d-1}}\left(\int_{\mathbb{R}} f(t,\theta)d\mu^\theta(t)\right)d\theta$$

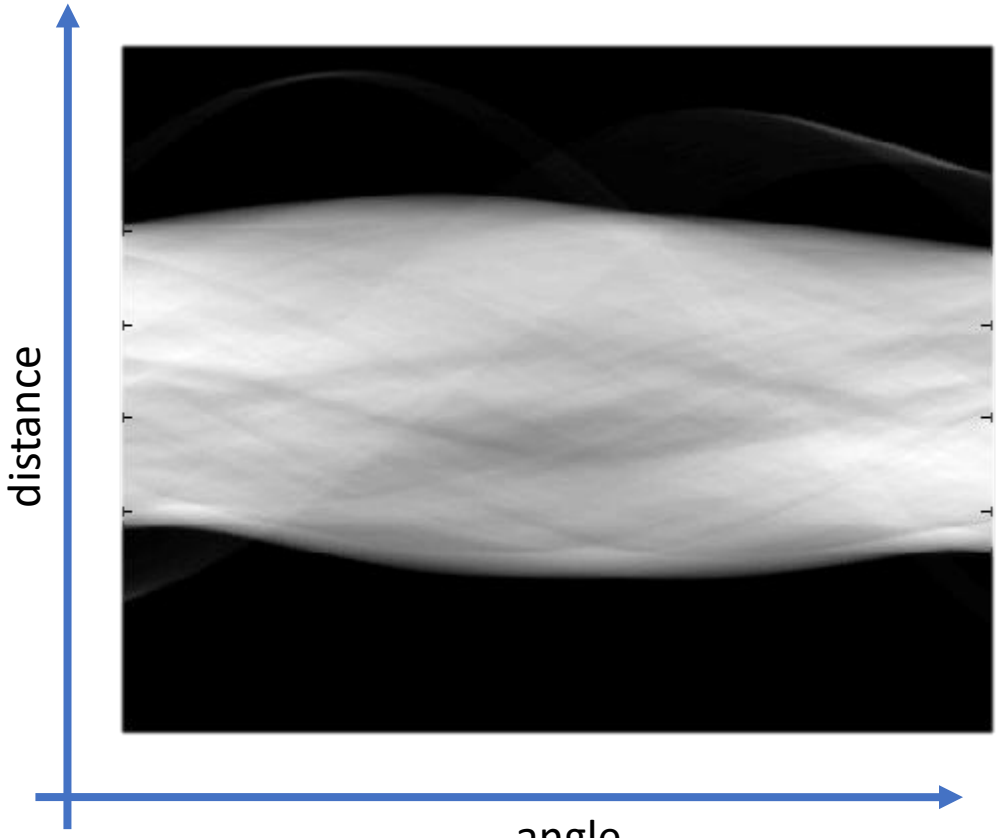
With $\Omega^{\mathrm{d}} = \mathbb{R}\times S^{d-1}$ and μ,ν probability measures on Ω^{d}

Radon transform



Radon transform



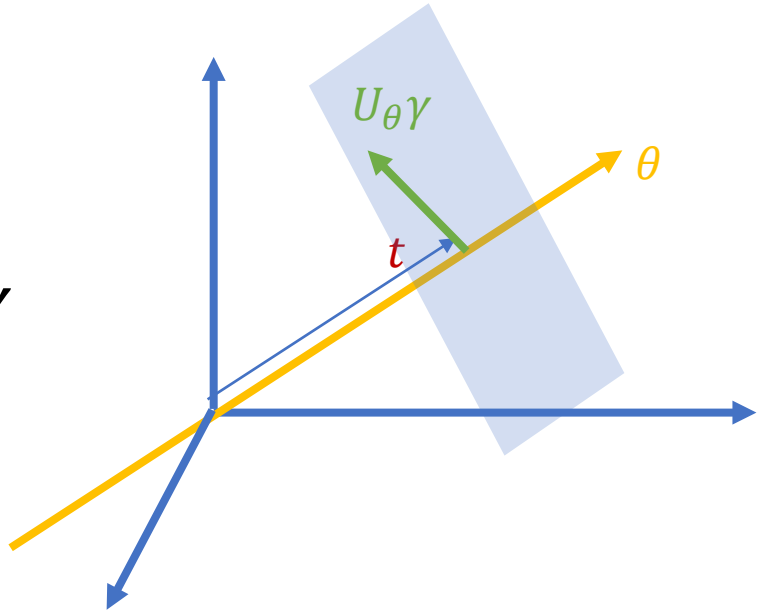


Radon transform

- Formally

$$Rf(t,\theta) = \int_{\mathbb{R}^{d-1}} f(t\theta + U_\theta \gamma) d\gamma$$

Can be extended to measures by duality



For functions, $R: L^1(\mathbb{R}) \to L^1(\Omega^d)$

Similarly for probability measures (not surjective).

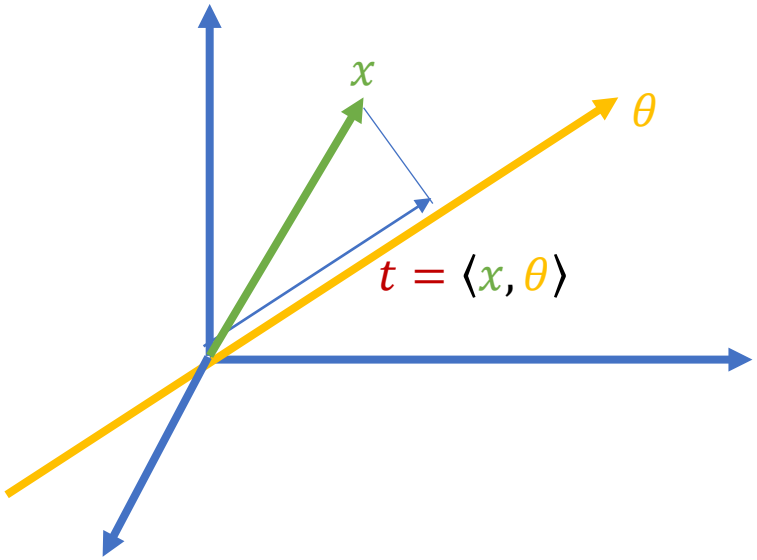
Radon transform

- Back-Projection

$$R^* f(x) = \int_{S^{d-1}} g(\langle x, \theta \rangle, \theta) d\theta$$

Can be extended to measures by duality.

We have $R^* R f = h \star f$, with h low pass filter

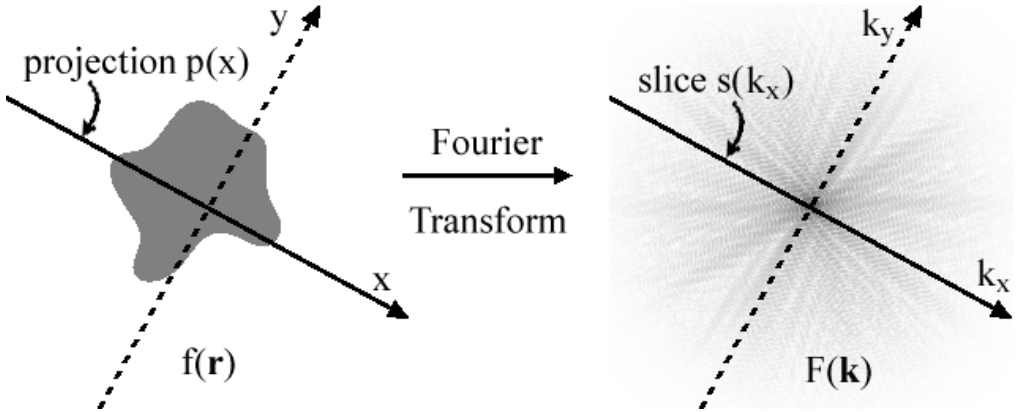


- Inverse Radon transform:

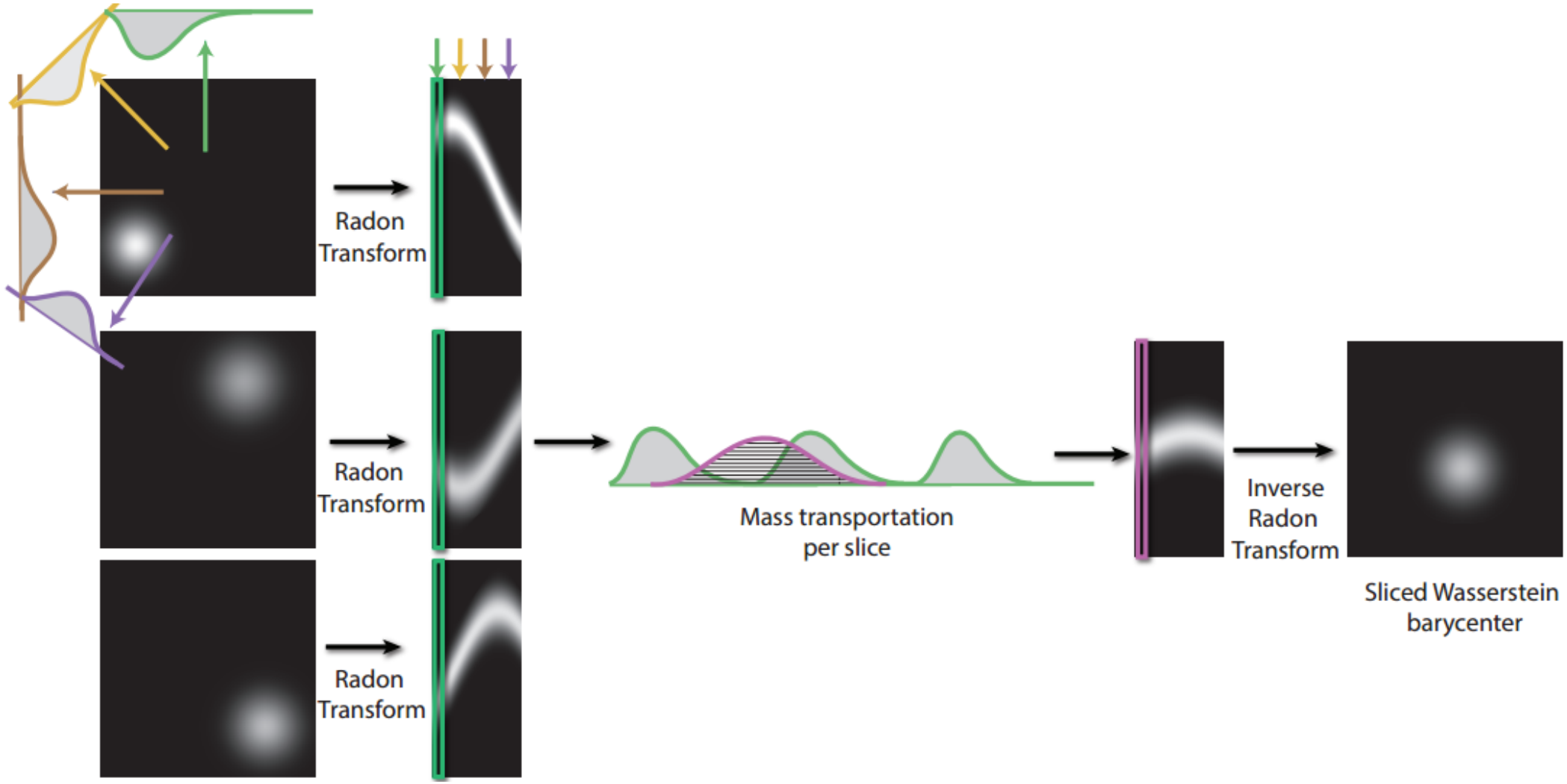
$$R^+ g = h^+ \star (\mathrm{R}^* \mathrm{g})$$

Radon transform

- Numerically, can be efficiently inverted with inverse FFTs
- Fourier Slice Theorem:
  - the Fourier transform of the projection of an *N*-dimensional function $f(\mathbf{r})$ onto an *m*-dimensional linear submanifold is equal to an *m*-dimensional slice of the *N*-dimensional Fourier transform of that function consisting of an *m*-dimensional linear submanifold through the origin in the Fourier space which is parallel to the projection submanifold

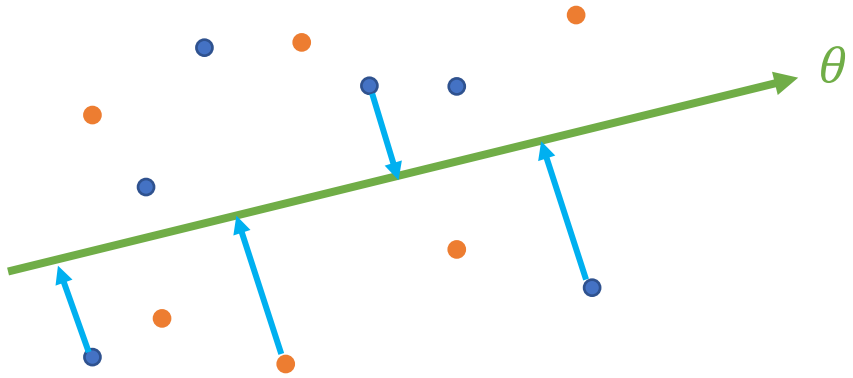


Radon Wasserstein barycenter



Sliced Wasserstein energy

- For discrete measures, consider instead the projections



- $W_{sliced}(\mu,\nu)^2 = \int_{S^{d-1}} W(P_\theta \#\mu, P_\theta \#\nu)^2 d\theta$
- Can be estimated via Monte Carlo: $\int_\Omega f(x)dx \approx \frac{1}{K}\sum_k f(x_k)$ with $x_k \sim \mathcal{U}(\Omega)$

Sliced Wasserstein Barycenter

- Defined similarly to Wasserstein barycenters as

$$\operatorname{argmin} \sum_i \lambda_i \, W^2_{sliced}(bary, \mu_i) = \operatorname{argmin} E(bary, \{(\lambda_i, \mu_i)\})$$

- Can be obtained by gradient descent
  - Start from $\text{bary}^0 = \mu_0$ (or anything else)
  - $bary^{n+1} = bary^n - \varepsilon \, \nabla E(bary^n, \{(\lambda_i, \mu_i)\})$
  - $\nabla E(bary^n, \{(\lambda_i, \mu_i)\}) = \sum_i \ \lambda_i \int_{S^{d-1}} \nabla W(P_\theta \# \mu, P_\theta \# \nu)^2 d\theta$

Discrete 1d: just 1d vectors !

Sliced Wasserstein flow

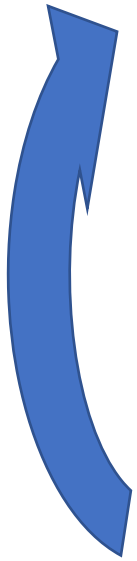
- Transport μ towards ν
- Not interested in their distance, but by the gradient flow:

$$\mu'(t) = -\nabla W_{sliced}(\mu(t), \nu)$$

- In practice, similar approach as Sliced Wasserstein barycenter
  - Consider the barycenter of $\{(\mu, 0), (\nu, 1)\}$ and start the gradient descent from μ

Lab : THAT'S WHAT YOU WILL IMPLEMENT
(see next slide for details)

Sliced Wasserstein flow



- Take one (or many) random direction(s) d
- Project samples of μ and ν on d : μ' = Proj(μ) and ν' = Proj(ν)
- Sort ν' and ν'
- Update μ by $\mu = \mu + (\nu' - \mu').d$

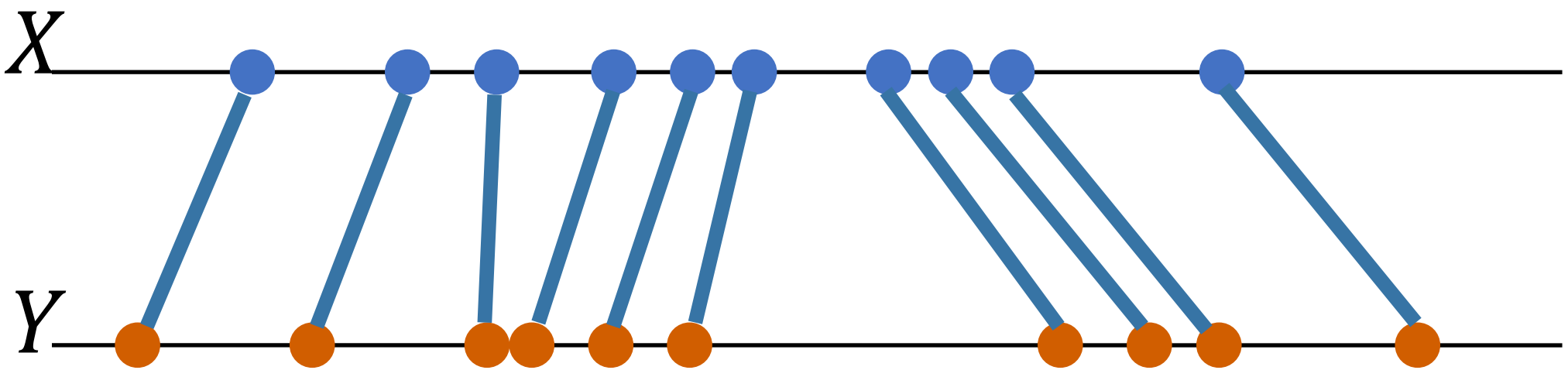
Wasserstein Barycenter and its Application to Texture Mixing. [Rabin et al. 2011]

Sliced Wasserstein flow

f

g

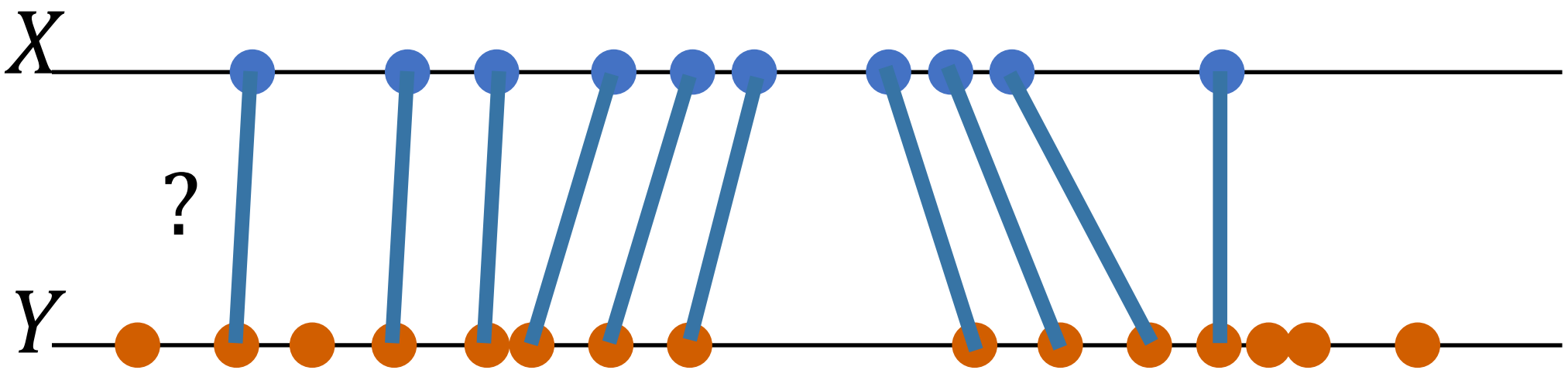
1-d Linear Assignment Problem is trivial*



*assuming the cost c is a convex function of |x-y|

Partial optimal assignment ?

=> Sliced Partial Optimal Transport, [Bonneel and Coeurjolly 2019]



$$W(f,g) = \min \sum_{i,j} c_{i,j}\pi_{i,j} \qquad \text{s.t.} \qquad \sum_j \pi_{i,j} = 1$$

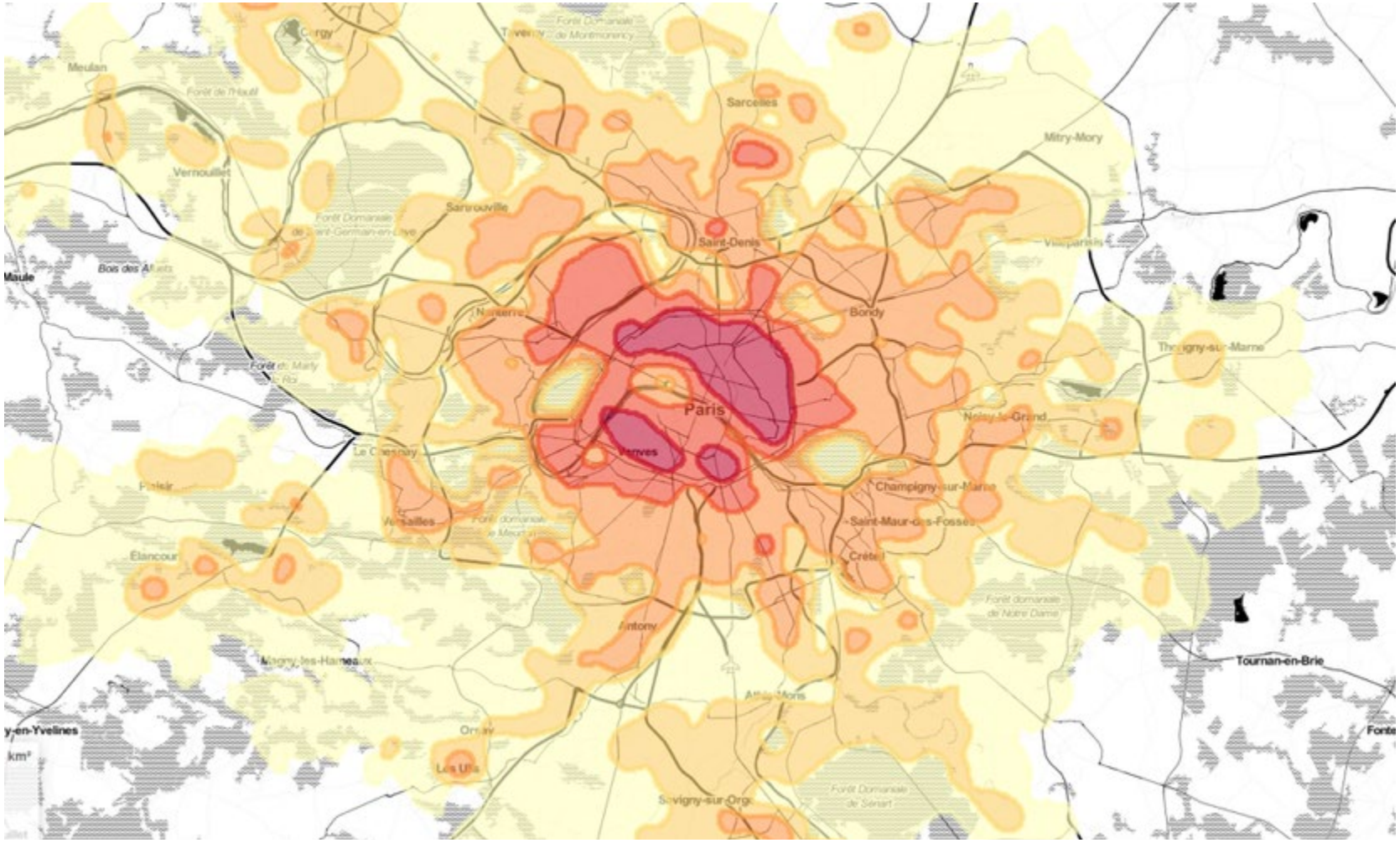
$$\sum_i \pi_{i,j} \leq 1$$

$$\pi_{i,j} \geq 0$$

$$\min_{T \text{ injective}} \sum_i c(x_i, y_{T(i)})$$

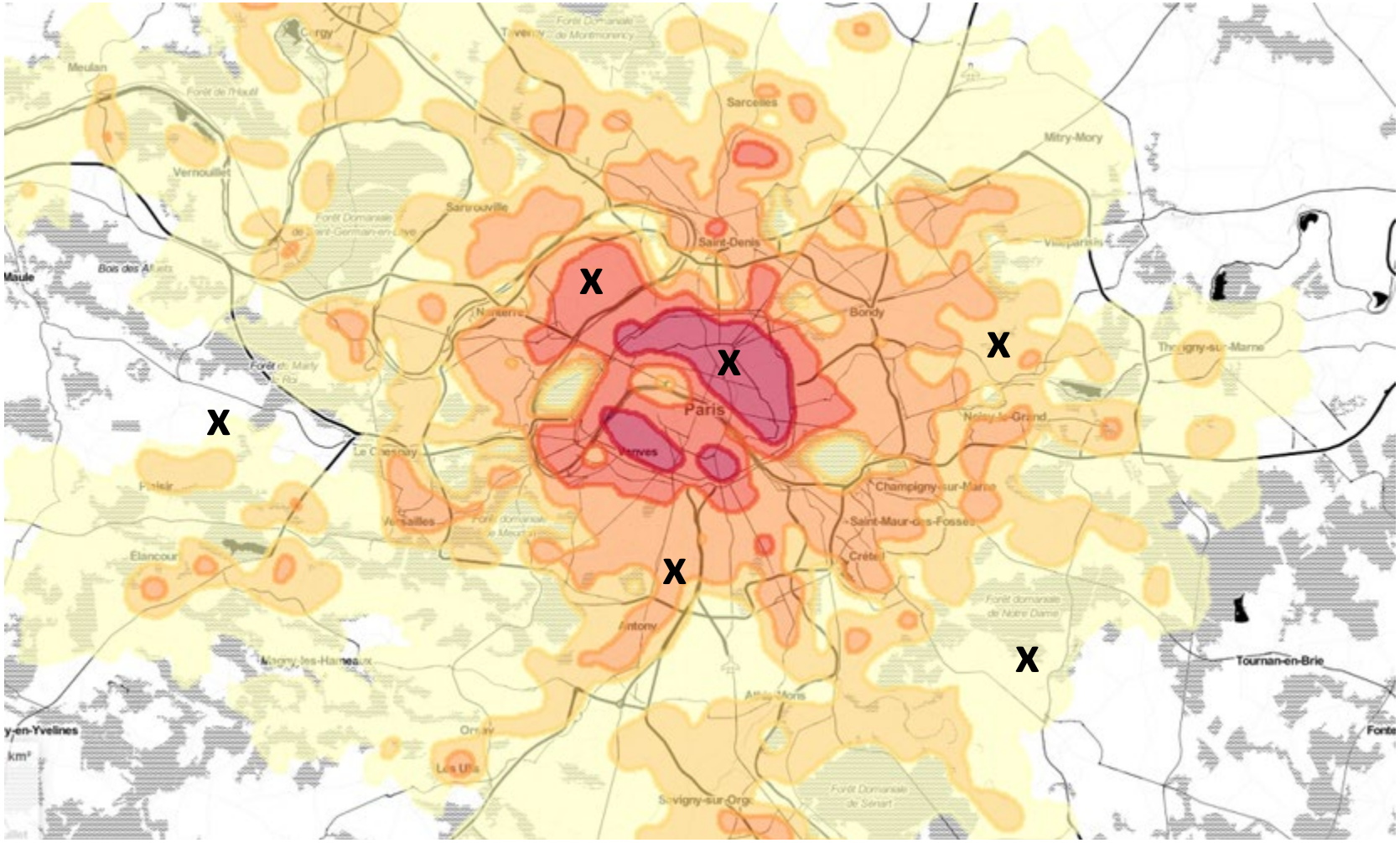
Semi-discrete optimal transport

Semi-discrete Optimal Transport



Population density f

Semi-discrete Optimal Transport



Set of bakeries, factories, …?

Semi-discrete Optimal Transport

No constraint on production: population go to their nearest bakery/factory/… regardless of population density

Semi-discrete Optimal Transport

Limited production: population go to the nearest bakery/factory **with sufficient production**!

Semi-discrete Optimal Transport

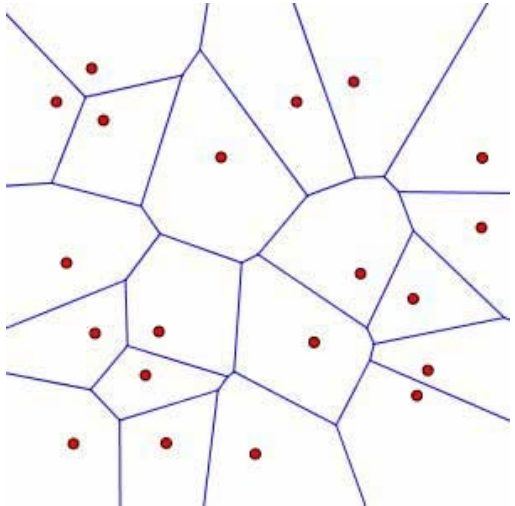
Limited production: population go to the nearest bakery/factory **with sufficient production**!

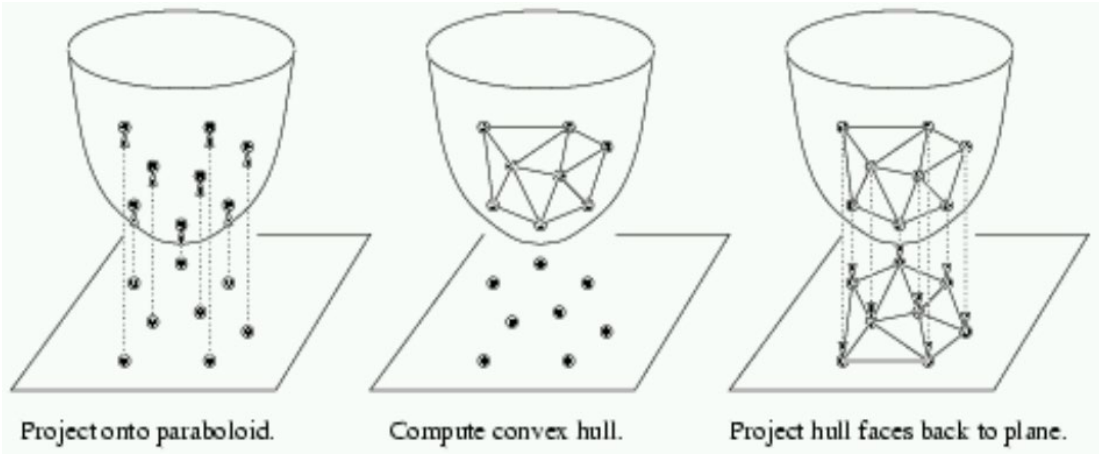
Voronoi diagram

- A partition such that each point x is assigned to its closest site x_i

$$\|x - x_i\|^2 \leq \left\|x - x_j\right\|^2 \quad \forall j$$

- The dual of a Delaunay triangulation: a triangulation of the sites such that no other site is encompassed by the circumcircle of a triangle
  - Also: convex hull of a parabolic lifting



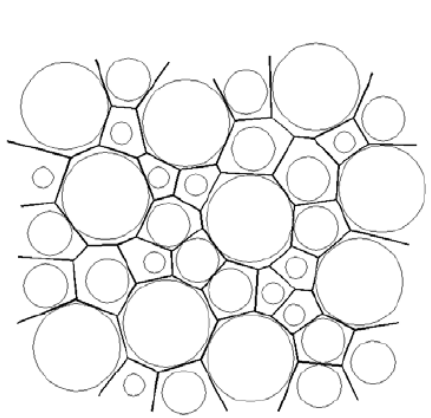


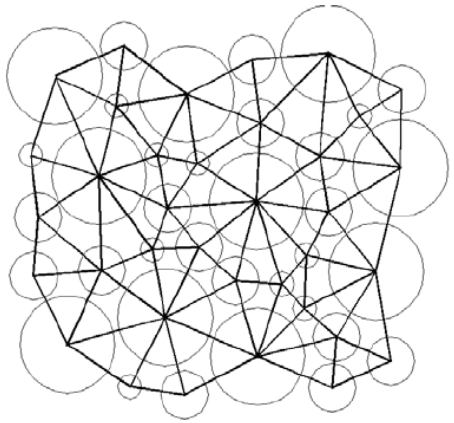
Power diagram (Laguerre diagram)

- A partition s.t. each point x is assigned to its closest site x_i with weight w_i

$$\|x - x_i\|^2 - w_i \leq \left\|x - x_j\right\|^2 - w_j \quad \forall j$$

- Can be computed by lifting a Voronoi diagram
  - Consider site coordinates $x_i' = \left(x_i \, ; \, \sqrt{c - w_i} \right)$ for large constant c ; $x' = (x \, ; 0)$
  - Then $\|x' - x_i'\|^2 \leq \left\|x' - x_j'\right\|^2 \quad \forall j$
- Any partition into convex polyhedral cells is a power diagram of some sites





Recall

- Primal:

$$\inf_{\gamma} \int_{X \times Y} c(x, y) d\gamma(x, y) \quad \text{s.t. } \gamma \in \Pi(\mu, \nu), \text{transport plan}$$

- Dual:

$$\max_{\psi, \phi} \int_{X} \psi(x) d\mu(x) + \int_{Y} \phi(y) d\nu(y) \quad s.t. \ \psi(x) + \phi(y) \leq c(x, y)$$

c-conjugate

- Replacing ψ by the c-conjugate of ϕ improves the cost:

$$\psi \leftarrow \phi^c(x) = \inf_y c(x,y) - \phi(y)$$

$$\Rightarrow$$

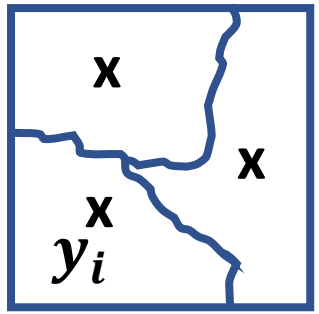
$$\max_{\phi} \int_X \left(\inf_y c(x,y) - \phi(y)\right) d\mu\,(x) + \int_Y \phi(y) d\nu(y)$$

Back to optimal transport

- Continuous: $\max_{\phi} \int_X \left(\inf_y c(x,y) - \phi(y)\right) d\mu(x) + \int_Y \phi(y) d\nu(y)$
- Replacing ϕ by discrete $\{\phi_i\}$ and $\nu = \sum_i \lambda_i \delta_{y_i}$

$$\max_{\{\phi_i\}} \int_X \left(\min_{\{y_i\}} c(x,y_i) - \phi_i\right) d\mu(x) + \sum_i \lambda_i \phi_i$$

Partitioning X:



$$\max_{\{\phi_i\}} \sum_i \int_{x \in Cell_i} \left(c(x,y_i) - \phi_i\right) d\mu(x) + \sum_i \lambda_i \phi_i$$

Back to optimal transport

- Particular case: $c(x, y) = \|x - y\|^2$
  - Cells are polygonal ⇒ Power diagram

- Goal: maximize:

$$\max_{\{\phi_i\}} \sum_i \int_{x \in Cell_i} (\|x - y_i\|^2 - \phi_i)\, d\mu\,(x) + \sum_i \lambda_i \phi_i = \max_{\{\phi_i\}} E(\{\phi_i\})$$

Derivative w.r.t. ϕ_j: $\frac{\partial E}{\partial \phi_j} = -\int_{x \in Cell_j} d\mu\,(x) + \lambda_j$

i.e. « the population in cell j minus how many croissants bakery j can make »

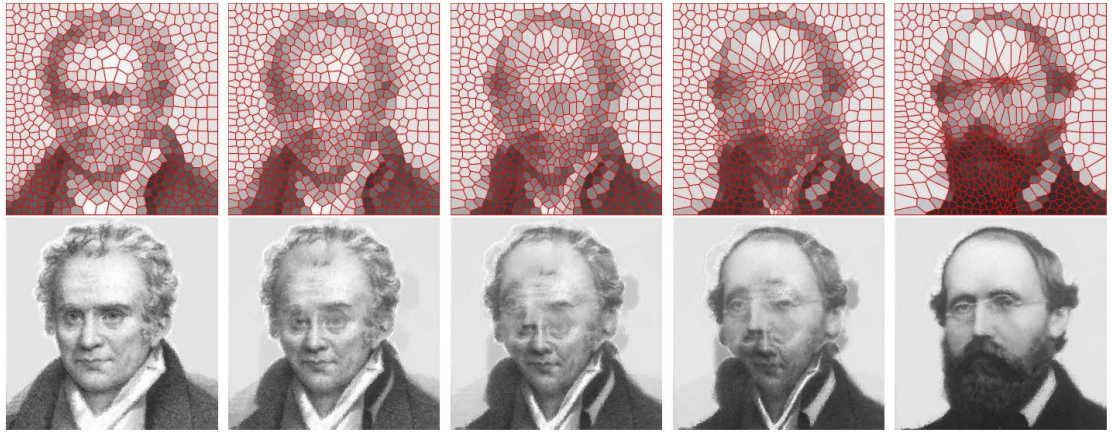
Back to optimal transport

- Easy way: gradient ascent:
  - Compute power diagram of $\{y_i\}$ with weights $\{\phi_i\}$
  - $\phi_i \leftarrow \phi_i + \epsilon \,.\left(\lambda_i - \int_{x \in Cell_i} d\mu\,(x)\right)$
  - Iterate
- Intuition: “bakery i increases its selling price ϕ_i until sufficiently many clients leave, or decreases its selling price until sufficiently many clients come”
- Again, if everybody increase its price by same amount: no change.

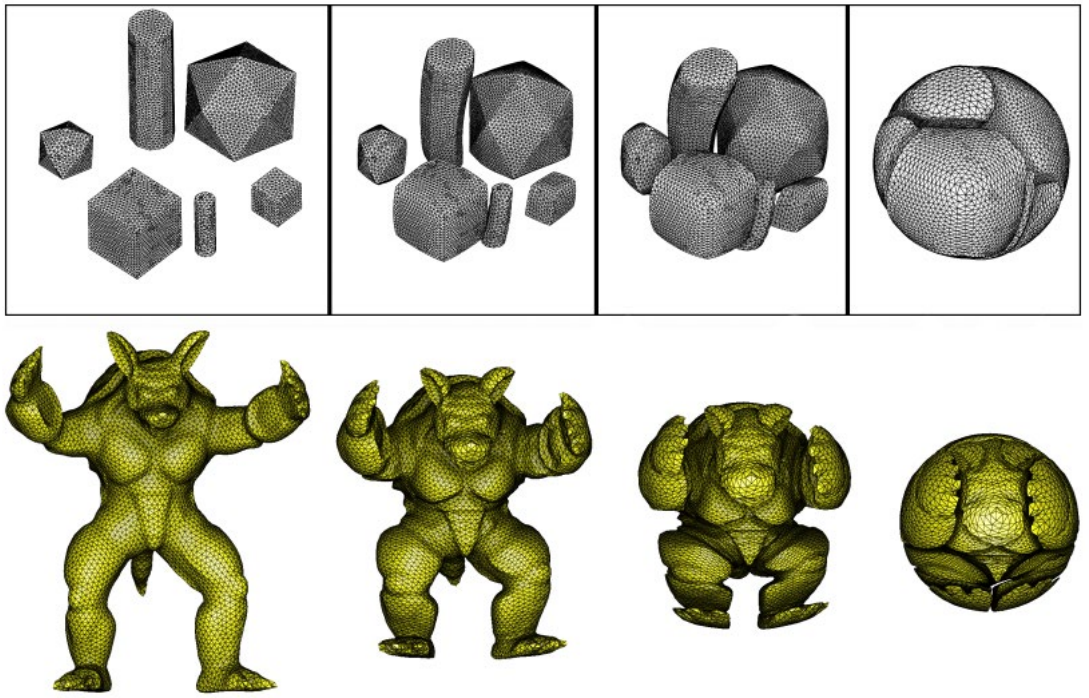
Back to optimal transport

- Harder but faster
  - Newton ; second order, requires Hessian $H = \left[\frac{\partial^2 E}{\partial \phi_i \partial \phi_j}\right]$
  - $\frac{\partial^2 E}{\partial \phi_i \partial \phi_j} = \int_{Cell_i \cap Cell_j} \frac{1}{2\|y_i - y_j\|}\, d\mu\,(x)$ if $i \neq j$
  - $\frac{\partial^2 E}{\partial \phi_i^2} = -\sum_{j \neq i} \frac{\partial^2 E}{\partial \phi_i \partial \phi_j}$
  - Newton algorithm $(\phi_i) \leftarrow (\phi_i) + \epsilon\,.H^{-1}\left(\frac{\partial E}{\partial \phi_j}\right)$

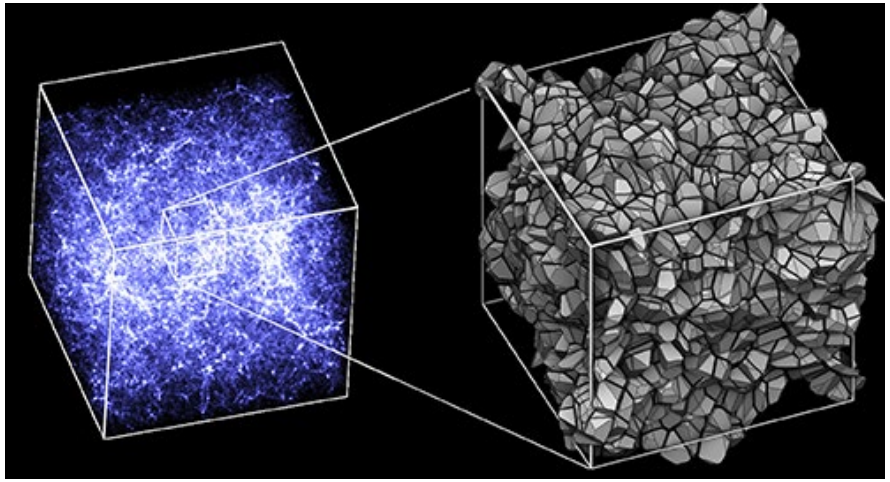
Back to optimal transport



A Multiscale Approach to Optimal Transport [Mérigot 2011]



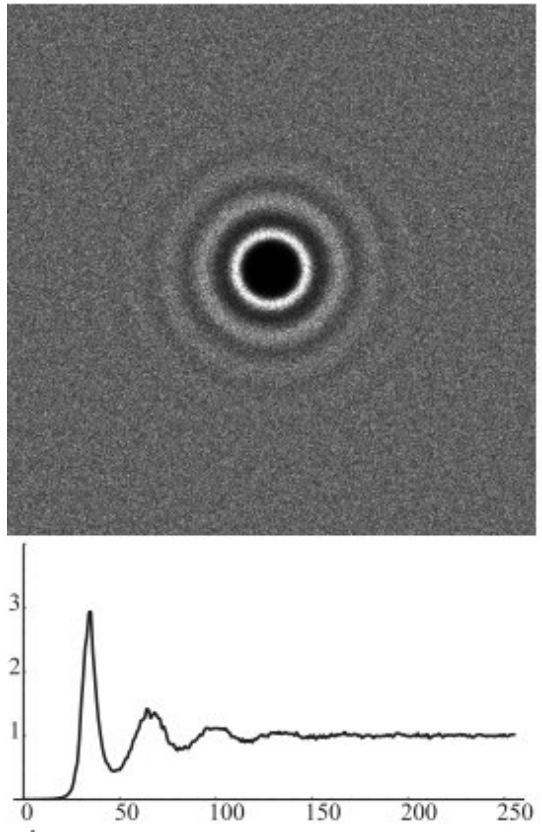
A Numerical Algorithm for L2 Semi-discrete Optimal Transport in 3D [Lévy 2015]



Optimal Transport Reconstruction of Biased Tracers in Redshift Space [Nikakhtar et al. 2023]

Application

- Also optimizes for the locations p



Blue Noise through Optimal Transport [de Goes et al. 2012]

Entropy-regularized optimal transport

The Sinkhorn algorithm

- Kantorovich optimal transport: $\min_{m} \sum_i \sum_j c_{i,j}\, m_{i \to j}$ with constraints
- Rewritten as :

$$\min_{M \in \mathcal{U}(r,c)} \langle C, M \rangle$$

with $\mathcal{U}(r,c)$ matrices whose rows sum to r and columns to c

- Idea: consider instead

$$\min_{M \in \mathcal{U}(r,c)} \langle C, M \rangle - \epsilon E(M)$$

where $E(M) = -\sum M_{ij}(\log(M_{ij}) - 1)$ is the entropy, ϵ a small constant

Iterative Bregman Projections for Regularized Transportation Problems [Benamou et al. 2014]

Sinkhorn Distances: Lightspeed Computation of Optimal Transport [Cuturi 2013]

The Sinkhorn algorithm

$$\min_{M \in \mathcal{U}(r,c)} \langle C, M \rangle - \epsilon E(M)$$

- Can be rewritten as a projection:

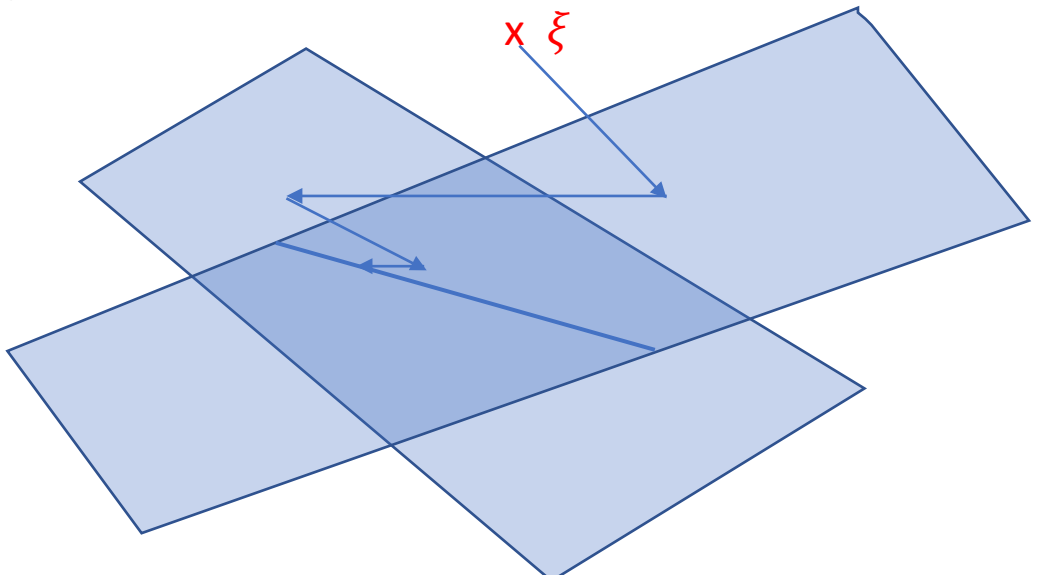
$$\min_{M \in \mathcal{U}(r,c)} KL(M, \xi)$$

where $\xi = \exp\left(-\frac{C}{\epsilon}\right)$ and $KL(M, \xi) = \sum M_{ij}\left(\log\left(\frac{M_{ij}}{\xi_{ij}}\right) - 1\right)$ the Kullback-Leibler divergence

The Sinkhorn algorithm

$$\min_{M \in \mathcal{U}(r,c)} KL(M, \xi)$$

- This is a projection on the intersection of two affine constraints, due to $\mathcal{U}(r, c)$
- We can thus apply Bregman projections: we iteratively project on each constraint



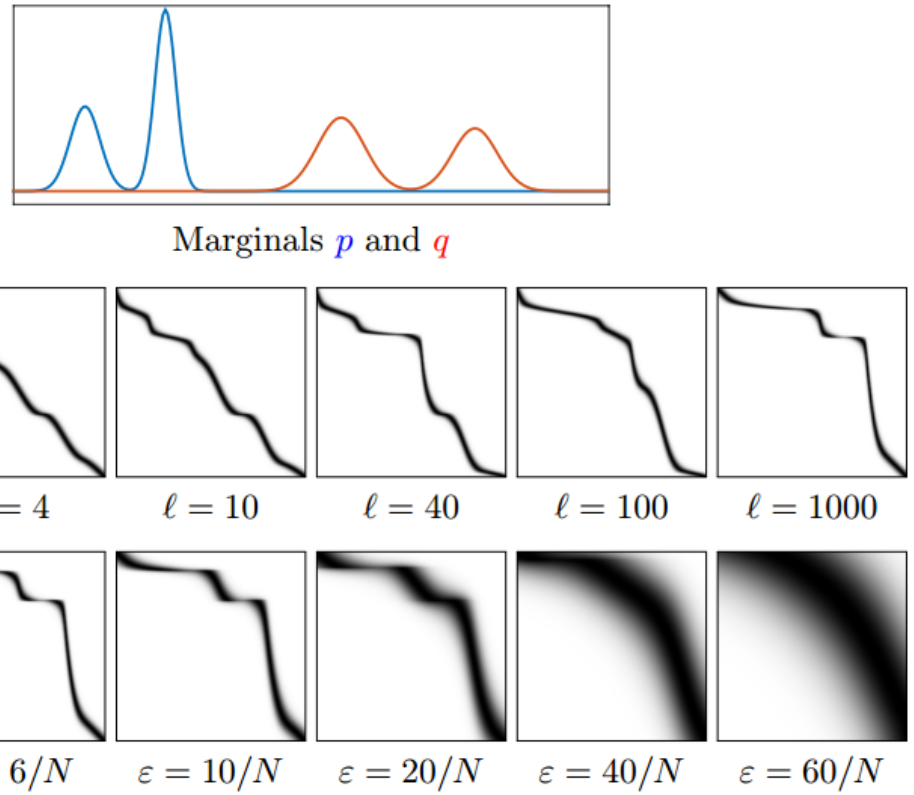
The Sinkhorn algorithm

- Projecting on constraints:
  - Constraints: $\sum_i M_{ij} = r_j$ and $\sum_j M_{ij} = c_i$
  - $M'_{ij} = \frac{M_{ij}}{\sum_i M_{ij}} . r_j$ and $M'_{ij} = \frac{M_{ij}}{\sum_j M_{ij}} . c_i$ corresponds to projection with KL
  - Row/column scaling
  - Corresponds to left/right multiplying M by diagonal matrix

The Sinkhorn algorithm

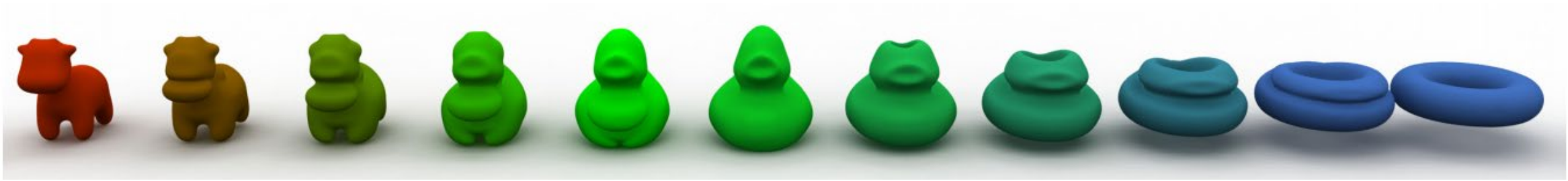
- We obtain the algorithm:

- $u^{(n)} = \frac{f}{\xi\, v^{(n)}}$
- $v^{(n+1)} = \frac{g}{\xi^T\, u^{(n)}}$
- $M = diag\left(u^{(n)}\right) \xi\, diag(v^{(n)})$



The Sinkhorn algorithm

- We realize that $\xi\ v^{(n)}$ can be computed efficiently
  - E.g., if $c(x,y) = \|x-y\|^2$, $\xi_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\epsilon}\right)$
  - Then $\xi\ v^{(n)}$ is just a Gaussian convolution
  - So, it is a separable operator, and efficiently done in high-dimension



Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains
[Solomon et al. 2015]

Implementation

- Should converge to OT when $\epsilon \to 0$
  - Not numerically stable → can be computed in log-domain (not convolutional anymore)
    - Note: $M = diag(u^{(n)})\xi\ diag(v^{(n)}) \to \epsilon \log M_{ij} = \phi_i - c_{ij} + \psi_j$
    - Scalings u, v are exponentials of [dual variables divided by ϵ]
    - Replace $\sum_i \exp\left(-\frac{c_{ij}}{\epsilon}\right) u_i$ by $(\log \sum_i \exp(-\frac{c_{ij}}{\epsilon} + \frac{\phi_i}{\epsilon} + \frac{\psi_j}{\epsilon})) - \frac{\psi_j}{\epsilon}$
  - Requires more iterations → multiscale approaches, GPU
  - Regularization can be important in ML
- Do not truncate gaussians
  - $\xi_{ij} = \exp\left(-\frac{c_{ij}}{\epsilon}\right) = 0 \Rightarrow c_{ij} = \infty$: may prevent solutions

Regularized Wasserstein Barycenters

- Finds barycenter p of input measures $\{p_s\}_s$ such that

$$p = \operatorname{argmin} \sum_s \lambda_s W_\epsilon^2(p, p_s)$$

- Rewritten as the problem of finding transport plans $\{M_s\}_s$

$$\min \sum_s \lambda_s KL(M_s, \xi_s)$$

With constraints $\forall s, \sum_i M_{s,ij} = p_{s,j}$ and $\exists p, \sum_j M_{s,ij} = p_i$

Regularized Wasserstein Barycenters

- How to project on the set $\{\{M_s\}_s \mid \exists p, \forall s \ \sum_j M_{s,ij} = p_i\}$?
- $\nabla_M KL(M, \xi) = \log \frac{M}{\xi}$ (division component-wise)
- Add Lagrange multipliers $\{l_s \in \mathbb{R}^n\}$

$$\sum_s \left(\lambda_s KL(M_s, \xi_s) + \sum_i l_{s,i} \left(\sum_j M_{s,ij} - p_i \right) \right)$$

- Differentiate wrt M_s, p_i, l_s and set to 0

$$\nabla_{M_{s,ij}}: \quad \forall s, i, j \quad \lambda_s \log \frac{M_{s,ij}}{\xi_{s,ij}} + l_{s,i} = 0$$

$$\nabla_p: \qquad \sum_s l_s = 0$$

$$\nabla_{l_{s,i}}: \ \forall s, i \qquad \sum_j M_{s,ij} - p_i = 0$$

Regularized Wasserstein Barycenters

$$\nabla_{M_{s,ij}}: \quad \forall s,i,j \quad \lambda_s \log \frac{M_{s,ij}}{\xi_{s,ij}} + l_{s,i} = 0$$

$$\nabla_p: \quad \sum_s l_s = 0$$

- Denote $a_s = \exp(-l_s)$:

$$\lambda_s \log \frac{M_{s,ij}}{\xi_{s,ij}\ a_{s,i}^{1/\lambda_s}} = 0 \quad \Rightarrow M_{s,ij} = \xi_{s,ij}\ a_{s,i}^{1/\lambda_s}$$

$$\Pi_s a_s = 1$$

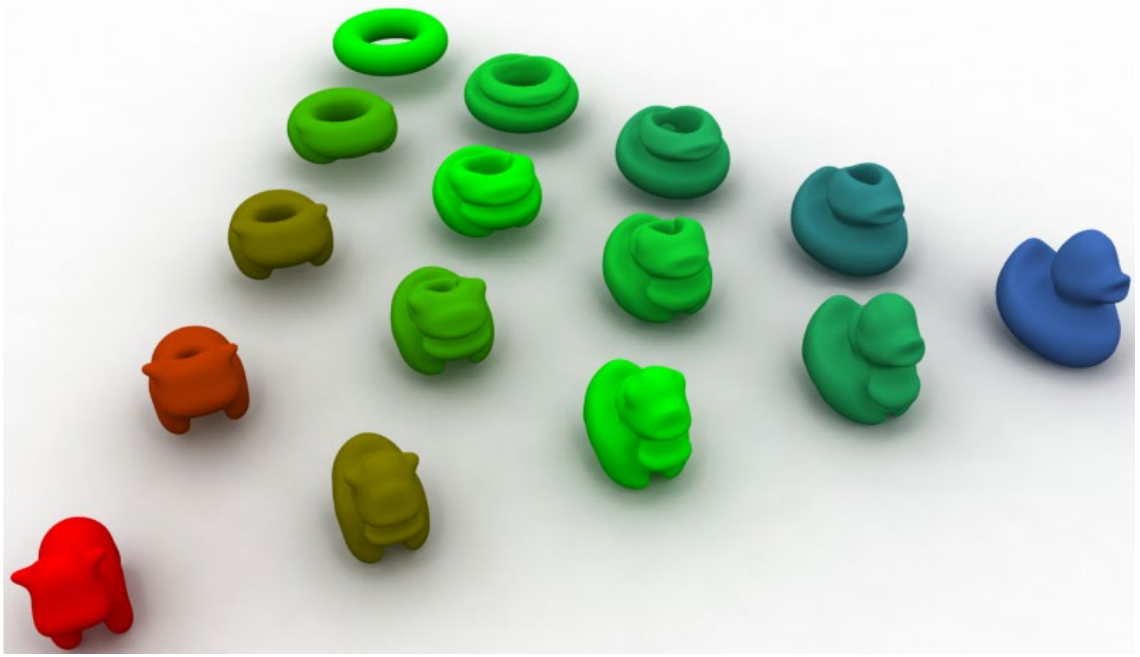
Using $\sum_j M_{s,ij} = p_i$, we have $\sum_j \xi_{s,ij}\ a_{s,i}^{1/\lambda_s} = p_i$ so $a_{s,i} = \left(\frac{p_i}{\sum_j \xi_{s,ij}}\right)^{\lambda_s}$

Using $\Pi_s a_s = 1$, we have $p_i = \Pi_s \left(\sum_j \xi_{s,ij}\right)^{\lambda_s}$ or in vector form, $p = \Pi_s (\xi_s 1)^{\lambda_s}$

Regularized Wasserstein Barycenters

- Generalized to compute displacement interpolation and barycenters

  - $b_s^{(0)} = 1 \ \forall s$
  - for $\ell = 0 \ \ldots L$
    - $a_s^{(\ell)} = \frac{p_s}{K\, b_s^{(l-1)}} \quad \forall s$
    - $p(\lambda) = \prod_s \left(K^T a_s^{(\ell)}\right)^{\lambda_s}$
    - $b_s^{(\ell)} = \frac{p(\lambda)}{K^T\, a_s^{(\ell)}} \quad \forall s$



(notations changed) (again) -> $K = \xi = \exp\left(-\frac{C}{\epsilon}\right)$

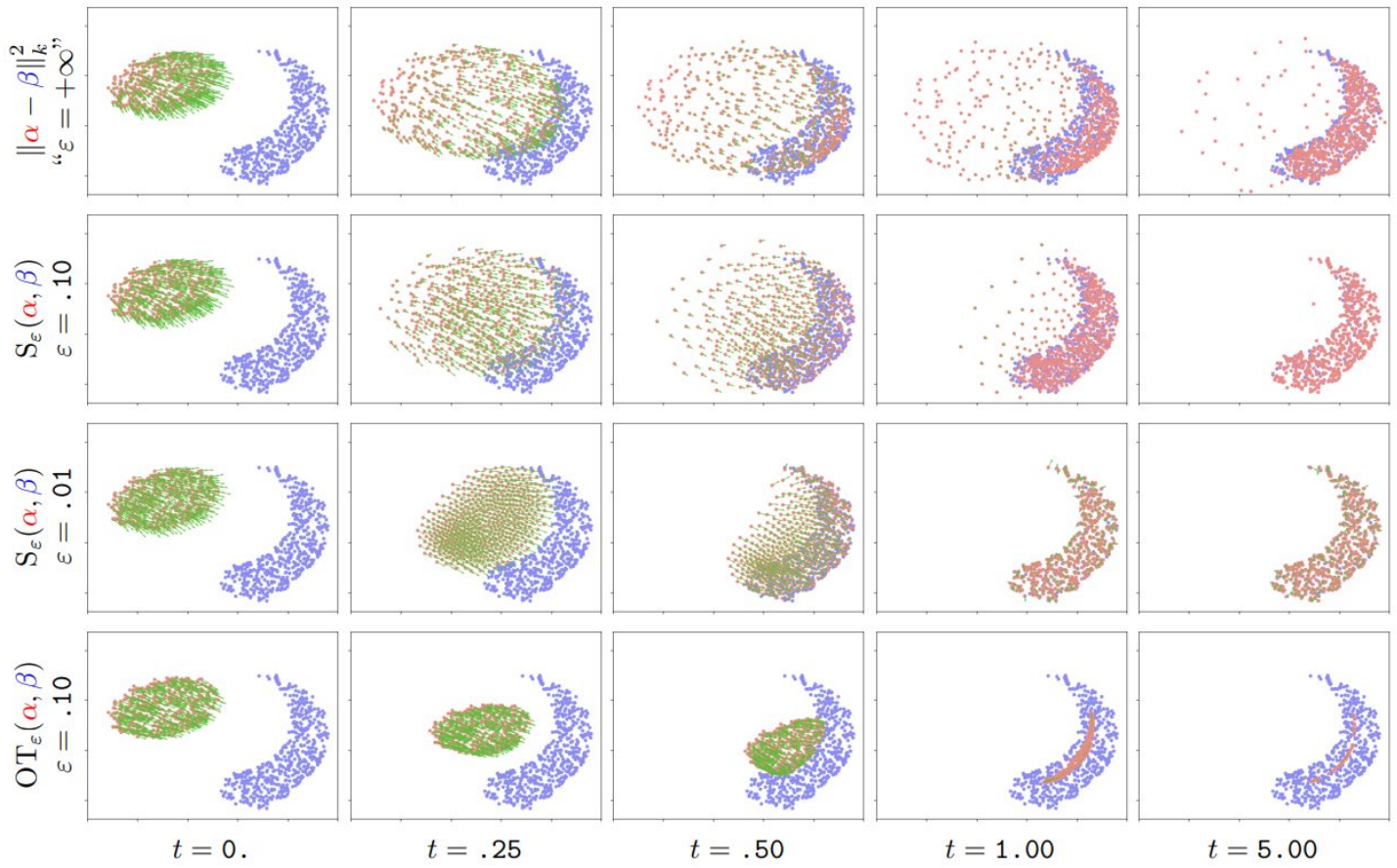
Sinkhorn divergences

- Issue:
  - $W_\epsilon(f, f) \neq 0$
- Instead, consider:

$$\widetilde{W}_\epsilon(f, g) = W_\epsilon(f, g) - \frac{1}{2} W_\epsilon(f, f) - \frac{1}{2} W_\epsilon(g, g)$$

- $\widetilde{W}_\epsilon(f, f) = 0$ by construction
- Better behavior, sharper barycenters
- State of the art GPU implem: GeomLoss

Sinkhorn divergences



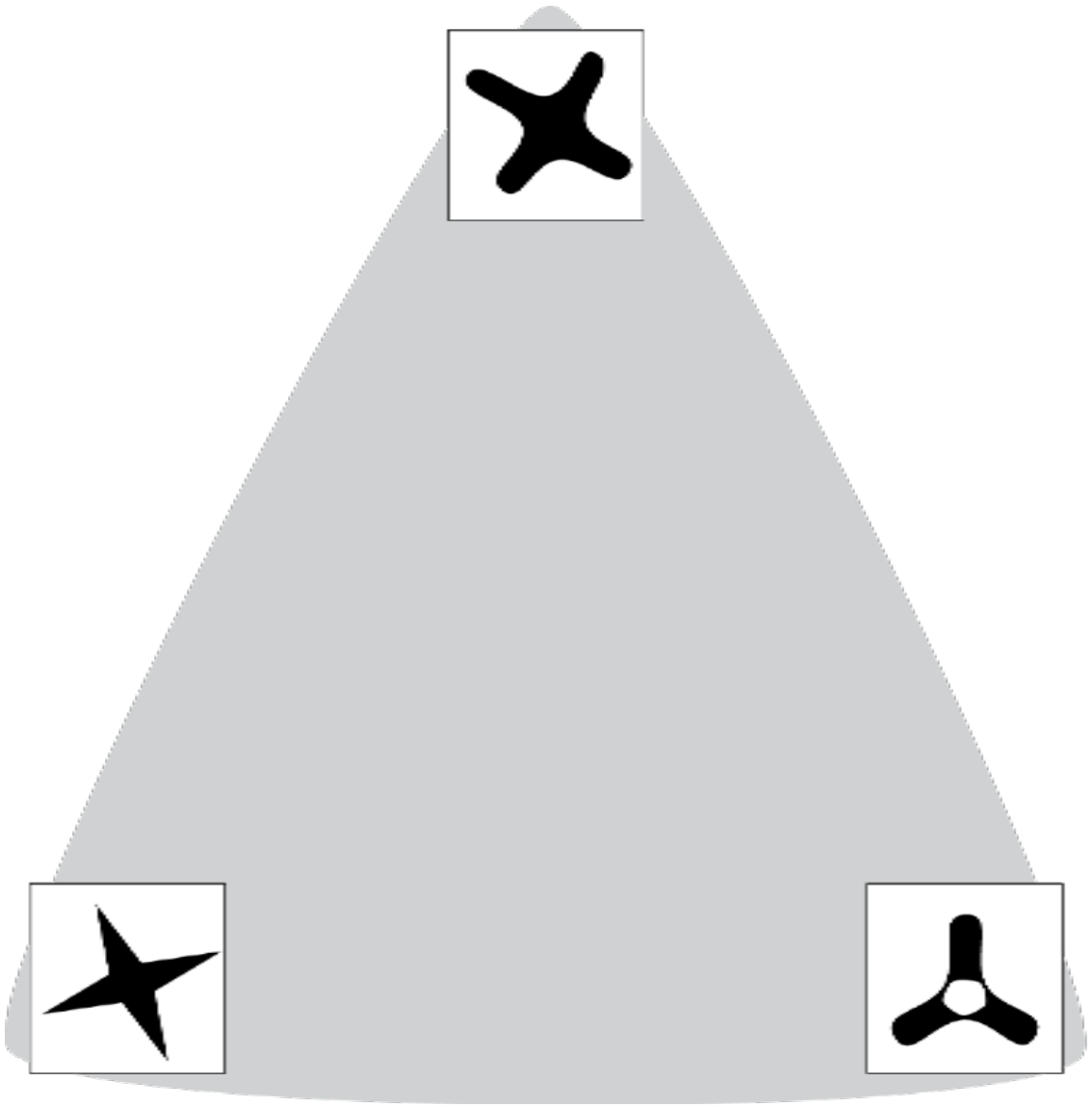
Wasserstein Barycentric Coordinates: Histogram Regression Using Optimal Transport

N. Bonneel, G. Peyré, M.Cuturi

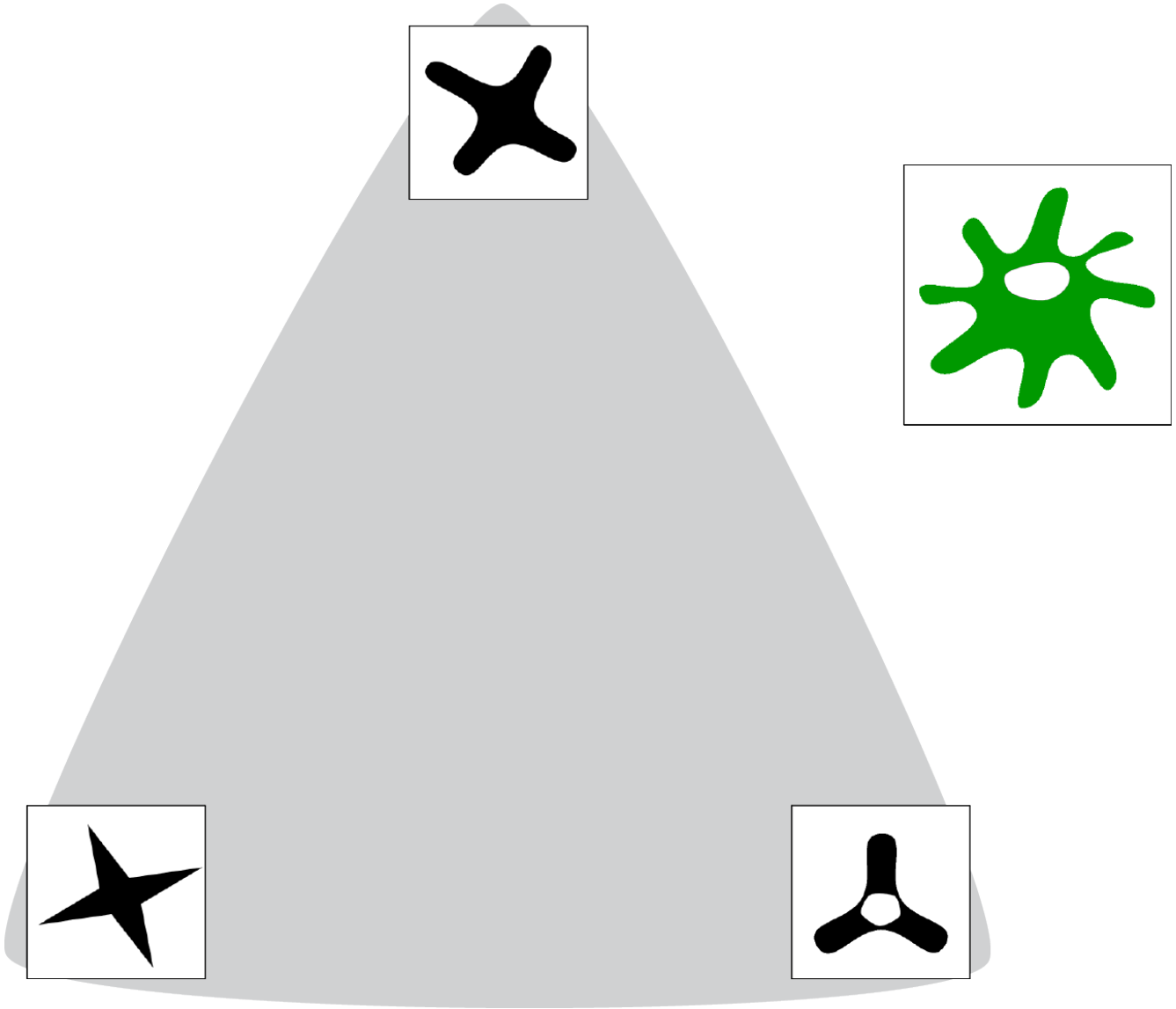
SIGGRAPH 2016

Entropy-regularized OT is nicely differentiable !

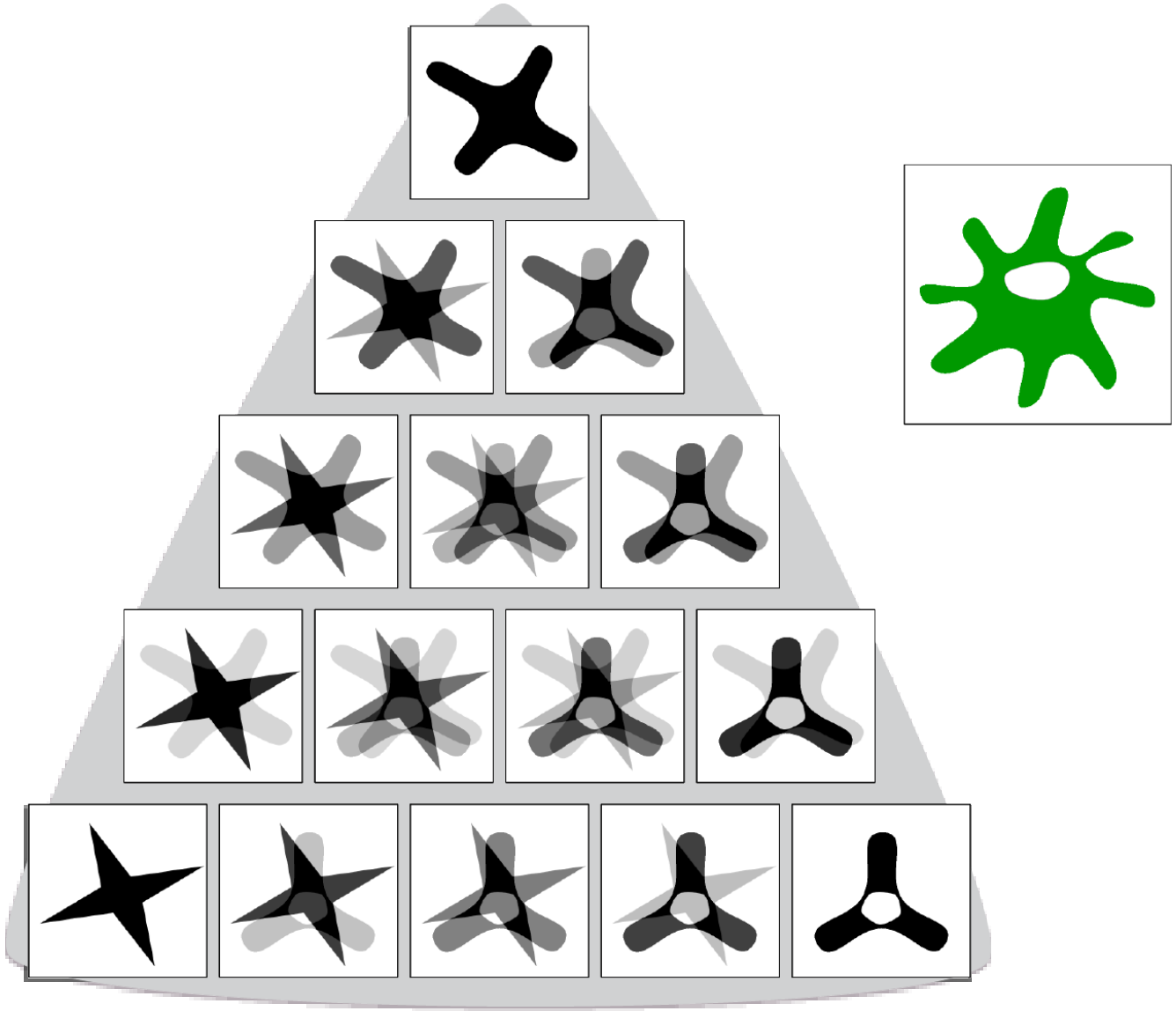
Barycentric coordinates



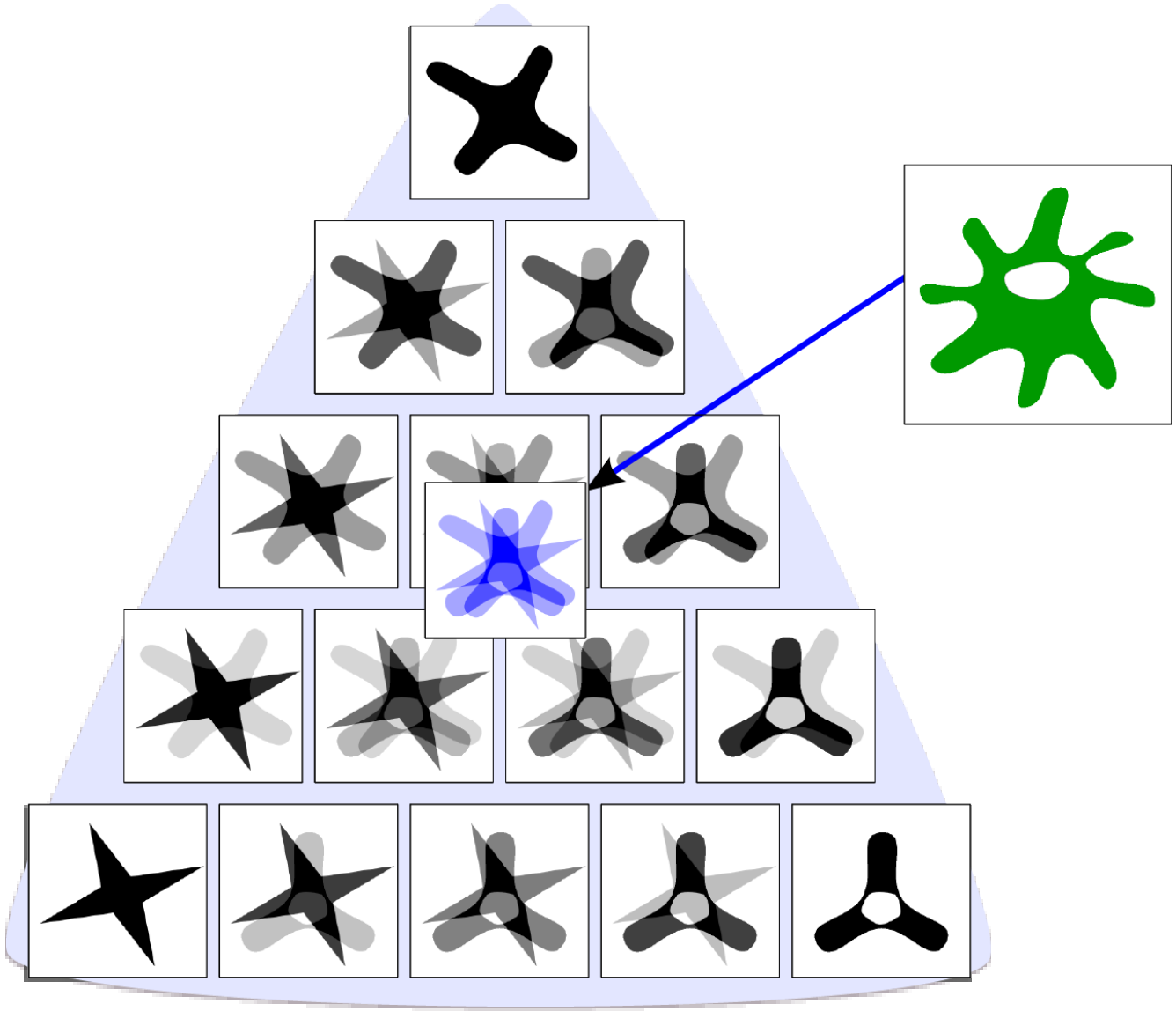
Barycentric coordinates



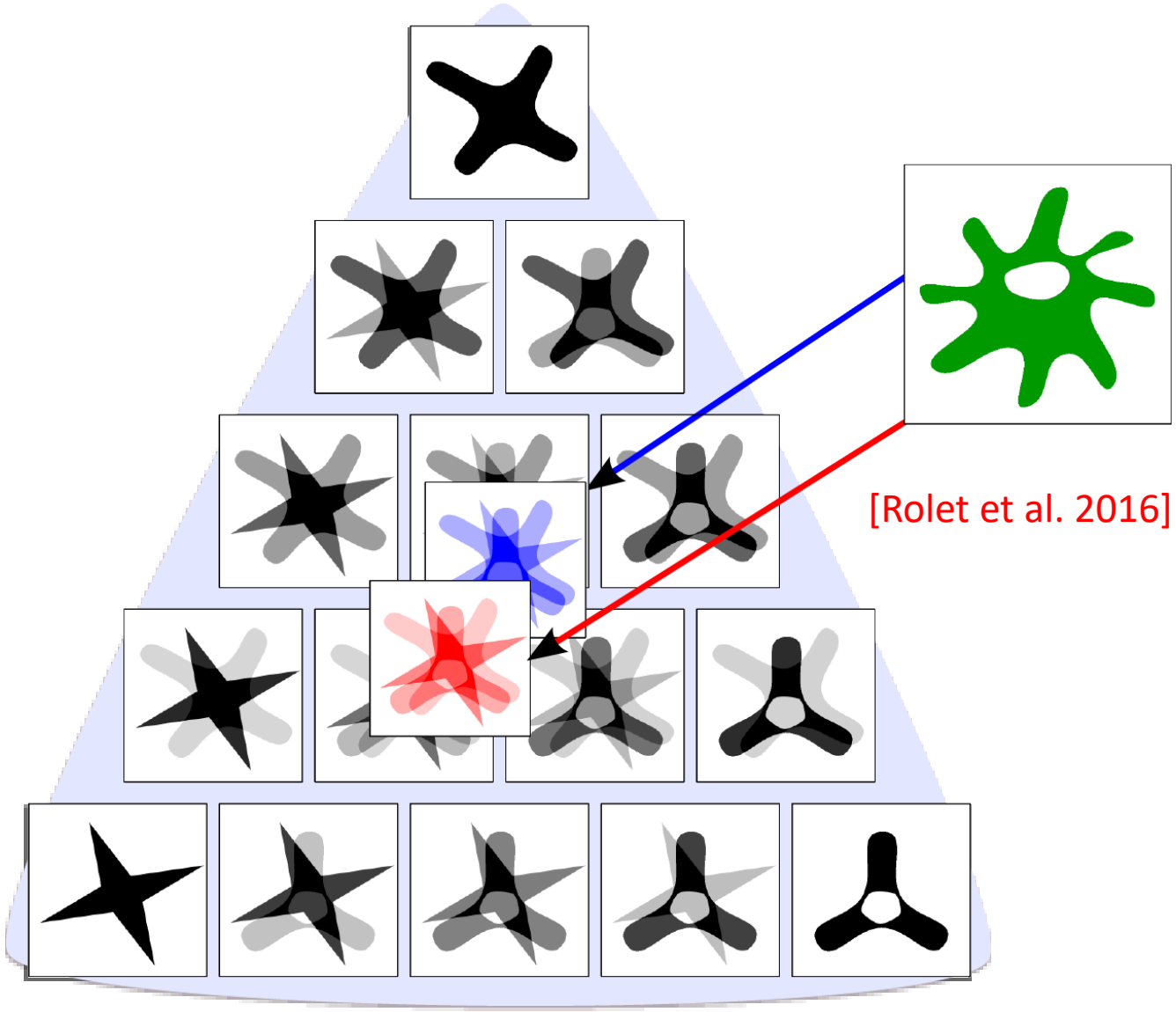
Barycentric coordinates



Barycentric coordinates

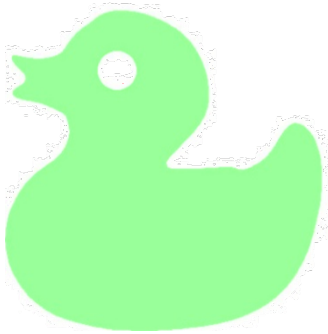


Barycentric coordinates

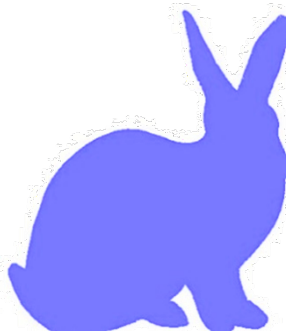


[Rolet et al. 2016]

Optimal Transport



$t = 0$



$t = 1$

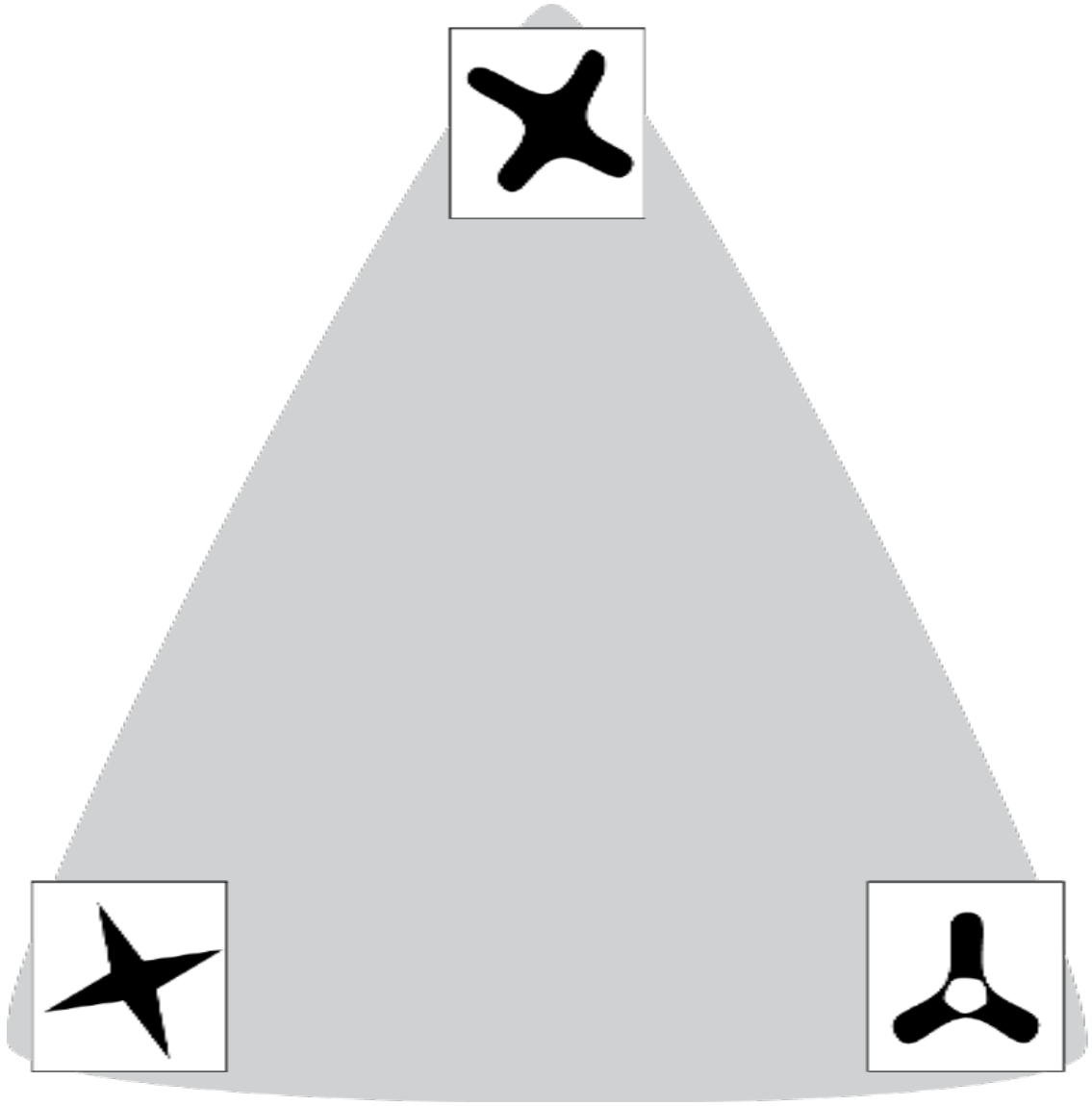
Optimal Transport

$t = 0$ $t = 1$

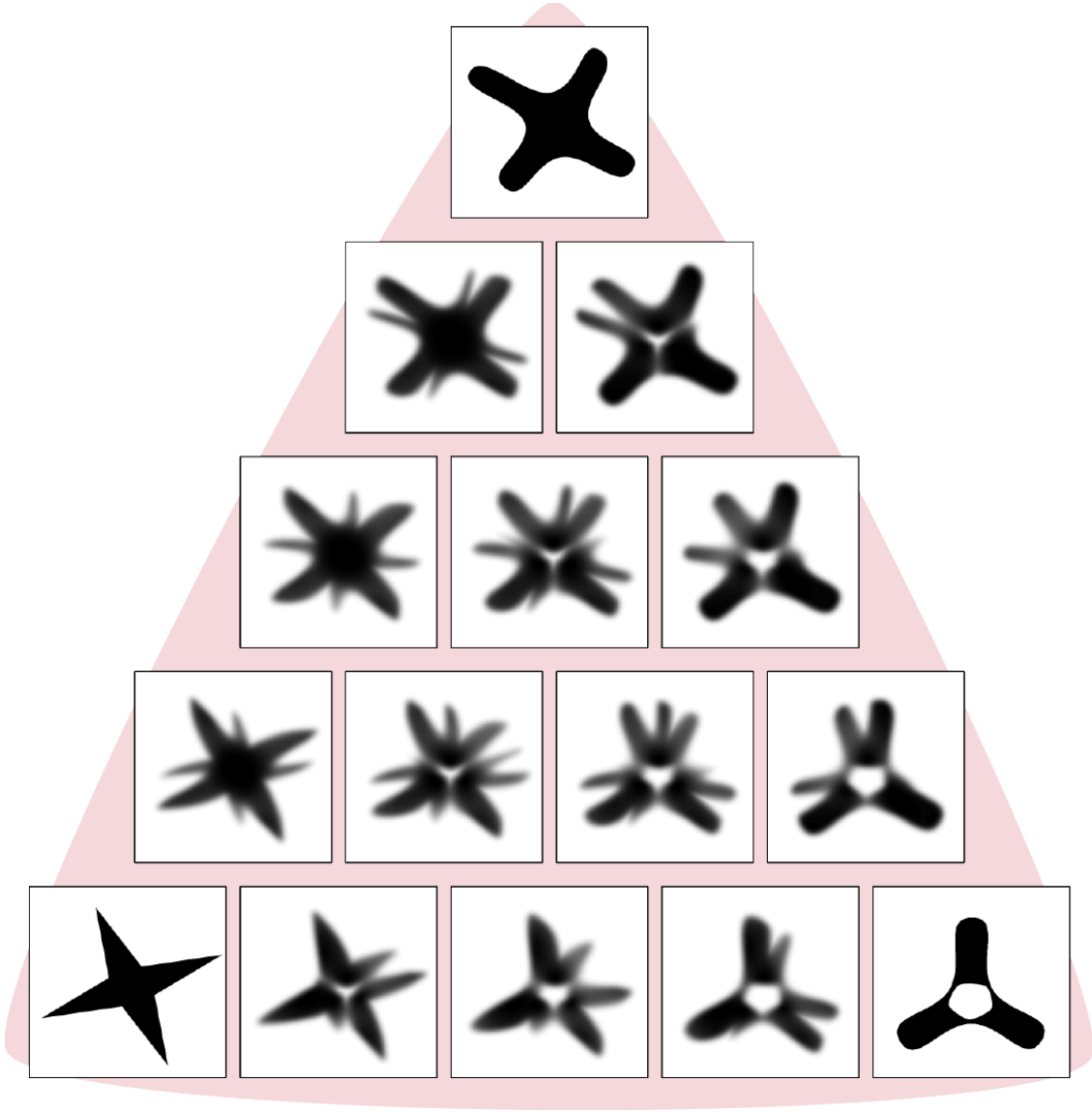
$$\mathrm{W}(f,g) = \min \sum\sum \left\| x_i - x_j \right\|^2 m_{ij}$$

$$\text{s.t.} \quad m_{ij} \geq 0 \;;\; \textstyle\sum_i m_{ij} = g(x_j) \;\;;\;\; \sum_j m_{ij} = f(x_i)$$

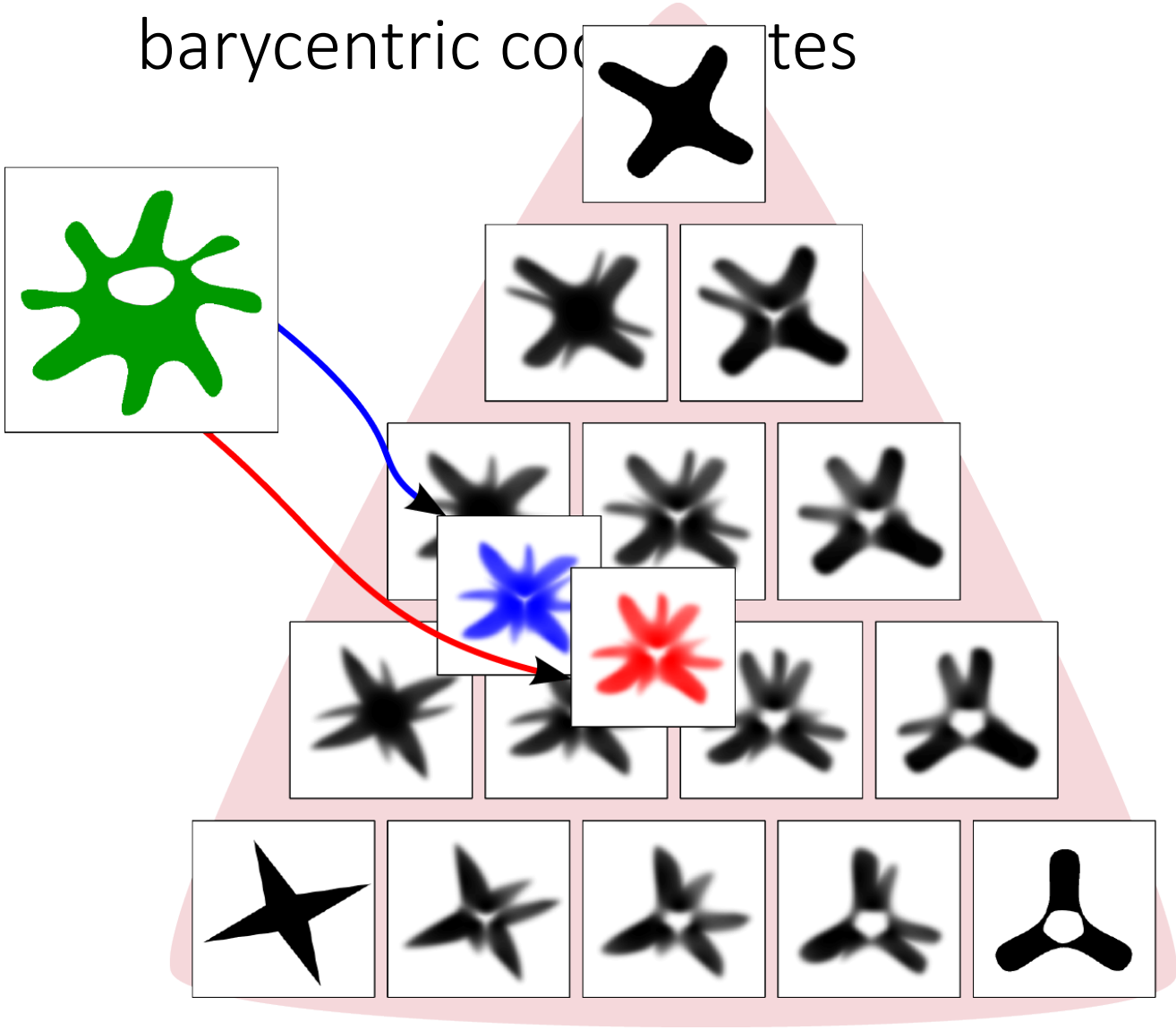
Optimal Transport

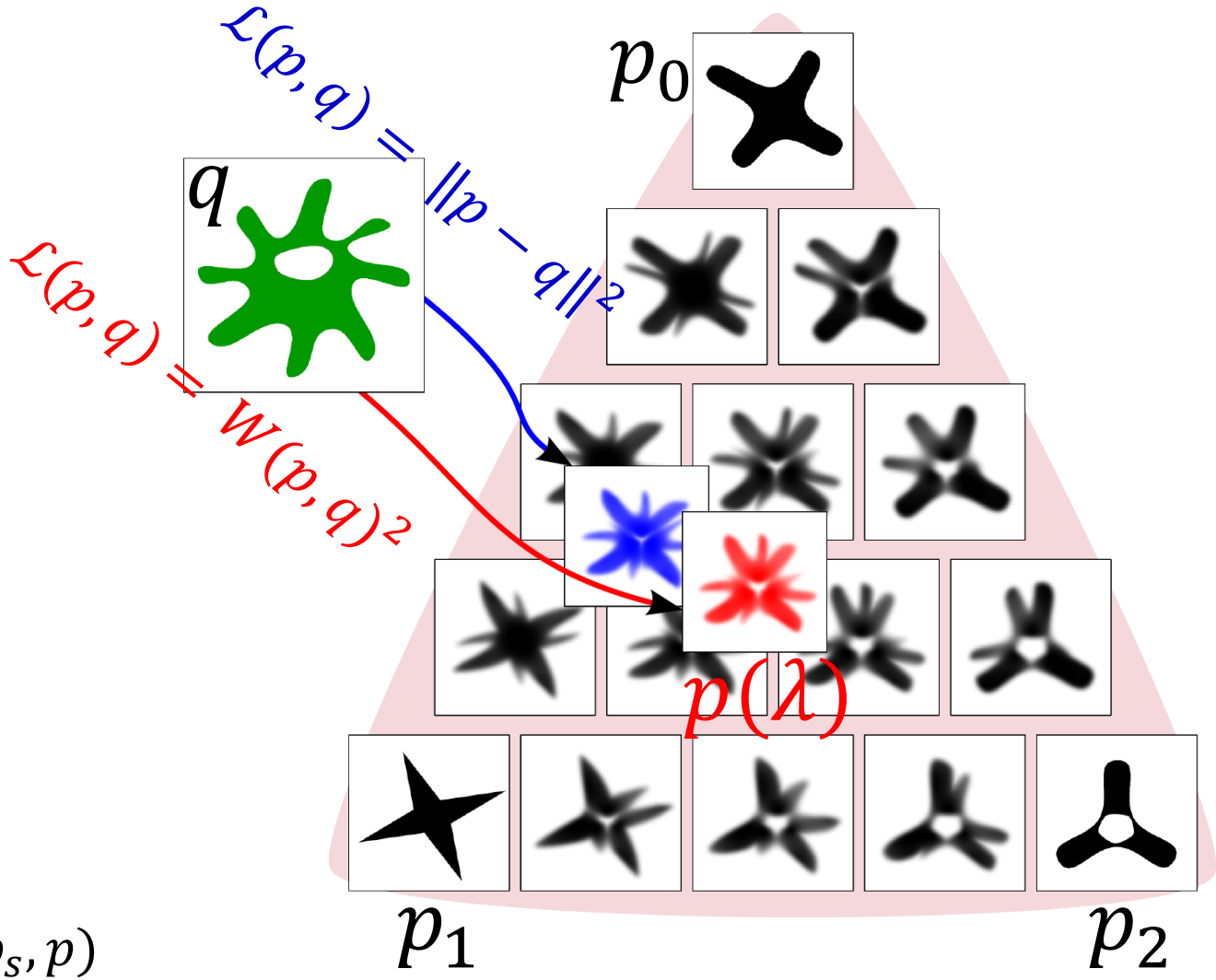


Optimal Transport



Optimal transport barycentric co tes





Formally:

$$\min_{\lambda} \mathcal{L}(p(\lambda), q)$$
$$st. \sum \lambda_i = 1, \lambda_i \geq 0$$

with $p(\lambda)$ a Wasserstein barycenter:

$$p(\lambda) = \text{argmin}_p \sum_s \lambda_s W^2(p_s, p)$$

and $\mathcal{L}(p,q)$ a cost function :

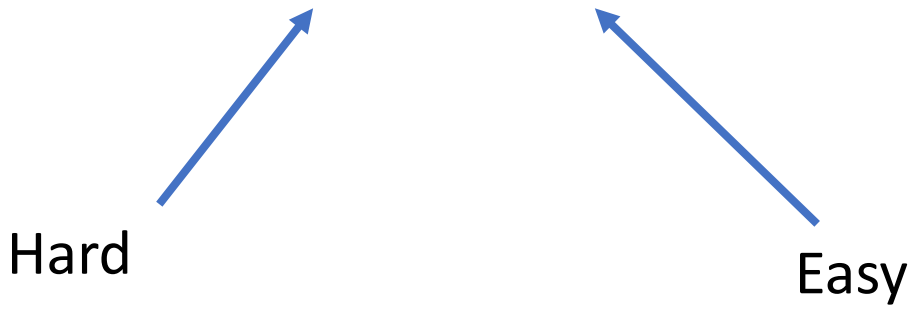
$$\mathcal{L}(p,q) = W(p,q)\ , \|p-q\|_2{}^2\ , \|p-q\|_1\ \ ,\ KL(p,q)$$

Method

$$\min_{\lambda} \mathcal{E}(\lambda) = \mathcal{L}(p(\lambda), q)$$

- We minimize using L-BFGS

- We use $\nabla\mathcal{E}(\lambda) = [\partial p(\lambda)]^T (\nabla\mathcal{L}(p(\lambda), q))$



Idea

- $[\partial p(\lambda)]^T$ by deriving the Sinkhorn algorithm [Solomon et al. 2015]

- To compute $p(\lambda)$ given λ, Sinkhorn iterations read:
  - $b_s^{(0)} = 1 \ \ \forall s$
  - for $\ell = 0 \ \ldots L$
    - $a_s^{(\ell)} = \frac{p_s}{K\, b_s^{(l-1)}} \quad \forall s$
    - $p(\lambda) = \ \prod_s \left(K^T a_s^{(\ell)}\right)^{\lambda_s}$
    - $b_s^{(\ell)} = \ \frac{p(\lambda)}{K^T\, a_s^{(\ell)}} \qquad \forall s$

Idea

- Automatic differentiation: given an iterative algorithm, apply the chain rule:
  - If

$$p^{(\ell+1)}(\lambda) = f\left(p^{(\ell)}(\lambda), \lambda\right)$$

  - Then

$$\frac{\partial p^{(\ell+1)}}{\partial \lambda} = \frac{\partial f}{\partial p^{(\ell)}} \frac{\partial p^{(\ell)}}{\partial \lambda} + \frac{\partial f}{\partial \lambda}$$

$q^{(\ell+1)}$ $q^{(\ell)}$

- We similarly compute the adjoint
- ...formulas in the paper

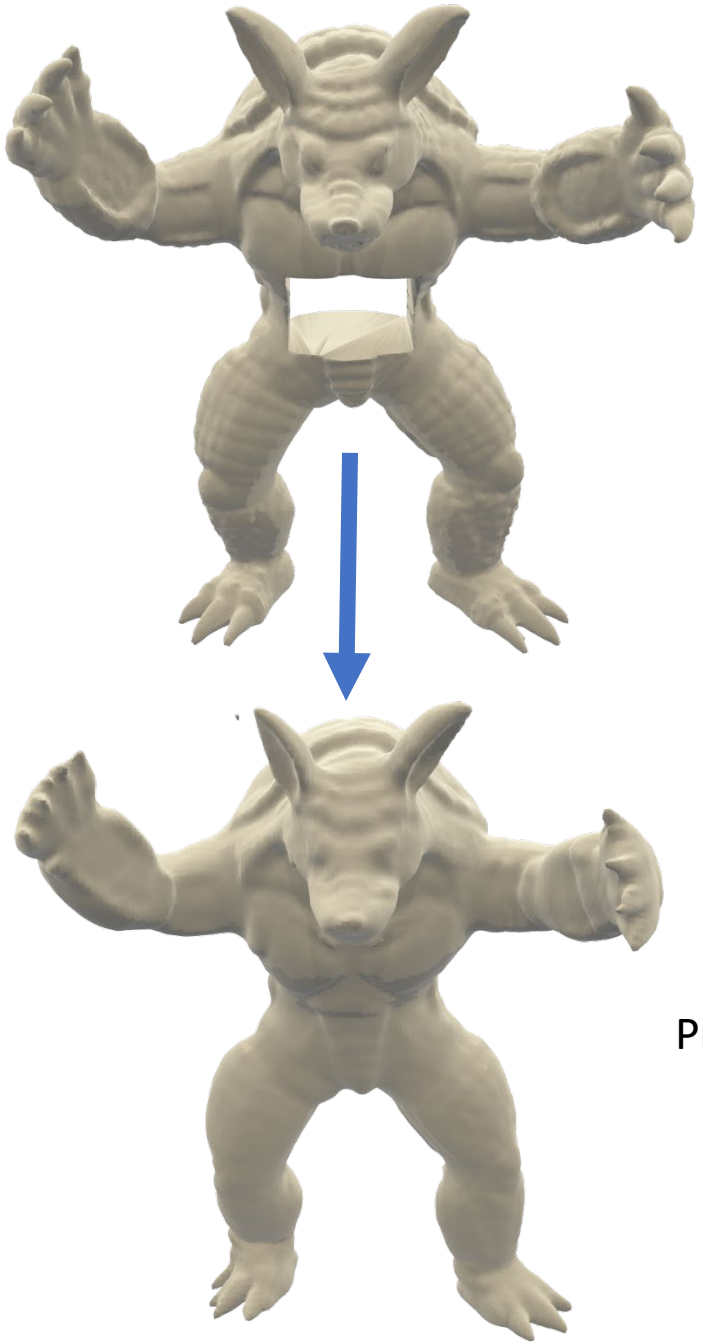
Gradient computation

- We obtain:
  - $q_s = 0 \; ; r_s = 0 \;\; \forall s$
  - $g \leftarrow \nabla \mathcal{L}(p(\lambda), q) \odot p(\lambda)$
  - for $\ell = L \;\; \ldots 1$
    - $q_s \leftarrow q_s + \left\langle \log K^T a_s^{(\ell)} \, , g \right\rangle \qquad \forall s$
    - $r_s \leftarrow -K^T \left(K \left(\frac{\lambda_s g - r_s}{K^T a_s^{(\ell)}} \right) \odot \frac{p_s}{\left(K b_s^{(\ell-1)} \right)^2} \right) \odot b_s^{(\ell-1)} \qquad \forall s$
    - $g \leftarrow \sum_s r_s$

$\nabla \mathcal{E}(\lambda)$

Applications

Database



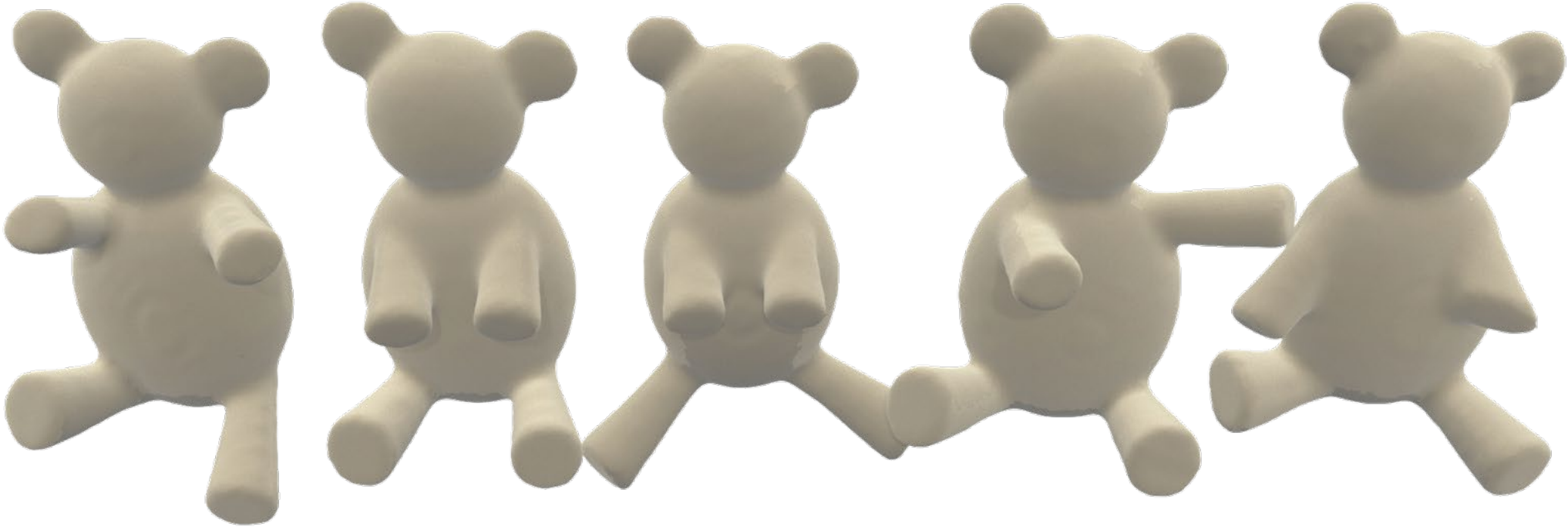
Input

Projection

Euclidean

Wasserstein

Database

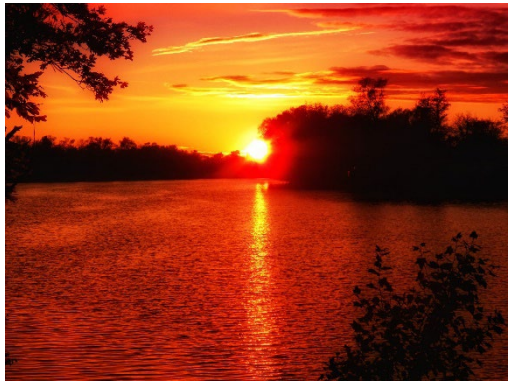


Input

Projection

Applications

3E-6



6E-6

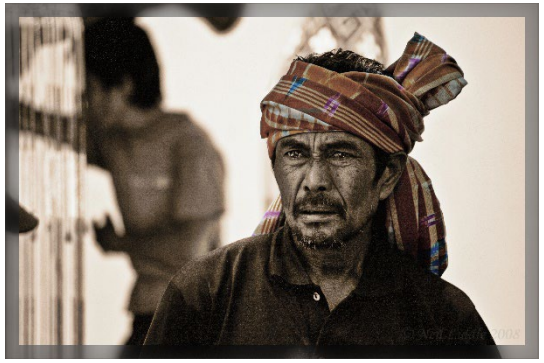
0.23

0.77

Database

Projection

Input



Database

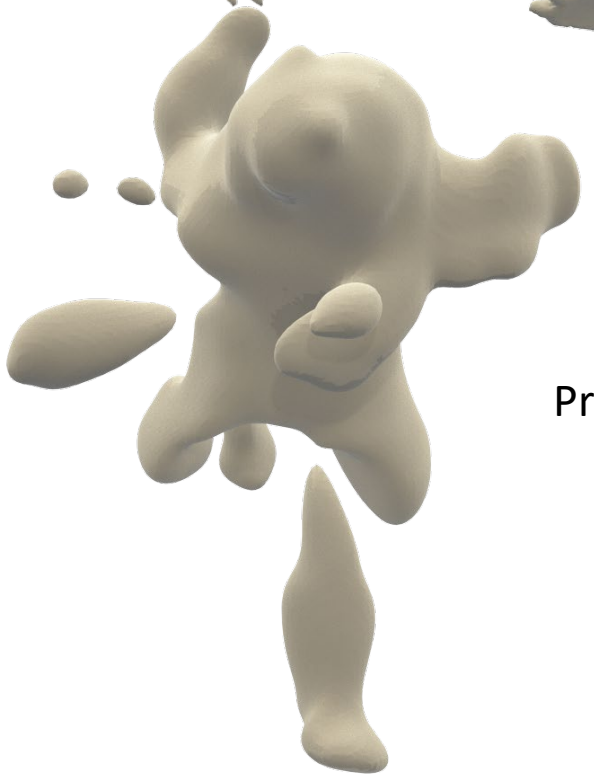
Projection

Flickr results for “Autumn”

Projection

Database

Input



Projection

Conclusion

- Notion of barycentric coordinates useful for computer graphics
- Tractable computations
  - Barycenter gradient requires 2x convolutions w.r.t to barycenter alone
  - Relatively large memory footprint
  - Takes between seconds to minutes
- Easy to implement
  - Code available: http://liris.cnrs.fr/~nbonneel/WassersteinBarycentricCoordinates/