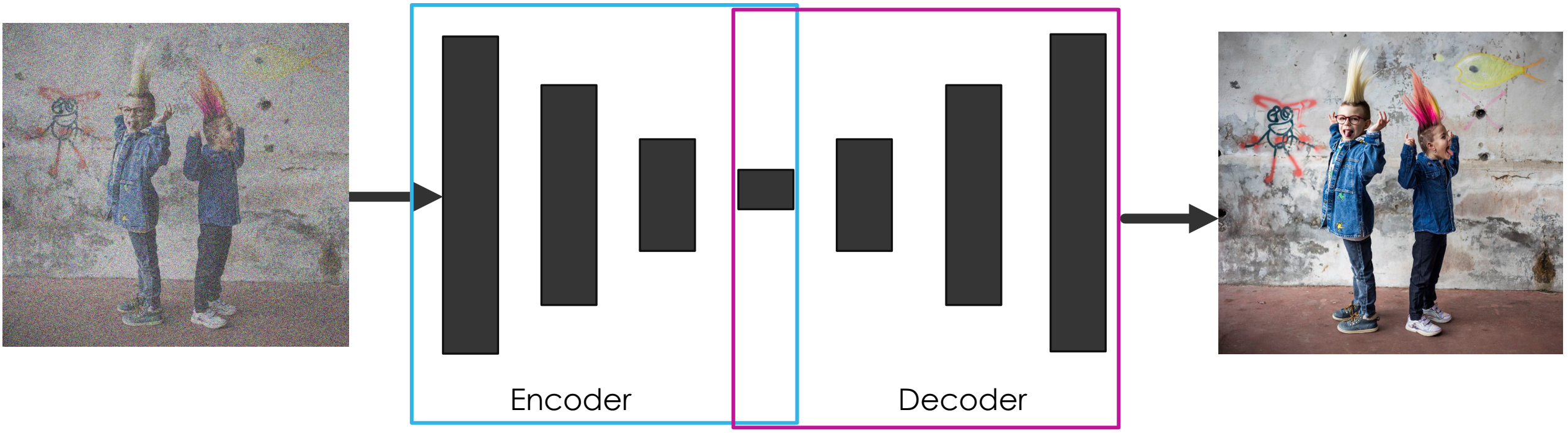
# Diffusion models & Optimal transport

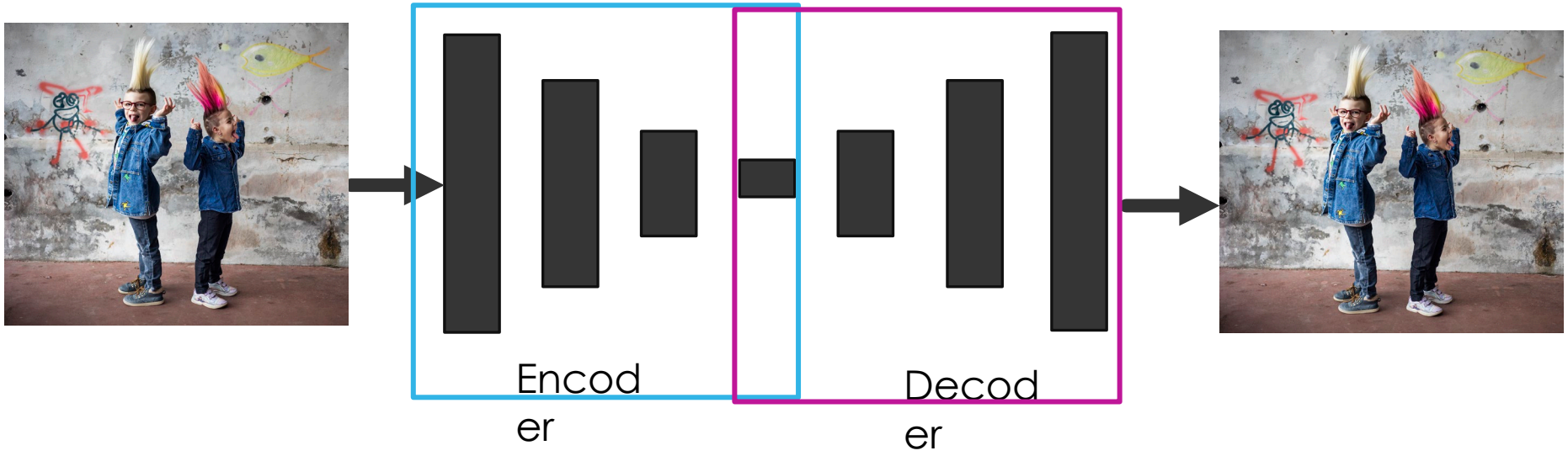
# Diffusion models

# Restoration via Auto-encoders




Bank of 3x3 convolutional filters + nonlinearity + downsampling / upsampling

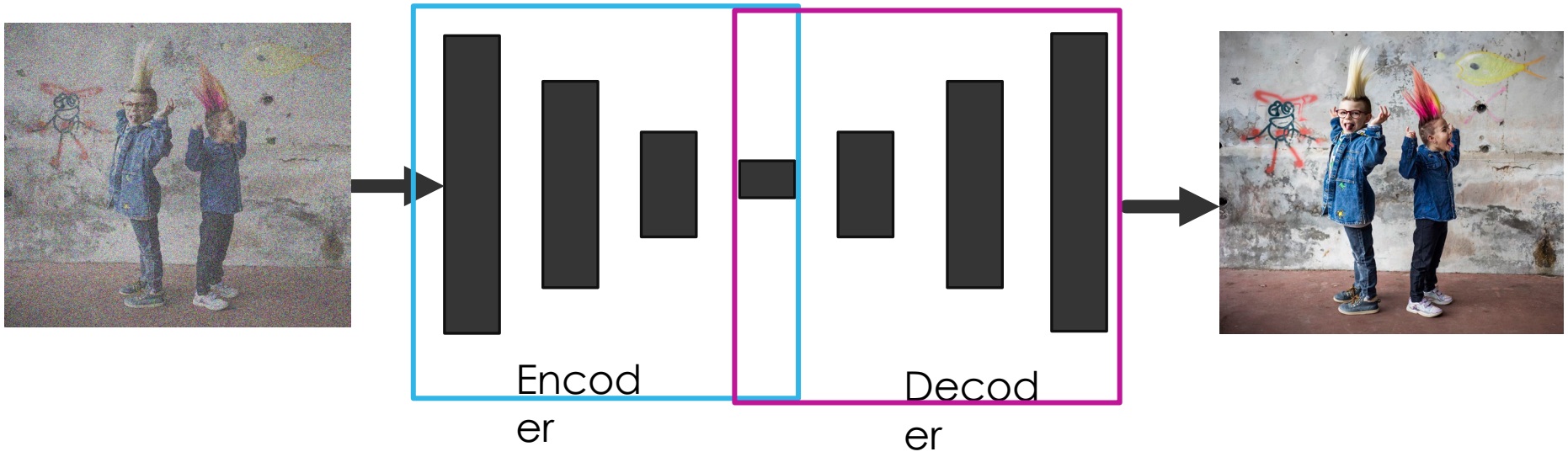
# Restoration via Auto-encoders




Training:

- For a given image, find convolution weights that minimizes differences between input and output images
- Assumes the use of many (many) input images : consider the sum of individual losses
- Find derivatives with automatic differentiation
- Stochastic gradient descent (for example), using small batches of images, i.e., a small random subset of input database

# Restoration via Auto-encoders

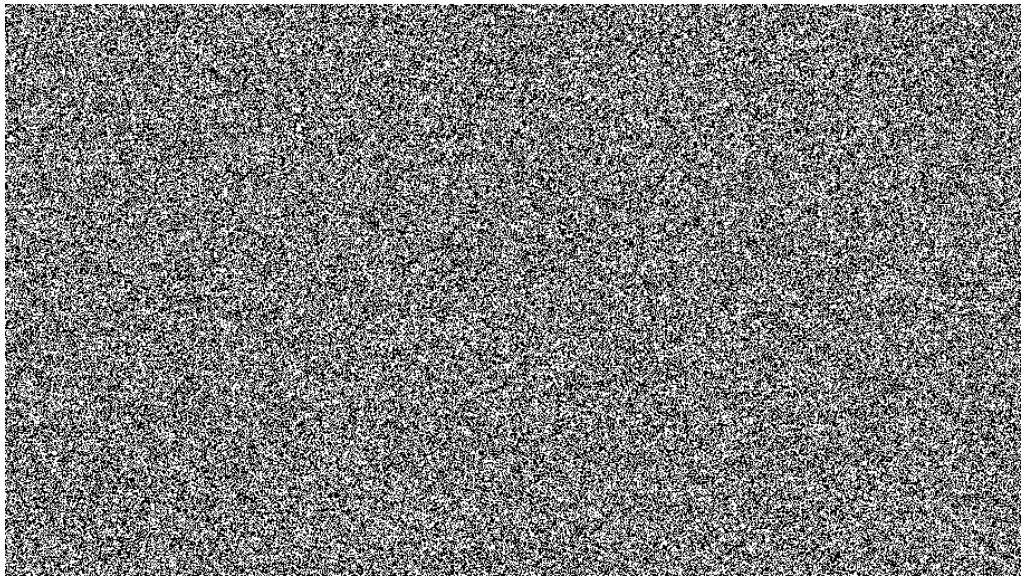



- Given trained network, encoder allows to compress input images into a small feature vector (“code”)
- Decoder recovers the image based on feature vector
- Useful for restoration ; Decoder could be used (with modest success) for generation

# Diffusion models for image generation

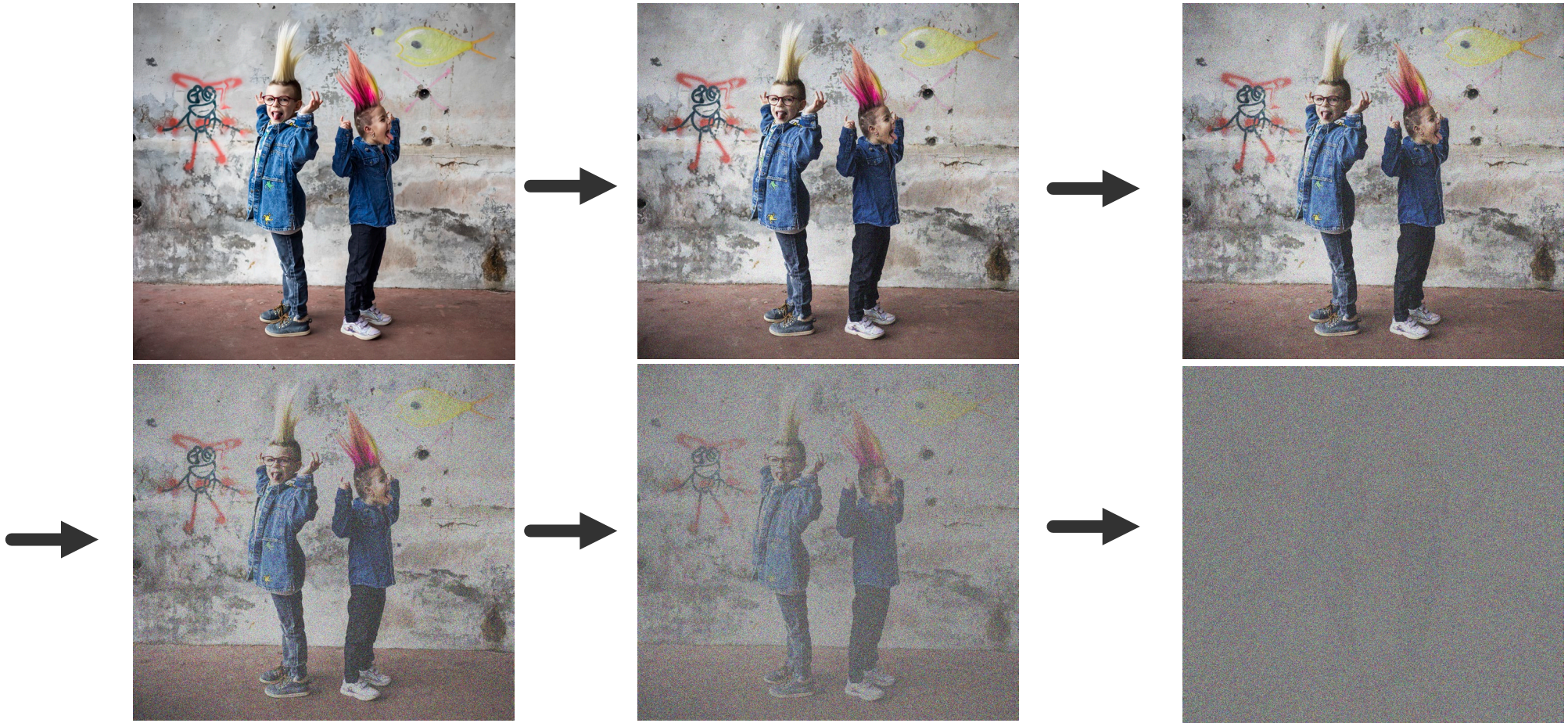
Add noise at training time

Denoise at synthesis time



# Diffusion models : training

- Idea: at training time, progressively add noise



# Diffusion models: training

Image 1 trajectory

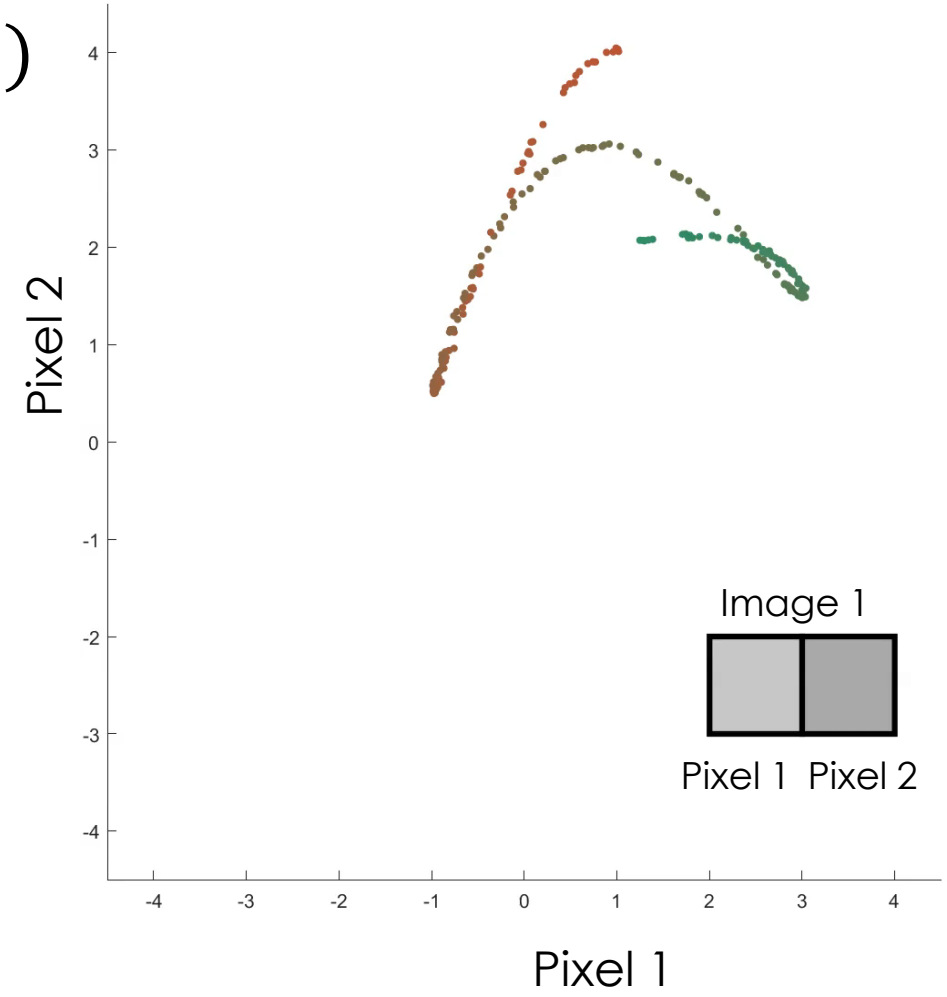
- From distribution of images to $\mathcal{N}(0,1)$
  - Remove (almost) all information
  - Repeat:

$$I^{t+1} = \sqrt{\alpha_t}\, I^t + \sqrt{1-\alpha_t}\, \varepsilon_{t\to t+1}$$

with $\varepsilon_{t\to t+1} \sim \mathcal{N}(0,1)$

$\alpha_t$ some parameter (could be constant)

- Converges to $I^{\infty} \sim \mathcal{N}(0,1)$



# Diffusion models: training

- Known distribution at time $t$
  - If you know $I^0$:

    $I^t \sim \mathcal{N}(\mu_t, \sigma_t^2)$

    with $\mu_t = \sqrt{\overline{\alpha_t}}\, I^0$, $\sigma_t^2 = (1 - \overline{\alpha_t})\,\mathrm{Id}$

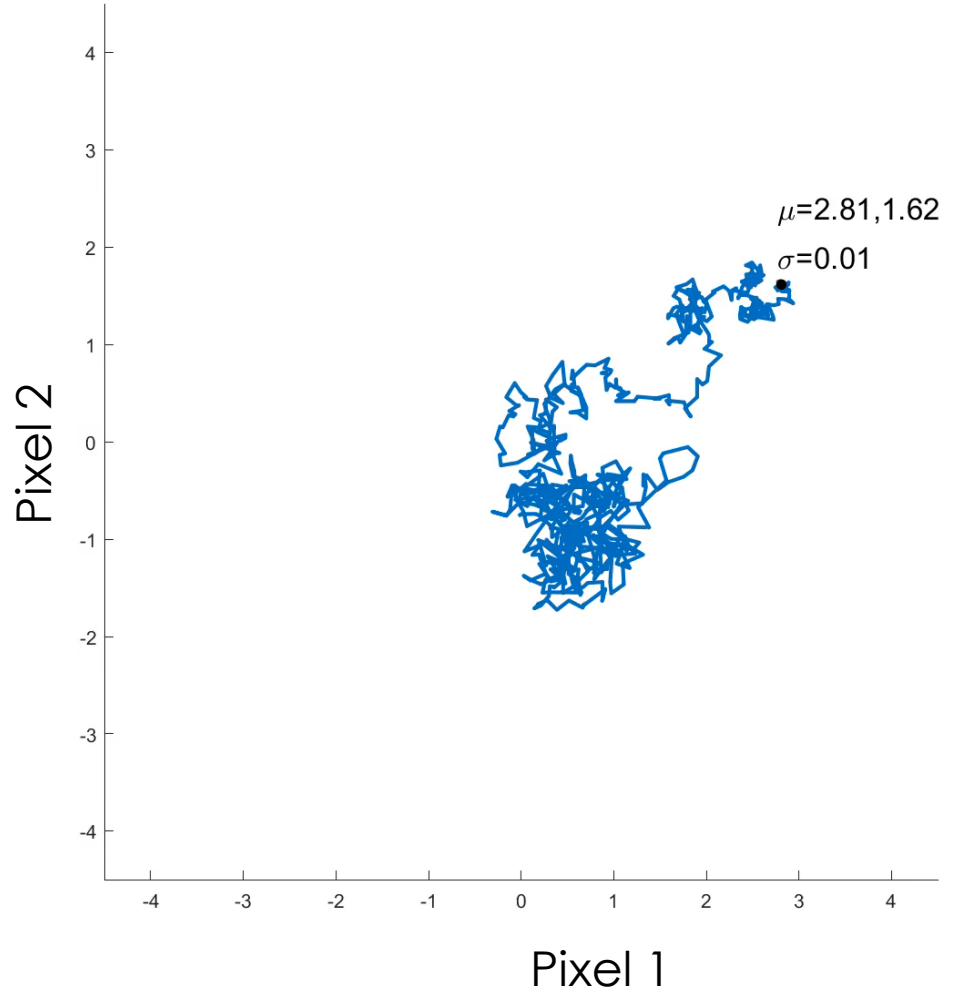
    and $\overline{\alpha_t} = \prod_{i=1}^{t} \alpha_i$

    $$I^t = \sqrt{\overline{\alpha_t}}\ I^0 + \sqrt{1 - \overline{\alpha_t}}\, \varepsilon_{0\to t}$$

    with $\varepsilon_{0\to t} \sim \mathcal{N}(0,1)$

  - You know $I^0$, the noise-free image, at training time (only)
  - Indeed converges to $\mathrm{I}^\infty \sim \mathcal{N}(0,1)$

  - Allows to get $I^t$ at any time $t$ "for free"

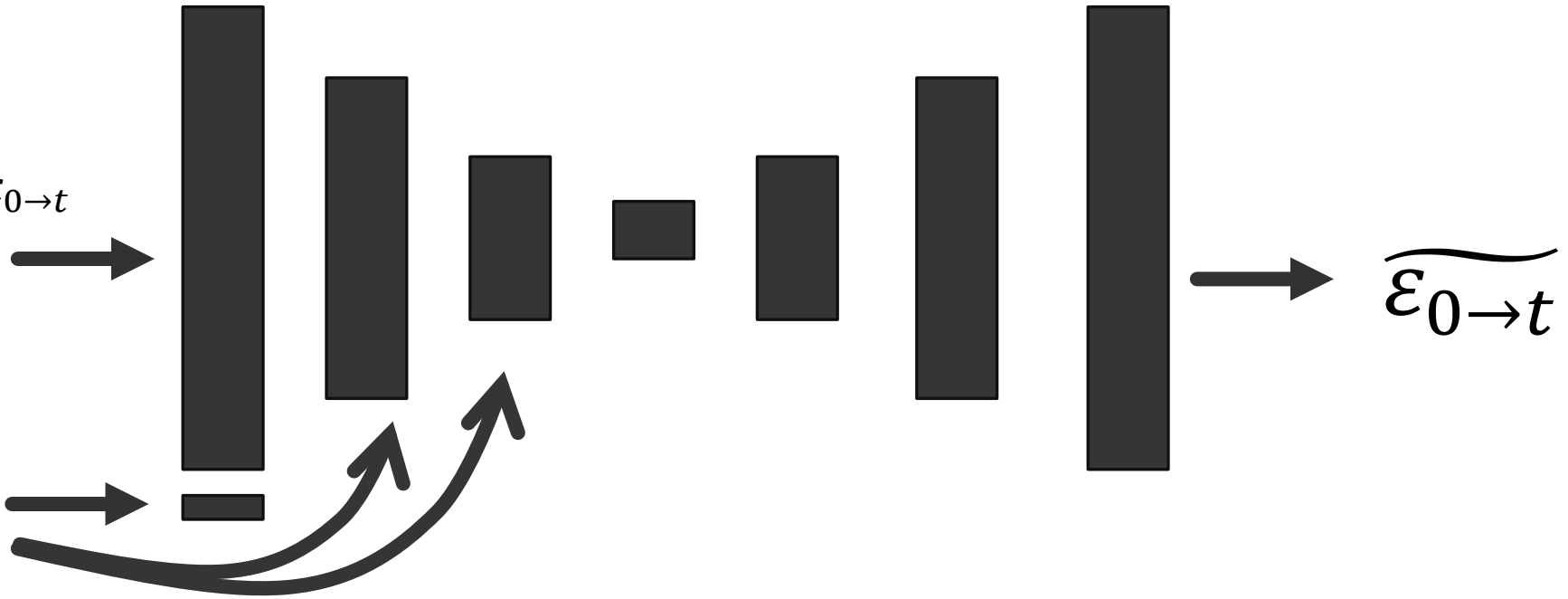


# Diffusion models: training

- Train a network to predict noise:

Loss: $\|\widetilde{\varepsilon_{0\to t}} - \varepsilon_{0\to t}\|^2$

$I^t = \sqrt{\overline{\alpha_t}}\, I^0 + \sqrt{1-\overline{\alpha_t}}\, \varepsilon_{0\to t}$




$\widetilde{\varepsilon_{0\to t}}$

- Feed the network with many noisy images at random times $t \in [0, t_{max}]$

# Diffusion models: inference

- Start with a random noise image at time $t = t_{max}$ :

$$I^{t_{max}} = \underbrace{\sqrt{\overline{\alpha}_{t_{max}}}\, I^0}_{\approx 0} + \underbrace{\sqrt{1-\overline{\alpha}_t}}_{\approx 1}\, \varepsilon_{0 \to t_{max}}$$

$$\varepsilon_{t_{max}} \sim \mathcal{N}(0,1)$$

- Predict noise level

$$I^{t_{max}} = \sqrt{\overline{\alpha}_{t_{max}}}\, I^0 + \sqrt{1-\overline{\alpha}_t}\, \varepsilon_{0 \to t_{max}}$$

# Diffusion models: inference

- Remove the noise

$$I^{t_{max}} = \sqrt{\overline{\alpha}_{t_{max}}}\, I^0 + \sqrt{1-\overline{\alpha}_{t_{max}}}\, \varepsilon_{0 \to t_{max}} \quad \sim \mathcal{N}(0,1)$$

$$\Rightarrow I^0 \approx \frac{I^{t_{max}} - \sqrt{1-\overline{\alpha}_{t_{max}}}\, \widetilde{\varepsilon_{0 \to t_{max}}}}{\sqrt{\overline{\alpha}_{t_{max}}}}$$

In practice, task is too difficult: do it progressively

# Diffusion models: inference

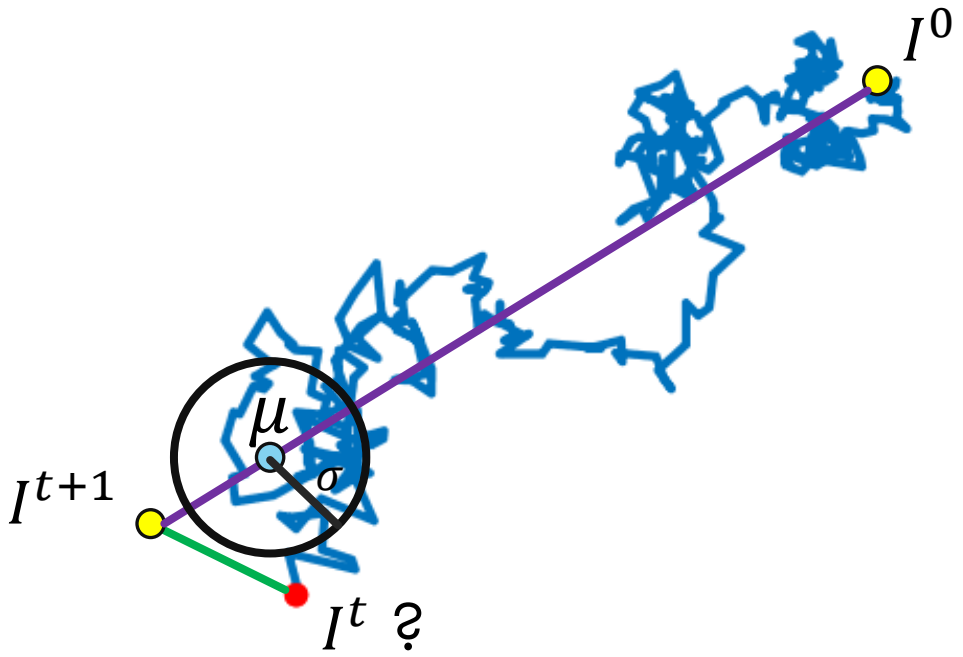
- Remove the noise progressively
  - Given $I^0$ and $I^{t+1}$, $I^t \sim \mathcal{N}(\mu, \sigma^2)$ with

$$\mu = \frac{\sqrt{\alpha_{t+1}}\,(1-\overline{\alpha_t})}{1-\overline{\alpha_{t+1}}}\,I^{t+1} + \frac{\sqrt{\overline{\alpha_t}}\,(1-\alpha_{t+1})}{1-\overline{\alpha_{t+1}}}\,I^0$$

$$\sigma = \frac{1-\overline{\alpha_{t-1}}}{1-\overline{\alpha_t}}(1-\alpha_t)$$

  - Don't know $I^0$
    - Take our previous (poor) estimate:

$$I^0 \approx \frac{I^{t+1} - \sqrt{1-\overline{\alpha_{t+1}}}\;\widetilde{\varepsilon_{0\to t+1}}}{\sqrt{\overline{\alpha_{t+1}}}}$$



# Diffusion models: inference

- Remove the noise progressively
  - We get

$$I^t = \frac{I^{t+1} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha_t}}}\widetilde{\varepsilon_{0\to t+1}}}{\sqrt{\alpha_t}} + \mathcal{N}(0,\sigma)$$

# Diffusion models

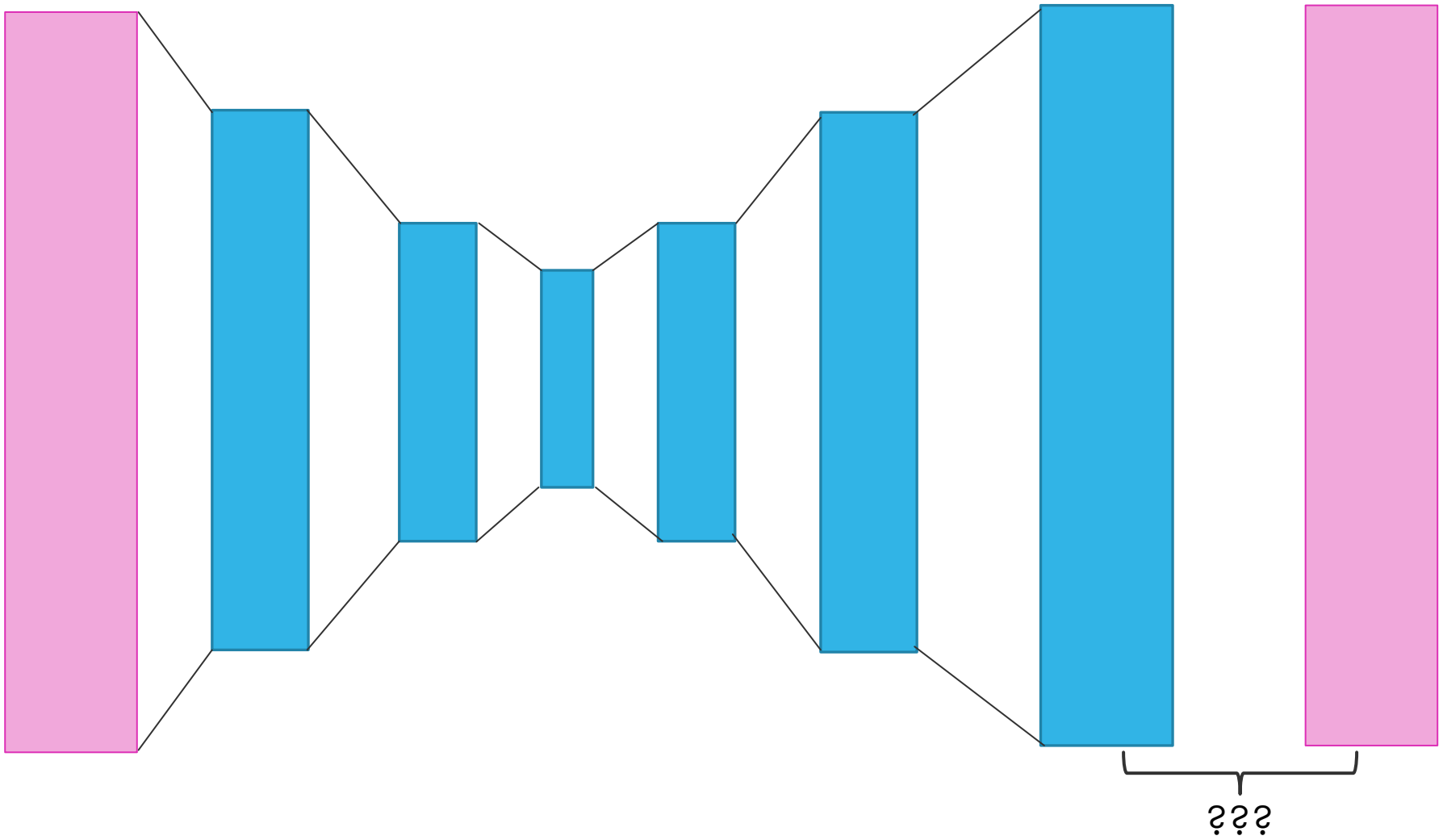
- Now used by most image generative models
  - Often with conditioning by text
  - Often variants, such as “Latent diffusion models”/Stable diffusion
- Has superseded most other approaches (GANs, VAE..)



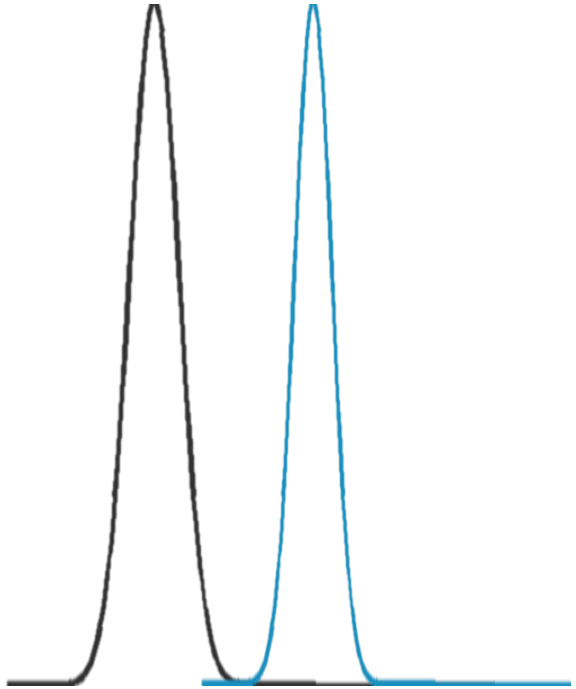
# Optimal Transport Principles for Computer Graphics and Machine Learning

Nicolas Bonneel

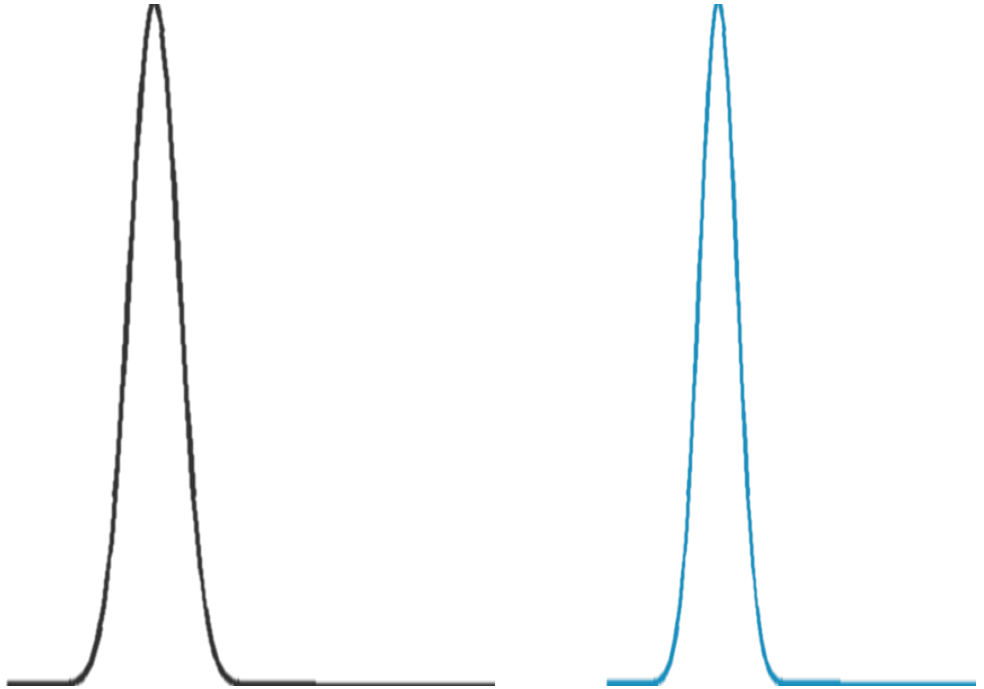
# Motivation in neural networks



# How to compare functions ?

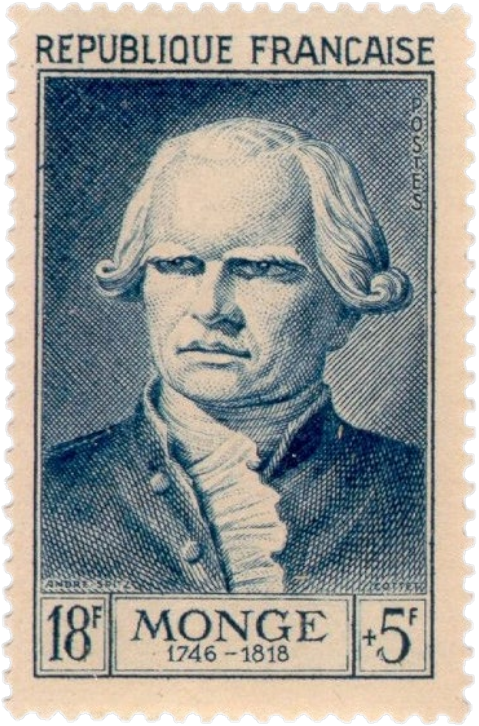


# How to compare functions ?



f g

*Mémoire sur la théorie des déblais et des remblais (1781)*




# MÉMOIRE
## SUR LA
# THÉORIE DES DÉBLAIS
## ET DES REMBLAIS.

Par M. MONGE.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'ensuit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, & le prix du transport total sera un *minimum*.

- Nobel prize in economy in 1975, for his "contribution to the theory of resources allocation"

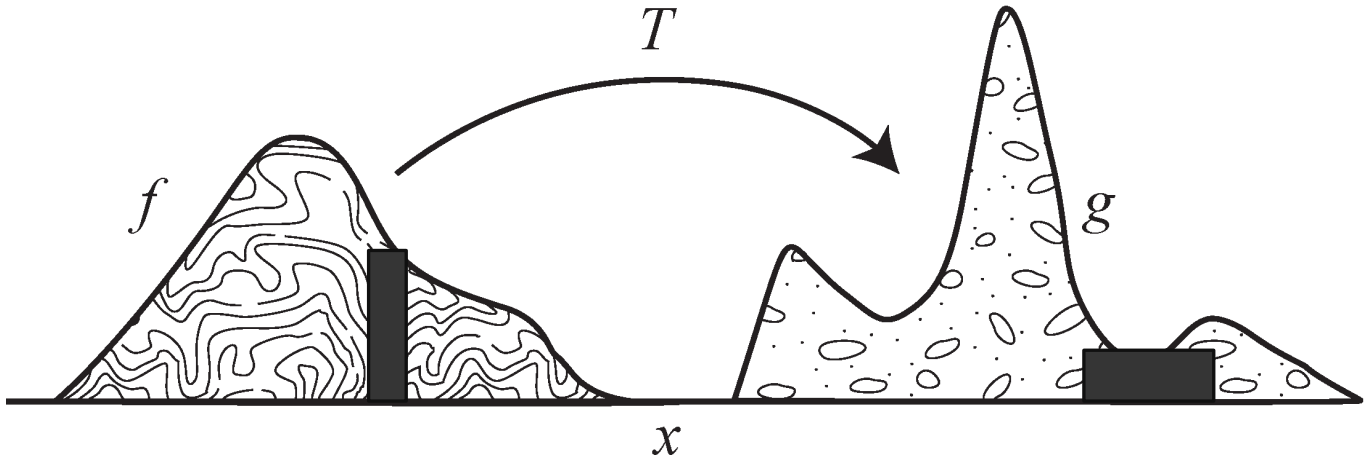
Leonid Kantorovich

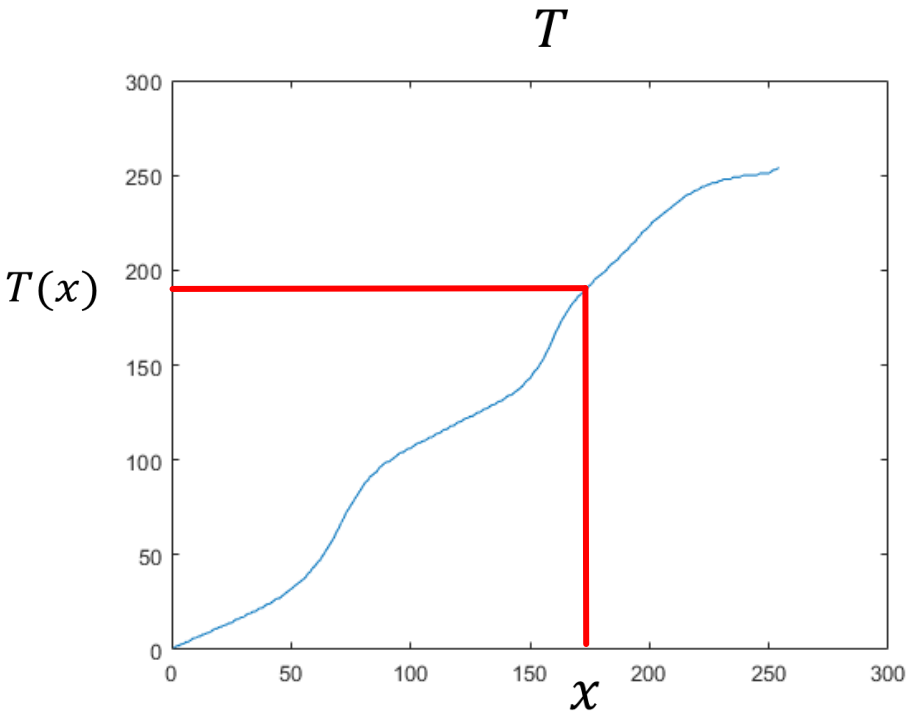
# Monge formulation

$$W(f,g) = \min_T \int_X c\left(x, T(x)\right) f(x)\, \mathrm{dx}$$

s.t. $\quad f(x) = g(T(x))\, |\det J_T(x)|$

Monge used $c(x,y) = |x-y|$




Same total mass

"find a good warping between $f$ and $g$ with the change of variable formula"

# Kantorovich formulation

Cost

$\mathrm{m}_{i \to j}$ particles will move from i to j

$$\min_{m} \sum_{i} \sum_{j} c_{i,j}\, m_{i \to j}$$

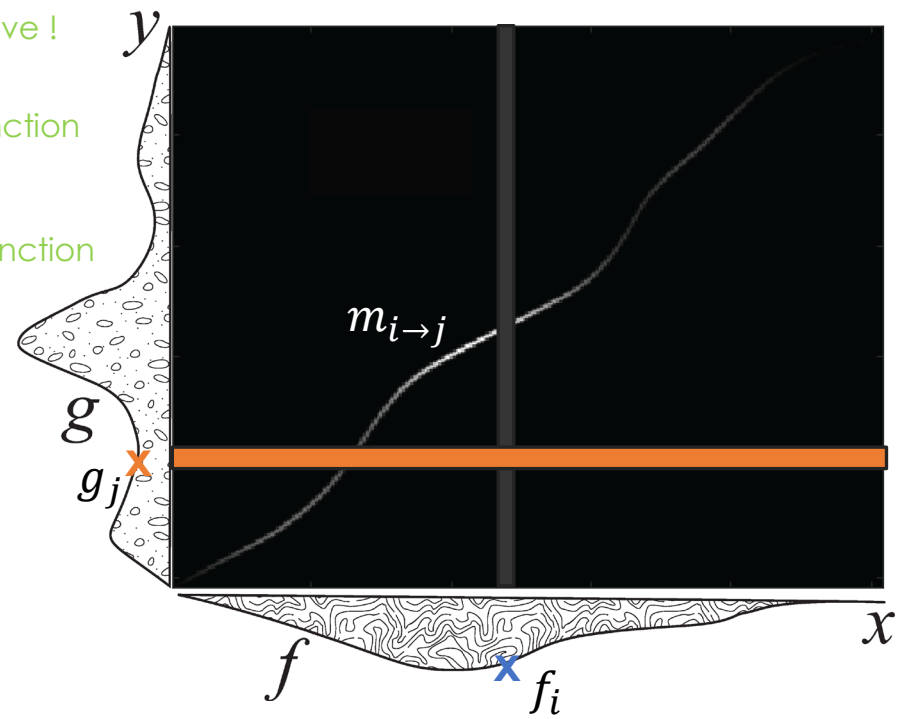
such that:

$$m_{i \to j} \geq 0$$ Nb of particles is positive !

$$\sum_{i} m_{i \to j} = g_j$$ Reconstruct target function

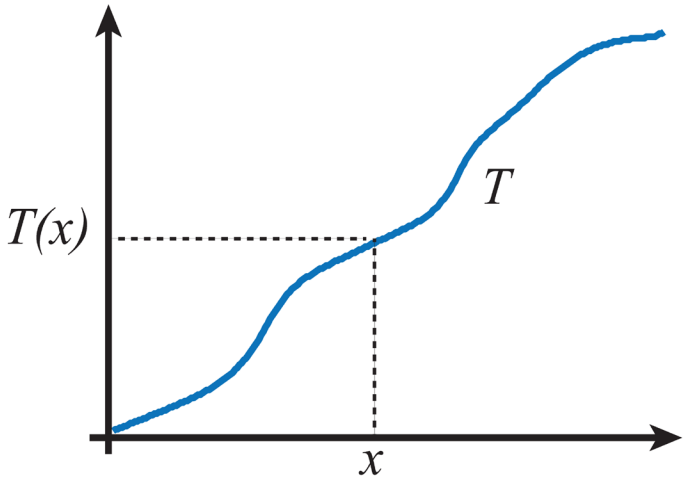
$$\sum_{j} m_{i \to j} = f_i$$ Reconstruct source function

Work for transforming f into g

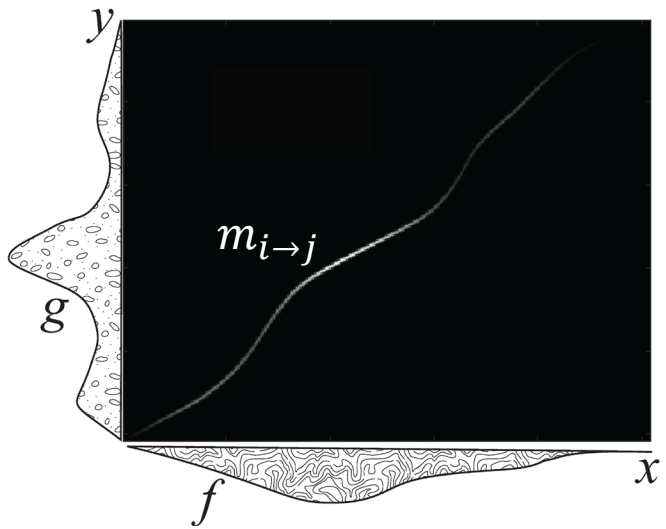



Discretization of the Kantorovich problem
Earth Mover's Distance

# Intuition: comparison




- Finds a "transport map"
- Difficult non-linear problem
- May have no solution
  (e.g., a Dirac splitting in two)
- Leads to PDEs




- Finds a "transport plan"
- Linear program
- "Always" has solution
  (i.e., under reasonable assumptions)
- Also has dual formulation

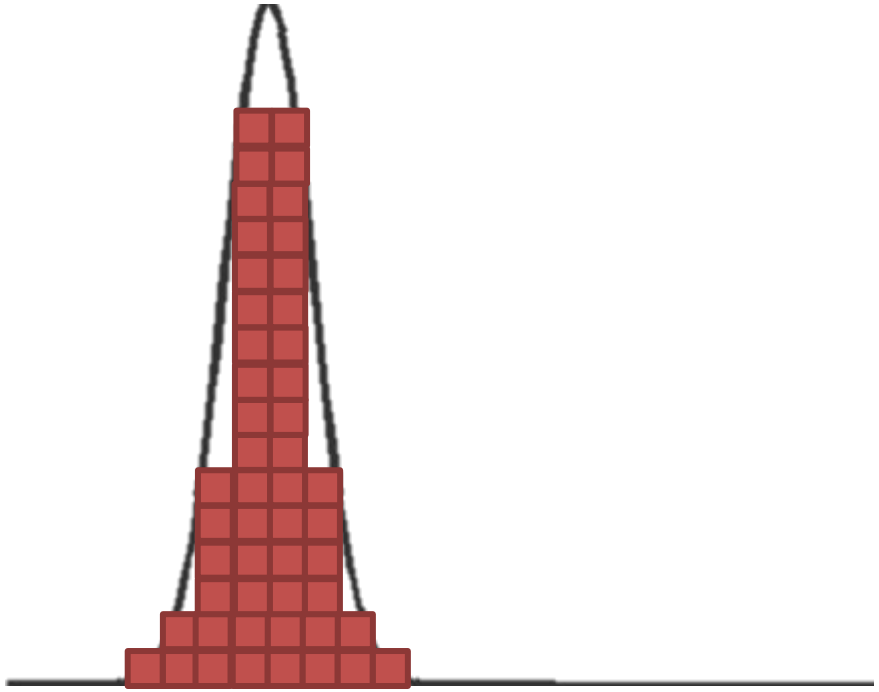
When it exists, the solution is the same.

Often, $c(x,y) = \|x-y\|_p^p \Rightarrow W_p^p$

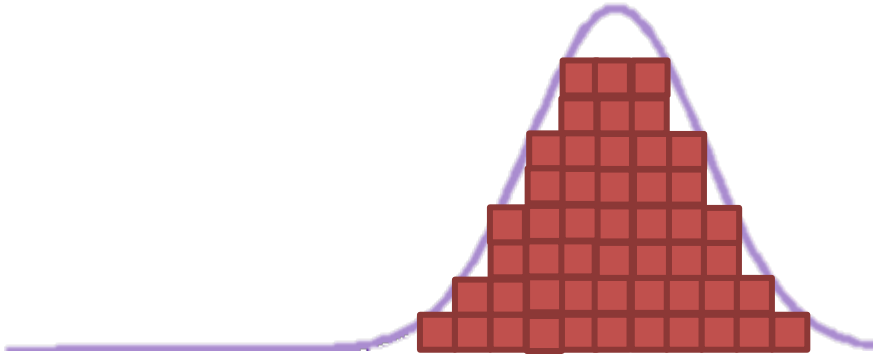
$W_p$ is a distance

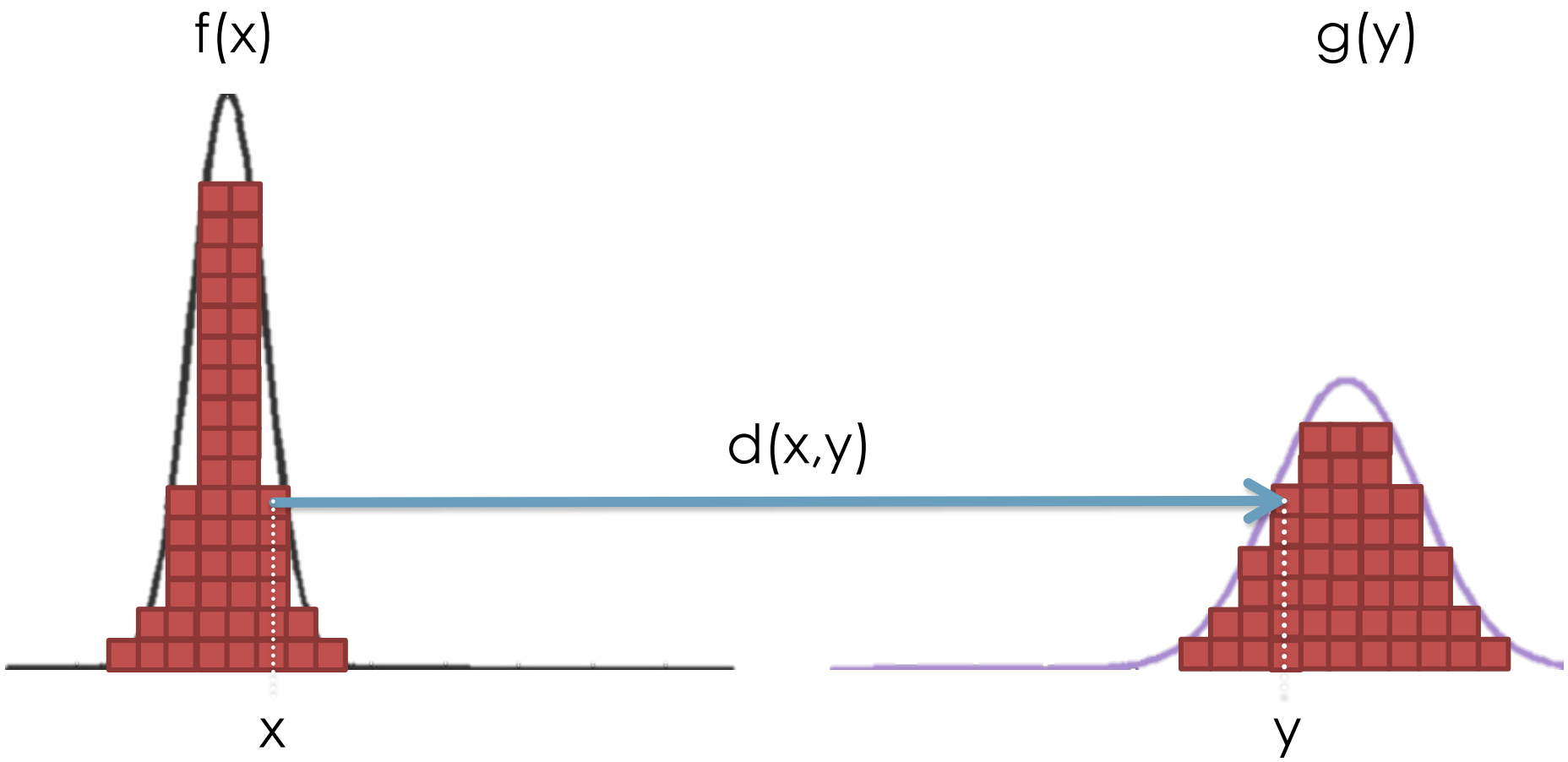

f(x)
g(y)

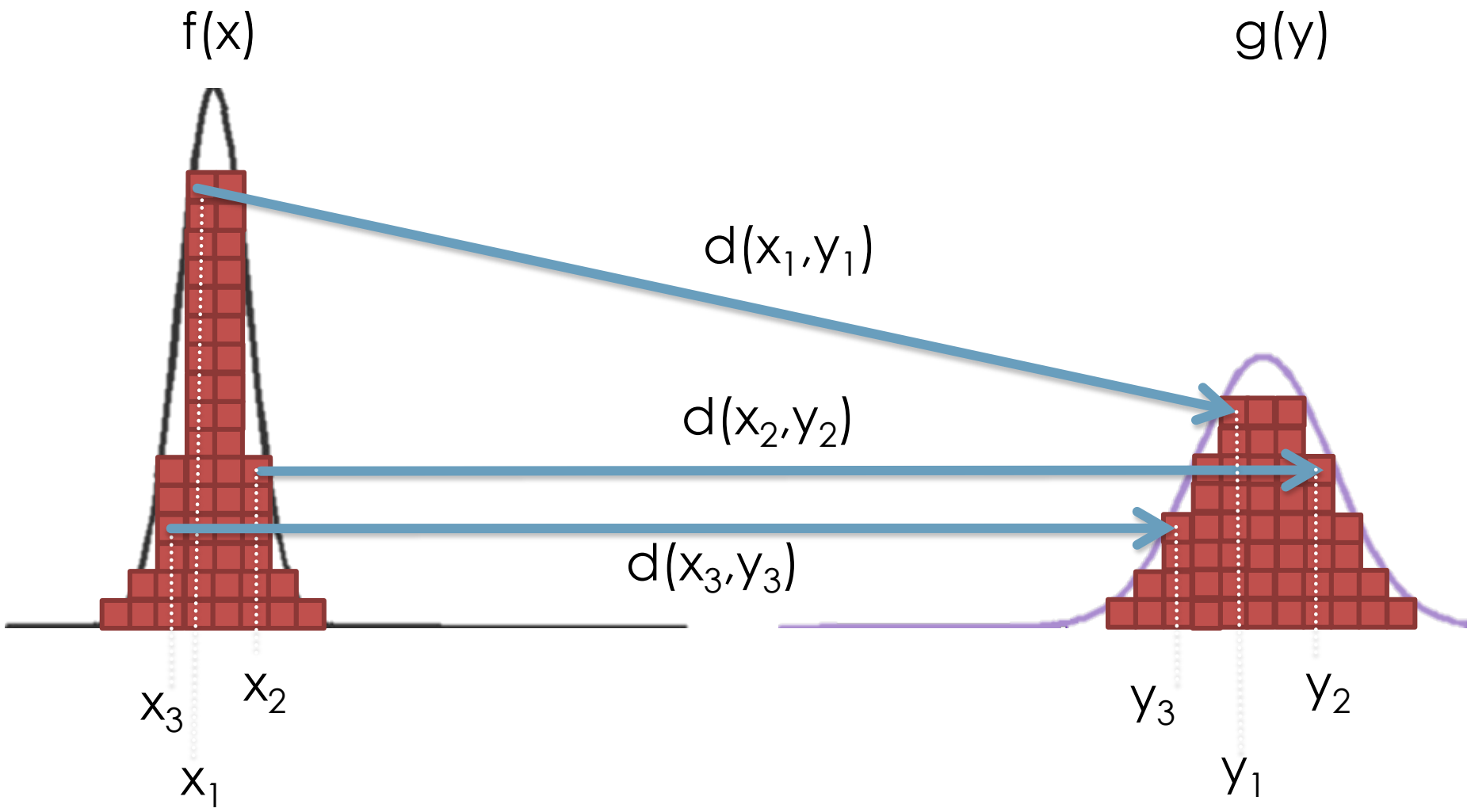
f(x)



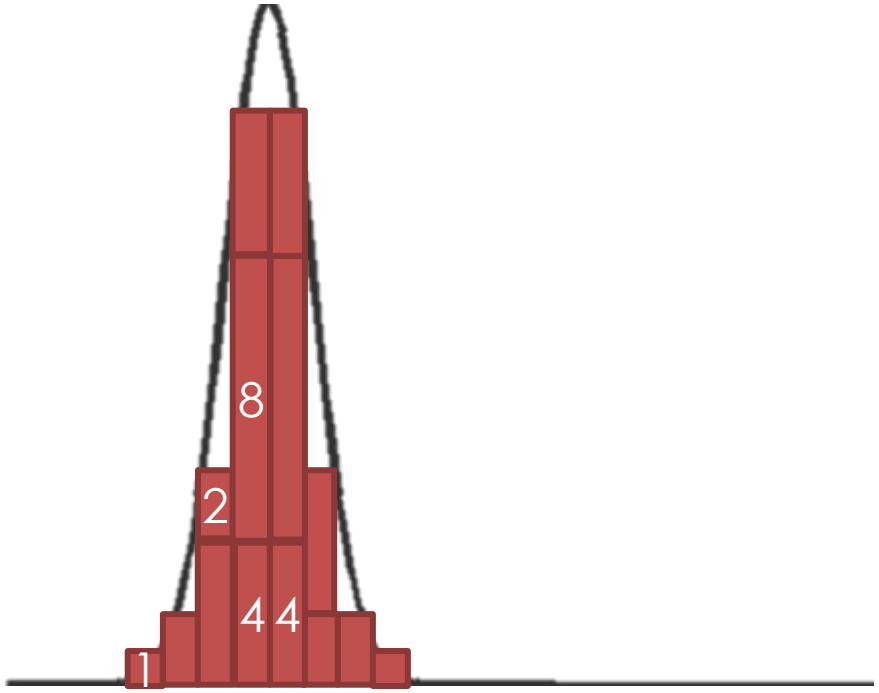
g(y)



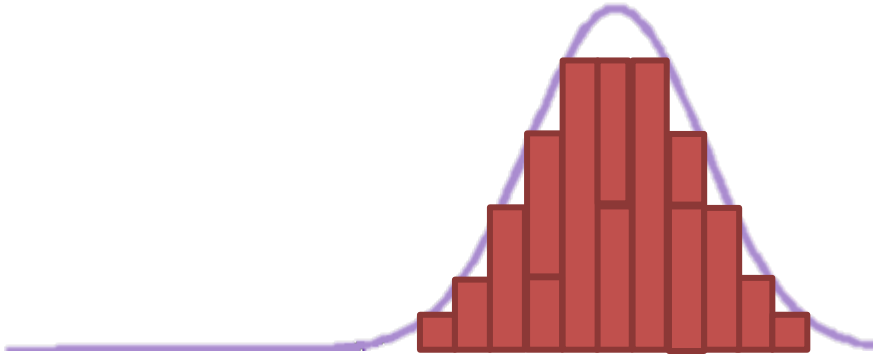

f(x)
g(y)
d(x,y)
x
y


f(x)
g(y)
d(x1,y1)
d(x2,y2)
d(x3,y3)
x3
x2
x1
y3
y2
y1

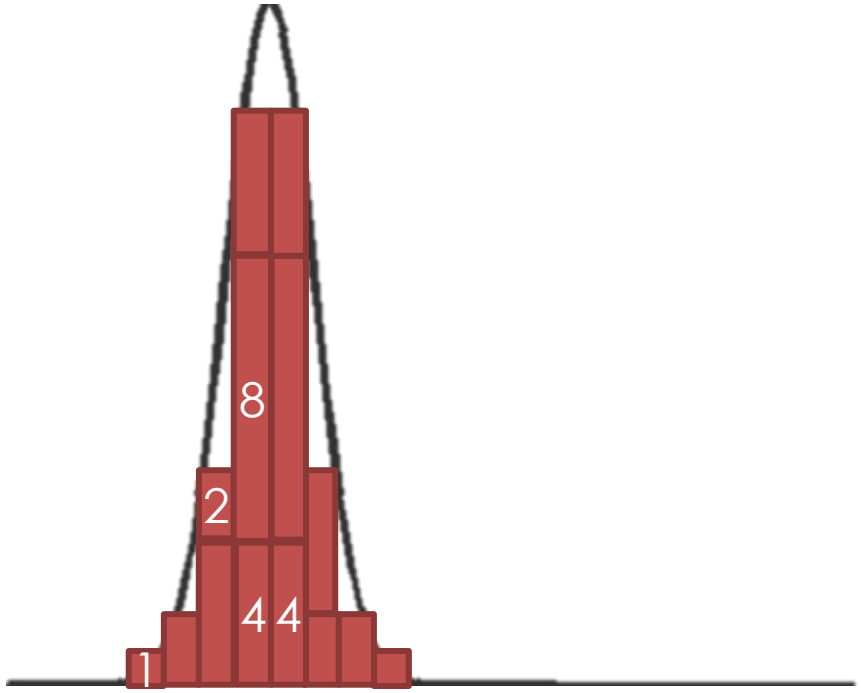
f(x)



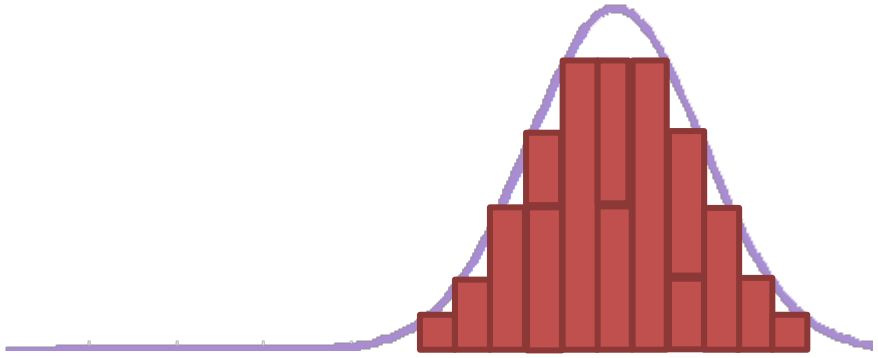

g(y)

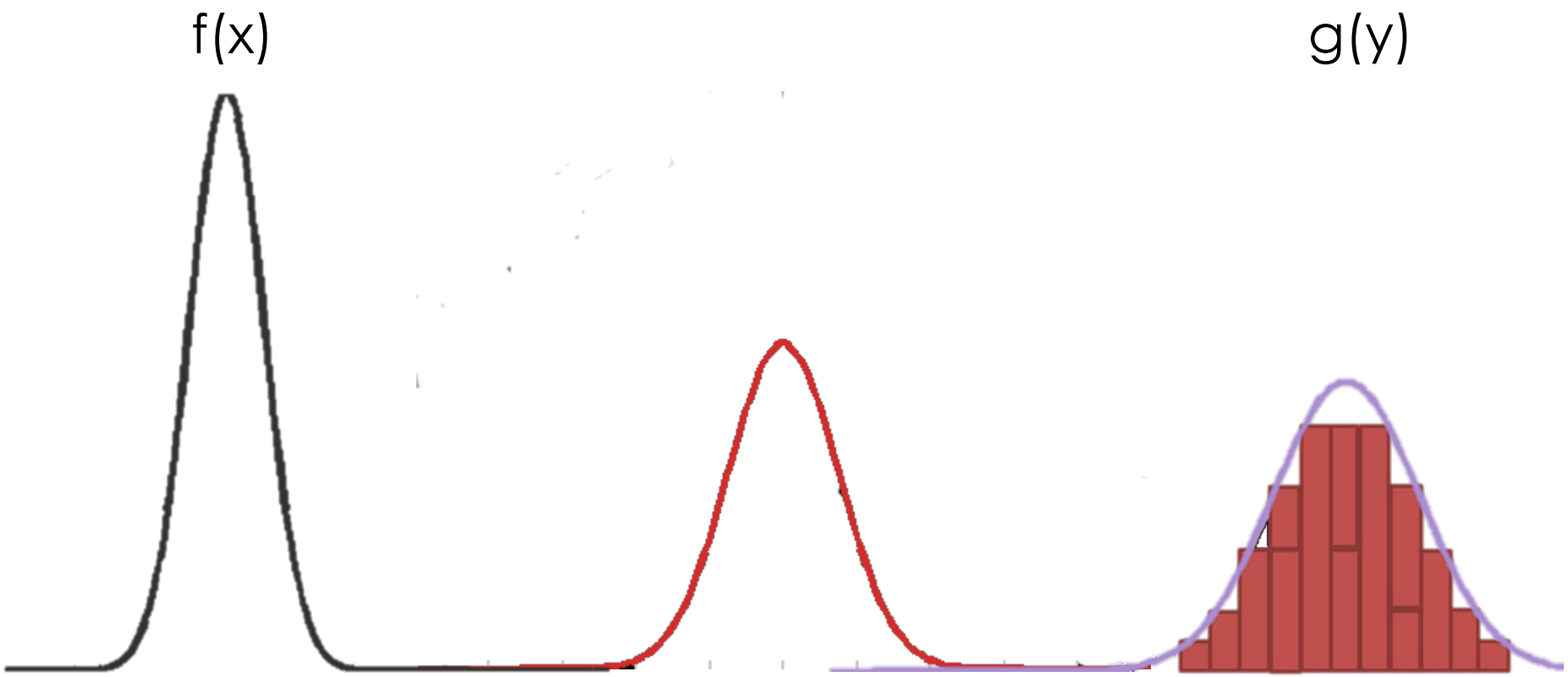


f(x)

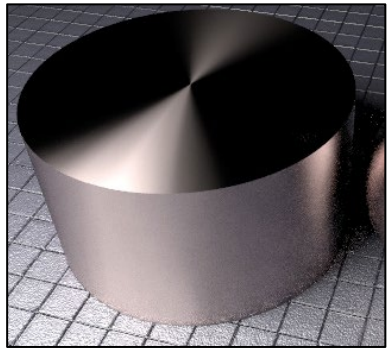



g(y)

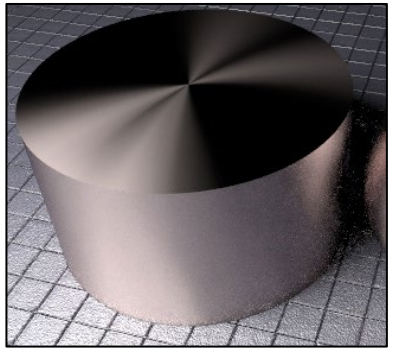



f(x)
g(y)

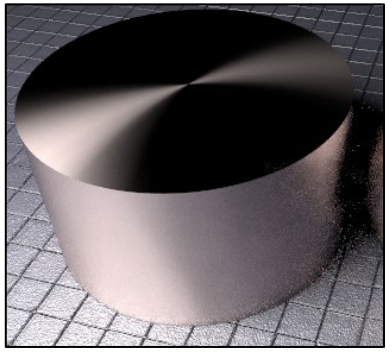
# Application: BRDF



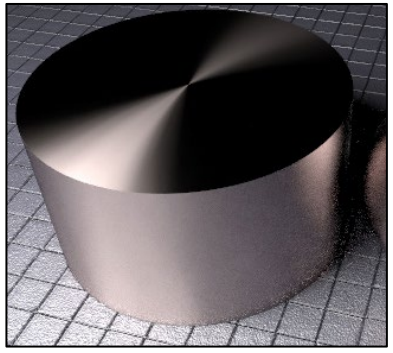
Function A



Linear interpolation



Function B



Displacement interpolation

[Bonneel et al. 2011] *Displacement Interpolation using Lagrangian Mass Transport*, Siggraph Asia

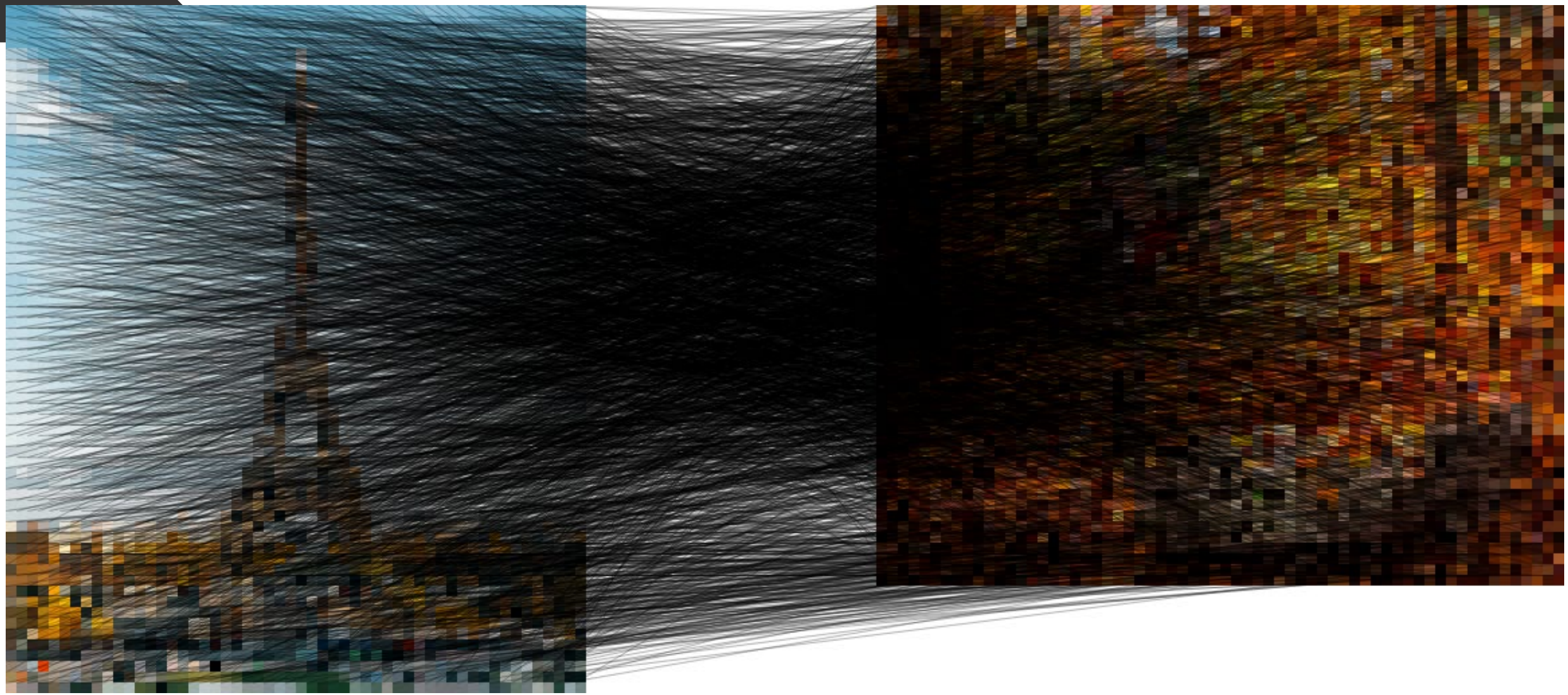
# Applications to Color Grading

Input photo

Target style

Input photo

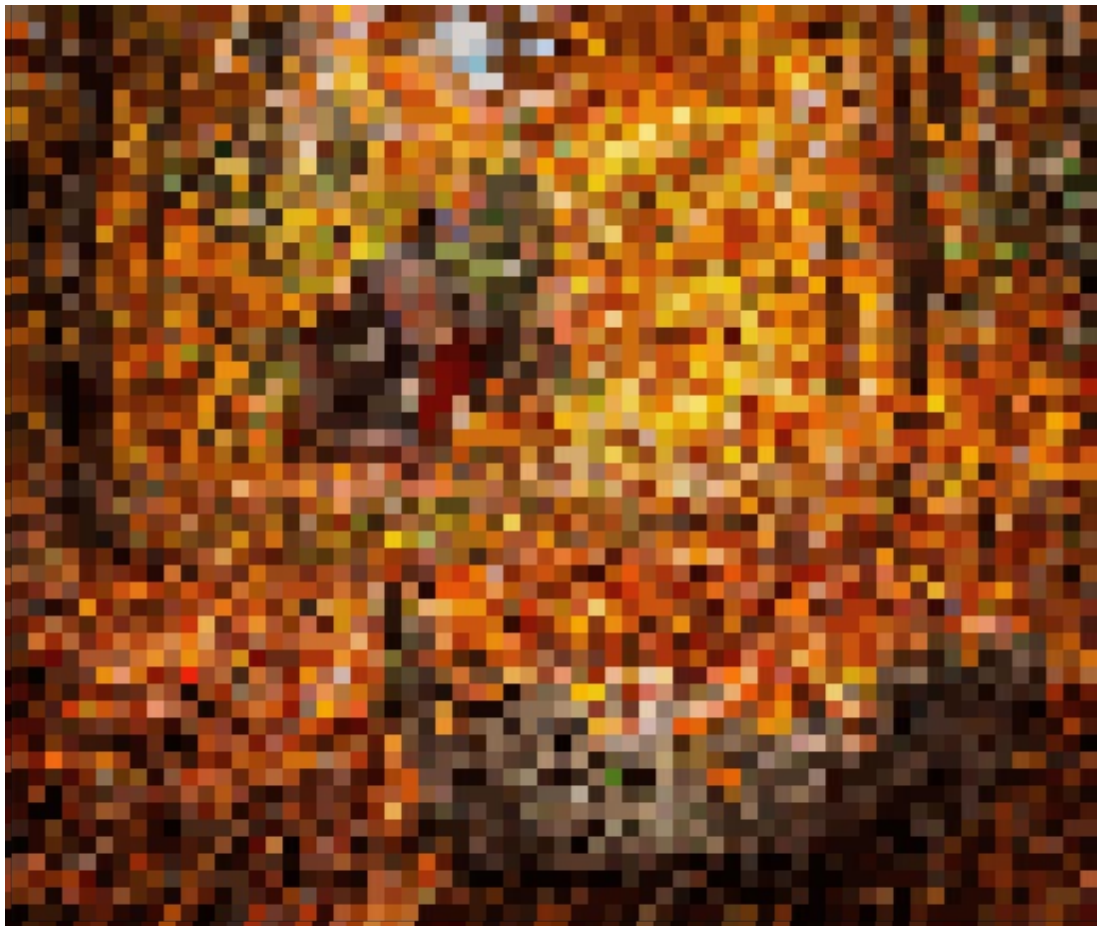
Target style



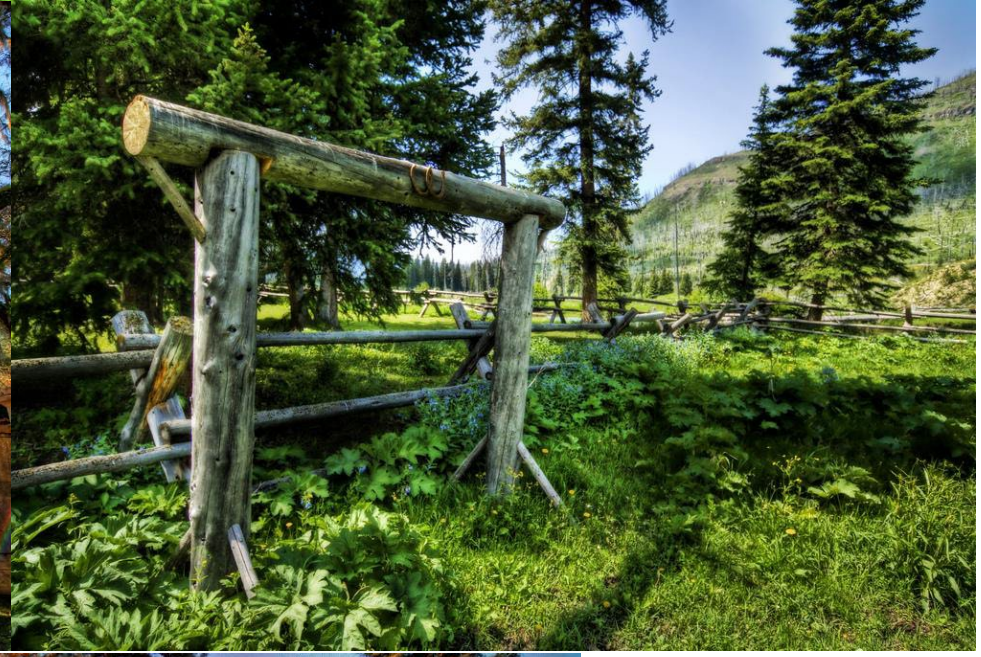
Input photo

Target style

Input photo



Target style



# Sliced and Radon Wasserstein Barycenters of Measures

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, Hanspeter Pfister

Journal of Mathematical Imaging and Vision (2014)

# Simple cases

- Transport 1 Gaussian ↔ 1 Gaussian
- Transport 1 Gaussian ↔ 1 Gaussian ↔ 1 Gaussian […]
- Transport = translation + scaling
- Transport 1D function ↔ 1D function (↔ 1D function […])

# Optimal transport is simple for Gaussians

- Optimal transport and barycenters trivially solved for
  - Gaussian distributions with $c(x, y) = \|x - y\|^2$
    - $W_2^2(\mathcal{N}_0, \mathcal{N}_1) = \mathrm{tr}\left(\Sigma_0 + \Sigma_1 - 2\Sigma_{0,1}\right) + \|\mu_0 - \mu_1\|$ with $\Sigma_{0,1} = \left(\Sigma_0^{\frac{1}{2}}\Sigma_1\Sigma_0^{\frac{1}{2}}\right)^{1/2}$
    - $T(x) = \Sigma_{0,1}x$
    - Barycenter: $\mathcal{N}(\mu, \Sigma)$ with $\mu = \sum_k \lambda_k \mu_k$ and iterations

$$\Sigma^{(n+1)} = \sum_k \lambda_k \left(\sqrt{\Sigma^{(n+1)}}\Sigma_k\sqrt{\Sigma^{(n+1)}}\right)^{1/2}$$

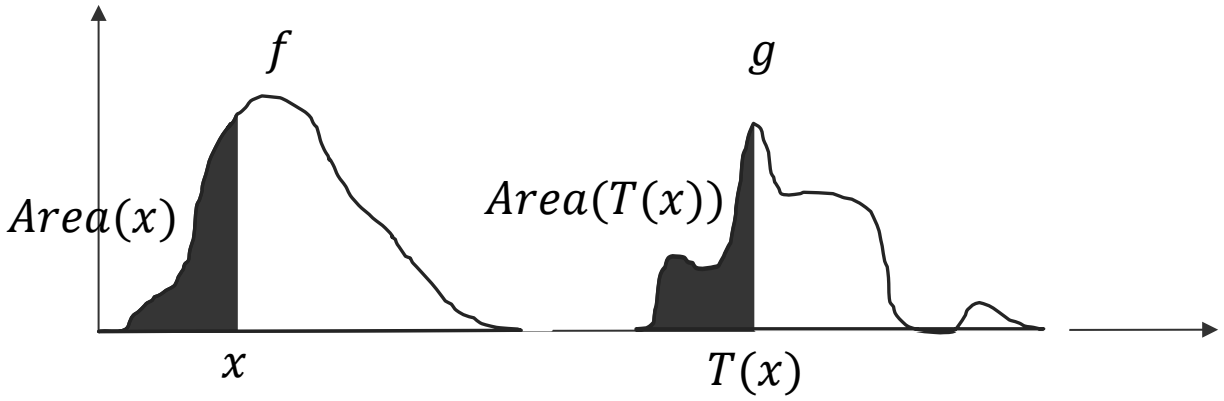
# Optimal transport is simple in 1D

- Continuous case with density, convex cost, $\mu = f\ dx$, $\nu = g\ dy$
  - Need: $\int_{-\infty}^{x} f(x)dx = \int_{-\infty}^{T(x)} g(x)dx$

$$T = G^{-1} \circ F$$

with $F(x) = \int_{-\infty}^{x} f(x)dx$ and $G(x) = \int_{-\infty}^{x} g(x)dx$

Generalize $G^{-1}$ : $G^{-1}(y) = \min_x \{y = G(x)\}$ → Quantile function: e.g.: "what salary corresponds to the first percentile"



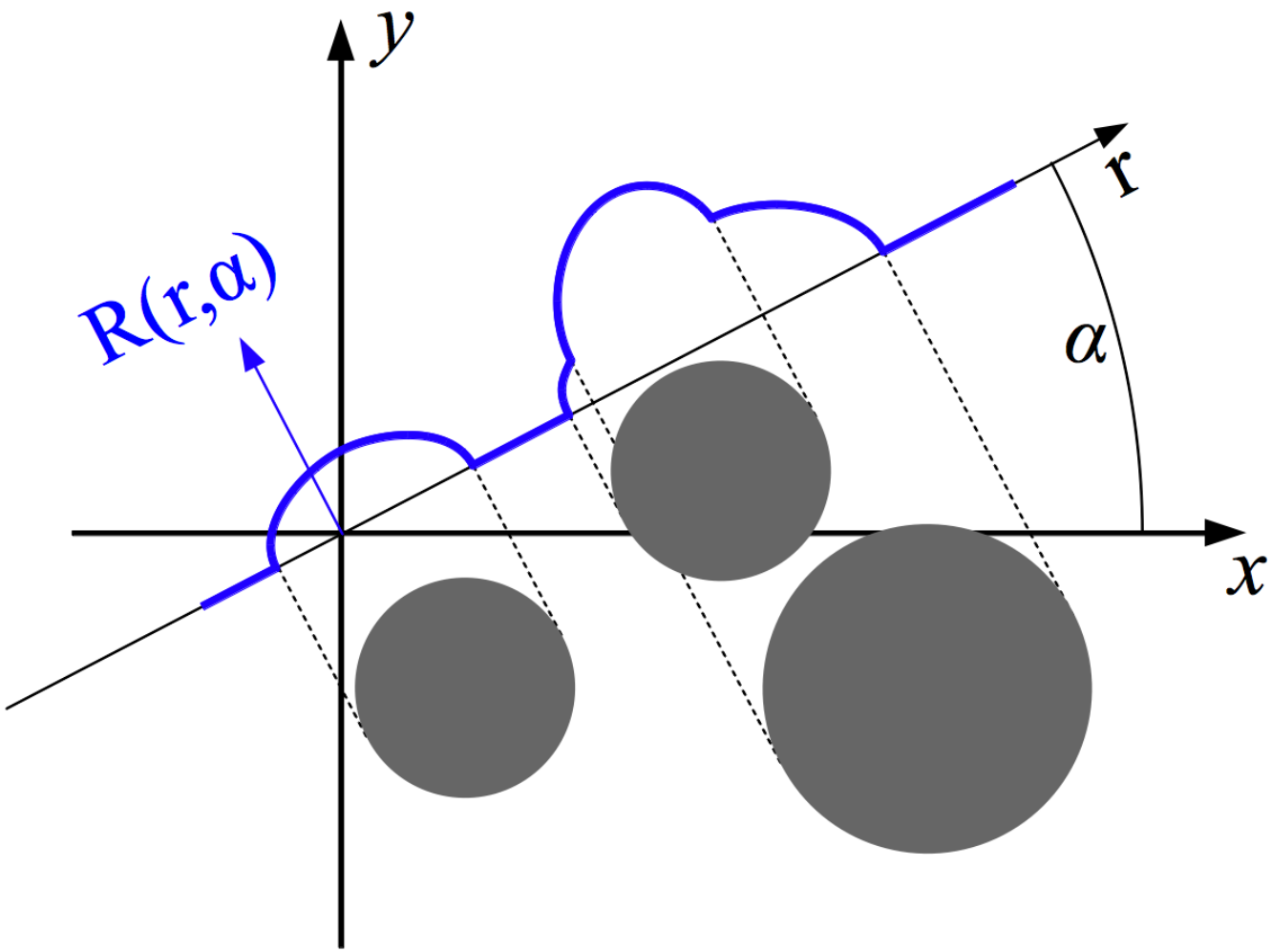
# 1D Case

OT Map: $T = G^{-1} \circ F$

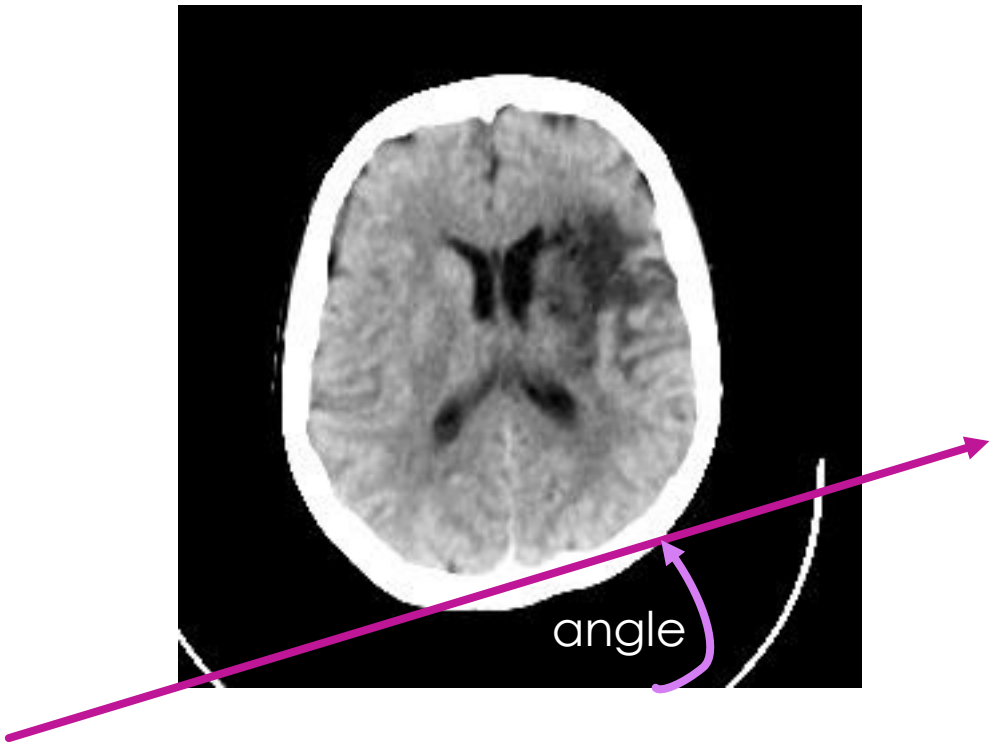
OT cost: $\int_0^1 c\left(F^{-1}(t) - G^{-1}(t)\right)dt$

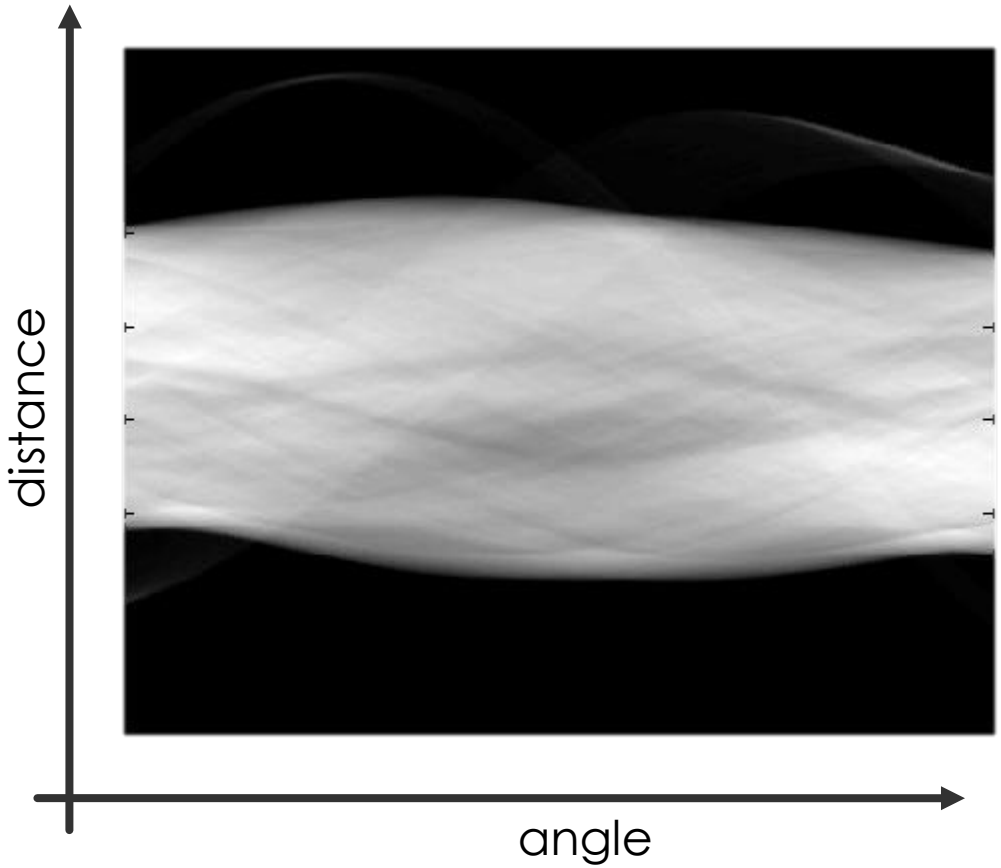
Interpolation: $F_{interp}^{-1}(x) = \sum_i \alpha_i \, F_i^{-1}(x)$

# Radon transform

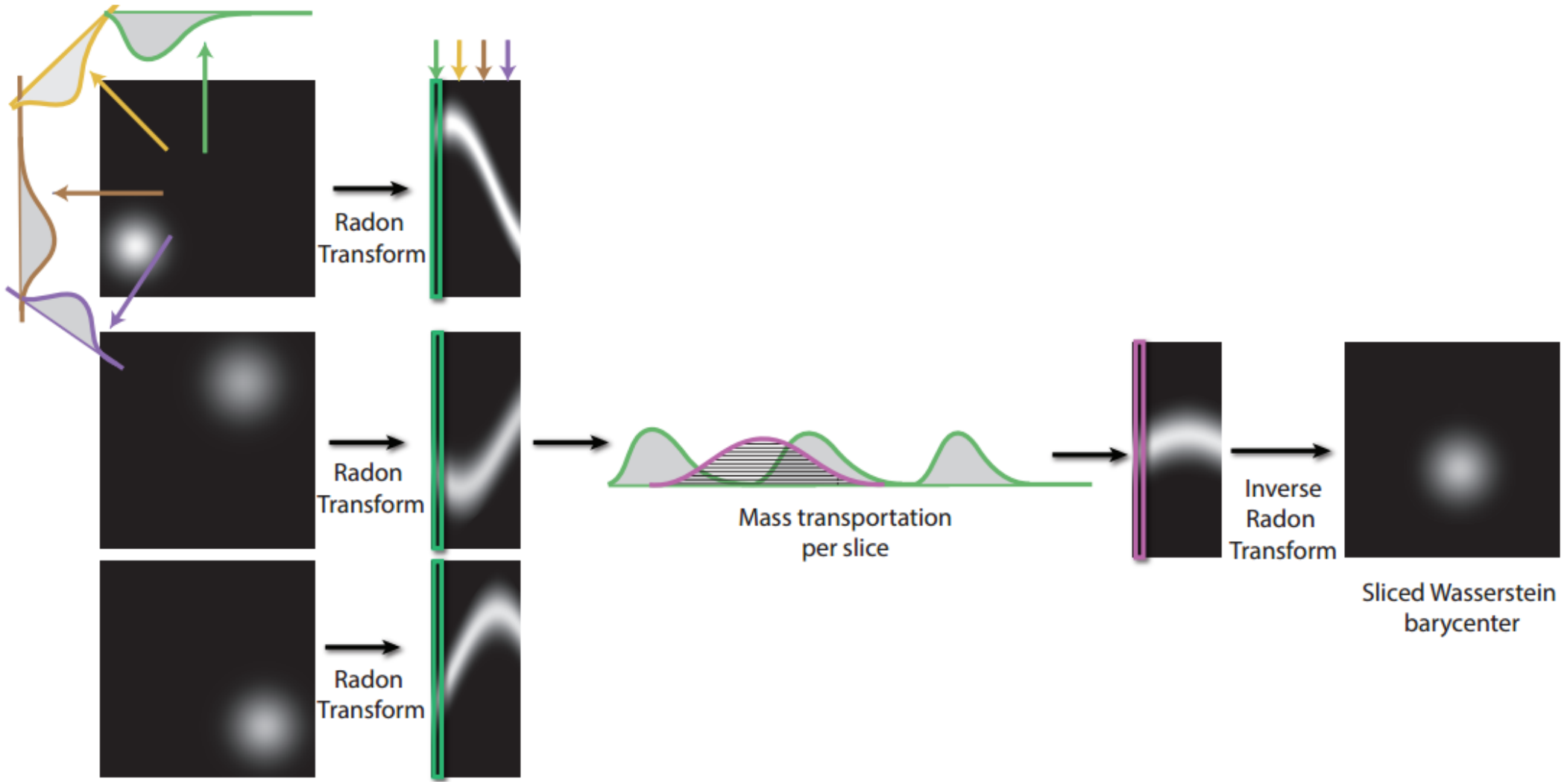


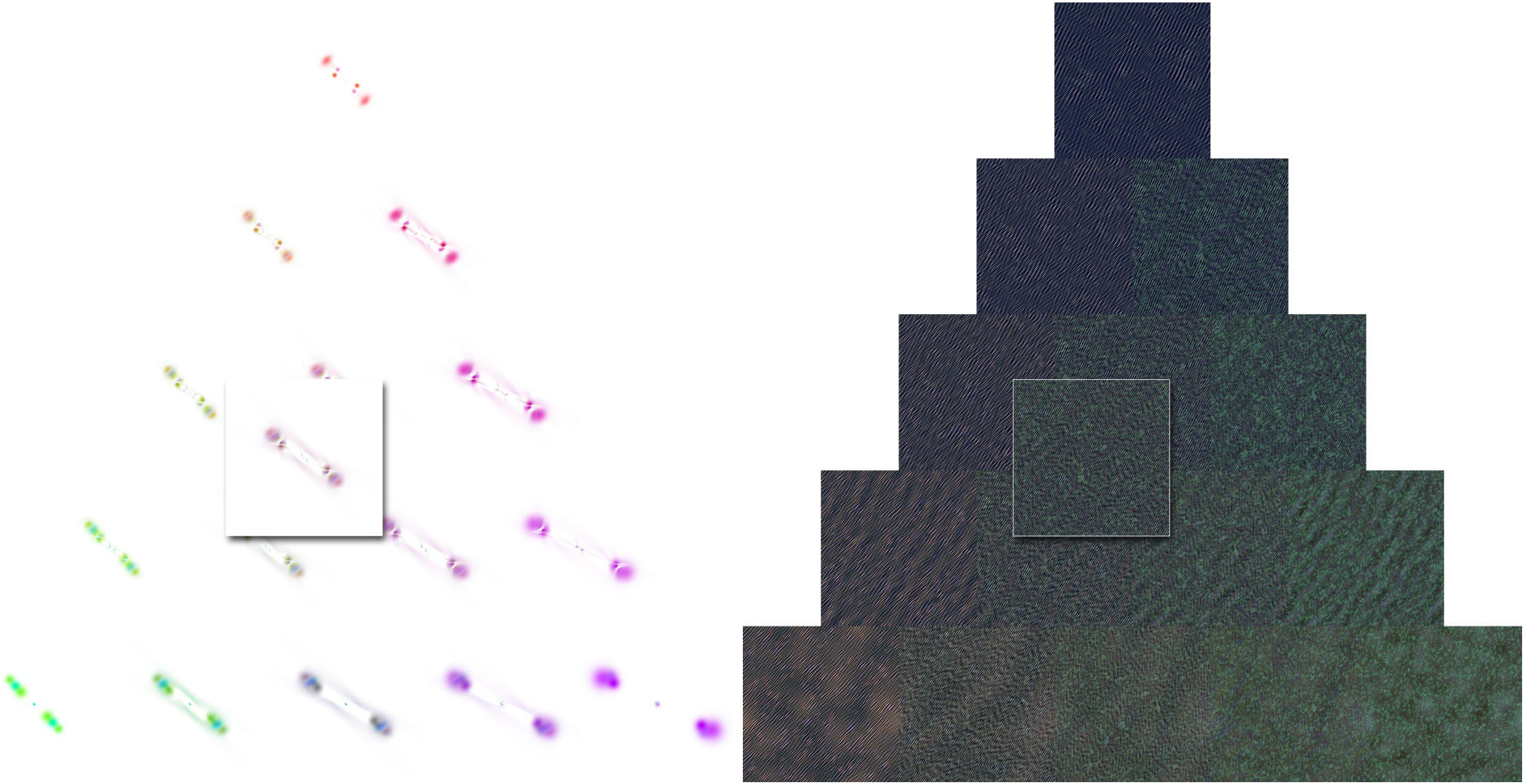
# Radon transform



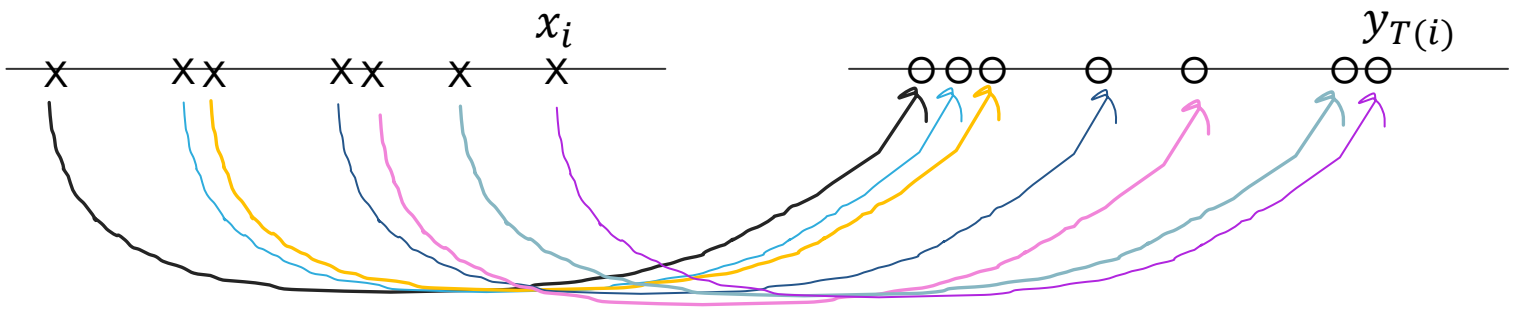
# Method


Radon
Transform
Radon
Transform
Radon
Transform
Mass transportation
per slice
Inverse
Radon
Transform
Sliced Wasserstein
barycenter



# 1D Case, discrete

- Discrete case, $\mu = \sum_{i=1}^{n} \delta_{x_i}$, $\nu = \sum_{i=1}^{n} \delta_{y_i}$ (same for interpolating between more than 2 measures)
- Optimal transport for convex cost = pairing sorted samples



# Sliced Wasserstein Distance

- For discrete high-dimensional distributions $\mu = \sum_{i=1}^{n} \delta_{x_i}$ and $\nu = \sum_{i=1}^{n} \delta_{y_i}$

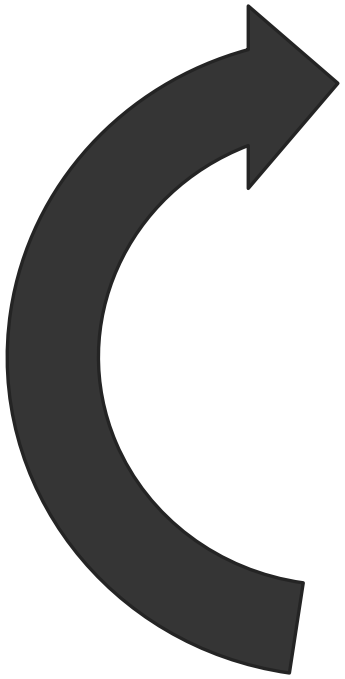
Consider energy

$$SW(\mu, \nu) = \int_S W_2^2\big(\mathrm{proj}(\mu, \omega), \mathrm{proj}(\nu, \omega)\big)\, \mathrm{d}\omega$$

Where $\mathrm{proj}(\mu, \omega)$ is the 1-d distribution : $\mathrm{proj}(\mu, \omega) = \sum_i \delta_{\langle x_i, \omega \rangle}$ (same for $\nu$)

And $W_2^2$ computes the 1-d squared Wasserstein distance

# Sliced Wasserstein Distance



- Take a uniform random direction $\omega$
  - $\omega \leftarrow \left(\mathcal{N}(0,1), \mathcal{N}(0,1), \mathcal{N}(0,1)\right)$ and normalize
- Project samples of $\mu$ and $\nu$ on $\omega$ : $\mu$' = Proj($\mu$) and $\nu$' = Proj($\nu$)
- Sort $\mu$' and $\nu$', i.e, find permutations $\sigma_\mu$ and $\sigma_\nu$

- To compute the Sliced Wasserstein Distance:

$$d^2 \leftarrow d^2 + \sum_i \left| \left\langle x_{\sigma_\mu(i)}, \omega \right\rangle - \left\langle y_{\sigma_\nu(i)}, \omega \right\rangle \right|^2$$

- or, to advect $\mu$ towards $\nu$ ("gradient flow")
  - Update $\mu$ by $x_{\sigma_\mu(i)} \leftarrow x_{\sigma_\mu(i)} + \left(\langle x_i, \omega \rangle - \langle y_i, \omega \rangle\right) \omega$

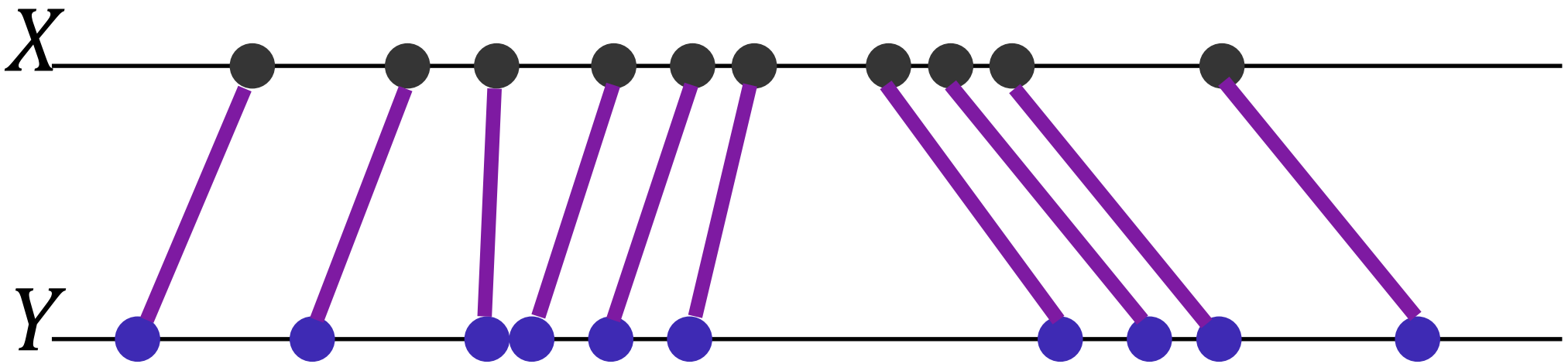
**Wasserstein Barycenter and its Application to Texture Mixing. [Rabin et al. 2011]**

# Sliced Wasserstein Distance


f

g

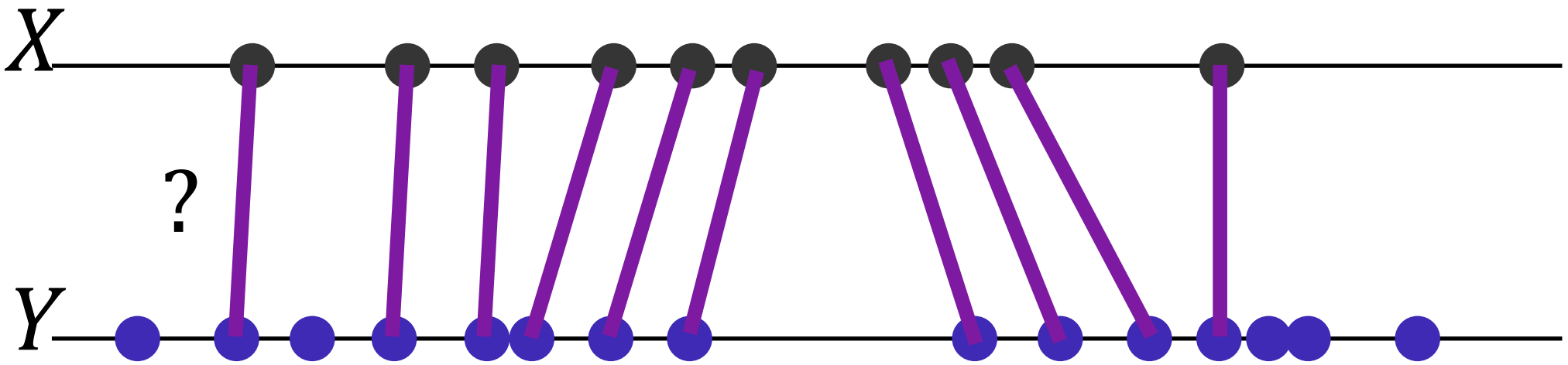
# 1-d Linear Assignment Problem is trivial*




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

# Partial optimal assignment ?

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

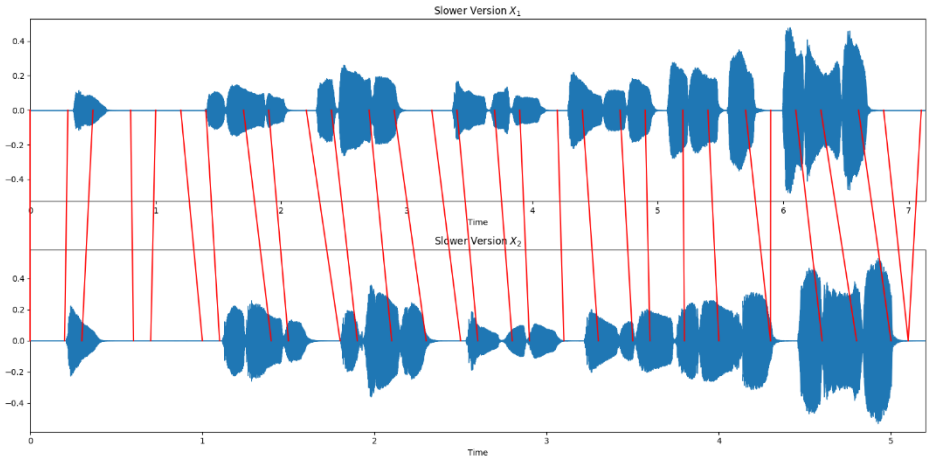


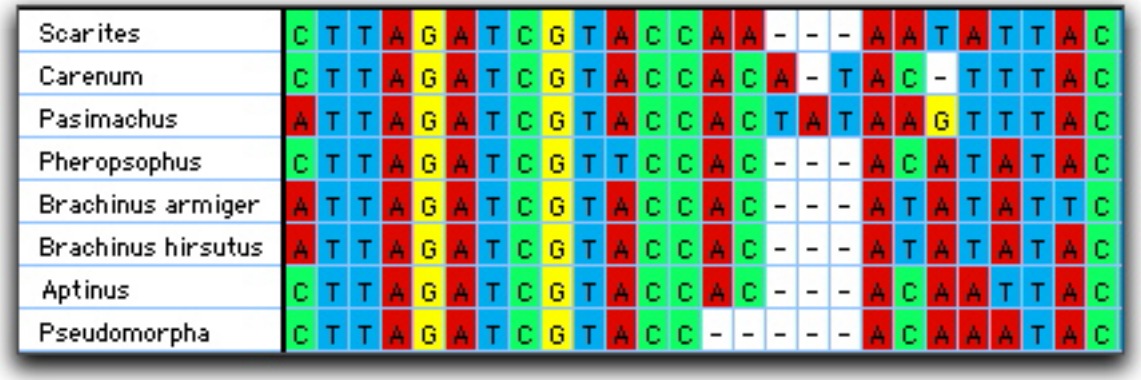

$$W(f,g) = \min \sum_{i,j} c_{i,j}\pi_{i,j} \quad \text{s.t.} \quad \sum_j \pi_{i,j} = 1 \quad \sum_i \pi_{i,j} \leq 1 \quad \pi_{i,j} \geq 0$$

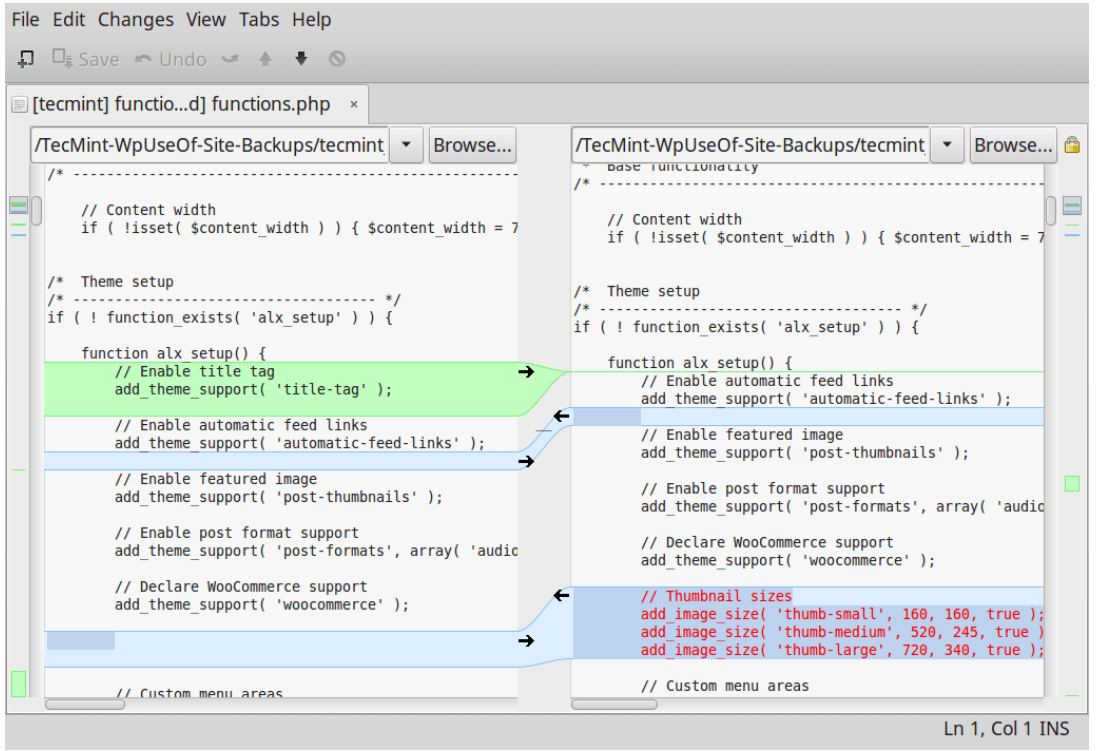
$$\min_{T \text{ injective}} \sum_i c(x_i, y_{T(i)})$$

# Similar problems

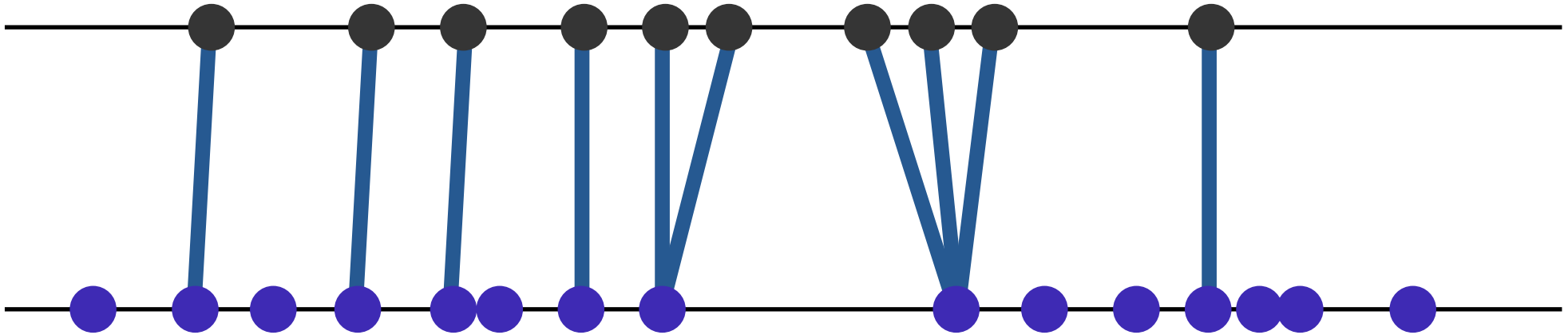
- DNA sequence alignment
- Text alignment
- Music synchronization
- …



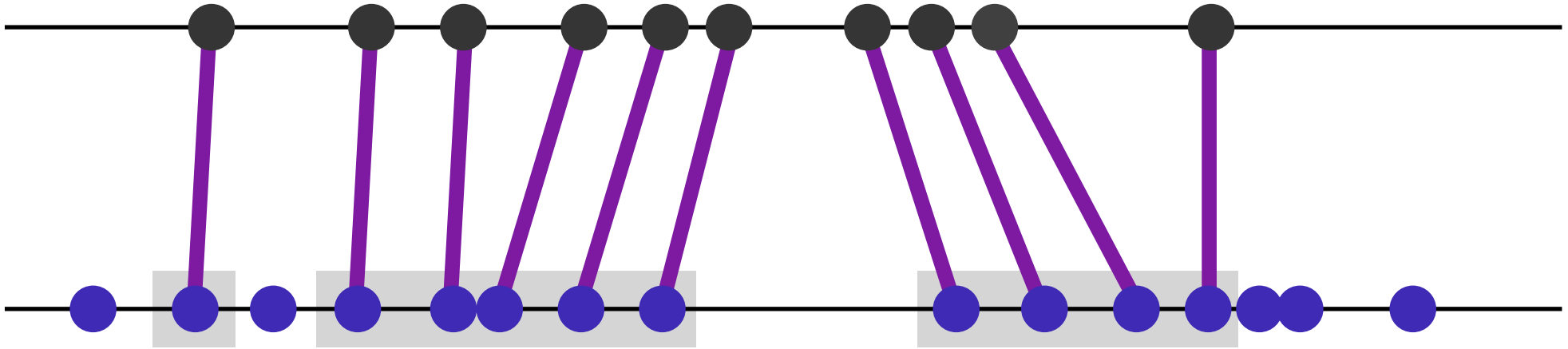



# Quadratic time complexity algorithm (linear space)



Euclidean Nearest Neighbor assignment

# Quadratic time complexity algorithm (linear space)



Euclidean Nearest Neighbor assignment

Optimal Transport assignment

Intervals of bijective assignments

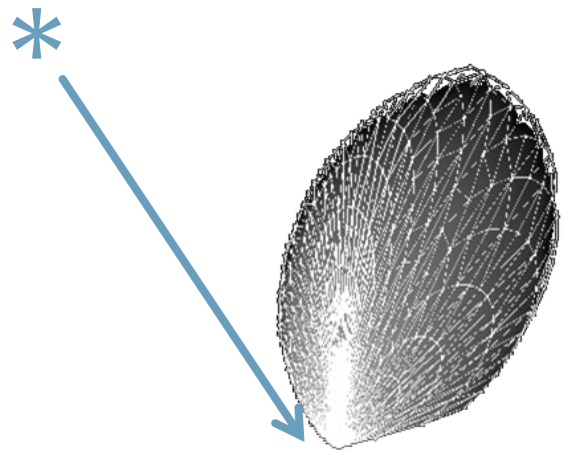
# Displacement Interpolation using Lagrangian Mass Transport

Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, Wolfgang Heidrich

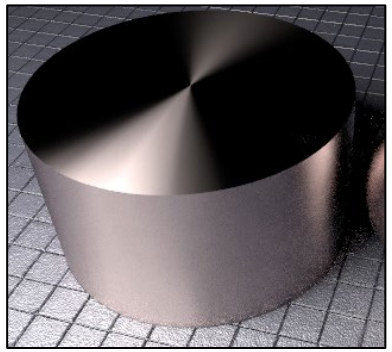
SIGGRAPH Asia 2011

# Example: BRDF

- "Bidirectional Reflectance Distribution Function"



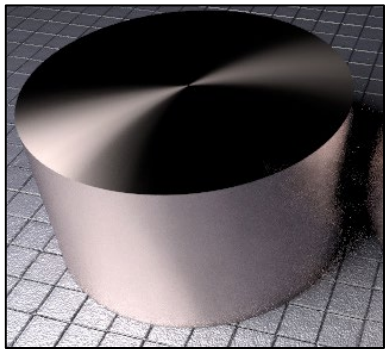
# Example: BRDF



Function A

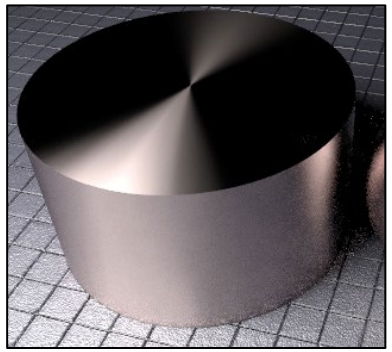
?

Interpolation

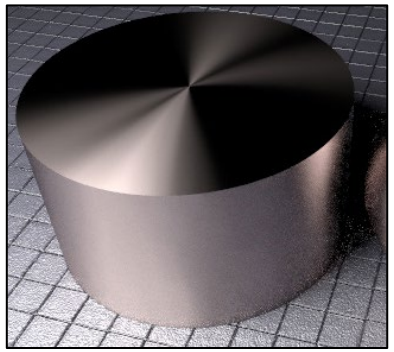


Function B

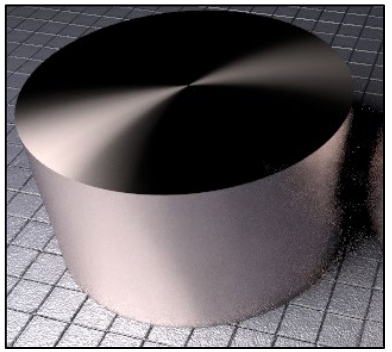
# Example: BRDF



Function A

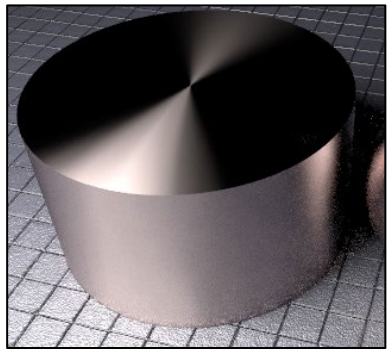


Linear interpolation

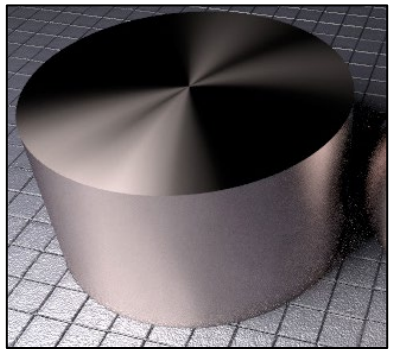


Function B

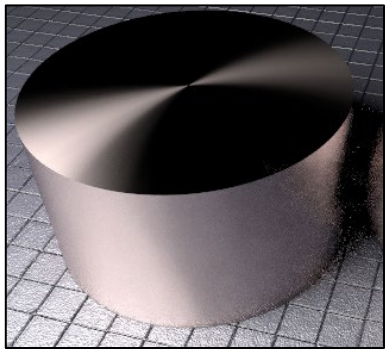
# Example: BRDF



Function A

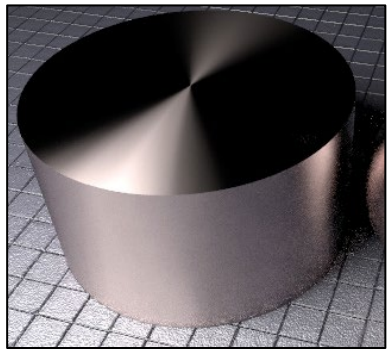


Linear interpolation

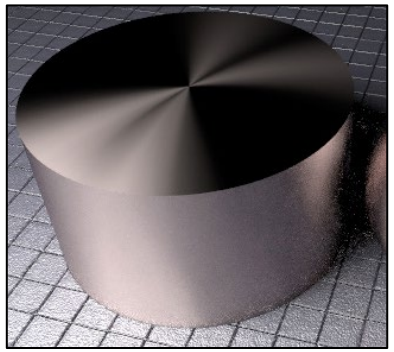


Function B

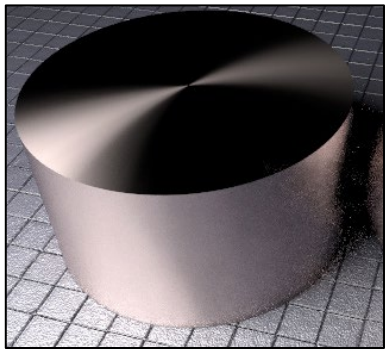
# Example: BRDF



Function A

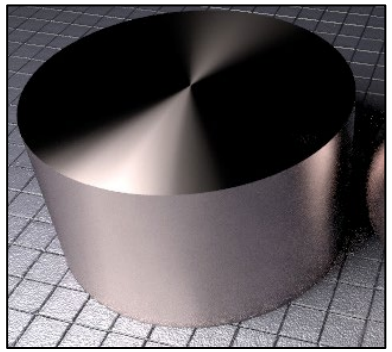


Linear interpolation

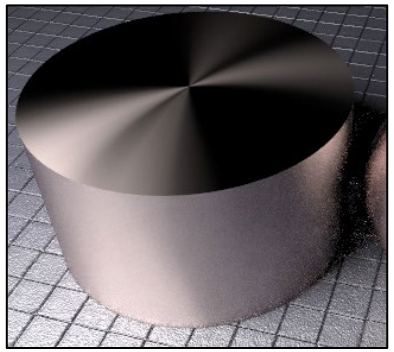


Function B

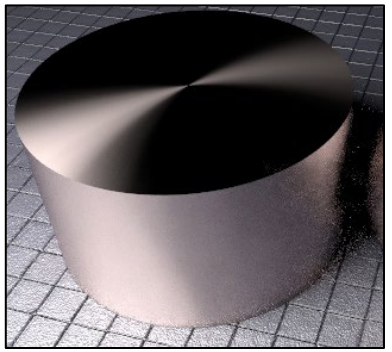
# Example: BRDF



Function A

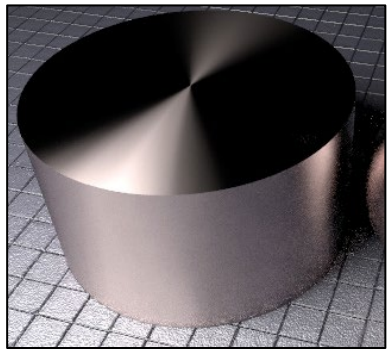


Linear interpolation

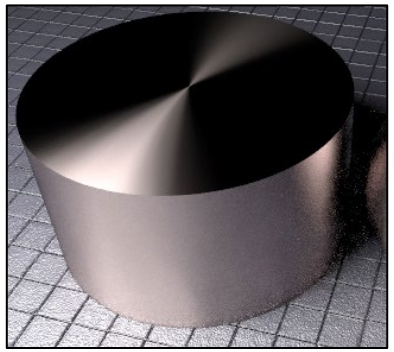


Function B

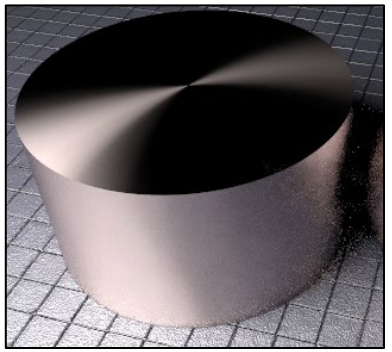
# Example: BRDF



Function A

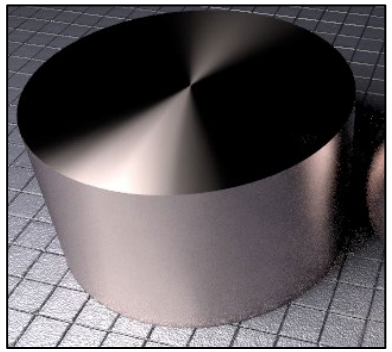


Displacement interpolation

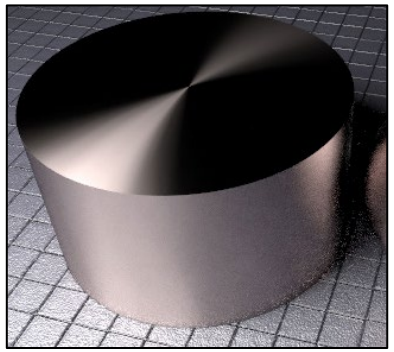


Function B

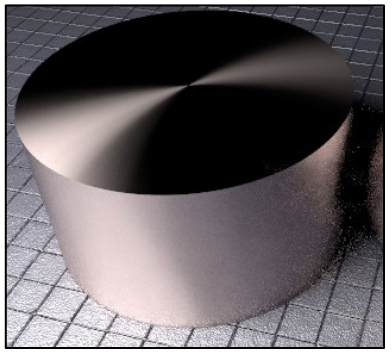
# Example: BRDF



Function A

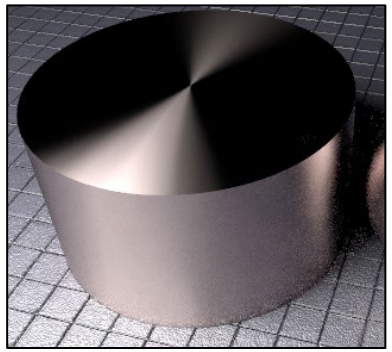


Displacement interpolation

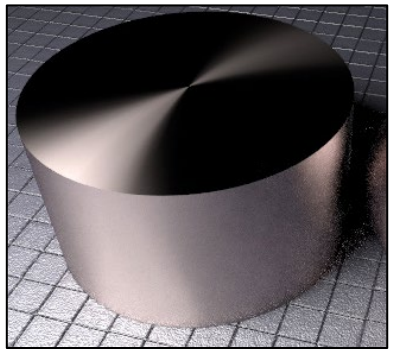


Function B

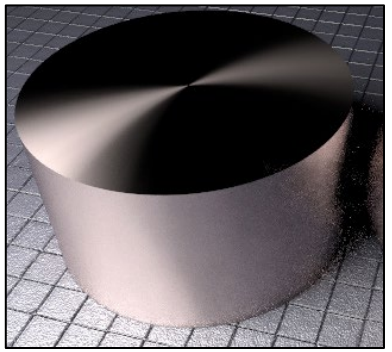
# Example: BRDF



Function A

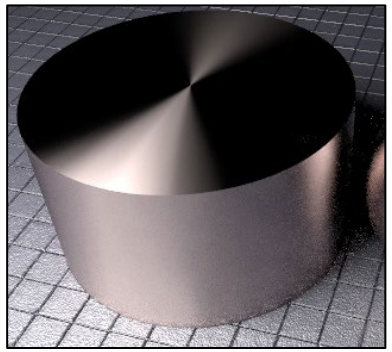


Displacement interpolation

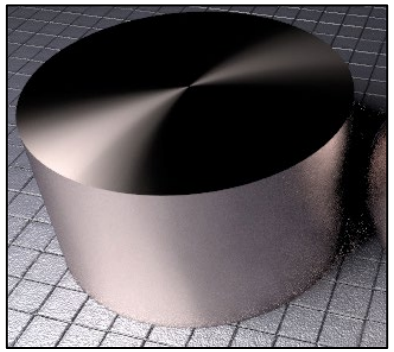


Function B

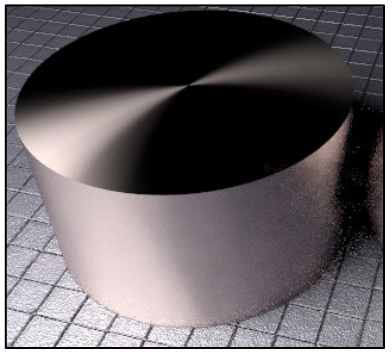
# Example: BRDF



Function A



Displacement interpolation



Function B

# Four steps

- Decompose PDFs into non-negative radial basis functions
- Optimal transport computation
- Partial advection
- Reconstruct interpolated PDF

- (+optional multiscale approach)

# Radial Basis Function decomposition

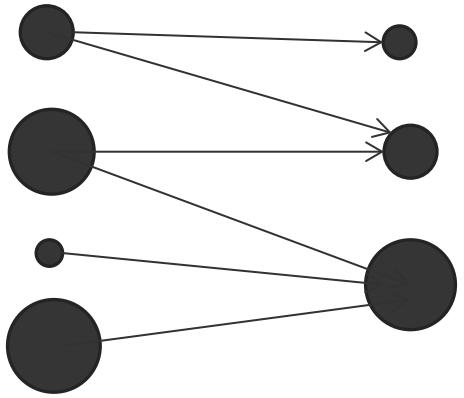
# Transport computation

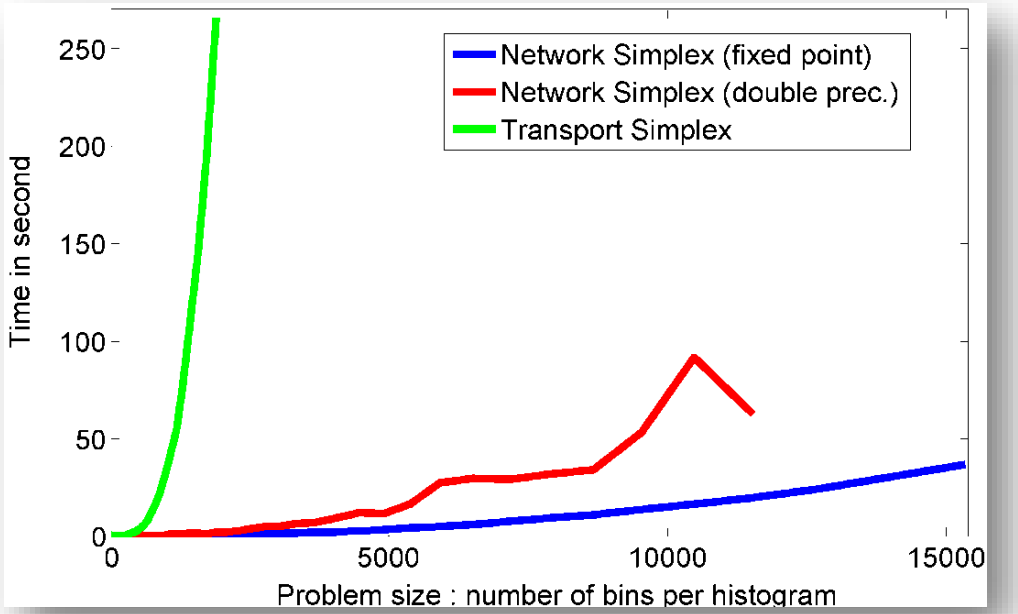
$$\min \sum_{i,j} c_{i,j} m_{i,j}$$

$$\text{s.t} \quad \sum_{j} m_{i,j} = f_i$$

$$\sum_{i} m_{i,j} = g_j$$

- Transport RBF weights
- Network simplex > Transportation simplex





# Auction algorithm for assignment

- Consider instead: $\max \sum a_{ij}$ over complete assignments $(i,j) \in S$ and $j \in A(i)$
  - $a_{ij}$ : how much person i is ready to pay for object j
    $p_j$ : Price person j will actually pay

Solve dual: $\min_{p,\pi} \sum \pi_i + \sum p_j$ s.t. $\pi_i + p_j \geq a_{ij} \ \forall i,j \in A(i)$

- At optimality $\pi_i = \max_{k \in A(i)} a_{ik} - p_k = a_{ij(i)} - p_{j(i)}$ (saturates constraint)
- Profit of person i: $\pi_i = \max_{j \in A(i)} v_{ij}$
  with benefit for object $j \in A(i)$: $v_{ij} = a_{ij} - p_j$
- Add some slack: $\pi_i - \epsilon = \max_{k \in A(i)} a_{ik} - p_k - \epsilon \leq a_{ij(i)} - p_{j(i)}$ optimal if $\epsilon < \frac{1}{N}$

"The auction algorithm", Bertsekas and Castanon

# Auction algorithm for assignment

- Start with some assignment S

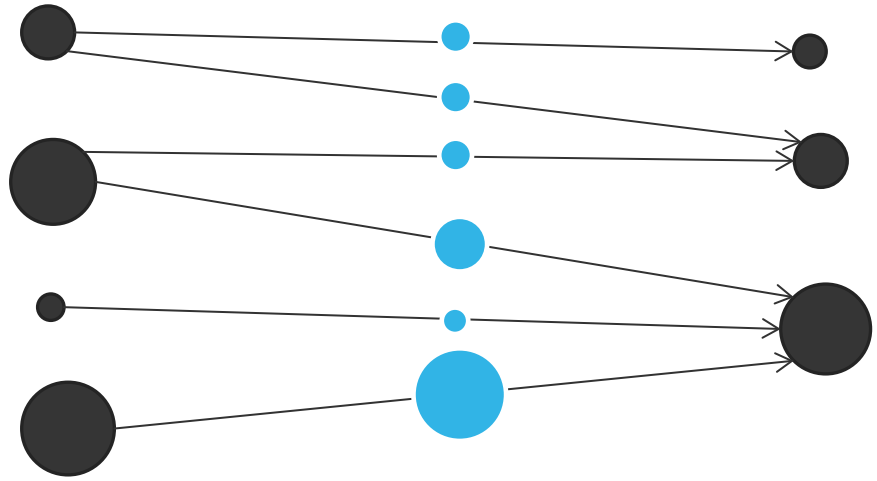
- For each unassigned person $i$, find object $j^*$ maximizing benefit, and the benefit $w_i$ of the second best.
  Compute bid : $b_{ij^*} = a_{ij^*} - w_i + \epsilon$
- For each object $j$ : $P(j)$ is the set of persons who bid for $j$.
  - If $P(j) \neq \emptyset$ : $p_j \leftarrow \max_{i \in P(j)} b_{ij}$ ; remove $(i, j)$ from S, and add $(i^*, j)$ ($i^*$ best bidder)
  - If $P(j) = \emptyset$ , $p_j$ unchanged

# Auction algorithm for optimal transport (1989)

- In O(N A log(N C))
- Idea: convert problem to assignment with duplicated sources/sinks
- Works on similarity classes
- In the previous algo, replace “second best” by “second best among other classes”

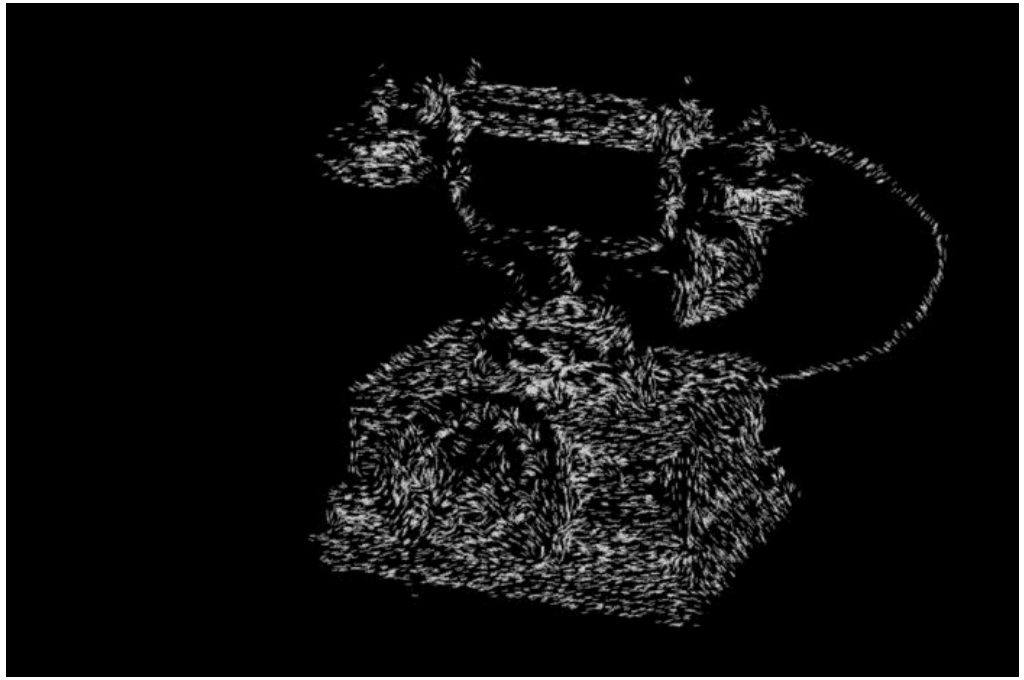
# Interpolation

- Divide Gaussian function w.r.t to transported weights
- We advect.

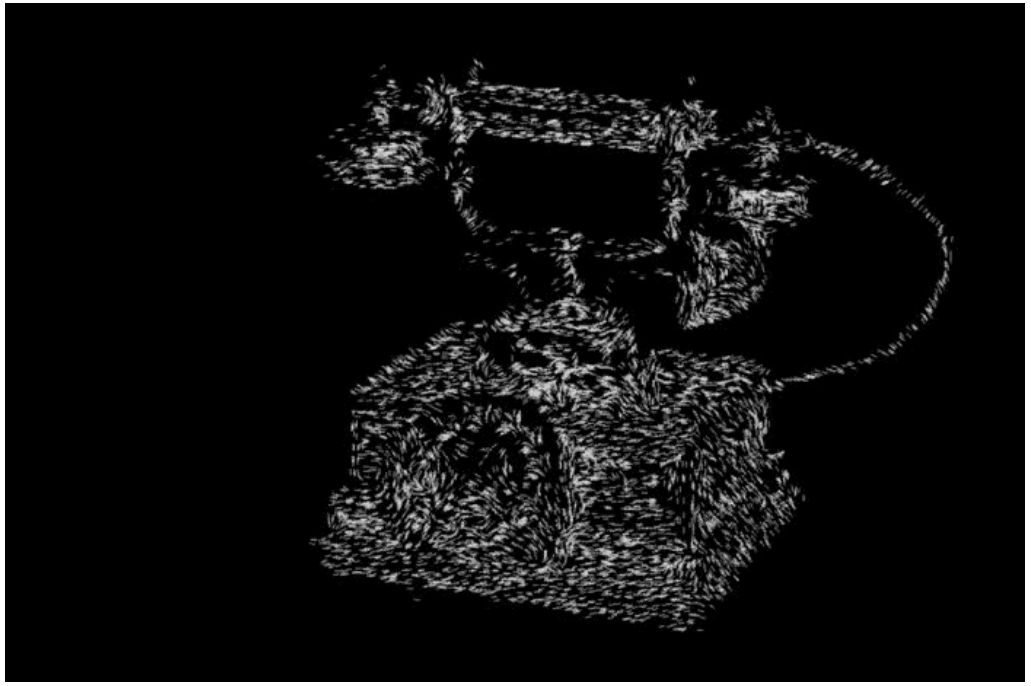


# Results

# Results



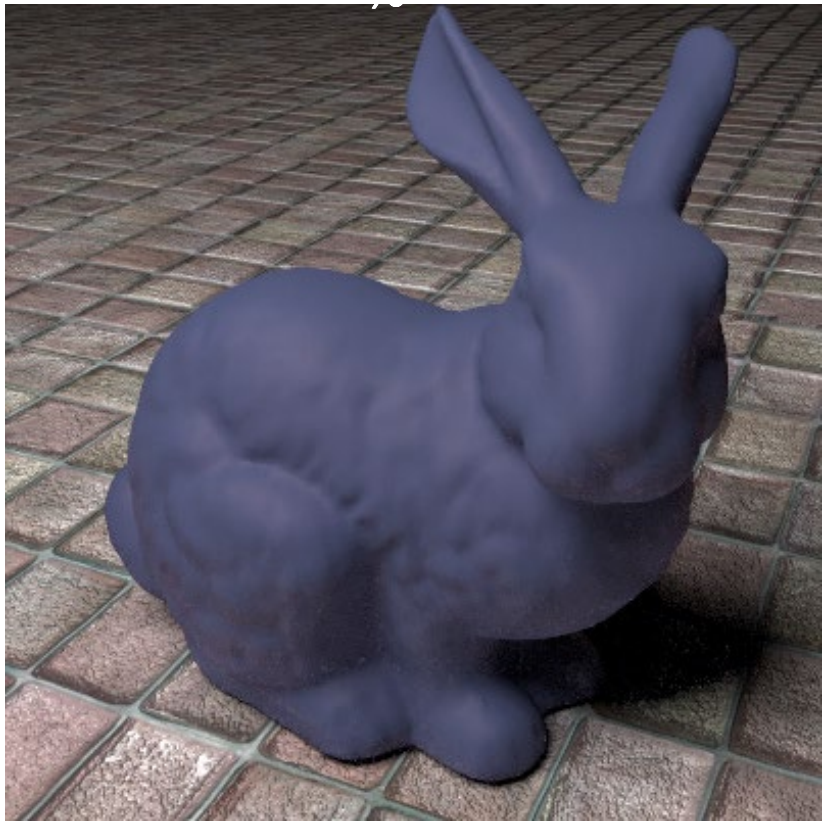
Naive



EMD
(minimize kinetic energy)

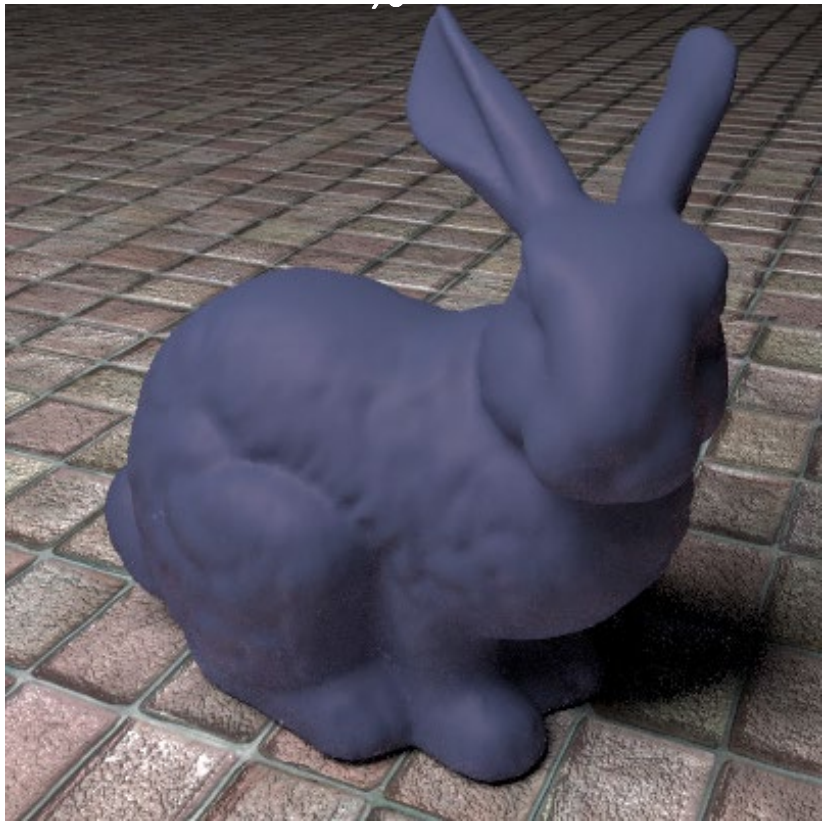
# Results

# Results



Linear interpolation

# Results



Displacement interpolation

# Results

Linear interpolation

Displacement interpolation

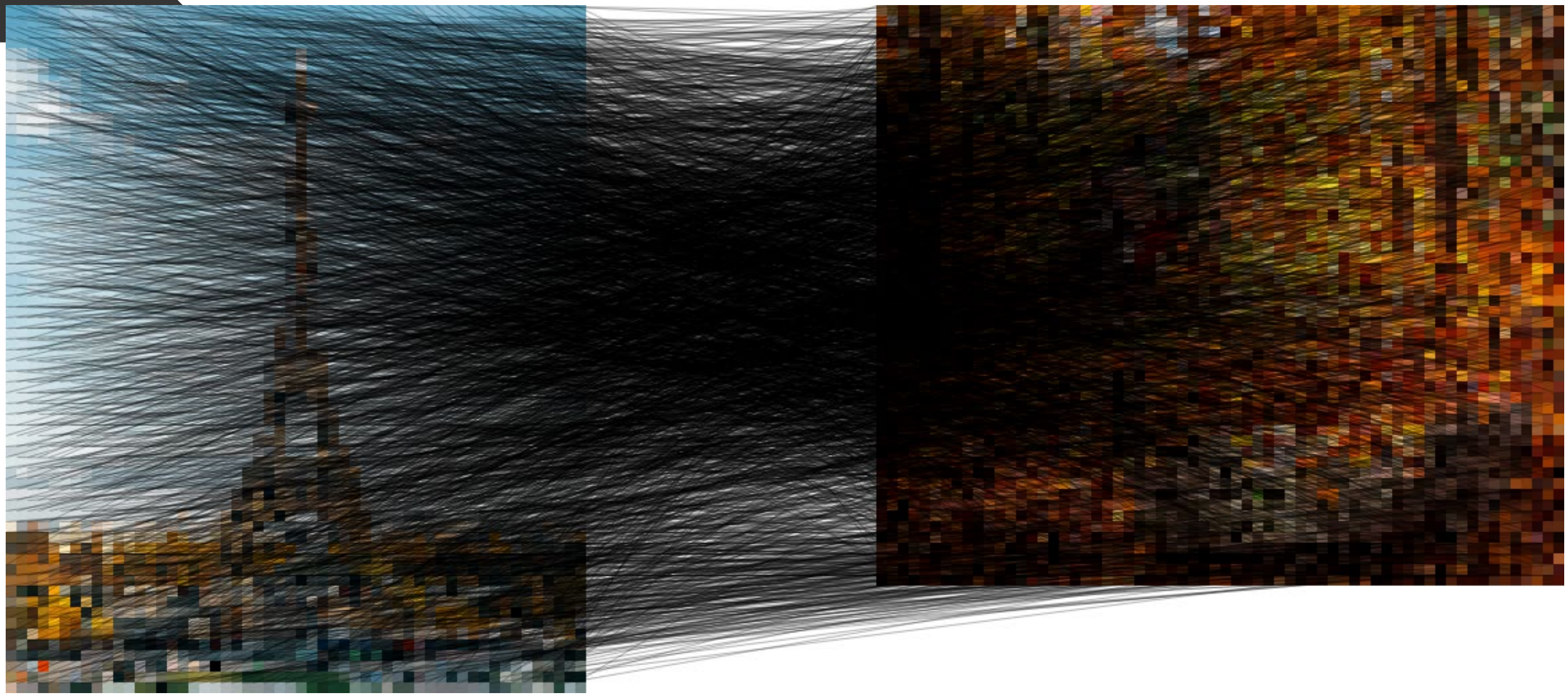
# Applications to Color Grading

Input photo

Target style

Input photo

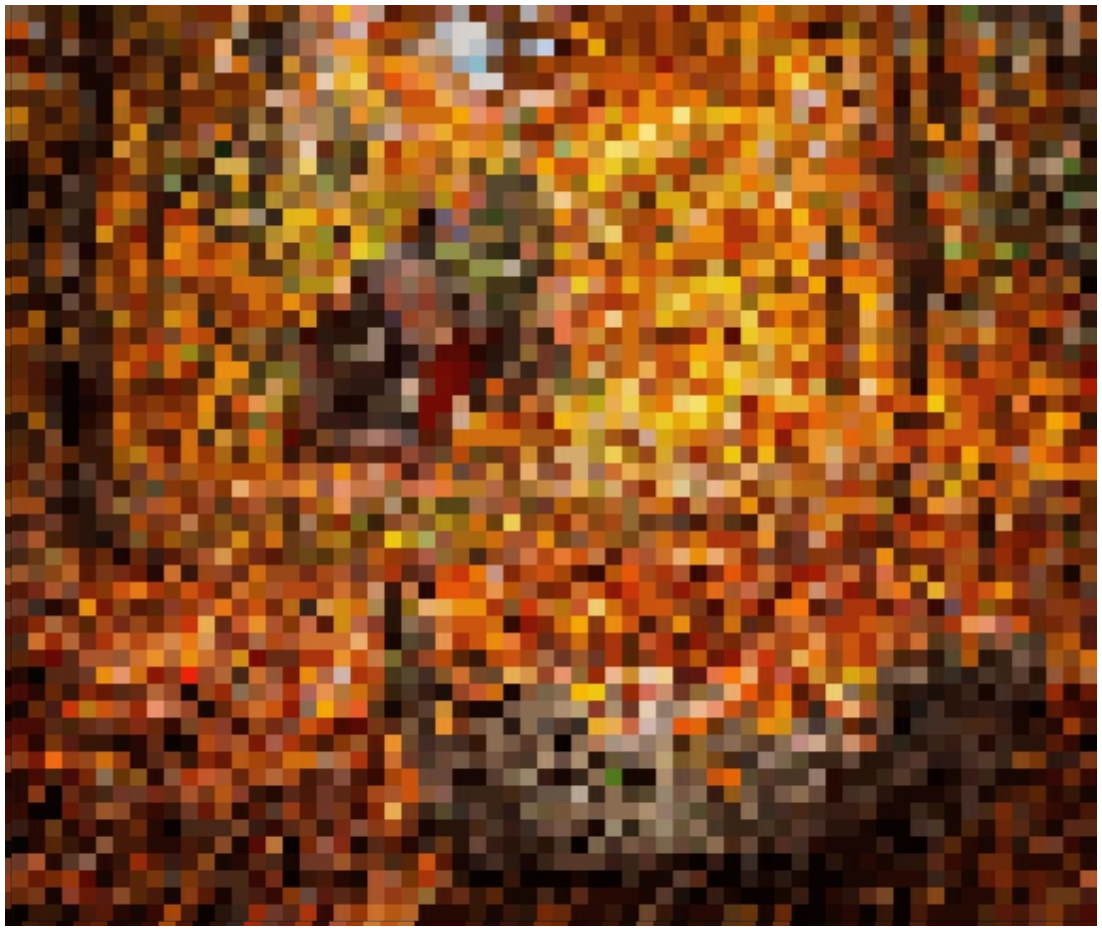
Target style



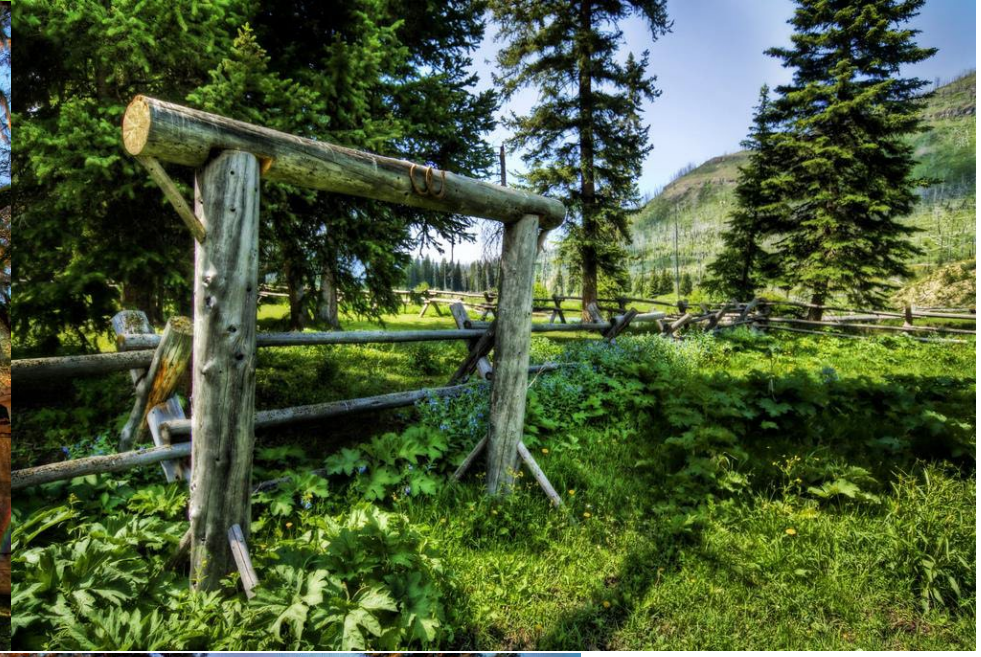
Input photo

Target style

Input photo

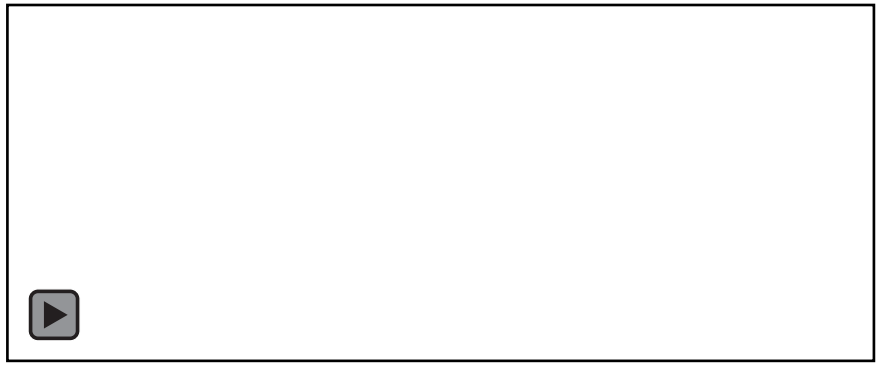


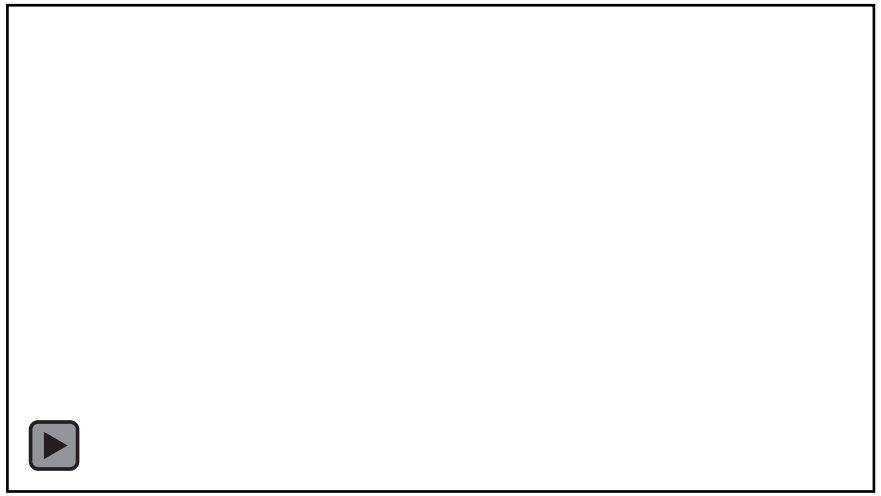
Target style

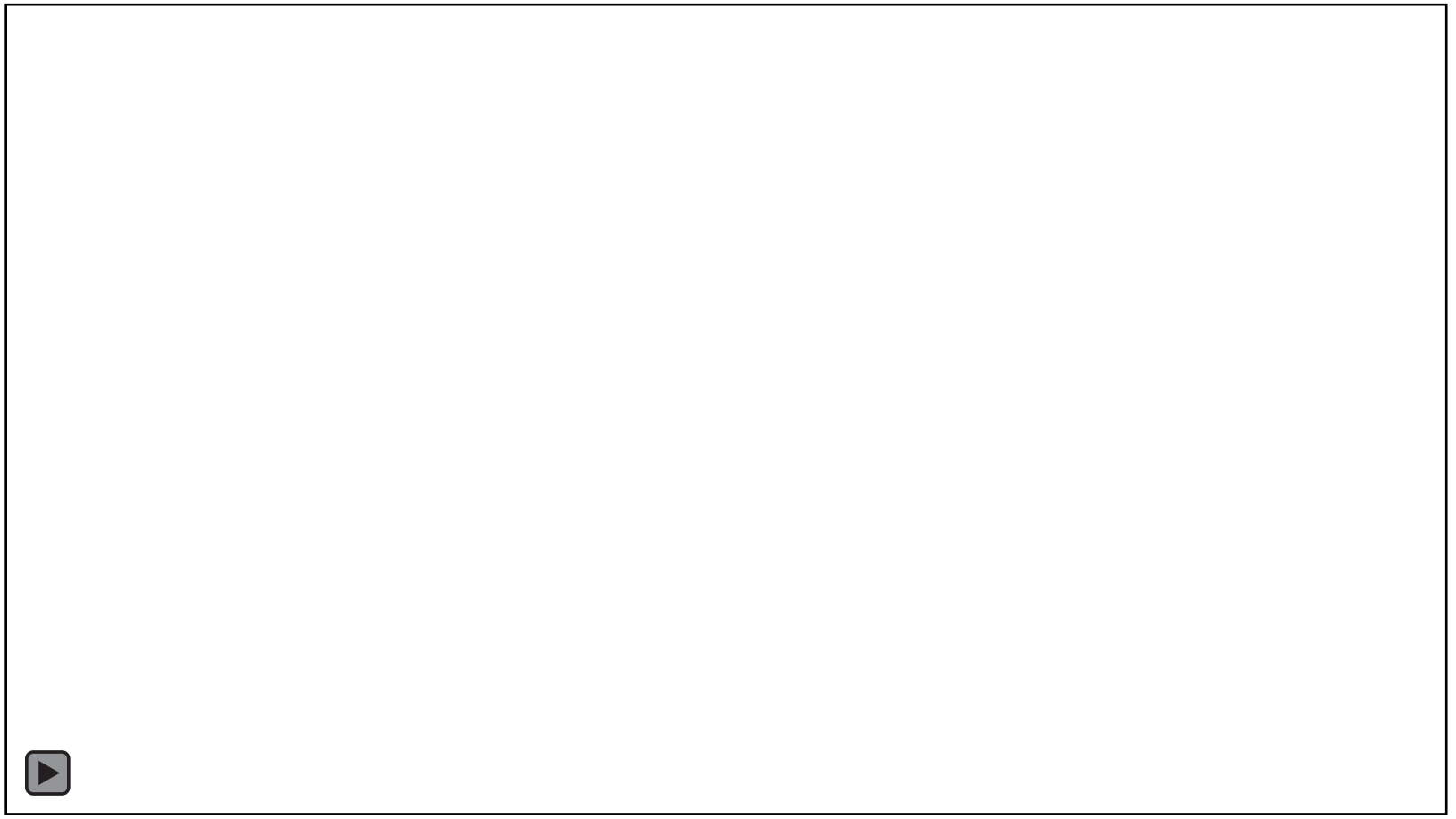


# Results

Model







Input

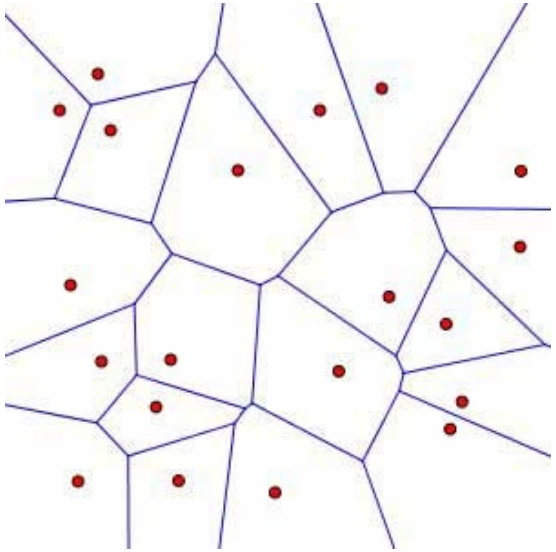
# Semi-discrete optimal transport

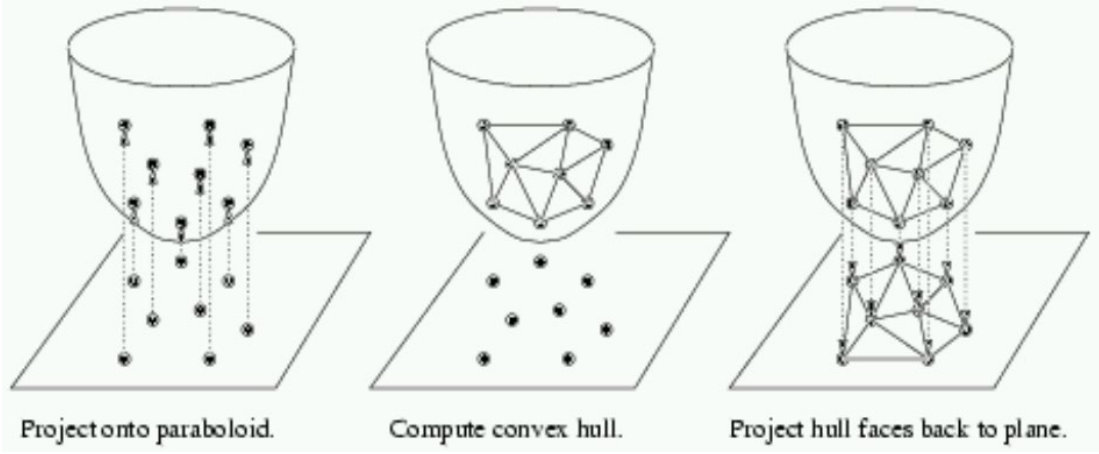
# Voronoi diagram

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

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

- 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





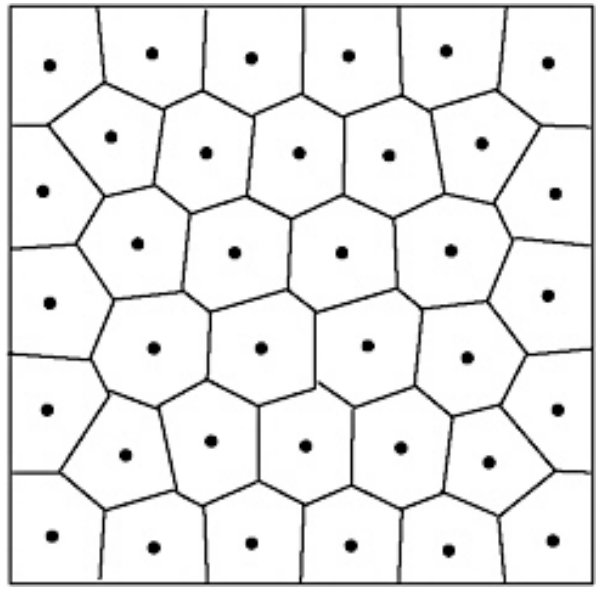
# Centroidal Voronoi Diagram

- Can be defined as the solution to a least-square problem

$$\min \int_{Vor_i} \sum_i \|x - x_i\|^2 dx$$

  Also says that the centroid of $Vor_i$ is the site $x_i$

- Can be computed by:
  - A Lloyd clustering algorithm
  - A descent approach on the above energy

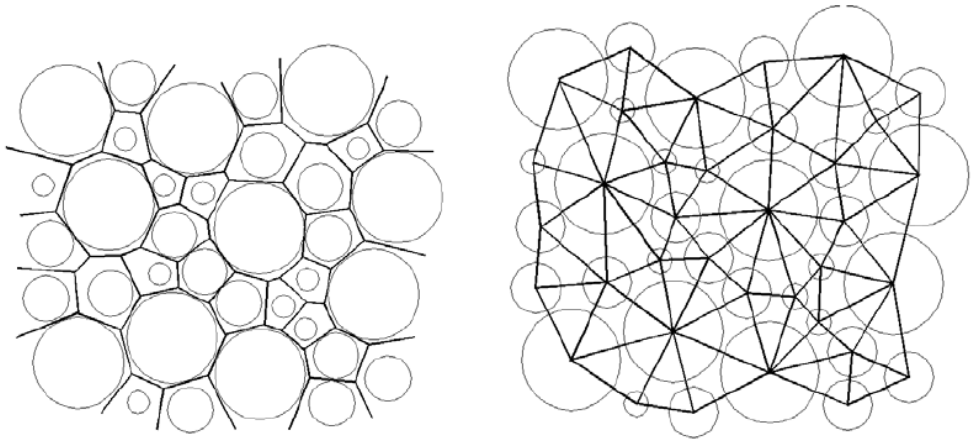


# Power diagram (Laguerre diagram)

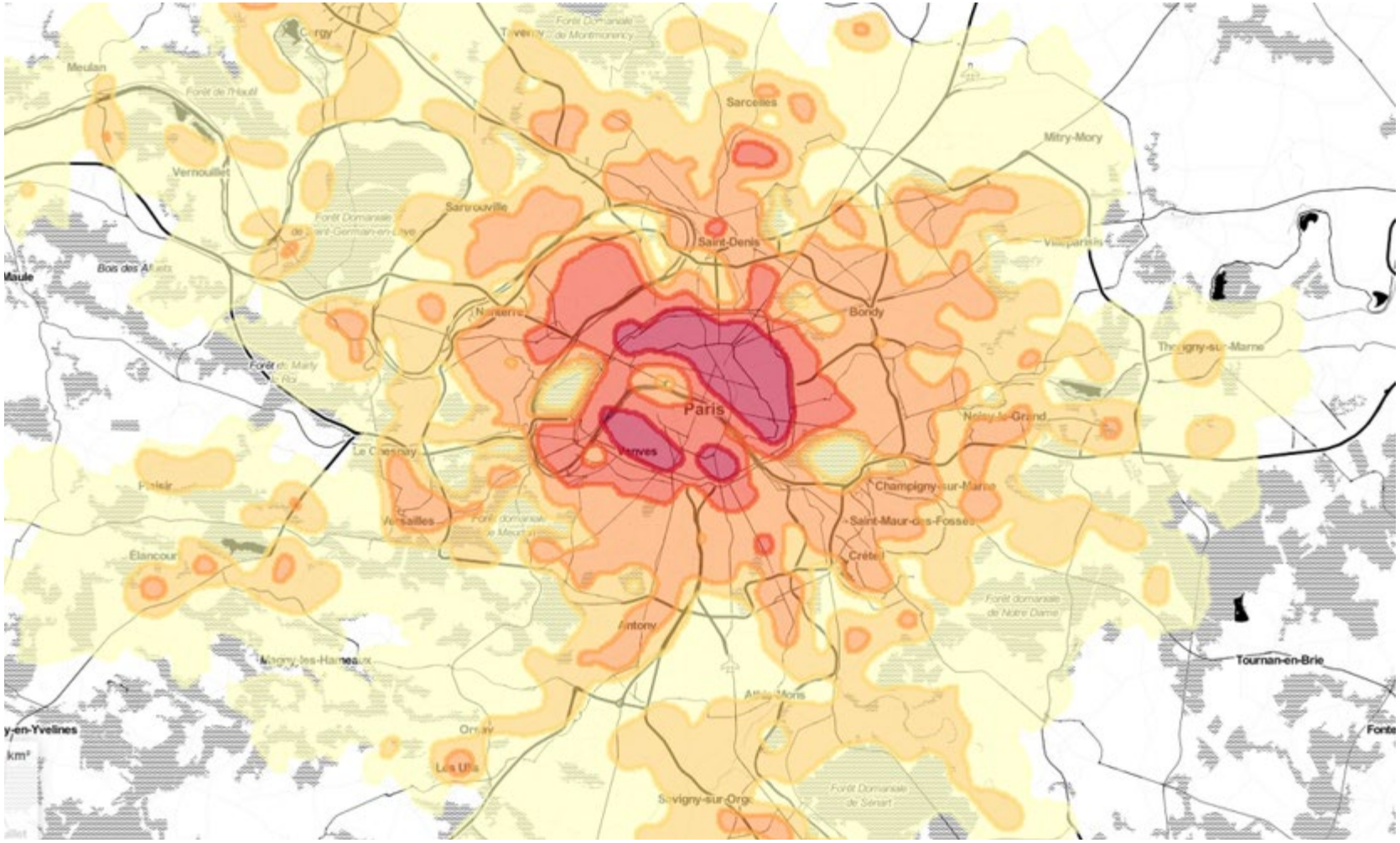
- A partition s.t. each point $x$ is assigned to its closest site $x_i$ with weight $w_i$

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

- 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 $\|x' - x_i'\|^2 \leq \left\|x' - x_j'\right\|^2 \ \forall j$
- Any partition into convex polyhedral cells is a power diagram of some sites

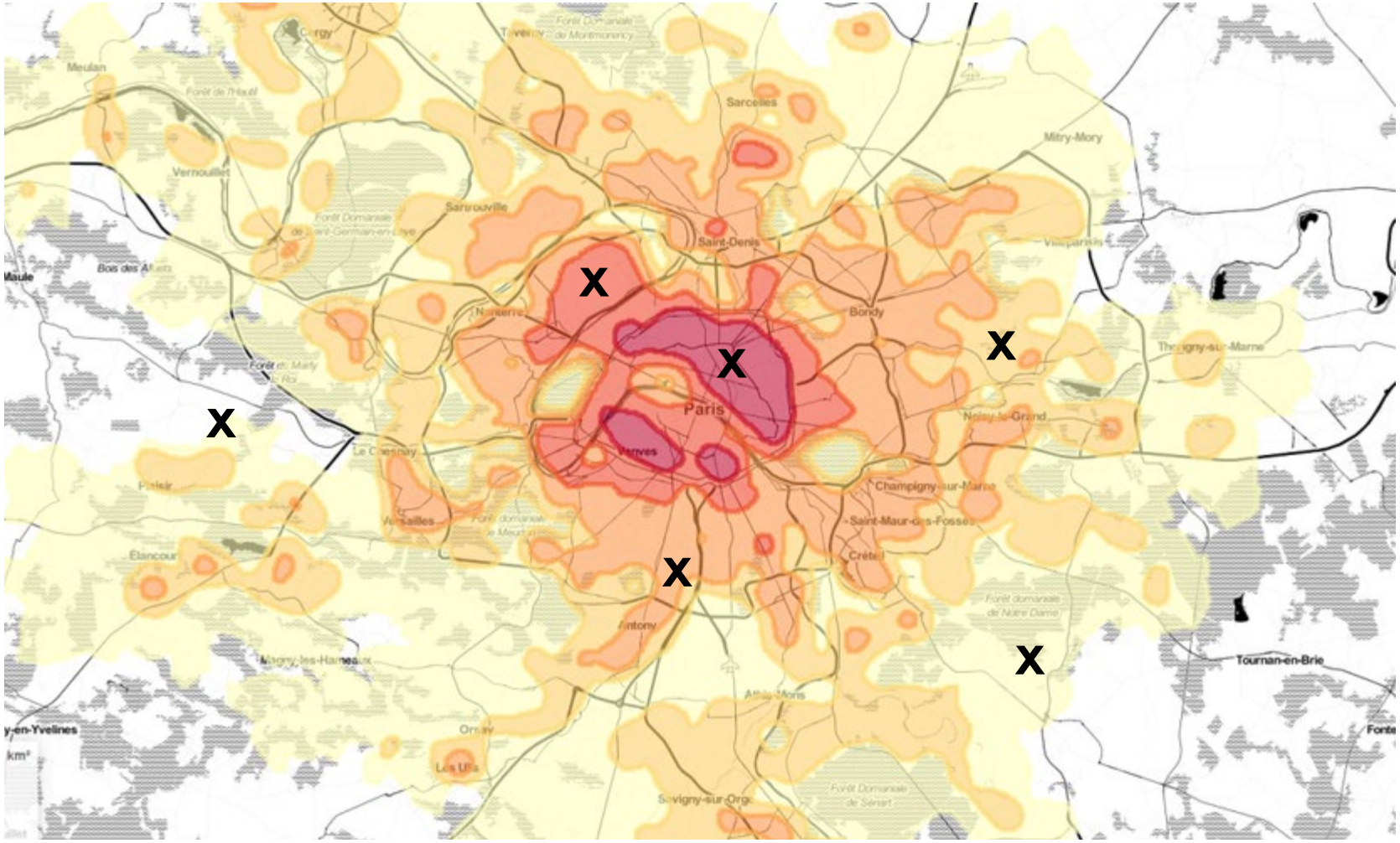


# Semi-discrete Optimal Transport



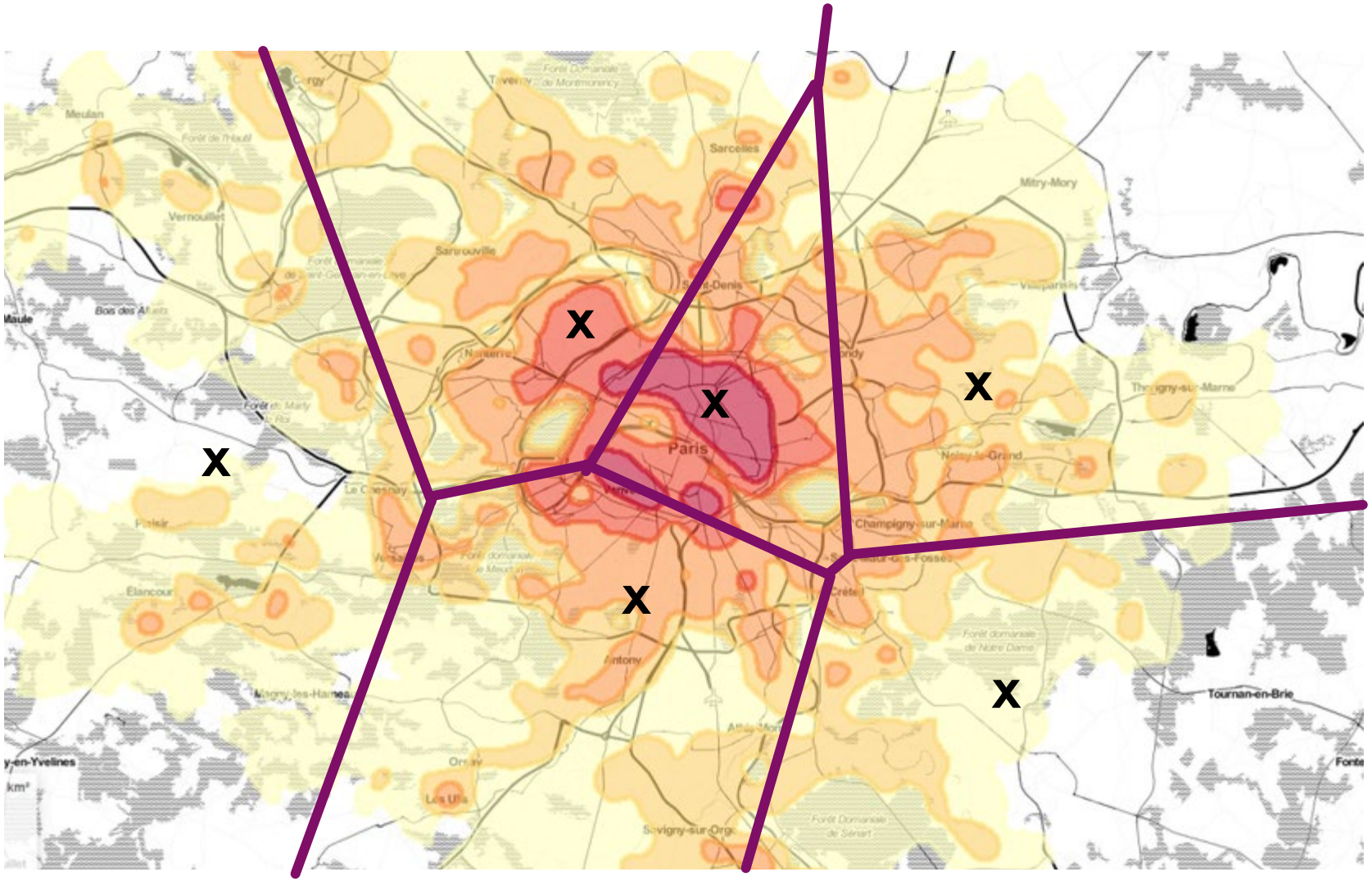

Population density $f$

# 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 populat

# Semi-discrete Optimal Transport

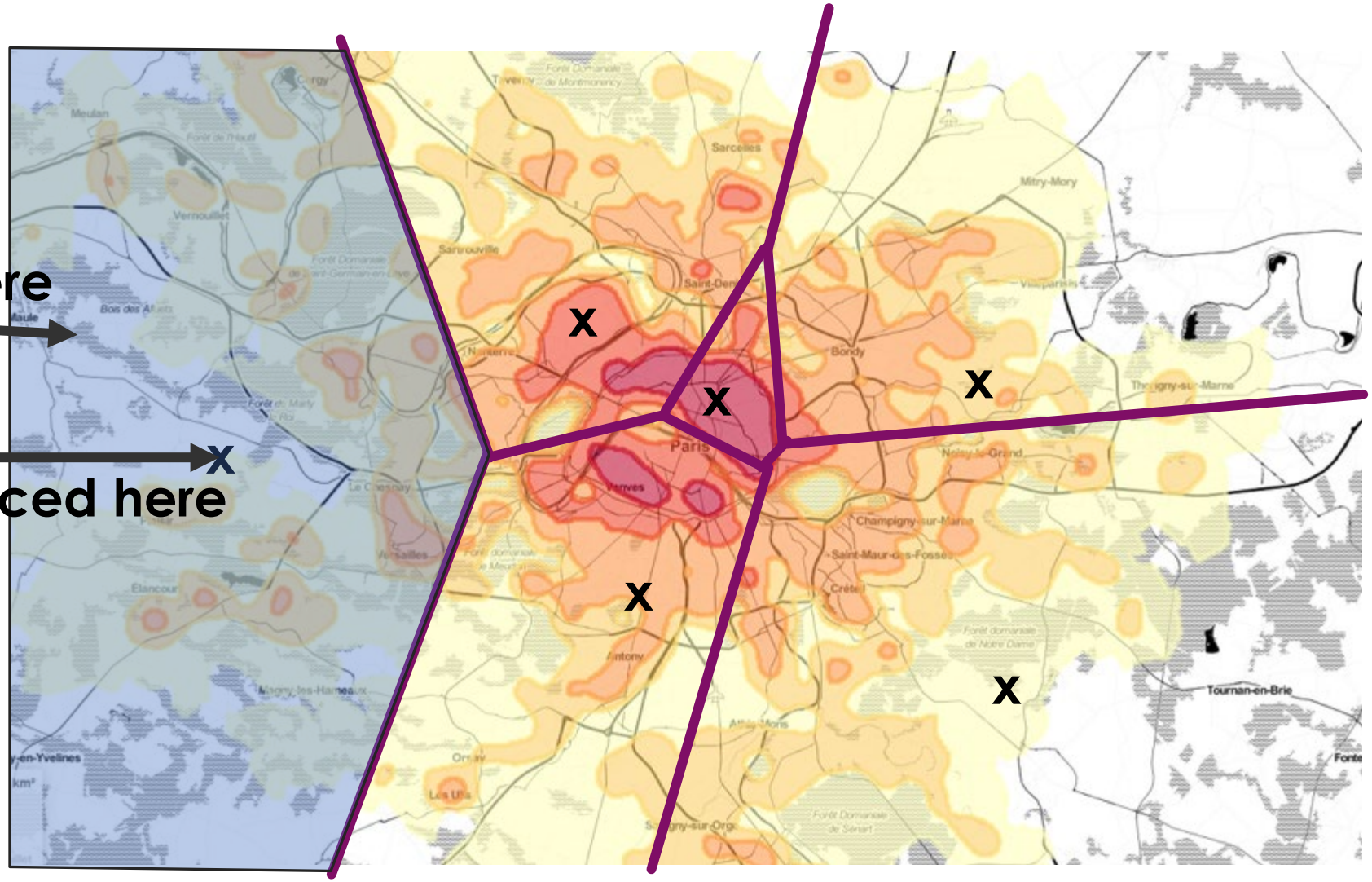

Limited production: population go to the nearest bakery/factory **with sufficient production**!

# Semi-discrete Optimal Transport

**(needs for) population here**

**=**

**quantity produced here**




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

# Back to optimal transport

- Optimal transport (Monge version) :

$$\min \int \|x - T(x)\|^2 \, d\mu(x)$$

Considering $\mu$ is continuous with density $\rho$

$$\min \int \|x - T(x)\|^2 \, \rho(x) dx$$

Considering $\nu$ (the target measure) discrete: $\nu = \sum \lambda_p \delta_p$

The mass preservation constraint is:

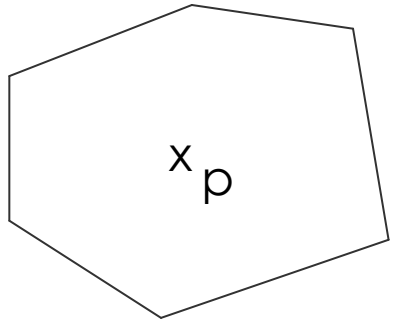
$$\lambda_p = \int_{T^{-1}(\{p\})} \rho(x) dx$$

A Multiscale Approach to Optimal Transport [Mérigot 2011]
Minkowski-Type Theorems and Least-Squares Clustering [Aurenhammer et al. 98]

# Back to optimal transport

- In this case : $T^{-1}(\{p\}) = Vor^W(p)$

a power cell for some weight $w_p$

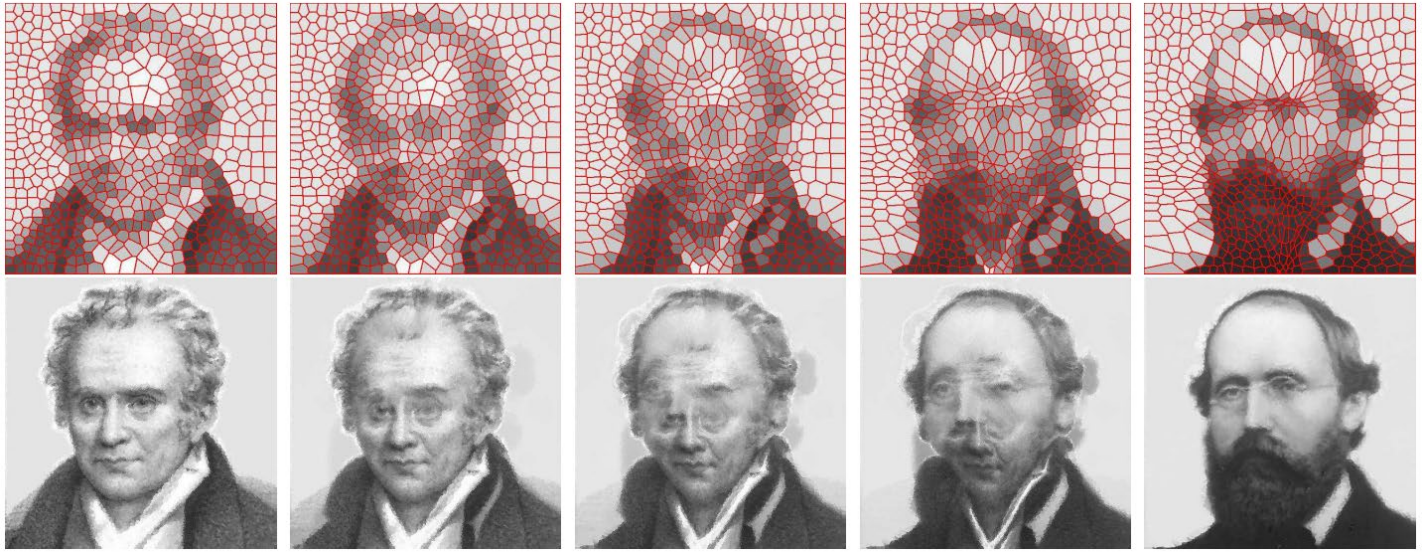



- This determines a partition, so Monge problem is:

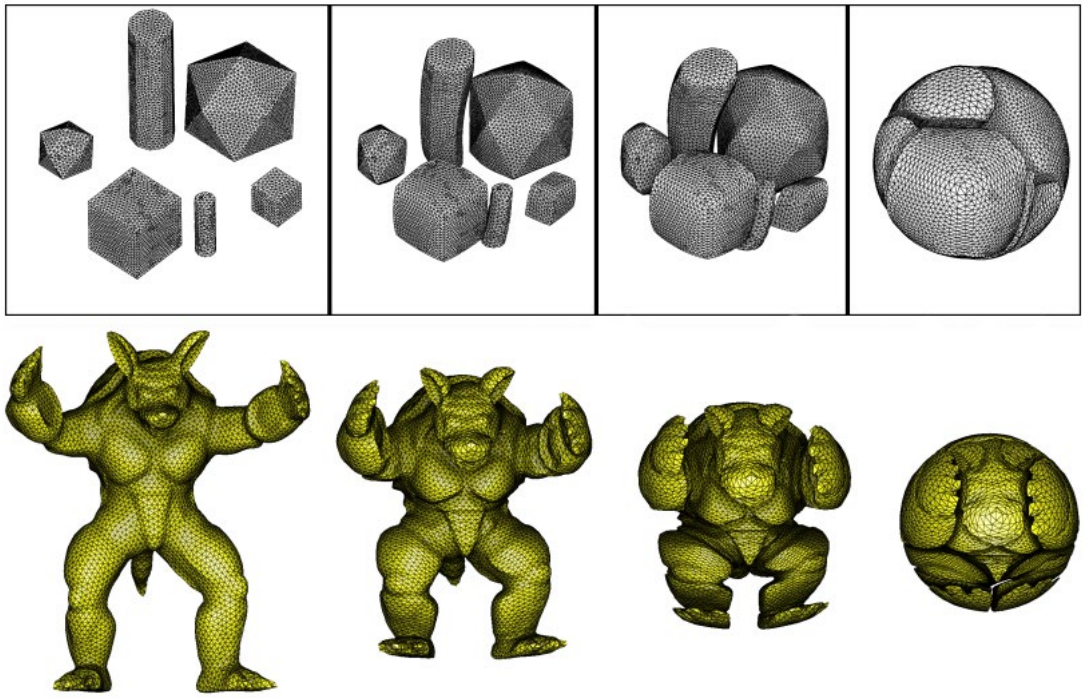
$$\min \sum_p \int_{Vor^W(p)} \|x - p\|^2 \, \rho(x) \, dx$$

- Idea: optimize weights $w$ for each site to grow/shrink power cells until $\lambda_p = \int_{T^{-1}(\{p\})} \rho(x) dx$
- Gradient of appropriate functional given by $\frac{\partial \phi}{\partial w(p)}(w) = \lambda_p - \int_{Vor^W(p)} \rho(x) \, dx$

# Back to optimal transport



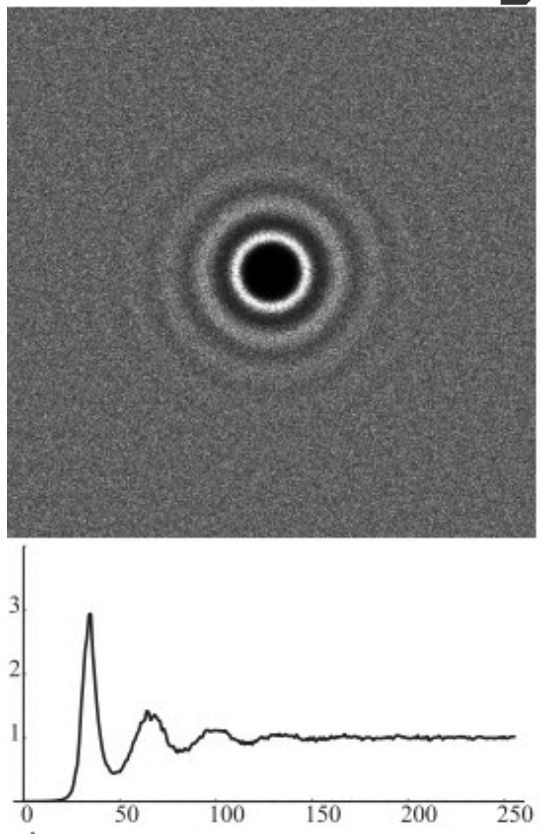
A Multiscale Approach to Optimal Transport [Mérigot 2011]



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

# Application

- Also optimizes for the locations $p$




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

# 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, $\epsilon$ 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:

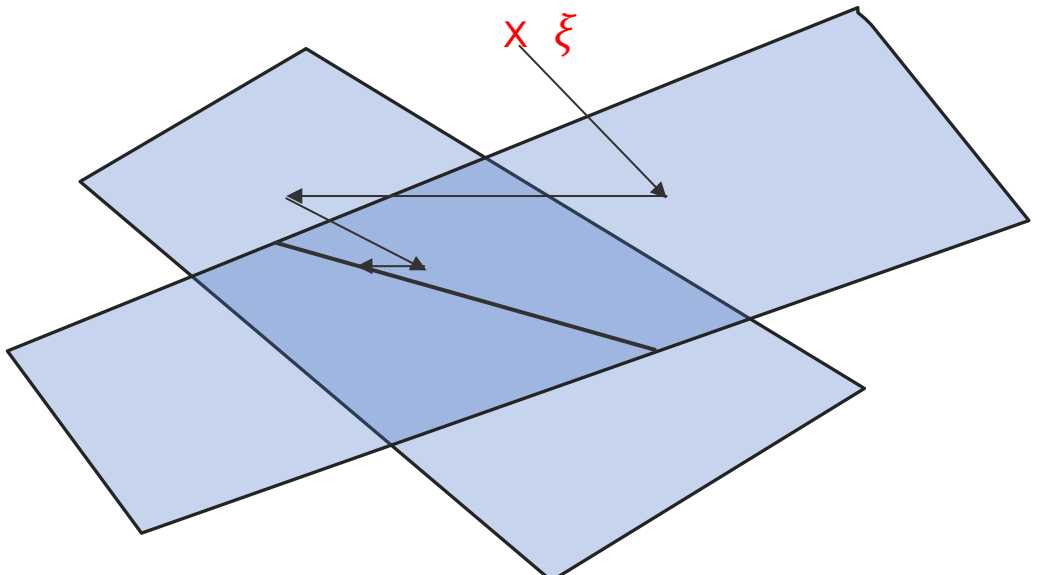
$$\min_{M \in \mathcal{U}(r,c)} KL(M, \xi)$$

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



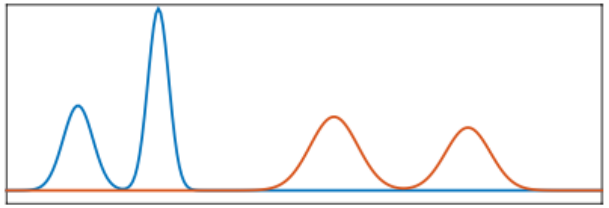
# The Sinkhorn algorithm

- 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

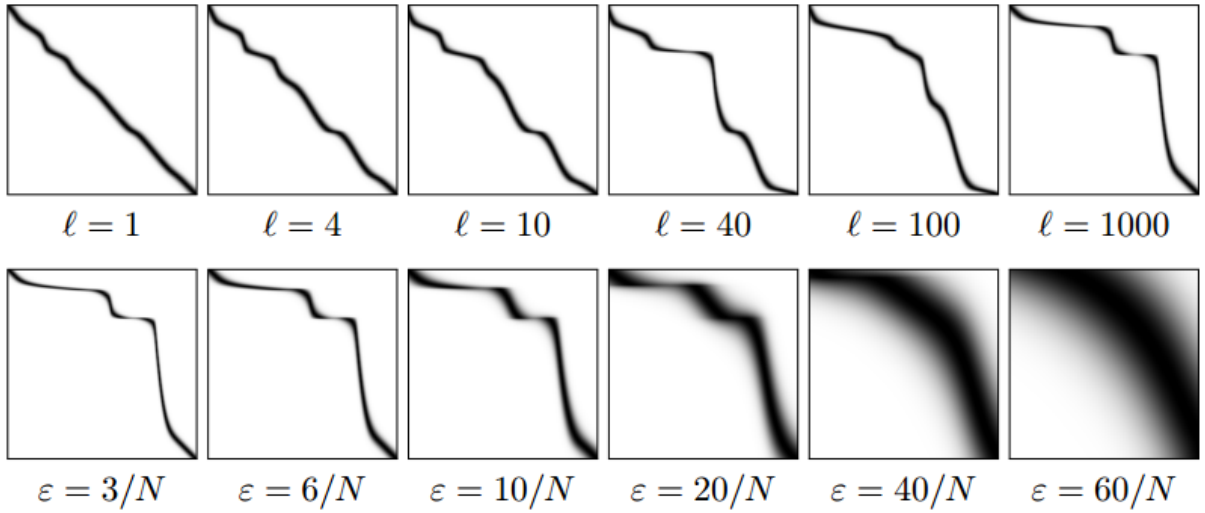
# The Sinkhorn algorithm

- We can thus apply Bregman projections: we iteratively project on each constraint
- We obtain the algorithm:

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

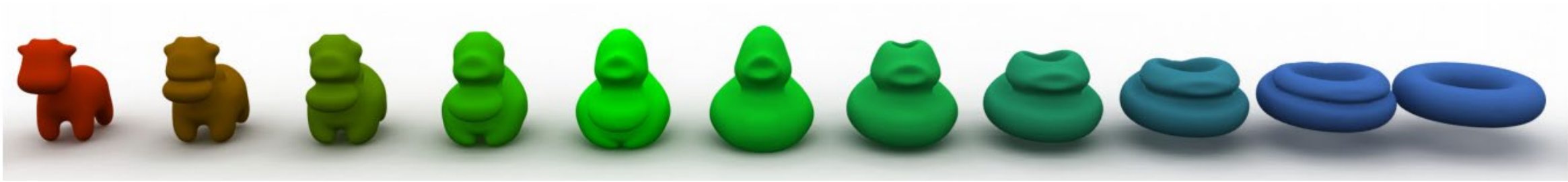


Marginals $p$ and $q$



# 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

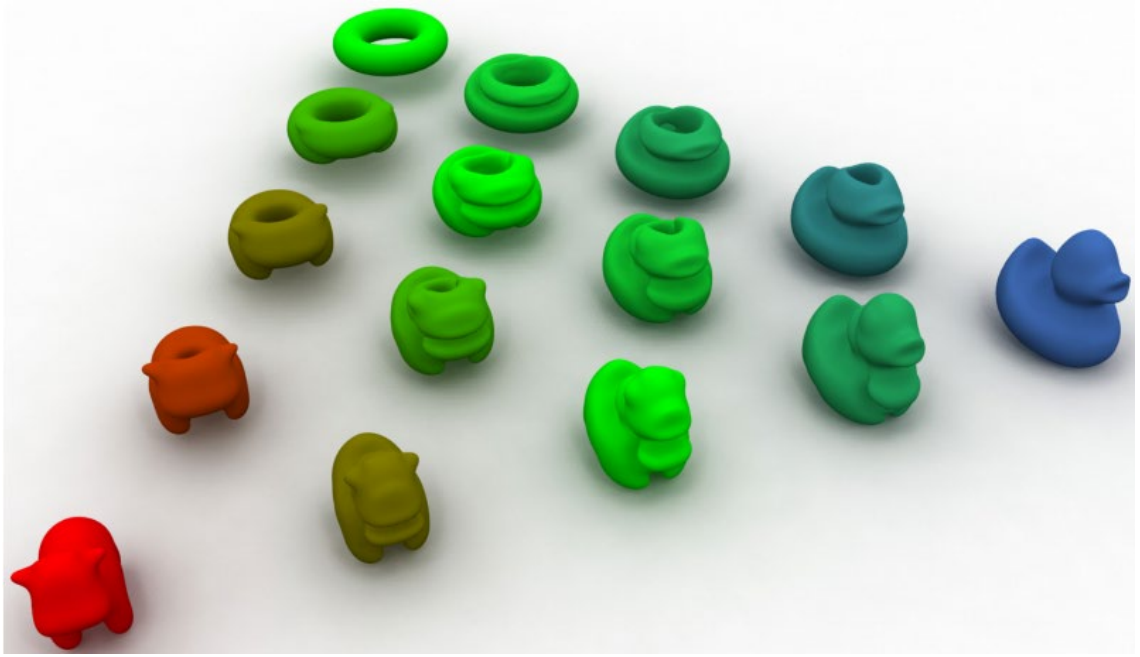


Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains [Solomon et al. 2015]

# The Sinkhorn algorithm

- Generalized to compute displacement interpolation and barycenters

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

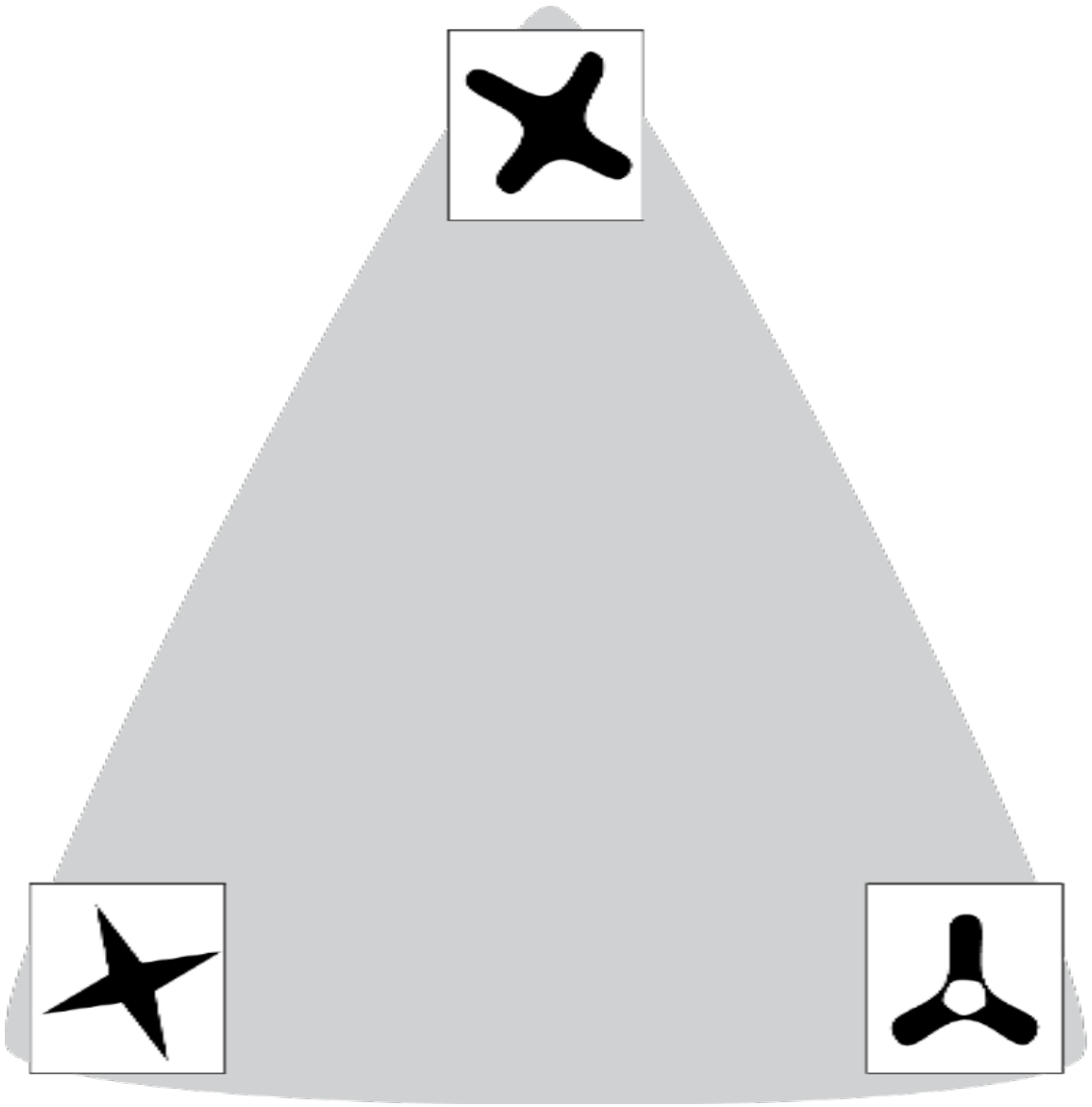


# Wasserstein Barycentric Coordinates: Histogram Regression Using Optimal Transport

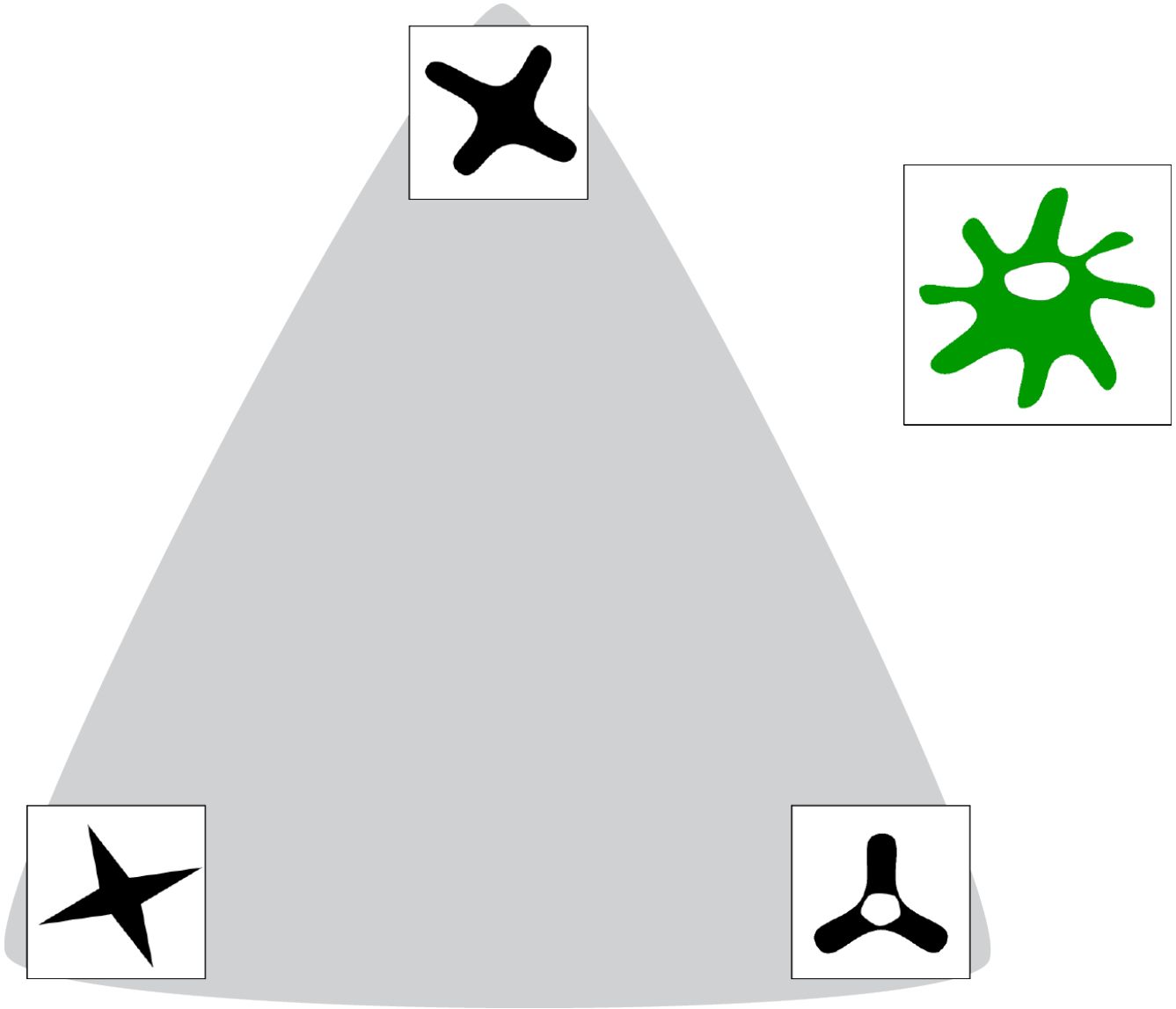
N. Bonneel, G. Peyré, M.Cuturi

SIGGRAPH 2016

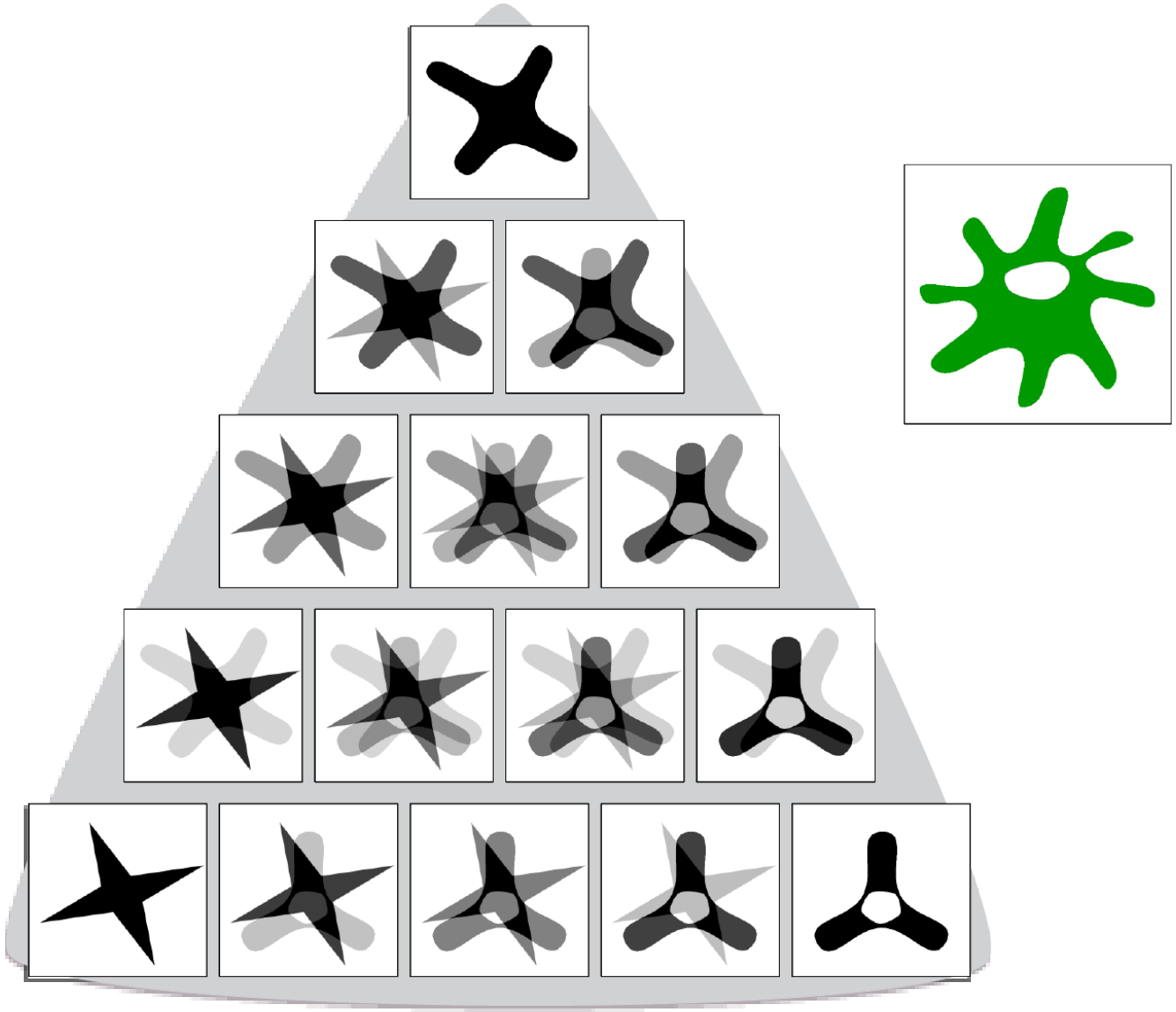
# Barycentric coordinates



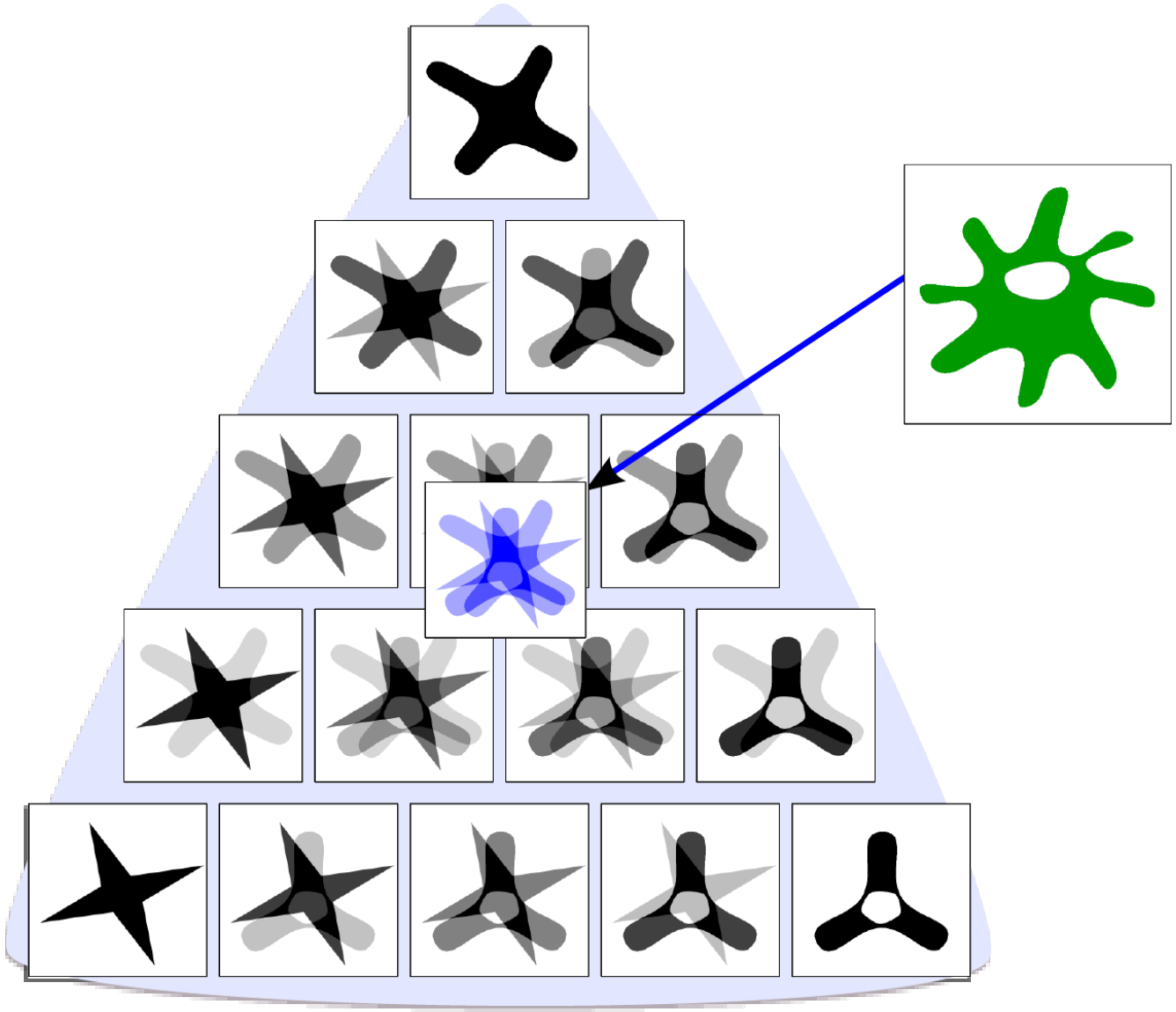
# Barycentric coordinates



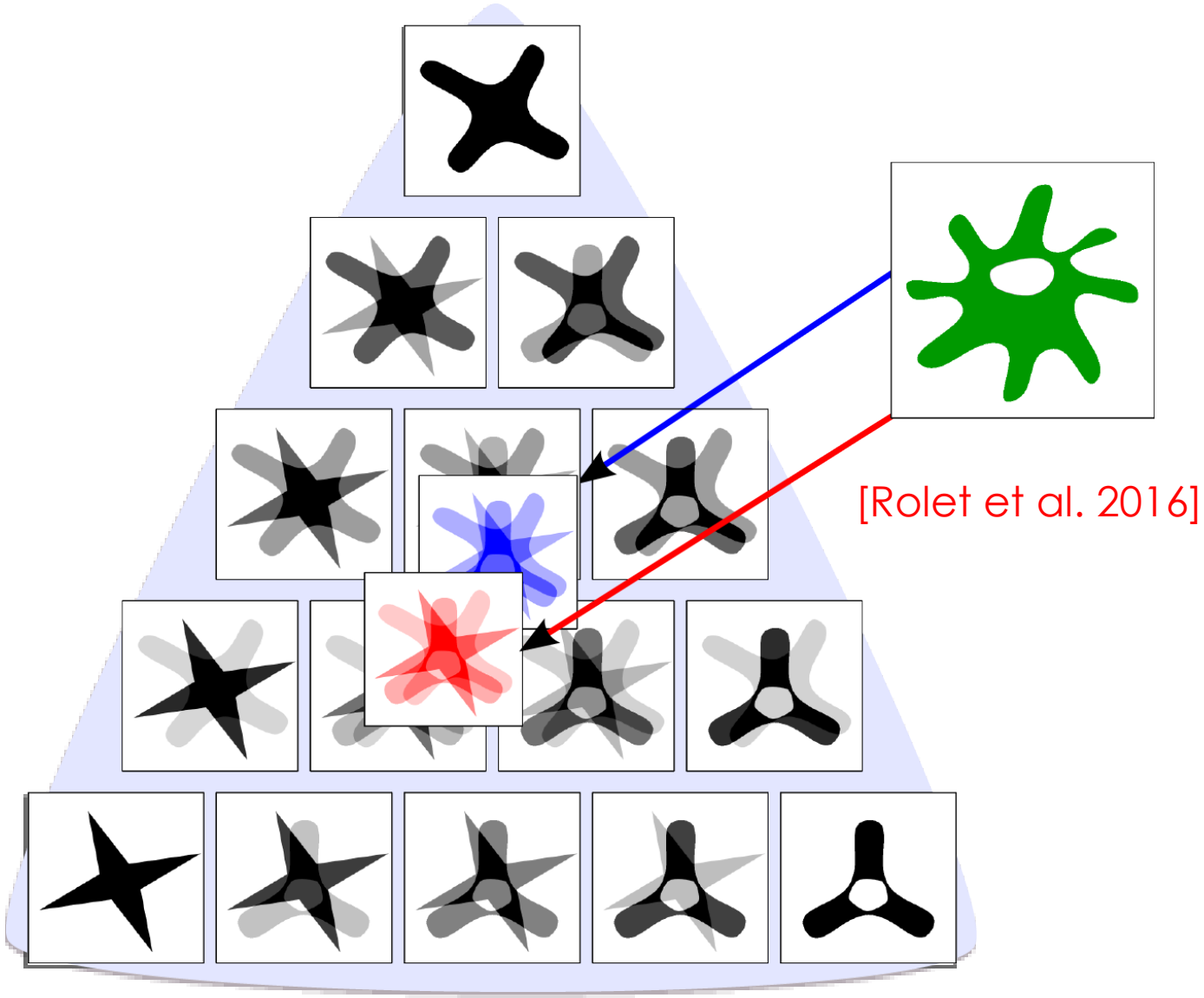
# Barycentric coordinates



# Barycentric coordinates

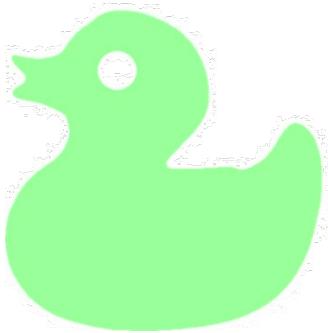


# Barycentric coordinates

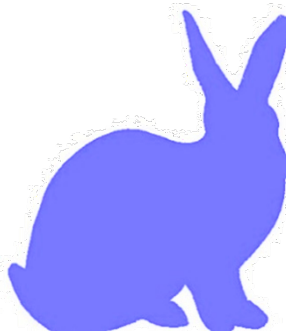


[Rolet et al. 2016]

# Optimal Transport



$t = 0$



$t = 1$

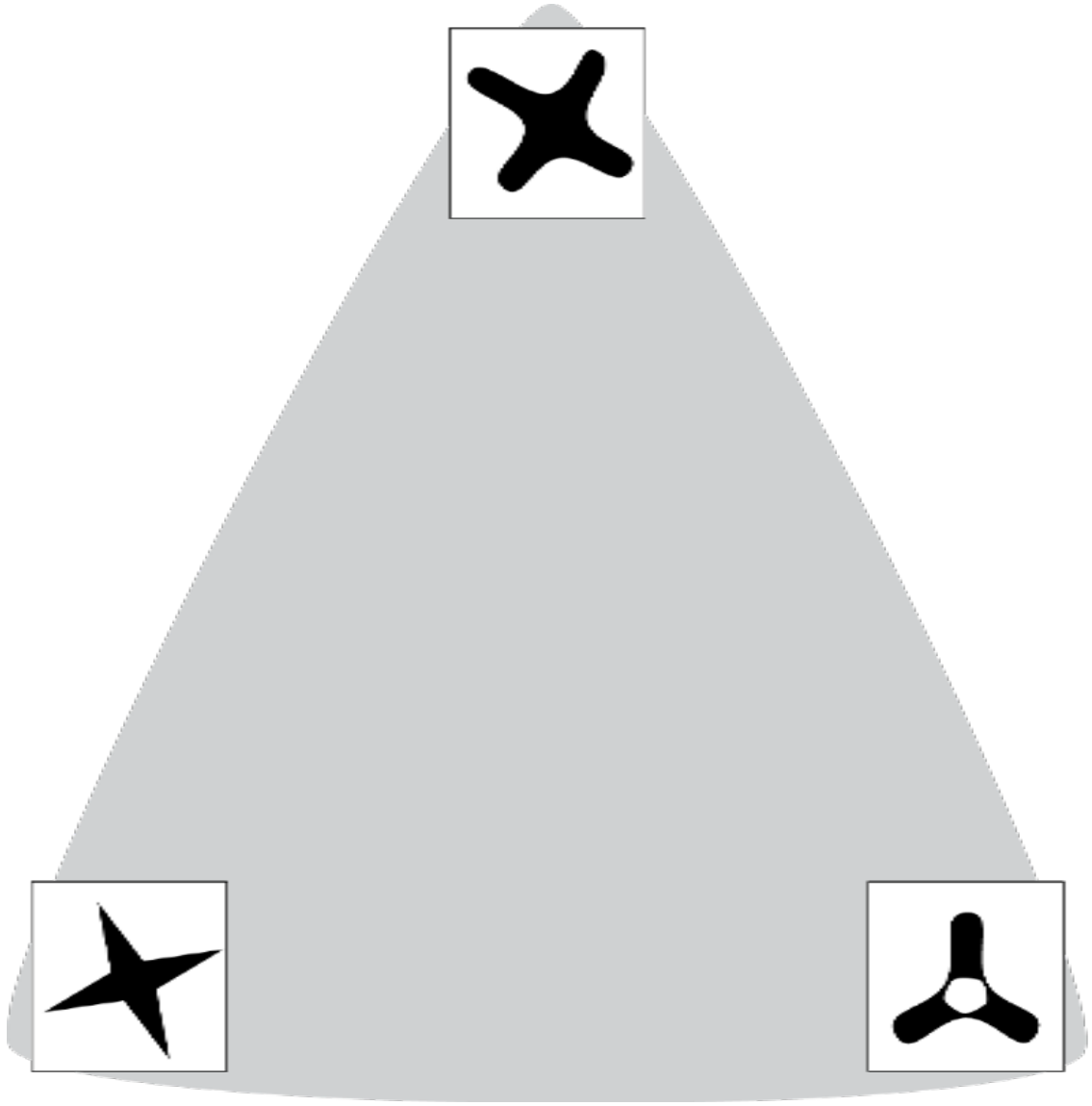
# Optimal Transport

$t = 0$ $t = 1$

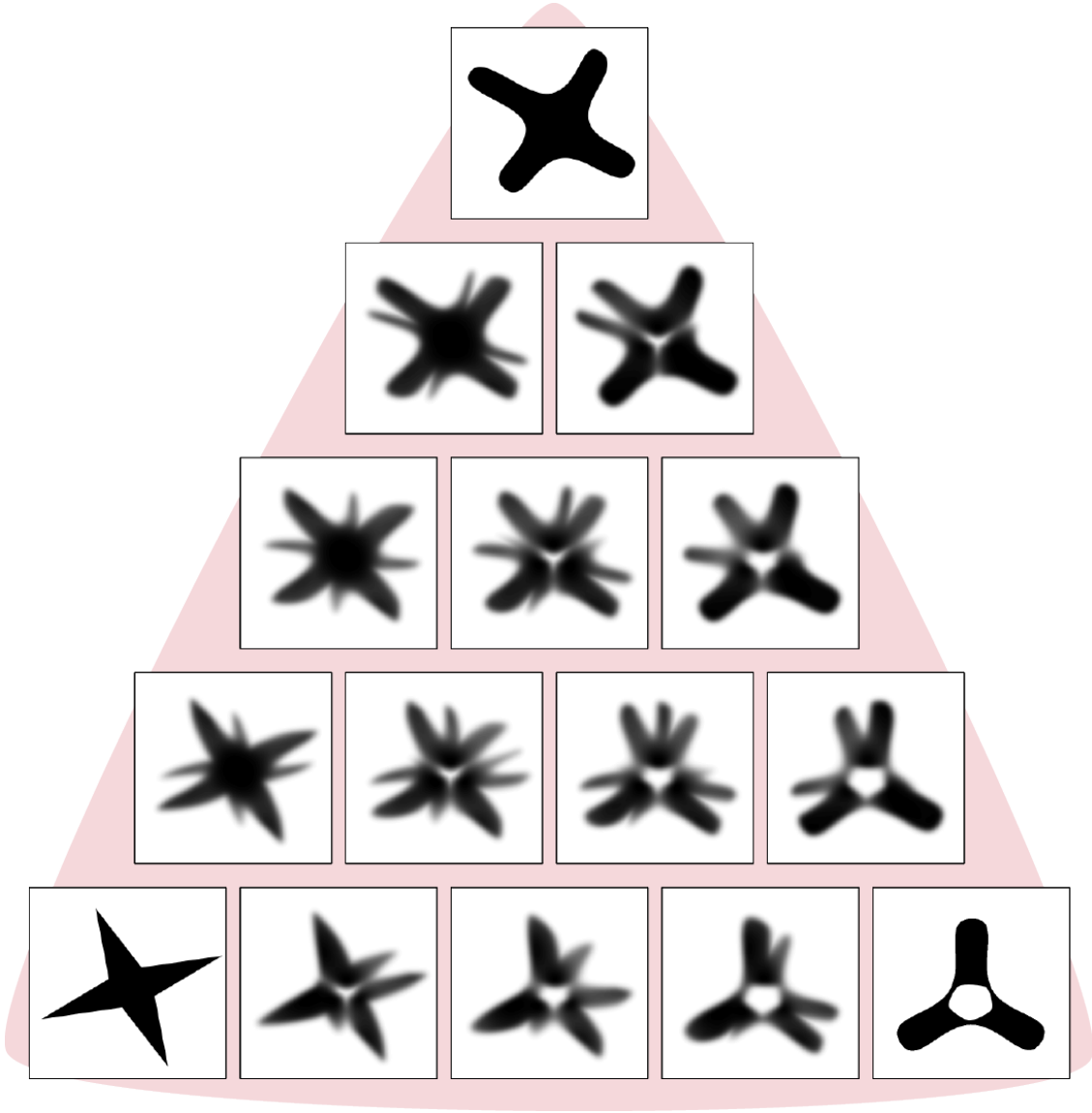
$$\mathrm{W}(f,g) = \min \sum\sum \left\| x_i - x_j \right\|^2 m_{ij}$$

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

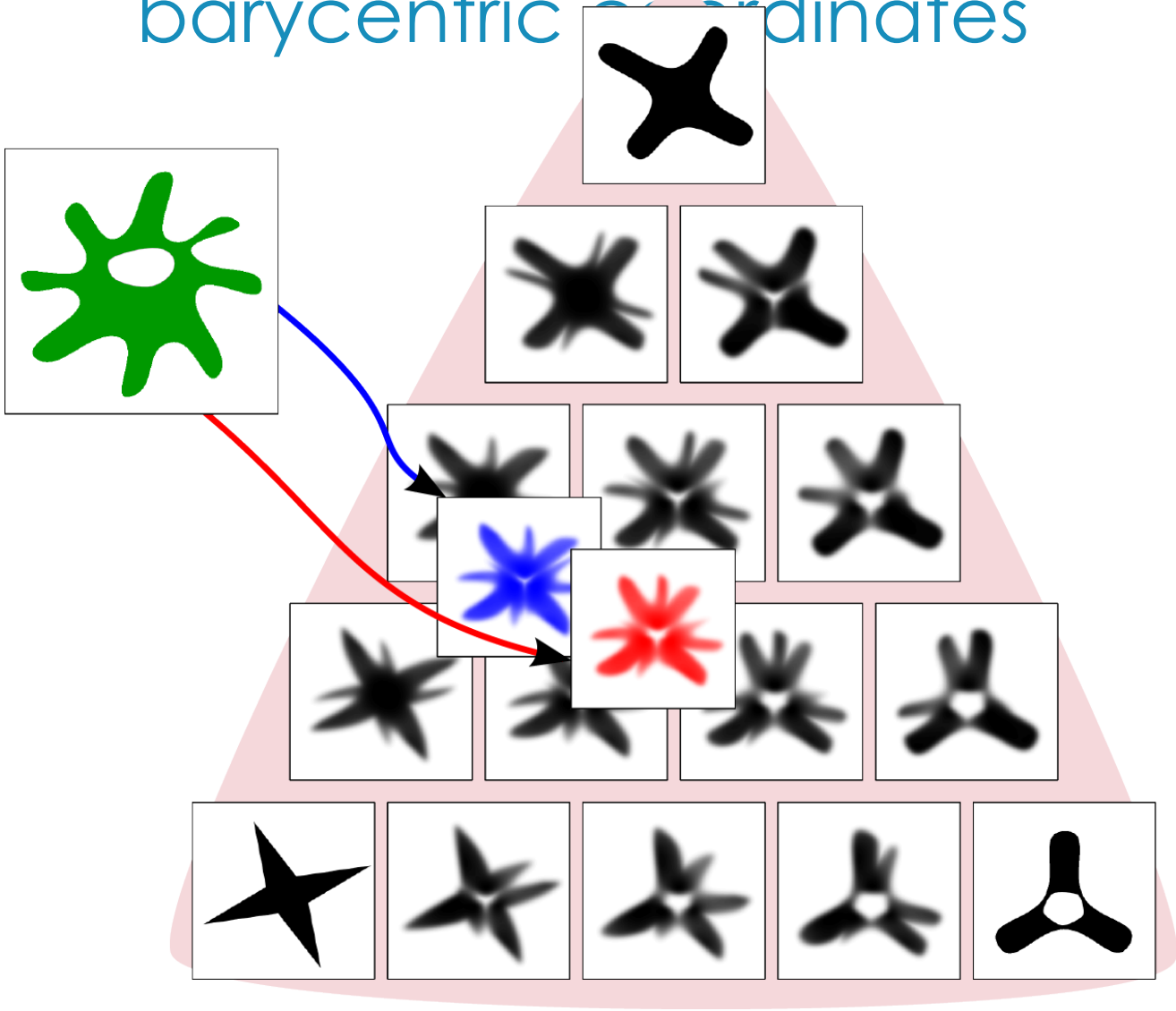
# Optimal Transport

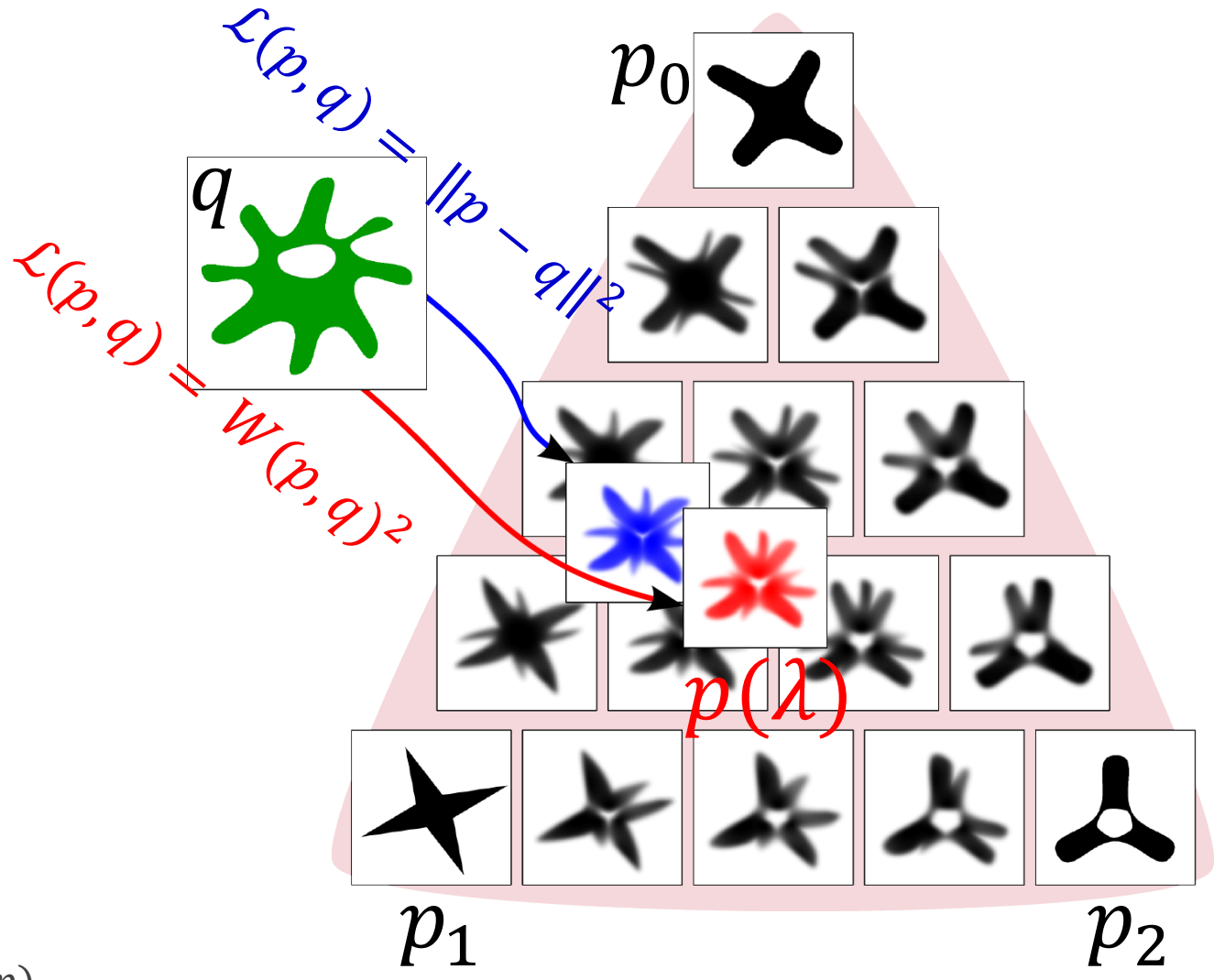


# Optimal Transport



# Optimal transport barycentric coordinates






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 :

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

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

# Idea

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

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

  - Then

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

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

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

# Gradient computation

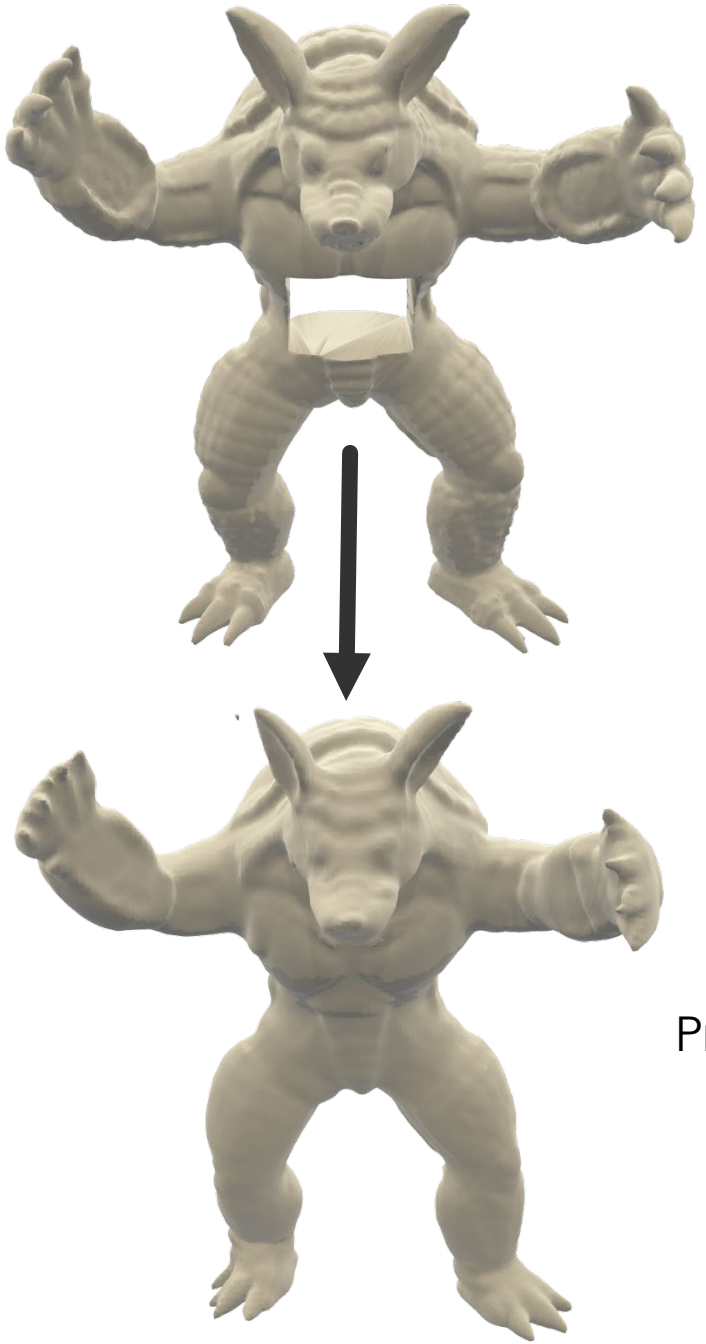
- We obtain:

- $q_s = 0 \; ; r_s = 0 \;\; \forall s$
- $g \leftarrow \nabla \mathcal{L}(p(\lambda), q) \odot p(\lambda)$
- 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



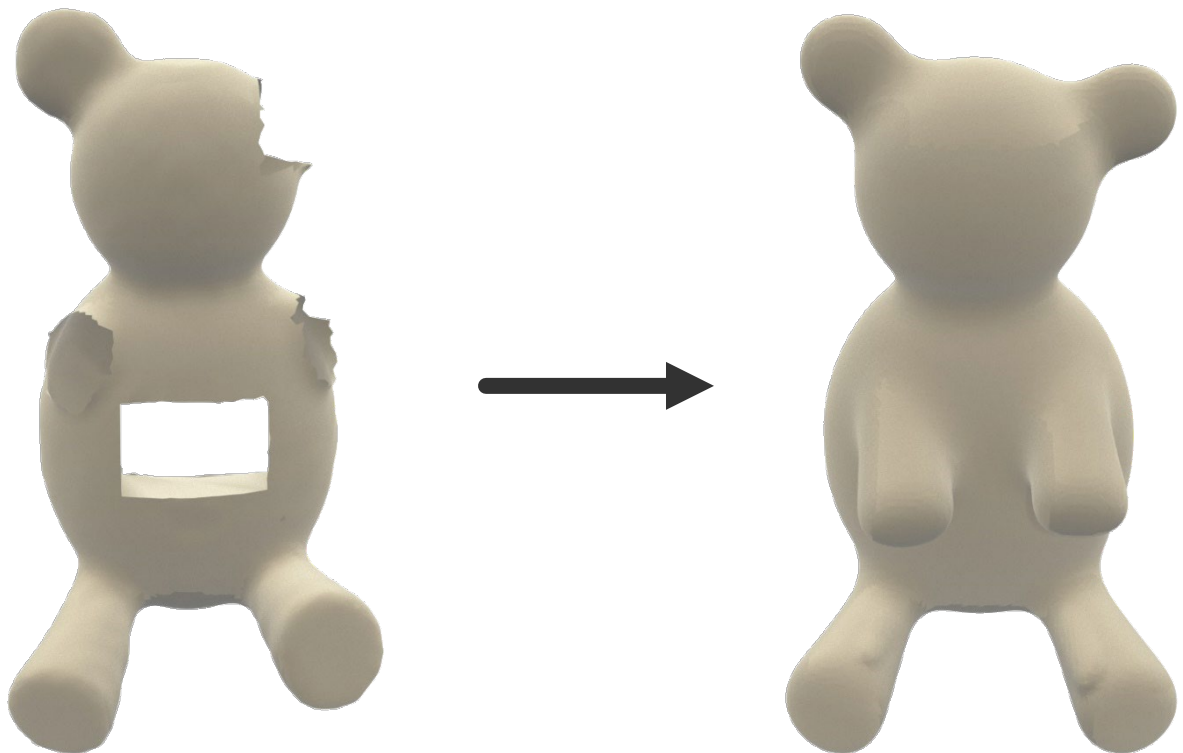
Input

Projection

Euclidean

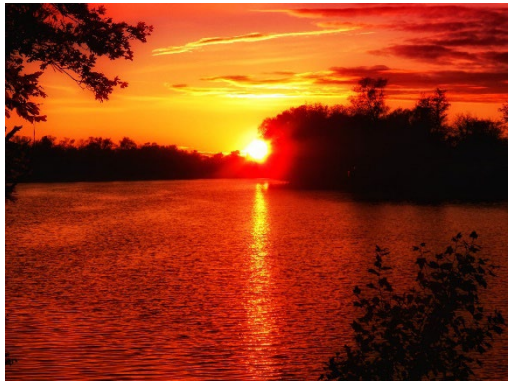
Wasserstein


Database


Input
Projection

# Applications

3E-6



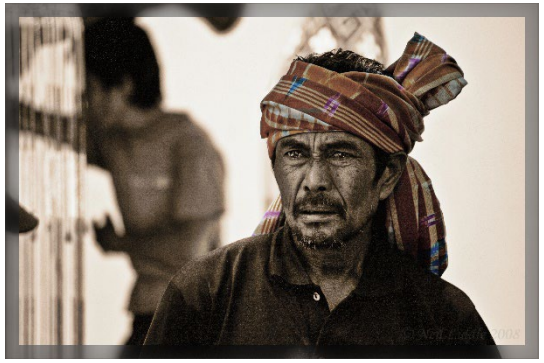
6E-6

0.23

0.77

Database

Projection



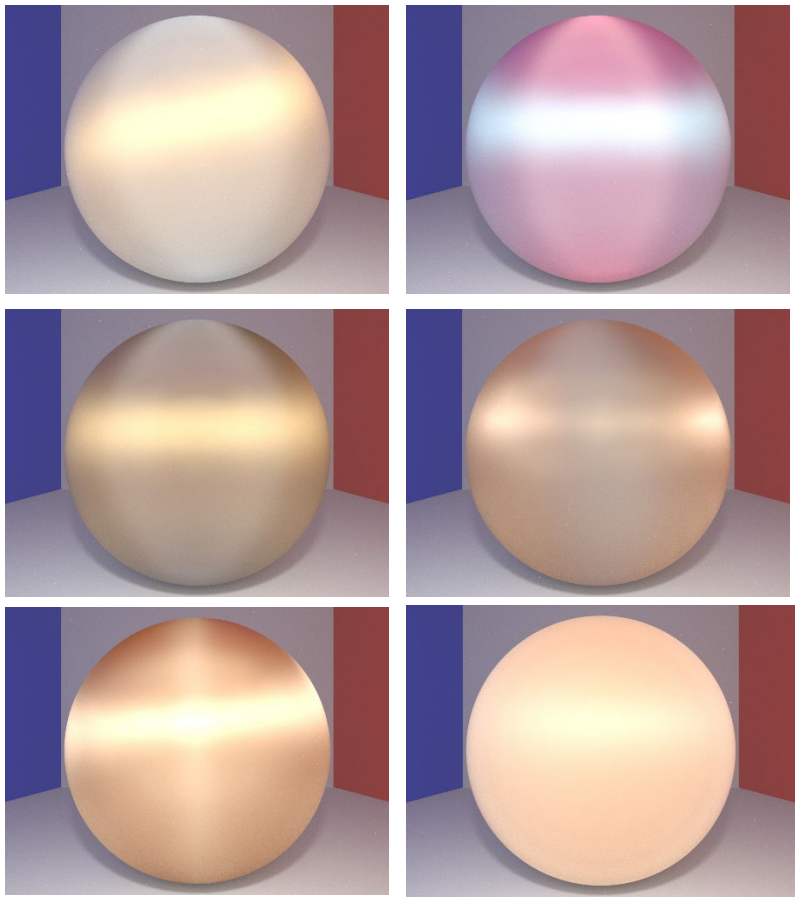
Database

Projection

Flickr results for “Autumn”

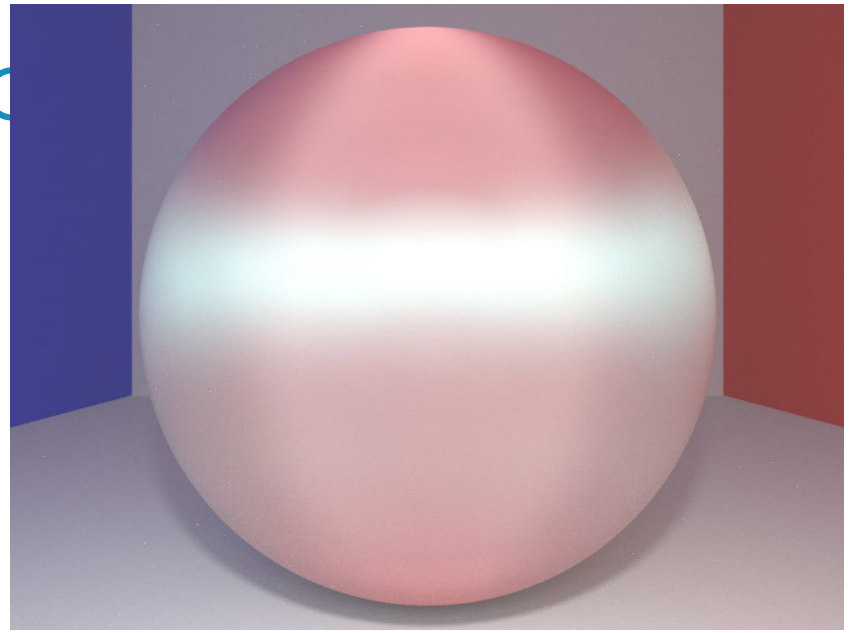
Projection

# Applic

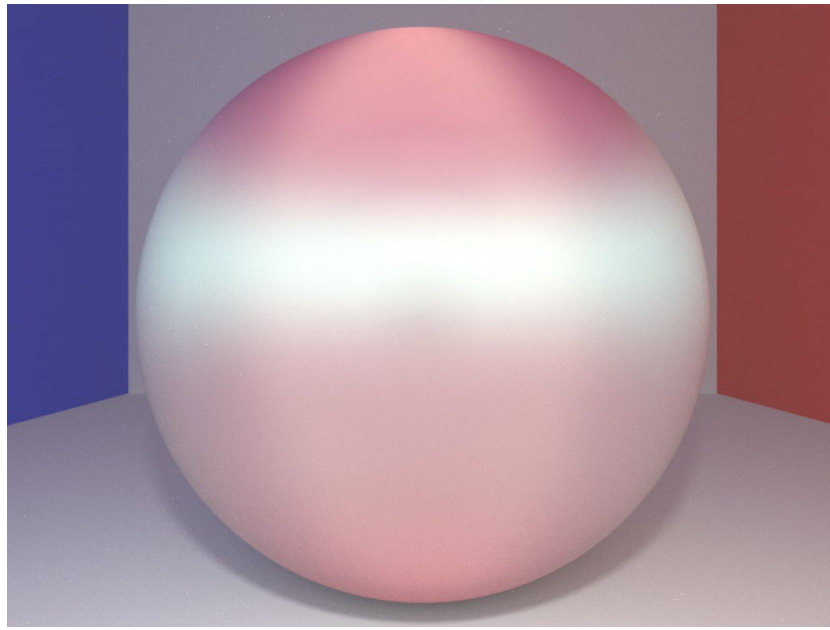


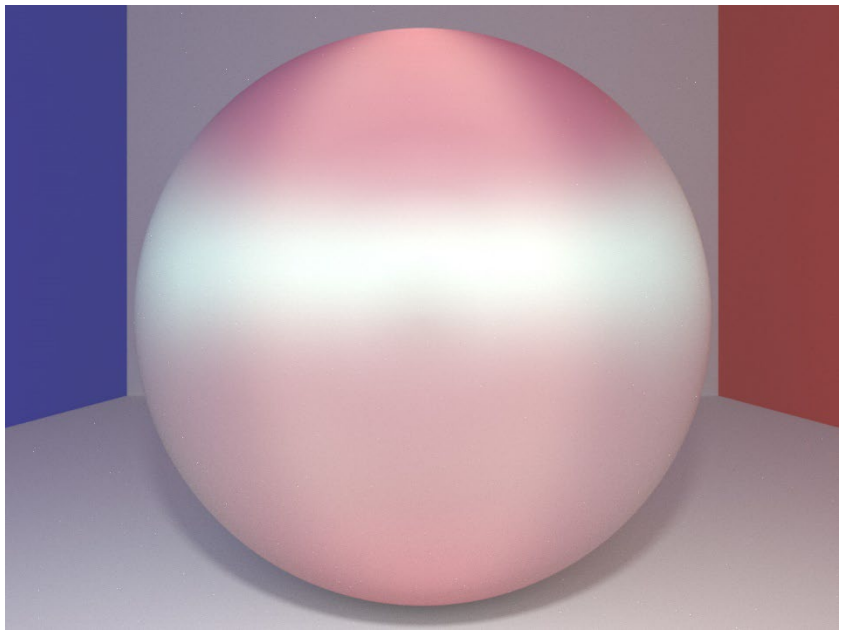
UTIA database

Reference

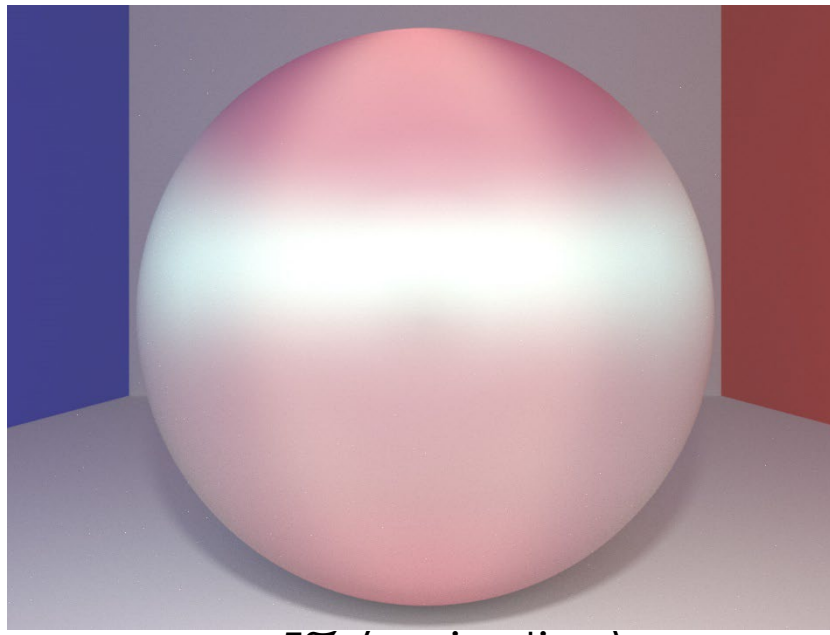


25% (projection)

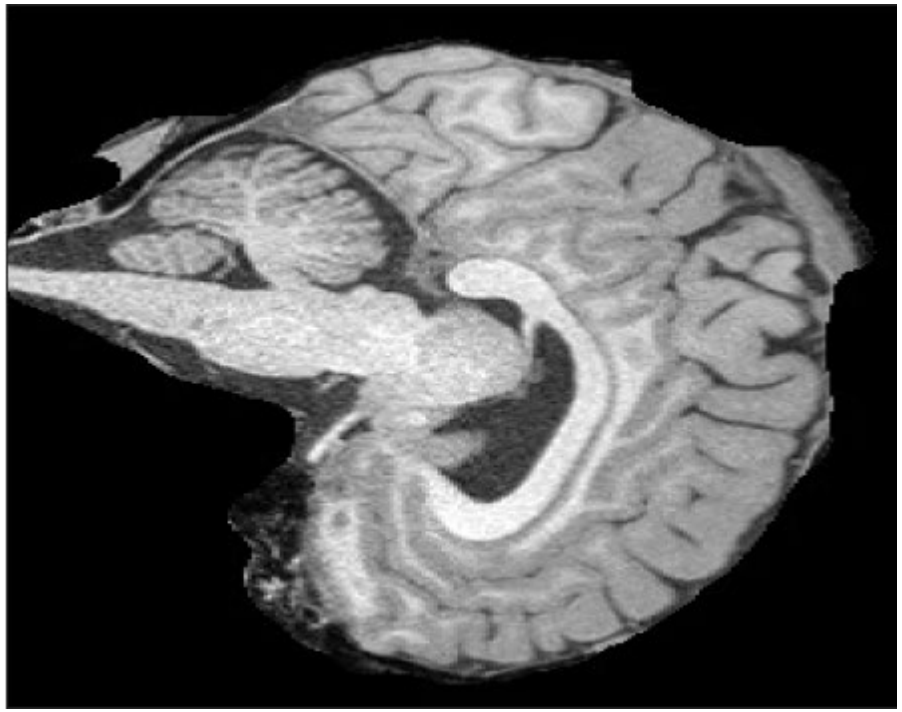




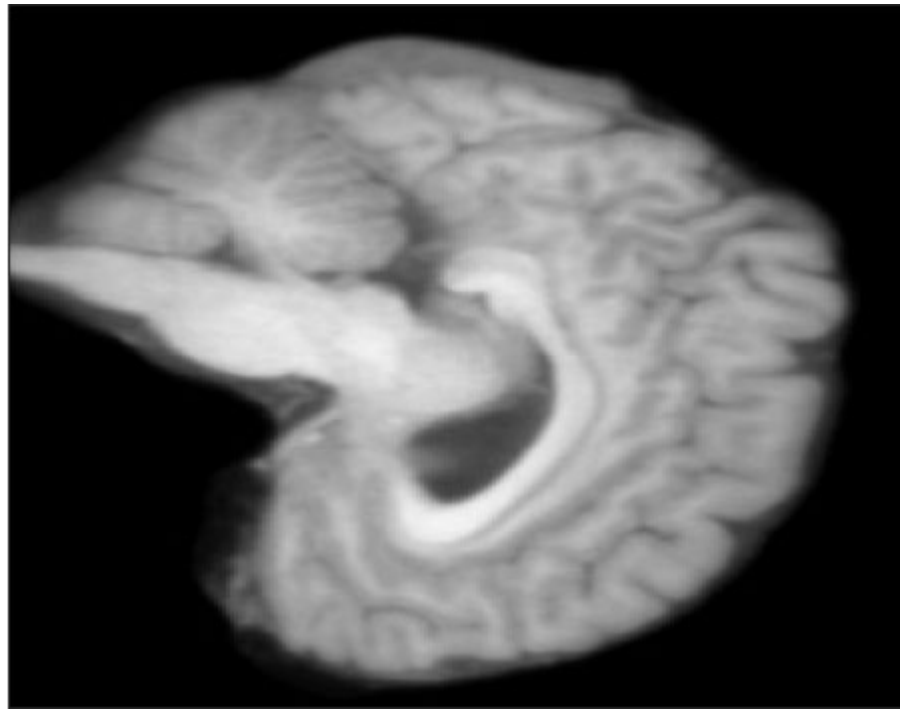
10% (projection)



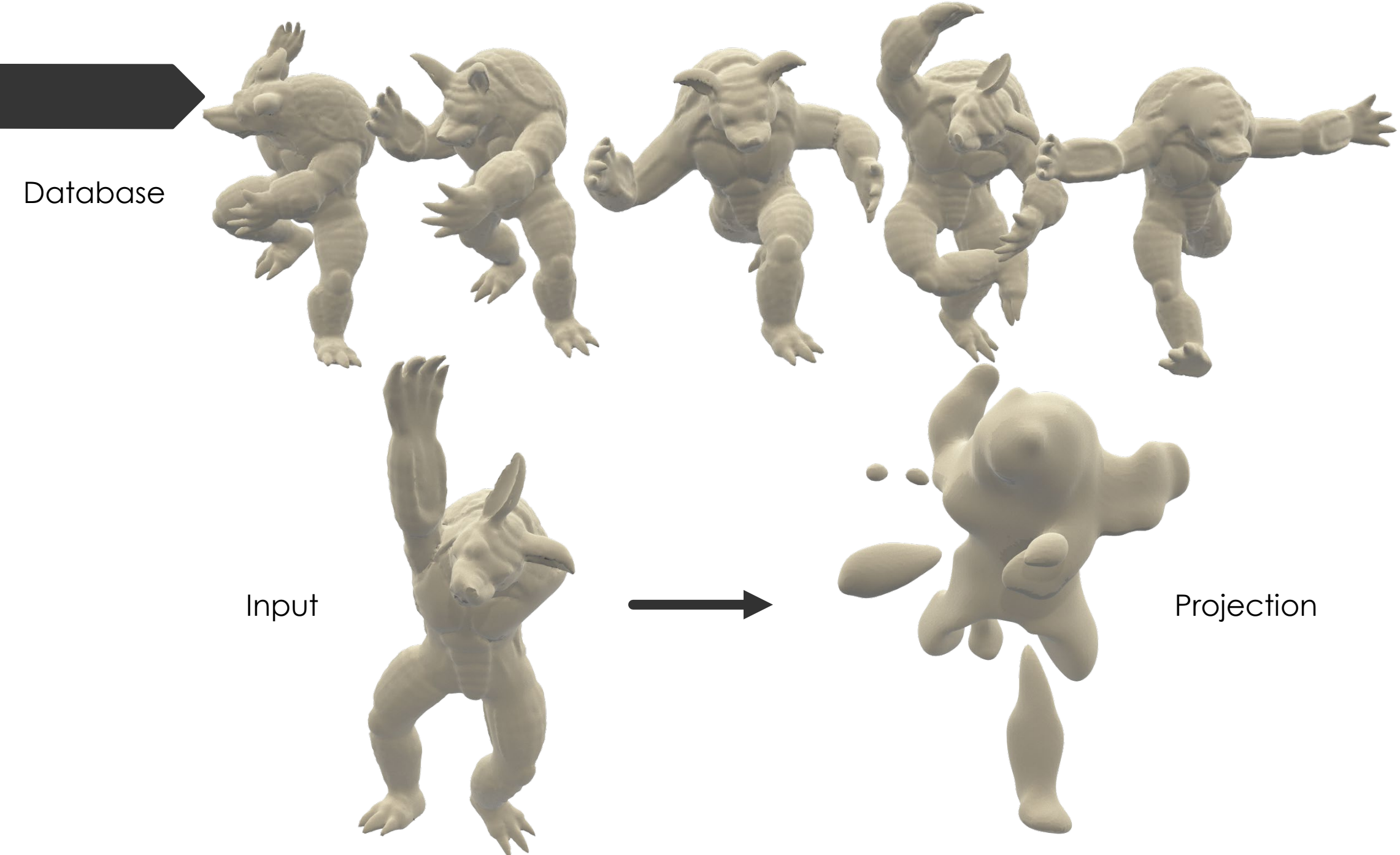
5% (projection)



Input



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/