Friedrich-Alexander-Universität
Naturwissenschaftliche Fakultät

Variational Graph Methods for Efficient Point Cloud Sparsification

Curves and Surfaces 2022

Martin Burger, Fjedor Gaede, Daniel Tenbrinck

Department Mathematik,
Friedrich-Alexander Universität Erlangen-Nürnberg

June 20th, 2022

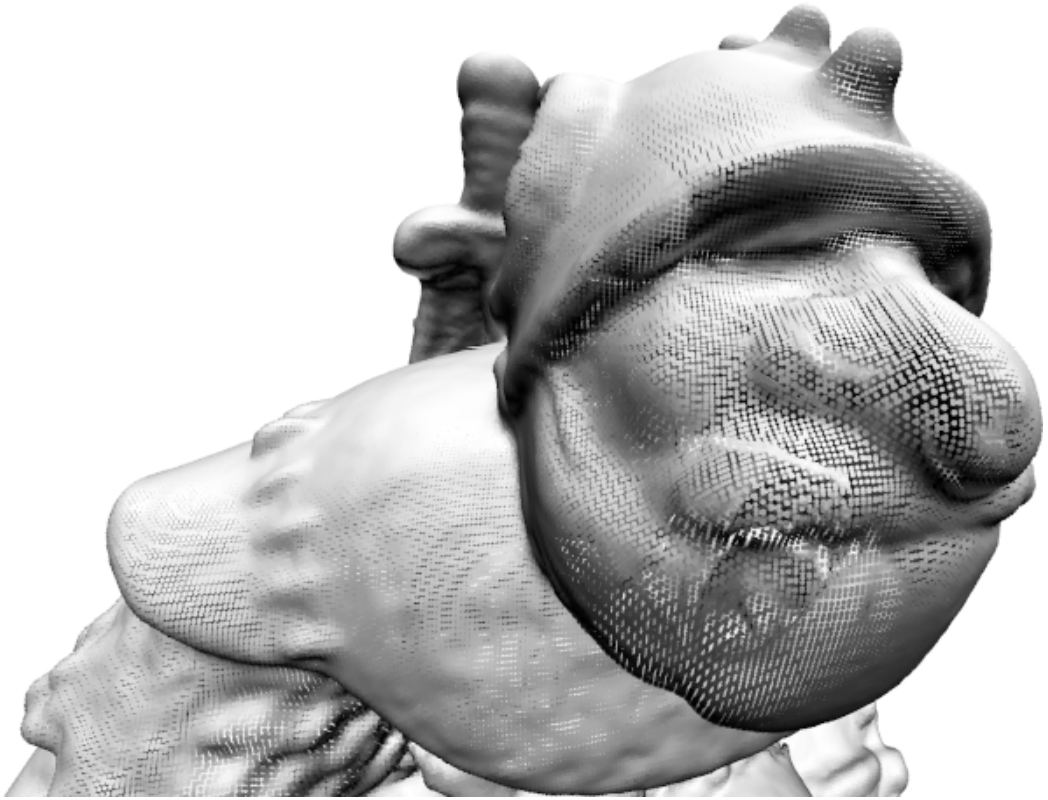
Outline

Motivation

High-resolution point cloud data

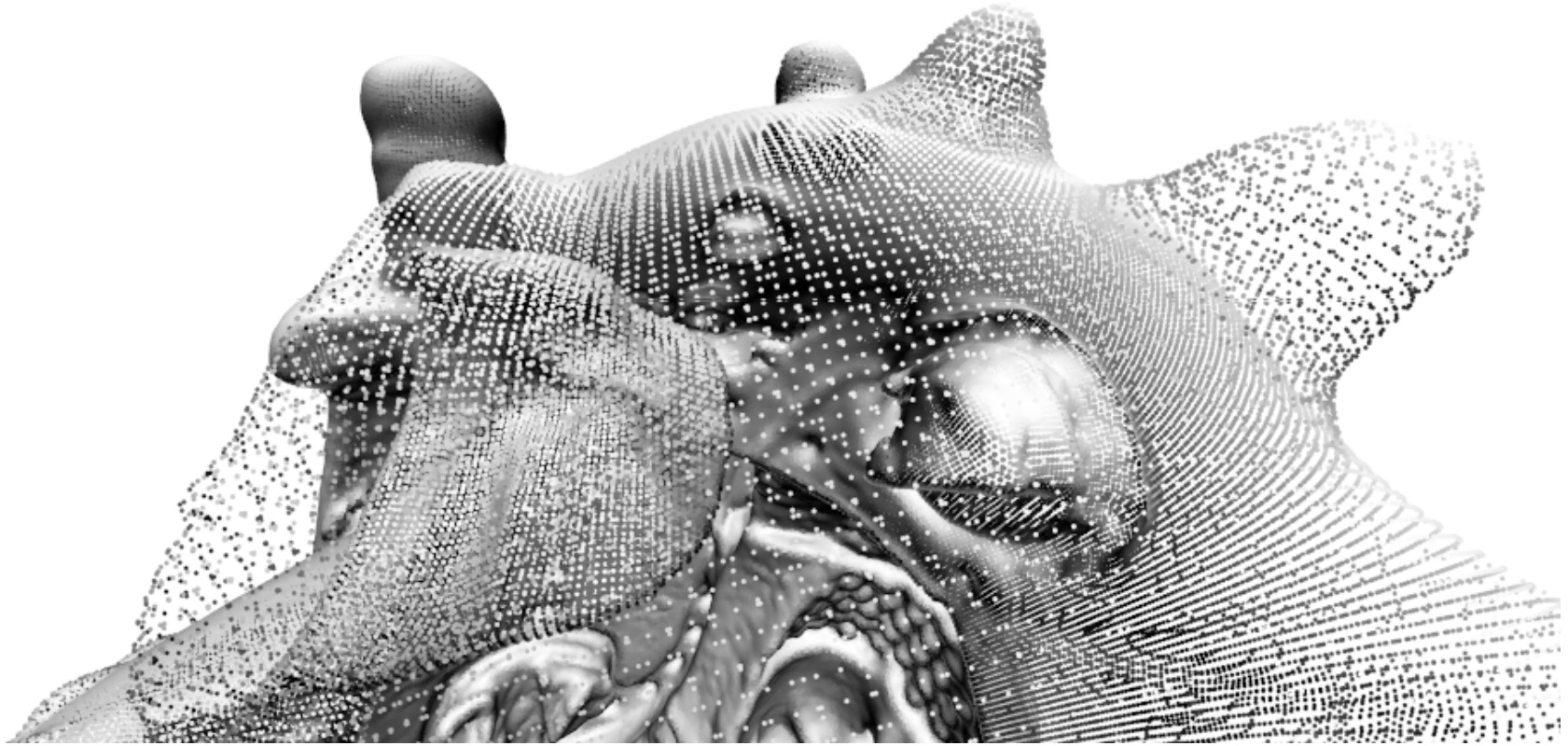
Motivation

High-resolution point cloud data



Motivation

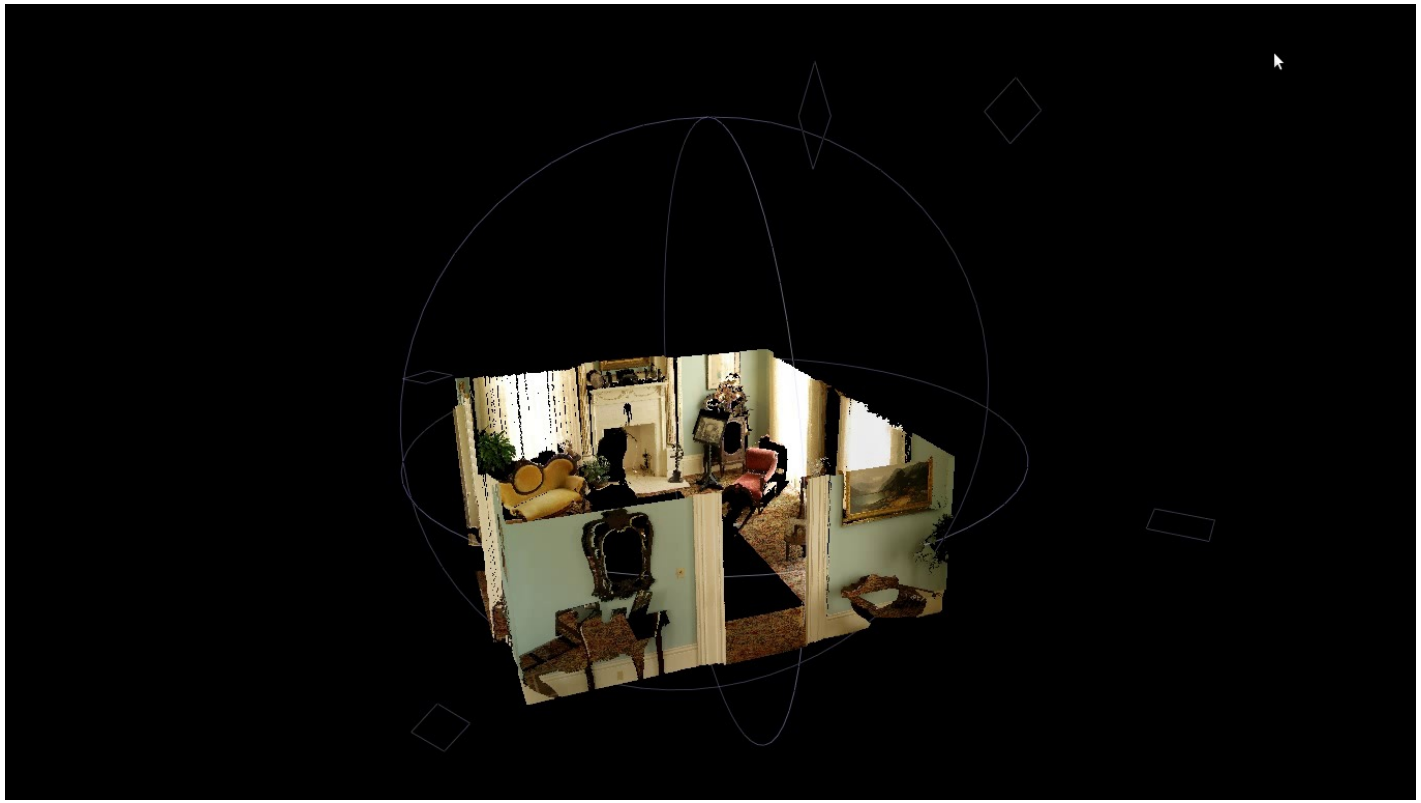
High-resolution point cloud data



Observation:
Many 3D point clouds contain redundant points, especially in regions with low curvature.

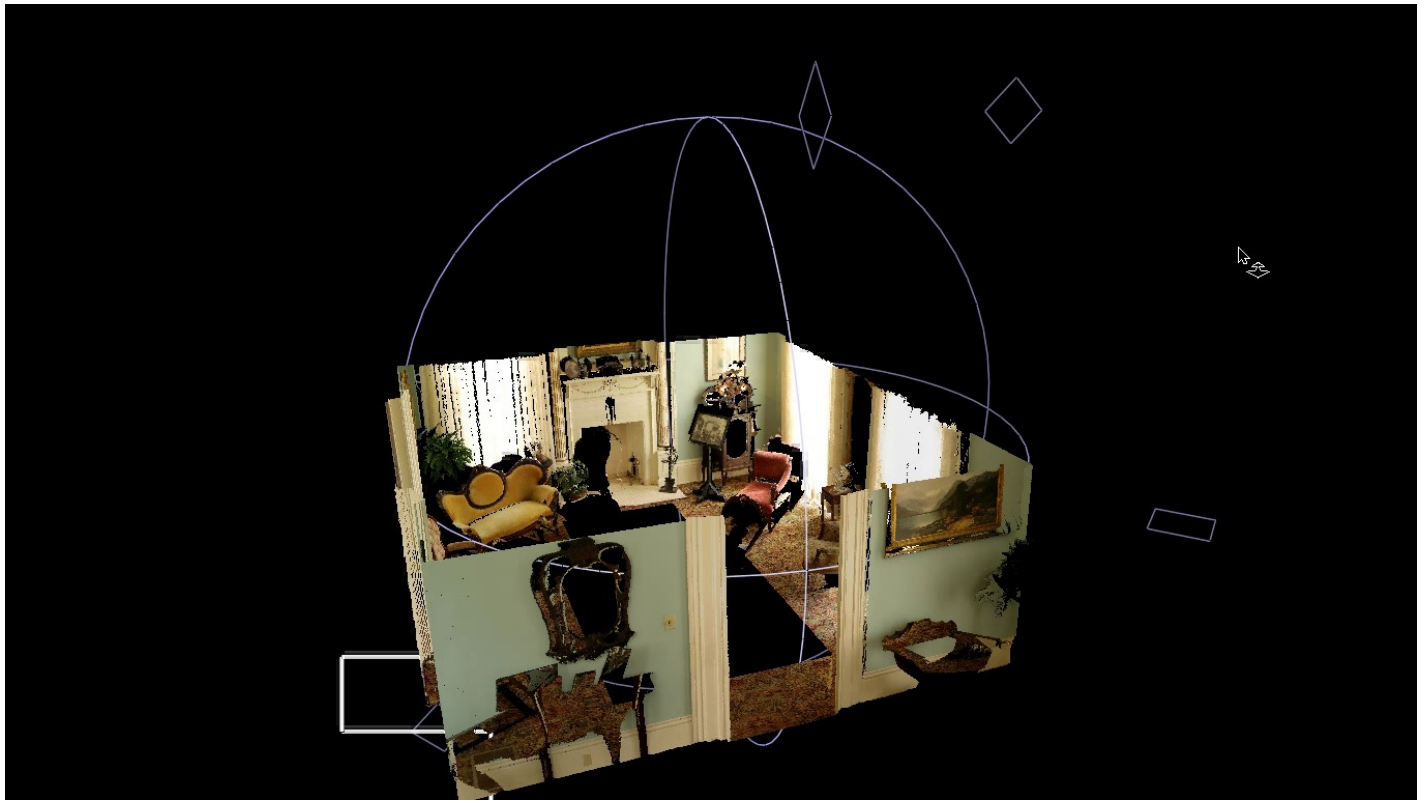
Motivation

High-resolution point cloud data



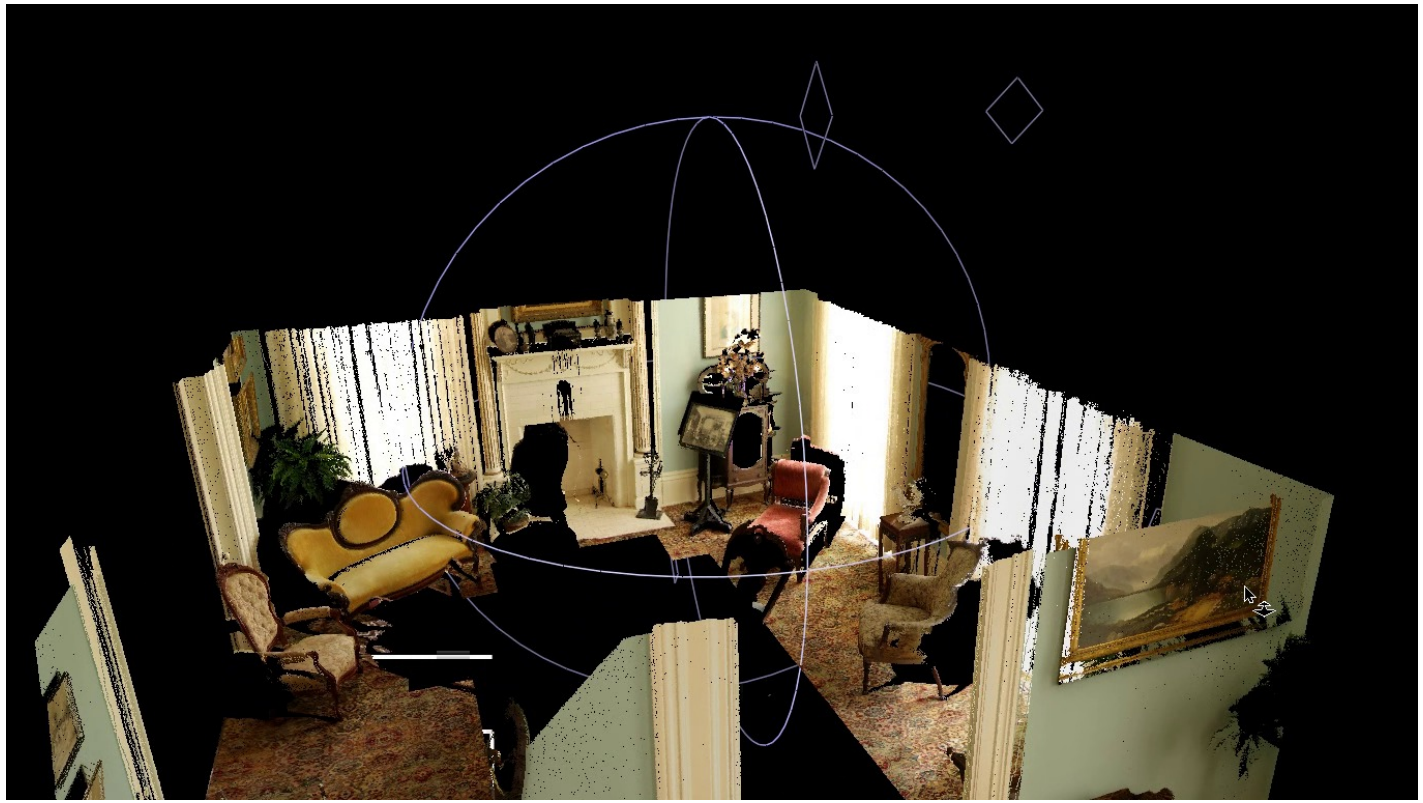
Motivation

High-resolution point cloud data



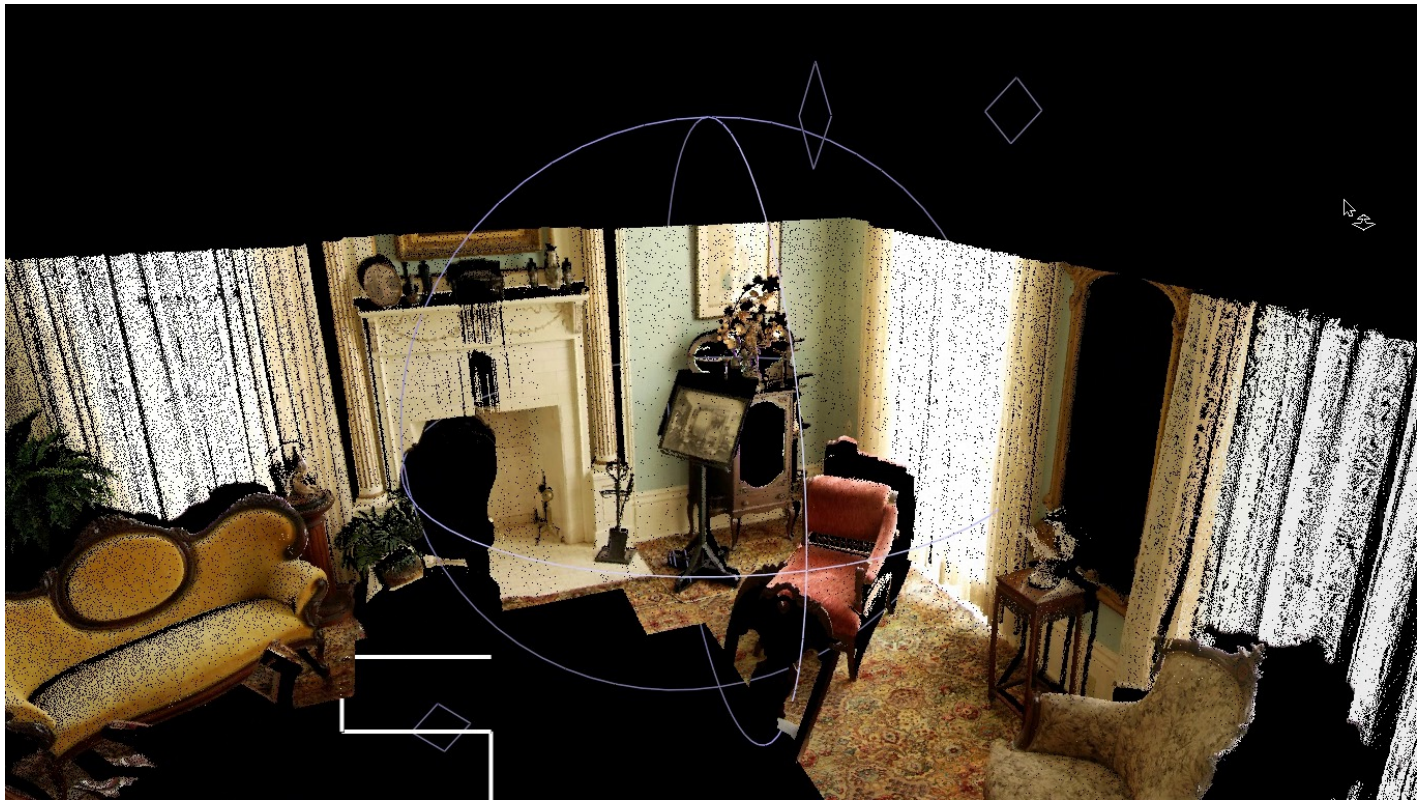
Motivation

High-resolution point cloud data



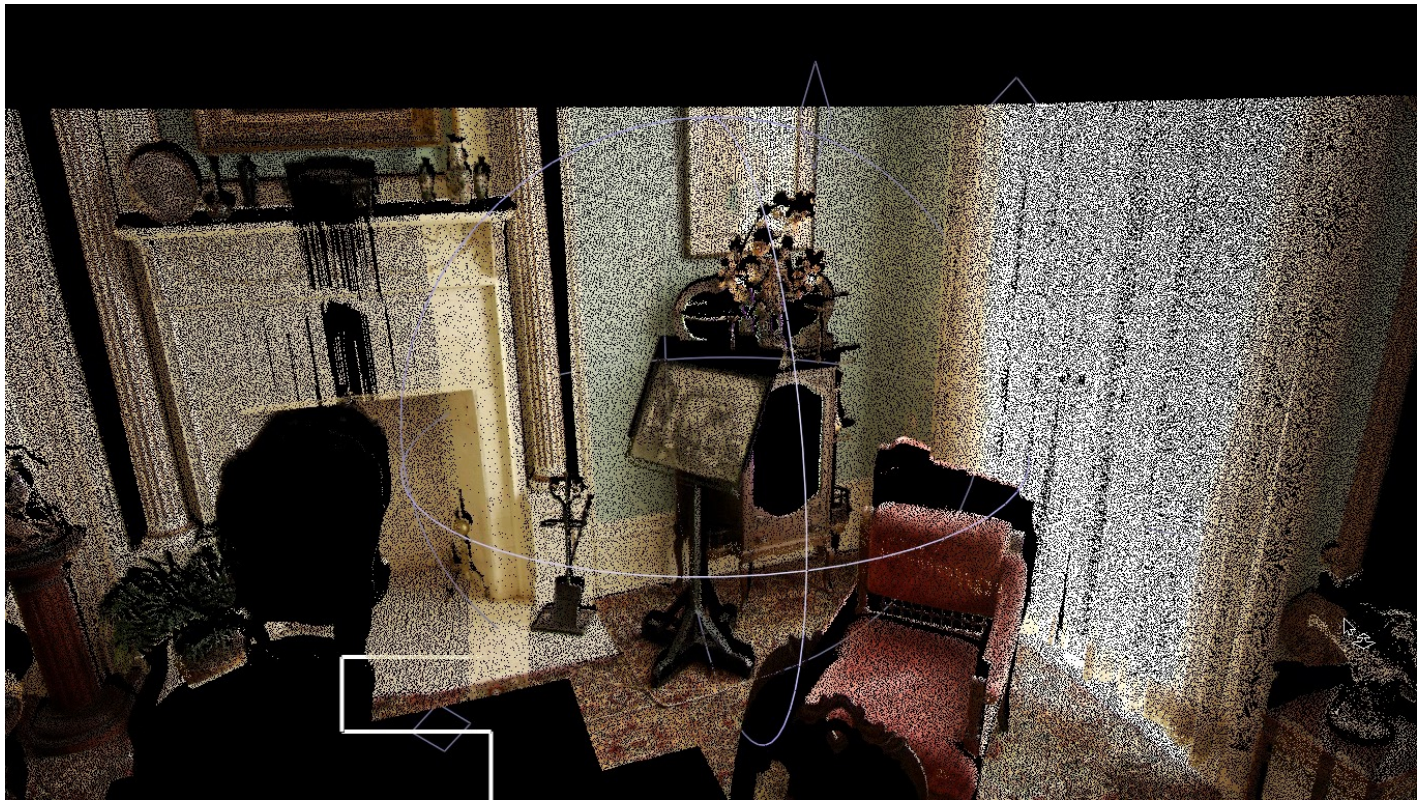
Motivation

High-resolution point cloud data



Motivation

High-resolution point cloud data

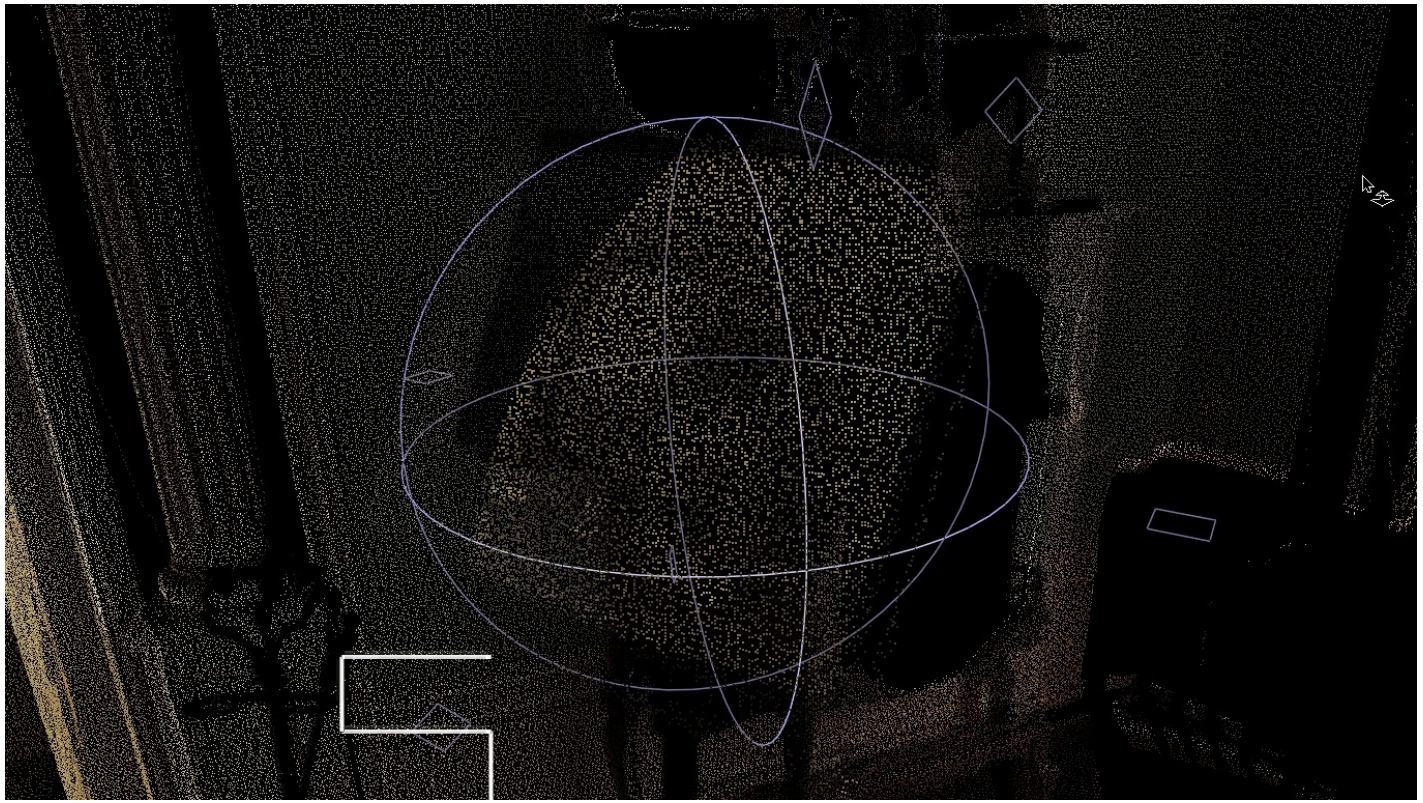


Motivation

High-resolution point cloud data

Motivation

High-resolution point cloud data



Observation:
Colored 3D point clouds of whole *rooms* or *buildings* can easily contain millions or even billions of points.

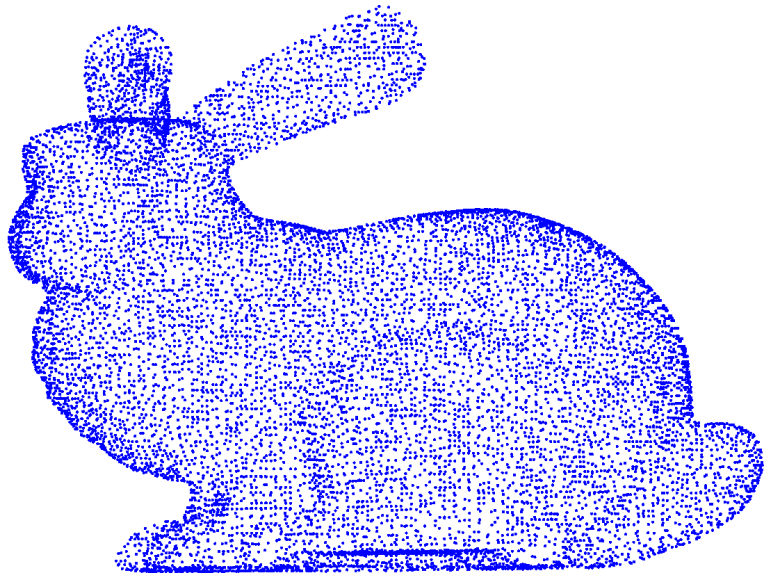
Motivation

Goal of point cloud sparsification

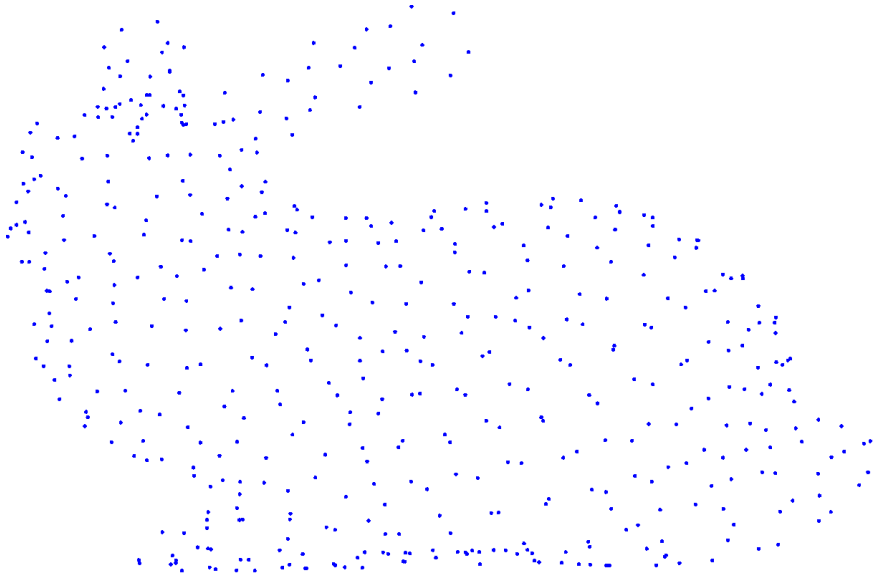
Problem: Possibly high computational costs in subsequent steps, e.g., *data analysis* and *machine learning*

Motivation

Goal of point cloud sparsification



3D point cloud with high resolution ($12,105$ points)



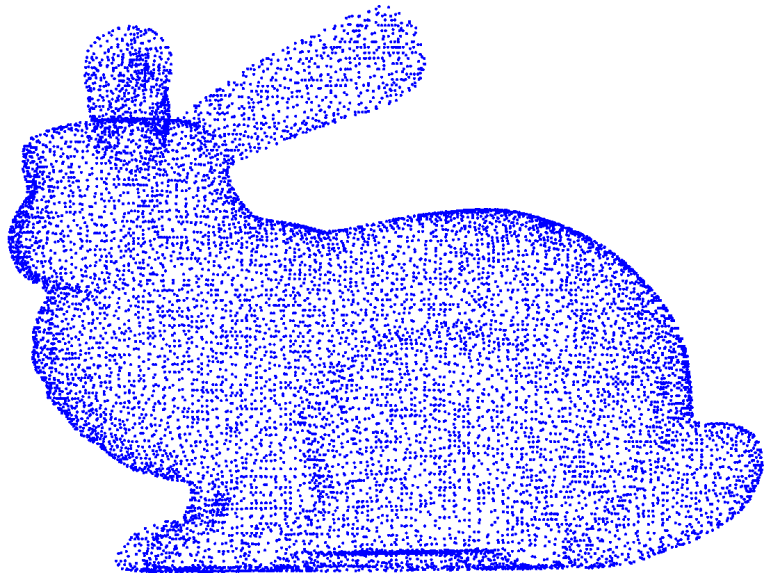
3D point cloud with low resolution (498 points)

Problem: Possibly high computational costs in subsequent steps, e.g., *data analysis* and *machine learning*

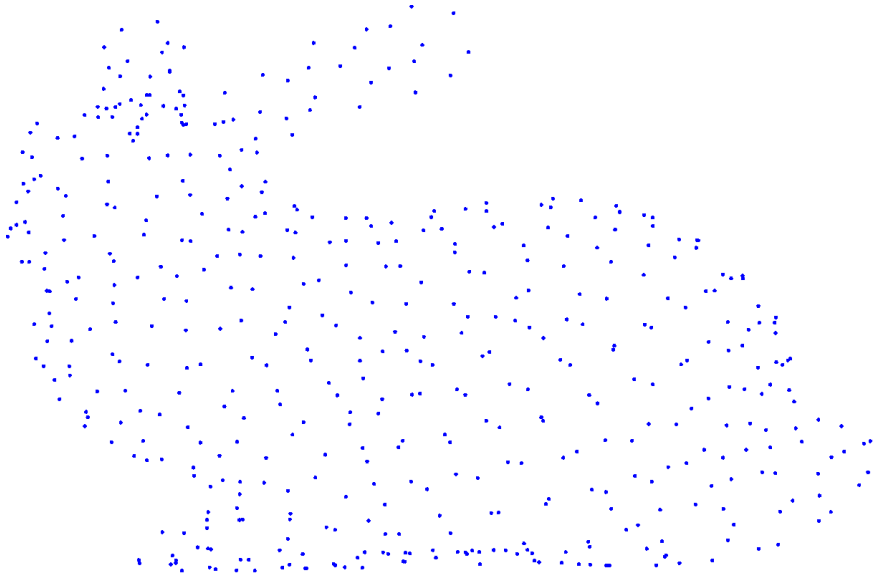
Goal: Capture geometry of 3D surfaces as good as possible, while *minimizing* the amount of *3D points needed*.

Motivation

Goal of point cloud sparsification



3D point cloud with high resolution ($12,105$ points)



3D point cloud with low resolution (498 points)

Problem: Possibly high computational costs in subsequent steps, e.g., *data analysis* and *machine learning*

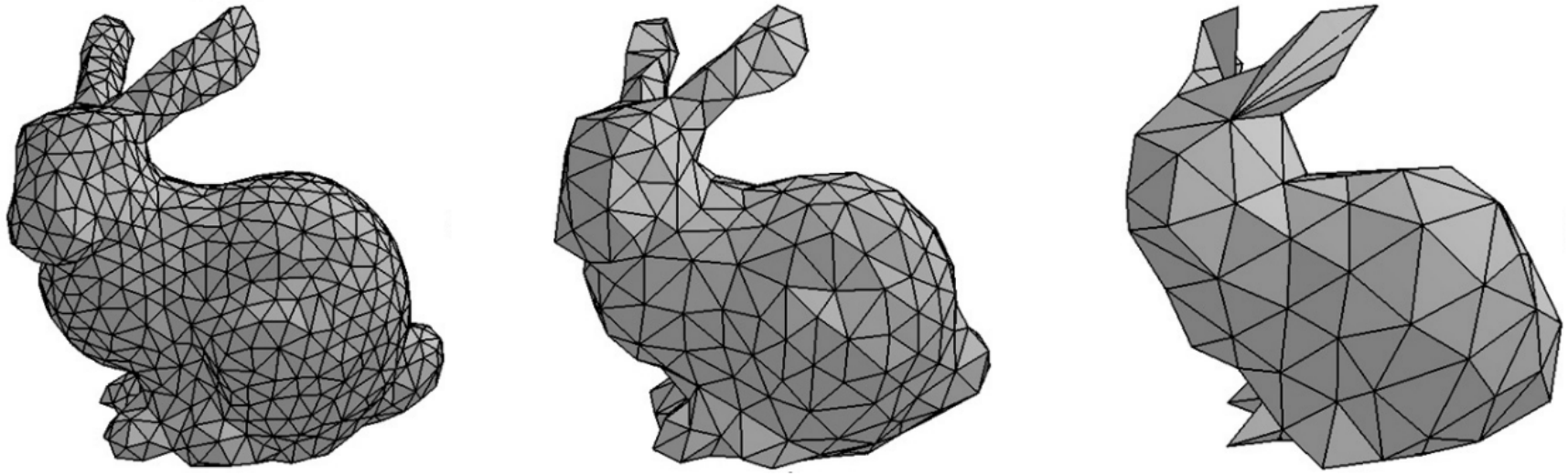
Goal: Capture geometry of 3D surfaces as good as possible, while *minimizing* the amount of *3D points needed*.

Question: How can we model and sparsify 3D surfaces consistently?

Finite weighted graphs

Graph-based data modeling

Question: How can we use graphs to describe polygon mesh surfaces?

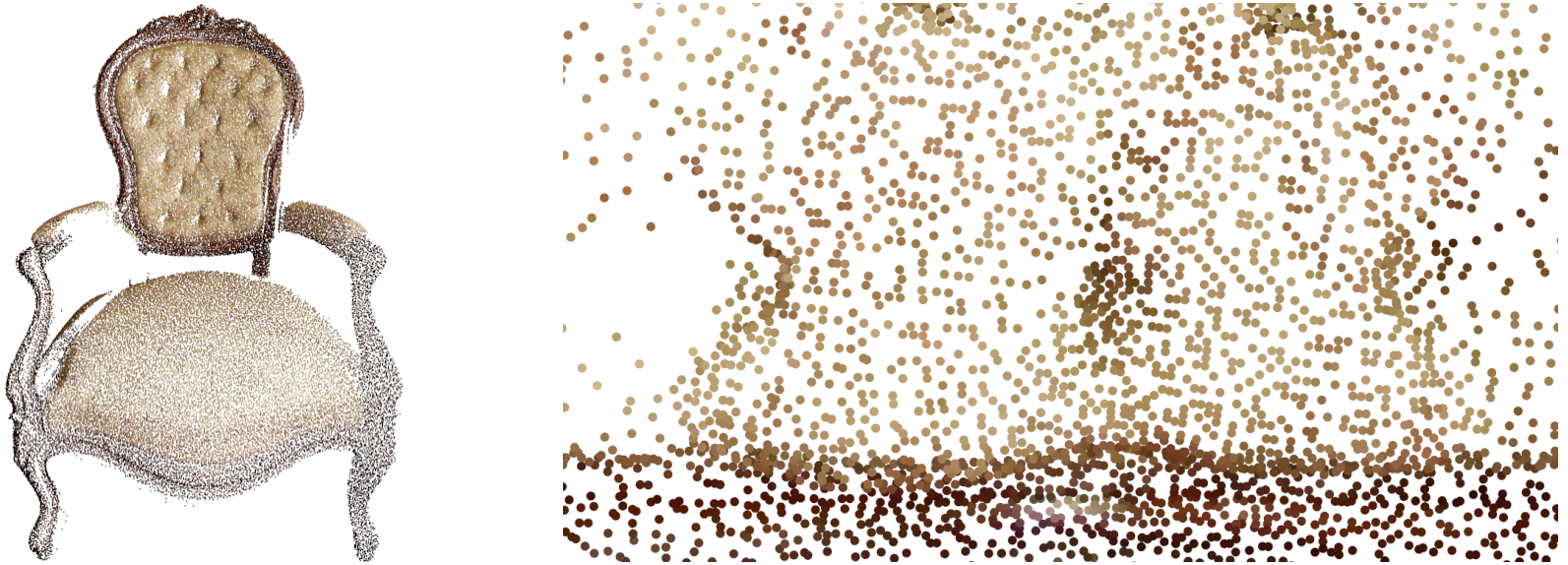


Polygon mesh representation of a 3D surface of a bunny model. Image courtesy: Gabriel Peyré

Finite weighted graphs

Graph-based data modeling

Question: How can we use graphs to describe 3D point clouds?

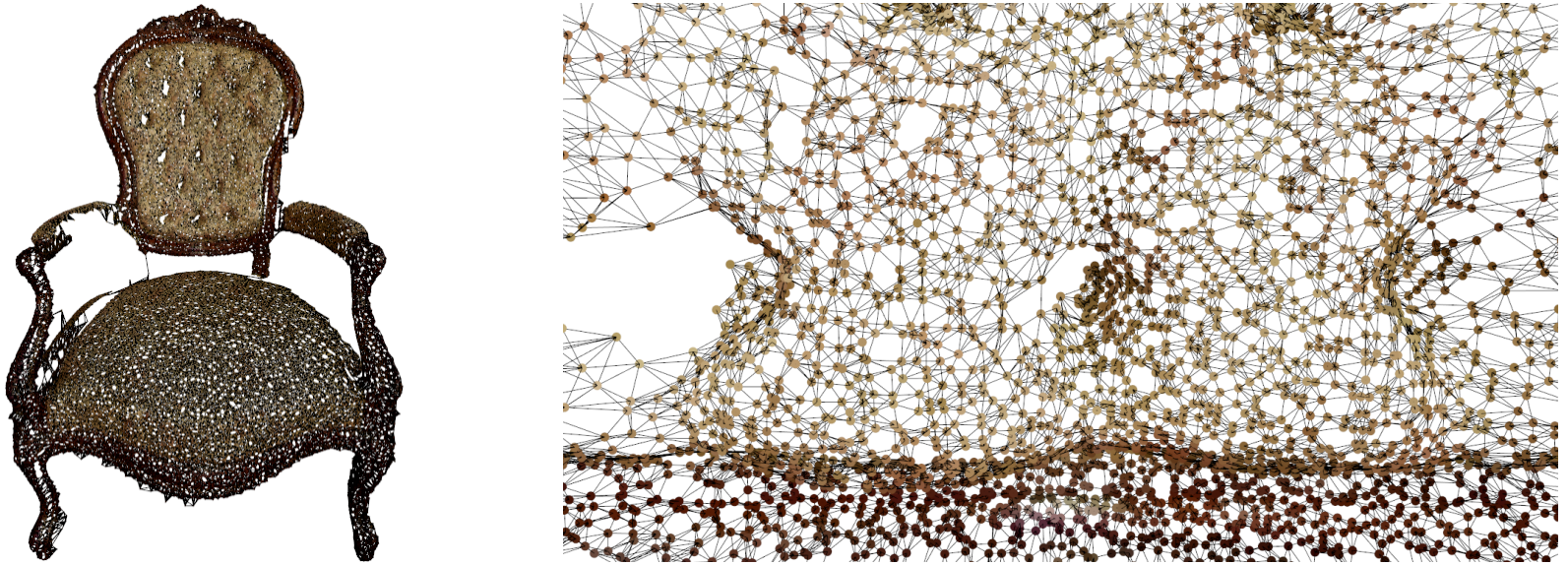


Colored 3D point cloud data of a scanned chair.

Finite weighted graphs

Graph-based data modeling

Question: How can we use graphs to describe 3D point clouds?



k-nearest neighbor graph construction on the colored 3D point cloud.

Finite weighted graphs

Basic notation

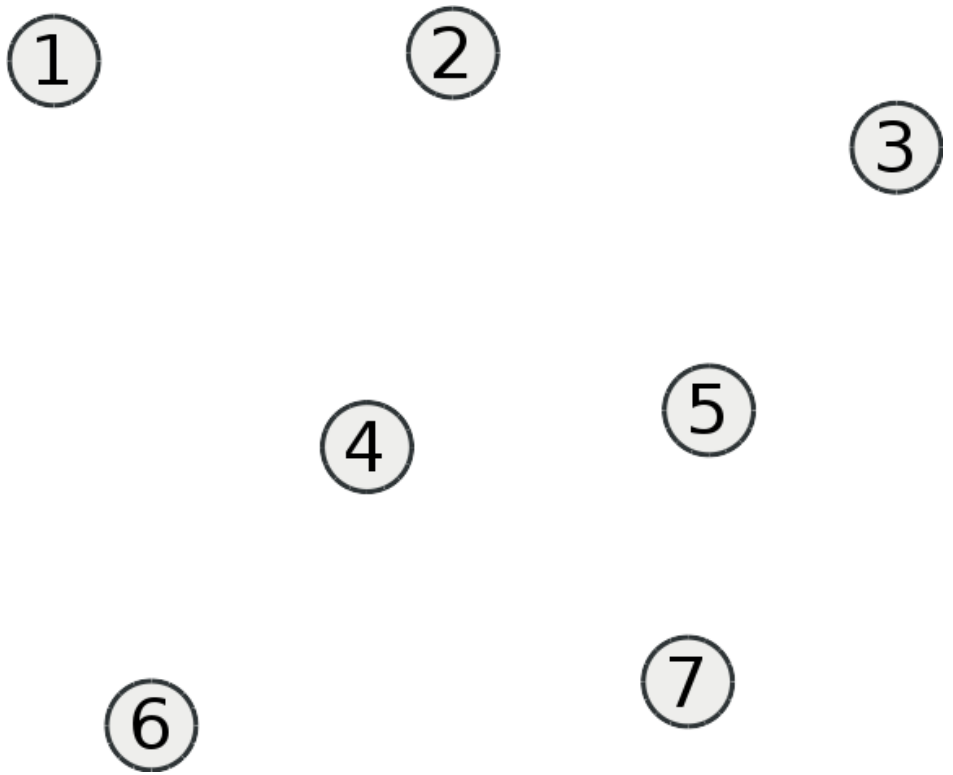
A finite weighted graph $G = (V, E, w)$ consists of:

Finite weighted graphs

Basic notation

A finite weighted graph $G = (V, E, w)$ consists of:

- a finite set of *vertices* $V = (v_1, \ldots, v_n)$

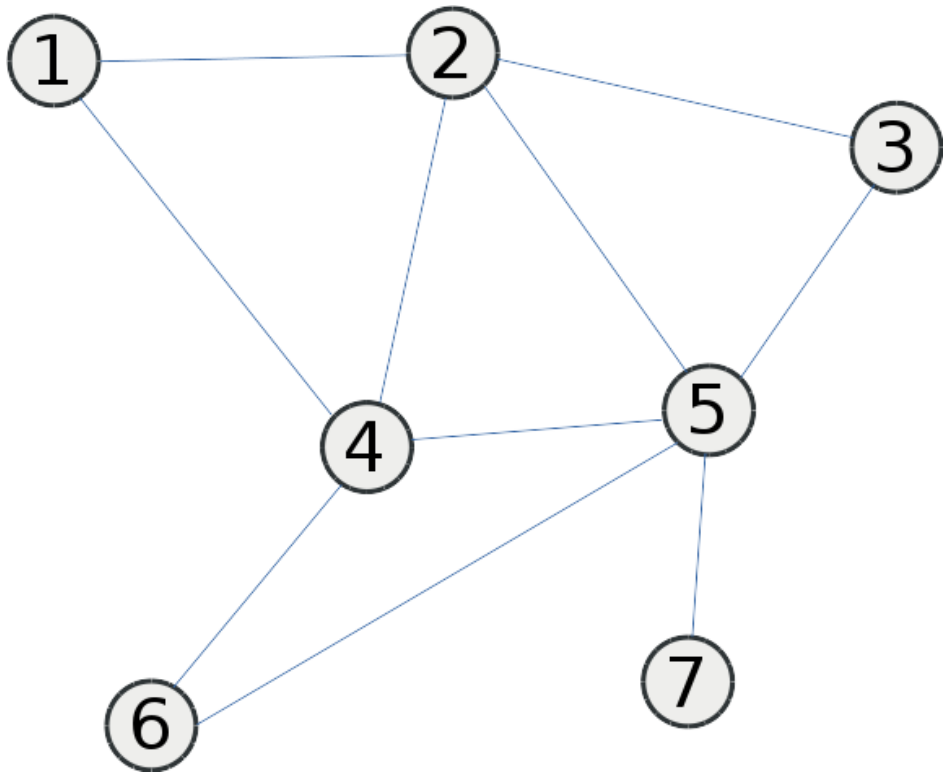


Finite weighted graphs

Basic notation

A finite weighted graph $G = (V, E, w)$ consists of:

- a finite set of *vertices* $V = (v_1, \ldots, v_n)$
- a finite set of *edges* $E \subset V \times V, \quad (v_i, v_j) \in E \rightarrow$ short: $v_i \sim v_j$

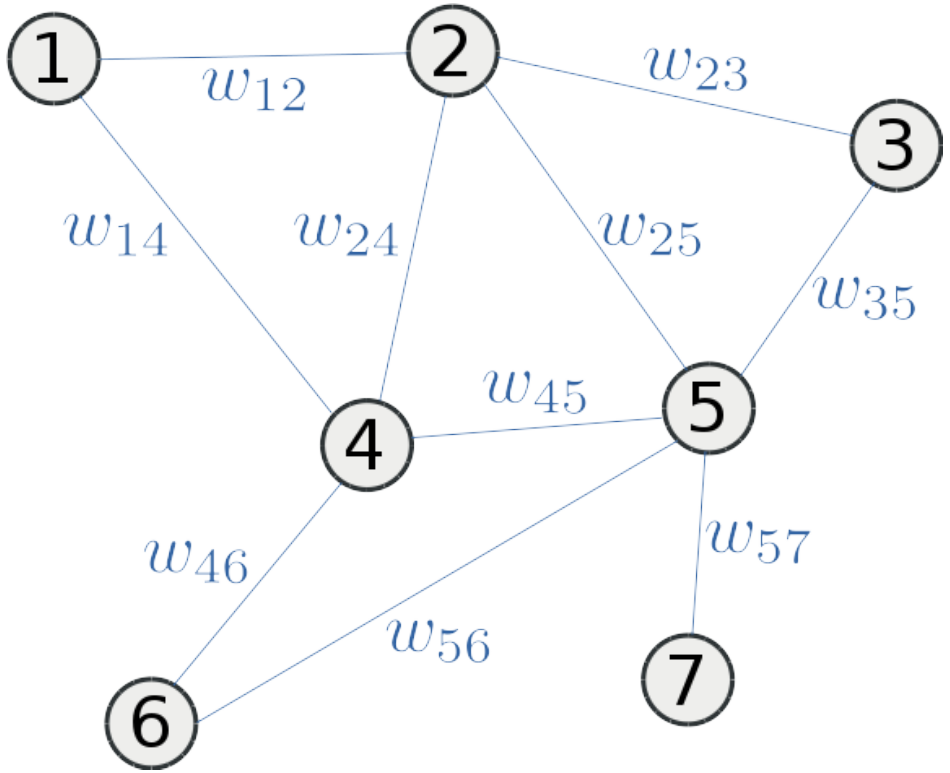


Finite weighted graphs

Basic notation

A finite weighted graph $G = (V, E, w)$ consists of:

- a finite set of *vertices* $V = (v_1, \ldots, v_n)$
- a finite set of *edges* $E \subset V \times V, \quad (v_i, v_j) \in E \rightarrow$ short: $v_i \sim v_j$
- a *weight function* $w \colon E \rightarrow [0, 1]$ with: $w(v_i, v_j) > 0 \Leftrightarrow (v_i, v_j) \in E$



Graph-based data modeling

Translating mathematical problems to graphs

Let (V, E, w) be a weighted graph and let $f \in H(V)$ with $f : V \to \mathbb{R}^m$ be a *vertex function* and $G \in H(E)$ an *edge function*. Then we can introduce the following first-order graph differential operators:

Weighted finite difference: $\nabla_w : H(V) \to H(E)$

$$\nabla_w f(v_i, v_j) = \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i)) \tag{1}$$

Graph-based data modeling

Translating mathematical problems to graphs

Let (V, E, w) be a weighted graph and let $f \in H(V)$ with $f : V \to \mathbb{R}^m$ be a *vertex function* and $G \in H(E)$ an *edge function*. Then we can introduce the following first-order graph differential operators:

Weighted finite difference: $\nabla_w : H(V) \to H(E)$

$$\nabla_w f(v_i, v_j) \;=\; \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i)) \tag{1}$$

Adjoint operator: $\nabla_w^* : H(E) \to H(V)$ of

$$\langle \nabla_w f, G \rangle_{H(E)} \;=\; \langle f, \nabla_w^* G \rangle_{H(V)} \tag{2}$$

Graph-based data modeling

Translating mathematical problems to graphs

Let (V, E, w) be a weighted graph and let $f \in H(V)$ with $f : V \to \mathbb{R}^m$ be a *vertex function* and $G \in H(E)$ an *edge function*. Then we can introduce the following first-order graph differential operators:

Weighted finite difference: $\nabla_w : H(V) \to H(E)$

$$\nabla_w f(v_i, v_j) \;=\; \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i)) \tag{1}$$

Adjoint operator: $\nabla_w^* : H(E) \to H(V)$ of

$$\langle \nabla_w f, G \rangle_{H(E)} \;=\; \langle f, \nabla_w^* G \rangle_{H(V)} \tag{2}$$

Divergence: $\operatorname{div}_w : H(E) \to H(V)$

$$\operatorname{div}_w G(u) \;=\; -\nabla_w^* G(u) \;=\; \sum_{v_j \sim v_i} \sqrt{w(v_i, v_j)}(G(v_i, v_j) - G(v_j, v_i)) \tag{3}$$

Finite weighted graphs

A variational denoising modelsfor multivariate data

For point cloud sparsification one can try to solve the following optimization problem on graphs:[1]

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = \frac{1}{2}\|u-f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;p}^p \right\}, \quad \alpha > 0,\ p \in (0,+\infty) \tag{4}$$

[1] F. Lozes, A. Elmoataz, O. Lézoray: Partial difference operators on weighted graphs for image processing on surfaces and point clouds.IEEE TIP 23(9), 2014

Finite weighted graphs

A variational denoising modelsfor multivariate data

For point cloud sparsification one can try to solve the following optimization problem on graphs:[1]

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = \frac{1}{2}\|u-f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;p}^p \right\}, \quad \alpha > 0,\ p \in (0,+\infty) \tag{4}$$

Idea:
Solving the above problem one performs geometric denoising:

- Compute a smooth approximation u of f
- Points concentrate around cluster points with *high curvature*
- Filter out all points in direct vicinity of cluster points

[1] F. Lozes, A. Elmoataz, O. Lézoray: Partial difference operators on weighted graphs for image processing on surfaces and point clouds.IEEE TIP 23(9), 2014

Finite weighted graphs

A variational denoising modelsfor multivariate data

For point cloud sparsification one can try to solve the following optimization problem on graphs:[1]

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = \frac{1}{2}\|u-f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;p}^p \right\}, \quad \alpha > 0,\ p \in (0,+\infty) \tag{4}$$

Idea:
Solving the above problem one performs geometric denoising:

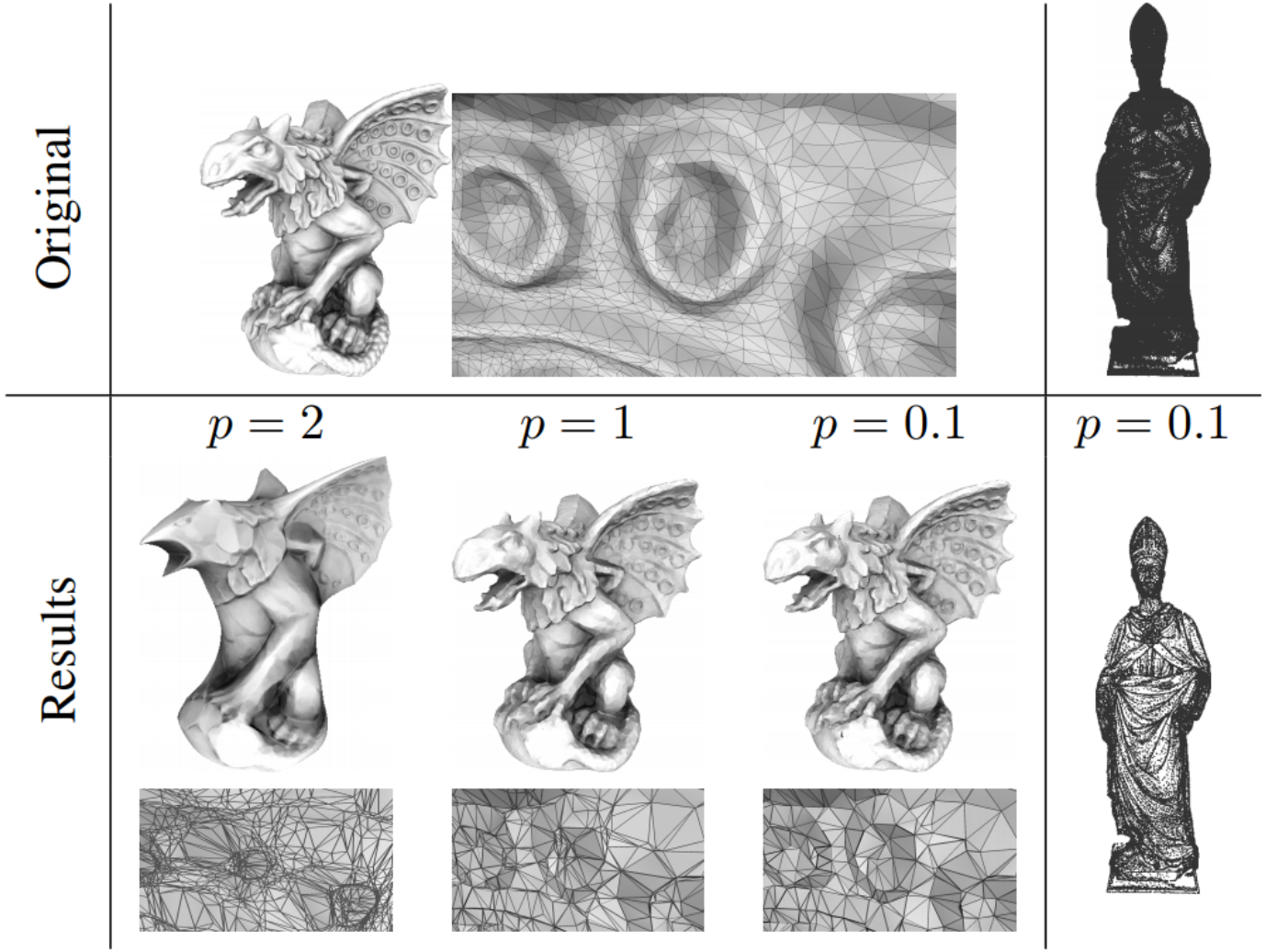
- Compute a smooth approximation u of f
- Points concentrate around cluster points with *high curvature*
- Filter out all points in direct vicinity of cluster points

$\rightarrow$ corresponds to a fine-to-coarse strategy!

[1] F. Lozes, A. Elmoataz, O. Lézoray: Partial difference operators on weighted graphs for image processing on surfaces and point clouds.IEEE TIP 23(9), 2014

Finite weighted graphs

Point cloud sparsication via geometric denoising



Finite weighted graphs

Model reduction on graphs

Computation of graph gradient and divergence operators is numerically expensive on huge sets of vertices and many neighbors per vertex.

Finite weighted graphs

Model reduction on graphs

Computation of graph gradient and divergence operators is numerically expensive on huge sets of vertices and many neighbors per vertex.

Observation:

- Homogeneous data subsets don't yield much information.
- Most information is obtained on subset boundaries.

Finite weighted graphs

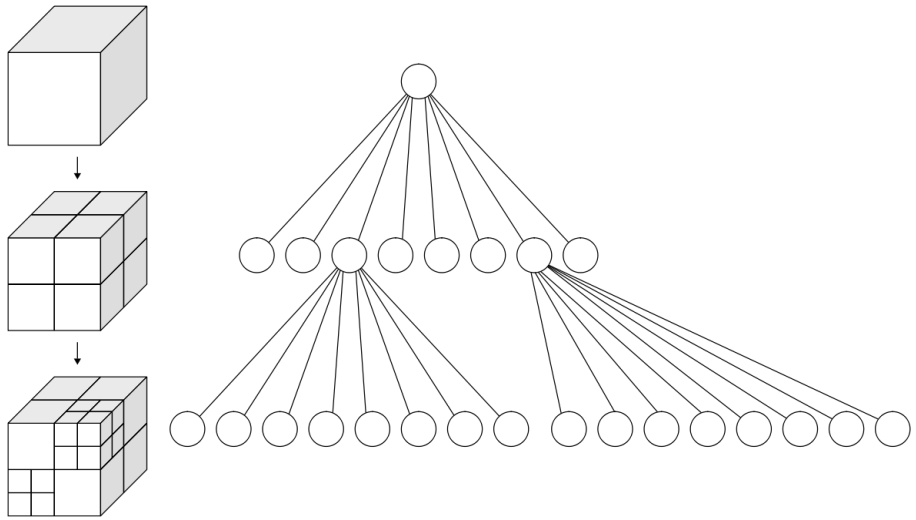
Model reduction on graphs

Computation of graph gradient and divergence operators is numerically expensive on huge sets of vertices and many neighbors per vertex.

Observation:

- Homogeneous data subsets don't yield much information.
- Most information is obtained on subset boundaries.

Idea: Classical approaches use multiscale strategies to model the data from coarse to fine, e.g., Octree data representation.



Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2][3]:

$$
\begin{aligned}
u_{\Pi} &= \underset{u \in \mathcal{H}(\Pi)}{\arg\min}\, D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017

[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2][3]:

$$
\begin{aligned}
u_{\Pi} &= \underset{u \in \mathcal{H}(\Pi)}{\arg\min}\, D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

Intuitively:

1. Initialize a partition Π

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017

[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2][3]:

$$
\begin{aligned}
u_{\Pi} &= \operatorname*{arg\,min}_{u \in \mathcal{H}(\Pi)} D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

Intuitively:

1. Initialize a partition Π
2. Repeat until convergence:
   - Compute reduced data f_{Π} based on Π

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017
[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2][3]:

$$
\begin{aligned}
u_{\Pi} &= \underset{u \in \mathcal{H}(\Pi)}{\arg\min}\, D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

Intuitively:

1. Initialize a partition Π
2. Repeat until convergence:
   - Compute reduced data f_{Π} based on Π
   - Solve energy functional for reduced problem

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017

[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2,3]:

$$
\begin{aligned}
u_{\Pi} &= \underset{u \in \mathcal{H}(\Pi)}{\arg\min}\, D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

Intuitively:

1. Initialize a partition Π
2. Repeat until convergence:
   - Compute reduced data f_{Π} based on Π
   - Solve energy functional for reduced problem
   - Expand solution u_{Π} as piecewise constant function $\overline{u}_{\Pi}$ to V

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017
[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2][3]:

$$
\begin{aligned}
u_{\Pi} &= \operatorname*{arg\,min}_{u \in \mathcal{H}(\Pi)} D(u, f_{\Pi}) + \alpha R(u) \\
\min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\}
\end{aligned}
\tag{5}
$$

Intuitively:

1. Initialize a partition Π
2. Repeat until convergence:
   - Compute reduced data f_{Π} based on Π
   - Solve energy functional for reduced problem
   - Expand solution u_{Π} as piecewise constant function $\overline{u}_{\Pi}$ to V
   - Compute new partition Π via graph cut based on distance $\overline{u}_{\Pi}$ to original data f

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017
[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

Cut Pursuit in a nutshell

The algorithm

The Cut Pursuit algorithm[2,3]:

$$\begin{aligned} u_{\Pi} &= \operatorname*{arg\,min}_{u \in \mathcal{H}(\Pi)} D(u, f_{\Pi}) + \alpha R(u) \\ \min_{B \subset V} \{J'(\overline{u}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_{\Pi}, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_{\Pi}), \mathbb{1}_B \rangle + \alpha R'_{S^c}(\overline{u}_{\Pi}; \mathbb{1}_B)\} \end{aligned} \tag{5}$$

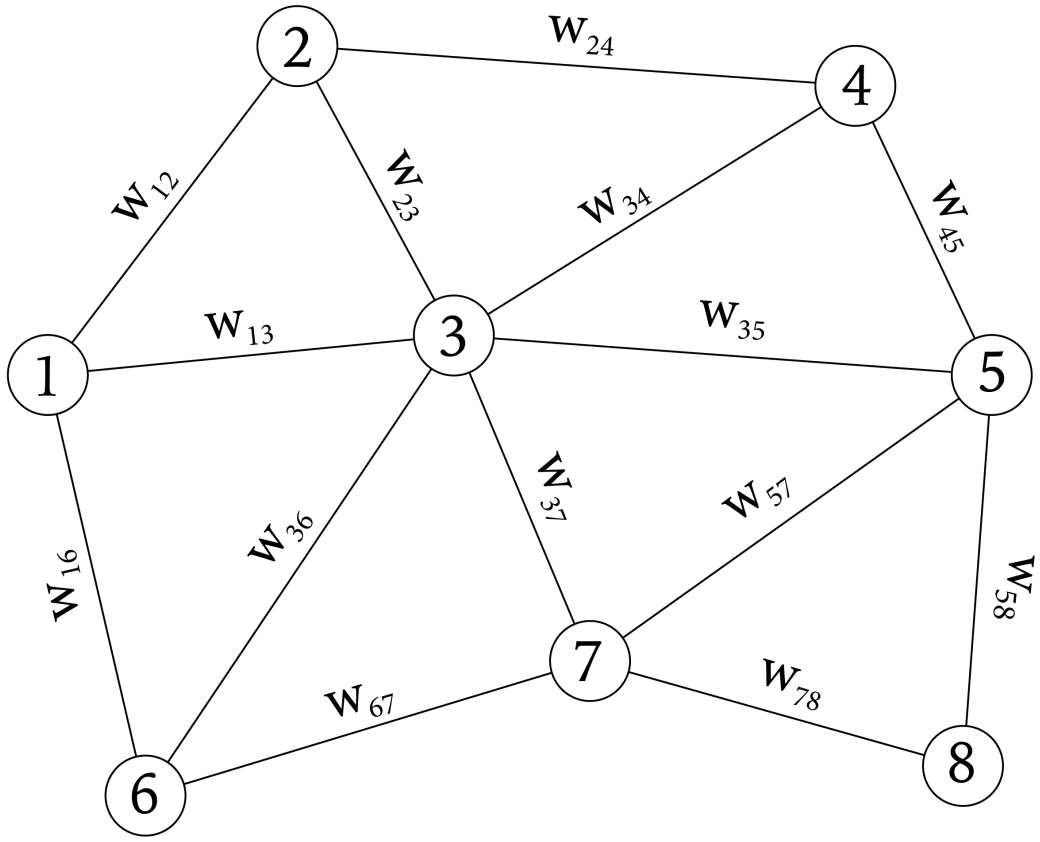
Convergence result:
In the case of total variation (TV) denoising it can be shown that:

$$\overline{u}_{\Pi} \in \operatorname*{arg\,min}_{u \in \mathcal{H}(V)} J(u) \quad \Leftrightarrow \quad \min_{B \subset V} J'(\overline{u}_{\Pi}; \mathbb{1}_B) = 0. \tag{6}$$

[2] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017
[3] H. Raguet, L. Landrieu: Cut-pursuit algorithm for regularizing nonsmooth functionals with graph total variation. International Conference on Machine Learning, 2018

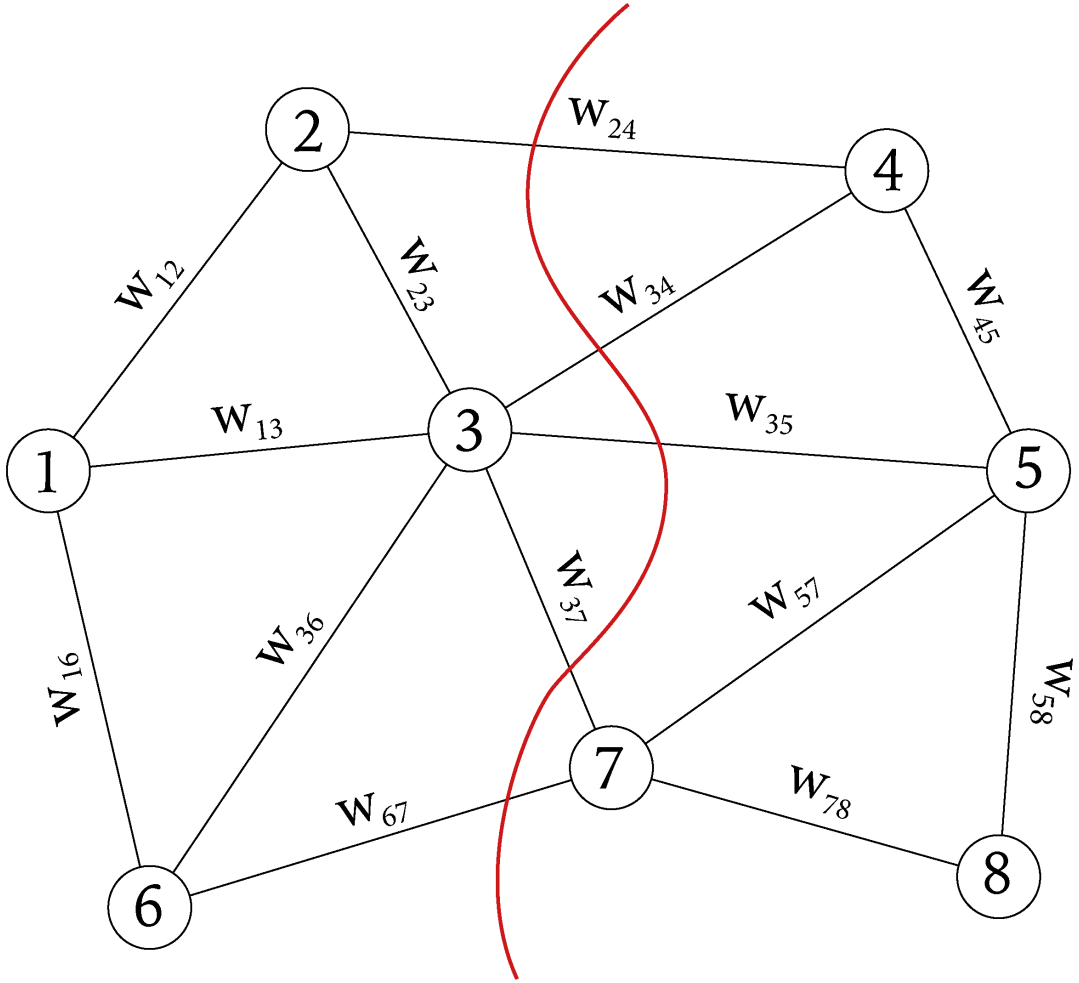
Cut Pursuit in a nutshell

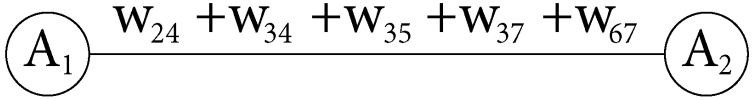
Illustration on a toy example



Cut Pursuit in a nutshell

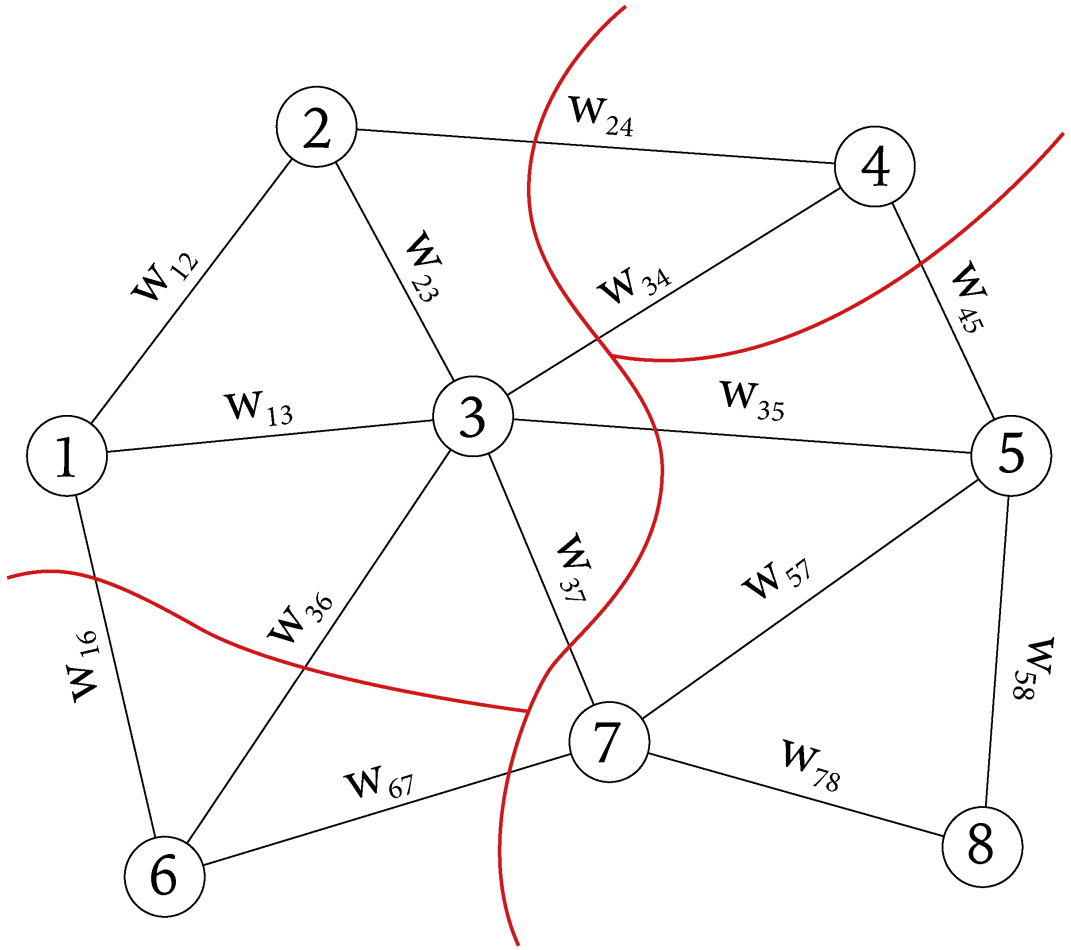
Illustration on a toy example

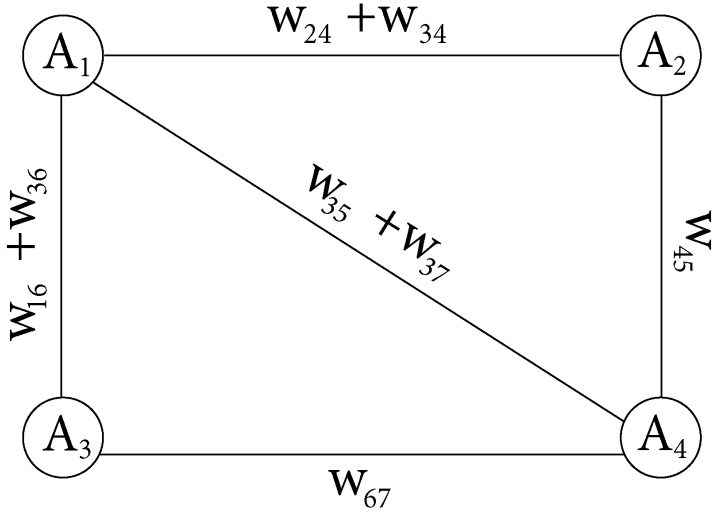




Cut Pursuit in a nutshell

Illustration on a toy example





Proposed approach[4]:

Idea: Cut Pursuit as coarse-to-fine strategy for point cloud sparsification

[4] D. Tenbrinck, F. Gaede, M. Burger: Variational Graph Methods for Efficient Point Cloud Sparsification, arXiv preprint (2019)

Proposed approach[4]:

Idea: Cut Pursuit as coarse-to-fine strategy for point cloud sparsification

- Find heuristics to extend graph cuts to *regularization terms with coupled dimensions*:

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = \frac{1}{2}\|u-f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;q}^p \right\}, \quad \alpha > 0,\ p \in [1,+\infty) \tag{7}$$

[4] D. Tenbrinck, F. Gaede, M. Burger: Variational Graph Methods for Efficient Point Cloud Sparsification, arXiv preprint (2019)

Proposed approach[4]:

Idea: Cut Pursuit as coarse-to-fine strategy for point cloud sparsification

- Find heuristics to extend graph cuts to *regularization terms with coupled dimensions*:

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = \frac{1}{2}\|u-f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;q}^p \right\}, \quad \alpha > 0,\ p \in [1,+\infty) \tag{7}$$

- *Decouple* partition problem and reduced problem:

$$\begin{aligned} u_\Pi &= \underset{u\in\mathcal{H}(\Pi)}{\arg\min}\, D(u, f_\Pi) + \beta Q(u) \\ \min_{B\subset V} \{J'(\overline{u}_\Pi; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_\Pi, f), \mathbb{1}_B\rangle + \alpha\langle \nabla R_S(\overline{u}_\Pi), \mathbb{1}_B\rangle + R'_{S^c}(\overline{u}_\Pi; \mathbb{1}_B)\} \end{aligned} \tag{8}$$

[4] D. Tenbrinck, F. Gaede, M. Burger: Variational Graph Methods for Efficient Point Cloud Sparsification, arXiv preprint (2019)

Proposed approach[4]:

Idea: Cut Pursuit as coarse-to-fine strategy for point cloud sparsification

- Find heuristics to extend graph cuts to *regularization terms with coupled dimensions*:

$$\operatorname*{argmin}_{u \in \mathcal{H}(V)} \left\{ J(u) = \frac{1}{2}\|u - f\|_2^2 + \frac{\alpha}{p}\|\nabla_w u\|_{p;q}^p \right\}, \quad \alpha > 0,\ p \in [1, +\infty) \tag{7}$$

- *Decouple* partition problem and reduced problem:

$$\begin{aligned} u_\Pi &= \operatorname*{arg\,min}_{u \in \mathcal{H}(\Pi)} D(u, f_\Pi) + \beta Q(u) \\ \min_{B \subset V} \{J'(\overline{u}_\Pi; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_\Pi, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_\Pi), \mathbb{1}_B \rangle + R'_{S^c}(\overline{u}_\Pi; \mathbb{1}_B)\} \end{aligned} \tag{8}$$

- Parameter α controls level-of-details, parameter β controls shape regularization

[4] D. Tenbrinck, F. Gaede, M. Burger: Variational Graph Methods for Efficient Point Cloud Sparsification, arXiv preprint (2019)

Numerical results

Special case: Octree partitioning

Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_\Pi) := \frac{1}{2}||u - f_\Pi||^2$.

$$
\begin{aligned}
u_\Pi &= \operatorname*{arg\,min}_{u \in \mathcal{H}(\Pi)} D(u, f_\Pi) + \beta Q(u) \\
\min_{B \subset V} \{J'(\overline{u}_\Pi; \mathbb{1}_B) &= \langle \nabla D(\overline{u}_\Pi, f), \mathbb{1}_B \rangle + \alpha \langle \nabla R_S(\overline{u}_\Pi), \mathbb{1}_B \rangle + R'_{S^c}(\overline{u}_\Pi; \mathbb{1}_B)\}
\end{aligned}
\tag{9}
$$

Numerical results

Special case: Octree partitioning

Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_{\Pi}) := \frac{1}{2}||u - f_{\Pi}||^2$.

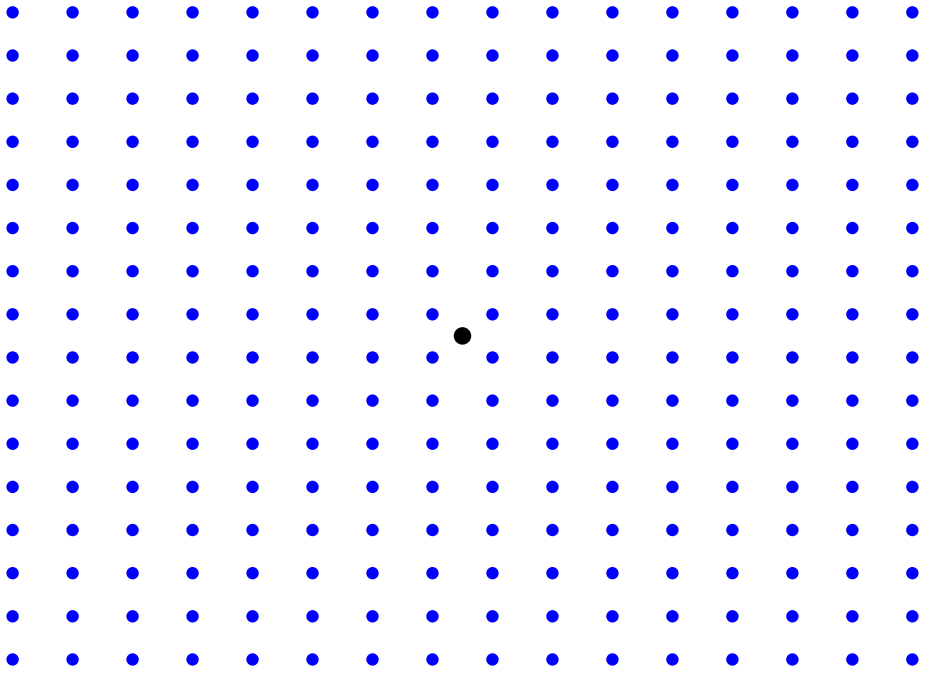
$$\begin{aligned} u_{\Pi} &= \overline{f}_{\Pi} \\ \min_{B \subset V} \{J'(\overline{f}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{f}_{\Pi}, f), \mathbb{1}_B \rangle\} \end{aligned} \tag{9}$$

Numerical results

Special case: Octree partitioning

Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_{\Pi}) := \frac{1}{2}||u - f_{\Pi}||^2$.

$$\begin{aligned} u_{\Pi} &= \overline{f}_{\Pi} \\ \min_{B \subset V} \{J'(\overline{f}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{f}_{\Pi}, f), \mathbb{1}_B \rangle\} \end{aligned} \tag{9}$$

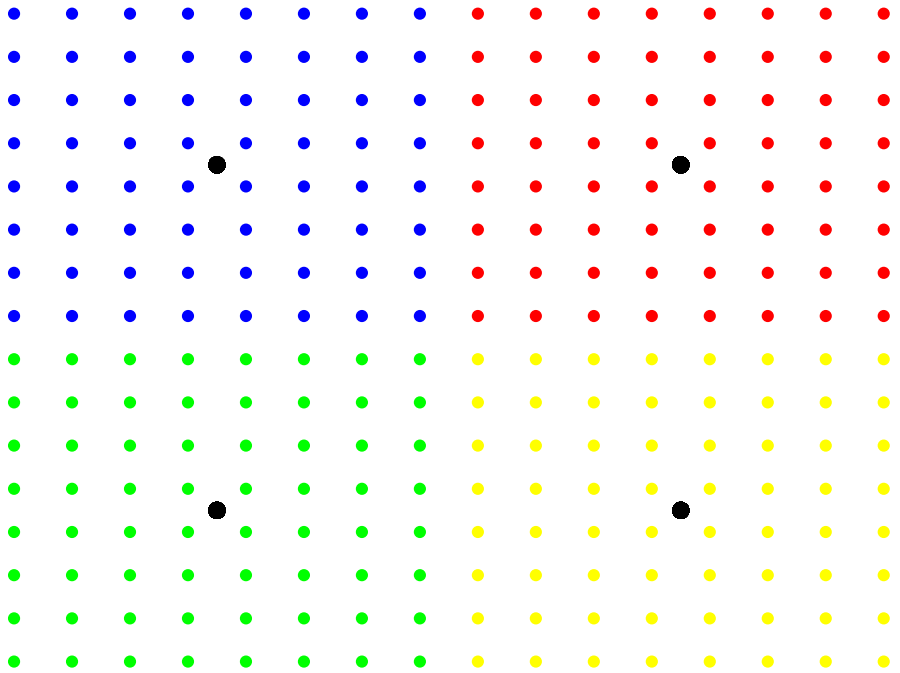


Numerical results

Special case: Octree partitioning

Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_{\Pi}) := \frac{1}{2}||u - f_{\Pi}||^2$.

$$\begin{aligned} u_{\Pi} &= \overline{f}_{\Pi} \\ \min_{B \subset V} \{J'(\overline{f}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{f}_{\Pi}, f), \mathbb{1}_B \rangle\} \end{aligned} \tag{9}$$

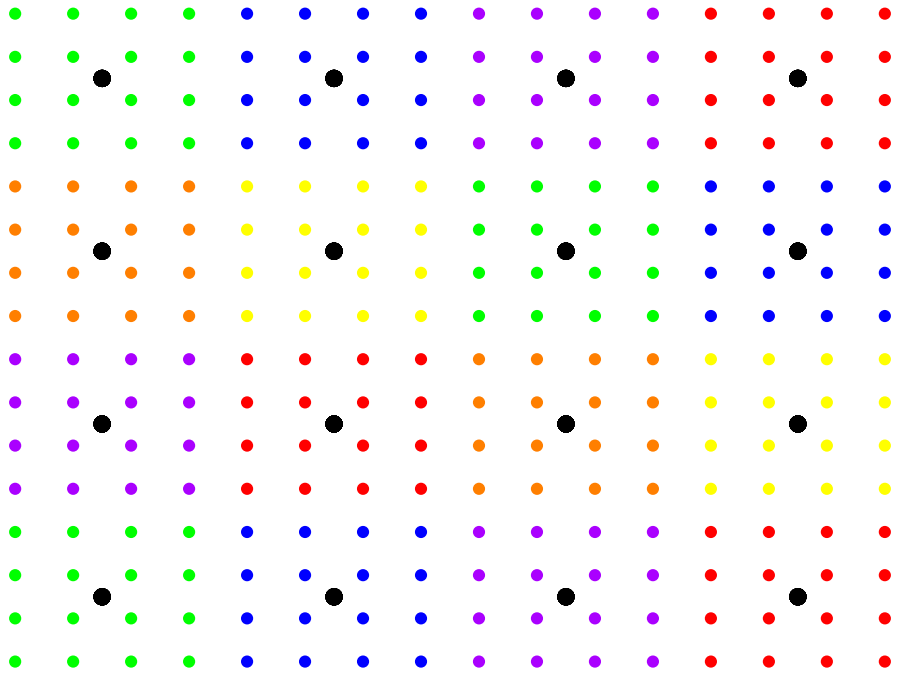


Numerical results

Special case: Octree partitioning

Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_{\Pi}) := \frac{1}{2}||u - f_{\Pi}||^2$.

$$\begin{aligned} u_{\Pi} &= \overline{f}_{\Pi} \\ \min_{B \subset V} \{J'(\overline{f}_{\Pi}; \mathbb{1}_B) &= \langle \nabla D(\overline{f}_{\Pi}, f), \mathbb{1}_B \rangle\} \end{aligned} \tag{9}$$

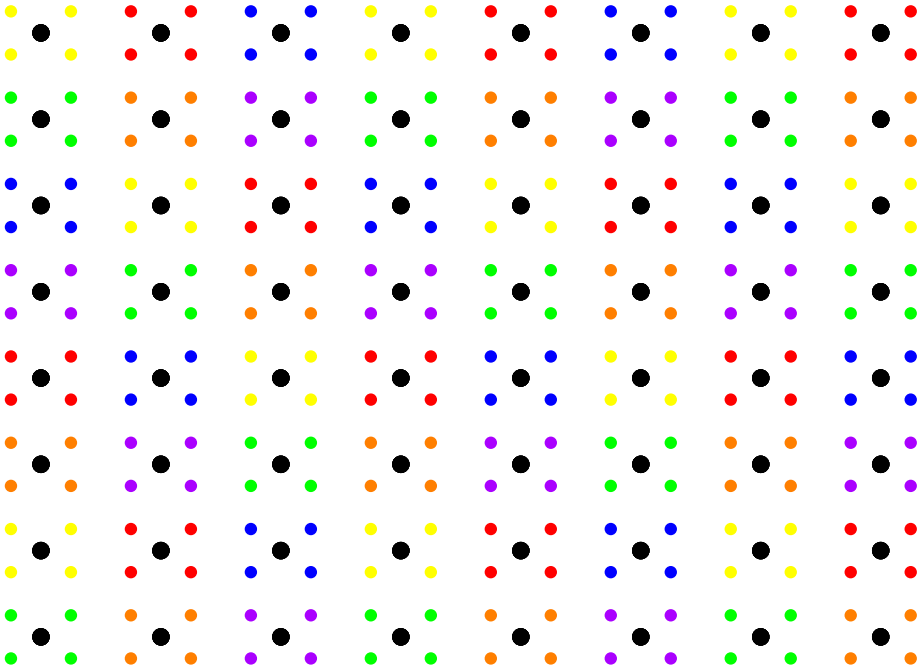


Numerical results

Special case: Octree partitioning

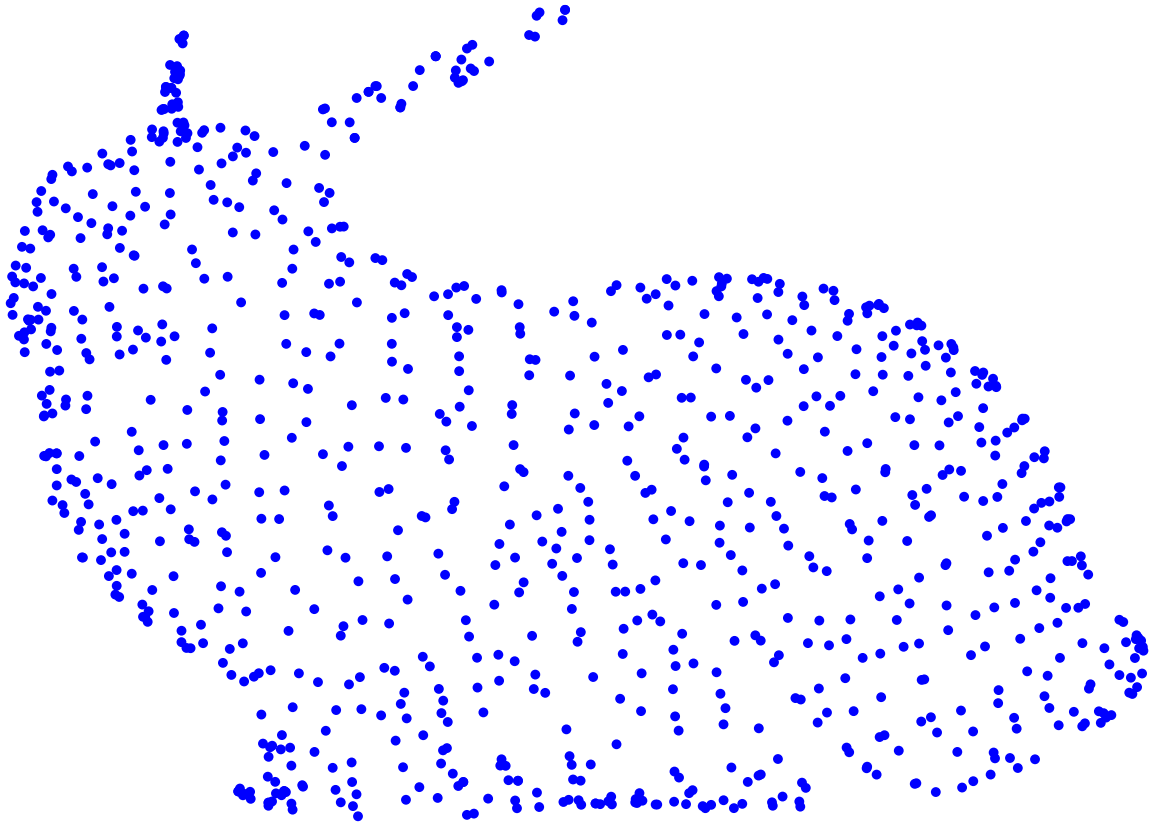
Observation: Setting $\alpha = \beta = 0$ reduces Cut Pursuit to **Octree partitioning**, if $D(u, f_{I\!I}) := \frac{1}{2}||u - f_{I\!I}||^2$.

$$\begin{aligned} u_{I\!I} &= \overline{f}_{I\!I} \\ \min_{B \subset V} \{J'(\overline{f}_{I\!I}; \mathbb{1}_B) &= \langle \nabla D(\overline{f}_{I\!I}, f), \mathbb{1}_B \rangle\} \end{aligned} \tag{9}$$

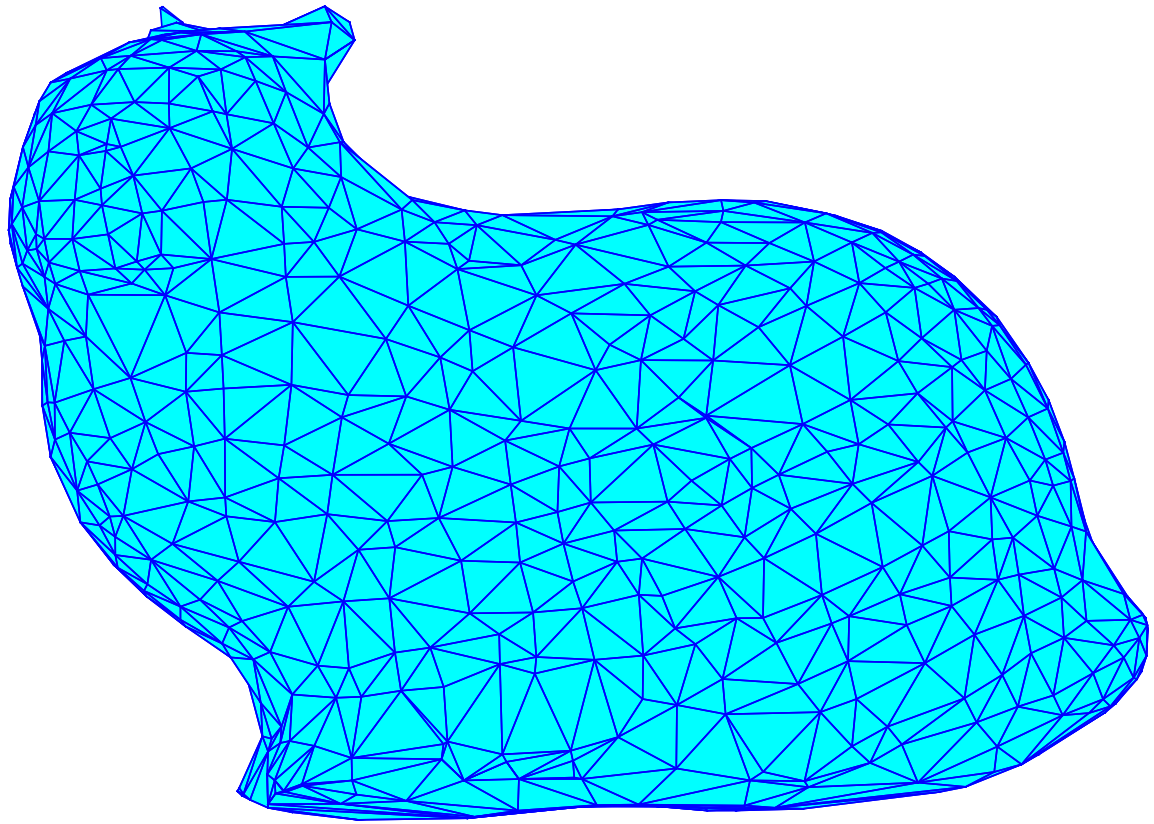


Numerical results

Regularization with ℓ^2 norm



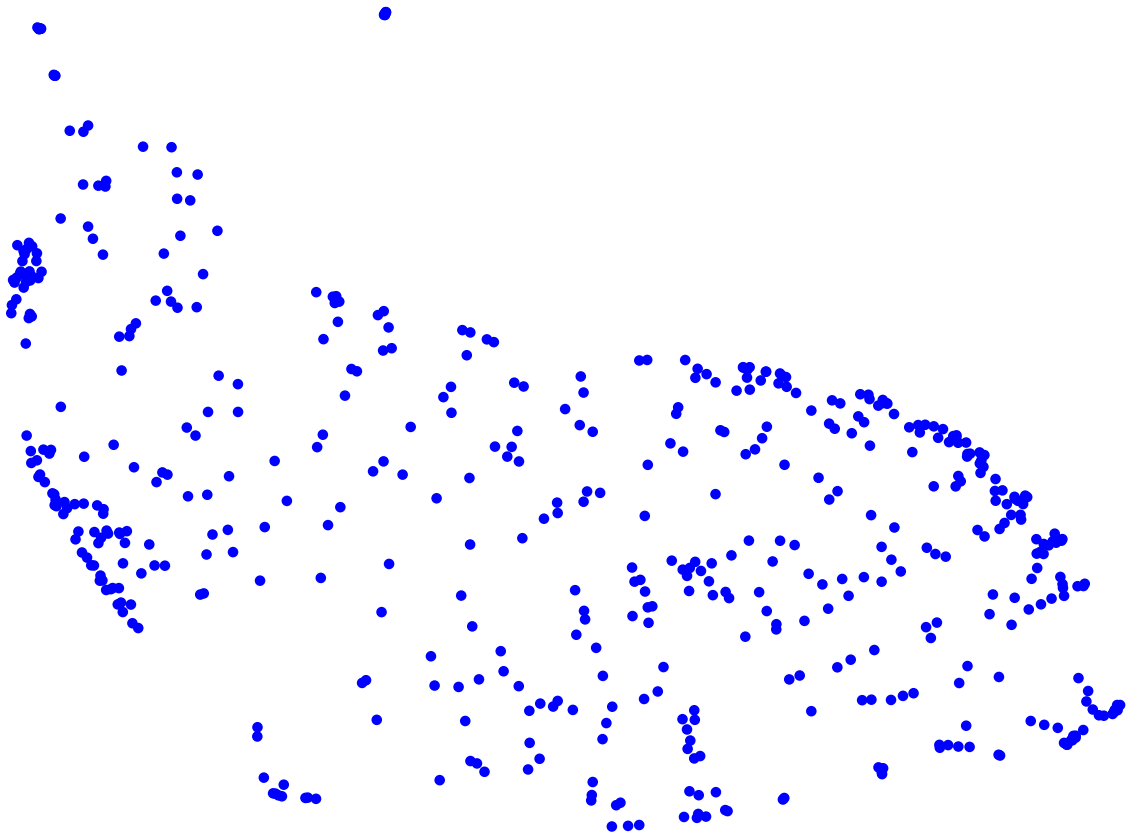
Sparsified point cloud with medium regularization ($\beta = 10$)



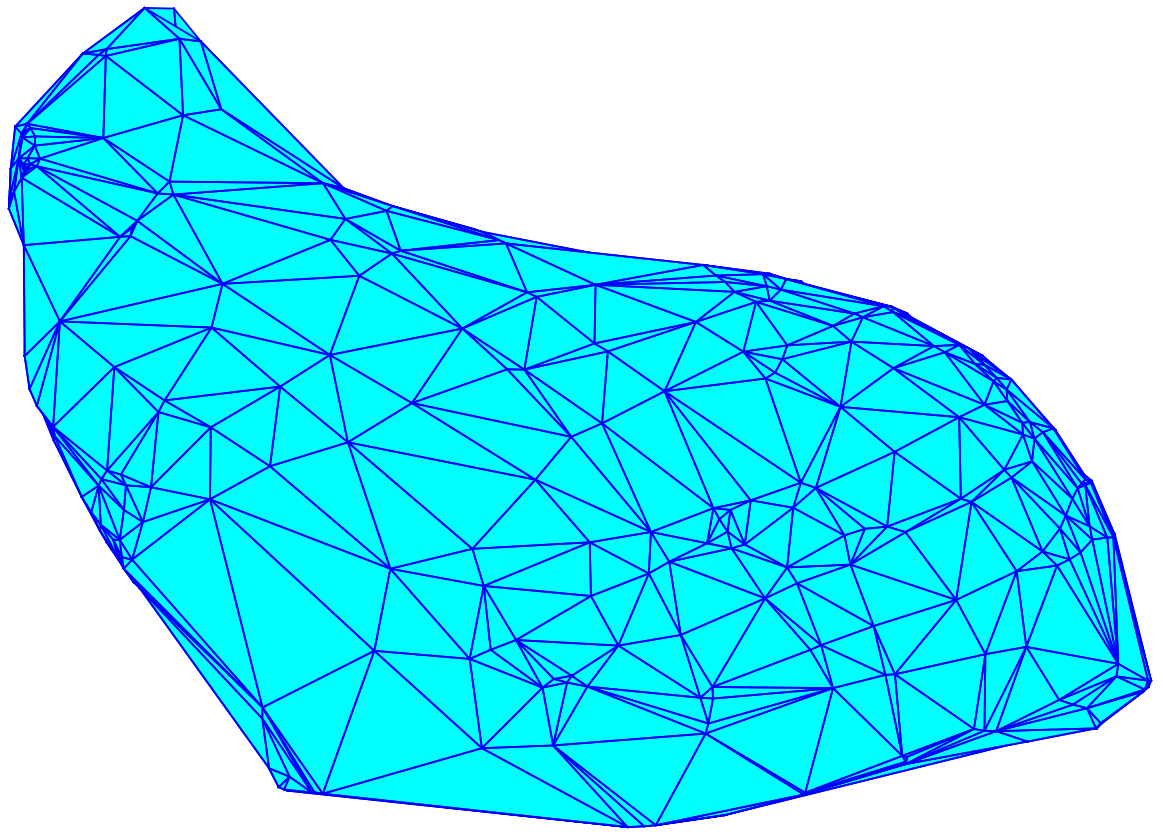
Triangulated point cloud with medium regularization ($\beta = 10$)

Numerical results

Regularization with ℓ^2 norm



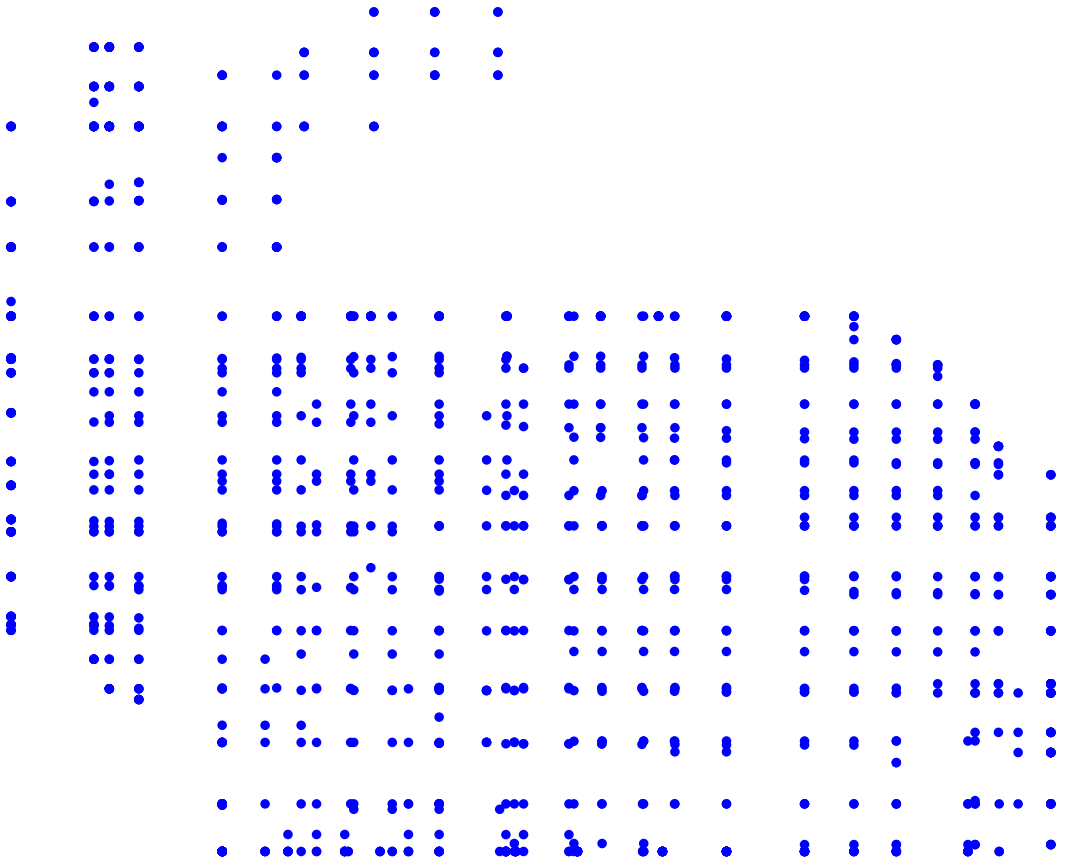
Sparsified point cloud with strong regularization ($\beta = 70$)



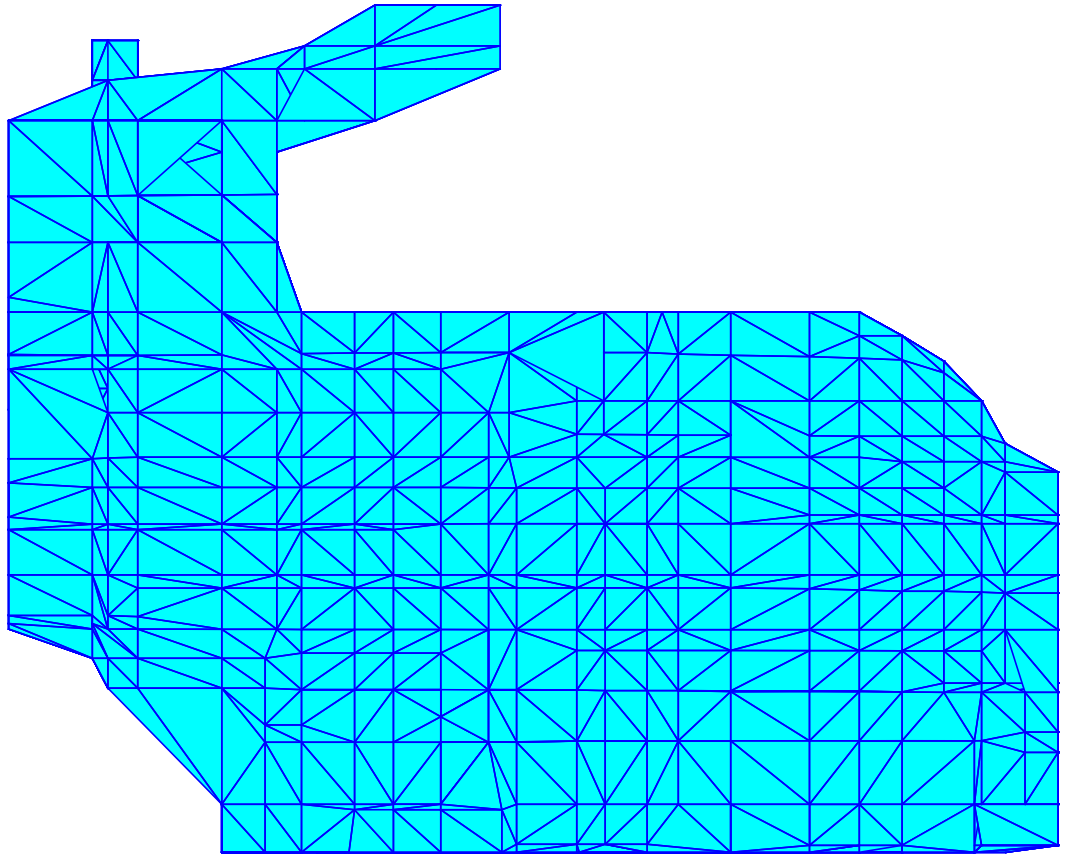
Triangulated point cloud with strong regularization ($\beta = 70$)

Numerical results

Regularization with (anisotropic) ℓ^1 norm



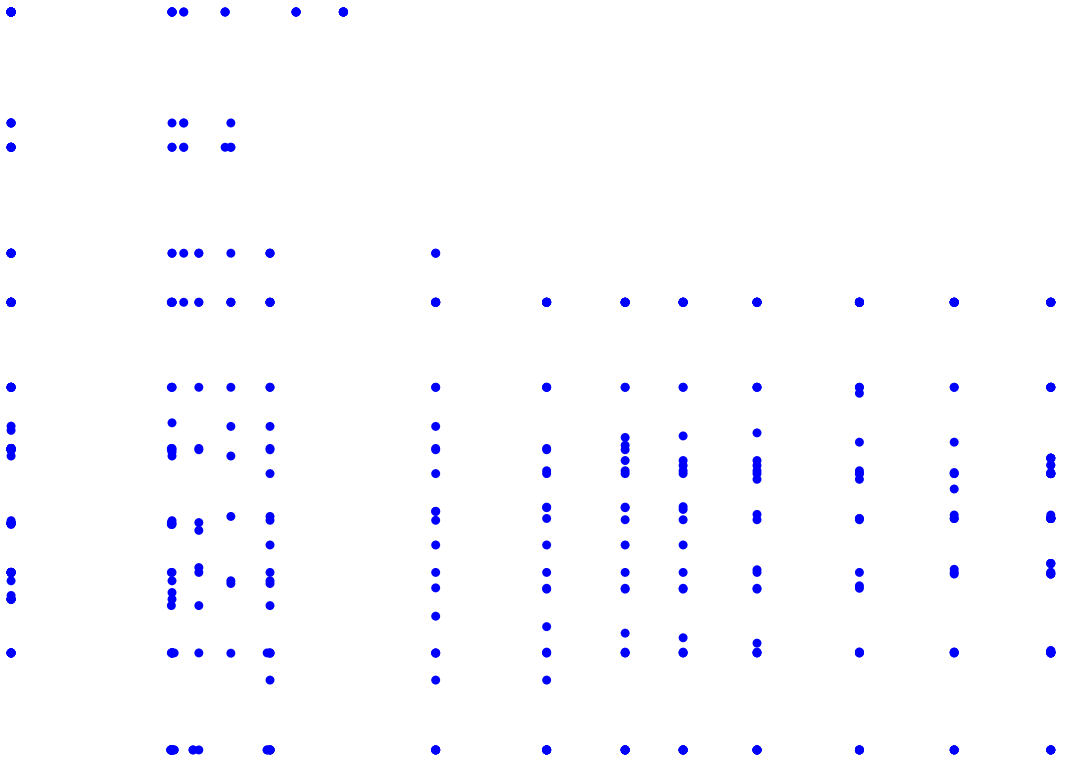
Sparsified point cloud with medium regularization ($\beta = 10$)



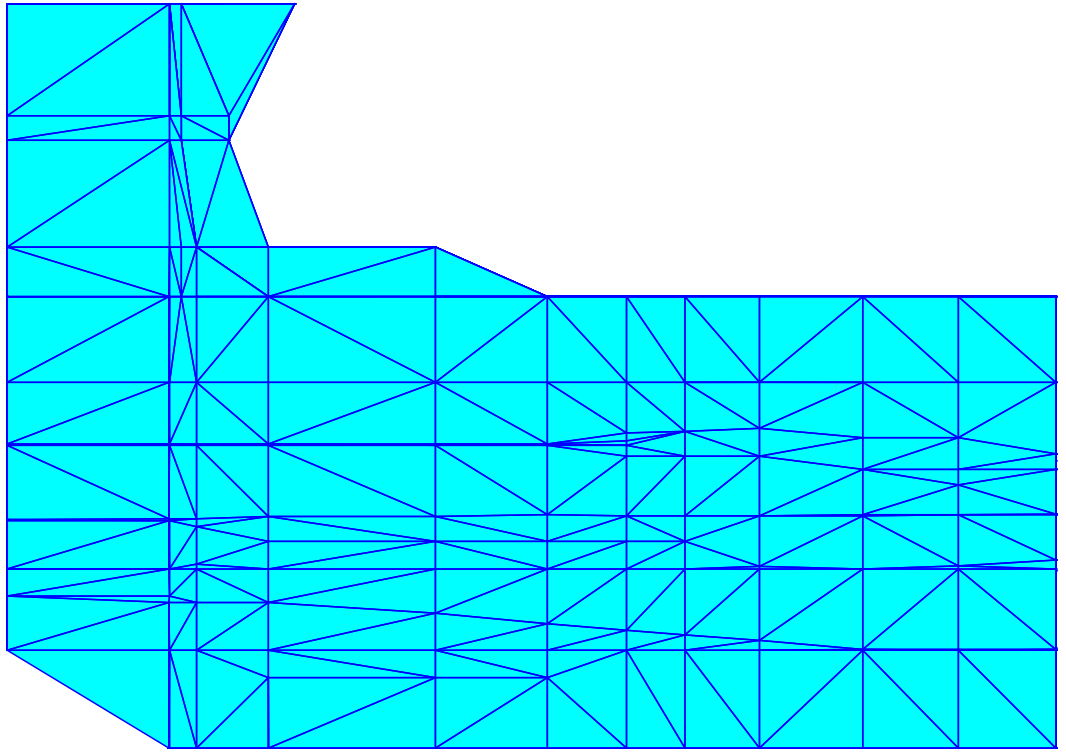
Triangulated point cloud with medium regularization ($\beta = 10$)

Numerical results

Regularization with (anisotropic) ℓ^1 norm



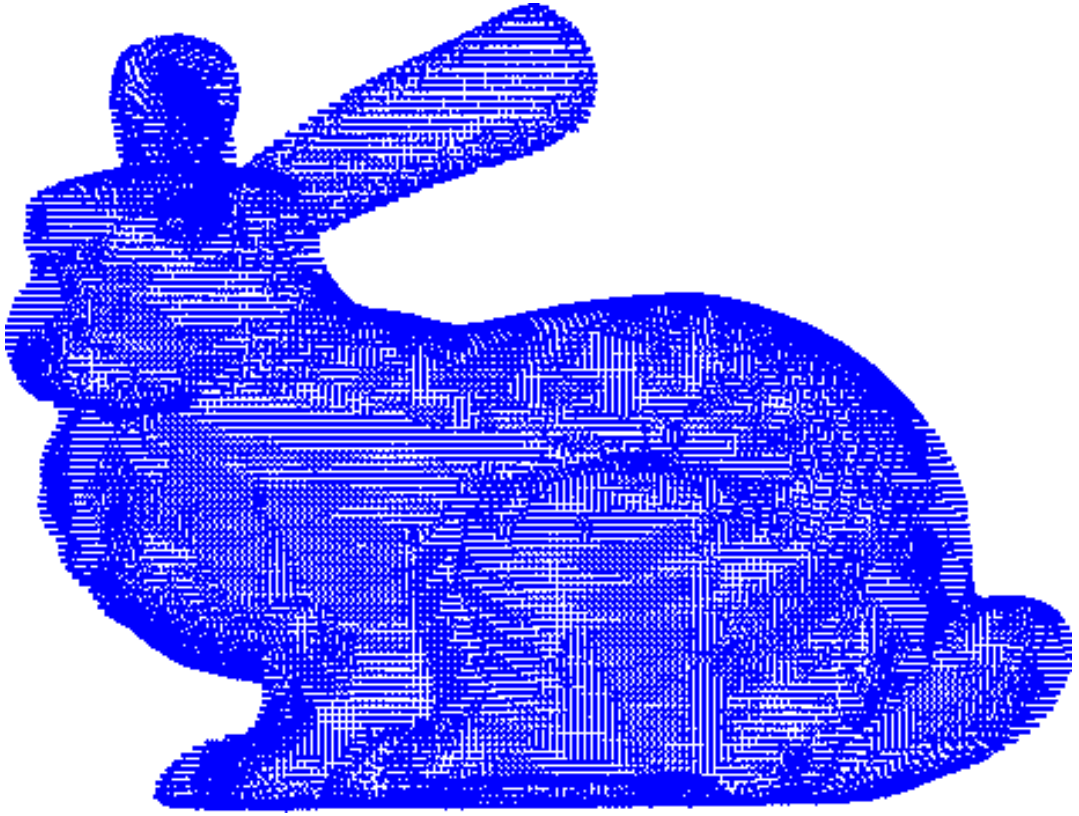
Sparsified point cloud with strong regularization ($\beta = 50$)



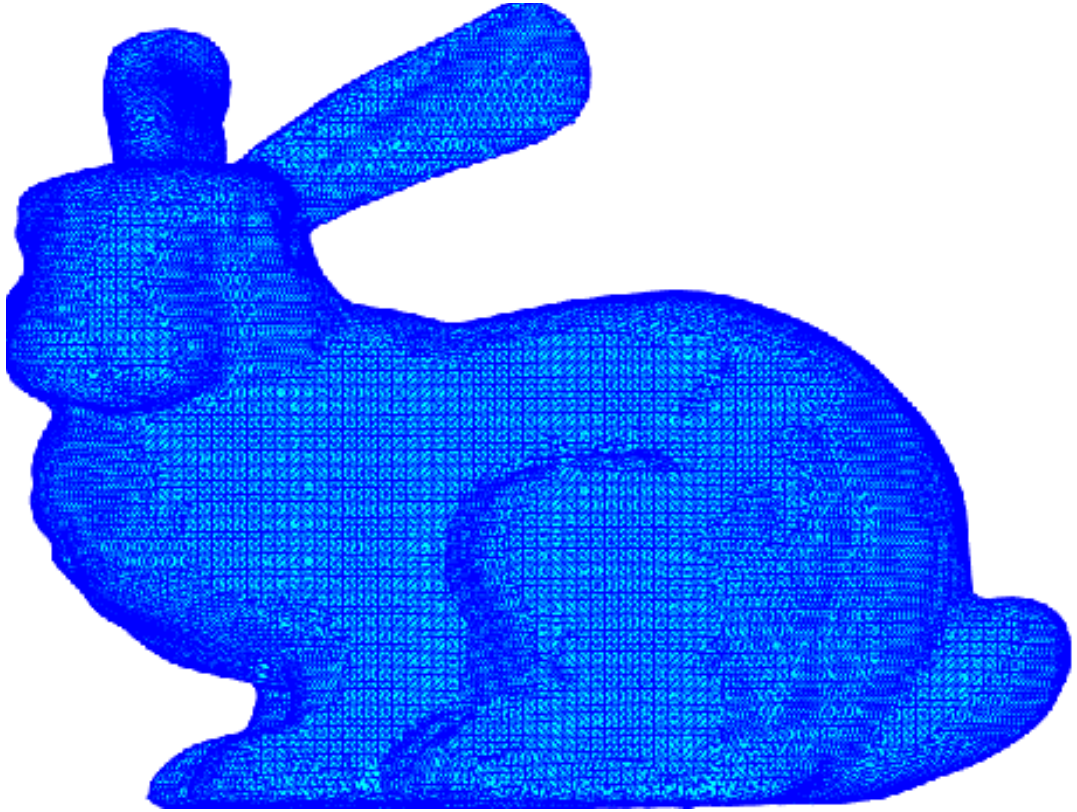
Triangulated point cloud with strong regularization ($\beta = 50$)

Numerical results

Regularization with weighted ℓ^0 norm



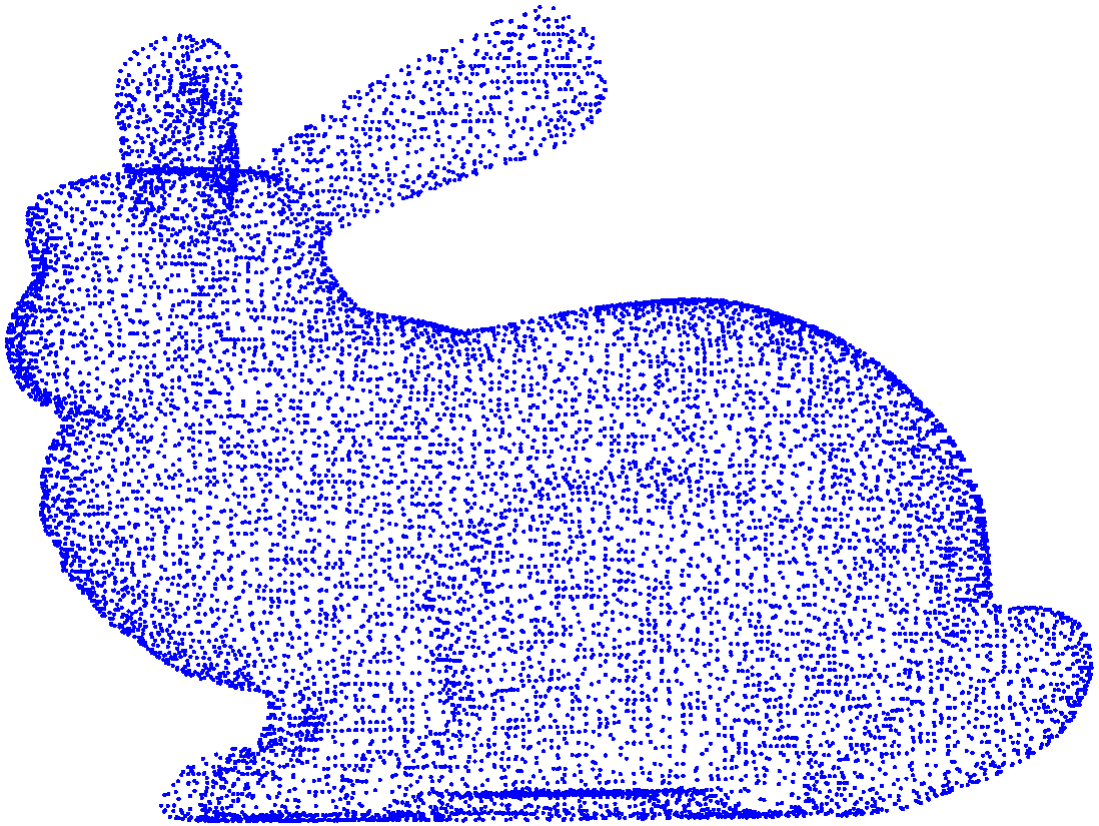
Original point cloud data ($35,947$ points)



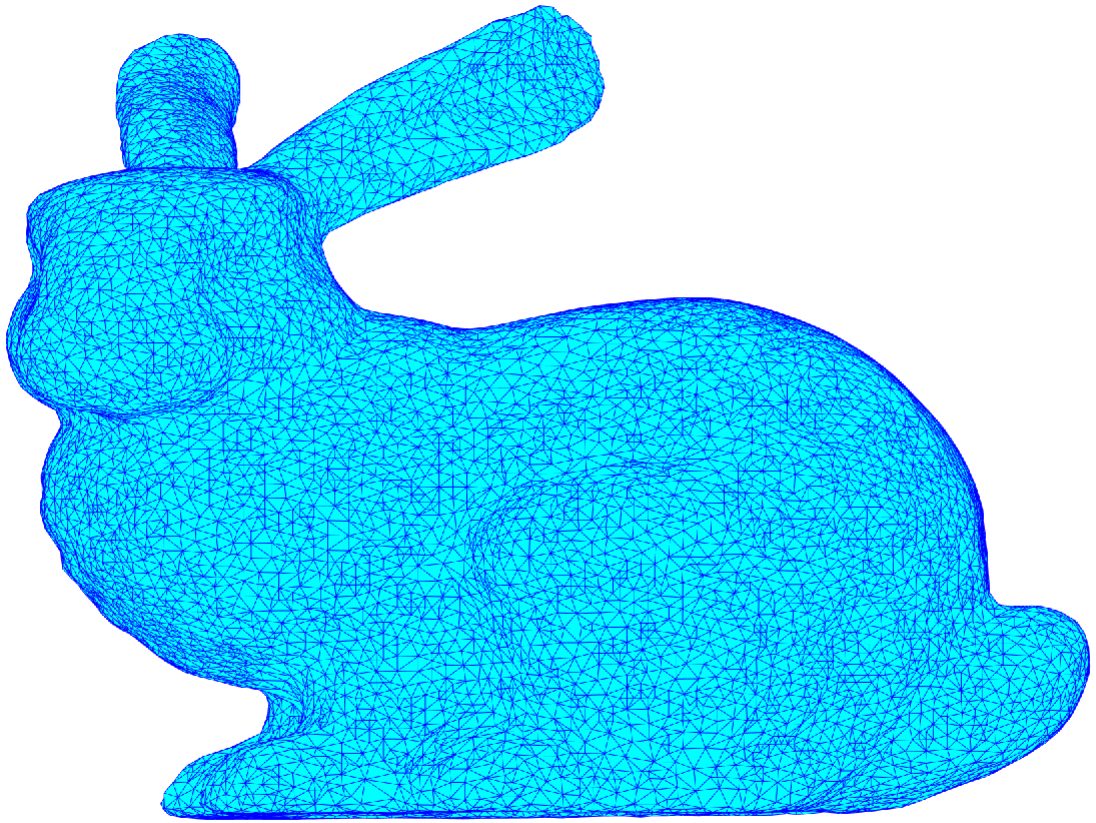
Triangulation of original point cloud data

Numerical results

Regularization with weighted ℓ^0 norm



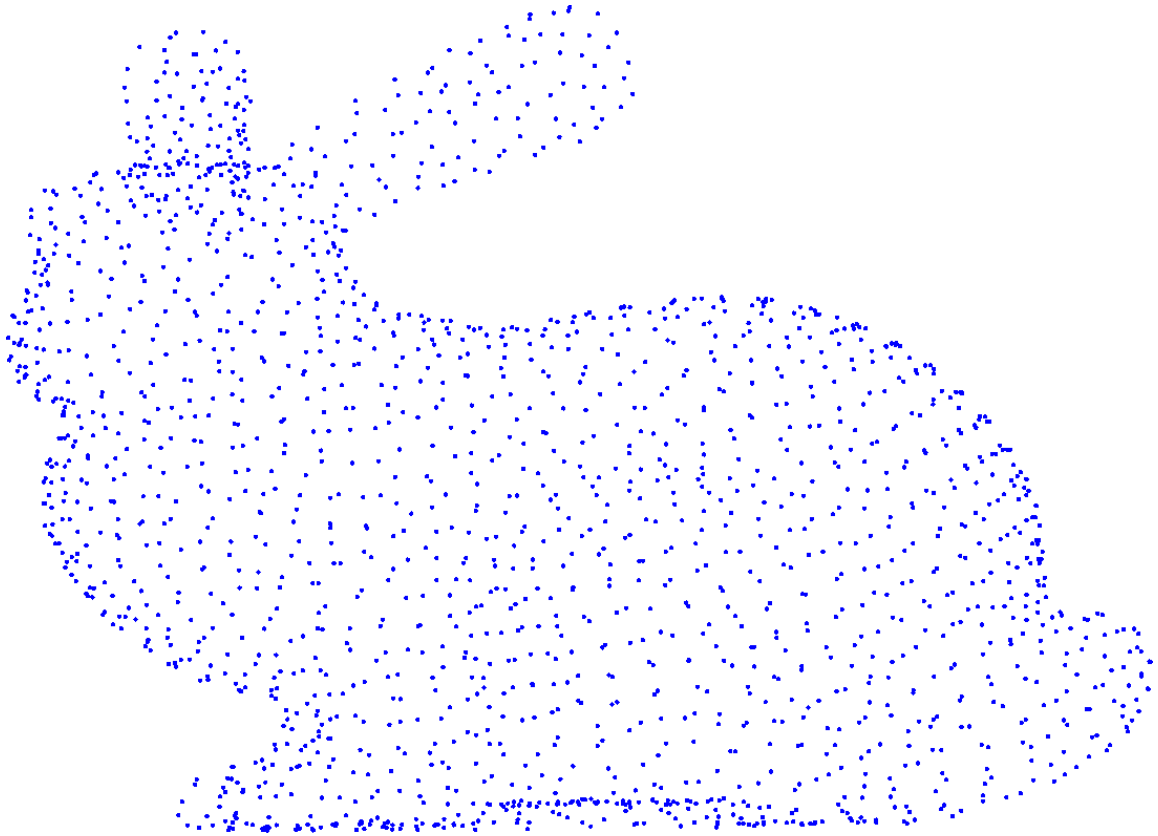
Point cloud sparsification for $\alpha = 0.3$ ($12,105$ points)



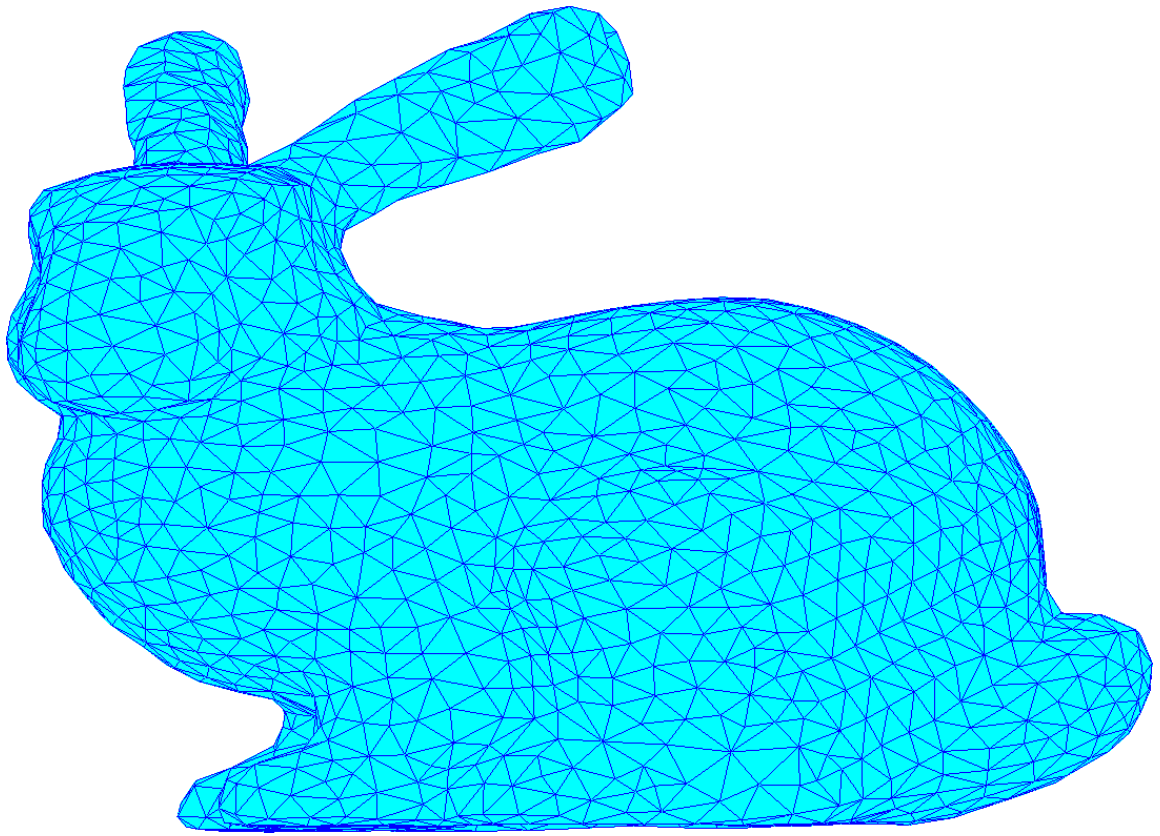
Triangulation of sparsified point cloud data

Numerical results

Regularization with weighted ℓ^0 norm



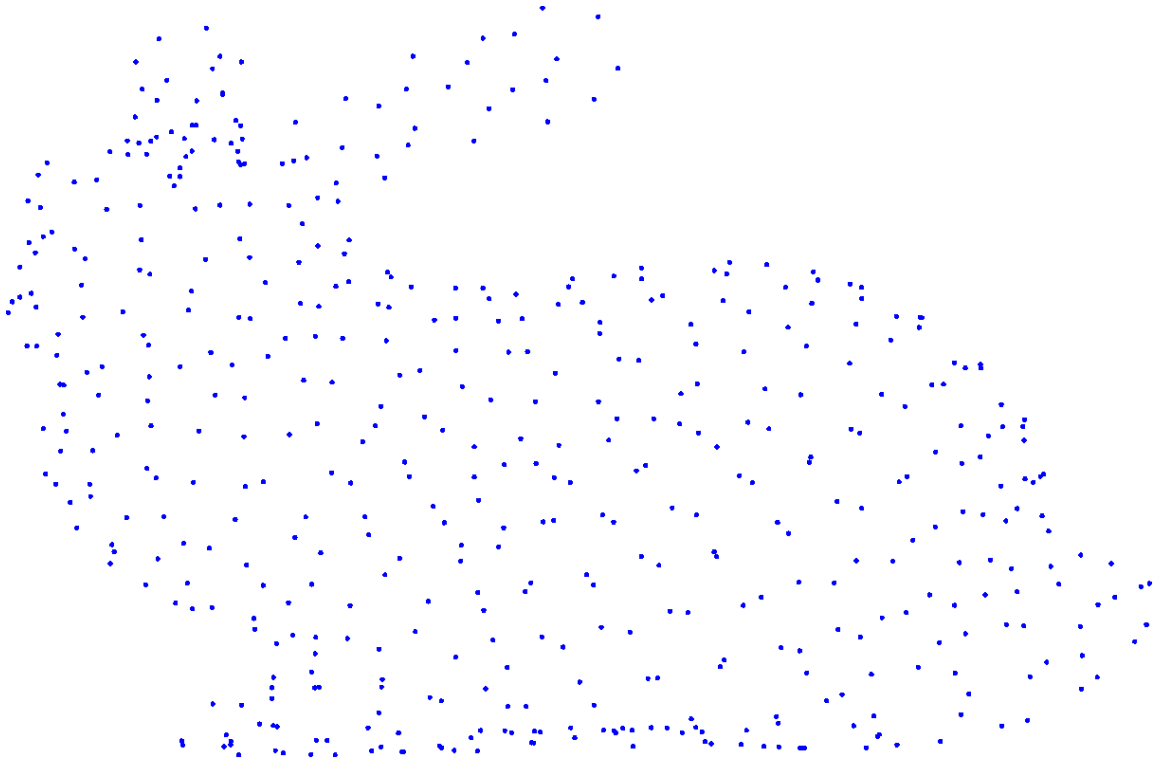
Point cloud sparsification for $\alpha = 1$ ($1,912$ points)



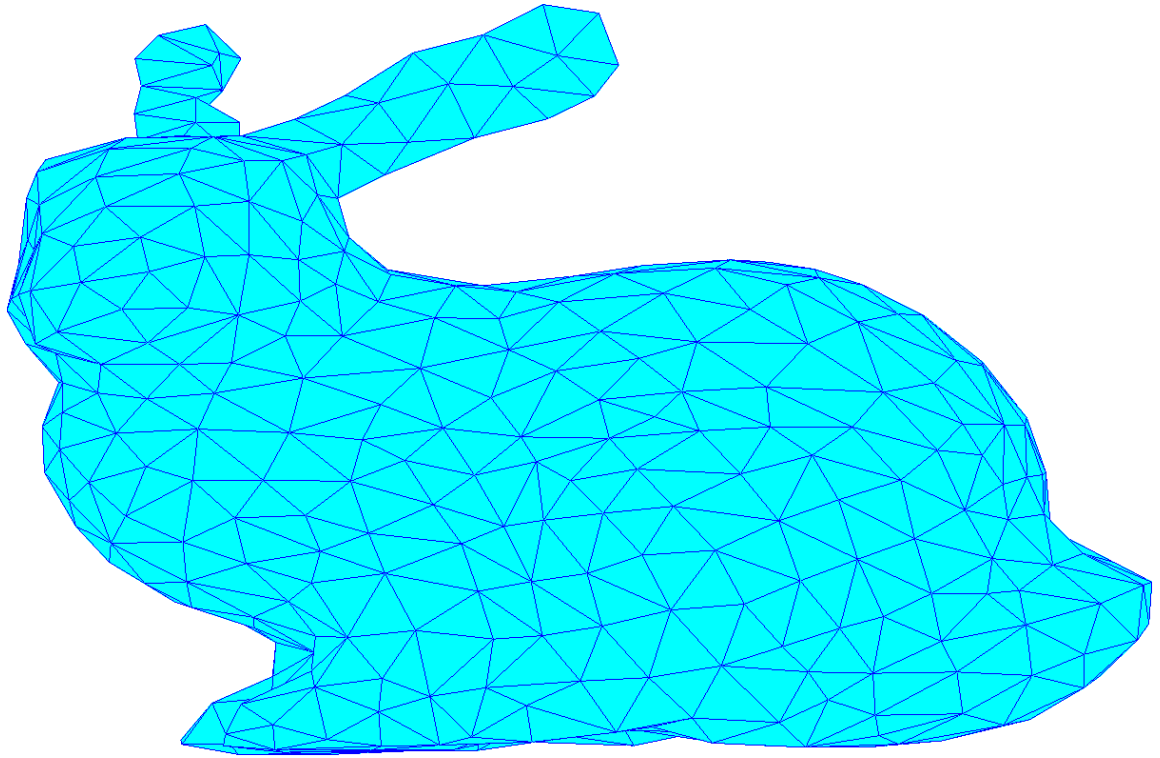
Triangulation of sparsified point cloud data

Numerical results

Regularization with weighted ℓ^0 norm



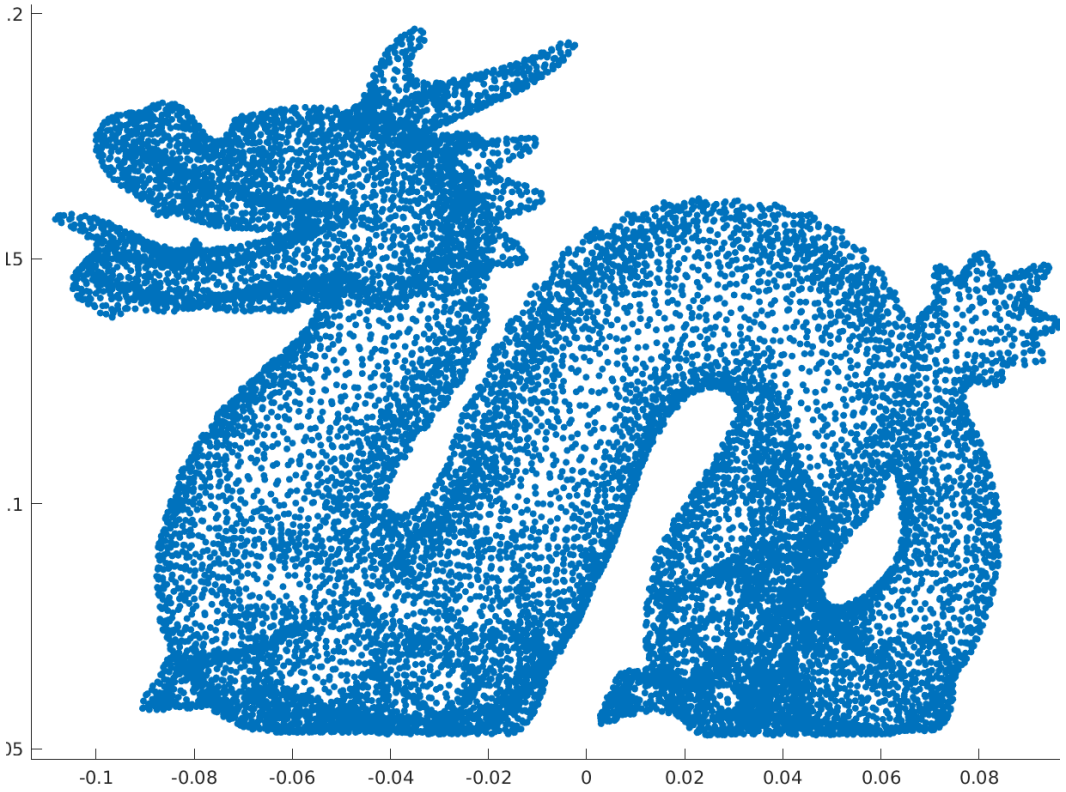
Point cloud sparsification for $\alpha = 3.5$ (498 points)

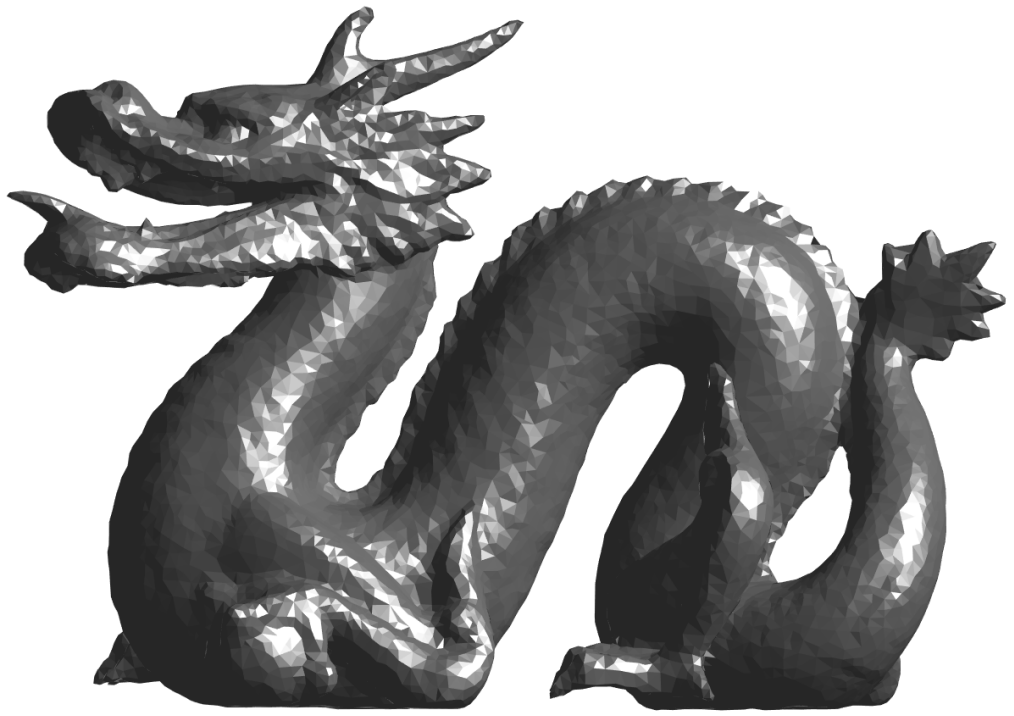


Triangulation of sparsified point cloud data

Numerical results

Regularization with ℓ^0 norm

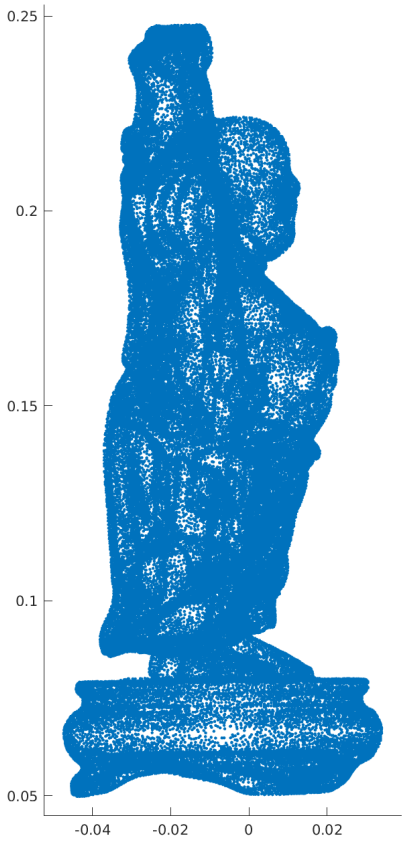


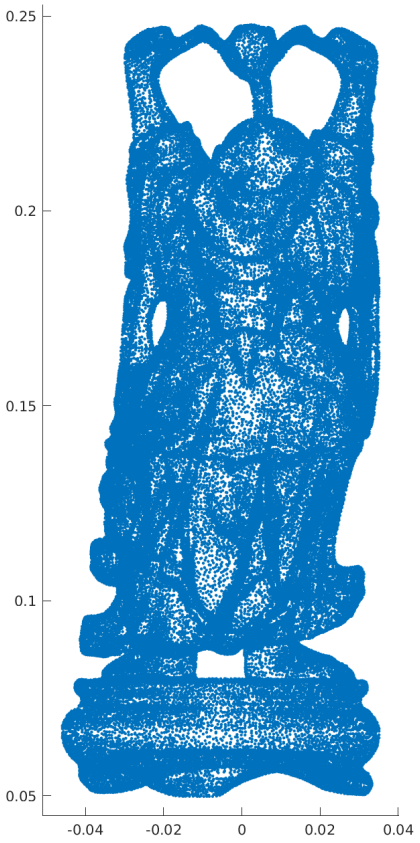


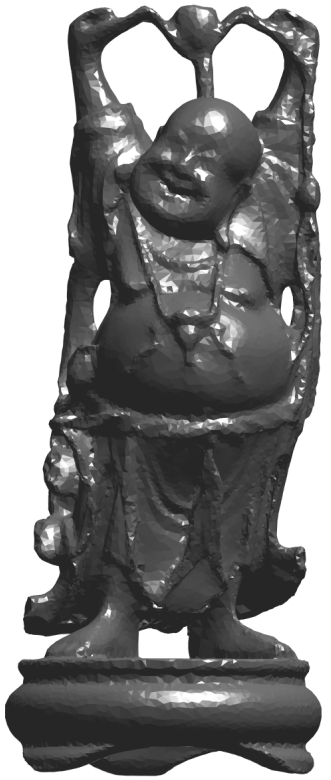
Dragon dataset ($435,545$ points). Time needed: 70.6 seconds. Points left: $16,138$ (3.71%)

Numerical results

Regularization with ℓ^0 norm







Buddha dataset ($543,524$ points).Time needed: 94 seconds. Points left: $29,168$ (5.37%)

Numerical results

Regularization with ℓ^0 norm

Data set	Fine-to-coarse approach	Cut Pursuit approach
Bunny: $35,947$ points	126s $8,034$ points left (22.35%)	**3.7s** $7,794$ points left (21.69%)
Dragon: $435,545$ points	$8,239$s $16,438$ points left (3.77%)	**70.6s** $16,138$ points left (3.71%)
Buddha: $543,524$ points	$3,305$s $29,168$ points left (5.37%)	**94s** $26,247$ points left (4.83%)

Table: Comparison of overall runtime in seconds between a direct optimization via preconditioned primal-dual optimization and the weighted ℓ_0 Cut Pursuit algorithm for point cloud sparsification tested on three different data sets.

Numerical results

Regularization with ℓ^0 norm

Data set	Fine-to-coarse approach	Cut Pursuit approach
Bunny: $35,947$ points	126s $8,034$ points left (22.35%)	**3.7s** $7,794$ points left (21.69%)
Dragon: $435,545$ points	$8,239$s $16,438$ points left (3.77%)	**70.6s** $16,138$ points left (3.71%)
Buddha: $543,524$ points	$3,305$s $29,168$ points left (5.37%)	**94s** $26,247$ points left (4.83%)

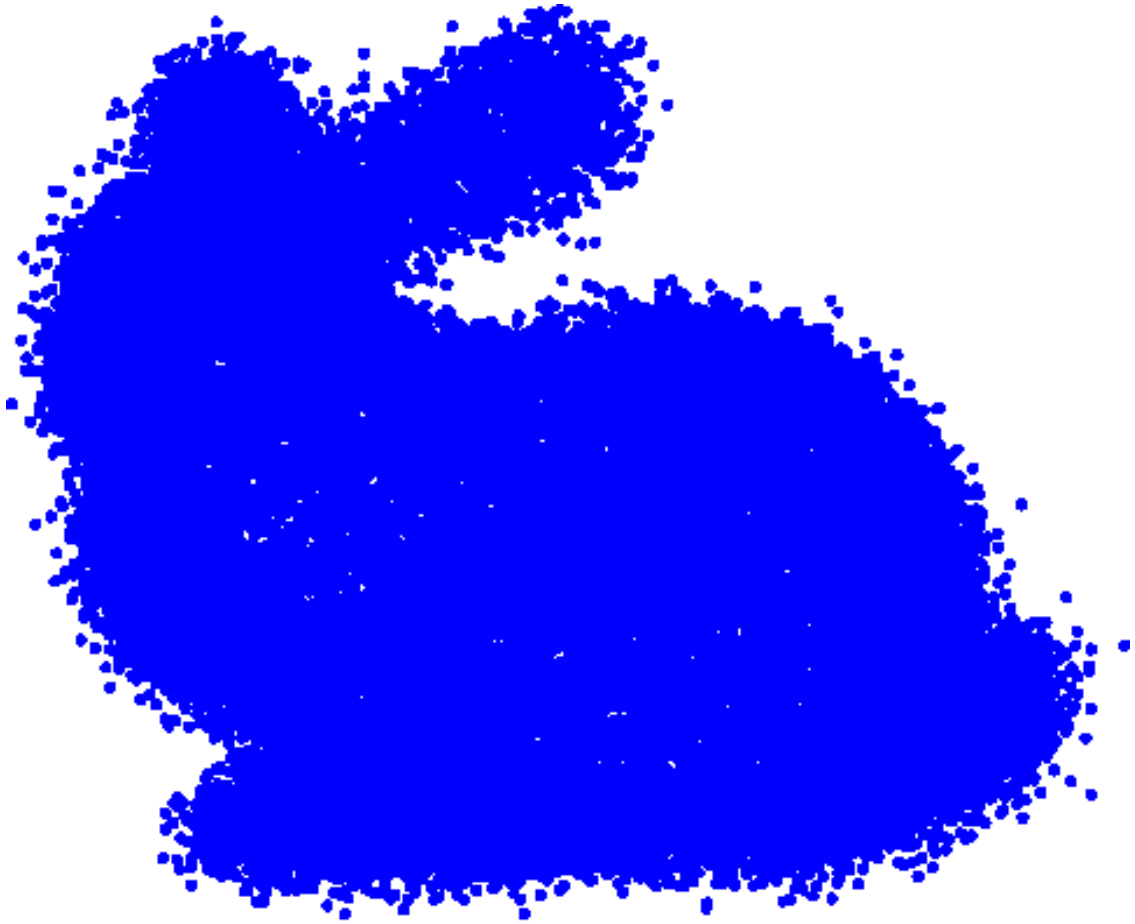
Table: Comparison of overall runtime in seconds between a direct optimization via preconditioned primal-dual optimization and the weighted ℓ_0 Cut Pursuit algorithm for point cloud sparsification tested on three different data sets.

$\rightarrow$ Speed up of factor 100 possible!

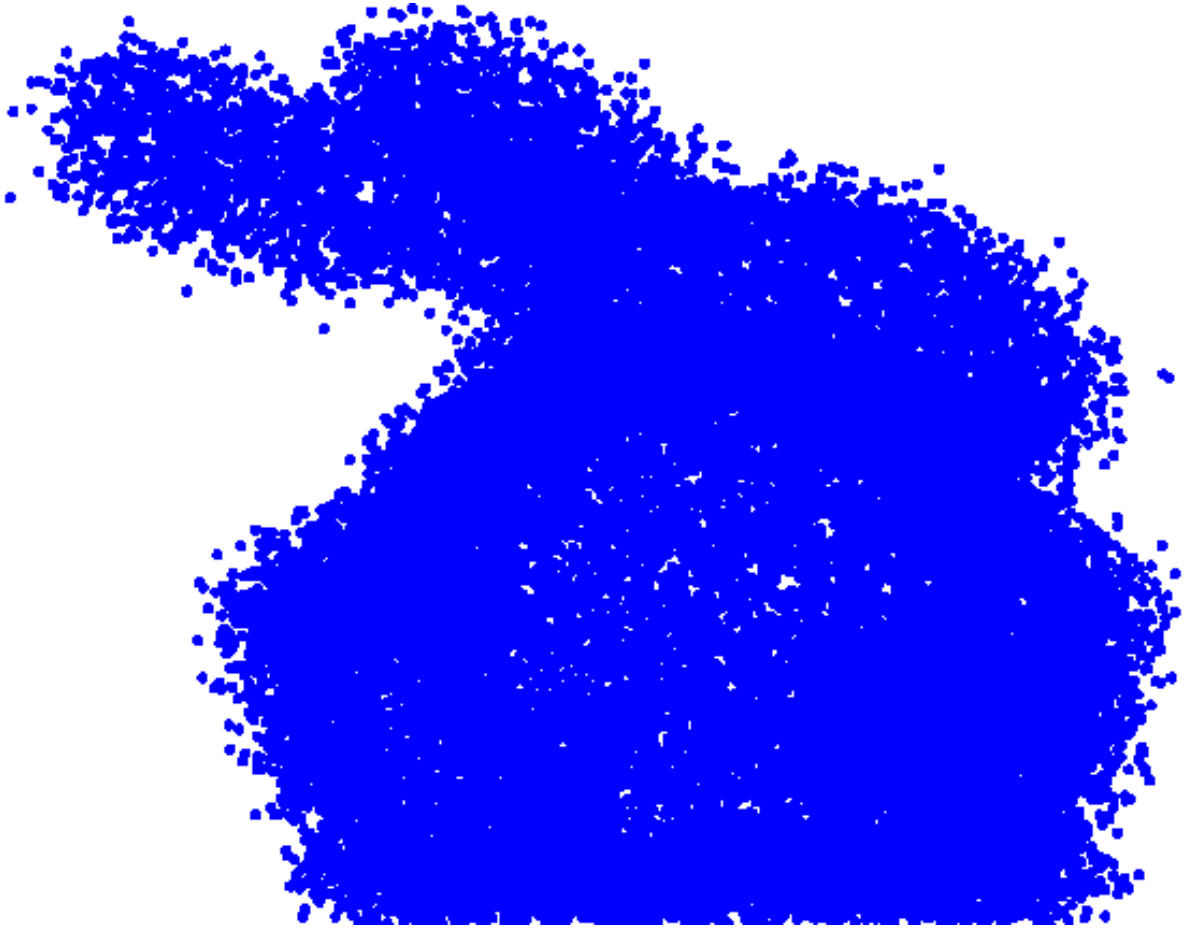
Numerical results

Impact of the regularization on noisy data

Noisy 3D point cloud:



Noisy point cloud of Bunny data set (front)

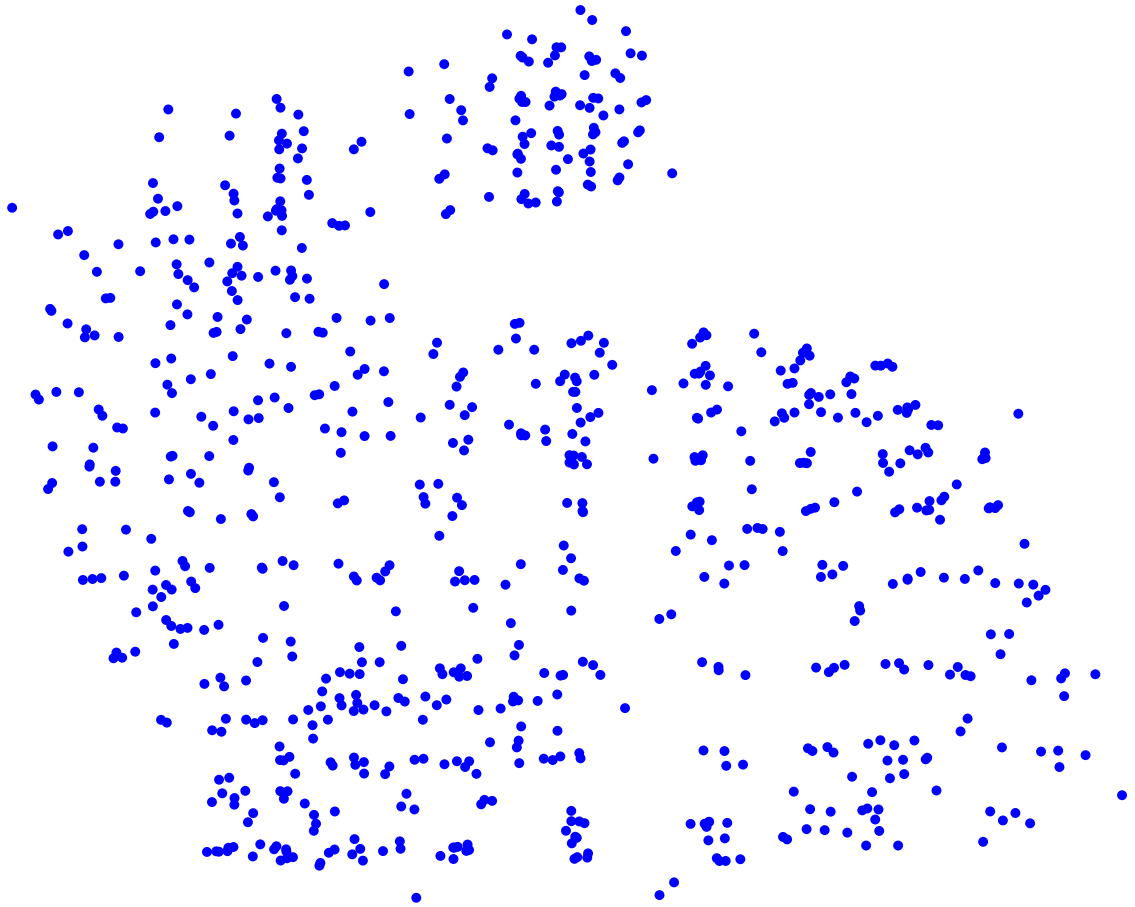


Noisy point cloud of Bunny data set (side)

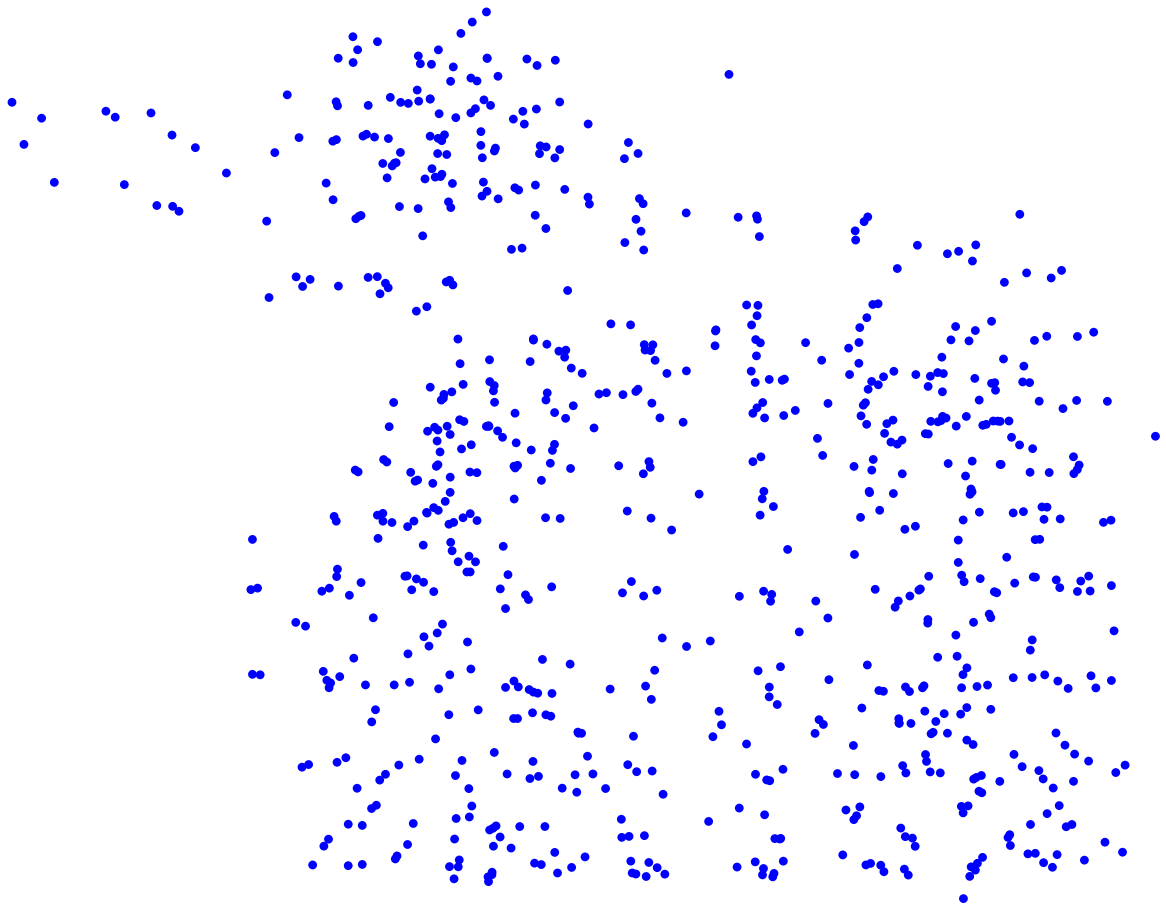
Numerical results

Impact of the regularization on noisy data

Sparsification without regularization (Octree):



Sparsified point cloud of Bunny data set (front)

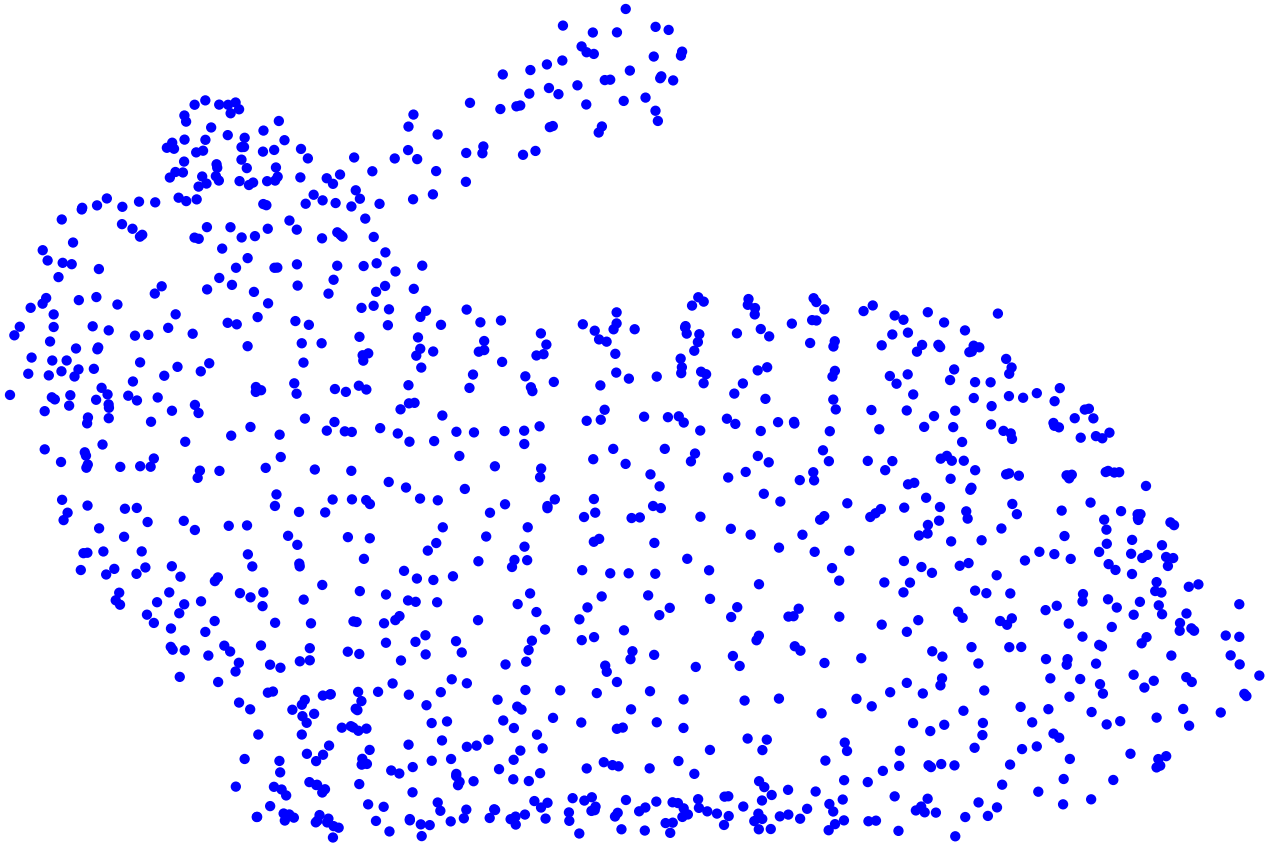


Sparsified point cloud of Bunny data set (side)

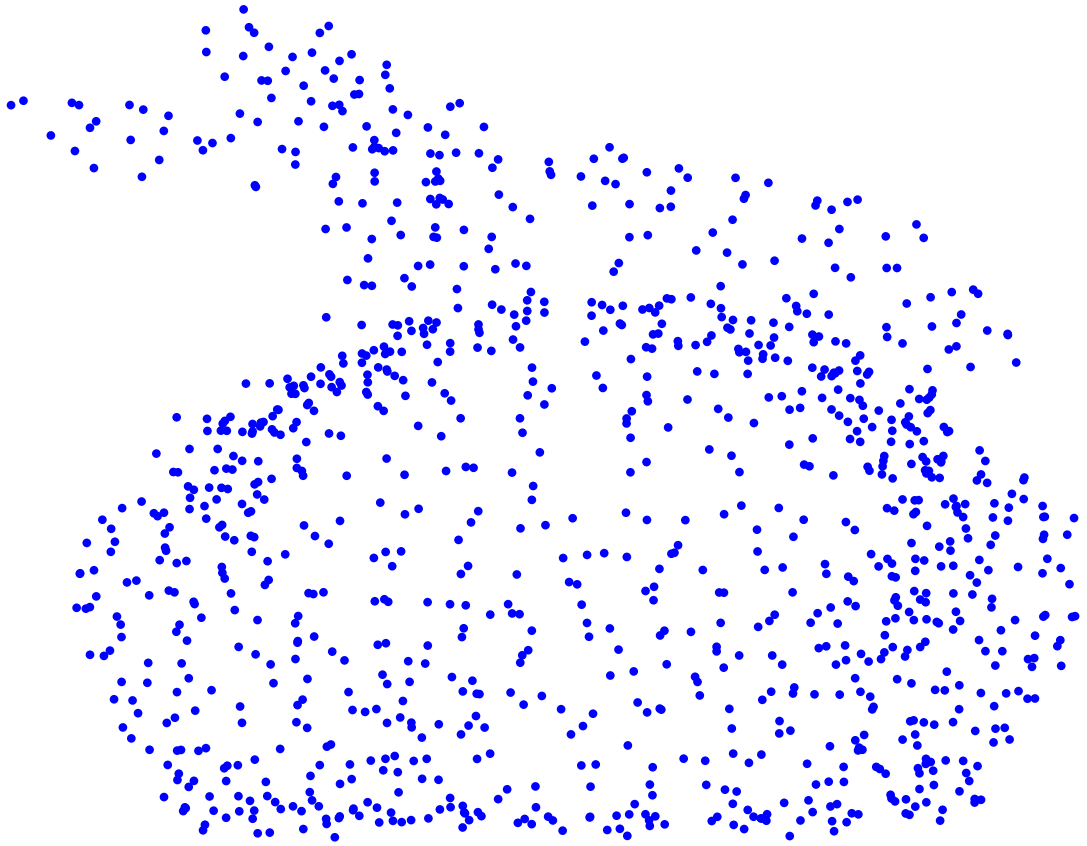
Numerical results

Impact of the regularization on noisy data

Sparsification using ℓ^2 regularization:



Sparsified point cloud of Bunny data set (front)



Sparsified point cloud of Bunny data set (side)

Conclusion and Outlook

What you should take home:

Conclusion and Outlook

What you should take home:

- If you are interested in coarse (intermediate) solutions use Cut Pursuit

Conclusion and Outlook

What you should take home:

- If you are interested in coarse (intermediate) solutions use Cut Pursuit
- Decouple partition problem and reduced problem for *additional flexibility* and to *incorporate a-priori knowledge*

Conclusion and Outlook

What you should take home:

- If you are interested in coarse (intermediate) solutions use Cut Pursuit
- Decouple partition problem and reduced problem for *additional flexibility* and to *incorporate a-priori knowledge*

Future work on Cut Pursuit:

Conclusion and Outlook

What you should take home:

- If you are interested in coarse (intermediate) solutions use Cut Pursuit
- Decouple partition problem and reduced problem for *additional flexibility* and to *incorporate a-priori knowledge*

Future work on Cut Pursuit:

- Cut Pursuit for high-dimensiobak point clouds (features) in machine learning

Conclusion and Outlook

What you should take home:

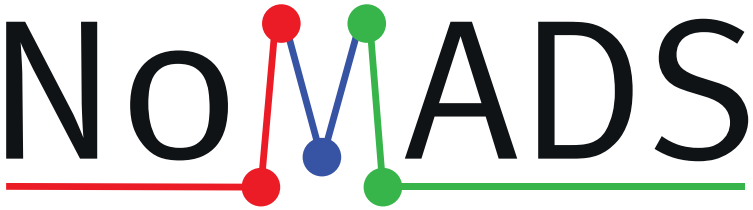
- If you are interested in coarse (intermediate) solutions use Cut Pursuit
- Decouple partition problem and reduced problem for *additional flexibility* and to *incorporate a-priori knowledge*

Future work on Cut Pursuit:

- Cut Pursuit for high-dimensiobak point clouds (features) in machine learning
- Gain additional speed-up by combining Cut Pursuit with random sampling for huge point cloud data

Acknowledgement

This work was supported by the European Union's Horizon 2020 research and innovation programme under the **Marie Sklodowska-Curie RISE** grant agreement No. 777826 (NoMADS).



Thank you very much for your attention!

Any questions?

daniel.tenbrinck@fau.de martin.burger@fau.de

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 1

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 2

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 3

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 4

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 5

Cut Pursuit in a nutshell

Efficient total variation denoising for multivariate data

Original RGB image data

Iteration 6

Proposed method

Weighted ℓ_0 regularization

$$\underset{u\in\mathcal{H}(V)}{\operatorname{argmin}} \left\{ J(u) = D(u,f) + \alpha \sum_{(v_i,v_j)\in E} \sqrt{w(v_i,v_j)} \mathbb{1}_{S_u} \right\}, \tag{10}$$
$$S_u := \{(v_i,v_j) \in E : u(v_i) \neq u(v_j)\}.$$

Cut Pursuit in a nutshell

Mathematical prerequisites

Assumptions:
Let $u, f : V \to \mathbb{R}$ be real vertex functions on a finite weighted graph G.

Cut Pursuit in a nutshell

Mathematical prerequisites

Assumptions:
Let $u, f : V \to \mathbb{R}$ be real vertex functions on a finite weighted graph G.
We are interested in a minimizing a variational model of the form:

$$J(u) \;=\; \mathcal{D}(u, f) + \alpha \mathcal{R}(u), \quad \alpha > 0 \tag{11}$$

with $\mathcal{D}$ being a data fidelity term and $\mathcal{R}$ a regularization term.

Cut Pursuit in a nutshell

Mathematical prerequisites

Assumptions:
Let $u, f : V \to \mathbb{R}$ be real vertex functions on a finite weighted graph G.
We are interested in a minimizing a variational model of the form:

$$J(u) \;=\; \mathcal{D}(u, f) + \alpha \mathcal{R}(u), \quad \alpha > 0 \tag{11}$$

with $\mathcal{D}$ being a data fidelity term and $\mathcal{R}$ a regularization term.

We assume:

1. $\mathcal{D}$ is *differentiable*
2. $\mathcal{R}$ is *separable* and *directional derivatives* exist, i.e., the term

$$R'(u; \vec{d}) \;=\; \lim_{t \to 0} \frac{R(u + t\vec{d}) - R(u)}{t} \tag{12}$$

should be well-defined for every vector u and every direction $\vec{d}$.

Cut Pursuit in a nutshell

Mathematical prerequisites

We decompose $\mathcal{R}(u)$ via a set of edges $S \subset E$ in a *differentiable* part and a *non-differentiable* part:

$$R(u) \;=\; R_S(u) + R_{S^c}(u)$$

Cut Pursuit in a nutshell

Mathematical prerequisites

We decompose $\mathcal{R}(u)$ via a set of edges $S \subset E$ in a *differentiable* part and a *non-differentiable* part:

$$R(u) \;=\; R_S(u) + R_{S^c}(u)$$

Let $\mathit{\Pi}$ be a partition of the vertex set V with

$$\mathit{\Pi} \;:=\; \left\{A_i \subset V \mid i \in I = \{1, \ldots, m\},\; V = \dot{\cup}_{i=1}^{m} A_i\right\}$$

Cut Pursuit in a nutshell

Mathematical prerequisites

We decompose $\mathcal{R}(u)$ via a set of edges $S \subset E$ in a *differentiable* part and a *non-differentiable* part:

$$R(u) \;=\; R_S(u) + R_{S^c}(u)$$

Let Π be a partition of the vertex set V with

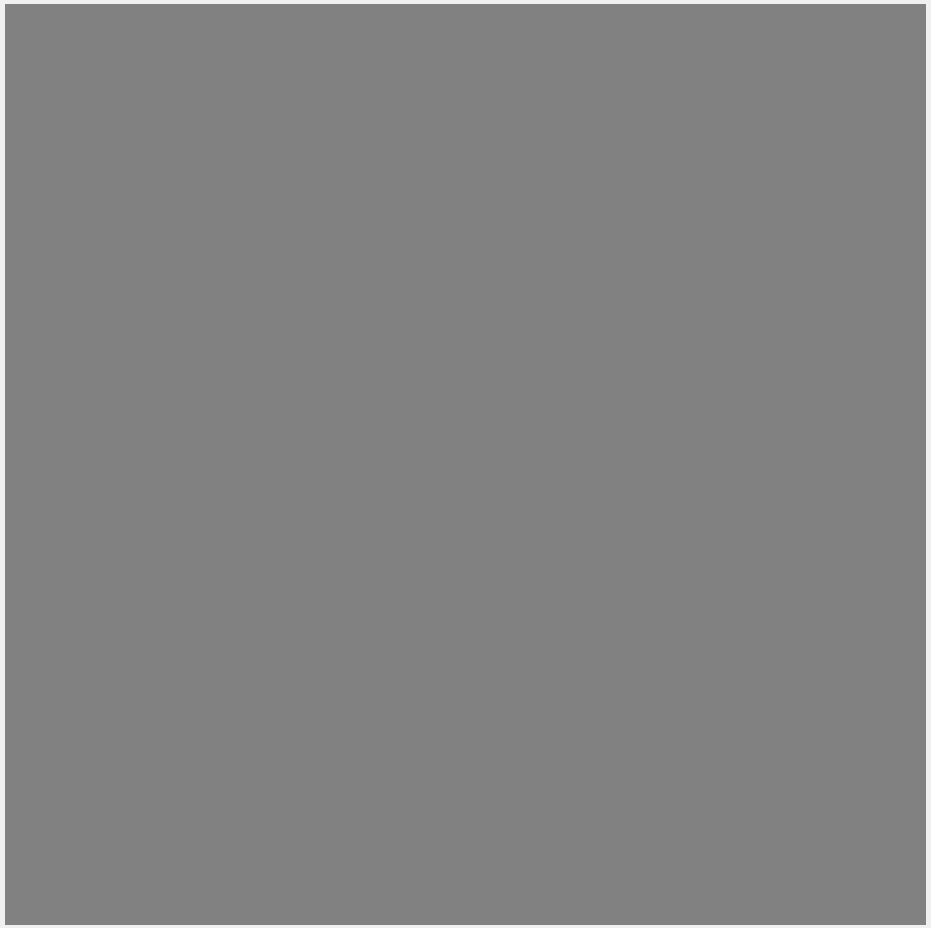
$$\Pi \;:=\; \left\{A_i \subset V \mid i \in I = \{1, \ldots, m\},\; V = \dot{\cup}_{i=1}^{m} A_i\right\}$$

We (essentially) compute a reduced function f_Π based on the partition Π as:

$$f_\Pi(v_i) \;=\; \frac{1}{|A_i|} \sum_{v_j \in A_i} f(v_j) \;\hat{=}\; \mathrm{mean}_{A_i}(f).$$

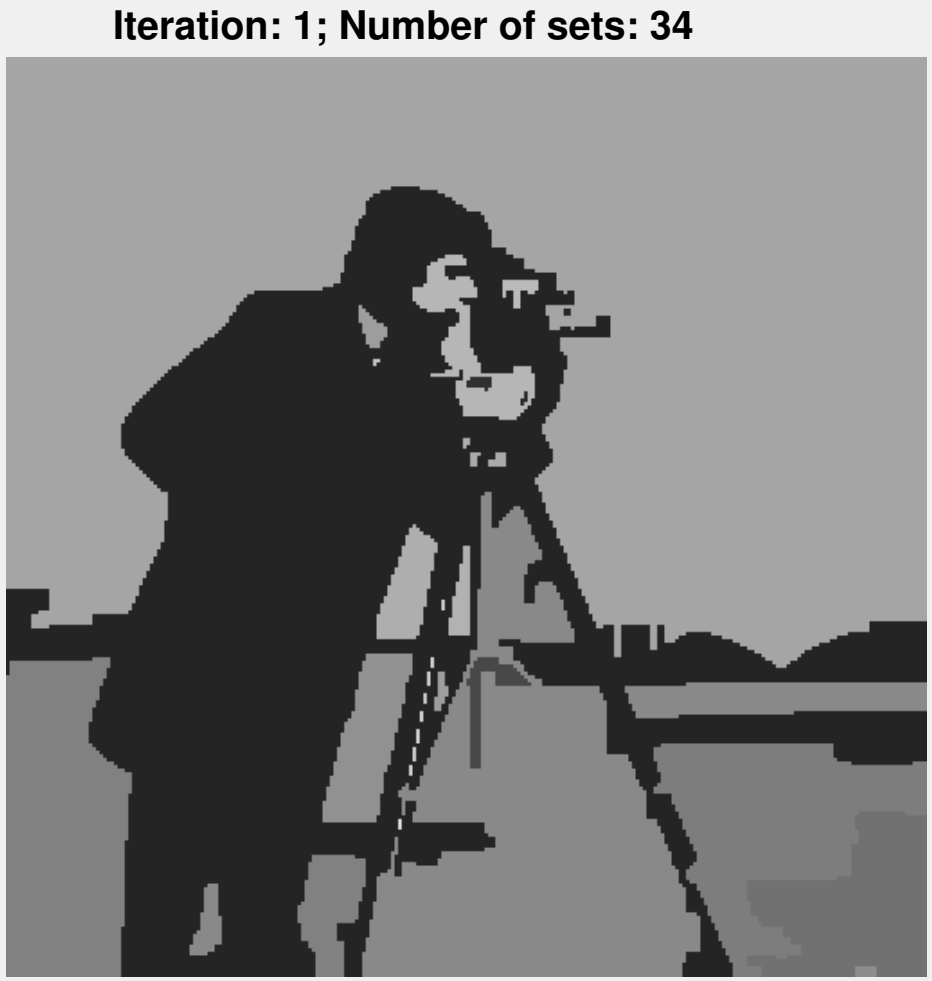
Example: Cut Pursuit for TV image denoising

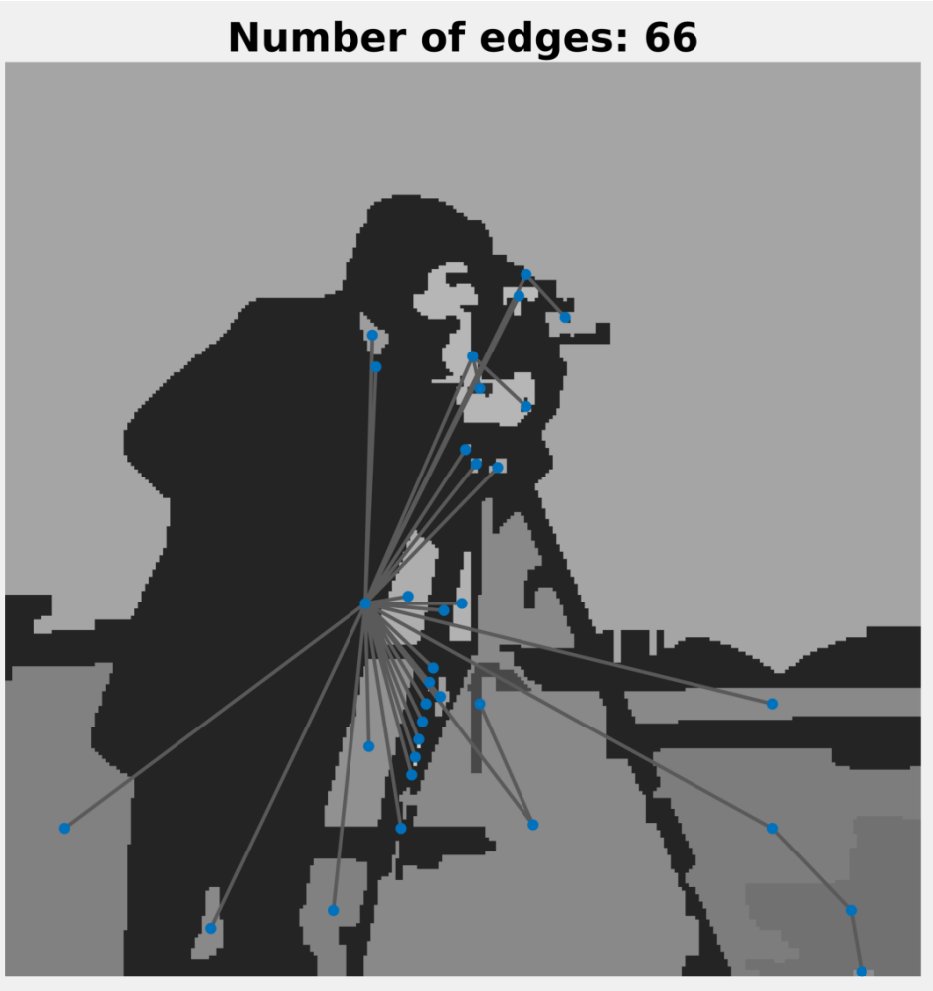
Mean value of image



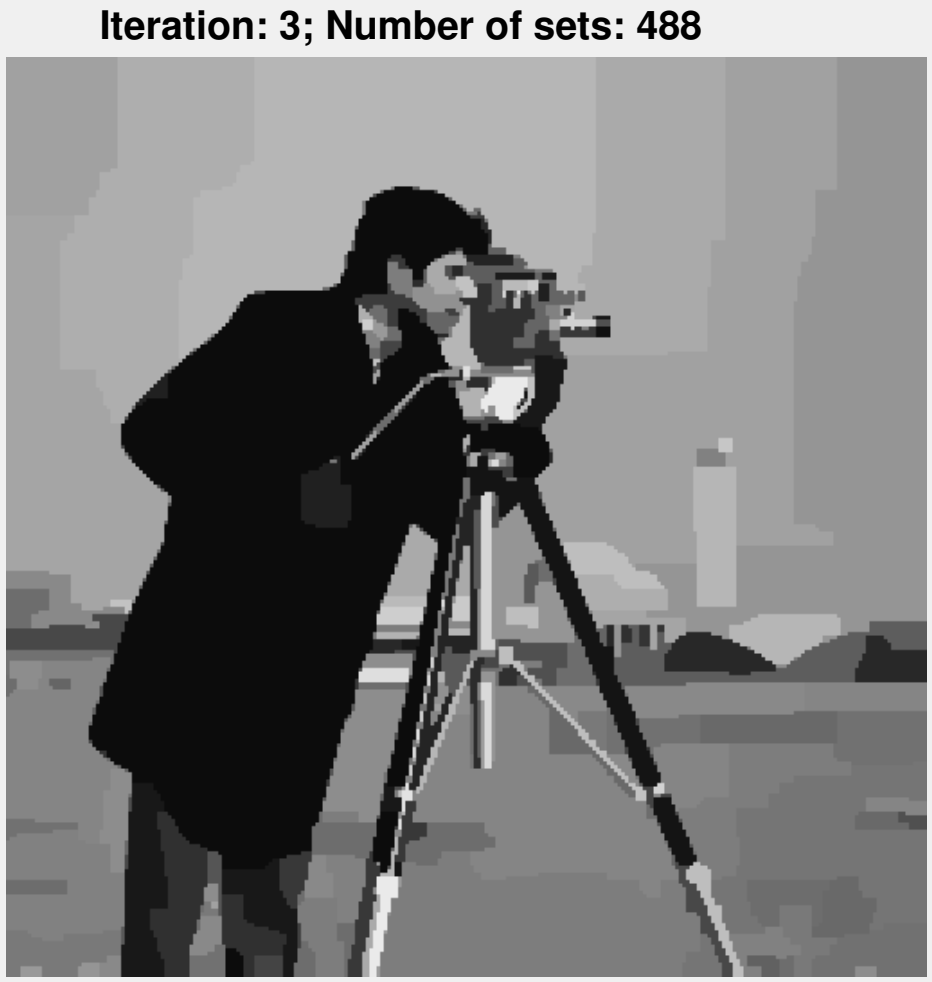
Mean value of image

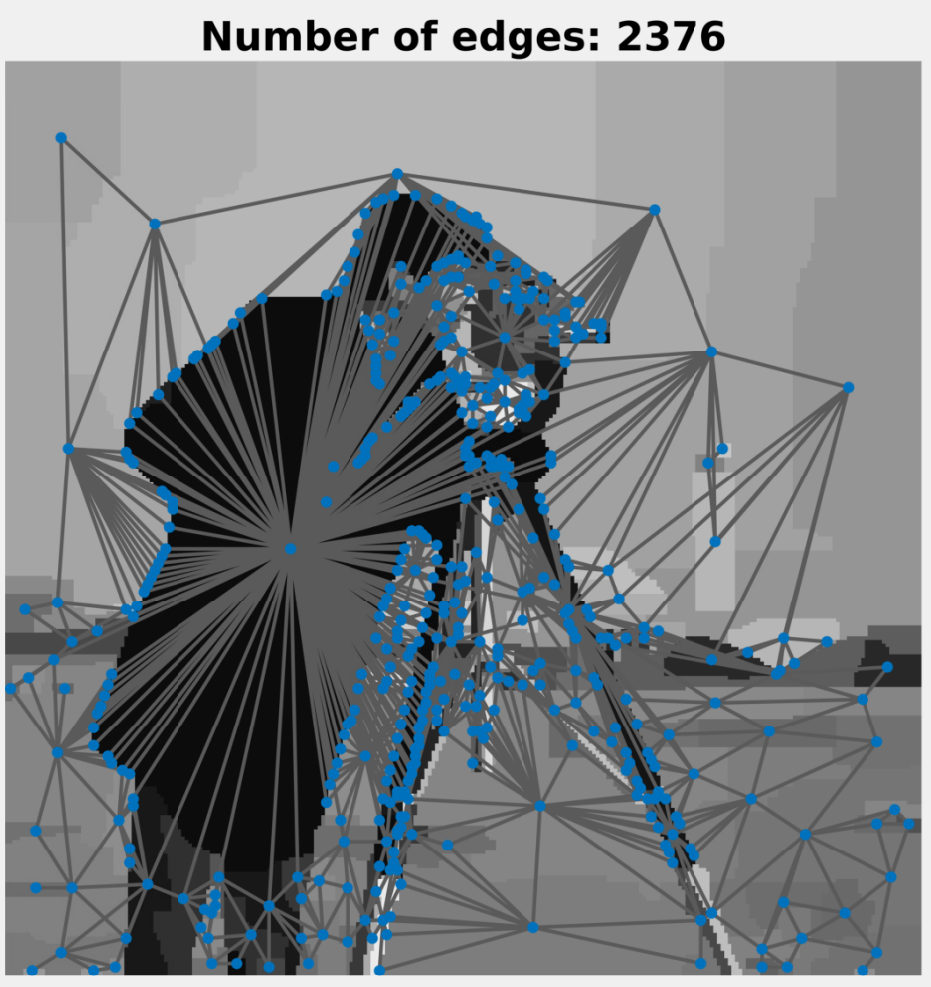
Example: Cut Pursuit for TV image denoising



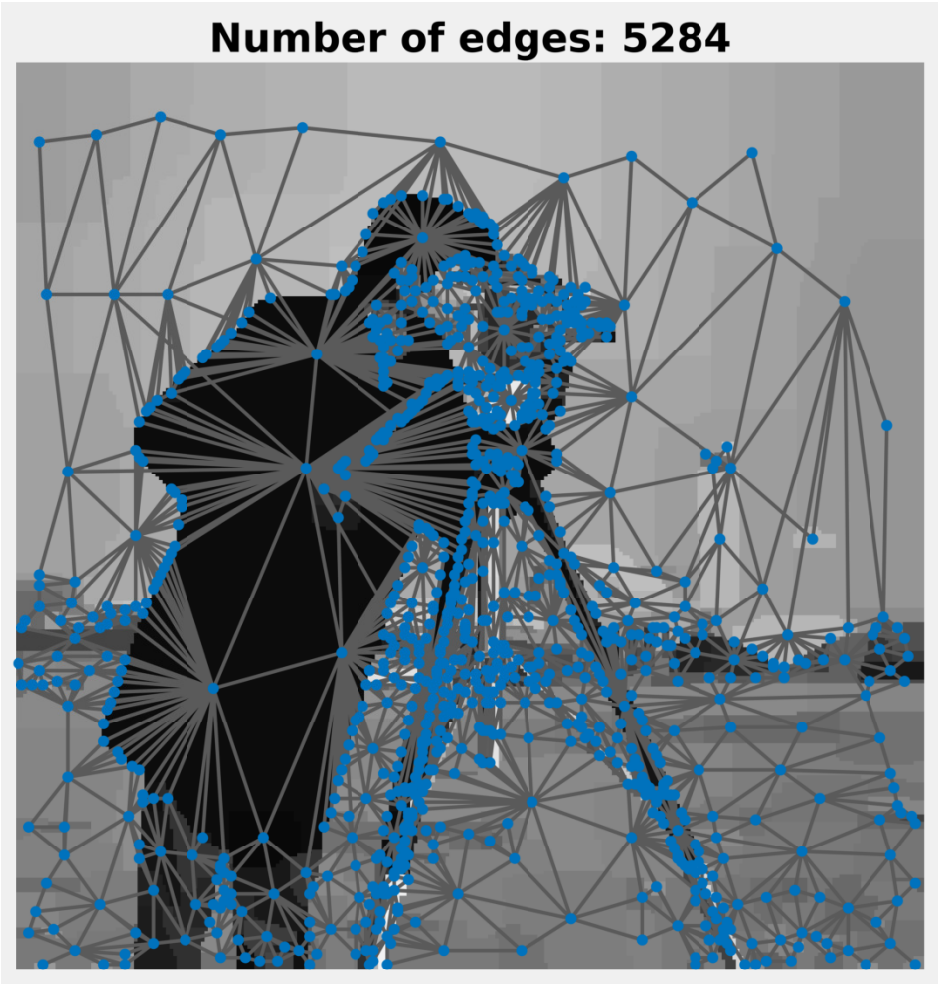


Example: Cut Pursuit for TV image denoising

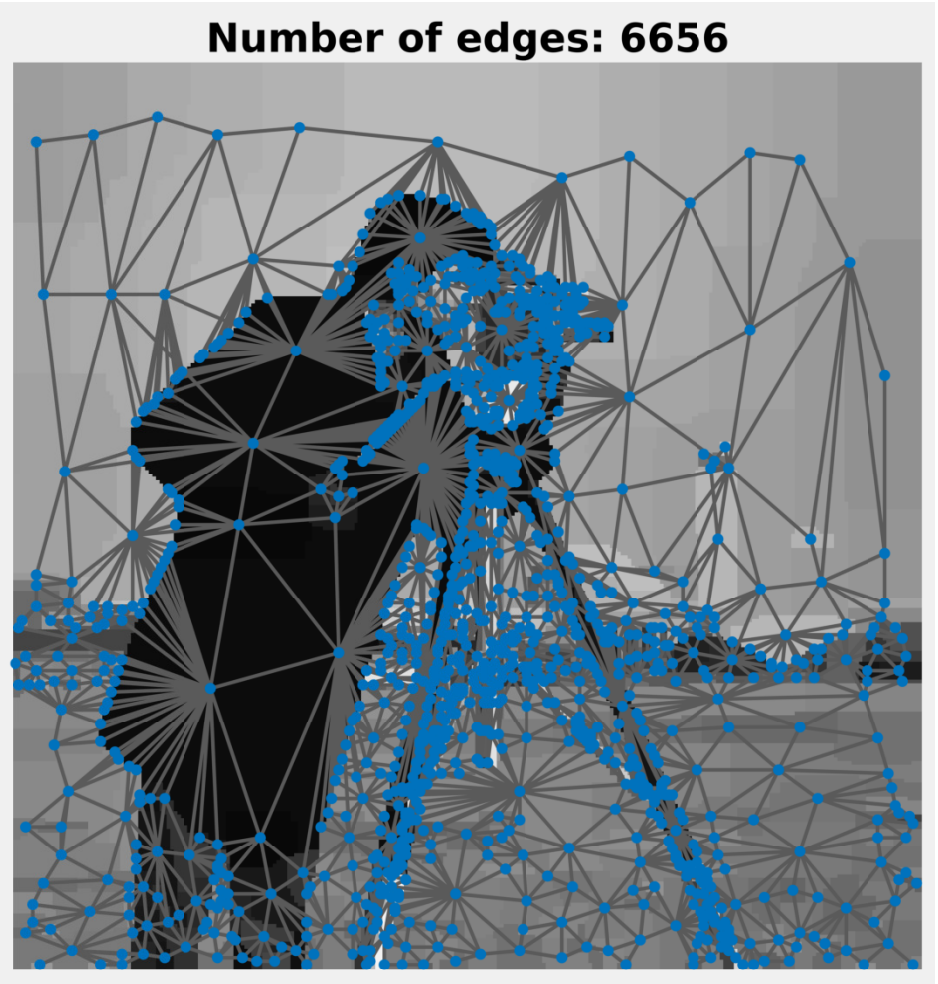




Example: Cut Pursuit for TV image denoising



Example: Cut Pursuit for TV image denoising



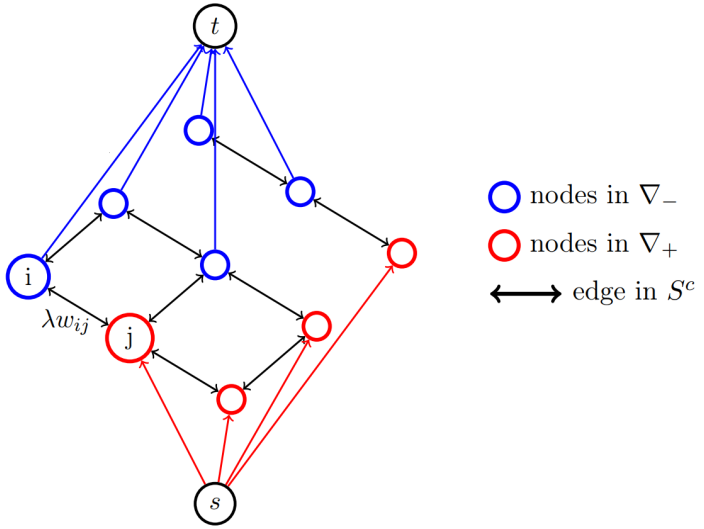
Appendix: Steepest binary cut

Aim:

Compute a finer partition Π with largest possible decrease of J. This is a NP hard problem for which a solution can be computed by a minimum graph cut on a flow graph[5]

G_{flow} with $V_{flow} = V \cup \{s, t\}$ with capacities defined as

$$\begin{cases} c(u,t) = |\nabla J_S(f)_u|, & u \in \nabla^- \\ c(s,u) = \nabla J_S(f)_u, & (u) \in \nabla^+ \\ c(u,v) = \alpha\sqrt{w(u,v)}, & f(u) = f(v) \end{cases}$$



We get the sets disjunct $S, T \subset V$ and set $B = S$ and $B^c = T$.

[5] L. Landrieu, G. Obozinski: Cut pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. SIAM Journal on Imaging Sciences 10(4), 2017

Directional derivative

Let $S(f) = \{(u,v) \in E | \partial_v f(u) \neq 0 \Leftrightarrow f(u) \neq f(v)\}$ differentiable.
Split in differentiable and non-differentiable parts

$$R'(f;\vec{d}) = \langle \nabla_S R(f), \vec{d} \rangle + R'_{S^c}(f;\vec{d})$$

with

$$\nabla R_S(f)(u,v) = \begin{cases} \sqrt{w(u,v)} \frac{f(u)-f(v)}{\|f(u)-f(v)\|_1}, & (u,v) \in S \\ 0, & \text{else} \end{cases}$$

and

$$R'_{S^c}(f;\vec{d}) = \sum_{(u,v)\in S^c} \sqrt{w(u,v)} |\vec{d}(v) - \vec{d}(u)|$$