Grid-Free Denoising of Point-Cloud Data via Non-Local Regularization[1]

Matthew Herman, Tom Goldstein, and Stan Osher

Dept. of Mathematics, UCLA

Workshop on Sparse and Low Rank Approximation

Banff International Research Station

March 7, 2011

[1]Support provided by NGA

Outline

Introduction and Problem Description

- Typical *Light Detection and Ranging* (LiDAR) systems transmit a laser pulse and receive a **deluge of data** in the form of a three-dimensional (3D) **point-cloud**.

Introduction and Problem Description

- Typical *Light Detection and Ranging* (LiDAR) systems transmit a laser pulse and receive a **deluge of data** in the form of a three-dimensional (3D) **point-cloud**.
- This point-cloud contains **important information** reflected from the scene of interest, e.g., a terrain profile.

Introduction and Problem Description

- Typical *Light Detection and Ranging* (LiDAR) systems transmit a laser pulse and receive a **deluge of data** in the form of a three-dimensional (3D) **point-cloud**.
- This point-cloud contains **important information** reflected from the scene of interest, e.g., a terrain profile.
- Unfortunately, there are **significant noise** issues, e.g., due to sensitivity of photodiodes.

Introduction and Problem Description

- Typical *Light Detection and Ranging* (LiDAR) systems transmit a laser pulse and receive a **deluge of data** in the form of a three-dimensional (3D) **point-cloud**.
- This point-cloud contains **important information** reflected from the scene of interest, e.g., a terrain profile.
- Unfortunately, there are **significant noise** issues, e.g., due to sensitivity of photodiodes.
- State-of-the-art 3D surface reconstruction algorithms require the **use of a grid** to determine measures of gradient and how "close" points are.

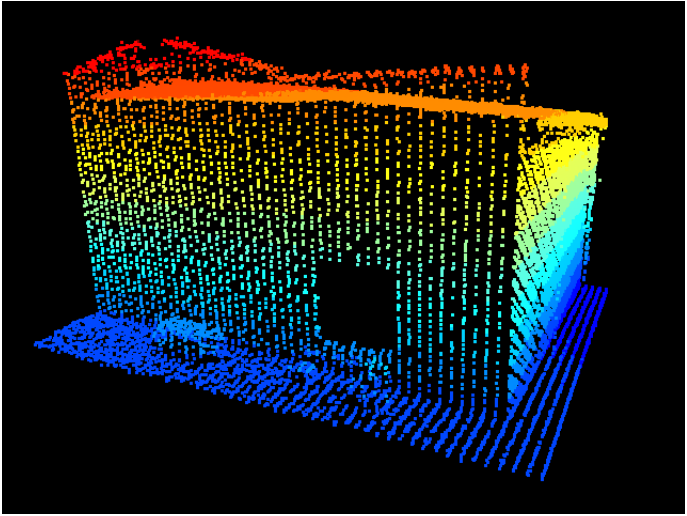
Introduction and Problem Description

- Typical *Light Detection and Ranging* (LiDAR) systems transmit a laser pulse and receive a **deluge of data** in the form of a three-dimensional (3D) **point-cloud**.
- This point-cloud contains **important information** reflected from the scene of interest, e.g., a terrain profile.
- Unfortunately, there are **significant noise** issues, e.g., due to sensitivity of photodiodes.
- State-of-the-art 3D surface reconstruction algorithms require the **use of a grid** to determine measures of gradient and how "close" points are.
- We propose a **3D grid-free** generalization of regularization with **non-local methods**, which is already used extensively to denoise 2D images.

Review of LiDAR

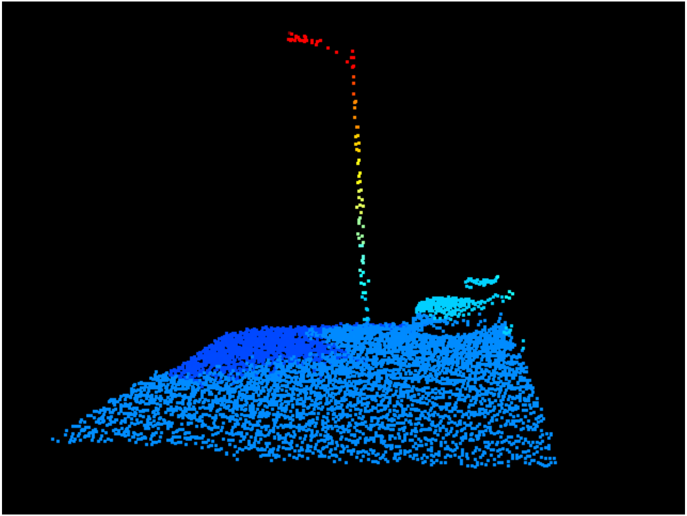
Review of LiDAR
Review of LiDAR
Review of LiDAR

Denoised LiDAR Scene, Low-Altitude (22,120 points)



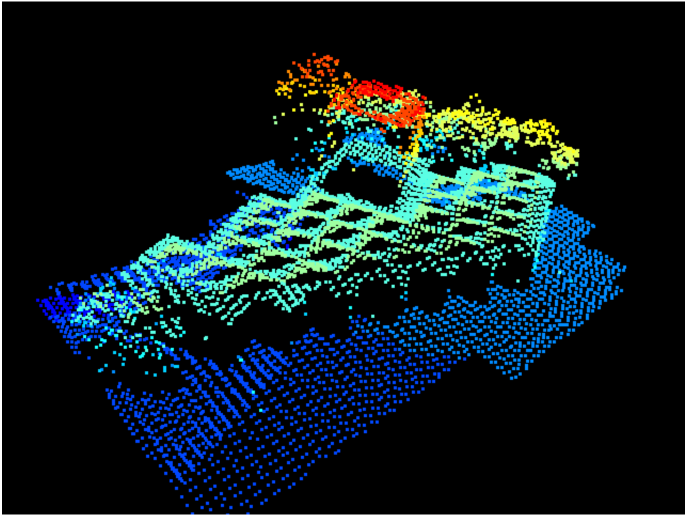
[Courtesy of AFRL/MNG VEAA Data Set #1]

Denoised LiDAR Scene, Low-Altitude (5,928 points)



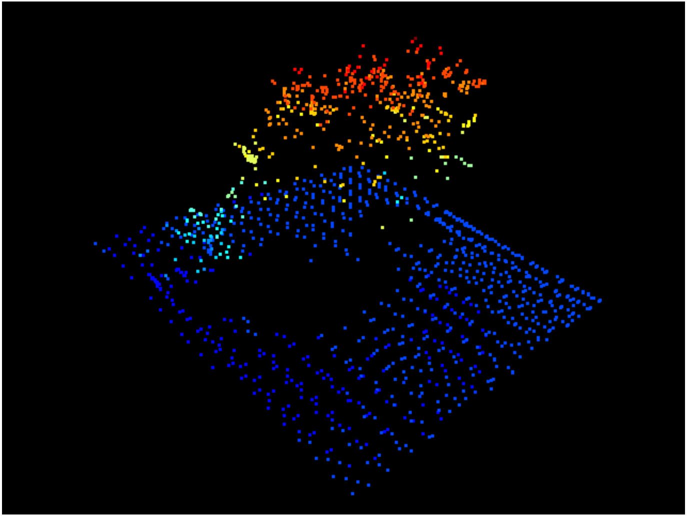
[Courtesy of AFRL/MNG VEAA Data Set #1]

Denoised LiDAR Scene, High-Altitude (10,118 points)



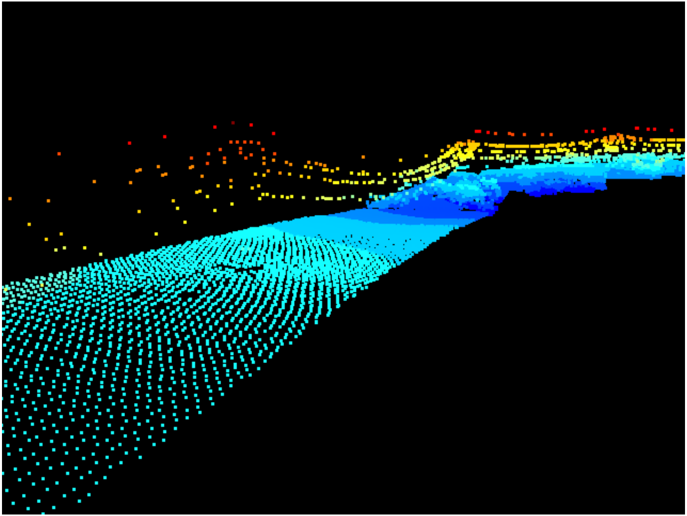
[Courtesy of ITC Data Set]

Denoised LiDAR Scene, High-Altitude (1,467 points)



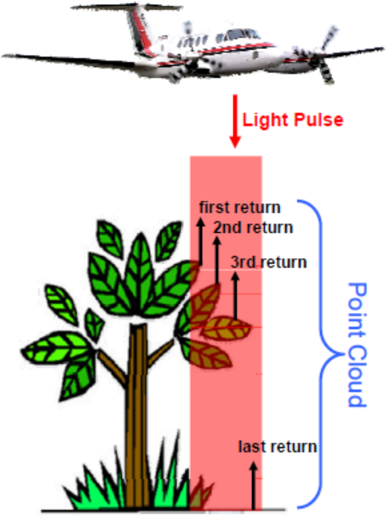
[Courtesy of ITC Data Set]

Denoised LiDAR Scene, High-Altitude (24,942 points)



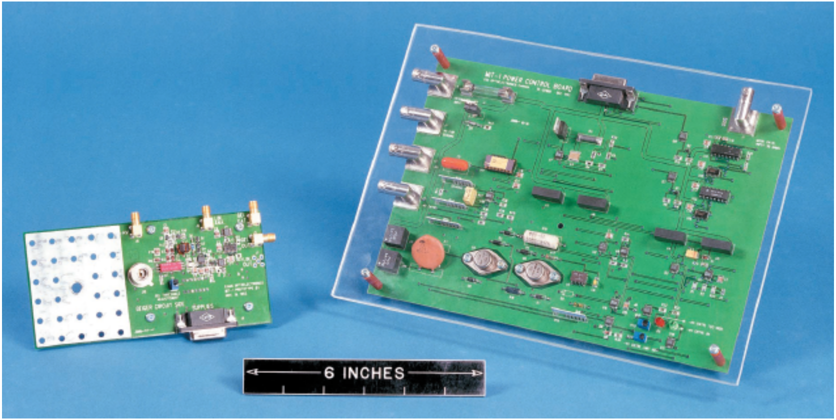
[Courtesy of ITC Data Set]

LiDAR Remote Sensing Data Collection



[Image from presentation by Myron Brown, Johns Hopkins University, Applied Physics Lab. Duke University 2009]

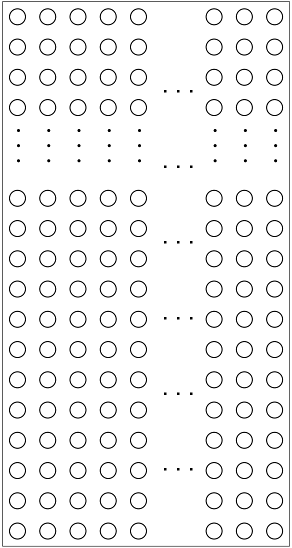
Early Version of a 6×5 GmAPD Detector Array



[Image from "Three-Dimensional Imaging Laser Radars with Geiger-Mode Avalanche Photodiode Arrays",

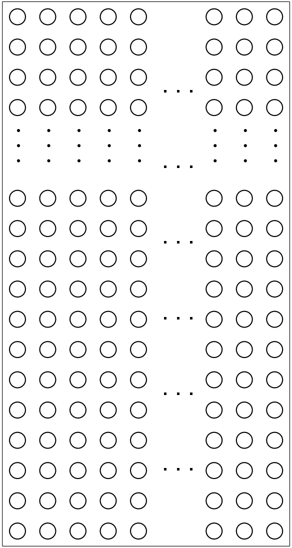
M. Albota *et al.*, *Lincoln Lab Journal* 2002]

More Laser and Detector Array Details



- Current GmAPD Arrays:
  $128 \times 32 =$ **4,096 elements**

More Laser and Detector Array Details



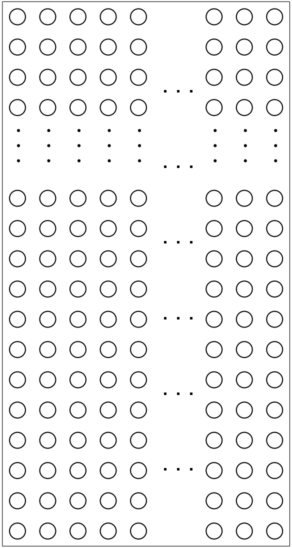
- Current GmAPD Arrays:

  $128 \times 32 =$ **4,096 elements**

- Near-Future GmAPD Arrays:

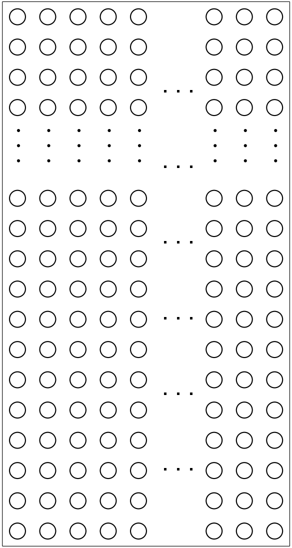
  $256 \times 64 =$ **16,384 elements**

More Laser and Detector Array Details



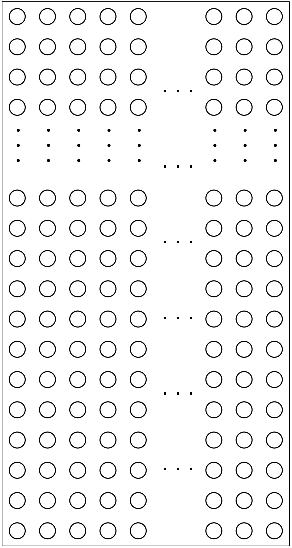
- Current GmAPD Arrays:
  $128 \times 32 =$ **4,096 elements**
- Near-Future GmAPD Arrays:
  $256 \times 64 =$ **16,384 elements**
- Laser pulse rate: **1-20 kHz**

More Laser and Detector Array Details



- Current GmAPD Arrays:
  $128 \times 32 =$ **4,096 elements**
- Near-Future GmAPD Arrays:
  $256 \times 64 =$ **16,384 elements**
- Laser pulse rate: **1-20 kHz**
- Information rate: **Mbps-Gbps**

More Laser and Detector Array Details

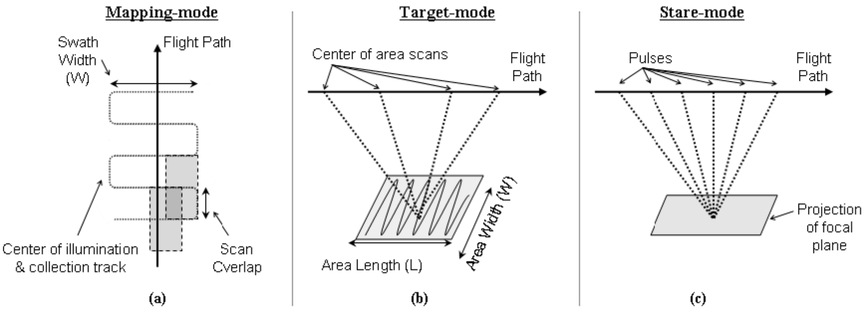


- Current GmAPD Arrays:
  $128 \times 32 =$ **4,096 elements**
- Near-Future GmAPD Arrays:
  $256 \times 64 =$ **16,384 elements**
- Laser pulse rate: **1-20 kHz**
- Information rate: **Mbps-Gbps**

Noise in measurements due to:

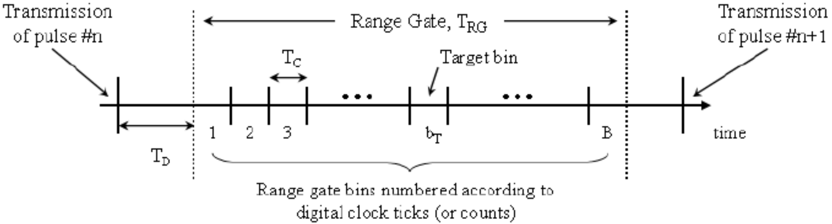
- Dark current, internal thermal energy
- Other light sources (solar, lunar, etc.)
- Cross-talk between elements

LiDAR Data Collection Scan Patterns



[Image from "Redundancy Analysis of Raw Geiger-mode Laser Radar Data", N. Lopez *et al.*, *SPIE* 2010]

Timing Model for Single GmAPD Detector Element



[Image from "Redundancy Analysis of Raw Geiger-mode Laser Radar Data", N. Lopez *et al.*, *SPIE* 2010]

Raw Point Cloud Formulation from GmAPD Counts

For each laser pulse, and each GmAPD element the value of b_T yields the range-to-target distance

$$R_T = \left(\frac{b_T T_C + T_D}{2} \right) \cdot c$$

Raw Point Cloud Formulation from GmAPD Counts

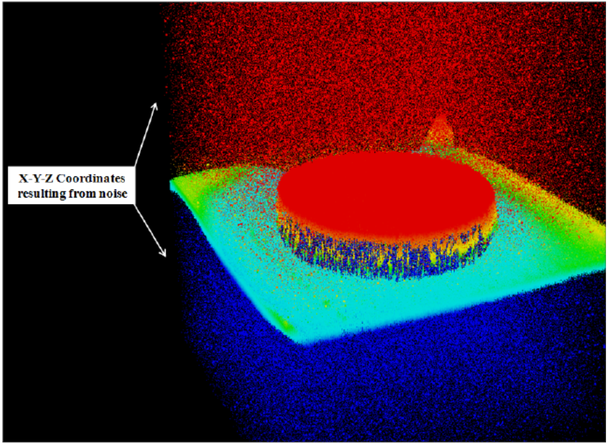
For each laser pulse, and each GmAPD element the value of b_T yields the range-to-target distance

$$R_T = \left(\frac{b_T T_C + T_D}{2} \right) \cdot c$$

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix} + R_T \cdot (M_P M_R M_H)(M_{IT} M_{CT})(M_{row} M_{col}) \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

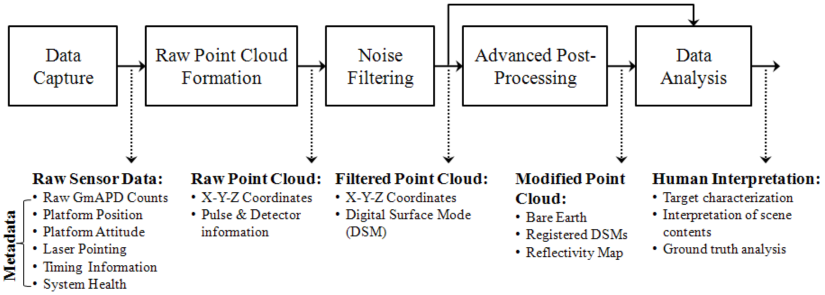
- $[X_P, Y_P, Z_P]$ is the platform position
- M_P, M_R, M_H are rotational matrices for the pitch, roll and heading (platform attitude)
- M_{IT}, M_{CT} are rotational matrices for the in-track and cross-track laser pointing angles
- M_{row}, M_{col} are rotational matrices calculated from angles related to a detector's position from the surface normal

Simulation of Raw LiDAR Data Point Cloud



[Courtesy of Myron Brown, Johns Hopkins University, Applied Physics Lab. Image from "Product Chain Analysis of Three-Dimensional Imaging Laser Radar Systems employing Geiger-mode Avalanche Photodiodes", N. Lopez *et al.*, *SPIE* 2010]

Product Chain for 3D imaging LiDAR Systems

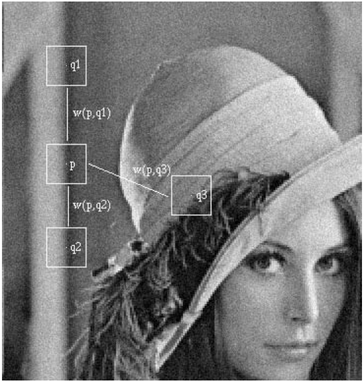


[Image from "Product Chain Analysis of Three-Dimensional Imaging Laser Radar Systems employing Geiger-mode Avalanche Photodiodes", N. Lopez *et al.*, *SPIE* 2010]

Review of 2D Non-Local Ideas
Review of 2D Non-Local Ideas
Review of 2D Non-Local Ideas
Review of 2D Non-Local Ideas

Background: 2D Non-Local Patches

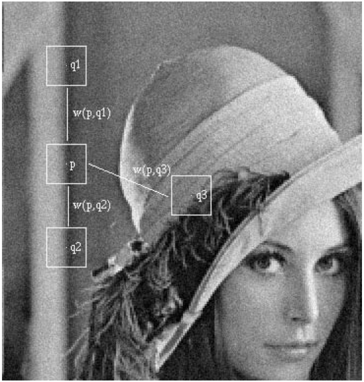
Noisy Image f



[Image from "Image Denoising by Non-Local Averaging", A. Buades, B. Coll, J.-M. Morel, *IEEE ICASSP* 2005]

Background: 2D Non-Local Patches

Noisy Image f

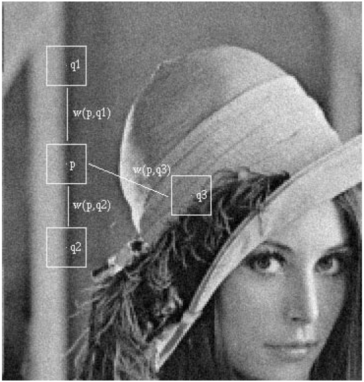


[Image from "Image Denoising by Non-Local Averaging", A. Buades, B. Coll, J.-M. Morel, *IEEE ICASSP* 2005]

Consider **pixels** p, q_1, q_2, q_3 and their respective **patches** $\mathcal{N}_p, \mathcal{N}_{q_1}, \mathcal{N}_{q_2}, \mathcal{N}_{q_3}$.

Background: 2D Non-Local Patches

Noisy Image f



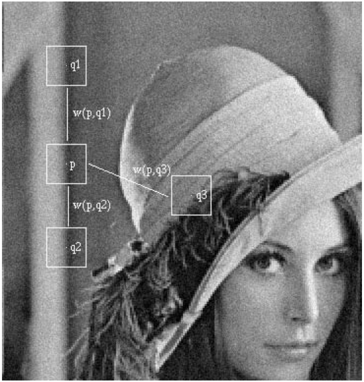
[Image from "Image Denoising by Non-Local Averaging", A. Buades, B. Coll, J.-M. Morel, *IEEE ICASSP* 2005]

Consider **pixels** p, q_1, q_2, q_3 and their respective **patches** $\mathcal{N}_p, \mathcal{N}_{q_1}, \mathcal{N}_{q_2}, \mathcal{N}_{q_3}$.

Patches $\mathcal{N}_p$, $\mathcal{N}_{q_1}$, and $\mathcal{N}_{q_2}$ look similar, while $\mathcal{N}_p$, $\mathcal{N}_{q_3}$ are quite different.

Background: 2D Non-Local Patches

Noisy Image f



[Image from "Image Denoising by Non-Local Averaging", A. Buades, B. Coll, J.-M. Morel, *IEEE ICASSP* 2005]

Consider **pixels** p, q_1, q_2, q_3 and their respective **patches** $\mathcal{N}_p, \mathcal{N}_{q_1}, \mathcal{N}_{q_2}, \mathcal{N}_{q_3}$.

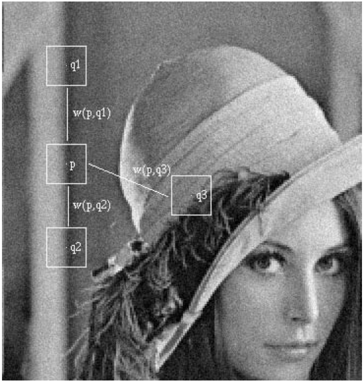
Patches $\mathcal{N}_p$, $\mathcal{N}_{q_1}$, and $\mathcal{N}_{q_2}$ look similar, while $\mathcal{N}_p$, $\mathcal{N}_{q_3}$ are quite different.

The **weight** between pixels p, q is

$$w_f(p,q) := \frac{1}{Z_p} \mathrm{e}^{-\frac{\|f(\mathcal{N}_q) - f(\mathcal{N}_p)\|_2^2}{h^2}}$$

with normalization factor Z_p.

Noisy Image f



[Image from "Image Denoising by Non-Local Averaging", A. Buades, B. Coll, J.-M. Morel, *IEEE ICASSP* 2005]

Consider **pixels** p, q_1, q_2, q_3 and their respective **patches** $\mathcal{N}_p, \mathcal{N}_{q_1}, \mathcal{N}_{q_2}, \mathcal{N}_{q_3}$.

Patches $\mathcal{N}_p$, $\mathcal{N}_{q_1}$, and $\mathcal{N}_{q_2}$ look similar, while $\mathcal{N}_p$, $\mathcal{N}_{q_3}$ are quite different.

The **weight** between pixels p, q is

$$w_f(p, q) := \frac{1}{Z_p} \mathrm{e}^{-\frac{\|f(\mathcal{N}_q) - f(\mathcal{N}_p)\|_2^2}{h^2}}$$

with normalization factor Z_p.

Thus, the weights $w(p, q_1)$ and $w(p, q_2)$ are relatively large compared to $w(p, q_3)$.

Background: Non-Local Total Variation Regularization

- Great deal of successful research using **Non-Local Total Variation** (NL-TV) to denoise 2D images.

Background: Non-Local Total Variation Regularization

- Great deal of successful research using **Non-Local Total Variation** (NL-TV) to denoise 2D images.
- For **noisy image** f, **NL-TVL2** solves

$$\min_u \left(\int \mu |\nabla_w u| + \frac{1}{2} \|u - f\|_2^2 \right),$$

and **NL-TVL1** solves

$$\min_u \left(\int \mu |\nabla_w u| + \|u - f\|_1 \right),$$

Background: Non-Local Total Variation Regularization

- Great deal of successful research using **Non-Local Total Variation** (NL-TV) to denoise 2D images.
- For **noisy image** f, **NL-TVL2** solves

$$\min_u \left(\int \mu |\nabla_w u| + \frac{1}{2}\|u - f\|_2^2 \right),$$

and **NL-TVL1** solves

$$\min_u \left(\int \mu |\nabla_w u| + \|u - f\|_1 \right),$$

where

$$\Big(\nabla_w u\Big)(p, q) \;:=\; \Big(u(q) - u(p)\Big)\sqrt{w_f(p, q)}.$$

Background: Non-Local Total Variation Regularization

- Great deal of successful research using **Non-Local Total Variation** (NL-TV) to denoise 2D images.
- For **noisy image** f, **NL-TVL2** solves

$$\min_u \left(\int \mu |\nabla_w u| + \frac{1}{2}\|u - f\|_2^2 \right),$$

and **NL-TVL1** solves

$$\min_u \left(\int \mu |\nabla_w u| + \|u - f\|_1 \right),$$

where

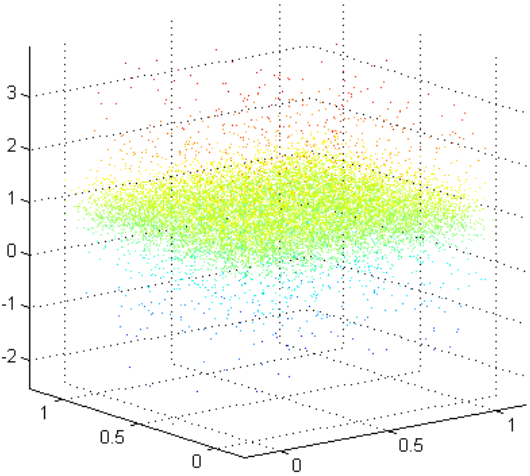
$$\Big(\nabla_w u\Big)(p,q) \;:=\; \Big(u(q) - u(p)\Big)\sqrt{w_f(p,q)}.$$

- **We want to use NL-TV to denoise point cloud data**!!!

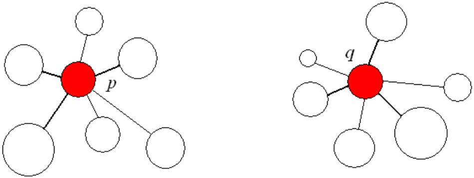
Non-Local Ideas in 3D
Non-Local Ideas in 3D
Non-Local Ideas in 3D
Non-Local Ideas in 3D

How to apply the NL concept to noisy 3D Cloud Data?

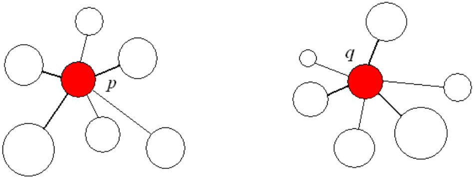
Simulated point cloud data of a noisy unit-plain



New Idea: "Patch Clouds" around points p and q

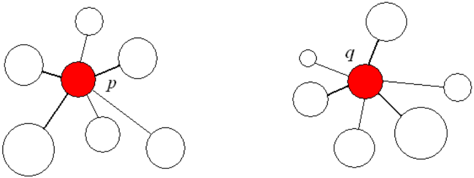


New Idea: "Patch Clouds" around points p and q



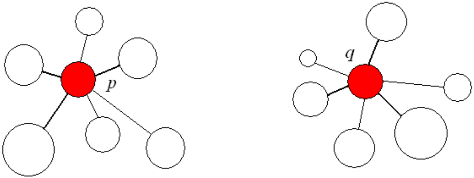
- We want to use the unique geometry of the individual patch clouds to **identify specific structures**.

New Idea: “Patch Clouds” around points p and q



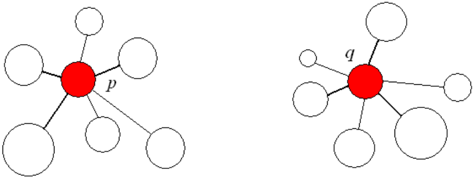
- We want to use the unique geometry of the individual patch clouds to **identify specific structures**.
- Each patch cloud contain its k **nearest-neighbors**.

New Idea: "Patch Clouds" around points p and q



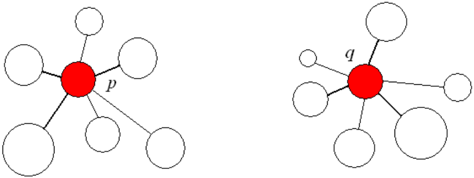
- We want to use the unique geometry of the individual patch clouds to **identify specific structures**.
- Each patch cloud contain its k **nearest-neighbors**.
- However, forcing 3D data onto a grid will lose **subtle geometric information** of the cloud.

New Idea: "Patch Clouds" around points p and q



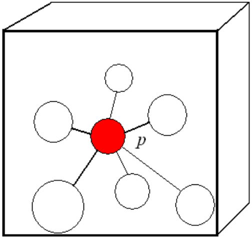
- We want to use the unique geometry of the individual patch clouds to **identify specific structures**.
- Each patch cloud contain its k **nearest-neighbors**.
- However, forcing 3D data onto a grid will lose **subtle geometric information** of the cloud.
- Moreover, in the **3D scenario** we need to find a **vector location** for each data point p in the denoised cloud.

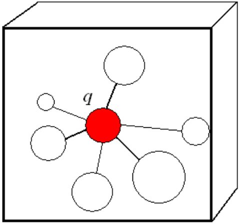
New Idea: "Patch Clouds" around points p and q



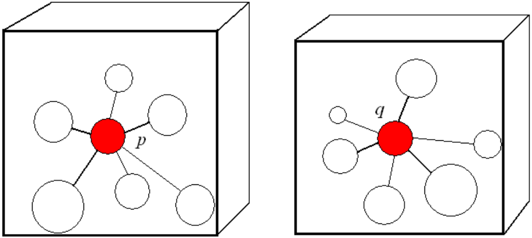
- We want to use the unique geometry of the individual patch clouds to **identify specific structures**.
- Each patch cloud contain its k **nearest-neighbors**.
- However, forcing 3D data onto a grid will lose **subtle geometric information** of the cloud.
- Moreover, in the **3D scenario** we need to find a **vector location** for each data point p in the denoised cloud.
- But if no grid, then what/where are the **reference points**?!?!

“Centered” and Normalized Patch Clouds



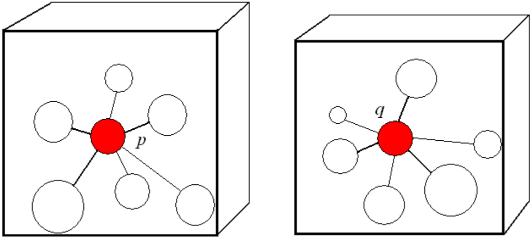


"Centered" and Normalized Patch Clouds



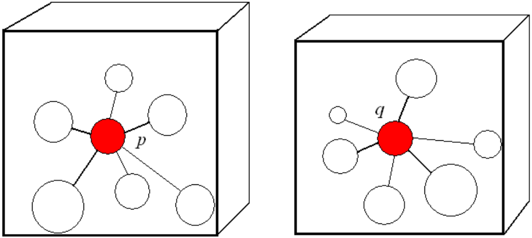
- First, **center** the patch cloud around point p by subtracting the **centroid** $\bar{p}$ from each of its points.

"Centered" and Normalized Patch Clouds



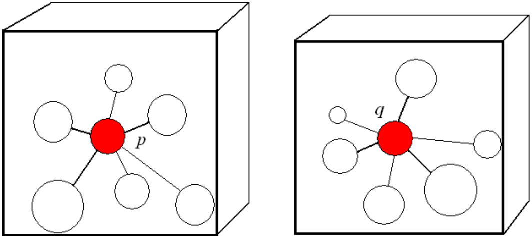
- First, **center** the patch cloud around point p by subtracting the **centroid** $\bar{p}$ from each of its points.
- In particular, the **characteristic** for point p is $\mathcal{C}_f(p) := p - \bar{p}$.

"Centered" and Normalized Patch Clouds



- First, **center** the patch cloud around point p by subtracting the **centroid** $\bar{p}$ from each of its points.
- In particular, the **characteristic** for point p is $\mathcal{C}_f(p) := p - \bar{p}$.
- The *characteristic* for a point in a 3D data cloud is **analogous** to the *intensity* of a pixel in a 2D image.

"Centered" and Normalized Patch Clouds



- First, **center** the patch cloud around point p by subtracting the **centroid** $\bar{p}$ from each of its points.
- In particular, the **characteristic** for point p is $\mathcal{C}_f(p) := p - \bar{p}$.
- The *characteristic* for a point in a 3D data cloud is **analogous** to the *intensity* of a pixel in a 2D image.
- Next, **normalize** by dividing by the largest component (i.e., $\max\{|x|_{\max}, |y|_{\max}, |z|_{\max}\}$) in the patch cloud.

How to Compare Geometries of Patch Clouds?

- For each corner of the centered-normalized cube, find the Euclidean distance to the **nearest point**.

How to Compare Geometries of Patch Clouds?

- For each corner of the centered-normalized cube, find the Euclidean distance to the **nearest point**.
- The function $\mathcal{G}(p) \in \mathbb{R}_+^8$ **encodes the geometry of the patch cloud** associated with point p.

How to Compare Geometries of Patch Clouds?

- For each corner of the centered-normalized cube, find the Euclidean distance to the **nearest point**.
- The function $\mathcal{G}(p) \in \mathbb{R}^8_+$ **encodes the geometry of the patch cloud** associated with point p.
  (Question: Is there a better way to do this?)

How to Compare Geometries of Patch Clouds?

- For each corner of the centered-normalized cube, find the Euclidean distance to the **nearest point**.
- The function $\mathcal{G}(p) \in \mathbb{R}_+^8$ **encodes the geometry of the patch cloud** associated with point p.
  (Question: Is there a better way to do this?)
- Now the weight between points p and q in point cloud f is

$$w_f(p, q) := \frac{1}{Z_p} \mathrm{e}^{-\frac{\|\mathcal{G}_f(q) - \mathcal{G}_f(p)\|_2^2}{h^2}}.$$

How to Compare Geometries of Patch Clouds?

- For each corner of the centered-normalized cube, find the Euclidean distance to the **nearest point**.
- The function $\mathcal{G}(p) \in \mathbb{R}^8_+$ **encodes the geometry of the patch cloud** associated with point p.
  (Question: Is there a better way to do this?)
- Now the weight between points p and q in point cloud f is

$$w_f(p,q) := \frac{1}{Z_p} \mathrm{e}^{-\frac{\|\mathcal{G}_f(q)-\mathcal{G}_f(p)\|_2^2}{h^2}}.$$

- Note: Number of points in each cloud need **not** be the same.
  (Consider a noisy plain with different densities of points.)

- Now we can apply the NL regularization techniques w.r.t. the **characteristic** $\mathcal{C}$ and **geometry** $\mathcal{G}$ of the patch clouds.

Grid-Free Denoising of Point Clouds via Non-Local Reg.

- Now we can apply the NL regularization techniques w.r.t. the **characteristic** $\mathcal{C}$ and **geometry** $\mathcal{G}$ of the patch clouds.
- Given **noisy point cloud** f, **NL-TVL2** solves:

$$\min_u \left(\int \mu |\nabla_w u| + \frac{1}{2} \|u - f\|_2^2 \right),$$

and **NL-TVL1** solves:

$$\min_u \left(\int \mu |\nabla_w u| + \|u - f\|_1 \right),$$

Grid-Free Denoising of Point Clouds via Non-Local Reg.

- Now we can apply the NL regularization techniques w.r.t. the **characteristic** $\mathcal{C}$ and **geometry** $\mathcal{G}$ of the patch clouds.
- Given **noisy point cloud** f, **NL-TVL2** solves:

$$\min_u \left(\int \mu |\nabla_w u| + \frac{1}{2} \|u - f\|_2^2 \right),$$

and **NL-TVL1** solves:

$$\min_u \left(\int \mu |\nabla_w u| + \|u - f\|_1 \right),$$

where

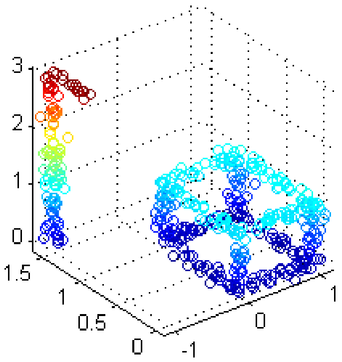
$$\Big(\nabla_w u\Big)(p, q) \ := \ \Big(\mathcal{C}_u(q) - \mathcal{C}_u(p)\Big) \sqrt{w_f(p, q)}.$$

Numerical Simulations

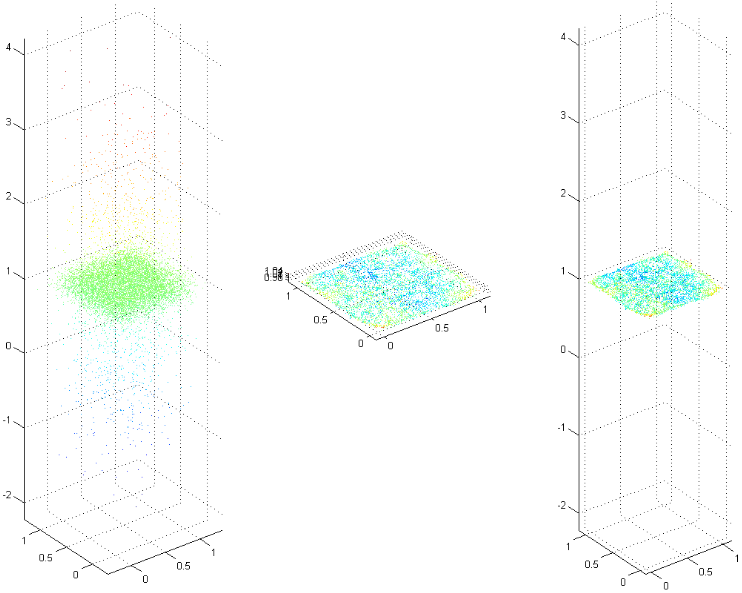
Numerical Simulations
Numerical Simulations
Numerical Simulations

Numerical Simulations

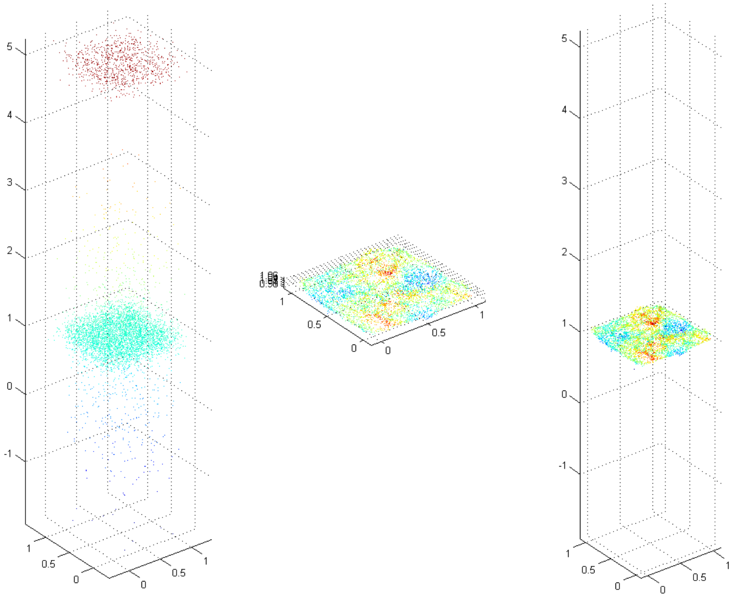
Using NL-TVL2



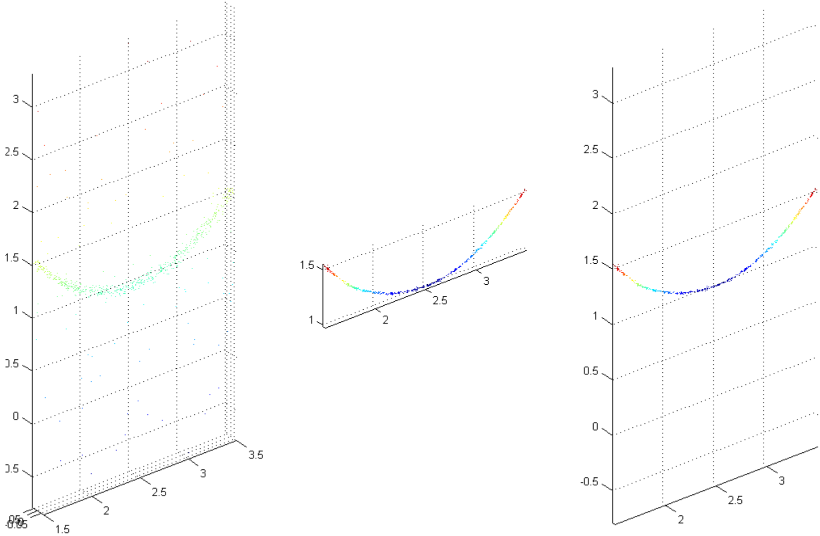
Left: Noisy Unit Plain   Center, Right: NL-TVL1 Recovery



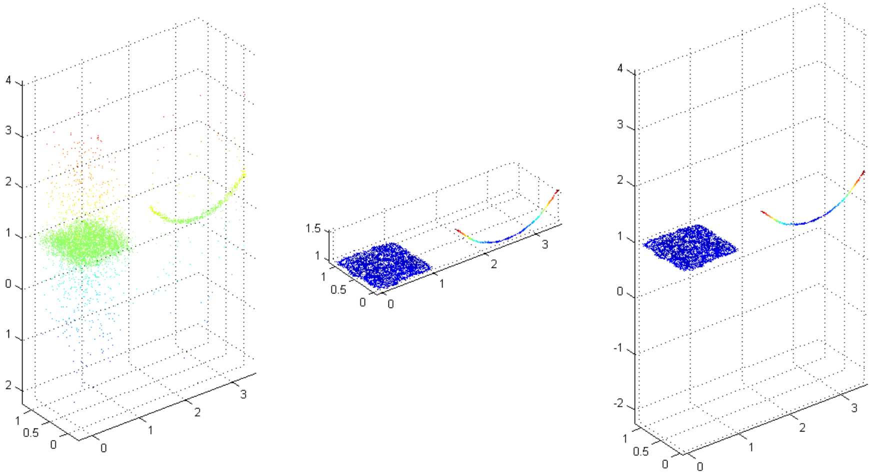
Left: Noisy Unit Plain with "Fog"   Center, Right: NL-TVL1



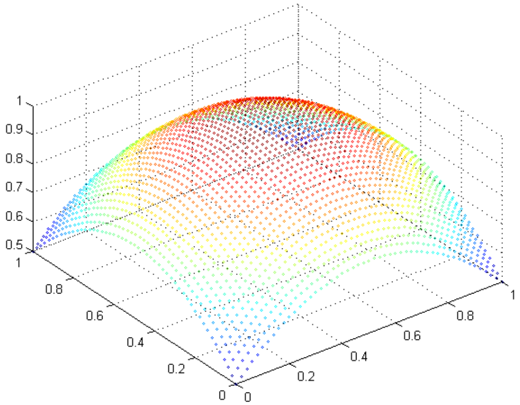
Left: Noisy Catenary Cable Center, Right: NL-TVL1



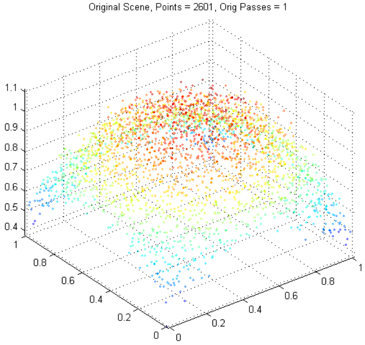
Left: Noisy Unit Plain with Cable   Center, Right: NL-TVL1

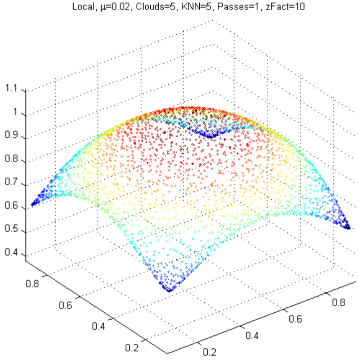


Clean and Gridded Hemisphere

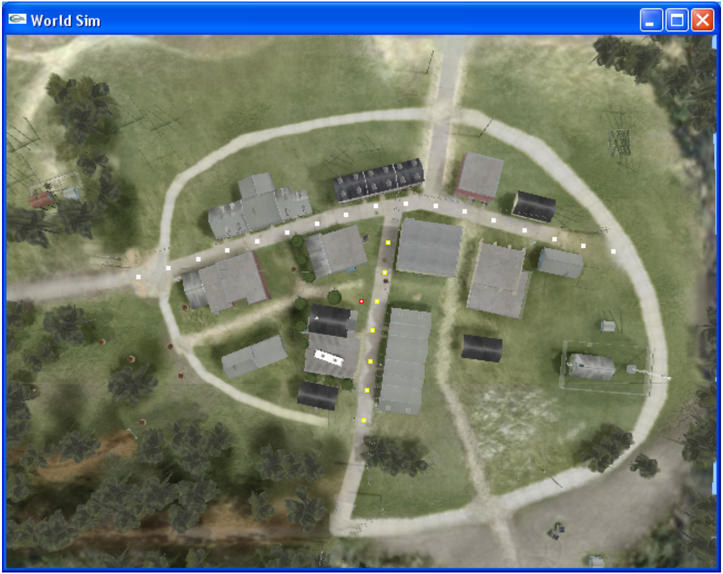


Left: Noisy Hemisphere   Right: NL-TVL1

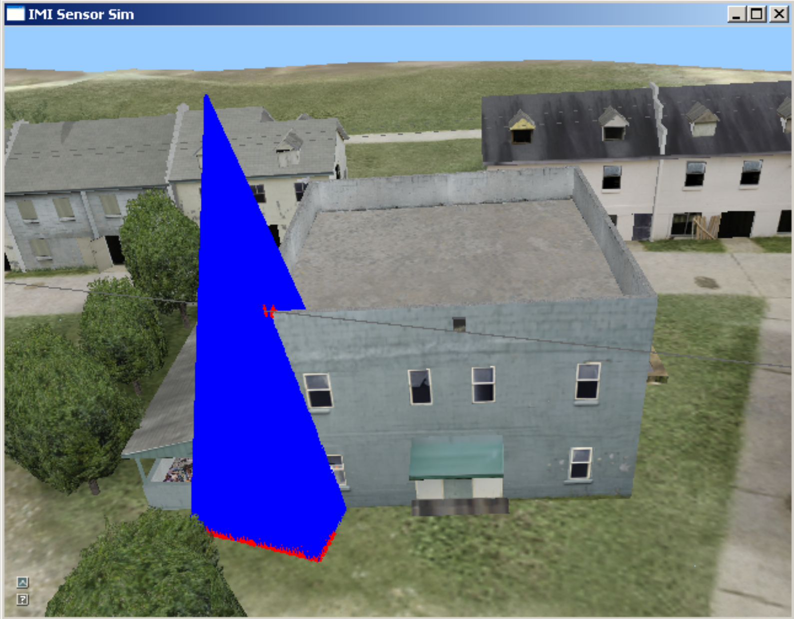


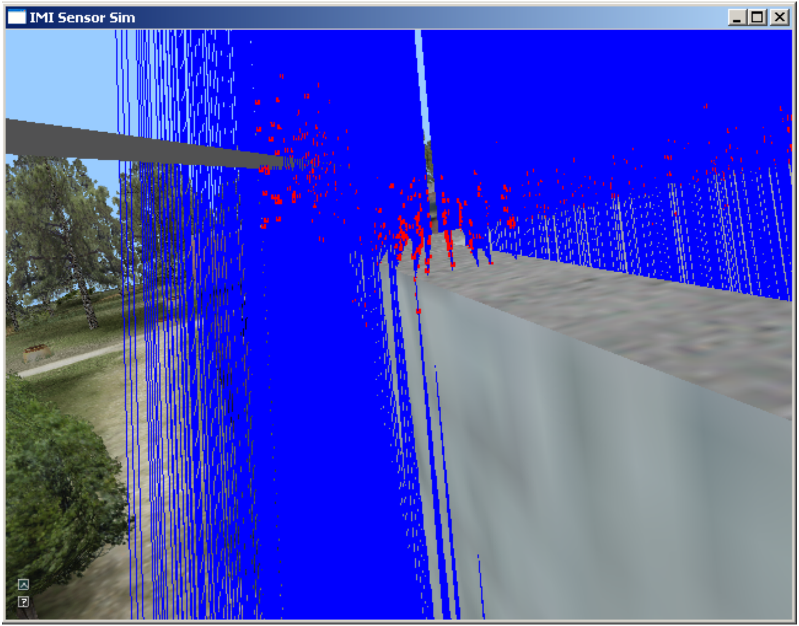


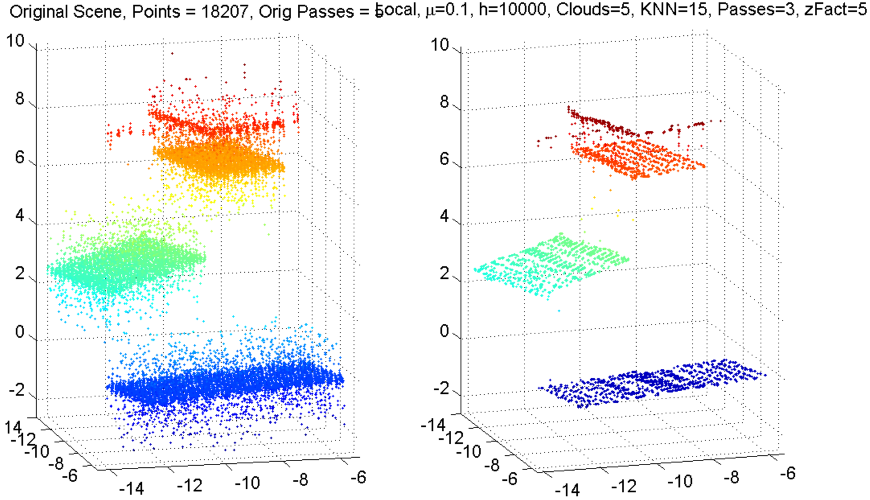
Simulated “Real” LiDAR Scene

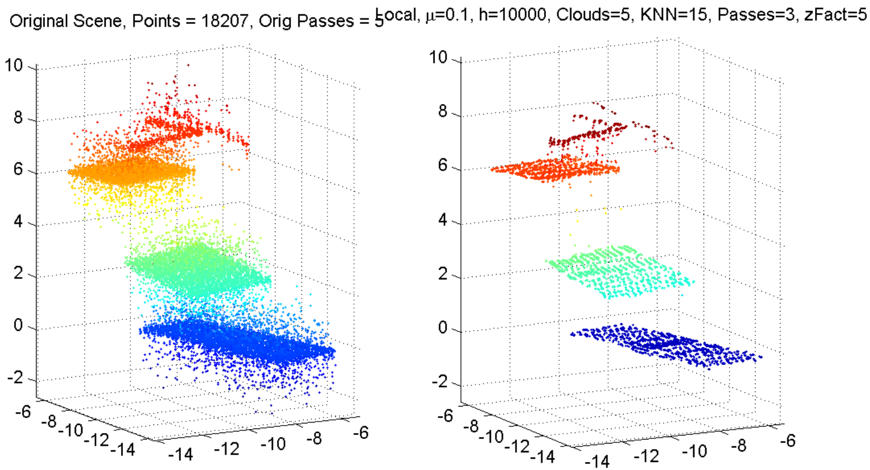


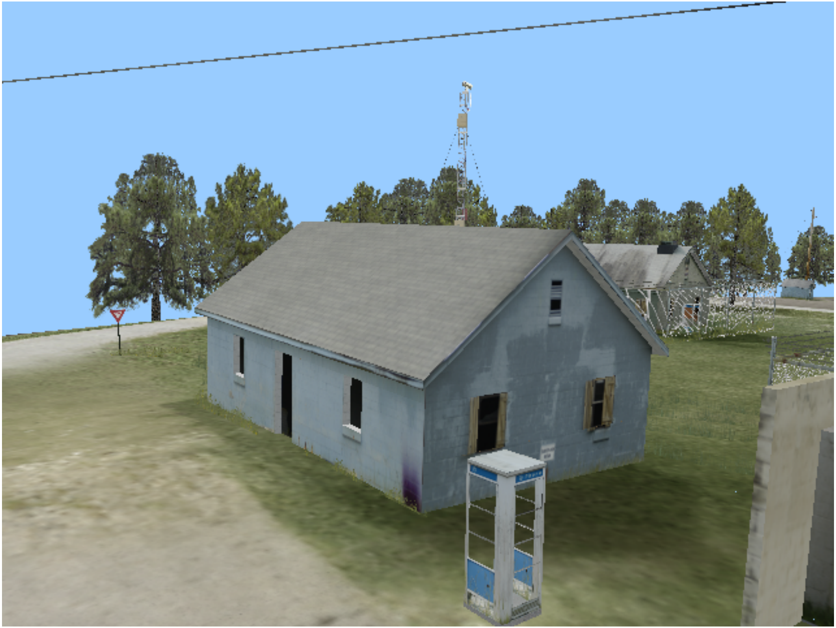

IMI Sensor Sim

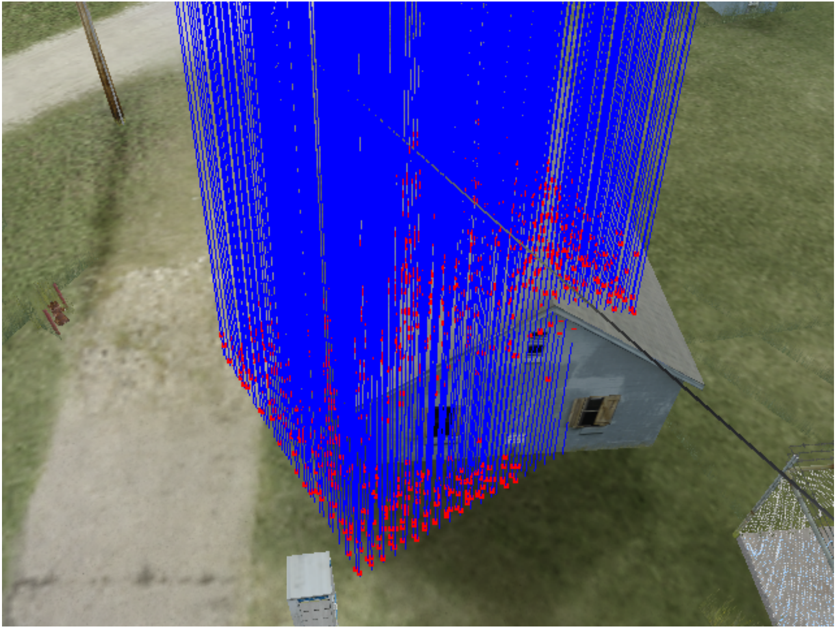

IMI Sensor Sim

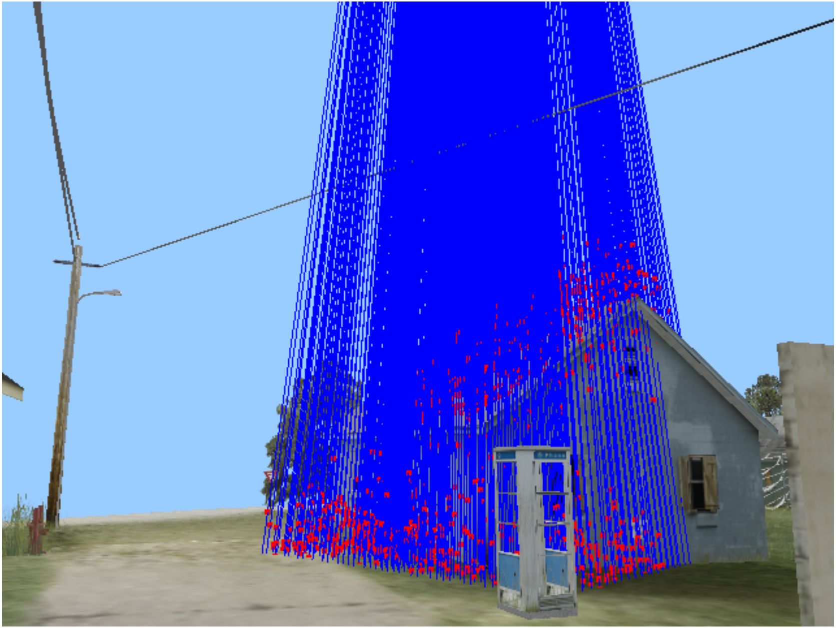

IMI Sensor Sim

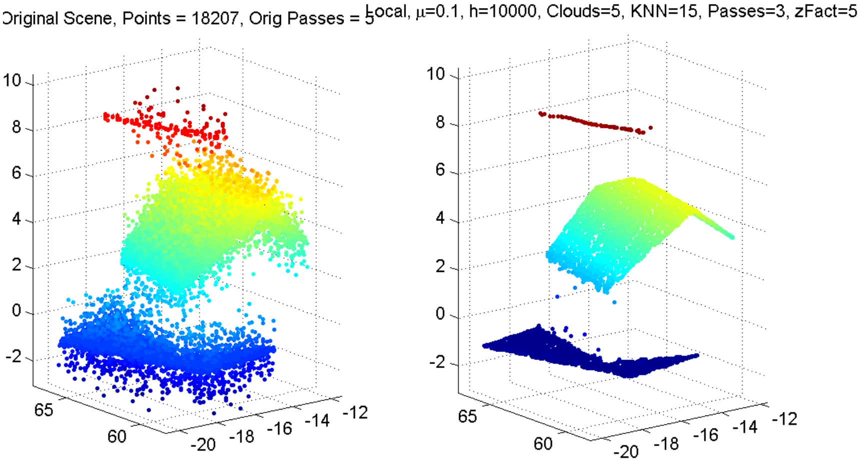

Original Scene, Points = 18207, Orig Passes = 5
Local, μ=0.1, h=10000, Clouds=5, KNN=15, Passes=3, zFact=5

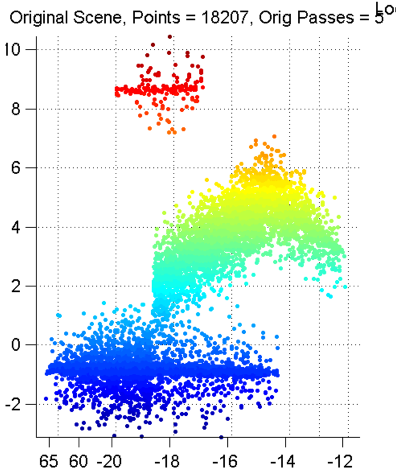

Original Scene, Points = 18207, Orig Passes = 5
Local, μ=0.1, h=10000, Clouds=5, KNN=15, Passes=3, zFact=5
10
8
6
4
2
0
-2
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6



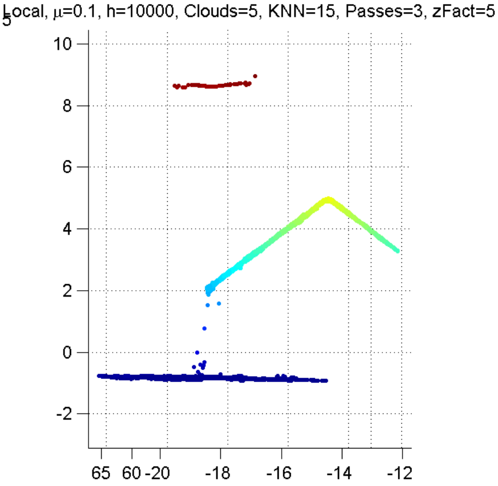




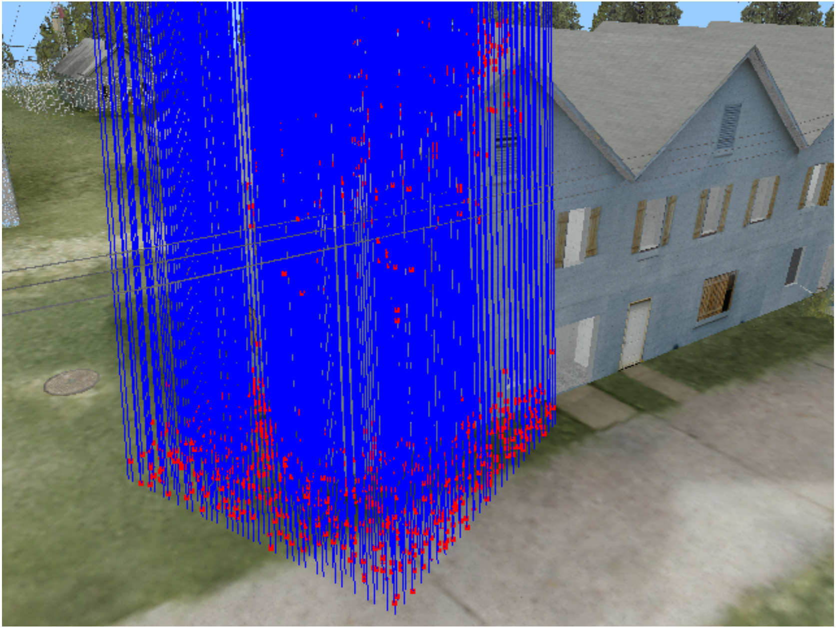

Original Scene, Points = 18207, Orig Passes = 5
Local, μ=0.1, h=10000, Clouds=5, KNN=15, Passes=3, zFact=5
10
8
6
4
2
0
-2
65
60
-20
-18
-16
-14
-12

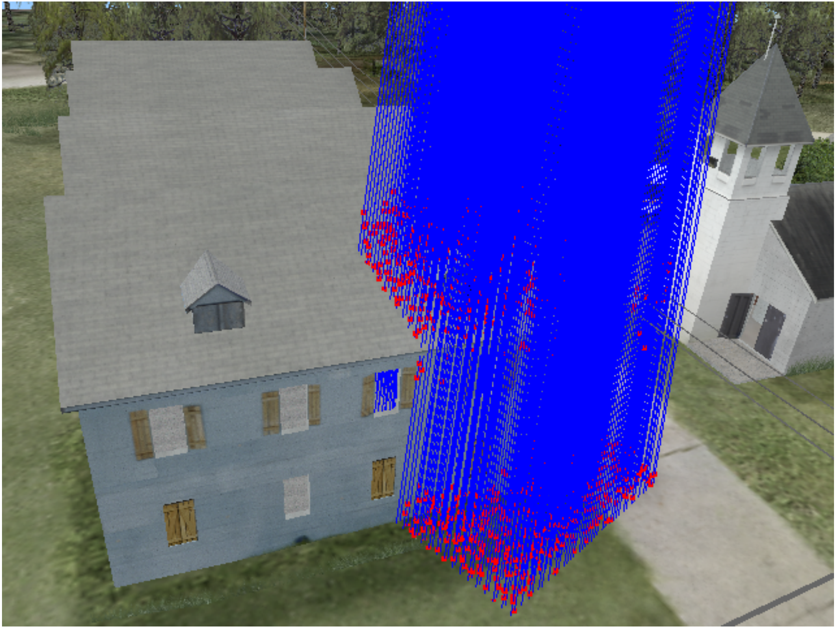


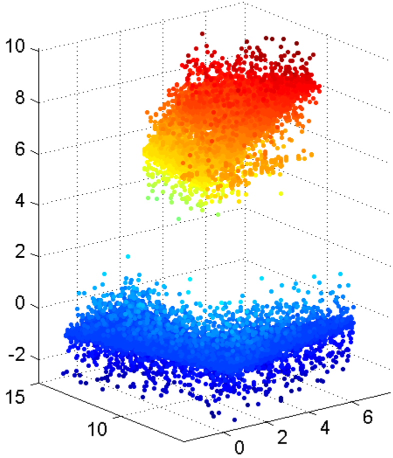
Original Scene, Points = 18207, Orig Passes = 5
10
8
6
4
2
0
-2
65
60
-20
-18
-16
-14
-12

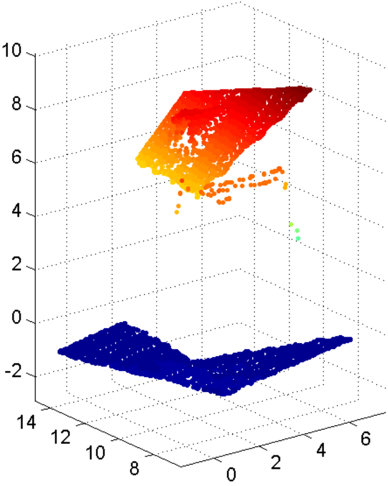


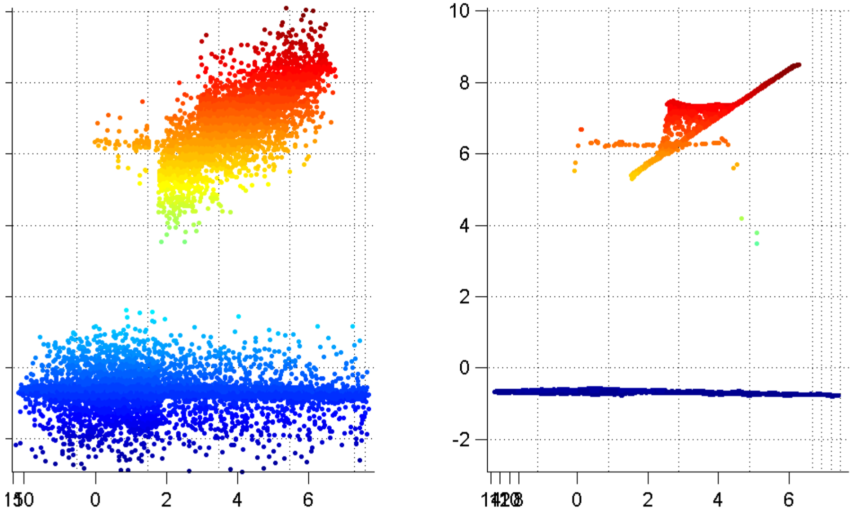
Local, μ=0.1, h=10000, Clouds=5, KNN=15, Passes=3, zFact=5
10
8
6
4
2
0
-2
65
60
-20
-18
-16
-14
-12






10
8
6
4
2
0
-2
15
10
0
2
4
6


10
8
6
4
2
0
-2
14
12
10
8
0
2
4
6


150
0
2
4
6
10
8
6
4
2
0
-2
14128
0
2
4
6

Comments about Simulations

- We assumed randomness in the xy-plane, but many LiDAR systems use **arrays of detectors** that have a strong grid structure, which can help us.

Comments about Simulations

- We assumed randomness in the xy-plane, but many LiDAR systems use **arrays of detectors** that have a strong grid structure, which can help us.
- Holes and open surfaces in the point cloud are OK.

Comments about Simulations

- We assumed randomness in the xy-plane, but many LiDAR systems use **arrays of detectors** that have a strong grid structure, which can help us.
- Holes and open surfaces in the point cloud are OK.
- Current histogram denoising method essentially "averages" (LPF) the data **and** forces a relatively coarse grid.

Comments about Simulations

- We assumed randomness in the xy-plane, but many LiDAR systems use **arrays of detectors** that have a strong grid structure, which can help us.
- Holes and open surfaces in the point cloud are OK.
- Current histogram denoising method essentially "averages" (LPF) the data **and** forces a relatively coarse grid.
- Our method does neither of these.

Comments about Simulations

- We assumed randomness in the *xy*-plane, but many LiDAR systems use **arrays of detectors** that have a strong grid structure, which can help us.
- Holes and open surfaces in the point cloud are OK.
- Current histogram denoising method essentially "averages" (LPF) the data **and** forces a relatively coarse grid.
- Our method does neither of these.
- Similar to non-local methods for 2D images, determining the weights has a huge impact.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.
- As far as we can tell this is novel.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.
- As far as we can tell this is novel.
- Permits simultaneous processing of objects with different codimension.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.
- As far as we can tell this is novel.
- Permits simultaneous processing of objects with different codimension.
- Can easily be extended to higher dimensional data sets.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.
- As far as we can tell this is novel.
- Permits simultaneous processing of objects with different codimension.
- Can easily be extended to higher dimensional data sets.
- Possible applications include robot navigation, face recognition, etc.

Conclusion

- We demonstrated the viability of denoising point cloud data using non-local techniques in a completely grid-free manner.
- This preserves the subtle geometric information of the cloud, and permits identification of specific geometric structures.
- As far as we can tell this is novel.
- Permits simultaneous processing of objects with different codimension.
- Can easily be extended to higher dimensional data sets.
- Possible applications include robot navigation, face recognition, etc.
- It may be possible to use the geometry encoder $\mathcal{G}$ to train a dictionary and obtain a sparse representation. This opens the possibility of obtaining further compression.

Thank you.

Questions?