INSTITUT NATIONAL DE L'INFORMATION GÉOGRAPHIQUE ET FORESTIÈRE

l'information grandeur nature

# Semantic Segmentation of 3D point Clouds


Loic Landrieu

Université Paris-Est - Machine Learning and Optimization working Group


March 2019

# Presentation

- Loic Landrieu, researcher at IGN (French Mapping Agency) in the AI department

# Presentation

- Loic Landrieu, researcher at IGN (French Mapping Agency) in the AI department
- PhD at INRIA/ENPC on *Graph-Structured Learning and Optimization*, w. Francis Bach and Guillaume Obozinski

# Presentation

- Loic Landrieu, researcher at IGN (French Mapping Agency) in the AI department
- PhD at INRIA/ENPC on *Graph-Structured Learning and Optimization*, w. Francis Bach and Guillaume Obozinski
- **Interest:** graph-structured functional optimization and deep learning.

# Presentation

- Loic Landrieu, researcher at IGN (French Mapping Agency) in the AI department
- PhD at INRIA/ENPC on *Graph-Structured Learning and Optimization*, w. Francis Bach and Guillaume Obozinski
- **Interest:** graph-structured functional optimization and deep learning.
- **Applications:** 3D point clouds, dynamic 3D for autonomous driving, superspectral satellite images, time series, medical inverse problems.

# Presentation outline


1. Deep Learning for 3D Point Clouds
2. Learning 3D Point Clouds Segmentation
3. The Cut Pursuit Algorithm
4. Conclusion
5. Bibliography

## Presentation Layout

1. **Deep Learning for 3D Point Clouds**
2. Learning 3D Point Clouds Segmentation
3. The Cut Pursuit Algorithm
4. Conclusion
5. Bibliography

## Presentation Layout

1. **Deep Learning for 3D Point Clouds**
   - Presentation of the Problem
   - Traditional Approaches
   - First Deep-Learning Approaches
   - Scaling Segmentation
2. **Learning 3D Point Clouds Segmentation**
3. **The Cut Pursuit Algorithm**
4. **Conclusion**
5. **Bibliography**

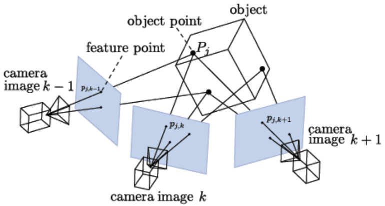
# Capturing a 3D world

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...


**credit:** medium, VisionSystemDesign, microsoft

## Capturing a 3D world

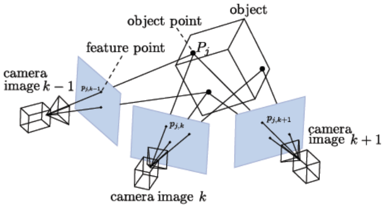
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).




**credit:** computervisionblog, velodynelidar

# Capturing a 3D world

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).

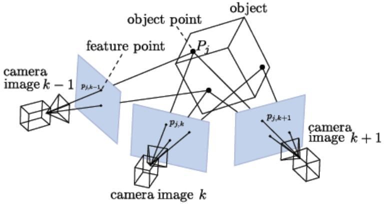




**credit:** computervisionblog, velodynelidar

# Capturing a 3D world

- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.

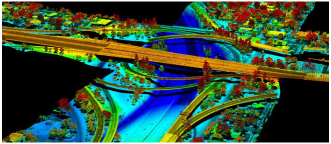




**credit:** computervisionblog, velodynelidar

## Capturing a 3D world

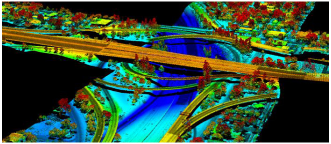
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.
- Produces a 3D point cloud: $P \in \mathbb{R}^{n \times 3}$.



**credit:** clearpath robotics, tuck mapping solutions

## Capturing a 3D world

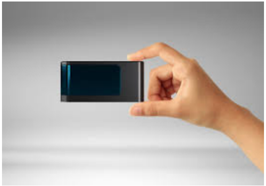
- 3D data crucial for robotics, autonomous vehicle, 3D scale models, virtual reality etc...
- Can be computed from images: stereo, SfM, SLAM (cheap, not precise).
- LiDAR (expensive, precise).
- Can be fixed, mobile, aerial, drone-embarked.
- Produces a 3D point cloud: $P \in \mathbb{R}^{n \times 3}$.
- Large acquisition: $n$ typically in the $10^8$s.



**credit:** clearpath robotics, tuck mapping solutions

## Future trends

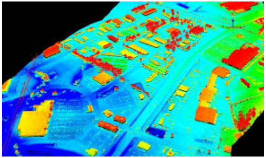
- LiDAR are getting cheaper :$100k\$ \rightarrow 2k\$$ in a few years.

**credit:** velodynelidar, green car congress

# Future trends

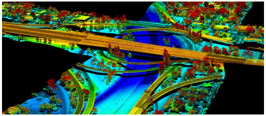
- LiDAR are getting cheaper :$100k\$ \to 2k\$$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).

**credit:** velodynelidar, spar3d

## Future trends

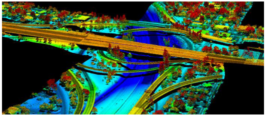
- LiDAR are getting cheaper :100$k$\$ $\rightarrow$ 2$k$\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...



**credit:** tuck mapping solutions, clearpath robotics

# Future trends

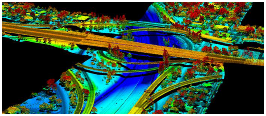
- LiDAR are getting cheaper :100$k$\$ $\rightarrow$ 2$k$\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...
- **Also to come:** major advances in automatic analysis of 3D data.



**credit:** tuck mapping solutions, clearpath robotics

# Future trends

- LiDAR are getting cheaper :100$k$\$ $\rightarrow$ 2$k$\$ in a few years.
- Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density).
- **Major industrial application:** autonomous driving, virtual models, land survey...
- **Also to come:** major advances in automatic analysis of 3D data.
- Rapid progress in harware and methodology + major applications = **a booming field.**

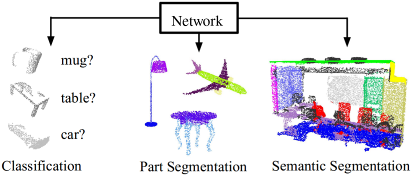


**credit:** tuck mapping solutions, clearpath robotics

# Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$




**credit:** Qi et. al. 2017a

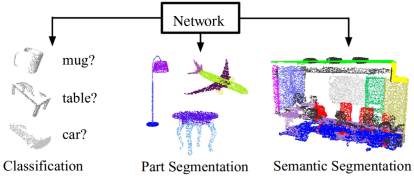
## Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in $C$ parts/object:

$$P_i \mapsto [1, \cdots, C]$$




**credit:** Qi et. al. 2017a

# Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

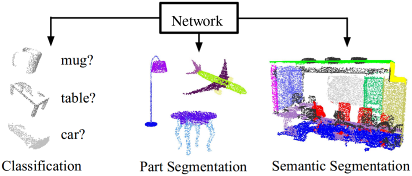
$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in $C$ parts/object:

$$P_i \mapsto [1, \cdots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between $K$ classes:

$$P_i \mapsto [1, \cdots, K]$$




**credit:** Qi et. al. 2017a

## Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in $C$ parts/object:

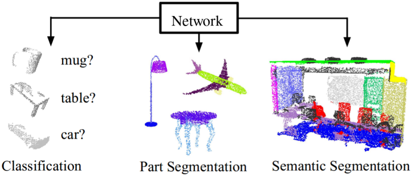
$$P_i \mapsto [1, \cdots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between $K$ classes:

$$P_i \mapsto [1, \cdots, K]$$

- **Instance Segmentation**: cluster the point cloud into semantically characterized objects:

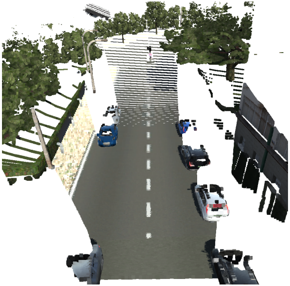
$$P_i \mapsto [1, \cdots, C]$$
$$[1, \cdots, C] \mapsto [1, \cdots, K]$$




**credit:** Qi et. al. 2017a

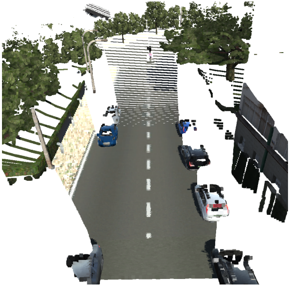
# What makes 3D analysis so hard

- Data volume considerable.



**credit:** Gaidon2016, Engelmann2017, Hackel2017

## What makes 3D analysis so hard

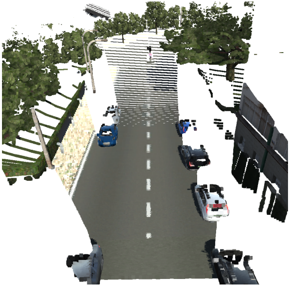


- Data volume considerable.
- Lack of grid-structure.

**credit:** Gaidon2016, Engelmann2017, Hackel2017

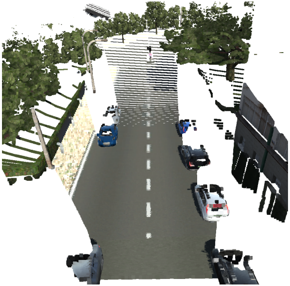
## What makes 3D analysis so hard

- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.



**credit:** Gaidon2016, Engelmann2017, Hackel2017

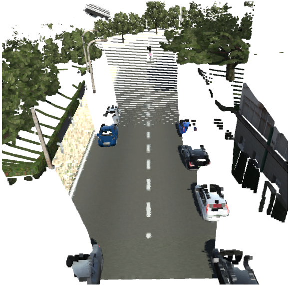
# What makes 3D analysis so hard



- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.

**credit:** Gaidon2016, Engelmann2017, Hackel2017

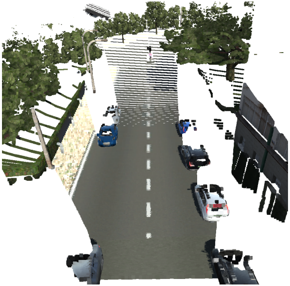
## What makes 3D analysis so hard



- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.

**credit:** Gaidon2016, Engelmann2017, Hackel2017

## What makes 3D analysis so hard

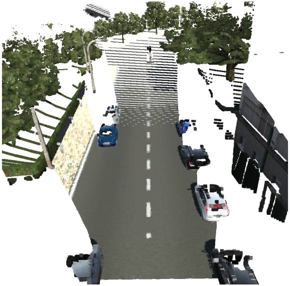


- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.
- Acquisition artifacts.

**credit:** Gaidon2016, Engelmann2017, Hackel2017

# What makes 3D analysis so hard

- Data volume considerable.
- Lack of grid-structure.
- Permutation-invariance.
- Sparsity.
- Highly variable density.
- Acquisition artifacts.
- Occlusions.



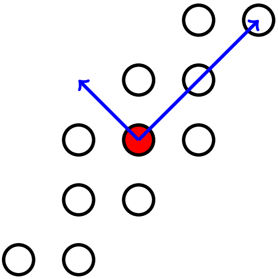
**credit:** Gaidon2016, Engelmann2017, Hackel2017

# Presentation Layout


1. **Deep Learning for 3D Point Clouds**
   - Presentation of the Problem
   - Traditional Approaches
   - First Deep-Learning Approaches
   - Scaling Segmentation
2. **Learning 3D Point Clouds Segmentation**
3. **The Cut Pursuit Algorithm**
4. **Conclusion**
5. **Bibliography**

# Pointwise classification

- **Step 1:** compute point features based on neighborhood



$$\text{Lin} = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1}}$$

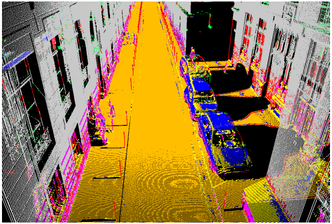
$$\text{Pla} = \frac{\sqrt{\lambda_2} - \sqrt{\lambda_3}}{\sqrt{\lambda_1}}$$

$$\text{Sca} = \frac{\sqrt{\lambda_3}}{\sqrt{\lambda_1}}$$

Demantke2011

# Pointwise classification

- **Step 1:** compute point features based on neighborhood
- **Step 2:** classification (RF, SVM, etc...)

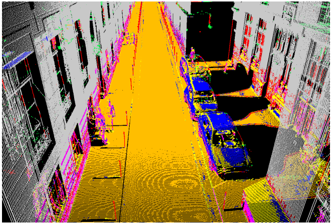


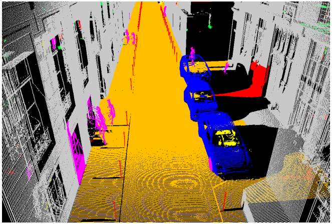
Demantke2011
Weimann2015

**credit:** landrieu et. al. 2017a

## Pointwise classification

- **Step 1:** compute point features based on neighborhood
- **Step 2:** classification (RF, SVM, etc...)
- **Step 3:** smoothing to increase spatial regularity (with CRFs, MRFs, graph-structured optimization, etc...)





Demantke2011
Weimann2015
Landrieu et. al. 2017a

**credit:** landrieu et. al. 2017a

# Presentation Layout

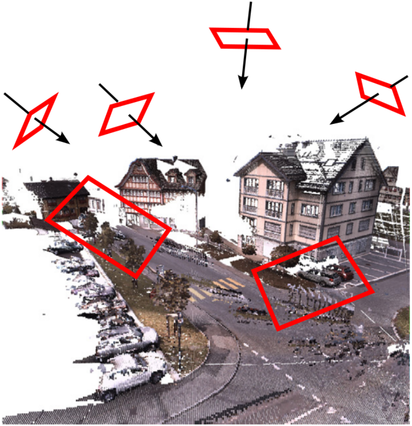
1. **Deep Learning for 3D Point Clouds**
   - Presentation of the Problem
   - Traditional Approaches
   - First Deep-Learning Approaches
   - Scaling Segmentation
2. **Learning 3D Point Clouds Segmentation**
3. **The Cut Pursuit Algorithm**
4. **Conclusion**
5. **Bibliography**

## Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?

# Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**

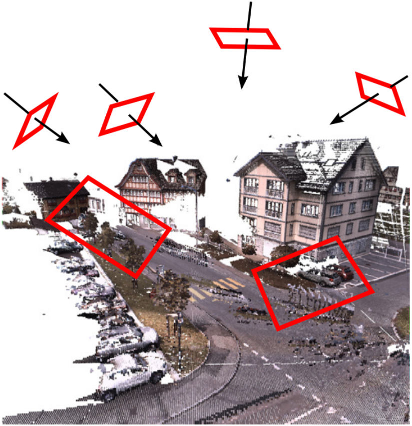


Boulch et. al. 2017

**credit:** Boulch et. al. 2017

## Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction

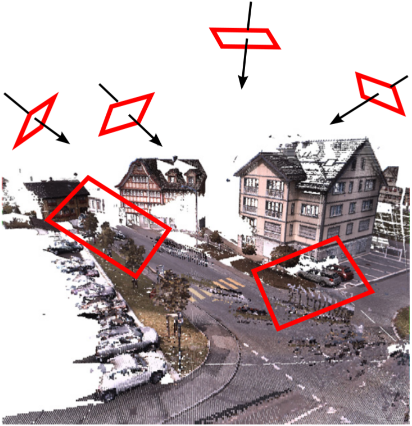


Boulch et. al. 2017

**credit:** Boulch et. al. 2017

## Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*

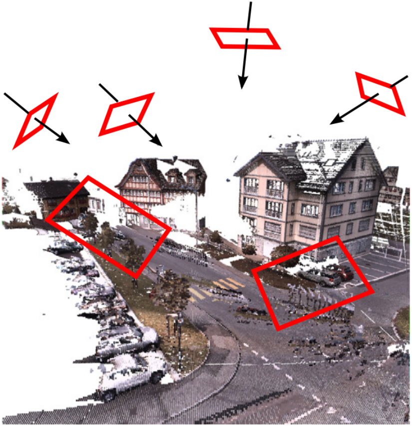


Boulch et. al. 2017

**credit:** Boulch et. al. 2017

# Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*
  - semantic segmentation of resulting images with CNNs

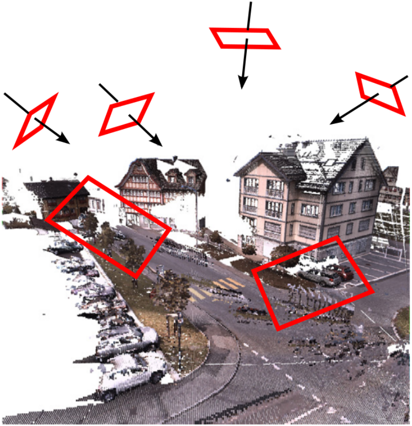


Boulch et. al. 2017

**credit:** Boulch et. al. 2017

## Image-Based Methods

- **A simple observation:** CNNs works great for images. Can we use images for 3D?
- **SnapNet:**
  - surface reconstruction
  - *virtual snapshots*
  - semantic segmentation of resulting images with CNNs
  - project prediction back to p.c.



Boulch et. al. 2017

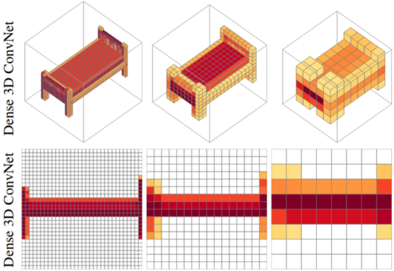
**credit:** Boulch et. al. 2017

# Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids

# Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets

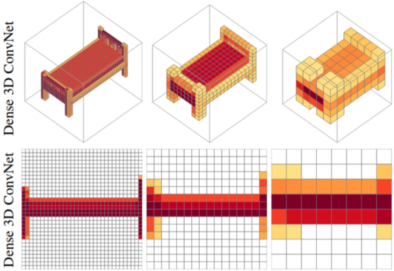



Wu2015

**credit:** Riegler2017, Tchapmi2017, Jampani2017

## Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)

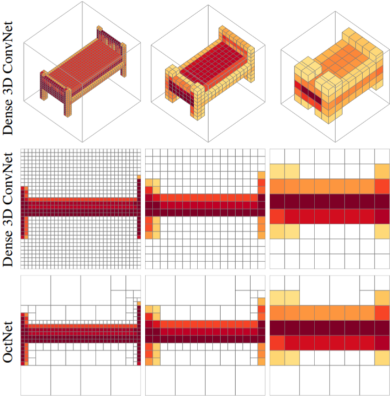



Wu2015

**credit:** Riegler2017, Tchapmi2017, Jampani2017

# Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach

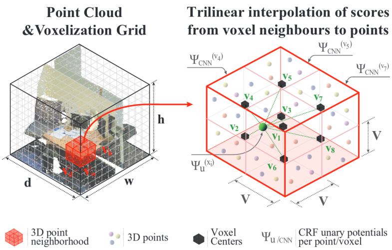



Wu2015 , Riegler2017

**credit:** Riegler2017, Tchapmi2017, Jampani2017

# Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach
- **Idea 2:** SegCloud, large voxels, subvoxel predictions with CRFs.

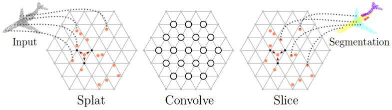



Wu2015 , Riegler2017 , Tchapmi2017, Jampani2018.

**credit:** Riegler2017, Tchapmi2017, Jampani2017

# Voxel-Based Methods

- **Idea:** generalize 2D convolutions to regular 3D grids
- Voxelization + 3D convNets
- **Problem:** inefficient representation, loss of invariance, costly (cubic)
- **Idea 1:** OctNet, OctTree based approach
- **Idea 2:** SegCloud, large voxels, subvoxel predictions with CRFs.
- **Idea 3:** SplatNet, sparse convolutions with hashmaps.




Wu2015 , Riegler2017 , Tchapmi2017, Jampani2018.

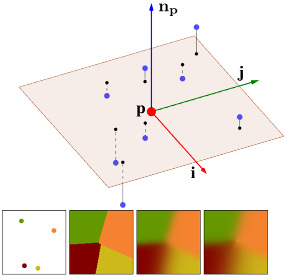
**credit:** Riegler2017, Tchapmi2017, Jampani2017

# 3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.

# 3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.

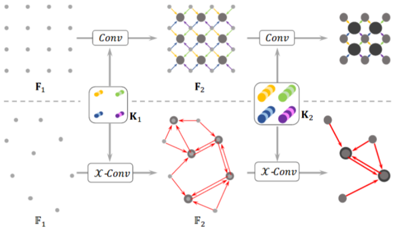



Tatarchenko2018

**credit:** Tatarchenko2018, Li2018

# 3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** $\chi$-convolutions: generalized convolutions for unordered inputs.

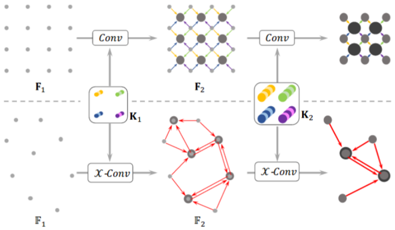



Tatarchenko2018 , Li2018.

**credit:** Tatarchenko2018, Li2018

## 3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** $\chi$-convolutions: generalized convolutions for unordered inputs.
- **Principle:** the network learns how to permute *ordered* inputs

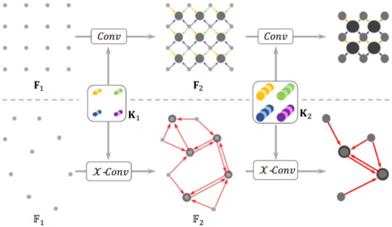



Tatarchenko2018 , Li2018.

**credit:** Tatarchenko2018, Li2018

## 3D Convolution-Based Methods

- **Idea:** generalize 2D convolutions to 3D point clouds as unordered data.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- **PointCNN :** $\chi$-convolutions: generalized convolutions for unordered inputs.
- **Principle:** the network learns how to permute *ordered* inputs
- The invariance is learnt!




Tatarchenko2018 , Li2018.

**credit:** Tatarchenko2018, Li2018

## PointNet

- **A fondamental constraint:** inputs are invariant by permutation

Qi et. al.2017a

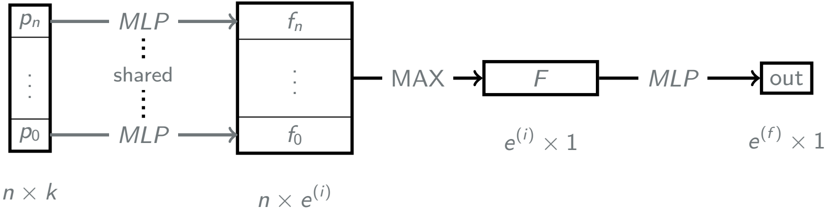
## PointNet

- **A fondamental constraint:** inputs are invariant by permutation
- **Solution:** process points independently, apply permutation-invariant pooling, process this feature with a MLP.

Qi et. al.2017a

# PointNet

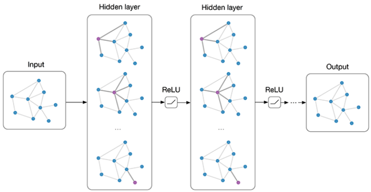
- **A fondamental constraint:** inputs are invariant by permutation
- **Solution:** process points independently, apply permutation-invariant pooling, process this feature with a MLP.
- $n$: number of points, $k$ size of observations, $e^{(i)}$ size of intermediary embeddings, $e^{(f)}$ size of output




Qi et. al.2017a

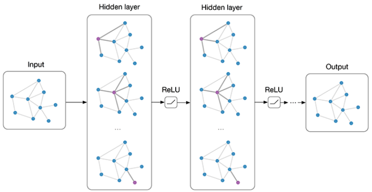
# Graph-Neural Network

- Generalize convolutions to the general graph setting.



## Graph-Neural Network

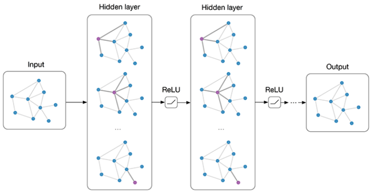
- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.




Qi2017, Simonovski2017

# Graph-Neural Network

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state $h_i$ influenced by its neighbors.

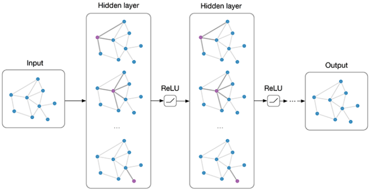



Qi2017, Simonovski2017

## Graph-Neural Network

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state $h_i$ influenced by its neighbors.
- GNN **Qi2017**: an iterative message-passing algorithm using a mapping $f$ and a RNN $g$:

$$h_i^{(t+1)} = g\left(\sum_{j \to i} f(h_i^t), h_i^t\right)$$




Qi2017, Simonovski2017

**credit:** Qi2017

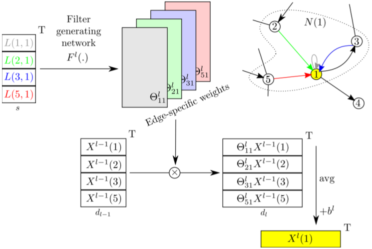
# Graph-Neural Network

- Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph of 3D points.
- **Idea:** Each point maintain a hidden state $h_i$ influenced by its neighbors.
- GNN **Qi2017**: an iterative message-passing algorithm using a mapping $f$ and a RNN $g$:

$$h_i^{(t+1)} = g(\sum_{j \to i} f(h_j^t), h_i^t)$$

- ECC **Simonovski2017** messages are conditioned by edge features:

$$h_i^{(t+1)} = g(\sum_{j \to i} \Theta_{i,j} \odot h_j^t, h_i^t)$$




Qi2017, Simonovski2017

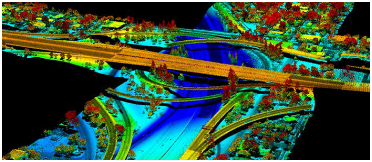
**credit:** Simonovski2017

## Presentation Layout


1. **Deep Learning for 3D Point Clouds**
   - Presentation of the Problem
   - Traditional Approaches
   - First Deep-Learning Approaches
   - Scaling Segmentation
2. **Learning 3D Point Clouds Segmentation**
3. **The Cut Pursuit Algorithm**
4. **Conclusion**
5. **Bibliography**

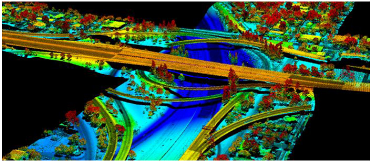
# Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.



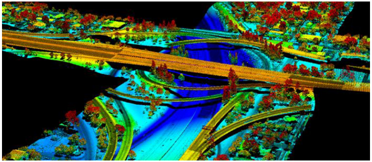
# Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.



## Why we need to scale

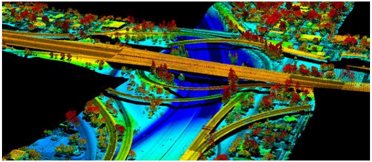
- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.
- **Naive strategies:**
  - **Aggressive subsampling**: loses a lot of information.



## Why we need to scale

- **Problem:** best approaches are very memory-hungry and the data volumes are huge.
- Previous methods only works with a few thousands points.
- **Naive strategies:**

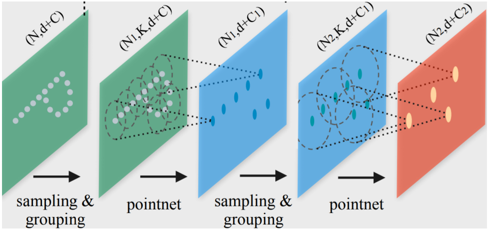
- **Aggressive subsampling**: loses a lot of information.
- **Sliding windows**: loses the global structure.



**credit:** tuck mapping solution

# PointNet++

- Pyramid structure for multi-scale feature extraction.

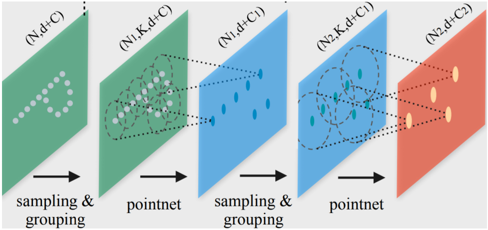



Qi et. al.2017b

**credit:** Qi et. al.2017b

# PointNet++

- Pyramid structure for multi-scale feature extraction.
- From local to global with with increasingly abstract features.

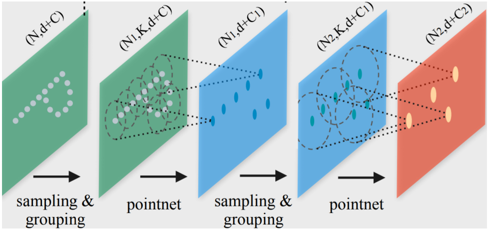



Qi et. al.2017b

**credit:** Qi et. al.2017b

# PointNet++

- Pyramid structure for multi-scale feature extraction.
- From local to global with with increasingly abstract features.
- Still require to process millions of points.

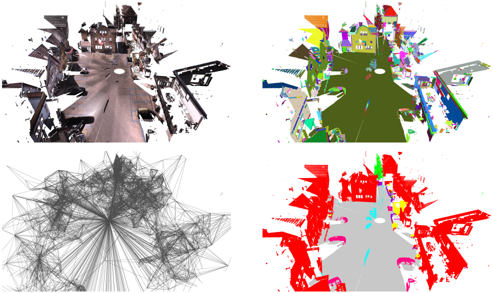



Qi et. al.2017b

**credit:** Qi et. al.2017b

# SuperPoint-Graph

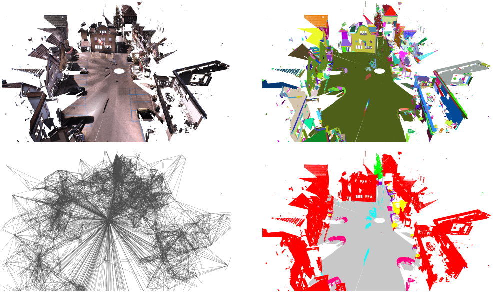
- **Observation:**
  $n_{\text{points}} \gg n_{\text{objects}}$.



Landrieu&Simonovski2018

# SuperPoint-Graph

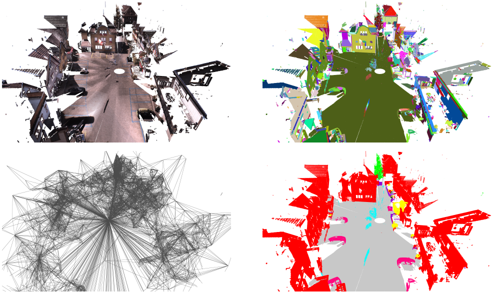
- **Observation:**
  $n_{\text{points}} \gg n_{\text{objects}}$.
- Partition scene into superpoints with simple shapes.



Landrieu&Simonovski2018

# SuperPoint-Graph

- **Observation:**
  $n_{\text{points}} \gg n_{\text{objects}}$.
- Partition scene into superpoints with simple shapes.
- Only a few superpoints, context leveraging with powerful graph methods.



Landrieu&Simonovski2018

## Pipeline

- Semantic segmentation down to 3 sub-problems:

# Pipeline

- Semantic segmentation down to 3 sub-problems:

- **Geometric Partition :** into simple shapes.

  Complexity: very high (clouds of $10^8$ points)

  Algorithm: $\ell_0$-cut pursuit

## Pipeline

- Semantic segmentation down to 3 sub-problems:
- **Geometric Partition :** into simple shapes.

  Complexity: very high (clouds of $10^8$ points)

  Algorithm: $\ell_0$-cut pursuit
- **Superpoint embedding:** learning shape descriptors

  Complexity: low (subsampling to 128 points $\times \sim$ 1000 points)

  Algorithm: PointNet

## Pipeline

- Semantic segmentation down to 3 sub-problems:
- **Geometric Partition :** into simple shapes.

  Complexity: very high (clouds of $10^8$ points)

  Algorithm: $\ell_0$-cut pursuit
- **Superpoint embedding:** learning shape descriptors

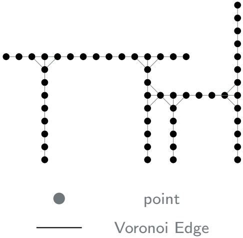
  Complexity: low (subsampling to 128 points $\times \sim$ 1000 points)

  Algorithm: PointNet
- **Contextual Segmentation:** using the global structure

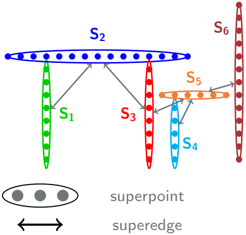
  Complexity: very low (superpoint graph $\sim$ 1000 sp)

  Algorithm: ECC with Gated Recurrent Unit (GRU)

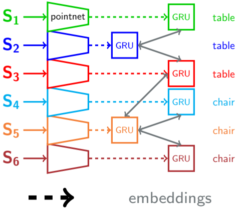
## Pipeline




(a) Point cloud



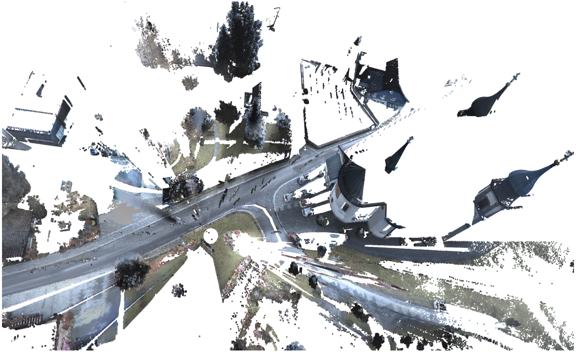

(b) Superpoint graph




(c) Convolution Network

# Qualitative Results: Semantic3D

Semantic3D: 3 billions points over 30 clouds

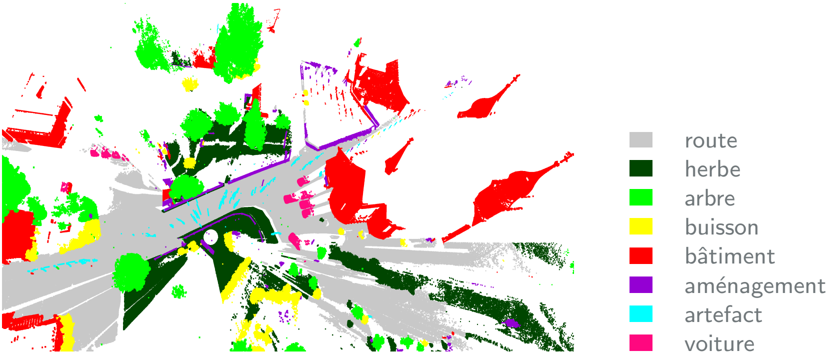


- route
- herbe
- arbre
- buisson
- bâtiment
- aménagement
- artefact
- voiture

Semantic3D: 3 billions points over 30 clouds

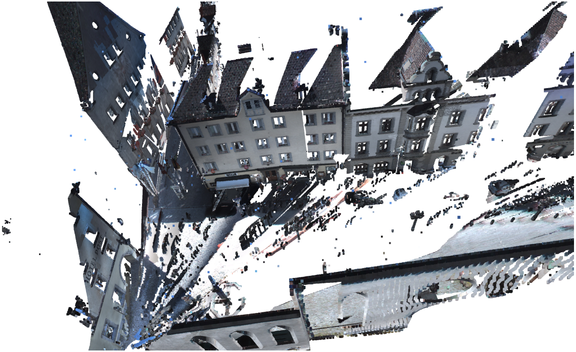
# Qualitative Results: Semantic3D

Semantic3D: 3 billions points over 30 clouds



## Qualitative Results: Semantic3D

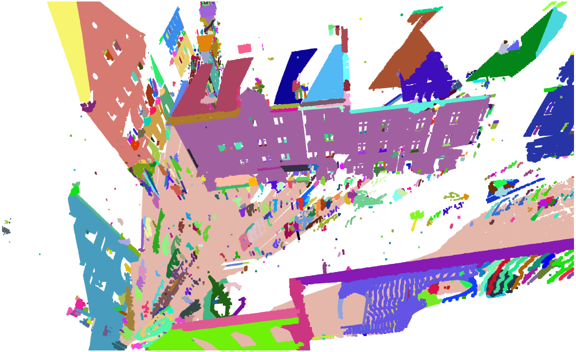
Semantic3D: 3 billions points over 30 clouds



- route
- herbe
- arbre
- buisson
- bâtiment
- aménagement
- artefact
- voiture

# Qualitative Results: Semantic3D

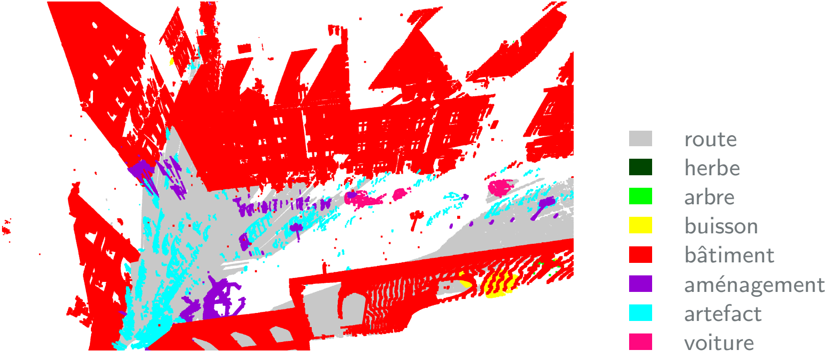
Semantic3D: 3 billions points over 30 clouds



- route
- herbe
- arbre
- buisson
- bâtiment
- aménagement
- artefact
- voiture

# Qualitative Results: Semantic3D

Semantic3D: 3 billions points over 30 clouds



# Quantitative Results: Semantic3D

| **Methode** | OA | mIoU | road | grass | tree | bush | build-ing | hard-scape | arti-fact | cars |
|---|---|---|---|---|---|---|---|---|---|---|
| reduced test set: 78 699 329 points | | | | | | | | | | |
| TMLC-MSR | 86.2 | 54.2 | 89.8 | 74.5 | 53.7 | 26.8 | 88.8 | 18.9 | 36.4 | 44.7 |
| DeePr3SS | 88.9 | 58.5 | 85.6 | 83.2 | 74.2 | 32.4 | 89.7 | 18.5 | 25.1 | 59.2 |
| SnapNet | 88.6 | 59.1 | 82.0 | 77.3 | 79.7 | 22.9 | 91.1 | 18.4 | 37.3 | 64.4 |
| SegCloud | 88.1 | 61.3 | 83.9 | 66.0 | 86.0 | 40.5 | 91.1 | 30.9 | 27.5 | 64.3 |
| SPG (Ours) | **94.0** | **73.2** | **97.4** | **92.6** | **87.9** | **44.0** | **93.2** | **31.0** | **63.5** | **76.2** |
| full test set: 2 091 952 018 points | | | | | | | | | | |
| TMLC-MS | 85.0 | 49.4 | 91.1 | 69.5 | 32.8 | 21.6 | 87.6 | 25.9 | 11.3 | 55.3 |
| SnapNet | 91.0 | 67.4 | 89.6 | **79.5** | 74.8 | 56.1 | 90.9 | 36.5 | 34.3 | 77.2 |
| SPG (Ours) | **92.9** | **76.2** | **91.5** | 75.6 | **78.3** | **71.7** | **94.4** | **56.8** | **52.9** | **88.4** |

# Qualitative Results: S3DIS

Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points

- ceiling
- ground
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- board
- other

# Qualitative Results: S3DIS

Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points

- ceiling
- ground
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- board
- other

# Qualitative Results: S3DIS

Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points

- ceiling
- ground
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- board
- other

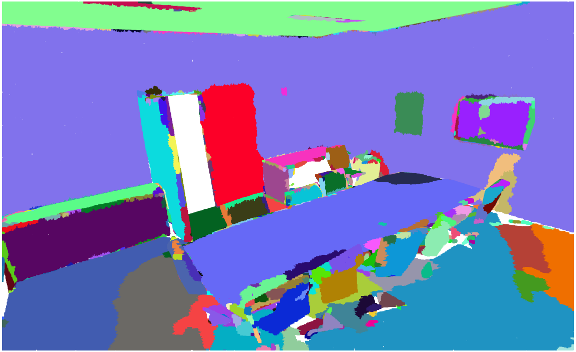
## Qualitative Results: S3DIS

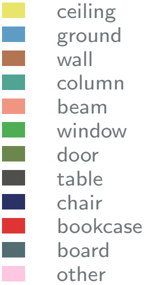
Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points

- ceiling
- ground
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- board
- other

# Qualitative Results: S3DIS

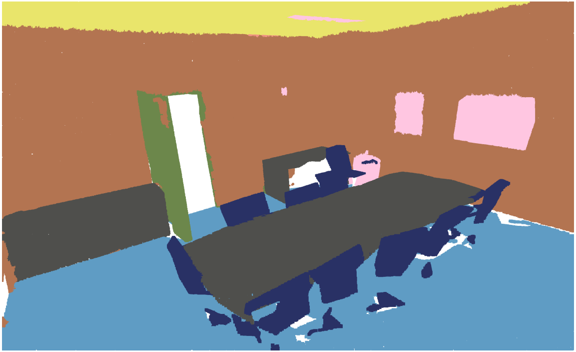
Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points

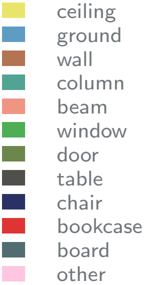




# Qualitative Results: S3DIS

Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points





# Résultats qualitatif: S3DIS

## Quantitative Results: S3DIS

| **Method** | OA | mAcc | mIoU | door | board |
|---|---|---|---|---|---|
| A5 PointNet | – | 48.5 | 41.1 | 10.7 | **26.3** |
| A5 SEGCloud | – | 57.3 | 48.9 | 23.1 | 13.0 |
| **A5 SPG** | 86.4 | **66.5** | **58.0** | **61.5** | 2.1 |
| PointNet | 78.5 | 66.2 | 47.6 | 51.6 | 29.4 |
| Engelmann | 81.1 | 66.4 | 49.7 | 51.2 | **30.0** |
| **SPG** | **85.5** | **73.0** | **62.1** | **68.4** | 8.7 |

## Quantitative Results: S3DIS

| Method | OA | mAcc | mIoU | door | board |
|---|---|---|---|---|---|
| A5 PointNet | – | 48.5 | 41.1 | 10.7 | **26.3** |
| A5 SEGCloud | – | 57.3 | 48.9 | 23.1 | 13.0 |
| **A5 SPG** | 86.4 | **66.5** | **58.0** | **61.5** | 2.1 |
| PointNet | 78.5 | 66.2 | 47.6 | 51.6 | 29.4 |
| Engelmann | 81.1 | 66.4 | 49.7 | 51.2 | **30.0** |
| **SPG** | **85.5** | **73.0** | **62.1** | **68.4** | 8.7 |

| Śtep | Full cloud | 2 cm | 3 cm | 4 cm |
|---|---|---|---|---|
| Voxelisation | 0 | 40 | 24 | 16 |
| Features | 439 | 194 | 88 | 43 |
| Partition | 3428 | 1013 | 447 | 238 |
| SPG computation | 3800 | 958 | 436 | 252 |
| Inference $\times 10$ | 240 | 110 | 60 | 50 |
| Total | 7907 | 2315 | 1055 | 599 |
| mIoU 6-fold | 54.1 | 60.2 | 62.1 | 57.1 |

## Superpoint Partition

$$f^\star = \underset{f\in\mathbb{R}^{C\times m}}{\arg\min} \sum_{i\in C} \|f_i - e_i\|^2 + \sum_{(i,j)\in E} w_{i,j}\,[f_i \neq f_j]\,,$$

- $e \in \mathbb{R}^{C\times m}$ : handcrafted descriptors of the local geometry/radiometry

## Superpoint Partition

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- $e \in \mathbb{R}^{C \times m}$ : handcrafted descriptors of the local geometry/radiometry
- Superpoints: connected components of a piecewise constant approximation of $e$ structured by an adjacency graph.

## Superpoint Partition

$$f^\star = \underset{f \in \mathbb{R}^{C\times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j)\in E} w_{i,j}\,[f_i \neq f_j]\,,$$

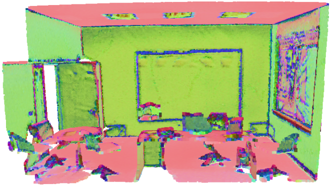
- $e \in \mathbb{R}^{C\times m}$ : handcrafted descriptors of the local geometry/radiometry
- Superpoints: connected components of a piecewise constant approximation of $e$ structured by an adjacency graph.
- **Problem:** any errors made in the partition will carry in the prediction...

# Presentation Layout

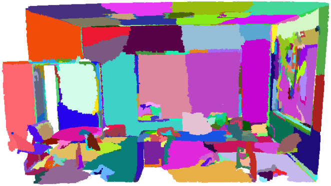

1. Deep Learning for 3D Point Clouds
2. **Learning 3D Point Clouds Segmentation**
3. The Cut Pursuit Algorithm
4. Conclusion
5. Bibliography

# The Pipeline

Input Point Cloud



Learned Embedding



Oversegmentation

True Objects

**General idea:**

1) Train a neural network to produce points embeddings with high contrast at the border of objects...

2) ... Which serve as inputs of a **nondifferentiable** segmentation algorithm.

# Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph

## Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant

## Adjacency Graph

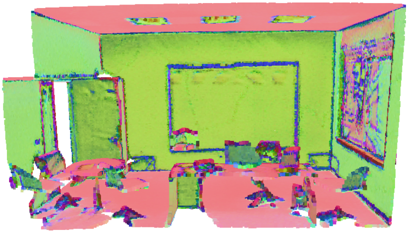
- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant
- $E_{\text{inter}}$ : set of inter-object edges

## Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant
- $E_{\text{inter}}$ : set of inter-object edges
- $E_{\text{intra}}$ : set of intra-object edges

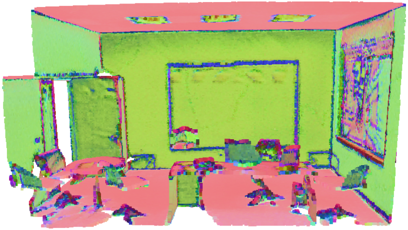
## Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant
- $E_{\text{inter}}$ : set of inter-object edges
- $E_{\text{intra}}$ : set of intra-object edges
- We want embeddings with high contrast at $E_{\text{inter}}$ and similar value at $E_{\text{intra}}$



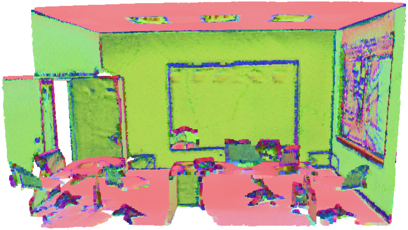
## Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant
- $E_{\text{inter}}$ : set of inter-object edges
- $E_{\text{intra}}$ : set of intra-object edges
- We want embeddings with high contrast at $E_{\text{inter}}$ and similar value at $E_{\text{intra}}$
- If we get $E_{\text{inter}}$ right, then we have automatically object purity!



## Adjacency Graph

- $G = (C, E)$ a meaningful adjacency graph
- Construction is problem-dependant
- $E_{\text{inter}}$ : set of inter-object edges
- $E_{\text{intra}}$ : set of intra-object edges
- We want embeddings with high contrast at $E_{\text{inter}}$ and similar value at $E_{\text{intra}}$
- If we get $E_{\text{inter}}$ right, then we have automatically object purity!
  almost!



# Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry

# Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry
- **Idea:** Superpoints are the component of a **piecewise-constant approximation** of the embedings

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

# Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry
- **Idea:** Superpoints are the component of a **piecewise-constant approximation** of the embedings

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- Superpoints: regions with homogeneous embeddings

# Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry
- **Idea:** Superpoints are the component of a **piecewise-constant approximation** of the embedings

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Superpoints: regions with homogeneous embeddings
- Works well with handcrafted embeddings, should work with learned ones!

## Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry
- **Idea:** Superpoints are the component of a **piecewise-constant approximation** of the embedings

$$f^\star = \arg\min_{f \in \mathbb{R}^{C \times m}} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Superpoints: regions with homogeneous embeddings
- Works well with handcrafted embeddings, should work with learned ones!
- **Problem:** a non-convex, nondifferentiable, noncontinuous problem

## Generalized Minimal Partition Problem

- $e_i$ embeddings of the local geometry/radiometry
- **Idea:** Superpoints are the component of a **piecewise-constant approximation** of the embedings

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \left\| f_i - e_i \right\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- Superpoints: regions with homogeneous embeddings
- Works well with handcrafted embeddings, should work with learned ones!
- **Problem:** a non-convex, nondifferentiable, noncontinuous problem
- Good approximations can be computed with $\ell_0$-cut pursuit [Landrieu & Obozinski SIIMS 2018]

## The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- Let consider our pipeline:

## The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder

## The Problem With the GMPP

$$f^{\star} = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings

## The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \| f_i - e_i \|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right] ,$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP

# The Problem With the GMPP

$$f^\star = \operatorname*{arg\,min}_{f \in \mathbb{R}^{C \times m}} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP
  - Let $CCC$ the constant connected component operator on $G$
  - The superpoints are: $\mathcal{S} = CCC(f^\star(e(x)))$

## The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP
  - Let $CCC$ the constant connected component operator on $G$
  - The superpoints are: $\mathcal{S} = CCC(f^\star(e(x)))$
- Let $M(\mathcal{S})$ be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)

# The Problem With the GMPP

$$f^\star = \arg\min_{f \in \mathbb{R}^{C \times m}} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[ f_i \neq f_j \right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP
  - Let $CCC$ the constant connected component operator on $G$
  - The superpoints are: $\mathcal{S} = CCC(f^\star(e(x)))$
- Let $M(\mathcal{S})$ be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)
- **Naive Approach:** $\ell(x) = -M(CCC(f^\star(e(x))))$

## The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP
  - Let $CCC$ the constant connected component operator on $G$
  - The superpoints are: $\mathcal{S} = CCC(f^\star(e(x)))$
- Let $M(\mathcal{S})$ be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)
- **Naive Approach:** $\ell(x) = -M(CCC(f^\star(e(x))))$
- To backpropagate we need: $\frac{\partial CCC}{\partial f^\star}$ and $\frac{\partial f^\star}{\partial e}$

# The Problem With the GMPP

$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- Let consider our pipeline:
  - Let $x$ be the parameters of the Local Point Embedder
  - Let $e(x)$ be the resulting embeddings
  - Let $f^\star(e(x))$ be the solution of the GMPP
  - Let $CCC$ the constant connected component operator on $G$
  - The superpoints are: $\mathcal{S} = CCC(f^\star(e(x)))$
- Let $M(\mathcal{S})$ be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)
- **Naive Approach:** $\ell(x) = -M(CCC(f^\star(e(x))))$
- To backpropagate we need: $\frac{\partial CCC}{\partial f^\star}$ and $\frac{\partial f^\star}{\partial e}$
- **Problem:**Those functions are **not backpropagable**.

## Graph-Structured Contrastive Loss

- We propose a *surrogate* loss to learn meaningful embeddings

$$\ell(e) = \frac{1}{|E|}\left(\sum_{(i,j)\in E_{\text{intra}}} \phi\left(e_i - e_j\right) + \sum_{(i,j)\in E_{\text{inter}}} \mu_{i,j}\psi\left(e_i - e_j\right)\right),$$

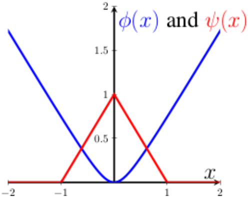
## Graph-Structured Contrastive Loss

- We propose a *surrogate* loss to learn meaningful embeddings

$$\ell(e) = \frac{1}{|E|}\left(\sum_{(i,j)\in E_{\text{intra}}} \phi\left(e_i - e_j\right) + \sum_{(i,j)\in E_{\text{inter}}} \mu_{i,j}\psi\left(e_i - e_j\right)\right),$$

- $\phi$ minimum at 0, $\psi$ maximum at 0

$$\begin{aligned}\phi(x) &= \delta(\sqrt{\|x\|^2/\delta^2+1}-1)\\ \psi(x) &= \max\left(1-\|x\|,0\right)\end{aligned}$$



## Graph-Structured Contrastive Loss

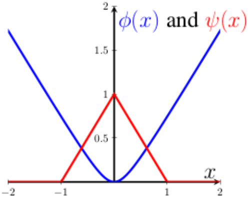
- We propose a *surrogate* loss to learn meaningful embeddings

$$\ell(e) = \frac{1}{|E|}\left(\sum_{(i,j)\in E_{\text{intra}}} \phi\left(e_i - e_j\right) + \sum_{(i,j)\in E_{\text{inter}}} \mu_{i,j}\psi\left(e_i - e_j\right)\right),$$

- $\phi$ minimum at 0, $\psi$ maximum at 0

$$\begin{aligned}\phi(x) &= \delta(\sqrt{\|x\|^2/\delta^2+1}-1)\\ \psi(x) &= \max\left(1-\|x\|,0\right)\end{aligned}$$

- Promotes homogeneity within objects and contrast at their borders



## Graph-Structured Contrastive Loss

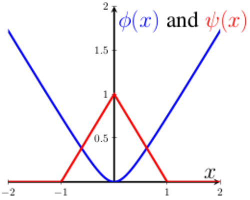
- We propose a *surrogate* loss to learn meaningful embeddings

$$\ell(e) = \frac{1}{|E|}\left(\sum_{(i,j)\in E_{\text{intra}}} \phi\left(e_i - e_j\right) + \sum_{(i,j)\in E_{\text{inter}}} \mu_{i,j}\psi\left(e_i - e_j\right)\right),$$

- $\phi$ minimum at 0, $\psi$ maximum at 0

$$\begin{aligned}\phi(x) &= \delta(\sqrt{\|x\|^2/\delta^2+1}-1)\\ \psi(x) &= \max\left(1-\|x\|, 0\right)\end{aligned}$$

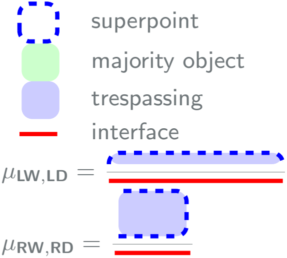
- Promotes homogeneity within objects and contrast at their borders
- $\mu_{i,j}$ : weight of inter-edges

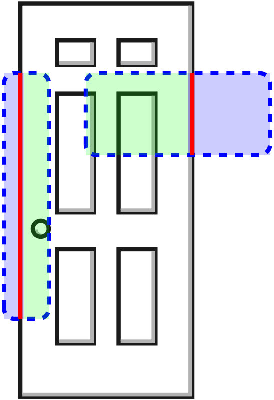


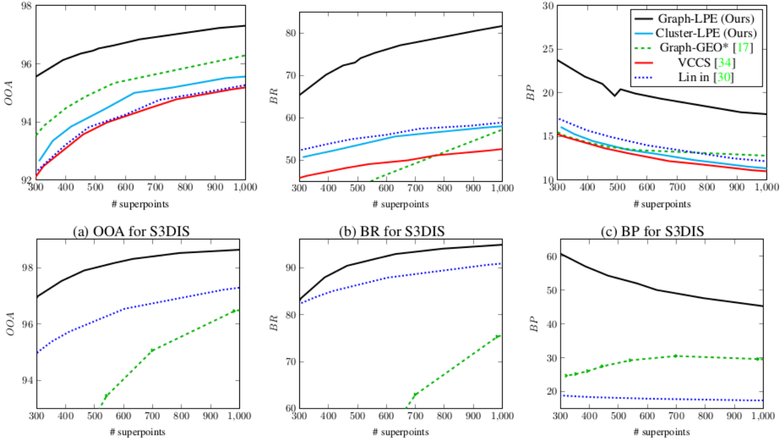
## Cross-Partition Weighting Strategy, cont'd

$$\mu_{U,V} = \mu \frac{\min(|U|, |V|)}{|(U,V)|} \quad \text{for } (U,V) \in \mathcal{E} \qquad \mu_{i,j} = \mu_{U,V} \text{ for all } (i,j) \in (U,V)$$

- Role of $\mu_{i,j}$ critical: assess impact of missed edge.
- Operate on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ adjacency graph of cross-partition between superpoints and real objects.



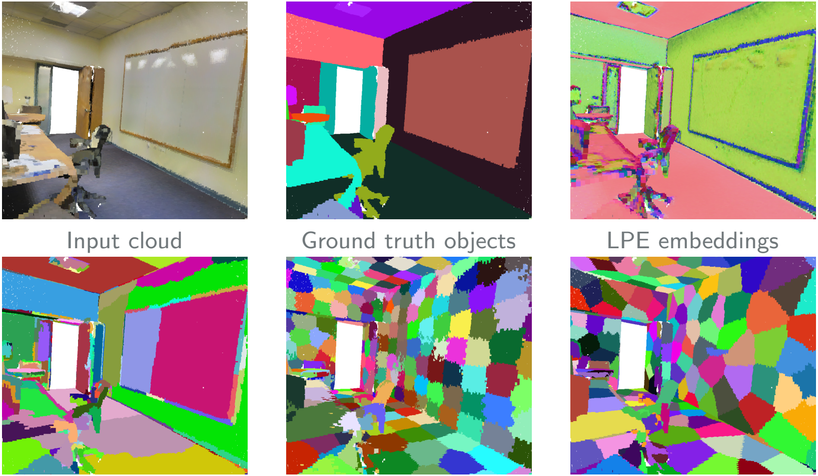



(a) OOA for S3DIS (b) BR for S3DIS (c) BP for S3DIS

(d) OOA for vKITTI (e) BR for vKITTI (f) BP for vKITTI

**We require 5 times less superpoints for similar performance!**

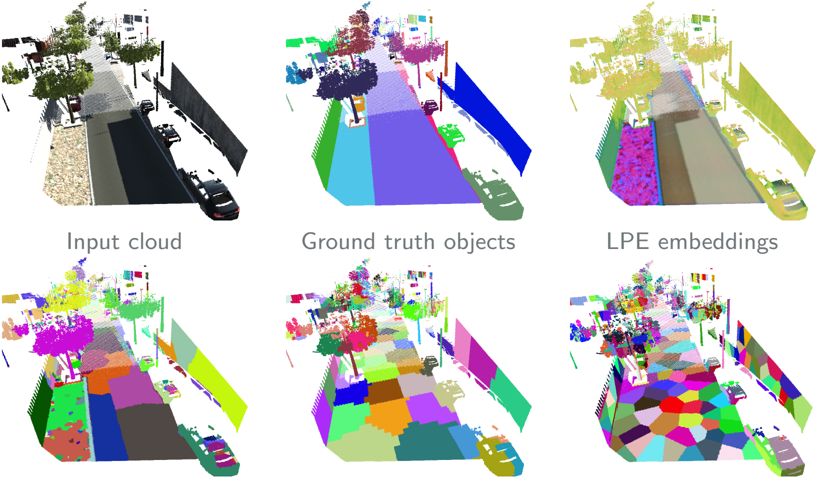
# Illustration



Input cloud | Ground truth objects | LPE embeddings

Graph-LPE (ours) | VCCS, Papon *et al.* 2013 | Lin *et al.* 2018

# Illustration



Input cloud — Ground truth objects — LPE embeddings

Graph-LPE (ours) — VCCS, Papon *et al.* 2013 — Lin *et al.* 2018

## Results

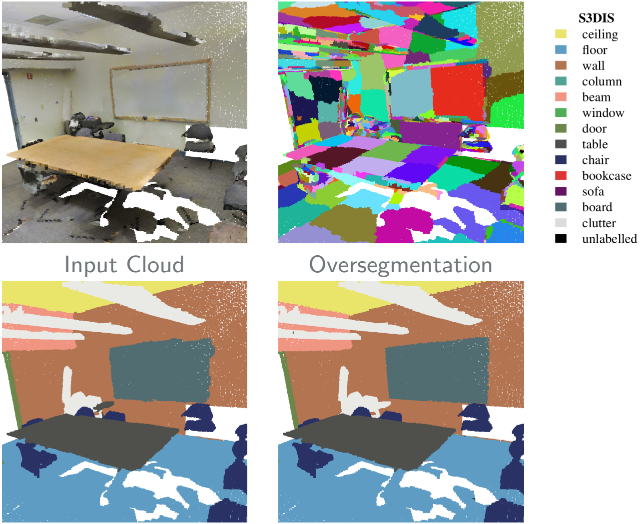
| **Method** | OA | mAcc | mIoU |
|---|---|---|---|
| 6-fold cross validation | | | |
| PointNet 2017 | 78.5 | 66.2 | 47.6 |
| Engelmann *et al.* in 2017 | 81.1 | 66.4 | 49.7 |
| PointNet++ 2017 | 81.0 | 67.1 | 54.5 |
| Engelmann *et al.* in 2018 | 84.0 | 67.8 | 58.3 |
| SPG 2018 | 85.5 | 73.0 | 62.1 |
| PointCNN 2018 | **88.1** | 75.6 | 65.4 |
| Graph-LPE + SPG (ours) | 87.8 | **77.5** | **67.6** |
| Fold 5 | | | |
| PointNet 2017 | - | 49.0 | 41.1 |
| Engelmann *et al.* in 2018 | 84.2 | 61.8 | 52.2 |
| pointCNN 2018 | 85.9 | 63.9 | 57.3 |
| SPG 2018 | 86.4 | 66.5 | 58.0 |
| PCCN 2018 | - | 67.0 | 58.3 |
| Graph-LPE + SPG (ours) | **87.8** | **69.1** | **61.5** |

**Table:** S3DIS

| **Method** | OA | mAcc | mIoU |
|---|---|---|---|
| PointNet 2017 | 79.7 | 47.0 | 34.4 |
| Engelmann 2018 | 79.7 | 57.6 | 35.6 |
| Engelmann 2017 | 80.6 | 49.7 | 36.2 |
| 3P-RNN 2018 | **87.8** | 54.1 | 41.6 |
| Graph-LPE + SPG (ours) | 85.2 | **62.4** | **49.7** |

**Table:** vKITTI

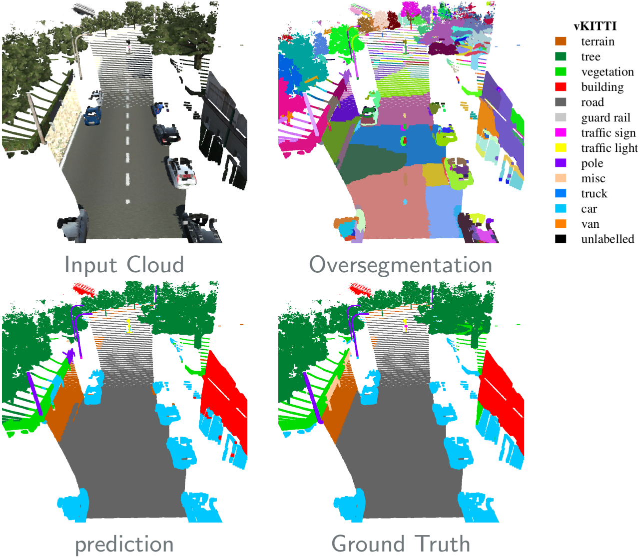
# Illustration




Input Cloud | Oversegmentation

prediction | Ground Truth

# Illustration



# Presentation Layout


1. Deep Learning for 3D Point Clouds
2. Learning 3D Point Clouds Segmentation
3. **The Cut Pursuit Algorithm**
4. Conclusion
5. Bibliography

# the Cut-Pursuit Algorithm

- A working-set approach to graph-structured spatial regularization

## the Cut-Pursuit Algorithm

- A working-set approach to graph-structured spatial regularization
- Joint work with Guillaume Obozinski and Hugo Raguet

L. Landrieu and G. Obozinski. Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions. In AISTATS, 2016

L. Landrieu and G. Obozinski. Cut pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs. SIAM Journal on Imaging Sciences, 2017

H. Raguet and L. Landrieu. Cut-pursuit Algorithm for Regularizing Nonsmooth Functionals With Graph Total Variation. In ICML, 2018

# the Cut-Pursuit Algorithm

- A working-set approach to graph-structured spatial regularization
- Joint work with Guillaume Obozinski and Hugo Raguet
- Initially designed for graph-total variation minimization

L. Landrieu and G. Obozinski. Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions. In AISTATS, 2016
L. Landrieu and G. Obozinski. Cut pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs. SIAM Journal on Imaging Sciences, 2017
H. Raguet and L. Landrieu. Cut-pursuit Algorithm for Regularizing Nonsmooth Functionals With Graph Total Variation. In ICML, 2018

## the Cut-Pursuit Algorithm

- A working-set approach to graph-structured spatial regularization
- Joint work with Guillaume Obozinski and Hugo Raguet
- Initially designed for graph-total variation minimization
- Can be generalized to the nonconvex setting of the GMMP.

L. Landrieu and G. Obozinski. Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions. In AISTATS, 2016

L. Landrieu and G. Obozinski. Cut pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs. SIAM Journal on Imaging Sciences, 2017

H. Raguet and L. Landrieu. Cut-pursuit Algorithm for Regularizing Nonsmooth Functionals With Graph Total Variation. In ICML, 2018

## the Cut-Pursuit Algorithm

- A working-set approach to graph-structured spatial regularization
- Joint work with Guillaume Obozinski and Hugo Raguet
- Initially designed for graph-total variation minimization
- Can be generalized to the nonconvex setting of the GMMP.
- **Main Idea:** exploiting the coarseness of the solutions of such problem.

L. Landrieu and G. Obozinski. Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions. In AISTATS, 2016

L. Landrieu and G. Obozinski. Cut pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs. SIAM Journal on Imaging Sciences, 2017

H. Raguet and L. Landrieu. Cut-pursuit Algorithm for Regularizing Nonsmooth Functionals With Graph Total Variation. In ICML, 2018

# Objective

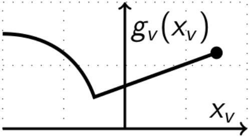
$$x^\star = \arg\min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w_{(u,v)} |x_u - x_v|$$

differentiable

dir. derivative in $]-\infty, \infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

graph total variation

- Optimization problem strutured by $G = (V, E, w)$



# Objective

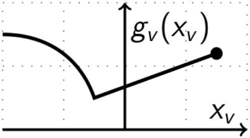
$$x^\star = \arg\min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w_{(u,v)} \left| x_u - x_v \right|$$

differentiable

dir. derivative in $]-\infty, \infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

graph total variation

- Optimization problem strutured by $G = (V, E, w)$
- Fairly general formulation



# Objective

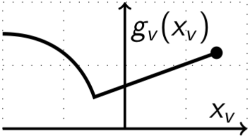
$$x^\star = \arg\min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w_{(u,v)} \left| x_u - x_v \right|$$

differentiable

dir. derivative in $]-\infty, \infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

graph total variation

- Optimization problem strutured by $G = (V, E, w)$
- Fairly general formulation
- Includes inverse problems: $f(x) = \|Ax - y\|^2$



# Objective

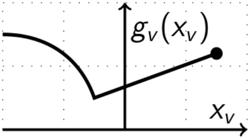
$$x^\star = \arg\min_{x\in\mathbb{R}^V} f(x) + \sum_{v\in V} g_v(x) + \sum_{(u,v)\in E} w_{(u,v)} |x_u - x_v|$$

differentiable

dir. derivative in $]-\infty, \infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

graph total variation

- Optimization problem strutured by $G = (V, E, w)$
- Fairly general formulation
- Includes inverse problems: $f(x) = \|Ax - y\|^2$
- L1 fidelity: $f(x) = 0$, $g_v(x) = |x_v - y_v|$



## Objective

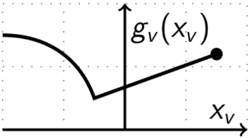
$$x^\star = \arg\min_{x\in\mathbb{R}^V} f(x) + \sum_{v\in V} g_v(x) + \sum_{(u,v)\in E} w_{(u,v)} |x_u - x_v|$$

differentiable

dir. derivative in $]-\infty,\infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

graph total variation

- Optimization problem strutured by $G = (V, E, w)$
- Fairly general formulation
- Includes inverse problems: $f(x) = \|Ax - y\|^2$
- L1 fidelity: $f(x) = 0$, $g_v(x) = |x_v - y_v|$
- Fused lasso regularization: $g_v(x) = |x_v|$



# Objective

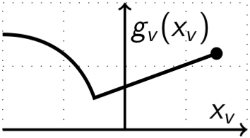
$$x^\star = \arg\min_{x\in\mathbb{R}^V} f(x) + \sum_{v\in V} g_v(x) + \sum_{(u,v)\in E} w_{(u,v)} |x_u - x_v|$$

differentiable

dir. derivative in $]-\infty,\infty]$
ex: $|\cdot|, \iota_\Omega(\cdot)$

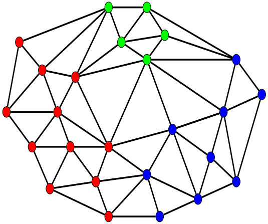
graph total variation

- Optimization problem strutured by $G = (V, E, w)$
- Fairly general formulation
- Includes inverse problems: $f(x) = \|Ax - y\|^2$
- L1 fidelity: $f(x) = 0$, $g_v(x) = |x_v - y_v|$
- Fused lasso regularization: $g_v(x) = |x_v|$
- No convexity requirement.



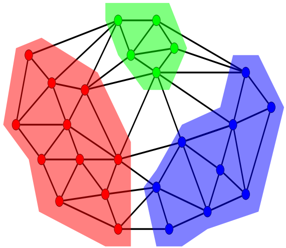
# Motivation

- TV regularization ⇒ solution piecewise constant.



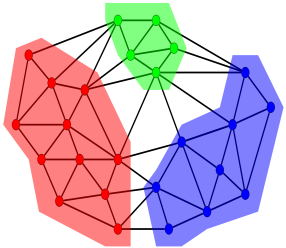
## Motivation

- TV regularization ⇒ solution piecewise constant.



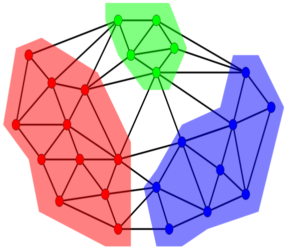
## Motivation

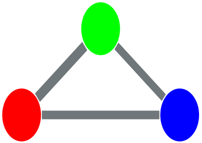
- TV regularization ⇒ solution piecewise constant.
- What if we knew this partition in advance?



## Motivation

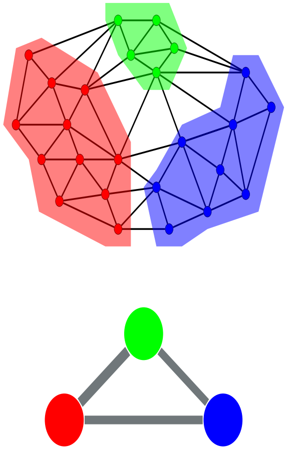
- TV regularization ⇒ solution piecewise constant.
- What if we knew this partition in advance?
- We could solve the problem on a much smaller **reduced graph**.





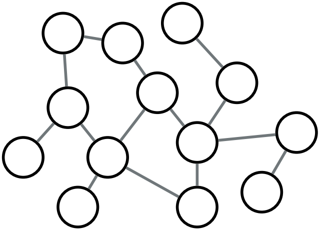
## Motivation

- TV regularization $\Rightarrow$ solution piecewise constant.
- What if we knew this partition in advance?
- We could solve the problem on a much smaller **reduced graph**.
- TV regularization constrained to piecewise constant solutions wrt a partition of $G \Leftrightarrow$ TV regularization wrt. the reduced graph.



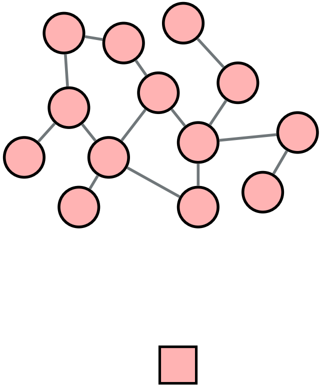
## Principle

1 • Start with a trivial partition $P = \{V\}$



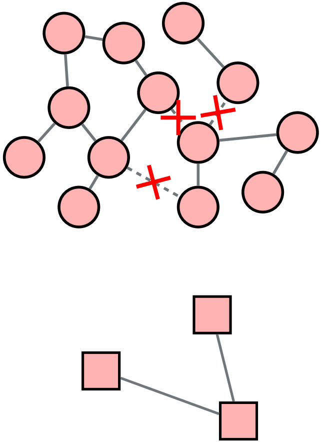
## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$



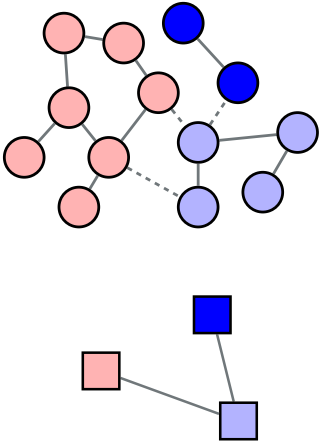
## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. • Refine current partition $P$



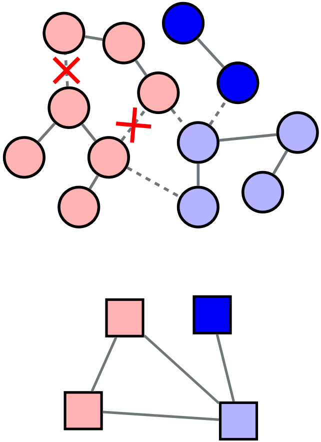
## Principle

1 Start with a trivial partition $P = \{V\}$
2 • Solve problem on reduced graph induced by $P$
3 Refine current partition $P$



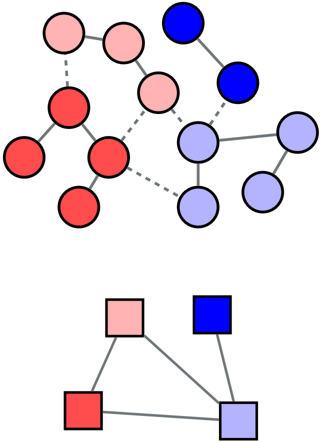
## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. • Refine current partition $P$



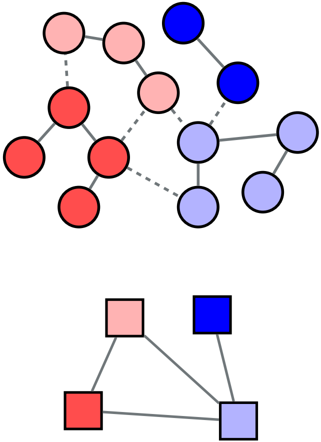
## Principle

1. Start with a trivial partition $P = \{V\}$
2. • Solve problem on reduced graph induced by $P$
3. Refine current partition $P$



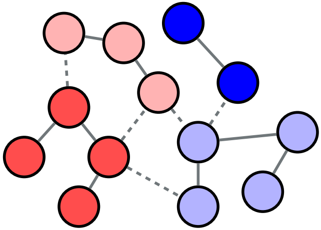
## Principle

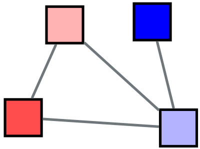
1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$



## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
4. • Critical point found.

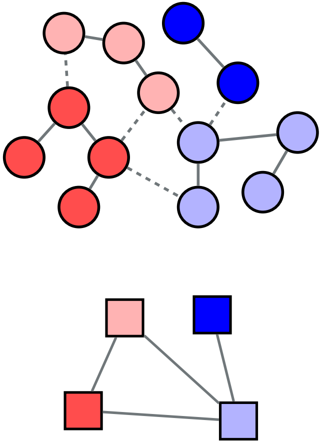




## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
4. Critical point found.

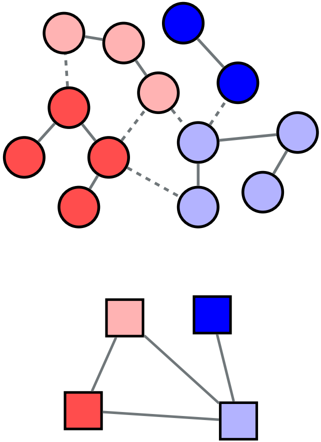
- Provable convergence in finite number of steps.



## Principle

1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
4. Critical point found.

- Provable convergence in finite number of steps.
- In practice only a few iterations necessary.



## Refinement step

- **Objective:** add degrees of liberty to the reduced problem to decrease $F$ as much as possible

# Refinement step

- **Objective:** add degrees of liberty to the reduced problem to decrease $F$ as much as possible
- **Solution:** use first order information at current solution $x$ to split along a steep descent direction

$$\text{find} \quad d^{(x)} \in \operatorname*{arg\,min}_{d \in D^V} F'(x, d) \ ,$$

## Refinement step

- **Objective:** add degrees of liberty to the reduced problem to decrease $F$ as much as possible
- **Solution:** use first order information at current solution $x$ to split along a steep descent direction

$$\text{find} \quad d^{(x)} \in \arg\min_{d \in D^V} F'(x, d) ,$$

with directional derivability:

$$F'(x, d) = \sum_{\substack{v \in V \\ d_v > 0}} \delta_v^+(x) - \sum_{\substack{v \in V \\ d_v < 0}} \delta_v^-(x) + \sum_{\substack{(u,v) \in E \\ x_u = x_v}} w_{(u,v)} |d_u - d_v| \ .$$

## Refinement step

- **Objective:** add degrees of liberty to the reduced problem to decrease $F$ as much as possible
- **Solution:** use first order information at current solution $x$ to split along a steep descent direction

$$\text{find} \quad d^{(x)} \in \underset{d \in D^V}{\arg\min}\ F'(x, d)\ ,$$

with directional derivability:

$$F'(x,d) = \sum_{\substack{v \in V \\ d_v > 0}} \delta_v^+(x) - \sum_{\substack{v \in V \\ d_v < 0}} \delta_v^-(x) + \sum_{\substack{(u,v) \in E \\ x_u = x_v}} w_{(u,v)} \left| d_u - d_v \right|\ .$$

- **In practice:** pick steepest direction in finite set $D^V$:

**Direction set:**

smooth case ($g_v = 0$ for all $v \in V$): $D = \{-1, +1\}$
nonsmooth case: $D = \{-1, 0, +1\}$
Steepest direction as a graph cut problem.

## Implementation and variants

- Reduced problem: proximal algorithm (Preconditoned Forward Douglas-Rachford) on **reduced graph**

H. Raguet and L. Landrieu. Preconditioning of a Generalized Forward-Backward Splitting and Application to Optimization on Graphs. SIAM Journal on Imaging Sciences, 2015.

## Implementation and variants

- Reduced problem: proximal algorithm (Preconditoned Forward Douglas-Rachford) on **reduced graph**
- Refinement: graph cut on full graph with Boykov's augmenting path.

H. Raguet and L. Landrieu. Preconditioning of a Generalized Forward-Backward Splitting and Application to Optimization on Graphs. SIAM Journal on Imaging Sciences, 2015.

## Implementation and variants

- Reduced problem: proximal algorithm (Preconditoned Forward Douglas-Rachford) on **reduced graph**
- Refinement: graph cut on full graph with Boykov's augmenting path.
- Can be extended to multidimensional data (heuristic).

H. Raguet and L. Landrieu. Preconditioning of a Generalized Forward-Backward Splitting and Application to Optimization on Graphs. SIAM Journal on Imaging Sciences, 2015.

## Implementation and variants

- Reduced problem: proximal algorithm (Preconditoned Forward Douglas-Rachford) on **reduced graph**
- Refinement: graph cut on full graph with Boykov's augmenting path.
- Can be extended to multidimensional data (heuristic).
- Can be extended to the GMPP (heuristic).

H. Raguet and L. Landrieu. Preconditioning of a Generalized Forward-Backward Splitting and Application to Optimization on Graphs. SIAM Journal on Imaging Sciences, 2015.

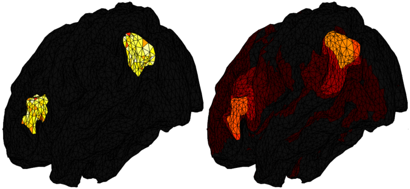
## Implementation and variants

- Reduced problem: proximal algorithm (Preconditoned Forward Douglas-Rachford) on **reduced graph**
- Refinement: graph cut on full graph with Boykov's augmenting path.
- Can be extended to multidimensional data (heuristic).
- Can be extended to the GMPP (heuristic).
- Can be fully parallelized, even the graph cuts-based phase.

H. Raguet and L. Landrieu. Preconditioning of a Generalized Forward-Backward Splitting and Application to Optimization on Graphs. SIAM Journal on Imaging Sciences, 2015.

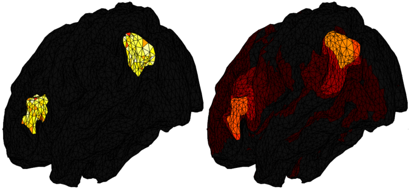
## EEG Experiment

- EEG : from 96 electrods to ∼20.000 triangles



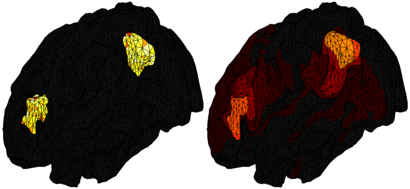
# EEG Experiment

- EEG : from 96 electrods to $\sim$20.000 triangles
- Underdetermined, ill-conditioned inverse problem



## EEG Experiment

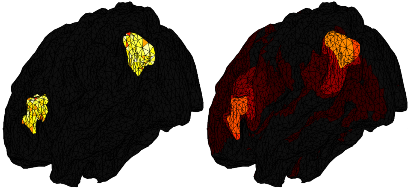
- EEG : from 96 electrods to ∼20.000 triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,



# EEG Experiment

- EEG : from 96 electrods to ∼20.000 triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,

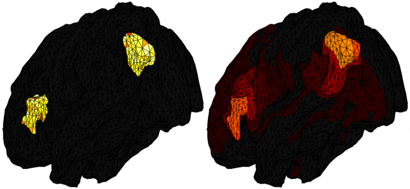
$$F : x \mapsto \frac{1}{2}\|y - \Phi x\|^2 + \sum_{v \in V} \left(\lambda_v |x_v| + \iota_{\mathbb{R}_+}(x_v)\right) + \sum_{(u,v) \in E} w_{(u,v)} |x_u - x_v| \ ,$$



# EEG Experiment

- EEG : from 96 electrods to ∼20.000 triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,
- Very coarse ground truth

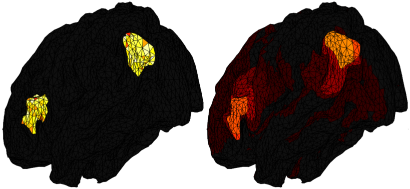
$$F : x \mapsto \tfrac{1}{2}\|y - \Phi x\|^2 + \sum_{v \in V} \left(\lambda_v |x_v| + \iota_{\mathbb{R}_+}(x_v)\right) + \sum_{(u,v)\in E} w_{(u,v)} |x_u - x_v| \ ,$$

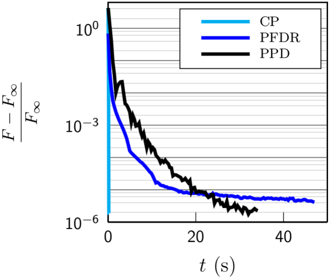


# EEG Experiment

- EEG : from 96 electrods to ∼20.000 triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,
- Very coarse ground truth

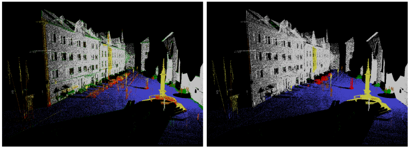
$$F : x \mapsto \frac{1}{2}\|y - \Phi x\|^2 + \sum_{v \in V} \left(\lambda_v |x_v| + \iota_{\mathbb{R}_+}(x_v)\right) + \sum_{(u,v)\in E} w_{(u,v)} |x_u - x_v| \ ,$$





# Semantic Segmentation Experiment

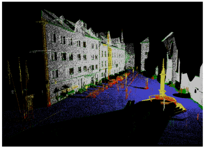
- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)



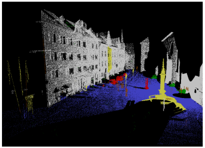
(b) Random forest predictions (c) Regularization

## Semantic Segmentation Experiment

- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)
- A probability vector for each vertex



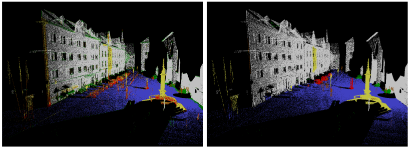
(b) Random forest predictions



(c) Regularization

# Semantic Segmentation Experiment

- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)
- A probability vector for each vertex
- KL-fidelity, simplex-bound, smoothness prior

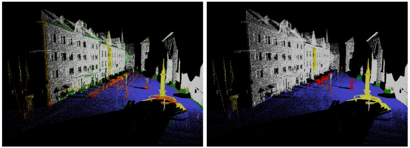


(b) Random forest predictions (c) Regularization

## Semantic Segmentation Experiment

- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)
- A probability vector for each vertex
- KL-fidelity, simplex-bound, smoothness prior

$$F : p \mapsto \sum_{v \in V} \mathrm{KL}\left(q_v, p_v\right) + \sum_{v \in V} \iota_{\triangle_K}(p_v) + \sum_{(u,v) \in E} w_{(u,v)} \|p_u - p_v\|_1 \ ,$$

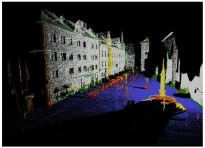


(b) Random forest predictions (c) Regularization

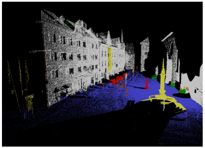
# Semantic Segmentation Experiment

- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)
- A probability vector for each vertex
- KL-fidelity, simplex-bound, smoothness prior

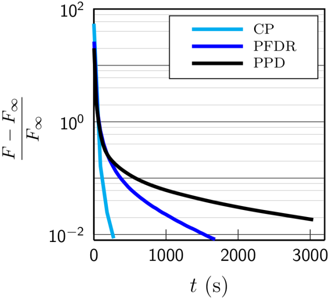
$$F : p \mapsto \sum_{v \in V} \mathrm{KL}\,(q_v, p_v) + \sum_{v \in V} \iota_{\triangle_K}(p_v) + \sum_{(u,v) \in E} w_{(u,v)} \|p_u - p_v\|_1 \ ,$$



(b) Random forest predictions



(c) Regularization



# Presentation Layout


1. Deep Learning for 3D Point Clouds
2. Learning 3D Point Clouds Segmentation
3. The Cut Pursuit Algorithm
4. Conclusion
5. Bibliography

# Conclusion

- Our paradigm for graph-structured learning and optimization:
  - Exploit the spatial regularity of the solution to increase speed and precision.

# Conclusion

- Our paradigm for graph-structured learning and optimization:
  - Exploit the spatial regularity of the solution to increase speed and precision.
  - Use neural networks to learn the inputs and parameters of efficient optimization algorithms.

# Conclusion

- Our paradigm for graph-structured learning and optimization:
  - Exploit the spatial regularity of the solution to increase speed and precision.
  - Use neural networks to learn the inputs and parameters of efficient optimization algorithms.
  - Use graph-structured optimization to compute the structure of neural network adapted to the data.

## Conclusion

- Our paradigm for graph-structured learning and optimization:
  - Exploit the spatial regularity of the solution to increase speed and precision.
  - Use neural networks to learn the inputs and parameters of efficient optimization algorithms.
  - Use graph-structured optimization to compute the structure of neural network adapted to the data.
- All our work is online:
  - `loicland/superpoint-graph` 252 ★ 75
  - `loicland/cut-pursuit` 22 ★ 7
  - `1a7r0ch3/parallel-cut-pursuit` very soon!

# Presentation Layout

1 Deep Learning for 3D Point Clouds

2 Learning 3D Point Clouds Segmentation

3 The Cut Pursuit Algorithm

4 Conclusion

5 **Bibliography**

# Bibliography I

**Qi et. al.2017a** Qi, C. R., Su, H., Mo, K., & Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR, 2017
**Gaidon2016** Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. Virtual worlds as proxy for multi-object tracking analysis. ,CVPR2016.
**Engelmann2017** Engelmann, F., Kontogianni, T., Hermans, A. & Leibe, B. Exploring spatial context for 3d semantic segmentation of point clouds. CVPR, DRMS Workshop, 2017.
**Hackel2017i** Timo Hackel and N. Savinov and L. Ladicky and Jan D. Wegner and K. Schindler and M. Pollefeys, SEMANTIC3D.NET: A new large-scale point cloud classification benchmark,ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,2017
**Armeni2016** Iro Armeni and Ozan Sener and Amir R. Zamir and Helen Jiang and Ioannis Brilakis and Martin Fischer and Silvio Savarese, 3D Semantic Parsing of Large-Scale Indoor Spaces, CVPR, 2016
**Demantke2011** Demantke, J., Mallet, C., David, N. & Vallet, B. Dimensionality based scale selection in 3D lidar point clouds. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2011.
**Weinmann2015** Weinmann, M., Jutzi, B., Hinz, S. & Mallet, C., Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS Journal of Photogrammetry and Remote Sensing, 2015.
**Landrieu2017a** Landrieu, L., Raguet, H., Vallet, B., Mallet, C., & Weinmann, M. A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 2017.

# Bibliography II


**Boulch2017** Boulch, Alexandre, Le Saux, Bertrand, and Audebert, Nicolas, Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks, 3DOR, 2017.

**Wu2015** Wu, Z., Song, S. Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. 3D shapenets: A deep representation for volumetric shapes. CVPR, 2015.

**Riegler2017** Riegler, G., Osman Ulusoy, A., & Geiger, A. Octnet: Learning deep 3d representations at high resolutions, CVPR, 2017.

**Tchapmi2017** Tchapmi, L., Choy, C., Armeni, I., Gwak, J. & Savarese, S., Segcloud: Semantic segmentation of 3d point clouds. 3DV, 2017.

**Jampani2018**, Jampani Su, H., , V. Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., & Kautz, J., Splatnet: Sparse lattice networks for point cloud processing. CVPR2018.

**Tatarchenko2018** Tatarchenko, M., Park, J., Koltun, V., & Zhou, Q. Y. Tangent Convolutions for Dense Prediction in 3D. CVPR, 2018

**Li2018** Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. PointCNN: Convolution On $\chi$-Transformed Points. NIPS, 2018.

**Qi2017** Qi, X., Liao, R., Jia, J., Fidler, S., & Urtasun, R. 3D Graph Neural Networks for RGBD Semantic Segmentation. In PCVPR, 2017.

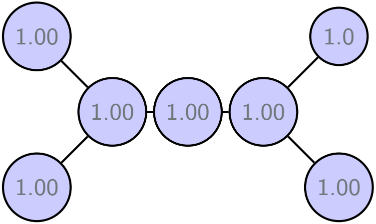
**Simonovsky2017** Simonovsky, M., & Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. CVPR, 2017.

**Landrieu&Simonovski2018** Landrieu, L., & Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. CVPR, 2018

**Qi et. al.2017b** Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems (pp. 5099-5108).

## Non-differentiability of the naive pipeline

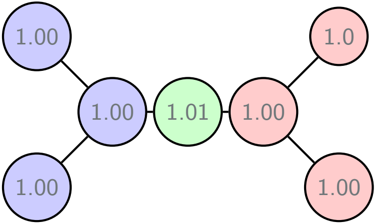
- Non differentiability of the CCC operator



# Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator

= Tiny changes - large consequence



## Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator

= Tiny changes - large consequence

- Non differentiability of $f^\star(e)$

$$f^\star = \arg\min \|f_0 - e_0\|^2 + \|x_1 - e_1\|^2 + 0.5[f_0 \neq f_1]$$

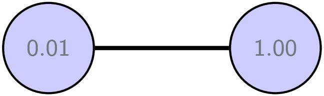
## Non-differentiability of the naive pipeline

$$f^\star = \arg\min \|f_0 - e_0\|^2 + \|x_1 - e_1\|^2 + 0.5[f_0 \neq f_1]$$

- Non differentiability of the CCC operator

= Tiny changes - large consequence

- Non differentiability of $f^\star(e)$




$$f_0^\star = 0.505, \quad f_1^\star = 0.505$$

## Non-differentiability of the naive pipeline

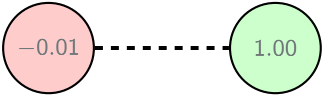
$$f^\star = \arg\min \|f_0 - e_0\|^2 + \|x_1 - e_1\|^2 + 0.5[f_0 \neq f_1]$$

- Non differentiability of the CCC operator

= Tiny changes - large consequence

- Non differentiability of $f^\star(e)$

= non-continuous w.r.t inputs




$$f_1^\star = -0.01, \quad f_1^\star = 1.00$$