Semantic Segmentation of 3D point Clouds

Loic Landrieu

Université Paris-Est - Machine Learning and Optimization working Group

March 2019
Loic Landrieu, researcher at IGN (French Mapping Agency) in the AI department
Loïc 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.
Loïc 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
Deep Learning for 3D Point Clouds

Learning 3D Point Clouds Segmentation

The Cut Pursuit Algorithm

Conclusion

Bibliography
Deep Learning for 3D Point Clouds

1. 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
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

Deep Learning for 3D Point Clouds
Presentation of the Problem
Future trends

- LiDAR are getting cheaper: $100k → 2k$ in a few years.
Future trends

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

- LiDAR are getting cheaper: $100k \rightarrow 2k$ 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: $100k → 2k$ 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: $100k \rightarrow 2k$ 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 hardware 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:
  
  $$P_i \mapsto [1, \ldots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between $K$ classes:
  
  $$P_i \mapsto [1, \ldots, K]$$

**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
Deep Learning for 3D Point Clouds

- Presentation of the Problem
- Traditional Approaches
- First Deep-Learning Approaches
- Scaling Segmentation

Learning 3D Point Clouds Segmentation

The Cut Pursuit Algorithm

Conclusion

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

SnapNet:

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
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
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

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)

**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.


*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.

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 fundamental constraint: inputs are invariant by permutation

Qi et. al. 2017a
PointNet

- **A fundamental 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 fundamental 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.

\[ \mathbf{h}_i^{(t+1)} = g \left( \sum_{j \rightarrow i} f \left( \mathbf{h}_j^{(t)}, \mathbf{h}_i^{(t)} \right), \mathbf{h}_i^{(t)} \right) \]

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

\[ \mathbf{h}_i^{(t+1)} = g \left( \sum_{j \rightarrow i} \Theta_{ij}, \mathbf{h}_i^{(t)}, \mathbf{h}_i^{(t)} \right) \]
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

Deep Learning for 3D Point Clouds
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 \( g \): 
  \[
  h_i^{(t+1)} = g\left(\sum_j \Theta_{ij} \otimes f(h_i^t, h_j^t)\right)
  \]
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 $g$:

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

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

  $$h_i^{(t+1)} = g\left(\sum_{j\rightarrow i} \Theta_{i,j} \odot h_i^t, h_i^t\right)$$

Qi2017, Simonovski2017

credit: Simonovski2017
Deep Learning for 3D Point Clouds
- Presentation of the Problem
- Traditional Approaches
- First Deep-Learning Approaches
- Scaling Segmentation

Learning 3D Point Clouds Segmentation

The Cut Pursuit Algorithm

Conclusion

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 work 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
PointNet++

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

Qi et. al. 2017b

credit: Qi et. al. 2017b
Observation:

\[ n_{\text{points}} \gg n_{\text{objects}}. \]
**SuperPoint-Graph**

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

- Partition scene into superpoints with simple shapes.

Landrieu & Simonovski 2018
**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:
  - Superpoint embedding: learning shape descriptors. Complexity: low (subsampling to $128 \times \sim 1000$ 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

- 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
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
Qualitative Results: Semantic3D

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
Qualitative Results: Semantic3D

Semantic3D: 3 billions points over 30 clouds
Qualitative Results: Semantic3D

Semantic3D: 3 billions points over 30 clouds
Quantitative Results: Semantic3D

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

![Diagram showing indoor scene with labeled objects]

- ceiling
- ground
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- board
- other
Résultats qualitatif: S3DIS

ceiling
ground
wall
column
beam
window
door
table
chair
bookcase
board
board
other
### Quantitative Results: S3DIS

<table>
<thead>
<tr>
<th>Method</th>
<th>OA</th>
<th>mAcc</th>
<th>mIoU</th>
<th>door</th>
<th>board</th>
</tr>
</thead>
<tbody>
<tr>
<td>A5 PointNet</td>
<td>–</td>
<td>48.5</td>
<td>41.1</td>
<td>10.7</td>
<td>26.3</td>
</tr>
<tr>
<td>A5 SEGCloud</td>
<td>–</td>
<td>57.3</td>
<td>48.9</td>
<td>23.1</td>
<td>13.0</td>
</tr>
<tr>
<td><strong>A5 SPG</strong></td>
<td>86.4</td>
<td>66.5</td>
<td>58.0</td>
<td><strong>61.5</strong></td>
<td>2.1</td>
</tr>
<tr>
<td>PointNet</td>
<td>78.5</td>
<td>66.2</td>
<td>47.6</td>
<td>51.6</td>
<td>29.4</td>
</tr>
<tr>
<td>Engelmann</td>
<td>81.1</td>
<td>66.4</td>
<td>49.7</td>
<td>51.2</td>
<td><strong>30.0</strong></td>
</tr>
<tr>
<td><strong>SPG</strong></td>
<td>85.5</td>
<td>73.0</td>
<td>62.1</td>
<td><strong>68.4</strong></td>
<td>8.7</td>
</tr>
</tbody>
</table>
## Quantitative Results: S3DIS

<table>
<thead>
<tr>
<th>Method</th>
<th>OA</th>
<th>mAcc</th>
<th>mIoU</th>
<th>door</th>
<th>board</th>
</tr>
</thead>
<tbody>
<tr>
<td>A5 PointNet</td>
<td>86.4</td>
<td>66.5</td>
<td>58.0</td>
<td>61.5</td>
<td>2.1</td>
</tr>
<tr>
<td>A5 SEGCloud</td>
<td>57.3</td>
<td>48.9</td>
<td>23.1</td>
<td>13.0</td>
<td></td>
</tr>
<tr>
<td>A5 SPG</td>
<td>78.5</td>
<td>66.2</td>
<td>47.6</td>
<td>51.6</td>
<td>29.4</td>
</tr>
<tr>
<td>PointNet</td>
<td>81.1</td>
<td>66.4</td>
<td>49.7</td>
<td>51.2</td>
<td>30.0</td>
</tr>
<tr>
<td>Engelmann</td>
<td>85.5</td>
<td>73.0</td>
<td>62.1</td>
<td>68.4</td>
<td>8.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Full cloud</th>
<th>2 cm</th>
<th>3 cm</th>
<th>4 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voxelisation</td>
<td>0</td>
<td>40</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>Features</td>
<td>439</td>
<td>194</td>
<td>88</td>
<td>43</td>
</tr>
<tr>
<td>Partition</td>
<td>3428</td>
<td>1013</td>
<td>447</td>
<td>238</td>
</tr>
<tr>
<td>SPG computation</td>
<td>3800</td>
<td>958</td>
<td>436</td>
<td>252</td>
</tr>
<tr>
<td>Inference ×10</td>
<td>240</td>
<td>110</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td>7907</td>
<td>2315</td>
<td>1055</td>
<td>599</td>
</tr>
</tbody>
</table>

mIoU 6-fold: 54.1, 60.2, 62.1, 57.1
Superpoint Partition

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

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

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

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

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

- \( e \in \mathbb{R}^{C \times m} \): 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...
1 Deep Learning for 3D Point Clouds

2 Learning 3D Point Clouds Segmentation

3 The Cut Pursuit Algorithm

4 Conclusion

5 Bibliography
The Pipeline

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

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 almost automatically object purity!
Adjacency Graph

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

$E_{inter}$: set of inter-object edges

$E_{intra}$: set of intra-object edges

We want embeddings with high contrast at $E_{inter}$ and similar value at $E_{intra}$. If we get $E_{inter}$ right, then we have almost object purity!
• $G = (C, E)$ 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 embeddings

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

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

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

- 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 embeddings

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

- 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 embeddings

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

- 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 embeddings

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

- 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

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

- Let consider our pipeline:
The Problem With the GMPP

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

- Let consider our pipeline:
  - Let \( x \) be the parameters of the Local Point Embedder
The Problem With the GMPP

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

- 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

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

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

- Let $x$ be the parameters of the Local Point Embedder
- Let $e(x)$ be the resulting embeddings
- Let $f^*(e(x))$ be the solution of the GMPP
- Let $CCC$ the constant connected component operator on $G$
- The superpoints are: $S = CCC(f^*(e(x)))$
The Problem With the GMPP

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

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

- Let \( M(S) \) be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)

Naive Approach:
\[ \ell(x) = -M(CCC(f^*(e(x)))) \]
To backpropagate we need:
\[ \frac{\partial CCC}{\partial f^*} \text{ and } \frac{\partial f^*}{\partial e(x)} \]
Problem: Those functions are not backpropagable.
The Problem With the GMPP

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

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

- Let \( M(S) \) be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)

**Naive Approach:** \( \ell(x) = -M(CCC(f^*(e(x)))) \)
\[ f^* = \arg \min_{f \in \mathbb{R}^{C \times m}} \sum_{i \in C} \| f_i - e_i \|^2 + \sum_{(i,j) \in E} w_{i,j} [f_i \neq f_j], \]

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

- Let \( M(S) \) be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)

**Naive Approach:** \( \ell(x) = -M(\text{CCC}(f^*(e(x)))) \)

- To backpropagate we need: \( \frac{\partial \text{CCC}}{\partial f^*} \) and \( \frac{\partial f^*}{\partial e} \)
The Problem With the GMPP

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

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

- Let \( M(S) \) be a measure of how good an oversegmentation is (implementing purity, border recall, etc...)

- **Naive Approach:** \( \ell(x) = -M(CCC(f^*(e(x)))) \)

- To backpropagate we need: \( \frac{\partial CCC}{\partial f^*} \) and \( \frac{\partial f^*}{\partial e} \)

- **Problem:** Those functions are not backpropagable.
Graph-Structured Contrastive Loss

- We propose a surrogate loss to learn meaningful embeddings

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

\( \phi(x) = \delta(\sqrt{\|x\|^2}/\delta^2 + 1 - 1) \)

\( \psi(x) = \max(1 - \|x\|, 0) \)

Promotes homogeneity within objects and contrast at their borders

\( \mu_{i,j} \): weight of inter-edges
Graph-Structured Contrastive Loss

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

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

- \(\phi\) minimum at 0, \(\psi\) maximum at 0

\[
\phi(x) = \delta(\sqrt{\|x\|^2/\delta^2 + 1} - 1)
\]

\[
\psi(x) = \max(1 - \|x\|, 0)
\]

Promotes homogeneity within objects and contrast at their borders

\(\mu_{i,j}\): weight of inter-edges
Graph-Structured Contrastive Loss

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

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

- \( \phi \) minimum at 0, \( \psi \) maximum at 0

\[
\phi(x) = \delta\left(\sqrt{\|x\|^2/\delta^2 + 1} - 1 \right)
\]

\[
\psi(x) = \max\left(1 - \|x\|, 0\right)
\]

- Promotes homogeneity within objects and contrast at their borders
Graph-Structured Contrastive Loss

- We propose a surrogate loss to learn meaningful embeddings

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

- \( \phi \) minimum at 0, \( \psi \) maximum at 0

\[
\phi(x) = \delta \left( \sqrt{\|x\|^2/\delta^2 + 1} - 1 \right)
\]

\[
\psi(x) = \max(1 - \|x\|, 0)
\]

- Promotes homogeneity within objects and contrast at their borders

- \( \mu_{i,j} \) : weight of inter-edges
Cross-Partition Weighting Strategy, cont’d

\[ \mu_{U,V} = \mu \frac{\min(|U|, |V|)}{|(U,V)|} \quad \text{for } (U, V) \in \mathcal{E} \]

\[ \mu_{i,j} = \mu_{U,V} \quad \text{for all } (i,j) \in (U, V) \]

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

Diagram:
- Superpoint
- Majority object
- Trespassing
- Interface

\[ \mu_{LW,LD} = \]
\[ \mu_{RW,RD} = \]
Results

We require 5 times less superpoints for similar performance!
Learning 3D Point Clouds Segmentation
Illustration

Input cloud

Ground truth objects

LPE embeddings

Graph-LPE (ours)

VCCS, Papon et al. 2013

Lin et al. 2018
## Results

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

**Table: S3DIS**

<table>
<thead>
<tr>
<th>Method</th>
<th>OA</th>
<th>mAcc</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet 2017</td>
<td>79.7</td>
<td>47.0</td>
<td>34.4</td>
</tr>
<tr>
<td>Engelmann 2018</td>
<td>79.7</td>
<td>57.6</td>
<td>35.6</td>
</tr>
<tr>
<td>Engelmann 2017</td>
<td>80.6</td>
<td>49.7</td>
<td>36.2</td>
</tr>
<tr>
<td>3P-RNN 2018</td>
<td><strong>87.8</strong></td>
<td>54.1</td>
<td>41.6</td>
</tr>
<tr>
<td>Graph-LPE + SPG (ours)</td>
<td>85.2</td>
<td><strong>62.4</strong></td>
<td><strong>49.7</strong></td>
</tr>
</tbody>
</table>

**Table: vKITTI**
Illustration

Input Cloud

Oversegmentation

prediction

Ground Truth

S3DIS
- ceiling
- floor
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- sofa
- board
- clutter
- unlabelled
Deep Learning for 3D Point Clouds

Learning 3D Point Clouds Segmentation

The Cut Pursuit Algorithm

Conclusion

Bibliography
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

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

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.

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


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

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

differentiable

dir. derivative in \([-\infty, \infty]\)

ex: \(\| \cdot \|, \iota_{\Omega}(\cdot)\)

- Optimization problem structured by \(G = (V, E, w)\)
**Objective**

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

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

- Optimization problem structured by \( G = (V, E, w) \)
- Fairly general formulation
Objective

\[ x^* = \arg \min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w(u,v) |x_u - x_v| \]

- Differentiable
  - Dir. derivative in \([-\infty, \infty]\)
  - Ex: \(|\cdot|, \iota_\Omega(\cdot)|

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

\[ x^* = \arg \min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w(u,v) |x_u - x_v| \]

- Optimization problem structured by \( G = (V, E, w) \)
- Fairly general formulation
- Includes inverse problems: \( f(x) = \|Ax - y\|^2 \)
- L1 fidelity: \( f(x) = 0, g_v(x) = |x_v - y_v| \)
- Differentiable direct derivative in \( ]-\infty, \infty[ \)
  
  *ex: \( |\cdot|, \iota_\Omega(\cdot) \)
Objective

\[ x^* = \arg \min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w(u,v) |x_u - x_v| \]

differentiable

dir. derivative in \([-\infty, \infty]\)

ex: \(|\cdot|, \iota_\Omega(\cdot)\)

graph total variation

- Optimization problem structured 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

\[ x^* = \arg \min_{x \in \mathbb{R}^V} f(x) + \sum_{v \in V} g_v(x) + \sum_{(u,v) \in E} w(u,v) |x_u - x_v| \]

differentiable

dir. derivative in \( ] - \infty, \infty [ \)

ex: \( |\cdot|, \iota_\Omega(\cdot) \)

- Optimization problem structured 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.

The Cut Pursuit Algorithm
- TV regularization ⇒ solution piecewise constant.
- TV regularization $\Rightarrow$ solution piecewise constant.
Motivation

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

The Cut Pursuit Algorithm
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**.
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**.
- TV regularization constrained to piecewise constant solutions wrt a partition of $G$ ⇔ TV regularization wrt. the reduced graph.
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 $P$
1. Start with a trivial partition $P = \{ V \}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
1. Start with a trivial partition \( P = \{ V \} \)
2. Solve problem on reduced graph induced by \( P \)
3. Refine current partition \( P \)
1. Start with a trivial partition $P = \{V\}$
2. Solve problem on reduced graph induced by $P$
3. Refine current partition $P$
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.
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

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

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

with directional derivability:

\[
F'(x, d) = \sum_{v \in V, d_v > 0} \delta^+_v(x) - \sum_{v \in V, d_v < 0} \delta^-_v(x) + \sum_{(u,v) \in E, x_u = x_v} w_{(u,v)} |d_u - d_v|
\]
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

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

with directional derivability:

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

- **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 (Preconditioned Forward Douglas-Rachford) on reduced graph

Implementation and variants

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

Implementation and variants

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

Implementation and variants

- Reduced problem: proximal algorithm (Preconditioned 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).

Implementation and variants

- Reduced problem: proximal algorithm (Preconditioned 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.

EEG Experiment

- EEG: from 96 electrodes to \( \sim 20,000 \) triangles
EEG Experiment

- EEG: from 96 electrodes to \( \sim 20,000 \) triangles
- Underdetermined, ill-conditioned inverse problem
EEG Experiment

- EEG: from 96 electrodes to $\sim 20,000$ triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,
EEG Experiment

- EEG: from 96 electrodes to $\sim 20,000$ triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,

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

- EEG : from 96 electrodes to $\sim 20,000$ triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,
- Very coarse ground truth

$$F : x \mapsto \frac{1}{2} \| y - \Phi x \|^2 + \sum_{v \in V} (\lambda_v |x_v| + \nu_{\mathbb{R}_+}(x_v)) + \sum_{(u,v) \in E} w_{(u,v)} |x_u - x_v|,$$
- EEG: from 96 electrodes to $\sim 20,000$ triangles
- Underdetermined, ill-conditioned inverse problem
- Sparsity, positivity, smoothness,
- Very coarse ground truth

$$F: x \mapsto \frac{1}{2} \| y - \Phi x \|^2 + \sum_{v \in V} (\lambda_v |x_v| + \nu_{\mathbb{R}^+}(x_v)) + \sum_{(u,v) \in E} w_{(u,v)} |x_u - x_v|,$$
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
- Spatial Regularization of pointwise probabilistic semantic segmentation $q$ (from local context)
- A probability vector for each vertex
- **KL-fidelity, simplex-bound, smoothness prior**
  \[ F: p \mapsto \sum_{v \in V} KL(q_v, p_v) + \sum_{v \in V} \iota \Delta_K(p_v) + \sum_{(u,v) \in E} w(u,v) \| p_u - p_v \|_1 , \]

(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**

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

(b) Random forest predictions          (c) Regularization

The Cut Pursuit Algorithm
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.
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!
Deep Learning for 3D Point Clouds

Learning 3D Point Clouds Segmentation

The Cut Pursuit Algorithm

Conclusion

Bibliography


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


Bibliography II


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


**Landrieu&Simonovsky2018** Landrieu, L., & Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. CVPR, 2018

Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator

\[
\text{Tiny changes} - \text{large consequence}
\]

\[
\text{Non differentiability of } f^\star(e) = \text{non-continuous w.r.t inputs}
\]

\[
f^\star = \arg \min \left\| f_0 - e_0 \right\|_2 + \left\| x_1 - e_1 \right\|_2 + 0.5 \quad \left[ f_0 \neq f_1 \right]
\]
Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator
  - Tiny changes - large consequence

\[
\text{Non differentiability of } f \ast (e) = \text{non-continuous w.r.t inputs}
\]

\[
f \ast = \arg \min_{f_0} \|f_0 - e_0\|^2 + \|x_1 - e_1\|^2 + 0.5 \quad \text{if } f_0 \neq f_1
\]
Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator
- Tiny changes - large consequence
- Non differentiability of $f^*(e)$

$$f^* = \arg \min \| f_0 - e_0 \|^2 + \| x_1 - e_1 \|^2 + 0.5[f_0 \neq f_1]$$
Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator

= Tiny changes - large consequence

- Non differentiability of $f^*(e)$

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

$$f_0^* = 0.505, \quad f_1^* = 0.505$$
Non-differentiability of the naive pipeline

- Non differentiability of the CCC operator
- Tiny changes - large consequence
- Non differentiability of $f^*(e)$
- Non-continuous w.r.t inputs

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

$$f^*_1 = -0.01, \quad f^*_1 = 1.00$$