

Deep Learning for 3D Point Cloud Semantic Segmentation

Loïc Landrieu

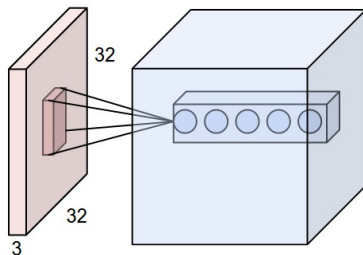
June 2019



- 1 Motivation
- 2 Deep-Learning Approaches
- 3 Scaling Segmentation
- 4 In Practice
- 5 Bibliography

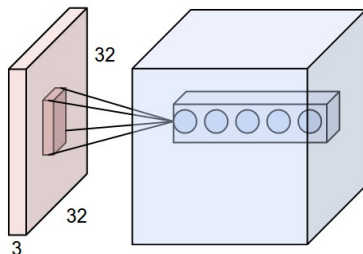
- 1 Motivation
- 2 Deep-Learning Approaches
- 3 Scaling Segmentation
- 4 In Practice
- 5 Bibliography

- *Overheard at a vision conference:*
"Computer Vision is a solved problem"
[not my words].



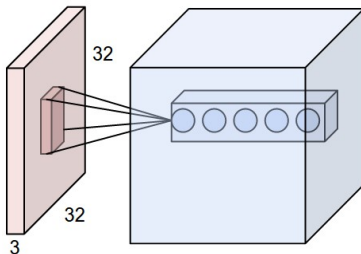
credit : pubs.sciepub.com/ajmm

- *Overheard at a vision conference:*
"Computer Vision is a solved problem"
[not my words].
- **Fact:** CNNs are very efficient at extracting visual features in images.



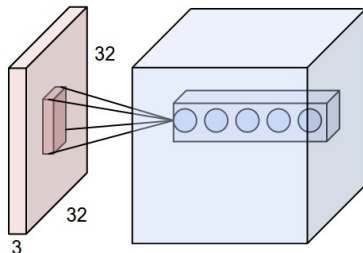
credit : pubs.sciepub.com/ajmm

- *Overheard at a vision conference:*
"Computer Vision is a solved problem"
[not my words].
- **Fact:** CNNs are very efficient at extracting visual features in images.
- **Fact:** has allowed tremendous improvements in many image-based tasks.



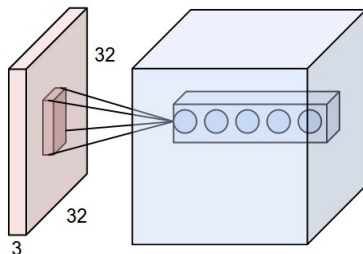
credit : pubs.sciepub.com/ajmm

- *Overheard at a vision conference:*
"Computer Vision is a solved problem"
[not my words].
- **Fact:** CNNs are very efficient at extracting visual features in images.
- **Fact:** has allowed tremendous improvements in many image-based tasks.
- **Fact:** In 2019, (almost) no vision paper on images rely on traditional methods.



credit : pubs.sciepub.com/ajmm

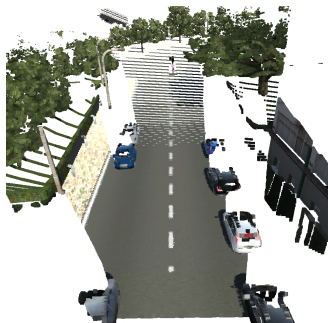
- *Overheard at a vision conference:*
"Computer Vision is a solved problem"
[not my words].
- **Fact:** CNNs are very efficient at extracting visual features in images.
- **Fact:** has allowed tremendous improvements in many image-based tasks.
- **Fact:** In 2019, (almost) no vision paper on images rely on traditional methods.
- **When is this "revolution" coming for the 3D community?**



credit : pubs.sciepub.com/ajmm

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**
 - Data volume considerable.

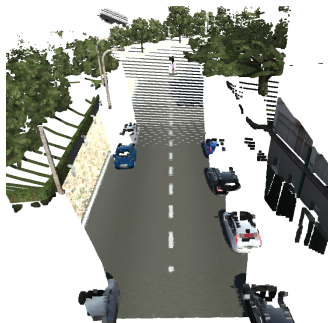


credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

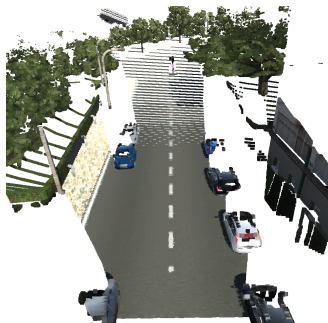


credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

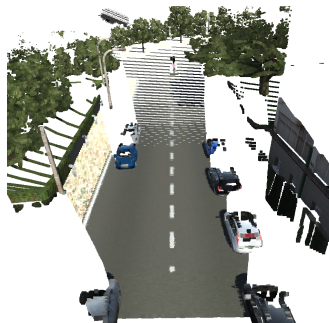


credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

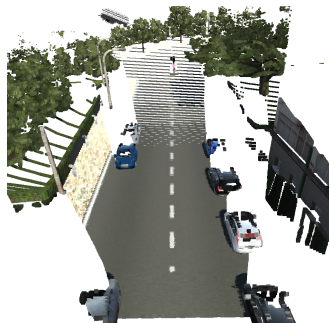


credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

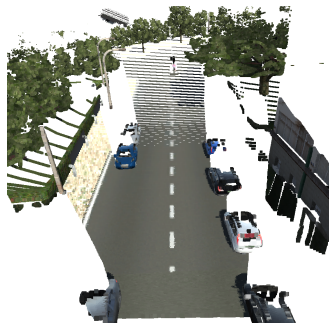


credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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



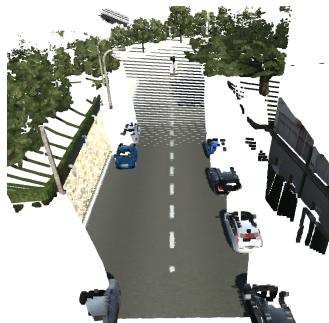
credit: Gaidon2016, Engelmann2017, Hackel2017

What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

credit: Gaidon2016, Engelmann2017, Hackel2017

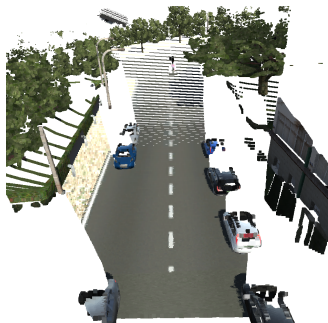


What Makes 3D Analysis so Hard?

- **3D data is much harder than images:**

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

credit: Gaidon2016, Engelmann2017, Hackel2017



1 Motivation

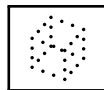
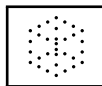
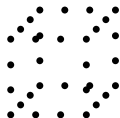
2 Deep-Learning Approaches

3 Scaling Segmentation

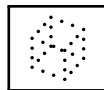
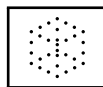
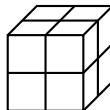
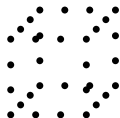
4 In Practice

5 Bibliography

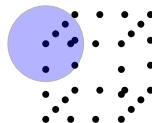
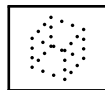
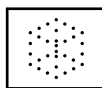
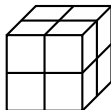
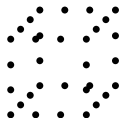
- Different ways of handling 3D point clouds:
 - As a set of images,



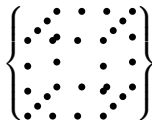
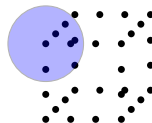
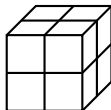
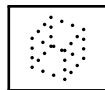
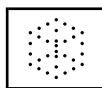
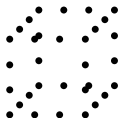
- Different ways of handling 3D point clouds:
 - As a set of images,
 - As a voxel occupancy grid,



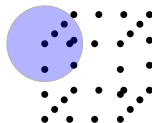
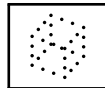
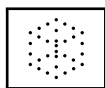
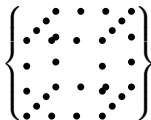
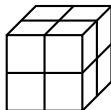
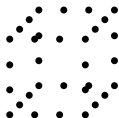
- Different ways of handling 3D point clouds:
 - As a set of images,
 - As a voxel occupancy grid,
 - As a convolution-enabling structure,



- Different ways of handling 3D point clouds:
 - As a set of images,
 - As a voxel occupancy grid,
 - As a convolution-enabling structure,
 - As a non-ordered set of points,

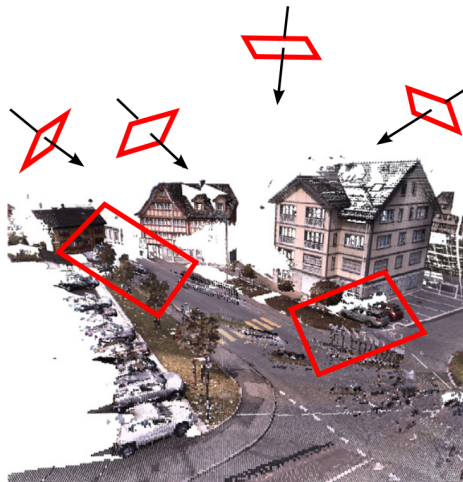


- Different ways of handling 3D point clouds:
 - As a set of images,
 - As a voxel occupancy grid,
 - As a convolution-enabling structure,
 - As a non-ordered set of points,
 - As a graph structure.



- **Idea:** CNNs work great for images.
Can we use images for 3D?

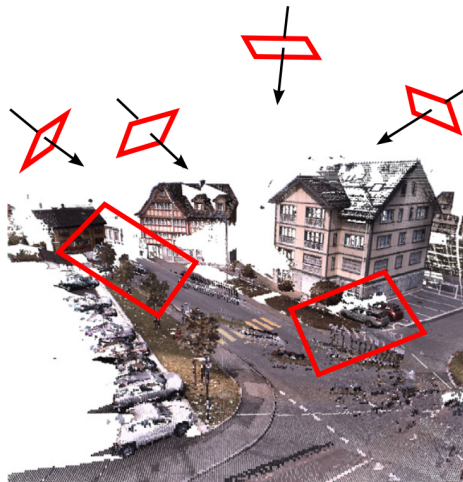
- **Idea:** CNNs work great for images.
Can we use images for 3D?
- **SnapNet:**



Boulch et. al. 2017

credit: Boulch et. al. 2017

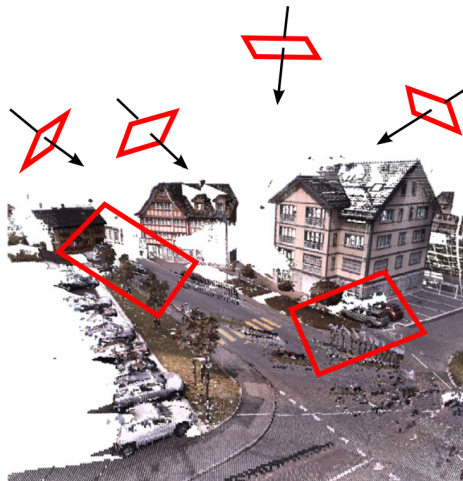
- **Idea:** CNNs work great for images.
Can we use images for 3D?
- **SnapNet:**
 - surface reconstruction,



Boulch et. al. 2017

credit: Boulch et. al. 2017

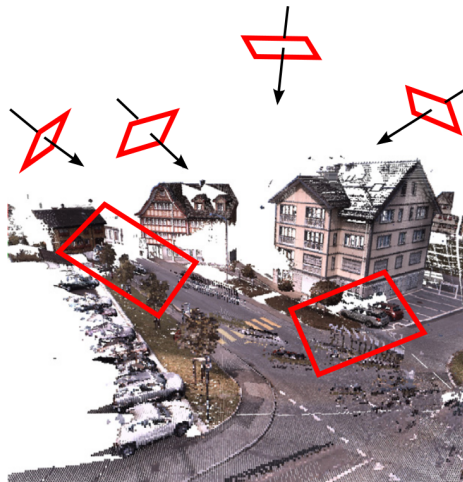
- **Idea:** CNNs work great for images.
Can we use images for 3D?
- **SnapNet:**
 - surface reconstruction,
 - *virtual snapshots*,



Boulch et. al. 2017

credit: Boulch et. al. 2017

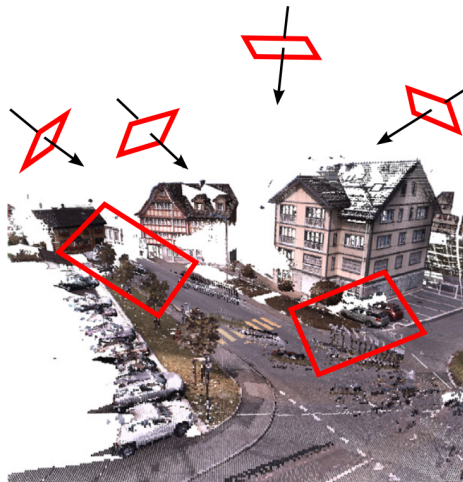
- **Idea:** CNNs work great for images.
Can we use images for 3D?
- **SnapNet:**
 - surface reconstruction,
 - *virtual snapshots*,
 - semantic segmentation of snapshots with CNNs,



Boulch et. al. 2017

credit: Boulch et. al. 2017

- **Idea:** CNNs work great for images.
Can we use images for 3D?
- **SnapNet:**
 - surface reconstruction,
 - *virtual snapshots*,
 - semantic segmentation of snapshots with CNNs,
 - project prediction back to point clouds.

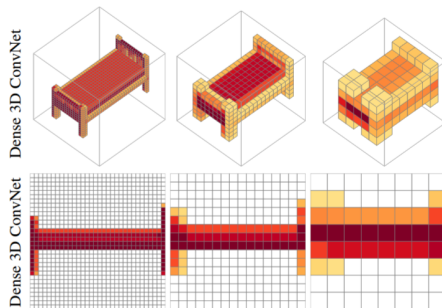


Boulch et. al. 2017

credit: Boulch et. al. 2017

- **Idea:** generalize image convolutions to 3D regular grids.

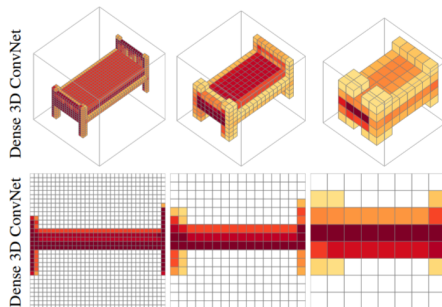
- **Idea:** generalize image convolutions to 3D regular grids.
- Voxelization + 3D convNets.



Wu2015

credit: Riegler2017, Tchapmi2017, Jampani2017

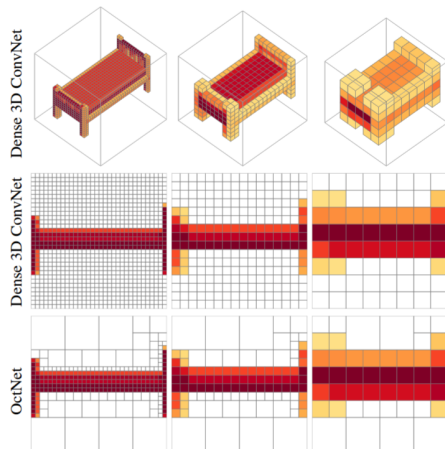
- **Idea:** generalize image convolutions to 3D regular grids.
- Voxelization + 3D convNets.
- **Problem:** inefficient representation, loss of invariance, costly (cubic).



Wu2015

credit: Riegler2017, Tchapmi2017, Jampani2017

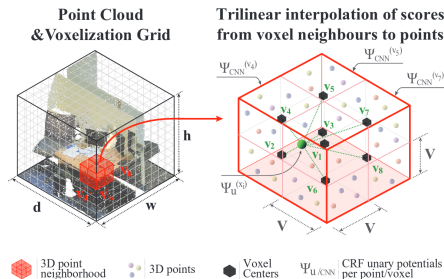
- **Idea:** generalize image convolutions to 3D regular grids.
- Voxelization + 3D convNets.
- **Problem:** inefficient representation, loss of invariance, costly (cubic).
- **Idea 1:** OctNet, OctTree based approach



Wu2015 , Riegler2017

credit: Riegler2017, Tchapmi2017, Jampani2017

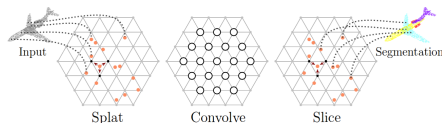
- **Idea:** generalize image convolutions to 3D regular grids.
- Voxelization + 3D convNets.
- **Problem:** inefficient representation, loss of invariance, costly (cubic).
- **Idea 1:** OctNet, OctTree based approach
- **Idea 2:** SegCloud, large voxels, subvoxel predictions with CRFs.



Wu2015 , Riegler2017 , Tchapmi2017, Jampani2018.

credit: Riegler2017, Tchapmi2017, Jampani2017

- **Idea:** generalize image convolutions to 3D regular 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, indexed with hashmaps.

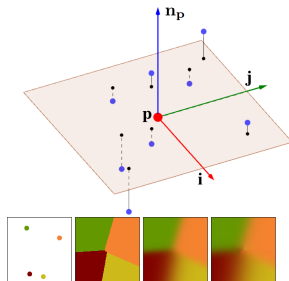


Wu2015 , Riegler2017 , Tchapmi2017,
Jampani2018.

credit: Riegler2017, Tchapmi2017, Jampani2017

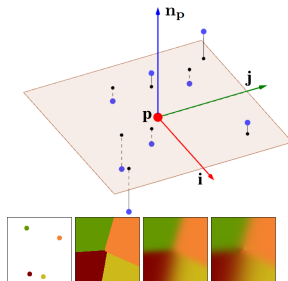
- **Idea:** generalize 2D convolutions to *spatial* convolutions.

- **Idea:** generalize 2D convolutions to *spatial* convolutions.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.



credit: Tatarchenko2018, Li2018

- **Idea:** generalize 2D convolutions to *spatial* convolutions.
- **Tangent Convolution:** 2D convolution in the tangent space of each point.
- Can now use 2D convolutions to learn spatial features.

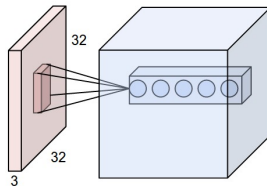


- Generalizing Spatial convolutions.

(i, j) point to embed, f features, K kernel,
 $c_d \in \mathbb{R}^d$ kernel weights.

2D convolutions on images:

$$(\mathcal{C} * f)(i, j) = \sum_{d \in K} c_d f[(i, j) + d] .$$



credit : pubs.sciepub.com/ajmm

- Generalizing Spatial convolutions.

(i, j) point to embed, f features, K kernel, $c_d \in \mathbb{R}^d$ kernel weights.

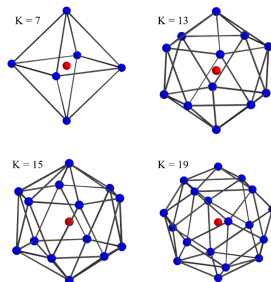
2D convolutions on images:

$$(\mathcal{C} * f)(i, j) = \sum_{d \in K} c_d f[(i, j) + d] .$$

- x spatial coordinates, N_x : neighbors of x , $y_j \in K$ spatial coordinate of kernel points.

3D convolutions on point clouds:

$$(\mathcal{C} * f)(x) = \sum_{j \in K} \sum_{i \in N_x} g(x_i, y_j) c_j f_i .$$



credit: Thomas et al. 2019, Boulch 2019

- Generalizing Spatial convolutions.

(i, j) point to embed, f features, K kernel,
 $c_d \in \mathbb{R}^d$ kernel weights.

2D convolutions on images:

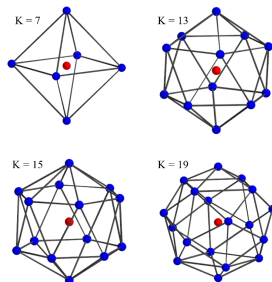
$$(\mathcal{C} * f)(i, j) = \sum_{d \in K} c_d f[(i, j) + d] .$$

- x spatial coordinates, N_x : neighbors of x ,
 $y_j \in K$ spatial coordinate of kernel points.

3D convolutions on point clouds:

$$(\mathcal{C} * f)(x) = \sum_{j \in K} \sum_{i \in N_x} g(x_i, y_j) c_j f_i .$$

- Kernels can be fixed or learnt.



credit: Thomas et al. 2019, Boulch 2019

- Generalizing Spatial convolutions.

(i, j) point to embed, f features, K kernel,
 $c_d \in \mathbb{R}^d$ kernel weights.

2D convolutions on images:

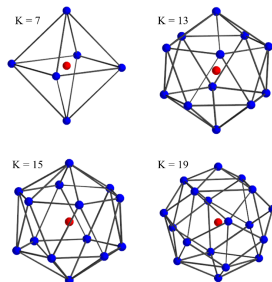
$$(\mathcal{C} * f)(i, j) = \sum_{d \in K} c_d f[(i, j) + d] .$$

- x spatial coordinates, N_x : neighbors of x ,
 $y_j \in K$ spatial coordinate of kernel points.

3D convolutions on point clouds:

$$(\mathcal{C} * f)(x) = \sum_{j \in K} \sum_{i \in N_x} g(x_i, y_j) c_j f_i .$$

- Kernels can be fixed or learnt.
- $g(x, y) = \max(0, 1 - \frac{1}{\sigma} \|x - y\|)$.



credit: Thomas et al. 2019, Boulch 2019

3D Convolution-Based Methods II

- Generalizing Spatial convolutions.

(i, j) point to embed, f features, K kernel,
 $c_d \in \mathbb{R}^d$ kernel weights.

2D convolutions on images:

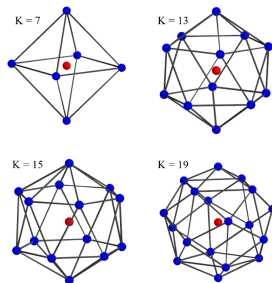
$$(\mathcal{C} * f)(i, j) = \sum_{d \in K} c_d f[(i, j) + d] .$$

- x spatial coordinates, N_x : neighbors of x ,
 $y_j \in K$ spatial coordinate of kernel points.

3D convolutions on point clouds:

$$(\mathcal{C} * f)(x) = \sum_{j \in K} \sum_{i \in N_x} g(x_i, y_j) c_j f_i .$$

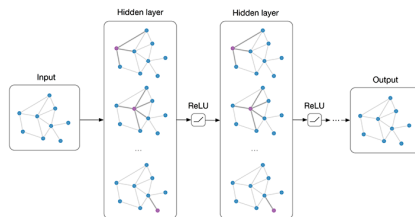
- Kernels can be fixed or learnt.
- $g(x, y) = \max(0, 1 - \frac{1}{\sigma} \|x - y\|)$.
- Or g can be a learnt function as well.



credit: Thomas etal. 2019, Boulch 2019

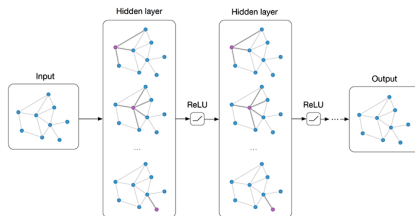
Graph-Neural Network

- **Idea:** Generalize convolutions to the general graph setting.



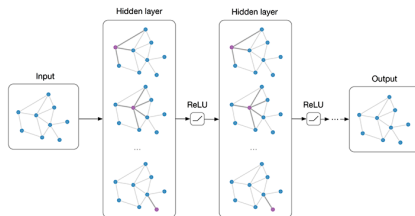
Graph-Neural Network

- **Idea:** Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph.



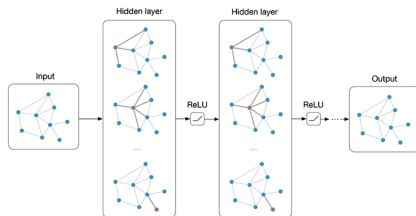
Graph-Neural Network

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



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

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

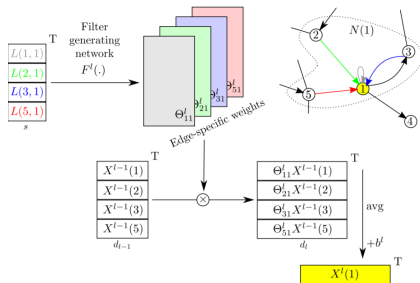


- **Idea:** Generalize convolutions to the general graph setting.
- For example: k-nearest neighbors graph.
- **Principle:** Each point maintain a hidden state h_i influenced by its neighbors.
- **GNN Qi2017:** iterative message-passing algorithm with mapping f and RNN g :

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

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

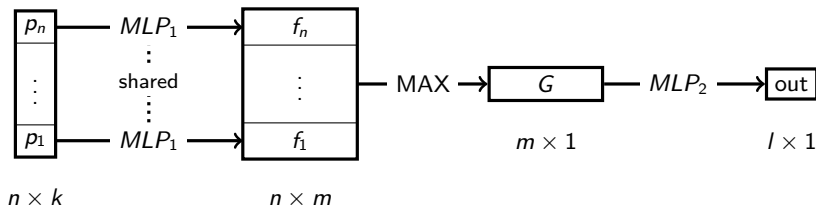
$$h_i^{(t+1)} = g\left(\sum_{j \rightarrow i} \Theta_{i,j} \odot h_j^t, h_i^t\right).$$



- A cornerstone of modern 3D analysis.

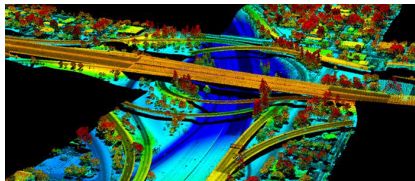
- A cornerstone of modern 3D analysis.
- **A fundamental constraint:** permutation-invariant inputs.

- A cornerstone of modern 3D analysis.
- **A fundamental constraint:** permutation-invariant inputs.
- **Solution:** n : number of points, k size of observations, m size of intermediary embeddings, l size of desired output:
 - Independantly compute point features $\{f_i\}_{i=1}^n$ through function $MLP_1 : \mathbb{R}^k \mapsto \mathbb{R}^m$
 - Computes permutation invariant global embedding $[G]_j = \max_{i=1 \dots n} [f_i]_j, j = 1 \dots m$
 - Process this global with function with $MLP_3 : \mathbb{R}^m \mapsto \mathbb{R}^l$



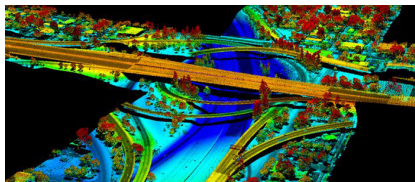
- 1 Motivation
- 2 Deep-Learning Approaches
- 3 Scaling Segmentation**
- 4 In Practice
- 5 Bibliography

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



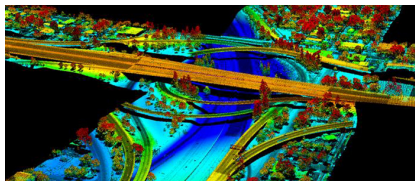
Why we need to scale

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



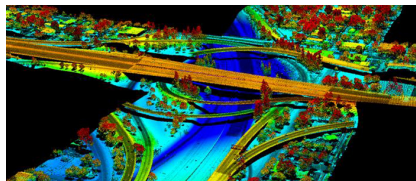
Why we need to scale

- **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

- Pyramid structure for multi-scale feature extraction.

- Initialization:

$$f_i^{(0)} = \text{initial features: xyz, RGB, etc...}$$

- Spatial Pooling:

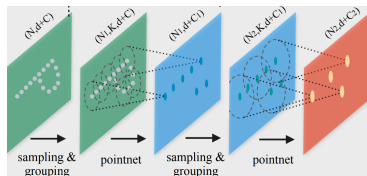
$$P^{(t+1)} = \text{subsample}(P^{(t)})$$

- Correspondence

$$N_i = \text{neighborhood of } i \in P^{(t+1)} \text{ in } P^{(t)}$$

- Next layer:

$$f_i^{(t+1)} = \text{PointNet}^{(t)} \left(\{f_j^{(t)}\}_{j \in N_i} \right)$$



credit: Qi et. al. 2017b

- Pyramid structure for multi-scale feature extraction.

- Initialization:

$$f_i^{(0)} = \text{initial features: xyz, RGB, etc...}$$

- Spatial Pooling:

$$P^{(t+1)} = \text{subsample}(P^{(t)})$$

- Correspondence

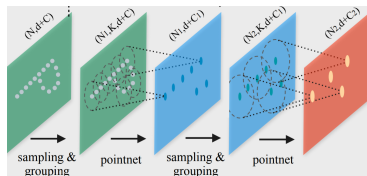
$$N_i = \text{neighborhood of } i \in P^{(t+1)} \text{ in } P^{(t)}$$

- Next layer:

$$f_i^{(t+1)} = \text{PointNet}^{(t)} \left(\{f_j^{(t)}\}_{j \in N_i} \right)$$

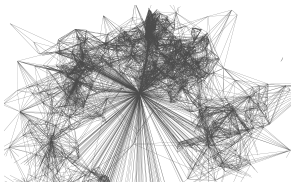
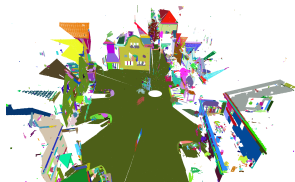
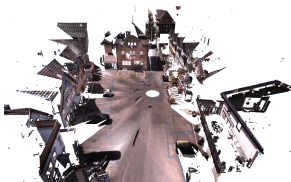
- From local to global with with increasingly abstract features.

credit: Qi et. al. 2017b



- **Observation:**

$$n_{\text{points}} \gg n_{\text{objects}}.$$

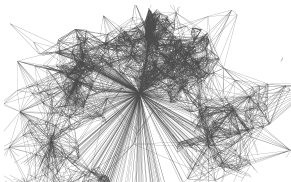
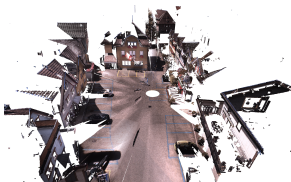


Landrieu&Simonovski, CVPR 2018

- **Observation:**

$n_{\text{points}} \gg n_{\text{objects}}$.

- Partition scene into superpoints with simple shapes.

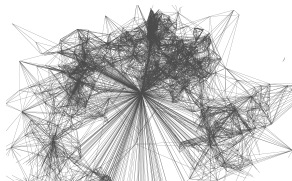
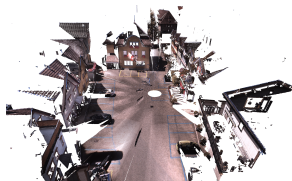


Landrieu&Simonovski, CVPR 2018

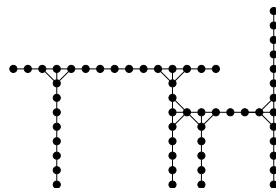
- **Observation:**

$$n_{\text{points}} \gg n_{\text{objects}}.$$

- Partition scene into superpoints with simple shapes.
- Only a few superpoints, context leveraging with powerful graph methods.

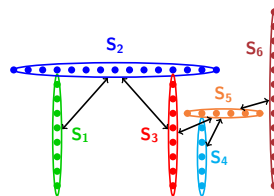


Landrieu&Simonovski, CVPR 2018



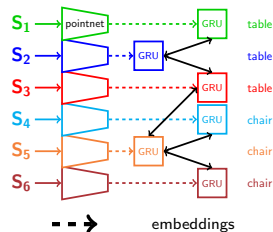
● point
— Voronoi Edge

(a) Point cloud



○ superpoint
↔ superedge

(b) Superpoint graph

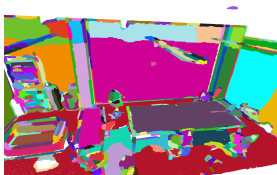


- - → embeddings

(c) Convolution Network

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

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



Superpoint Partition

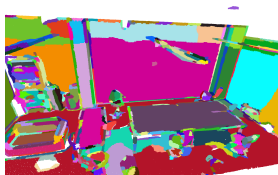
$$f^* = \operatorname{argmin}_{f \in \mathbb{R}^{C \times m}} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} [f_i \neq f_j],$$

- $e \in \mathbb{R}^{C \times m}$: handcrafted descriptors of the local geometry/radiometry.
- f^* : piecewise constant approximation of e .



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

- $e \in \mathbb{R}^{C \times m}$: handcrafted descriptors of the local geometry/radiometry.
- f^* : piecewise constant approximation of e .
- Solutions can be efficiently approximated with ℓ_0 -cut pursuit.



Landrieu & obozinski, SIIMS 2017, Raguet & Landrieu, ICML 2018, 2019

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

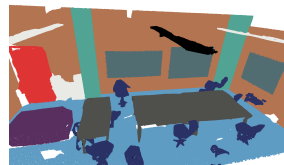
- $e \in \mathbb{R}^{C \times m}$: handcrafted descriptors of the local geometry/radiometry.
- f^* : piecewise constant approximation of e .
- Solutions can be efficiently approximated with ℓ_0 -cut pursuit.
- Superpoints: constant connected components of f^* .



Landrieu & obozinski, SIIMS 2017, Raguet & Landrieu, ICML 2018, 2019

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

- $e \in \mathbb{R}^{C \times m}$: handcrafted descriptors of the local geometry/radiometry.
- f^* : piecewise constant approximation of e .
- Solutions can be efficiently approximated with ℓ_0 -cut pursuit.
- Superpoints: constant connected components of f^* .
- **Problem:** any errors made in the partition will carry in the prediction...

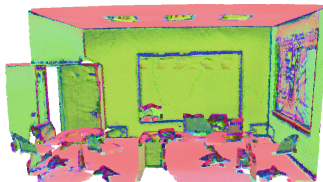


Landrieu & obozinski, SIIMS 2017, Raguet & Landrieu, ICML 2018, 2019

Learning to Partition



Input Point Cloud



Learned Embedding



Oversegmentation

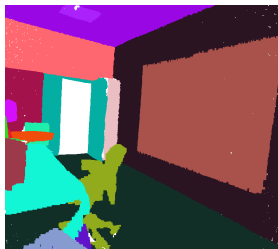


True Objects

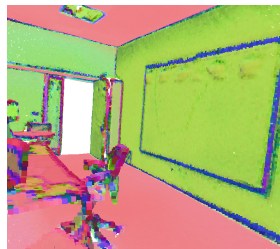
- 1) Train a neural network to produce points embeddings with high contrast at the border of objects...
 - 2) ... which serve as inputs of a segmentation algorithm.
- Requires 5 times less superpoints than unsupervised approaches.



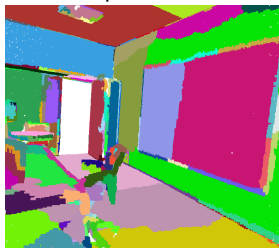
Input cloud



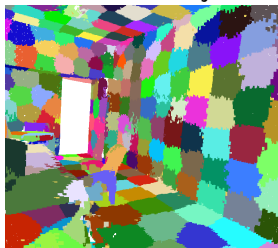
Ground truth objects



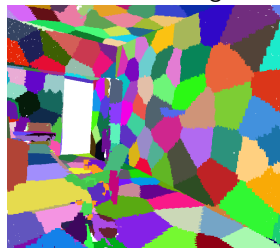
LCE embeddings



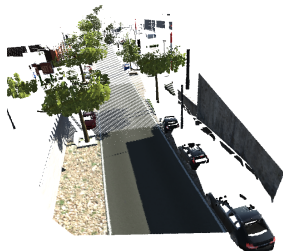
Graph-LCE (ours)



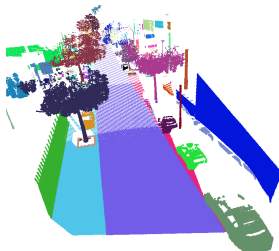
VCCS, Papon *et al.* 2013



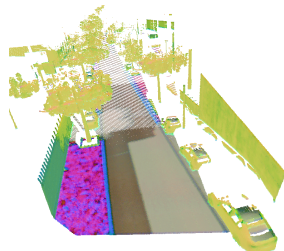
Lin *et al.* 2018



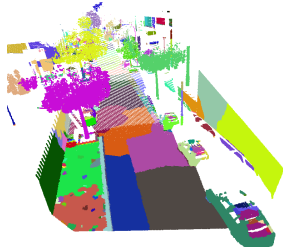
Input cloud



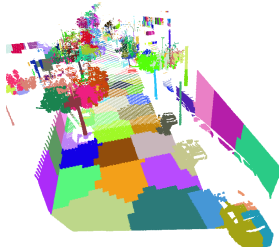
Ground truth objects



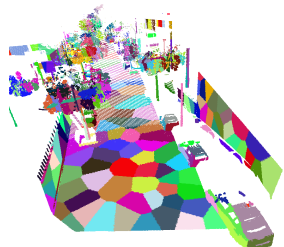
LCE embeddings



Graph-LCE (ours)



VCCS, Papon *et al.* 2013



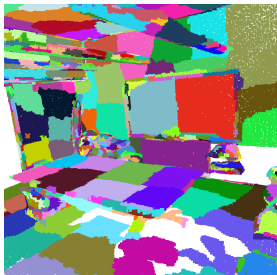
Lin *et al.* 2018

- 1 Motivation
- 2 Deep-Learning Approaches
- 3 Scaling Segmentation
- 4 In Practice**
- 5 Bibliography

Illustration



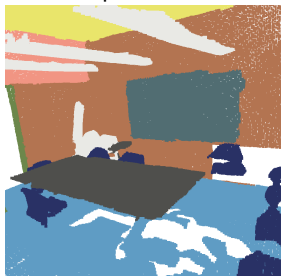
Input Cloud



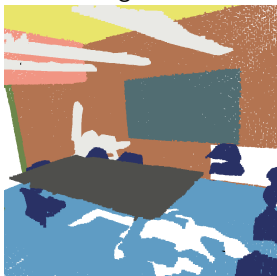
Oversegmentation

S3DIS

- ceiling
- floor
- wall
- column
- beam
- window
- door
- table
- chair
- bookcase
- sofa
- board
- clutter
- unlabelled

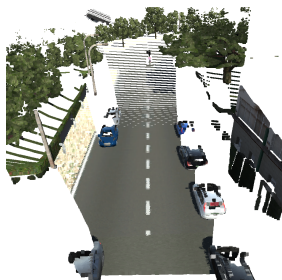


prediction

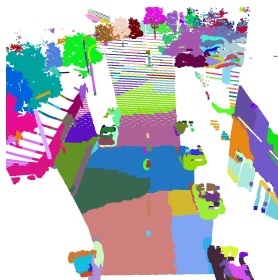


Ground Truth

Illustration



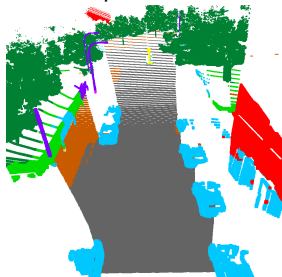
Input Cloud



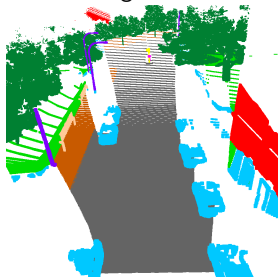
Oversegmentation

vKITTI

- terrain
- tree
- vegetation
- building
- road
- guard rail
- traffic sign
- traffic light
- pole
- misc
- truck
- car
- van
- unlabelled



prediction



Ground Truth

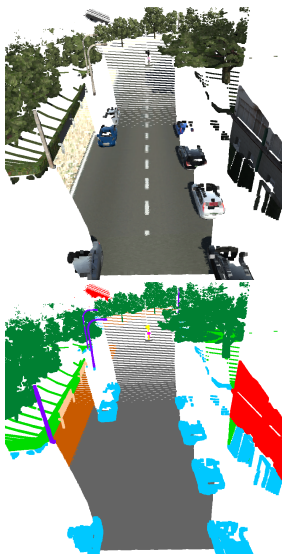
Quantitative Results on S3DIS:



Method	OA	mAcc	mIoU
6-fold cross validation			
PointNet	78.5	66.2	47.6
Engelmann <i>et al.</i> 2017	81.1	66.4	49.7
PointNet++	81.0	67.1	54.5
Engelmann <i>et al.</i> 2018	84.0	67.8	58.3
SPG	85.5	73.0	62.1
PointCNN	88.1	75.6	65.4
ConvPoint	88.1	75.6	68.2
SSP + SPG	87.9	78.3	68.4
PointSIFT	88.7	-	70.2
KPConv	88.8	79.1	70.6
Fold 5			
PointNet	-	49.0	41.1
Engelmann <i>et al.</i>	84.2	61.8	52.2
pointCNN	85.9	63.9	57.3
SPG	86.4	66.5	58.0
PCCN	-	67.0	58.3
SSP + SPG	87.9	68.2	61.7

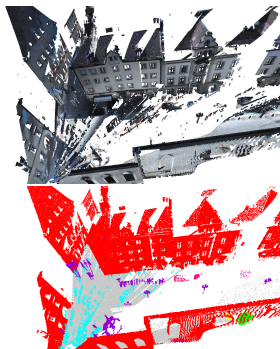
Table: State-of-the-art 2019: **OA** : Overall Accuracy, **mAcc** : average class accuracy, **mIoU**: average class Intersection over Union.

Quantitative Results on vKITTI3D:



Method	OA	mAcc	mIoU
PointNet	79.7	47.0	34.4
Engelmann <i>et al.</i> 2018	79.7	57.6	35.6
Engelmann 2017	80.6	49.7	36.2
3P-RNN	87.8	54.1	41.6
SSP + SPG	84.3	67.3	52.0

Table: STOTA 2019 on vKITTI3D with 6-fold cross validation.



Method	OA	mIoU
reduced test set: 78 699 329 points		
TMLC-MSR	86.2	54.2
DeePr3SS	88.9	58.5
SnapNet	88.6	59.1
SegCloud	88.1	61.3
SPG	94.0	73.2
KP-FCNN	92.9	74.6
full test set: 2 091 952 018 points		
TMLC-MS	85.0	49.4
SnapNet	91.0	67.4
SPG	92.9	76.2
ConvPoint	93.4	76.5

Table: STOTA 2019 on Semantic3D.

- **Which algorithm to chose in practice:**

- **Which algorithm to chose in practice:**
 - Start with a classical algorithm for baseline / data cleaning
 - If enough annotation, PointNet + sliding window.

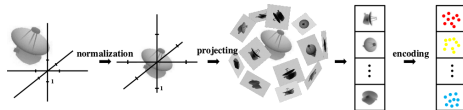
- **Which algorithm to chose in practice:**

- Start with a classical algorithm for baseline / data cleaning
- If enough annotation, PointNet + sliding window.
- If not powerful enough, PointCNN/KPconv/ConvPoint.

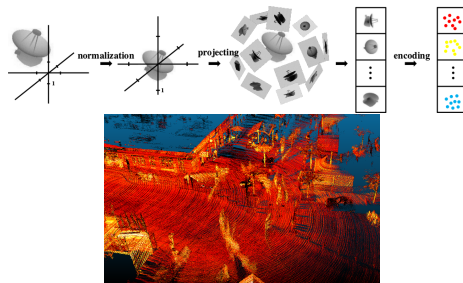
- **Which algorithm to chose in practice:**

- Start with a classical algorithm for baseline / data cleaning
- If enough annotation, PointNet + sliding window.
- If not powerful enough, PointCNN/KPconv/ConvPoint.
- If too slow / loss of global structure: SPG/PointNet++.

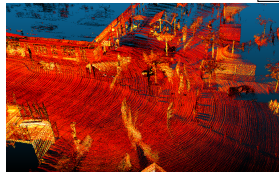
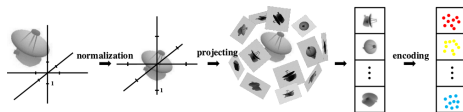
- Semi-supervised learning **Zhu2016**.



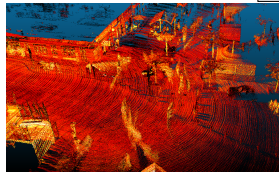
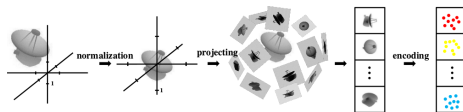
- Semi-supervised learning **Zhu2016**.
- Real-time analysis for dynamic 3D data for autonomous driving (ANR READY3D).



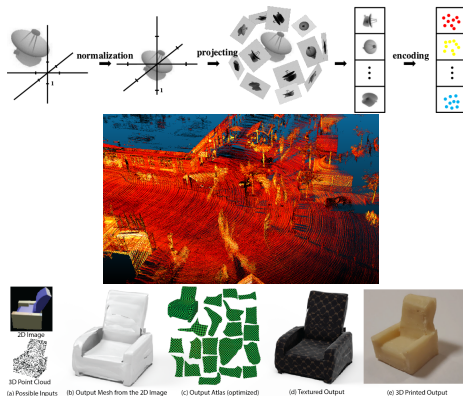
- Semi-supervised learning **Zhu2016**.
- Real-time analysis for dynamic 3D data for autonomous driving (ANR READY3D).
- Deep learning for other remote sensing tasks: segmentation, object detection, surface reconstruction. **Groueix2018**.



- Semi-supervised learning **Zhu2016**.
- Real-time analysis for dynamic 3D data for autonomous driving (ANR READY3D).
- Deep learning for other remote sensing tasks: segmentation, object detection, surface reconstruction. **Groueix2018**.
- Multi-source, multi-task, etc...



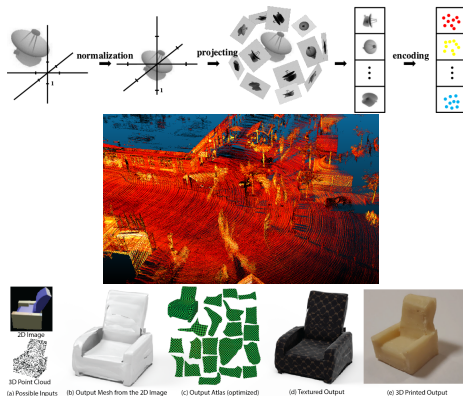
- Semi-supervised learning **Zhu2016**.
- Real-time analysis for dynamic 3D data for autonomous driving (ANR READY3D).
- Deep learning for other remote sensing tasks: segmentation, object detection, surface reconstruction. **Groueix2018**.
- Multi-source, multi-task, etc...
- Most codes are now open-source.



- Semi-supervised learning **Zhu2016**.
- Real-time analysis for dynamic 3D data for autonomous driving (ANR READY3D).
- Deep learning for other remote sensing tasks: segmentation, object detection, surface reconstruction. **Groueix2018**.
- Multi-source, multi-task, etc...
- Most codes are now open-source.

[github/loicland/superpoint-graph](https://github.com/loicland/superpoint-graph)

389 ★ 127 🐾



- 1 Motivation
- 2 Deep-Learning Approaches
- 3 Scaling Segmentation
- 4 In Practice
- 5 Bibliography**

- 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., Raguét, 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.
- 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 χ -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
- Landrieu2019** Landrieu, L. & Boussaha, M. Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning, CVPR2019.
- 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).
- Zhu2016** Zhu, Z., Wang, X., Bai, S., Yao, C., & Bai, X.. Deep learning representation using autoencoder for 3D shape retrieval. Neurocomputing, 2016.
- Groueix2018** Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., & Aubry, M. (2018). A papier-mâché approach to learning 3d surface generation. CVPR, 2018.
- Boulch2019** Boulch, A. (2019). Generalizing discrete convolutions for unstructured point clouds. arXiv:1904.02375.