TUM Data Innovation Lab

Final presentation

3D Matching of TerraSAR-X Derived Ground Control Points to Mobile Mapping Data

Authors	Louise Loesch, Islam Mansour, Magdalena Reich, Xianbin Xie
Mentor(s)	Dr. Wolfgang Koppe, Tatjana Bürgmann (Airbus Defence and Space GmbH)
Co-Mentor	Laure Vuaille
Project lead	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Team Members:

Louise Loesch
Background: Computer vision and & Deep Learning
Study: Informatics

Islam Mansour
Background: Space & Automotive Engineering
Study: Mathematics & Remote Sensing

Magdalena Reich
Background: Optimization & Operations Research
Study: Mathematics

Xianbin Xie
Background: Urban Planning & Cartography
Study: Urbanism

Outline

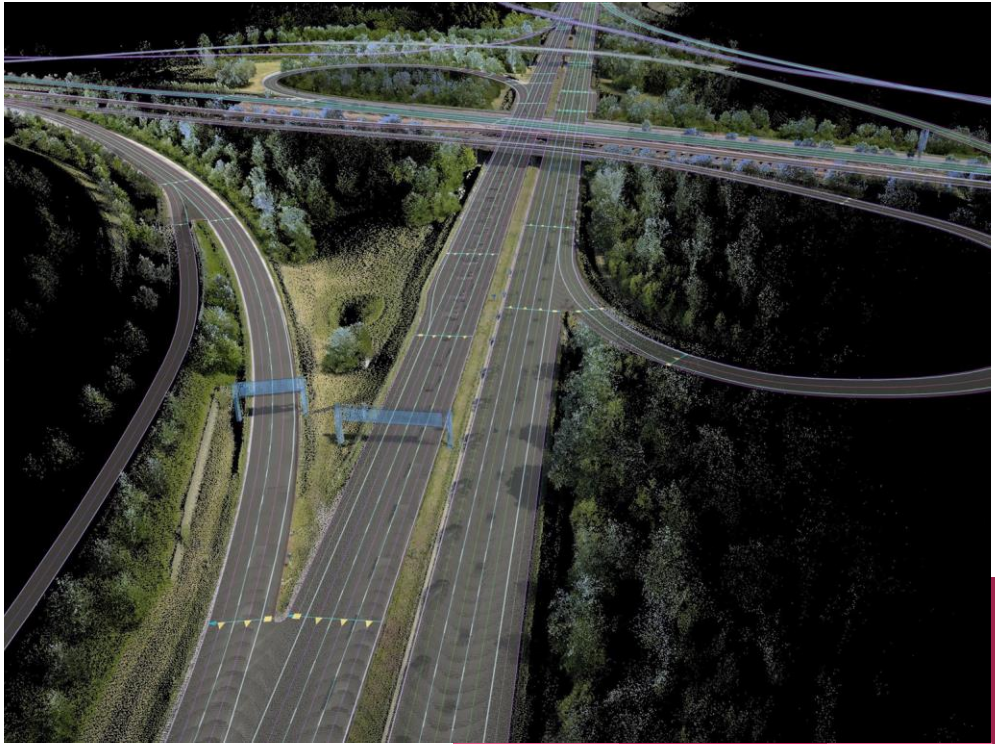
- Background & Problem Definition

- LiDAR Segmentation and Classification
- Extraction of Pole Control Points (PCPs) for matching
- Matching of PCPs and Ground Control Points (GCPs)

- Conclusion
- Demonstration of Prototype

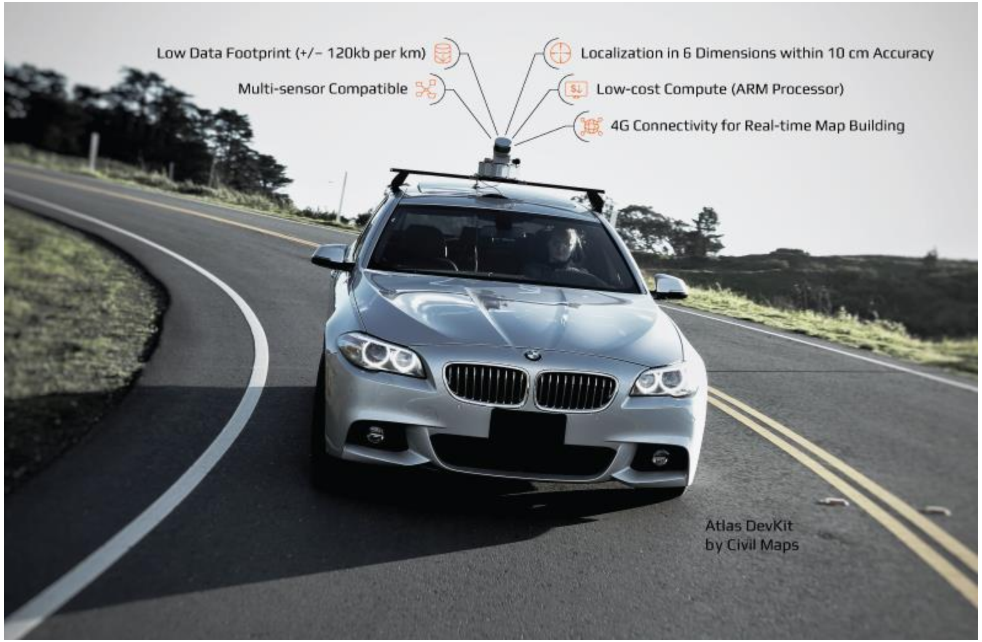
Background (High-Definition 3D Maps)

- Provide HD Maps for highly automated driving



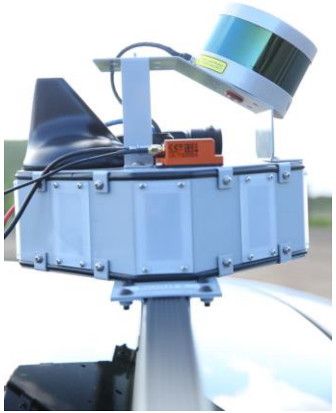
Source: [23]

Background (LiDAR Data)



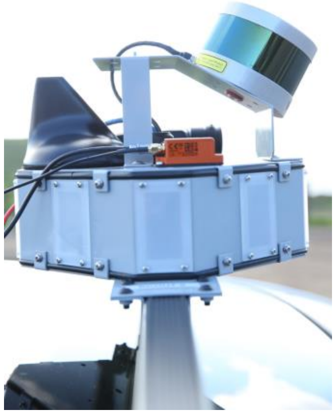
Source: [23]

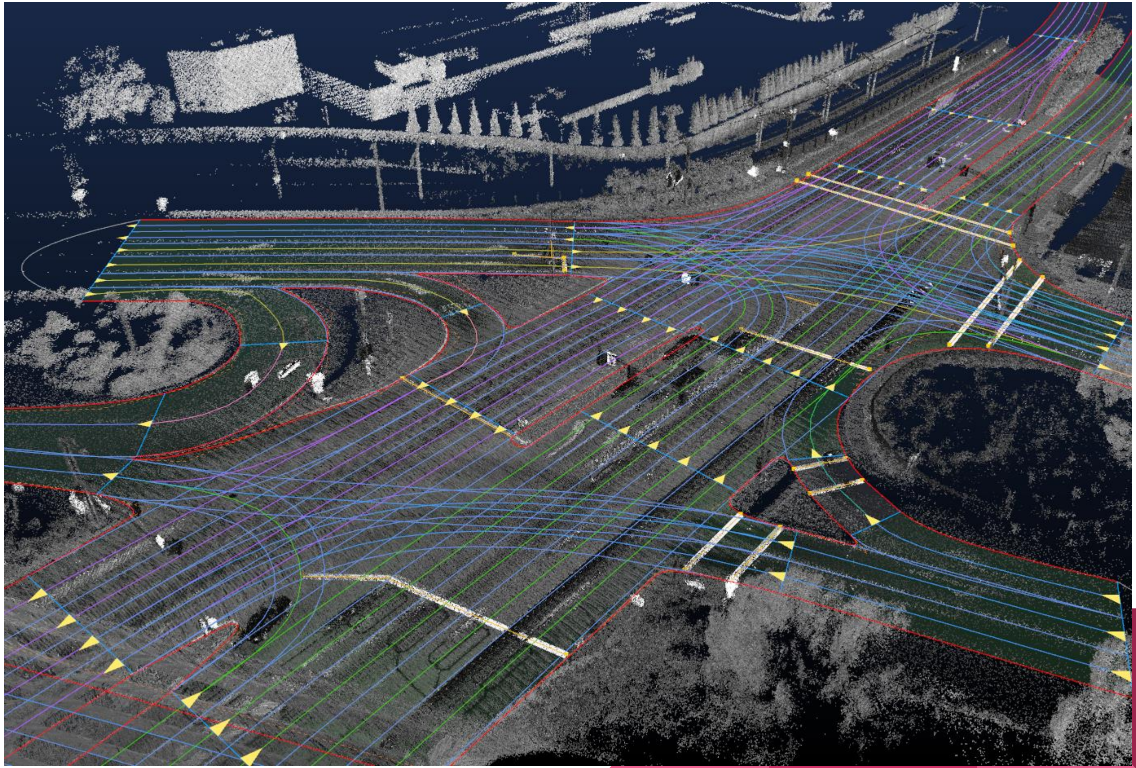
Background (LiDAR Data)



Source: [23]

Background (LiDAR Data)

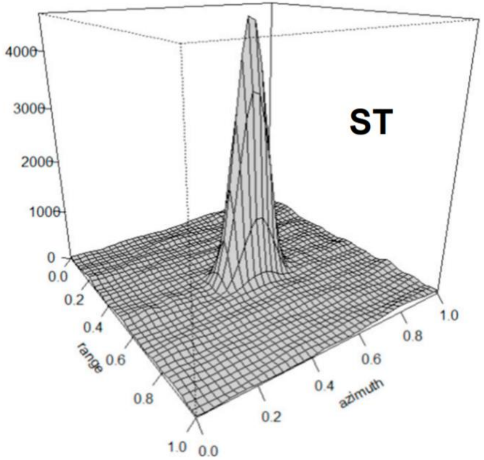




Source: [23]

Background (TerraSAR-X GCPs)

- Traffic and Light Pole shows in SAR images as bright isolated points

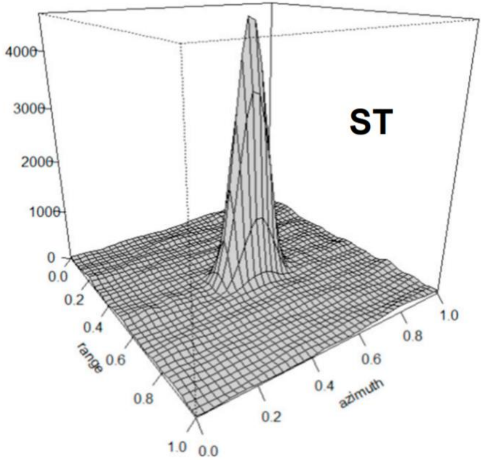


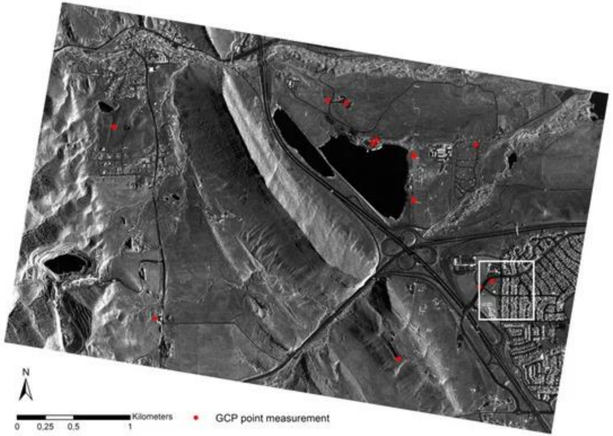
Signal response of a point target (centre peak) in the TerraSAR-X ST image

Source: [22]

Background (TerraSAR-X GCPs)

- Traffic and Light Pole shows in SAR images as bright isolated points

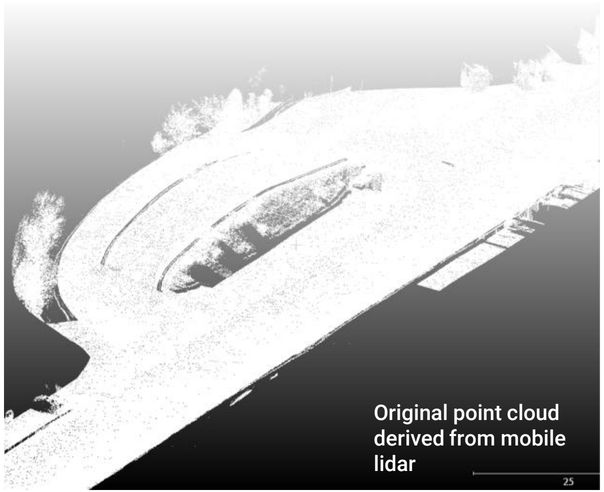




Signal response of a point target (centre peak) in the TerraSAR-X ST image

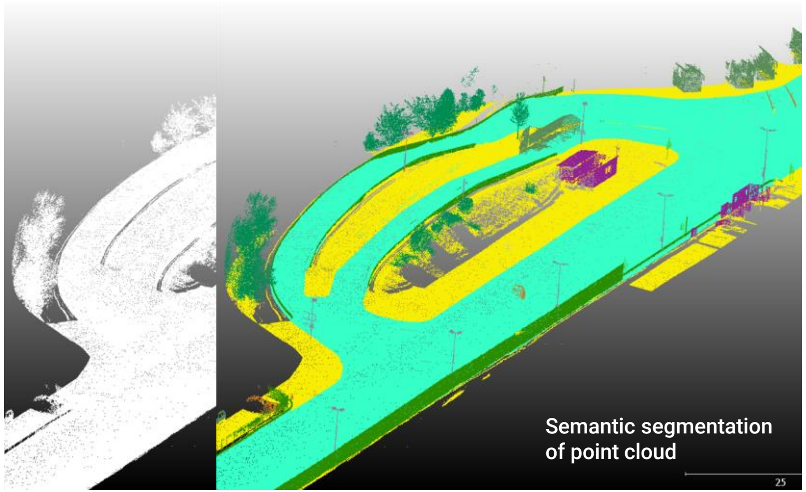
Source: [22]

Problem Definition



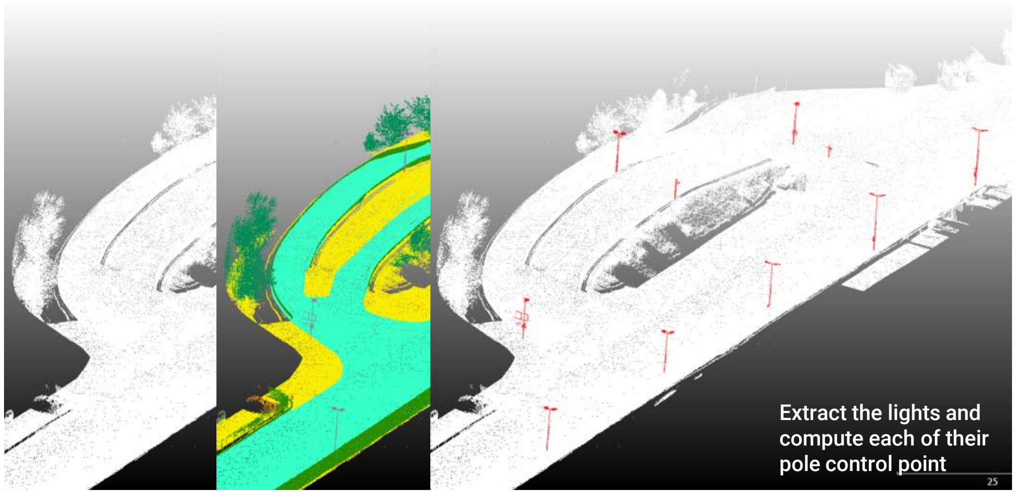
Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

Problem Definition



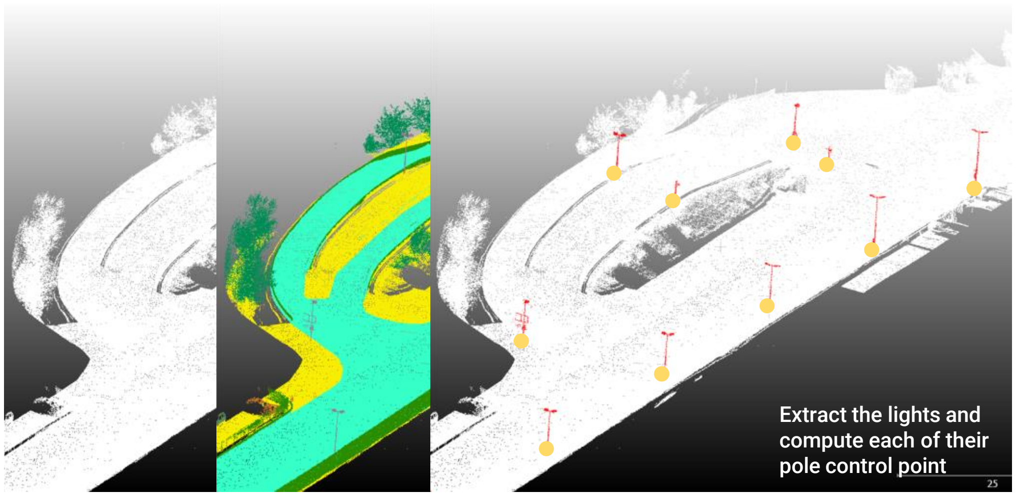
Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

Problem Definition



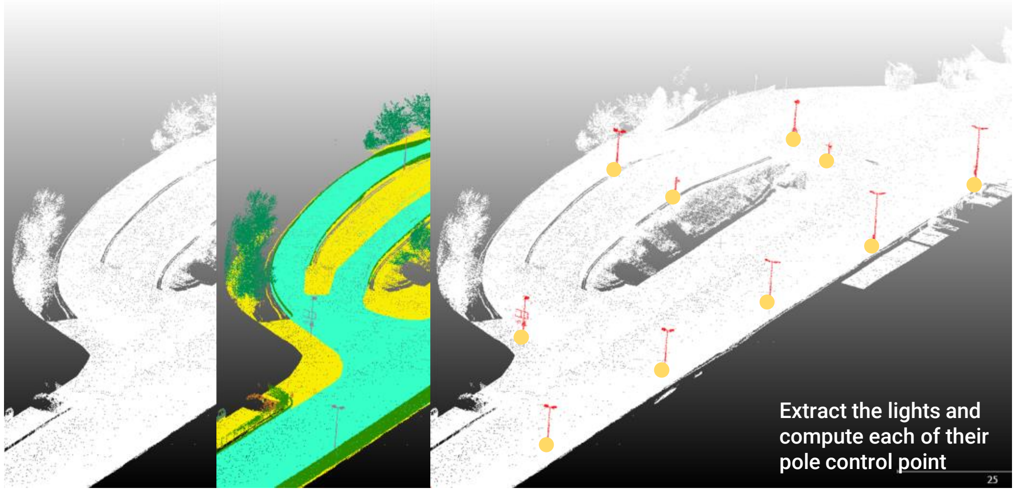
Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

Problem Definition



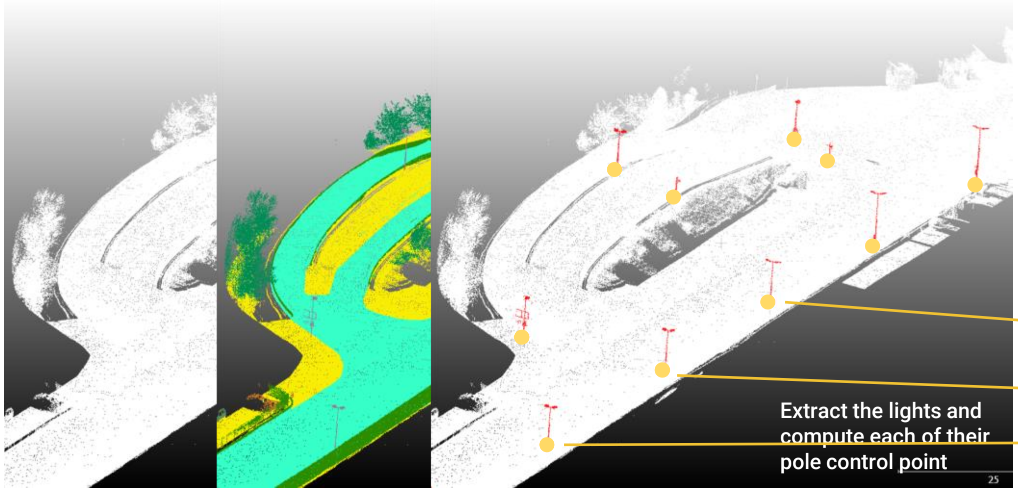
Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

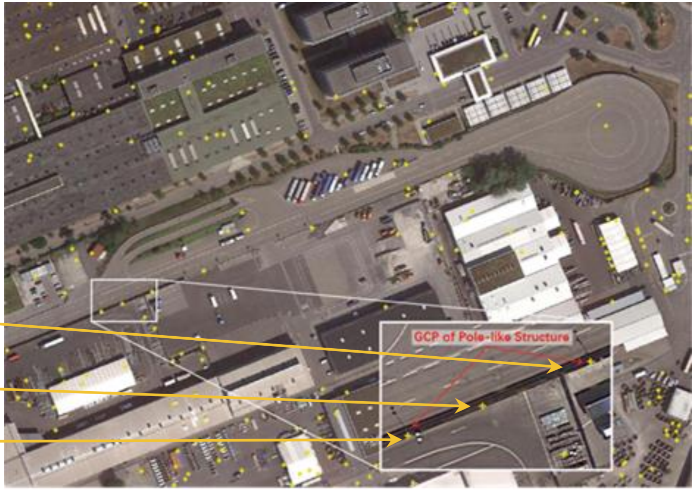
Problem Definition



Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

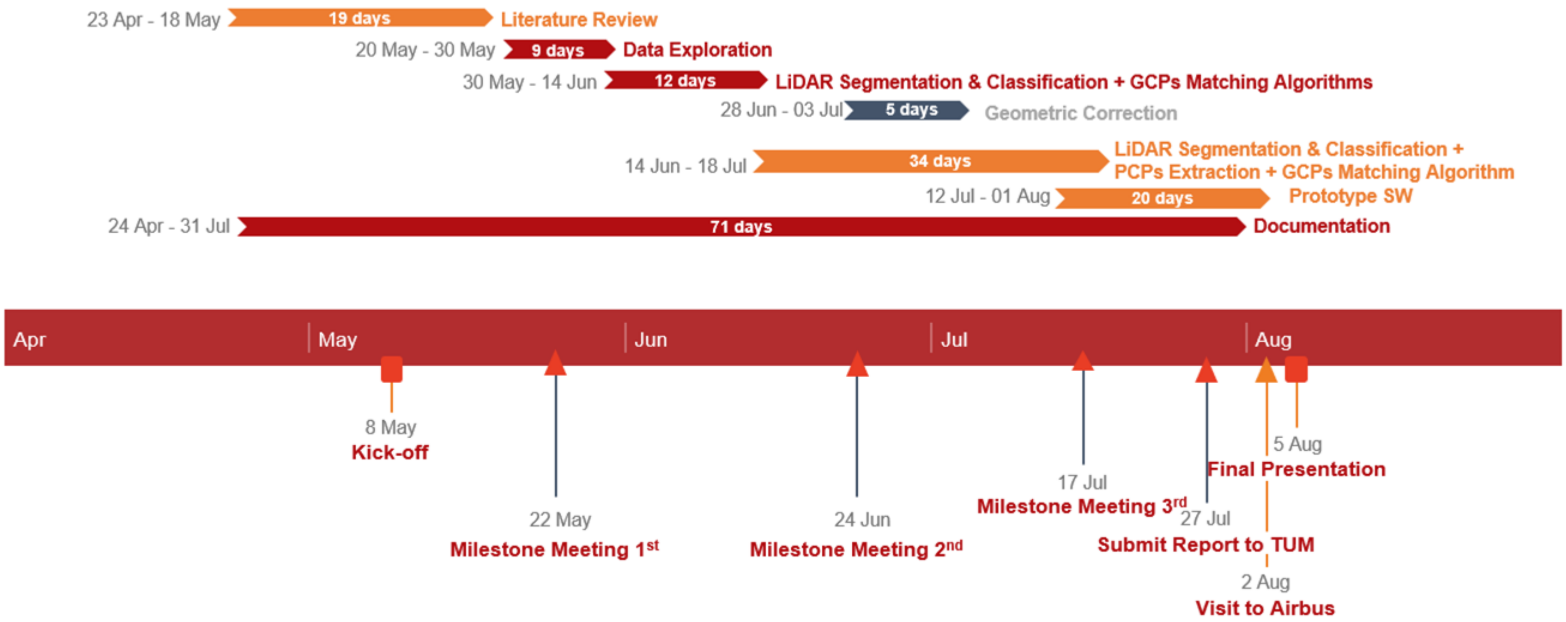
Problem Definition



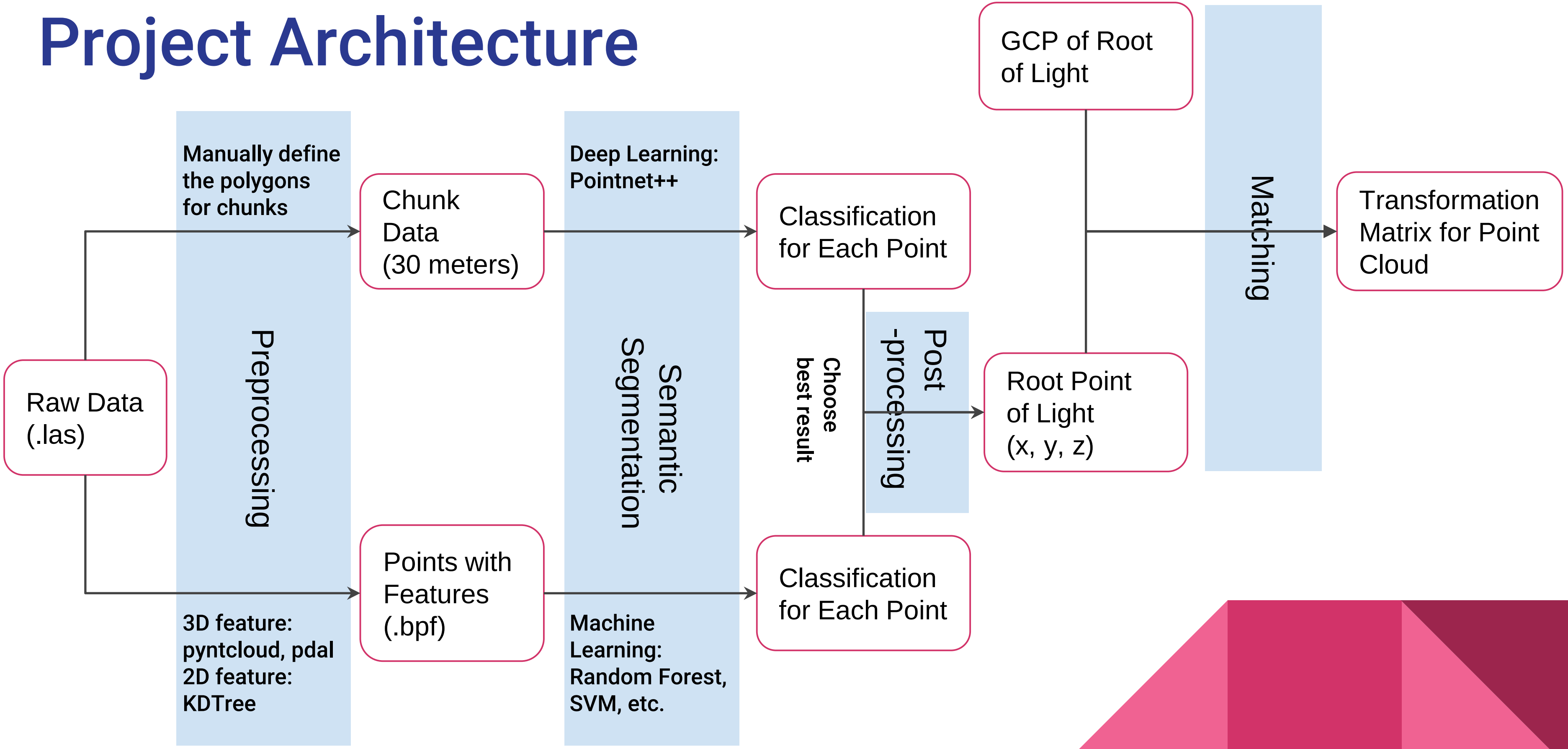


Objective:
Matching extracted PCPs (Pole Control Point) with corresponding GCPs (Ground Control Point)

Project Plan



Project Architecture



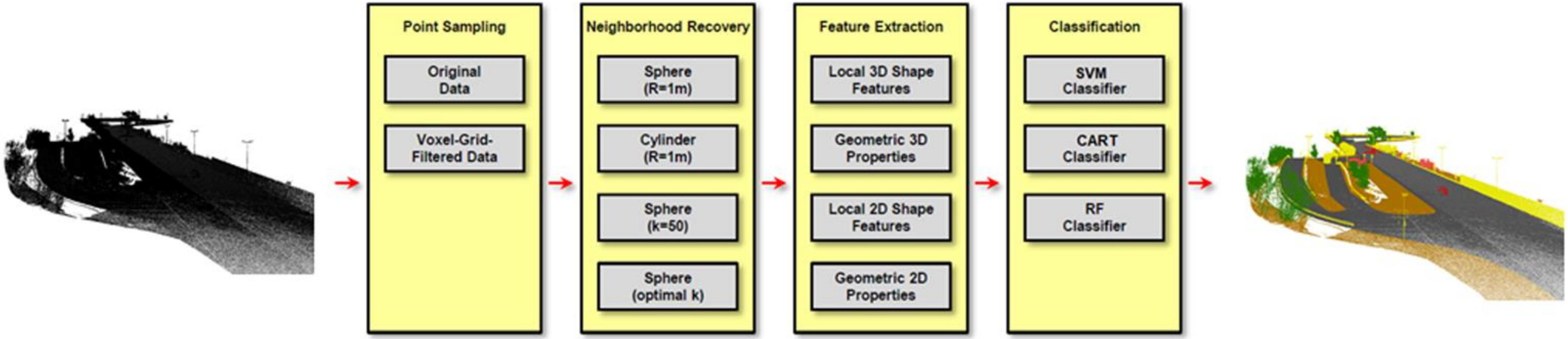
Preprocessing Pipeline

Point Clouds Complexity

Dataset Name	*Number of points*	*Environment*
086b_classified	9,175,355	Urbanized residential area
Werk2_classified_part1	2,044,148	Urbanized industrial area
Werk2_classified_part2	22,043,528	Urbanized industrial area

Classic Machine Learning Based on Features Extraction

- Overview of the methodology



Classic Machine learning architecture for point cloud segmentation

Classic Machine Learning Based on Features Extraction

- 3D Feature Selection

  1. Features of neighbor generated with KD-Tree

	Feature Name	*Description of Feature*
1	*KDistance*	The Euclidean distance to a point's 8-th nearest neighbour
2	*LocalReachabilityDistance*	The inverse of the mean of all reachability distances for a neighbourhood of points
3	*LocalOutlierFactor*	The mean of all LocalReachabilityDistance values for the neighbourhood
4	*NNDistance*	Similar to KDistance
5	*Eigenvalue2*	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	*Eigenvalue1*	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	*Eigenvalue0*	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	*Rank*	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	*NormalX*	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	*NormalY*	
11	*NormalZ*	
12	*Curvature*	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	*RadialDensity*	The density of points in a sphere of a given radius. Here the radius is 2.
14	*Coplanar*	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	*Linearity*	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	*Planarity*	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	*Scattering*	Eigenvalue0 / Eigenvalue2
18	*Omnivariance*	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	*Anisotropy*	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	*Eigentropy*	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	*Eigen_Sum*	The sum of all three eigenvalues
22	*Curvature_Change*	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	*density_2d*	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	*e1_2d*	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	*e2_2d*	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	*Z*	Coordinate in Z-axis.

Classic Machine Learning Based on Features Extraction

- 3D Feature Selection

1. Features of neighbor generated with KD-Tree
2. Eigen-features based on the neighbors

	Feature Name	Description of Feature
1	KDistance	The Euclidean distance to a point's 8-th nearest neighbour
2	LocalReachabilityDistance	The inverse of the mean of all reachability distances for a neighbourhood of points
3	LocalOutlierFactor	The mean of all LocalReachabilityDistance values for the neighbourhood
4	NNDistance	Similar to KDistance
5	Eigenvalue2	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	Eigenvalue1	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	Eigenvalue0	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	Rank	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	NormalX	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	NormalY	
11	NormalZ	
12	Curvature	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	RadialDensity	The density of points in a sphere of a given radius. Here the radius is 2.
14	Coplanar	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	Linearity	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	Planarity	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	Scattering	Eigenvalue0 / Eigenvalue2
18	Omnivariance	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	Anisotropy	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	Eigentropy	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	Eigen_Sum	The sum of all three eigenvalues
22	Curvature_Change	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	density_2d	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	e1_2d	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	e2_2d	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	Z	Coordinate in Z-axis.

Classic Machine Learning Based on Features Extraction

- 3D Feature Selection

1. Features of neighbor generated with KD-Tree
2. Eigen-features based on the neighbors
3. Geometrical features based on Eigen-feature

	Feature Name	*Description of Feature*
1	KDistance	The Euclidean distance to a point's 8-th nearest neighbour
2	LocalReachabilityDistance	The inverse of the mean of all reachability distances for a neighbourhood of points
3	LocalOutlierFactor	The mean of all LocalReachabilityDistance values for the neighbourhood
4	NNDistance	Similar to KDistance
5	Eigenvalue2	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	Eigenvalue1	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	Eigenvalue0	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	Rank	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	NormalX	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	NormalY	
11	NormalZ	
12	Curvature	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	RadialDensity	The density of points in a sphere of a given radius. Here the radius is 2.
14	Coplanar	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	Linearity	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	Planarity	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	Scattering	Eigenvalue0 / Eigenvalue2
18	Omnivariance	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	Anisotropy	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	Eigentropy	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	Eigen_Sum	The sum of all three eigenvalues
22	Curvature_Change	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	density_2d	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	e1_2d	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	e2_2d	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	Z	Coordinate in Z-axis.

Classic Machine Learning Based on Features Extraction

- 3D Feature Selection

1. Features of neighbor generated with KD-Tree
2. Eigen-features based on the neighbors
3. Geometrical features based on Eigen-feature

PDAL is a library used to process point cloud data. it is based on C/C++, so the process is much faster than python.

	Feature Name	Description of Feature
1	KDistance	The Euclidean distance to a point's 8-th nearest neighbour
2	LocalReachabilityDistance	The inverse of the mean of all reachability distances for a neighbourhood of points
3	LocalOutlierFactor	The mean of all LocalReachabilityDistance values for the neighbourhood
4	NNDistance	Similar to KDistance
5	Eigenvalue2	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	Eigenvalue1	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	Eigenvalue0	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	Rank	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	NormalX	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	NormalY	
11	NormalZ	
12	Curvature	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	RadialDensity	The density of points in a sphere of a given radius. Here the radius is 2.
14	Coplanar	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	Linearity	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	Planarity	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	Scattering	Eigenvalue0 / Eigenvalue2
18	Omnivariance	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	Anisotropy	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	Eigentropy	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	Eigen_Sum	The sum of all three eigenvalues
22	Curvature_Change	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	density_2d	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	e1_2d	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	e2_2d	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	Z	Coordinate in Z-axis.

Classic Machine Learning Based on Features Extraction

- 3D Feature Selection

1. Features of neighbor generated with KD-Tree
2. Eigen-features based on the neighbors
3. Geometrical features based on Eigen-feature

PDAL is a library used to process point cloud data. it is based on C/C++, so the process is much faster than python.

Computed as DataFrame with Pandas

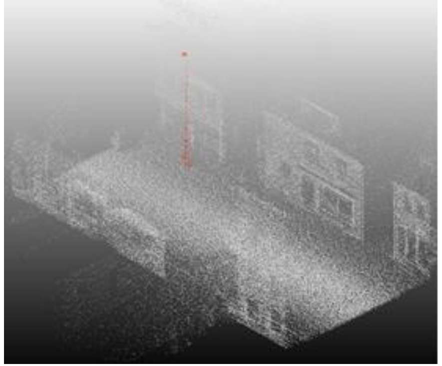
	Feature Name	*Description of Feature*
1	*KDistance*	The Euclidean distance to a point's 8-th nearest neighbour
2	*LocalReachabilityDistance*	The inverse of the mean of all reachability distances for a neighbourhood of points
3	*LocalOutlierFactor*	The mean of all LocalReachabilityDistance values for the neighbourhood
4	*NNDistance*	Similar to KDistance
5	*Eigenvalue2*	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	*Eigenvalue1*	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	*Eigenvalue0*	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	*Rank*	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	*NormalX*	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	*NormalY*	
11	*NormalZ*	
12	*Curvature*	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	*RadialDensity*	The density of points in a sphere of a given radius. Here the radius is 2.
14	*Coplanar*	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	*Linearity*	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	*Planarity*	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	*Scattering*	Eigenvalue0 / Eigenvalue2
18	*Omnivariance*	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	*Anisotropy*	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	*Eigentropy*	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2))
21	*Eigen_Sum*	The sum of all three eigenvalues
22	*Curvature_Change*	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	*density_2d*	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	*e1_2d*	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	*e2_2d*	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	*Z*	Coordinate in Z-axis.

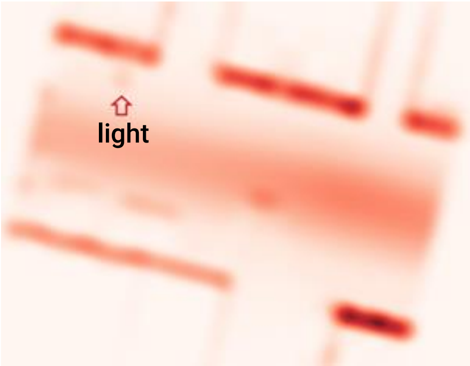
Classic Machine Learning Based on Features Extraction

- 2D Feature Selection

Observation:
Artificial objects are often vertically distributed.
=> Their density is higher than others on 2D plane

Assumption:
Adding 2D features could improve the prediction.



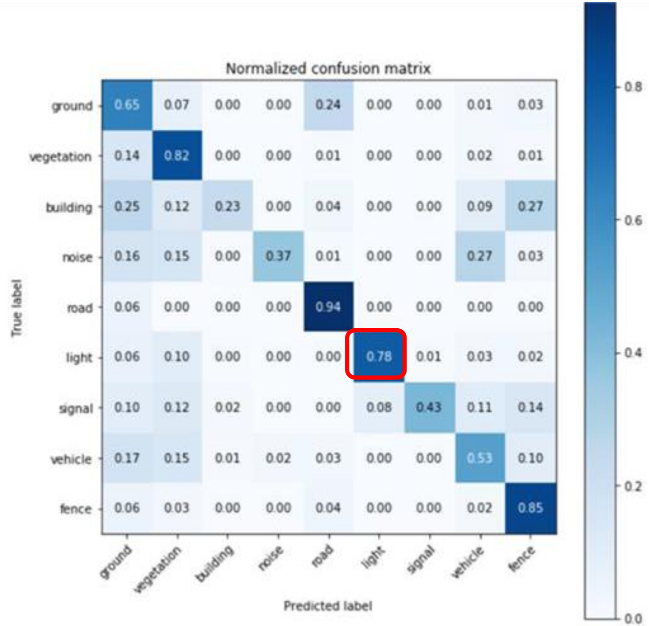


	Feature Name	Description of Feature
1	*KDistance*	The Euclidean distance to a point's 8-th nearest neighbour
2	*LocalReachabilityDistance*	The inverse of the mean of all reachability distances for a neighbourhood of points
3	*LocalOutlierFactor*	The mean of all LocalReachabilityDistance values for the neighbourhood
4	*NNDistance*	Similar to KDistance
5	*Eigenvalue2*	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	*Eigenvalue1*	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	*Eigenvalue0*	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	*Rank*	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	*NormalX*	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	*NormalY*	
11	*NormalZ*	
12	*Curvature*	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	*RadialDensity*	The density of points in a sphere of a given radius. Here the radius is 2.
14	*Coplanar*	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]
15	*Linearity*	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	*Planarity*	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	*Scattering*	Eigenvalue0 / Eigenvalue2
18	*Omnivariance*	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	*Anisotropy*	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	*Eigentropy*	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	*Eigen_Sum*	The sum of all three eigenvalues
22	*Curvature_Change*	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	*density_2d*	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	*e1_2d*	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	*e2_2d*	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	*Z*	Coordinate in Z-axis.

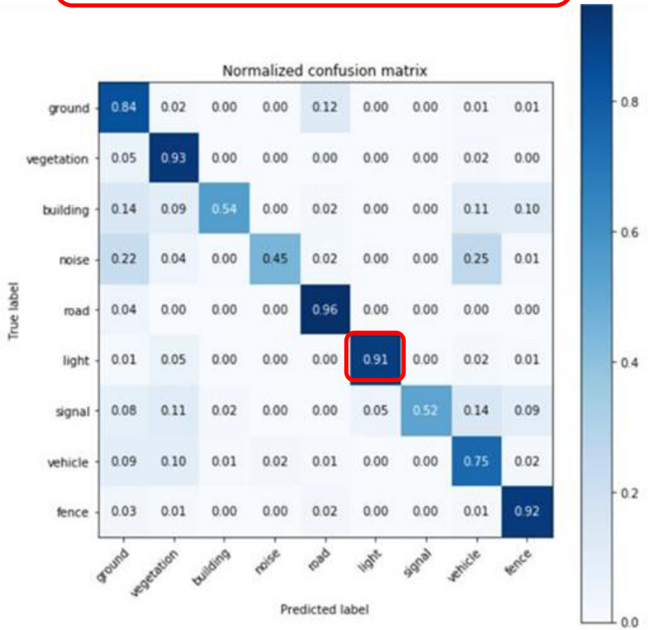
Final Result of ML Algorithm (Comparison)

```
Normalized confusion matrix
predict score is: 0.8357713048923584
```

```
Normalized confusion matrix
predict score is: 0.9144521251694725
```



Prediction result from the model trained without 2D features

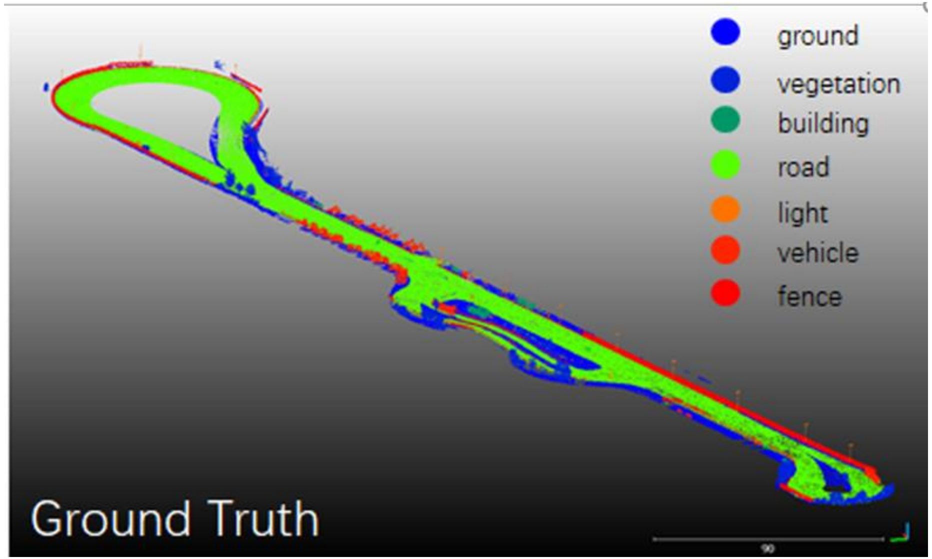


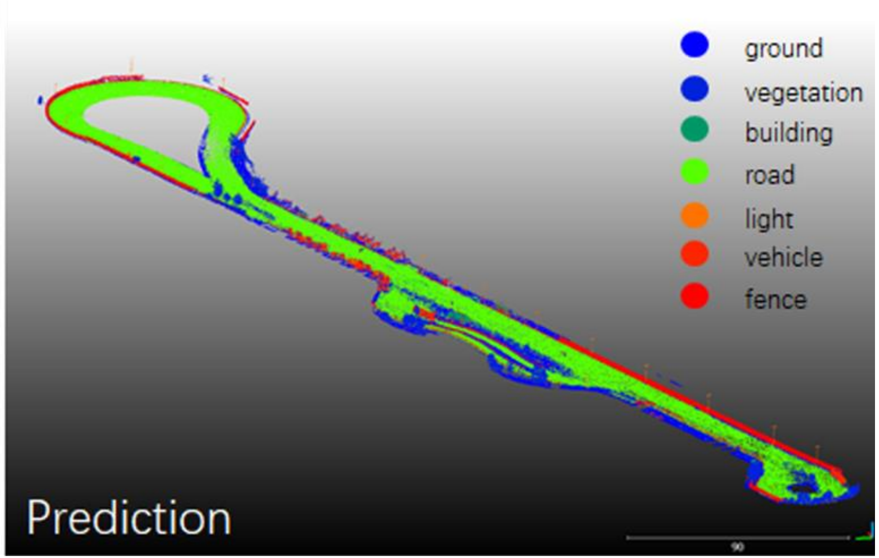
Prediction result from the model trained with 2D features

1. Overall accuracy is improved from 83.6% to 91.4%.

2. The prediction precision for each class is improved, especially 31% better for building and 13% better for light.

Final Result of ML Algorithm (Part1, industrial area)

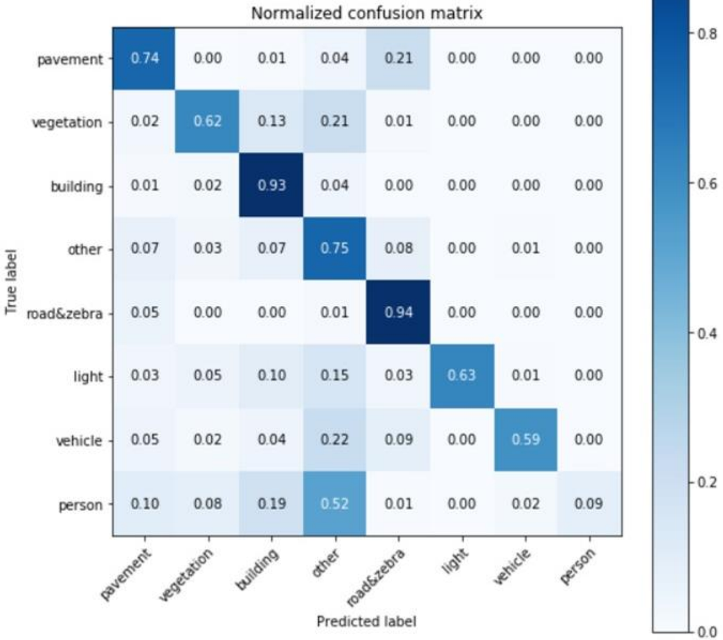




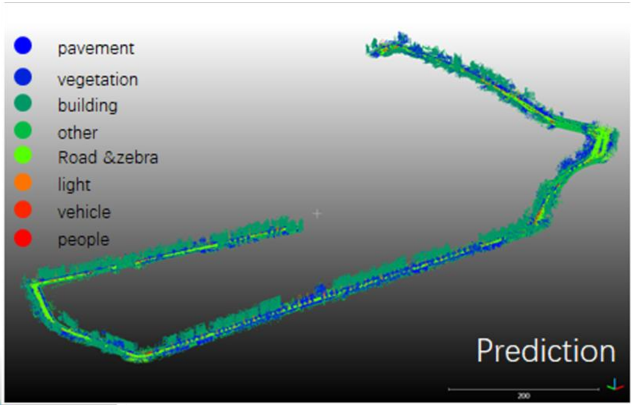
1. Overall accuracy is 91.4%.
2. Precision of class “light” is 91%

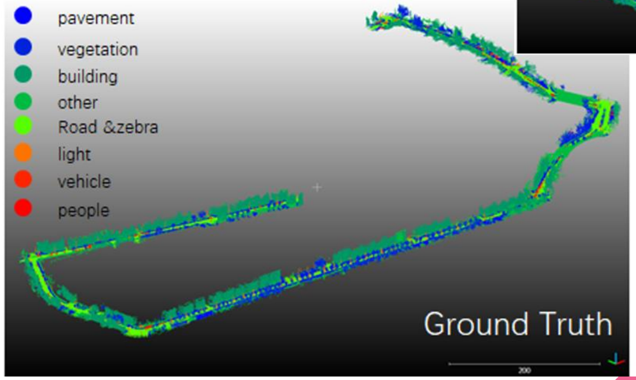
Final Result of ML Algorithm (086b, residential

```
Normalized confusion matrix
predict score is: 0.858173927252852
```

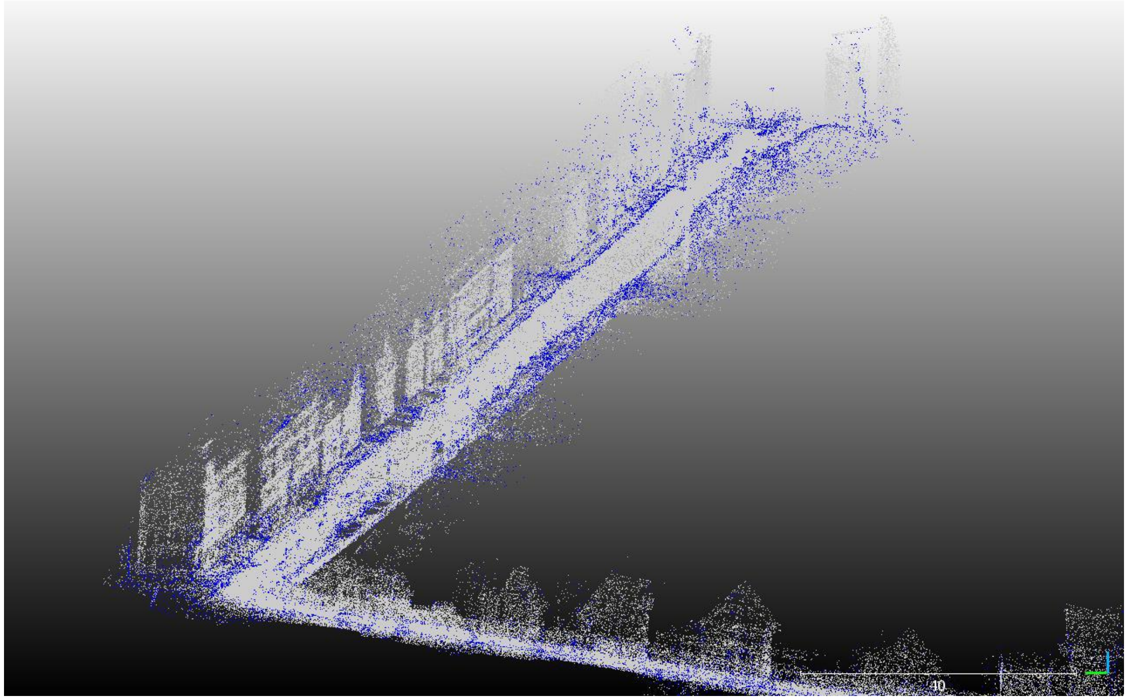


True label \ Predicted label	pavement	vegetation	building	other	road&zebra	light	vehicle	person
pavement	0.74	0.00	0.01	0.04	0.21	0.00	0.00	0.00
vegetation	0.02	0.62	0.13	0.21	0.01	0.00	0.00	0.00
building	0.01	0.02	0.93	0.04	0.00	0.00	0.00	0.00
other	0.07	0.03	0.07	0.75	0.08	0.00	0.01	0.00
road&zebra	0.05	0.00	0.00	0.01	0.94	0.00	0.00	0.00
light	0.03	0.05	0.10	0.15	0.03	0.63	0.01	0.00
vehicle	0.05	0.02	0.04	0.22	0.09	0.00	0.59	0.00
person	0.10	0.08	0.19	0.52	0.01	0.00	0.02	0.09





Final Result of ML Algorithm (086b)



Points assigned to wrong classes are shown in blue

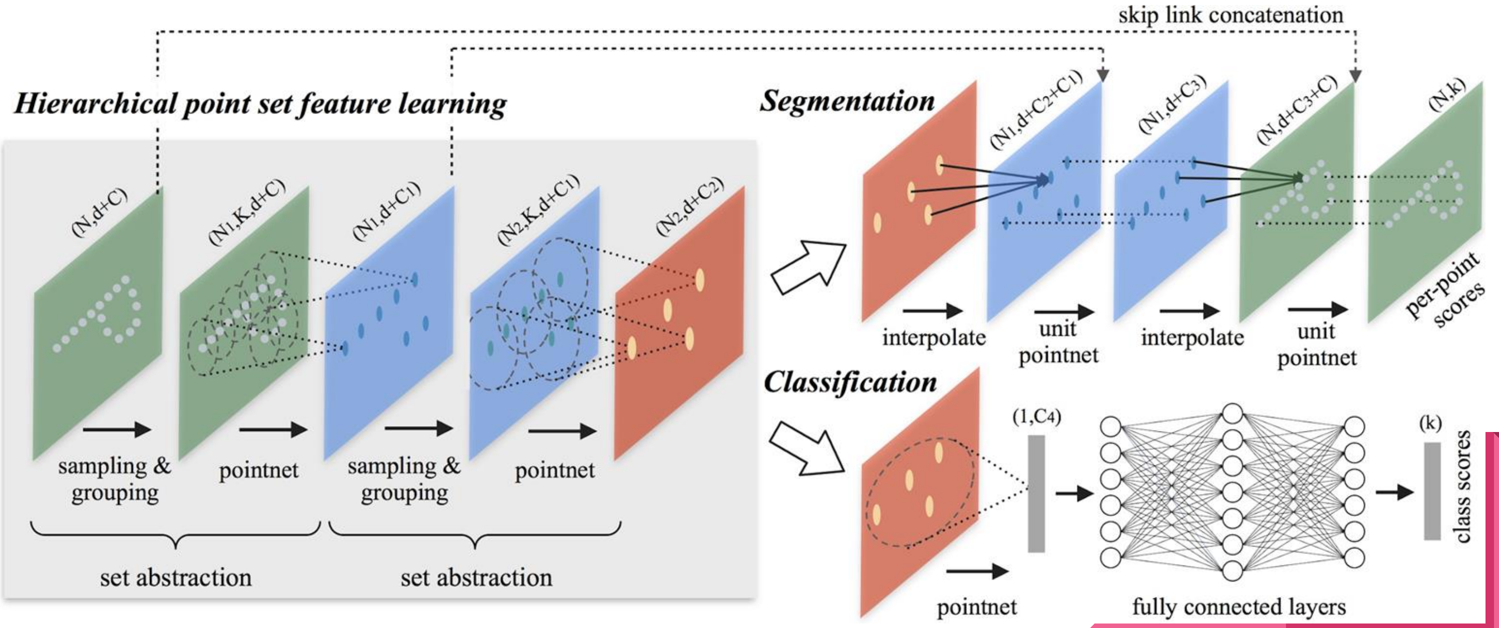
1. The overall accuracy for 086b is 85.8%, about 6% less than the accuracy of Part1.

2. Many points of “pavement” is wrongly predicted as road, and it’s understandable.

3. Precision of “light” is lower in 086b. Possible reason: there are many trees and the light may sometime hided in the branches of the trees.

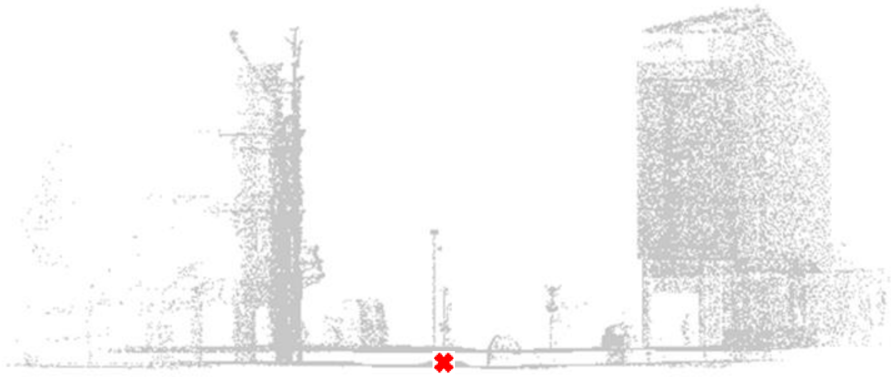
PointNet++

- Unordered point set as input
- Custom partitioning
- PointNet on each partition



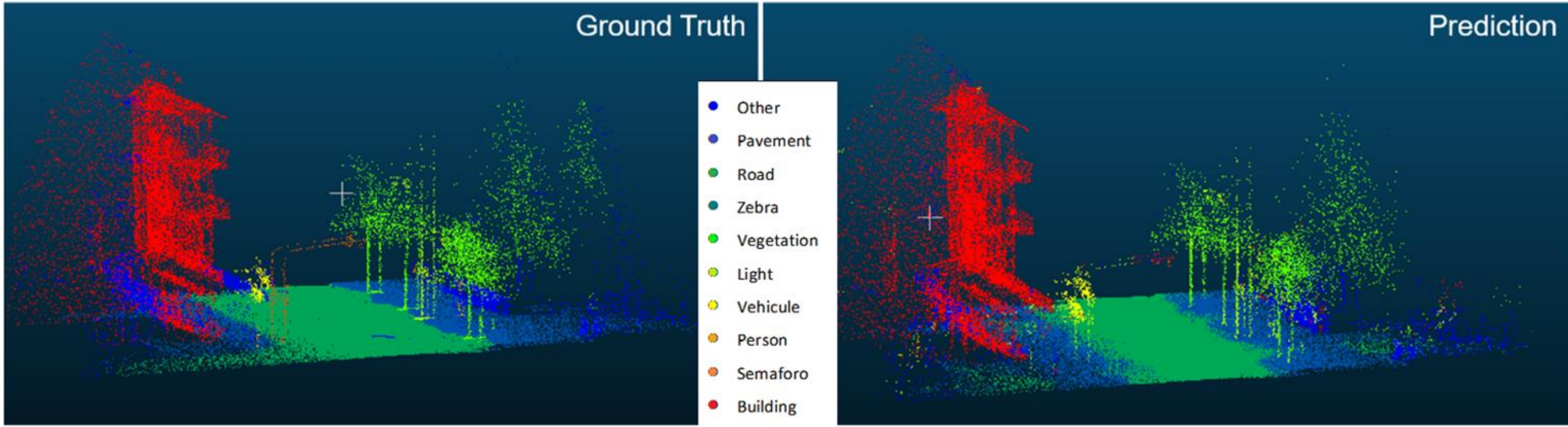
Data preprocessing for PointNet++

Cut scenery into chunks and center each chunk



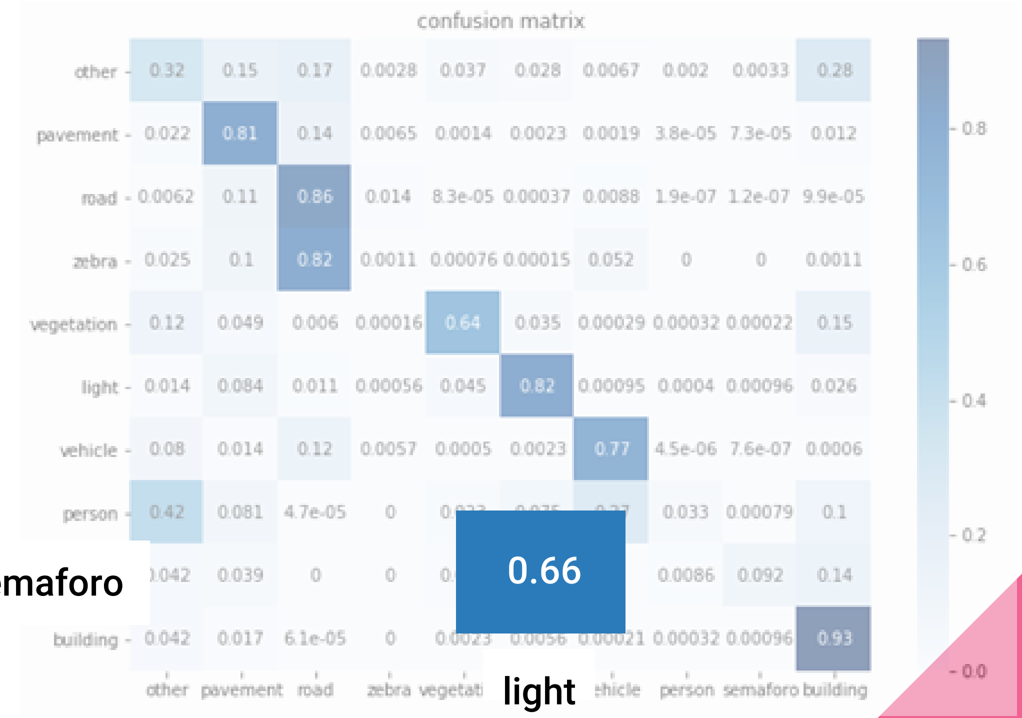
Final result of PointNet++: Chunk of “086b_classified”

overall accuracy: 87%
average IoU (ignoring) label ‘other’: 47%



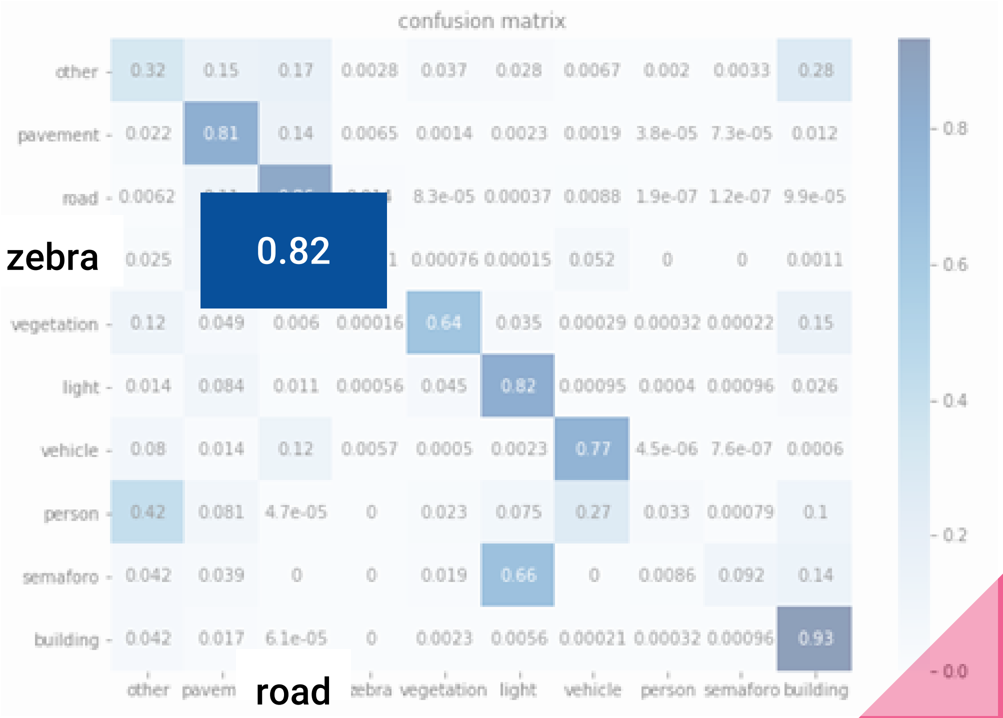
Final result of PointNet++: Chunk of “086b_classified”

overall accuracy: 87%
average IoU (ignoring) label ‘other’: 47%



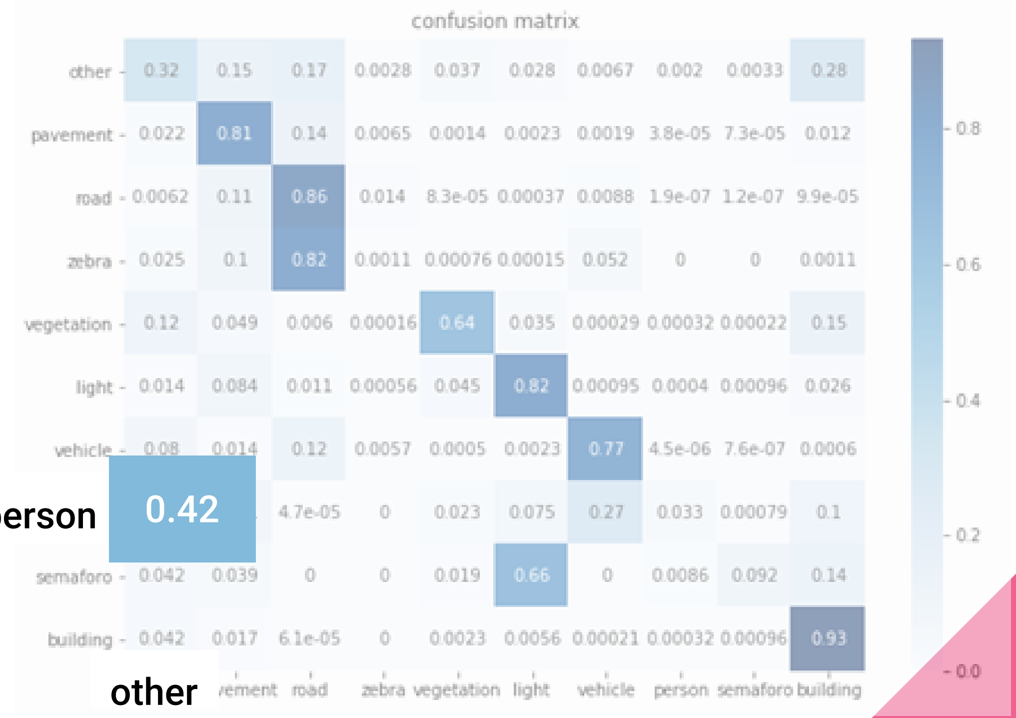
Final result of PointNet++: Chunk of “086b_classified”

overall accuracy: 87%
average IoU (ignoring) label ‘other’: 47%



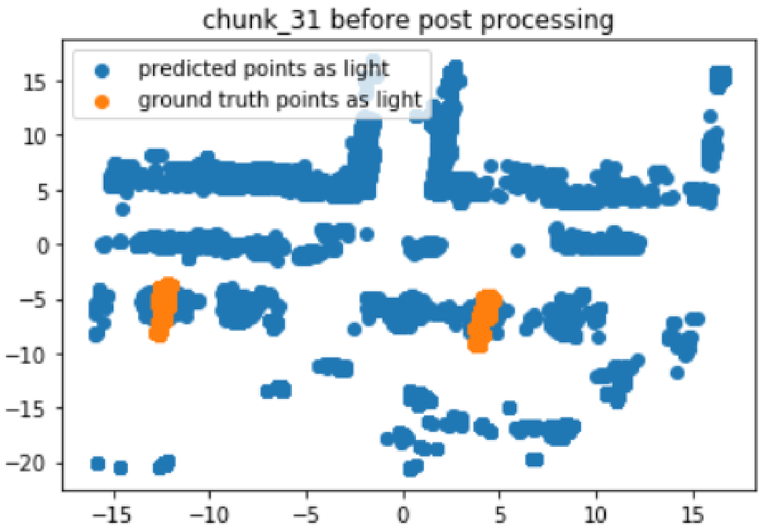
Final result of PointNet++: Chunk of “086b_classified”

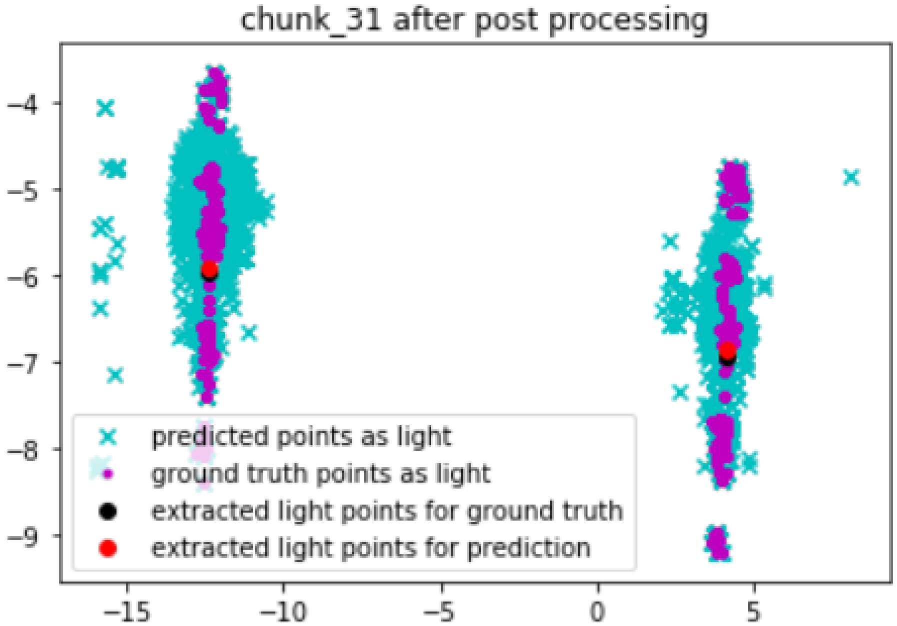
overall accuracy: 87%
average IoU (ignoring) label ‘other’: 47%



Extraction of light poles: 1st approach

- calculate k clusters on the predicted points as light with KMean
- select only the clusters with the largest number of points
- calculate the intersection over union (IoU) of those selected clusters on the ground truth
- iterate for different numbers of clusters k
- pick the cluster with the highest IoU

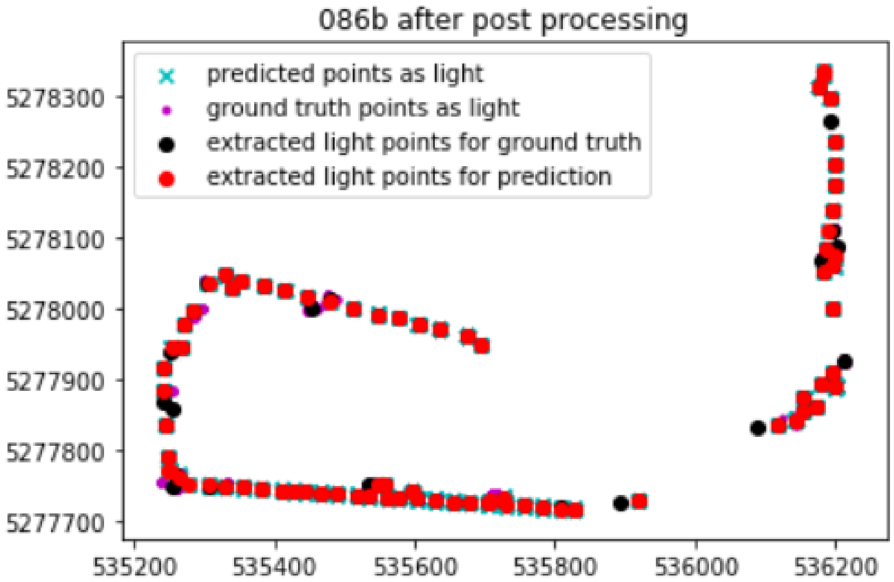


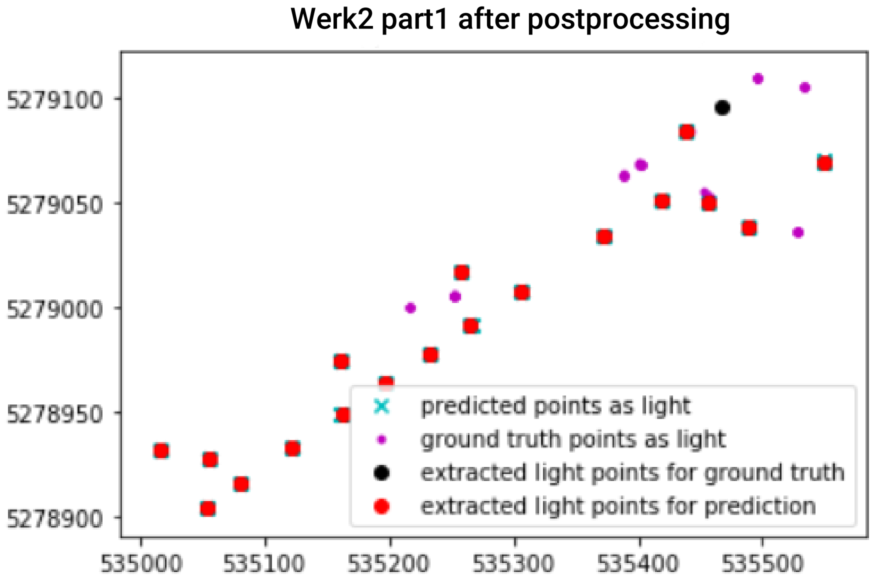


Extraction of light poles: final approach

Instead of Kmean, we use DBSCAN: clustering done with input based on distance between points and not number of cluster => As the number of cluster varies a lot from a scene to another, DBSCAN is more appropriate.

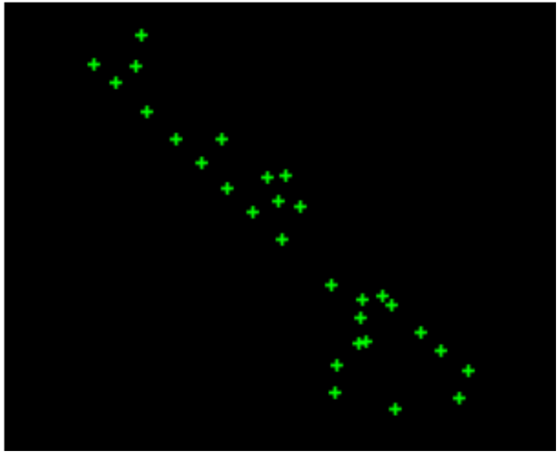
Parameters: for ground truth, t=0.003, min samples=10, for prediction t=0.006, min samples=10



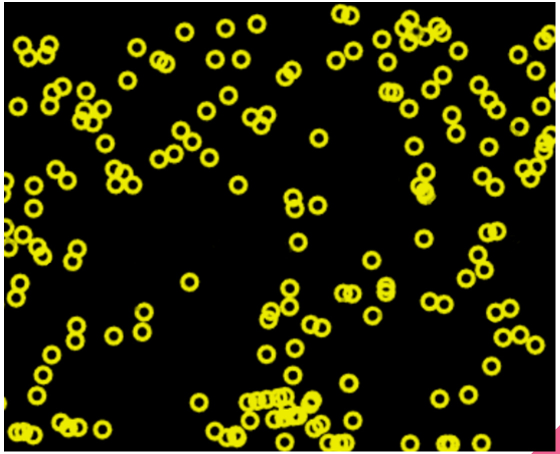


Matching of Mobile Mapping Data to GCPs

- Goal: Match the base points of poles extracted from the segmentation to Ground Control Points derived from satellite data



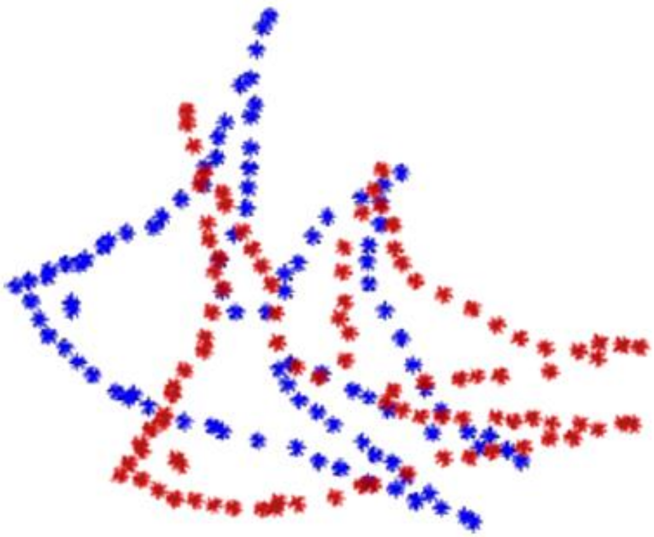
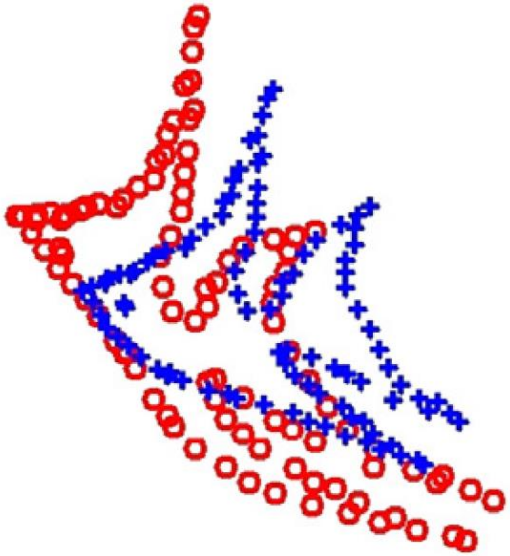
PCPs



local GCPs

Point Set Registration

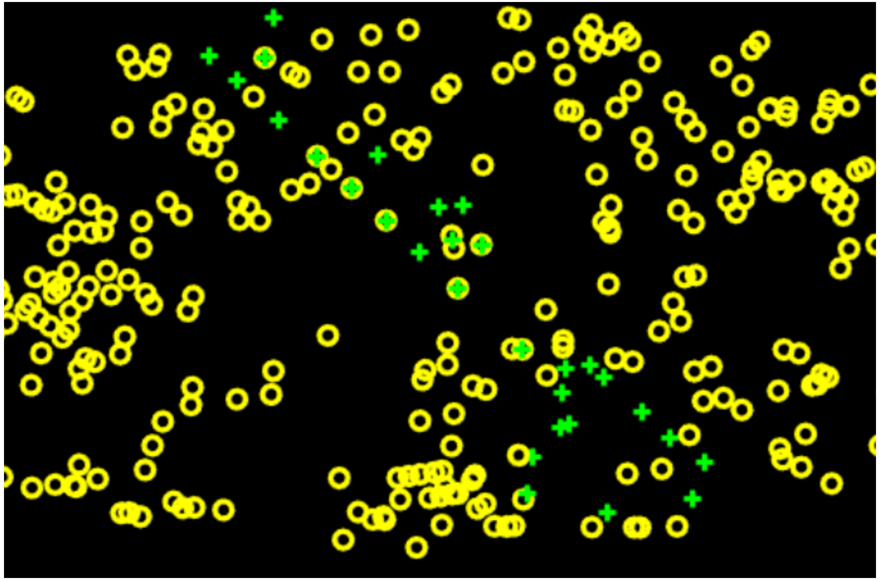
- **Match** *source set* **to** *target set*

Source: [14]

Matching undistorted data

- GPS mostly accurate during acquisition: Inaccuracy as slight “noise”



Rigid Point Set Registration: ICP

Steps:

1. For each point in source set, compute closest point in target set
2. Compute rigid transformation that minimizes distance

Source: [9]

Rigid Point Set Registration: ICP

Steps:

1. For each point in source set, compute closest point in target set
2. Compute rigid transformation that minimizes distance

- Computationally simple
- Converges monotonically to closest local minimum

Source: [9]

Rigid Point Set Registration: ICP

Steps:

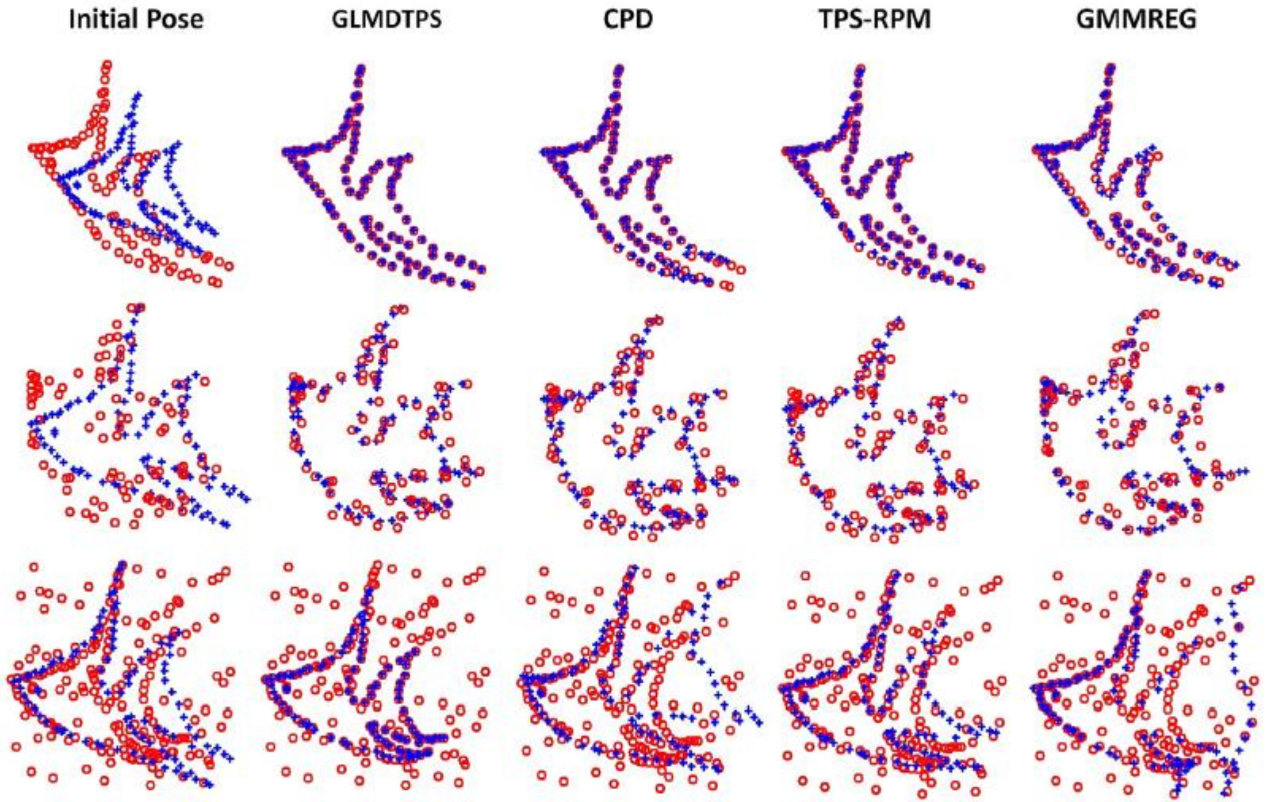
1. For each point in source set, compute closest point in target set
2. Compute rigid transformation that minimizes distance

- Computationally simple
- Converges monotonically to closest local minimum

- Provides correct results for non-distorted data
- Nearest Neighbour matching gives same results

Source: [9]

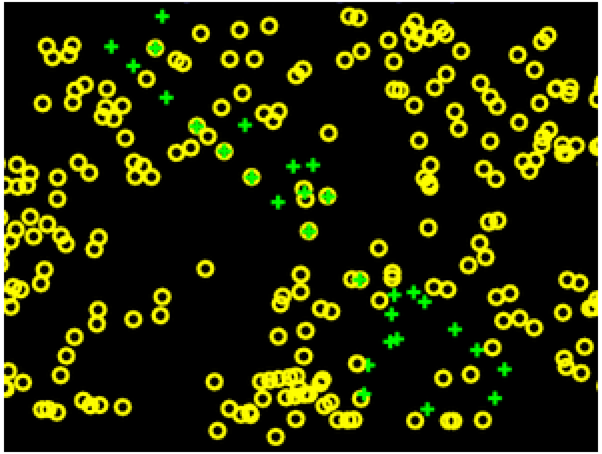
Non-rigid Point Set Registration: GLMDTPS

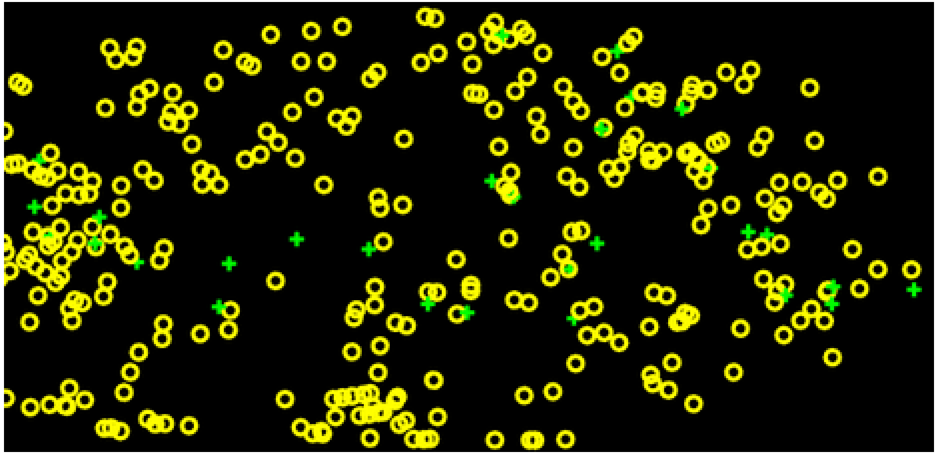


Source: [14]

Testing: GLMDTPS

- Incorrect results for non-distorted data





- Better transformation given only matches

Idea: Iterated GLMDTPS

Steps:

1. Nearest Neighbour Matching
2. GLMDTPS on matches
- Reduce matching radius in later iterations

Performance:

- Correct matches
- Better transformation than ICP

Results: Iterated GLMDTPS

On non-distorted data:

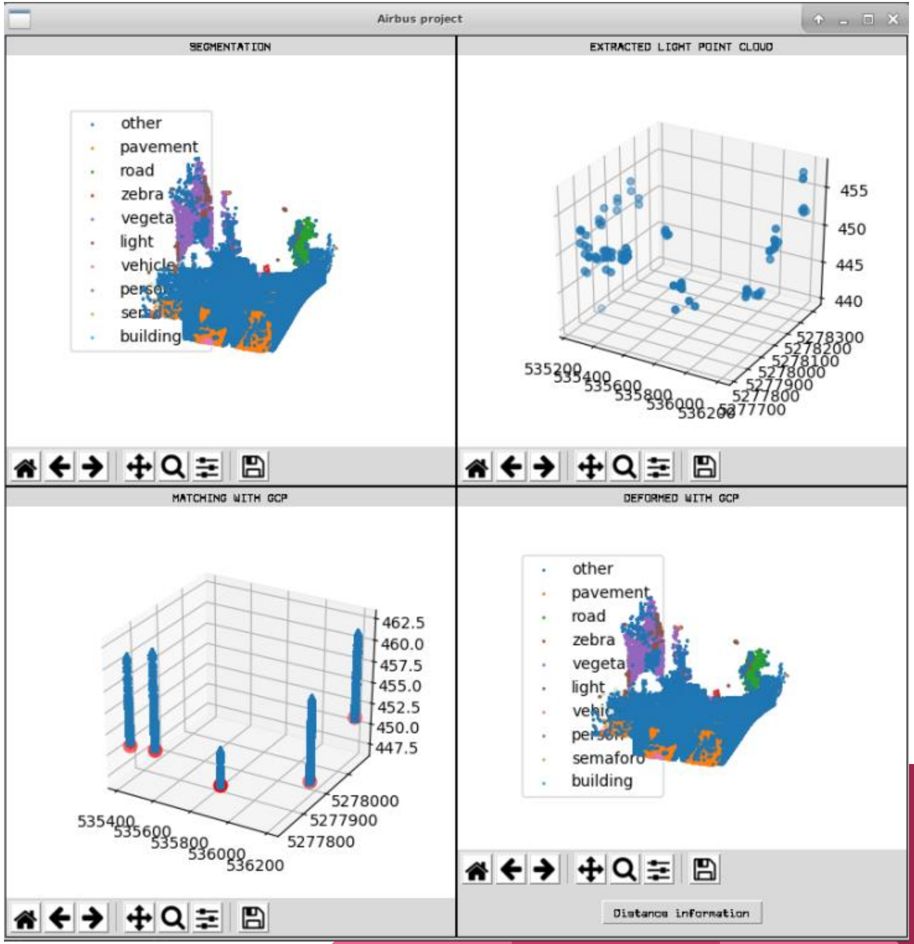
ID	*OD*	*ND*
5	0.551	0.311
6	0.565	0.152
7	0.744	0.0516
9	0.741	0.231
10	0.421	0.045
24	0.613	0.163
27	0.513	0.155
avg	***0.592***	***0.158***

Distances in meter

Software development

Creation of a software to integrate all algorithms together:

- Classification:
  - by Random Forest: feature extraction and classification
  - by PointNet++: classification of each chunks
- light extraction
- matching
- deformation



Conclusion

Segmentation & Classification

- Models trained on urbanized area in Germany => probably **not robust to a different type of data**

➢ To improve the ML algorithm, training with **data from different environment** will be necessary.

Conclusion

- Random Forest algorithm is **robust to the size** of the input.
- Machine learning (ML) algorithm needs to extract features that is **time consuming**
- PointNet++ algorithm is much **faster**

➢ The processing time for ML algorithm can be decreased by supporting **parallelization**.

Conclusion

Light Pole Extraction

- Improve the fine-tuning for clustering
- Create or find a Machine Learning or Deep Learning approach for clustering and pole extraction

➢ include this architecture to the segmentation approach

Conclusion

<u>Matching</u>

- Iterated algorithm that
  - produces **correct** matches in test cases
  - **robust** to high outlier-ratio
  - provides **non-rigid transformation** as base for correction of LiDAR data

- Outlook:
  - Improve transformation

Reference:

[1] S. Montazeri, C. Gisinger, M. Eineder and X. X. Zhu, "Automatic Detection and Positioning of Ground Control Points Using TerraSAR-X Multiaspect Acquisitions," IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 5, pp. 2613-2632, 1 5 2018.

[2] E. Grilli, F. Menna and F. Remondino, "A review of point clouds segmentation and classification algorithms," in International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 2017.

[3] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017.

[4] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer and S. Silvio, 3d semantic parsing of large-scale indoor spaces, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2016.

[5] Y. Wang, T. Shi, P. Yun, L. Tai and M. Liu, "PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud," 17 7 2018.

[6] C. R. Qi, L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," 7 6 2017.

[7] G. Elbaz, T. Avraham and A. Fischer, "3D point cloud registration for localization using a deep neural network auto-encoder," in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017.

[8] J. Huang and S. You, "Point cloud matching based on 3D self-similarity," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2012.

[9] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, 1992.

[10] S. Gold, R. A. Rangarajan, C.-P. Lu, S. Pappus and E. Mjolsness, "NEW ALGORITHMS FOR 2D AND 3D POINT MATCHING: POSE ESTIMATION AND CORRESPONDENCE Point-matching Pose estimation Correspondence Neural networks Optimization Softassign Deterministic annealing Affine transformation," 1998.

[11] G. P. Penney, P. J. Edwards, A. P. King, J. M. Blackall, P. G. Batchelor and D. J. Hawkes, "A stochastic iterative closest point algorithm (StochastICP)," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2001.

[12] J. Yang, H. Li, D. Campbell and Y. Jia, "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 11, pp. 2241-2254, 1 11 2016.

Reference:

[13] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," Computer Vision and Image Understanding, vol. 89, no. 2-3, pp. 114-141, 2003.

[14] Y. Yang, S. H. Ong and K. W. C. Foong, "A robust global and local mixture distance based non-rigid point set registration," Pattern Recognition, vol. 48, no. 1, pp. 156-173, 1 1 2015.

[15] M. Weinmann, B. Jutzi, C. Mallet and M. Weinmann, "GEOMETRIC FEATURES and THEIR RELEVANCE for 3D POINT CLOUD CLASSIFICATION," in ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2017.

[16] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," Pattern Recognition, vol. 48, no. 6, pp. 2043-2053, 1 6 2015.

[17] M. Weinmann, B. Jutzi and C. Mallet, "Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features," ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vols. II-3, pp. 181-188, 7 8 2014.

[18] A. Myronenko, X. Song, M. A. Carreira-Perpiñán and P. Perpiñán, "Non-rigid point set registration: Coherent Point Drift".

[19] T. Hackel, J. D. Wegner and K. Schindler, "FAST SEMANTIC SEGMENTATION of 3D POINT CLOUDS with STRONGLY VARYING DENSITY," in ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016.

[20] J. Balado, L. Díaz-Vilariño, P. Arias and L. M. González-Desantos, "Automatic LOD0 classification of airborne LiDAR data in urban and non-urban areas," European Journal of Remote Sensing, vol. 51, no. 1, pp. 978-990, 1 1 2018.

[21] M. Weinmann, M. Weinmann, C. Mallet and M. Brédif, "A classification-segmentation framework for the detection of individual trees in dense MMS point cloud data acquired in urban areas," Remote Sensing, vol. 9, no. 3, 1 3 2017.

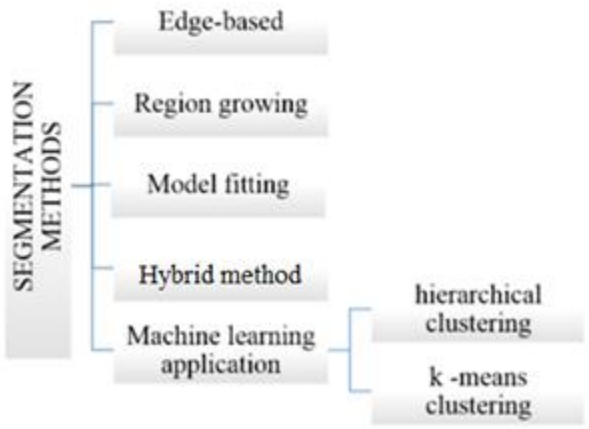
[22] O. Lang, P. Lumsdon, D. Walter, J. Anderssohn, W. Koppe, J. Janoth, T. Koban and C. Stahl, "Development of Operational Applications for TerraSAR-X", Remote Sensing, vol. 10, no. 10, p. 1535, 2018.

[23] J. Gitlin, "The most detailed maps of the world will be for cars, not humans", Ars Technica, 2019. [Online]. Available: https://arstechnica.com/cars/2017/03/the-most-detailed-maps-of-the-world-will-be-for-cars-not-humans/. [Accessed: 01- Aug- 2019]

Thank You!

Backup Segmentation methods

Synthetic representation of the segmentation methods



Backup Extracted Feature-1

	Feature Name	Description of Feature
1	KDistance	The Euclidean distance to a point's 8-th nearest neighbour
2	LocalReachabilityDistance	The inverse of the mean of all reachability distances for a neighbourhood of points
3	LocalOutlierFactor	The mean of all LocalReachabilityDistance values for the neighbourhood
4	NNDistance	Similar to KDistance
5	Eigenvalue2	The largest Eigenvalue based on its 8-nearest neighbours in 3D.
6	Eigenvalue1	The second-largest Eigenvalue based on its 8-nearest neighbours in 3D.
7	Eigenvalue0	The smallest Eigenvalue based on its 8-nearest neighbors in 3D.
8	Rank	Computed by SVD with 8-nearest neighbours. Point sets with rank 1 correspond to linear features, while rank 2 correspond to planar features and rank 3 corresponds to a full 3D feature.
9	NormalX	The normal is taken as the eigenvector corresponding to the smallest eigenvalue.
10	NormalY	
11	NormalZ	
12	Curvature	Smallest eigenvalue divided by the sum of all three eigenvalues.
13	RadialDensity	The density of points in a sphere of a given radius. Here the radius is 2.
14	Coplanar	Technique to performs a fast and robust octree-based segmentation of approximately coplanar clusters of samples. [16]

Backup Extracted Feature-2

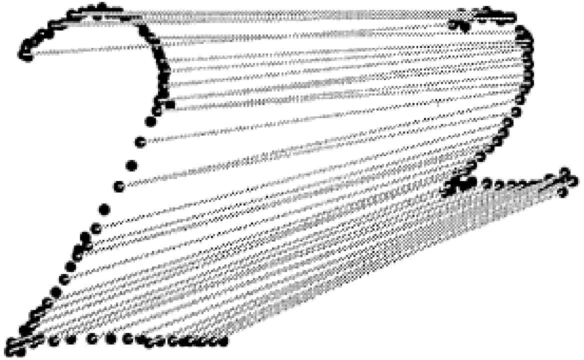
15	Linearity	(Eigenvalue0 - Eigenvalue1) / Eigenvalue2
16	Planarity	(Eigenvalue1 - Eigenvalue0) / Eigenvalue2
17	Scattering	Eigenvalue0 / Eigenvalue2
18	Omnivariance	(Eigenvalue0* Eigenvalue1* Eigenvalue2)**(1/3)
19	Anisotropy	(Eigenvalue2 - Eigenvalue0) / Eigenvalue2
20	Eigentropy	-(Eigenvalue0 * log(Eigenvalue0) - Eigenvalue1 * log(Eigenvalue1) - Eigenvalue2 * log(Eigenvalue2)
21	Eigen_Sum	The sum of all three eigenvalues
22	Curvature_Change	Eigenvalue0 / (Eigenvalue0 + Eigenvalue1 + Eigenvalue2)
23	density_2d	The density of points, which are projected to X-Y plane, in a circle of a given radius. Here the radius is 20 cm.
24	e1_2d	The smallest Eigenvalue based on its neighbours within 20 cm in 2D.
25	e2_2d	The largest Eigenvalue based on its neighbours within 20 cm in 2D.
26	Z	Coordinate in Z-axis.

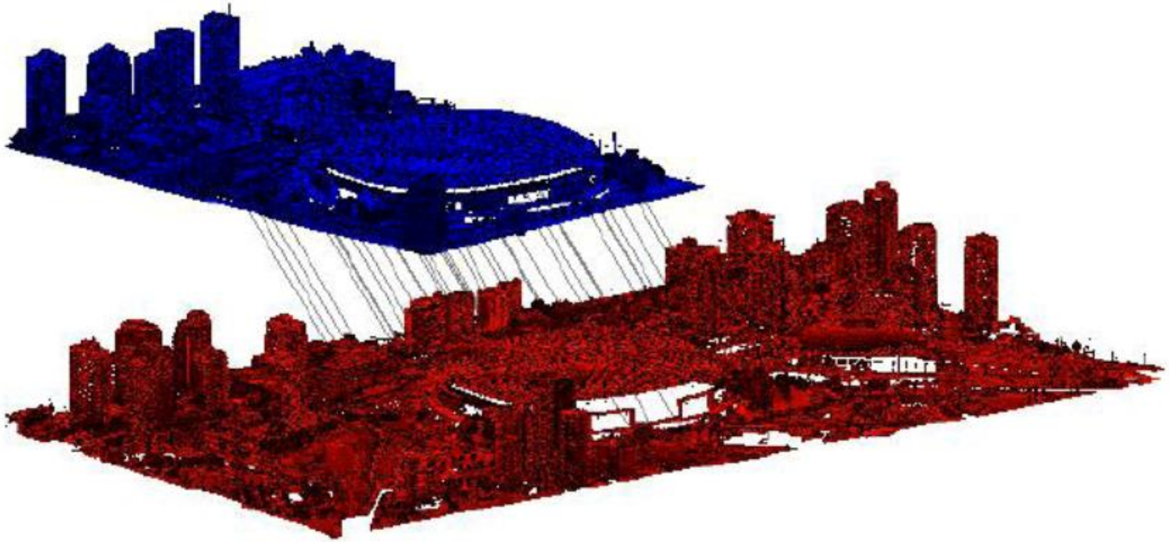
Comparison of Point Cloud file format

TYPE	EXTENSION(S)	DESCRIPTION	READ	WRITE	BINARY/ASCII	POINT CLOUD(S)	MESH(ES)	OTHER	FEATURES
BIN	.bin	CloudCompare own format	X	X	binary	>1	>1	>1	Normals, colors (RGB), scalar fields (>1), labels, viewports, display options, etc.
ASCII	.asc,.txt,.xyz,.neu,.pts	ASCII point cloud file (X,Y,Z,etc.)	X	X	ASCII	1	0	0	Normals, colors (RGB), scalar fields (all)
LAS	.las	ASPRS lidar point clouds	X	X	binary	1	0	0	Colors (RGB) and various scalar fields (see LAS 1.4 specifications)
E57	.e57	ASTM E57 file format	X	X	mixed	>1	0	Calibrated picture(s)	Normals, colors (RGB or I), scalar field (intensity)
PCD	.pcd	Point Cloud Library format	X	X	binary	>1	0	0	Colors (RGB), normals, scalar fields (>1)
PLY	.ply	Stanford 3D geometry format (cloud or mesh)	X	X	both	1	1	0	Normals, colors (RGB or I), one ore several scalar fields, a single texture
OBJ	.obj	Wavefront mesh	X	X	ASCII	1	>1	Polyline(s)	Normals, materials and textures
VTK	.vtk	VTK file format (triangular mesh or cloud only)	X	X	ASCII	1	1	0	Normals, colors (RGB), scalar field(s) (>1)
STL	.stl	STereoLithography file format(mesh)	X	X	ASCII	0	1	0	Normals
OFF	.off	Object File Format (mesh)	X	X	ASCII	0	1	0	0
FBX	.fbx	Autodesk (Filmbox) File Format	X	X	ASCII or BINARY	0	>1	0	Normals, colors (RGB), materials and textures
DXF	.dxf	Autocad DXF format	X	X	ASCII	>1	>1	polyline(s)	Normals, colors (RGB)
SHP	.shp	ESRI Shape file format	X	X	binary	>1	0	Polyline(s), polygon(s), contour plot(s), etc.	Scalar fields (1 per entity)

[1] FILE I/O - CloudCompareWiki", Cloudcompare.org, 2019. [Online]. Available: https://www.cloudcompare.org/doc/wiki/index.php?title=FILE_I/O. [Accessed: 27- Jul- 2019]

Point Set Registration





Case 2: Matching distorted data

- Loss of GPS signal during acquisition: possibly non-linear distortion
- Approach: Non-rigid point set registration