TrueView: A LIDAR Only Perception System for Autonomous Vehicle

Mohammed Yazid Lachachi[1]
University of sciences and technologies d'Oran - Mohamed-Boudiaf, LMSE, Algeria
yazid.lachachi@univ-usto.dz

Mohamed Ouslim
University of sciences and technologies d'Oran - Mohamed-Boudiaf, LMSE, Algeria
ouslim@yahoo.com

Smail Niar
Université Polytechnique Hauts-de-France, France
smail.niar@uphf.fr

Abdelmalik Taleb-Ahmed
Université Polytechnique Hauts-de-France, France
Abdelmalik.Taleb-Ahmed@uphf.fr

Abstract

Real time perception and understanding of the environment is essential for an autonomous vehicle. To obtain the most accurate perception, existing solutions propose to combine multiple sensors. However, a large number of embedded sensors in the vehicle implies to process a large amount of data thus increasing the system complexity and cost. In this work, we present a novel approach that uses only one LIDAR sensor. Our approach enables reducing the size and complexity of the used machine learning algorithm. A novel approach is proposed to generate multiple 2D representation from 3D points cloud using the LIDAR sensor. The obtained representation solves the sparsity and connectivity issues encountered with LIDAR-based solution.

2012 ACM Subject Classification Computing methodologies → Computer vision representations

Keywords and phrases Ranging Data, Computer Vision, Machine Learning

Digital Object Identifier 10.4230/OASIcs.ASD.2019.8

Category Interactive Presentation

1 Introduction

Camera, Stereo-camera, RADAR, and LIDAR are the sensors that provide our view of the world. To enable machines to understand our world, multiple approaches have been developed. Camera-based systems are the most dominant [2, 19, 3]. Recently, approaches using depth information generated from stereo-camera have started to gain popularity, as they have proven to improve accuracy. Knowing that camera and stereo-camera are almost the same since the stereo-camera enable the recovery of depth information compared to the monocular camera. Similar Machine Learning (ML) algorithms for multi-class object detection and segmentation are used with minor modifications. On the other hand, the LIDAR has fallen behind performing mainly detection tasks [20, 17, 14] as the variance in the points coordinates ease the task.

The richness and variation of data fed to an ML determine its complexity and size. These two factors have a direct impact on network accuracy. Images are well structured and rich in information, as each pixel has a non-zero value and adjacent to another pixel on a 2D plane.

[1] Corresponding author

However, this richness adds more challenges like illumination conditions, shadows, colors, and textures. All these parameters require an ML architecture to have more layers to capture abstract aspects like car tires and window edges. A large amount of filters is also needed to accommodate the infinite changes that we can obtain like car shapes and the possible textures of an object. The LIDAR is an active sensor capable of operating in the majority of weather conditions and at any daytime. Unfortunately, due to its rotatory movement, the obtained points are unstructured, unordered and have no connectivity between them. These weaknesses make it difficult to use for multiple purposes when compared to the camera sensor.

In this work, we propose a new approach to generate multiple 2D representations using only a 3D LIDAR. These representations reflect the real world and are immune to camera limitations. These representations allow us to see the silhouette of objects without the added textures, colors and illumination condition. Some of the generated representations are an up-sampled version of the ranging data projected into an image, where the new values are filled using interpolation. Further, the obtained representations simplify the ML architecture and reduce its size thus gaining in processing time and getting closer to real time.

Our Contributions are as follow:

- Novel range data up-sampling process to enable camera-based approaches to be used with LIDAR data.
- Robust normal map estimation.
- We present a new approach to reducing the complexity of current machine learning algorithms and their computation overhead.

In Section 2, we present the related work to up-sampling the ranging data, then we introduce the concept in Section 3 and method to generate the representation in Section 4. We then conclude the paper by presenting the benefits of the proposed representations for machine learning algorithms.

2 Related Work

The first attempt to generate dense information from range data which corresponds to LIDAR points cloud in our case was made by Diebel et al [4]. The authors proposed a range of data and image fusion to generate a dense depth map from a low-resolution one. They fed the projected range data with the image to Markov Random Field regressor, then they generate the high-resolution depth map by iterating through the MRF. Their experiments showed that textured surfaces had a negative impact on the estimated values as color value changed.

In [1], Andreasson et al. proposed 5 interpolation methods as a replacement to the MFR proposed by [4]. The nearest range (NR) and multi-linear interpolation (MLI) are color free estimations. The color was considered in the two modified version NRC and LIC. These approaches relied on an empirically predefined constant and parameter-free version of LIC was further introduced as PLIC. Their results in a laboratory environment under controlled illumination has proved that color improved accuracy. However, experiments in real conditions showed that color had the opposite effect. This limitation is caused mainly by variant illumination conditions.

In [10], Yuhang et al proposed a new approach to generate depth, height and a normal map from the range data and image. They used different features to determine the value of the new pixels. The features were used to minimize the effects of textured surfaces and illumination conditions on the final result. However, their result accuracy varies according to the used window size. These representations have proven their effectiveness in the work of [9].

Further, the KITTI benchmark recently launched a competition to generate the depth map from the LIDAR data, when used solely or in conjunction with a camera. This competition saw the introduction of machine learning algorithms, where the best at the time of writing this paper is [15]. The authors considered the up-sampling task as a deep regression problem. They feed depth data and color sequences to the regression network.

Ku et al [13] proposed a simpler approach to up-sample the range data, by using a computational fast process like morphological dilatation, filling and blurring. They were able to generate the depth map in 11ms on a CPU and their results were comparable to results obtained by convolutional neural networks.

In [16], Schneider et al. propose to use the edge information to guide the up-sampling. The authors considered the up-sampling as a global energy minimization problem. Although the approach was developed for time-of-flight cameras, the authors were able to extend it to range data and compete in the KITTI depth completion.

Dimitrievski et al [5] proposed a morphological neural network, their approach approximates morphological operations using a novel Contraharmonic Mean Filter layers. The proposed network is modified U-net architecture with morphological layers.

Multiscale networks have produced very interesting results in the KITTI challenge. In [11], Huang et al proposed a hierarchical multi-scale network, they introduced three sparsity-invariant operations. These operations were used to create a sparsity-invariant multi-scale encoder-decoder network. The method was developed to deal with the sparsity problem in the range data generated by LIDAR.

Although these approaches are able to generate dense depth map representations and more representations in some of them, the following problems arise:

- A degraded image quality is obtained under unfavorable illumination conditions.
- A large number of points must be used for the estimation.
- The whole process must be repeated much time to obtain a different representation. This factor increases the processing time with the number of generated representations.

Existing approaches use a window to retrieve the close points for the estimation. A wide window utilization could augment greatly the number of used points, thus generating false estimations at high variance regions. Furthermore, the number of points cannot be reduced due to adapt the algorithm to the hardware processing constraints. In this work, we consider a newly generated pixels as part of a triangle surface, where the value is an interpolation between three closest points. This approach removes the need for a fixed window and allows the generation of the representations without the need to repeat the process.

3 Optimized 3D representation for LIDAR data

Real objects are in general modeled as 3D model mesh consisting of 3D points and their connectivity list. In this work, we propose to generate a 3D mesh from the LIDAR ranging data. The high accuracy of this data enables the creation of a mesh that mimics the environment of the vehicle. Greater control over desired the desired output resolution is then possible. This obtained mesh enable the generation of a set of representations, which we formulate as follows:

$$\mathbb{R}^3 \equiv \{\mathbb{R}^2, \mathbb{R}^2, \mathbb{R}^2, ...\} \tag{1}$$

Figure 1 presents a 3D mesh obtained by our approach and Figure 2 shows the multiple obtainable representations. The Figures 2.b to 2.d representations show (respectively) the rendered X, Z, Y and the reflection of the surface. These representation using only the mesh

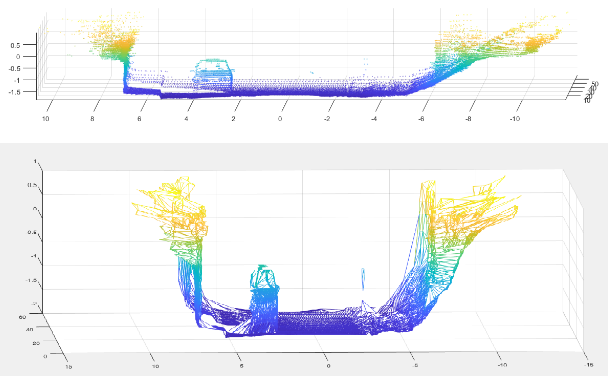

Figure 1 Reconstructed mesh from point cloud.

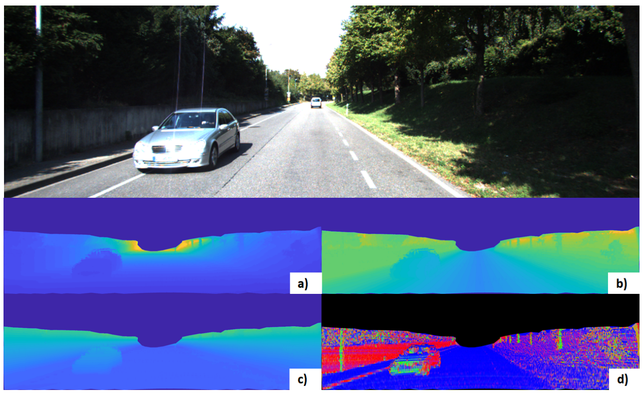

Figure 2 Rendered representations (from top to bottom) a) Initial image, b) up-sampled x coordinates, c) Up-sampled z coordinates, d) Up-sampled y coordinates, e) Reflection data, f) Estimated normal map (final result).

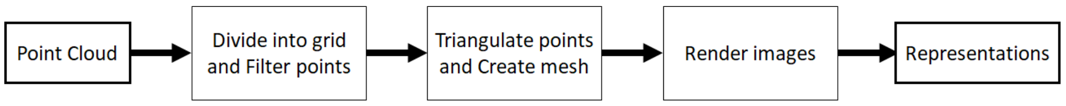
Point Cloud
Divide into grid and Filter points
Triangulate points and Create mesh
Render images
Representations

Figure 3 Overall Algorithm.

and points coordinates in the real world. The last representation is a 3-channel image, where each channel contains one of the normal parameter estimated from the three points making the surface. In the last representation (Figure 2.f), the coordinates of the points and their connectivity were used to generate the results.

Further results can be acquired, such as object contours, using the mesh connectivity list and the angle between the points. This last feature is very interesting as it allows to obtains the contours with a relatively reduced processing. In the next section, we present our method to obtain the connectivity list.

4 Method

A 3D mesh is a virtual construct composed of a 3D point set and their connectivity list. The LIDAR generates an accurate point cloud from its environment. These points can be considered as the first half of the mesh. However, the rotatory movement, the angular speed, the number of receptors and the used algorithm have a great impact on the number of points. These factors have an important variance in the possible results obtained.

The steps to generate the representation are shown In Figure 3. We bring the reader attention to the fact that all the steps can be done in parallel.

4.1 Division and Filtering

Point clouds are unordered and have no connectivity by nature. Attempting to generate a mesh in this format will be time-consuming. A simpler and effective approach is to process the points in 2D space. This allows to reduce the complexity of the algorithm and gives faster processing compared to 3D space representation.

The projected point number can be also be reduced to accelerate the processing. Our experiments show that reducing the number of points will not greatly reduce the quality of the results. This point will be discussed in the experiments section.

After the projection, the points are divided into a grid with equal dimensions. The number of points in each grid cell is then reduced based on distance criteria. This distance controls the number of points used to create the mesh. In this work, the Manhattan distance is used for its computation simplicity. In Figure 4, reduced points with different distances are shown. We attract the reader attention to the blue rectangle. As can be seen, the details of the scene are preserved even with a big distance.

4.2 Triangulation and mesh creation

Delaunay triangulation is used to create the connectivity list between the points. Multiple variants of the algorithm exist from which we denote: Flip algorithms [12], Incremental [8, 7] and Divide and Conquer [6]. The Divide and conquer variant is chosen in our approach for two main reasons. First, generating these representations have to be as fast as possible. The work of Su et. Al [18] proved that this variant is the most performing approach. Secondly, the algorithm has been developed with GPUs in mind. The points in the grid cells can be processed separately and in parallel then merged at the end.

4.3 Rendering

The rendering step is where the up-sampling is performed. For each triangle, the value and positions of the delimiting points are used to interpolate the new values at rendering time. Figure 5 presents four examples of rendered triangles.

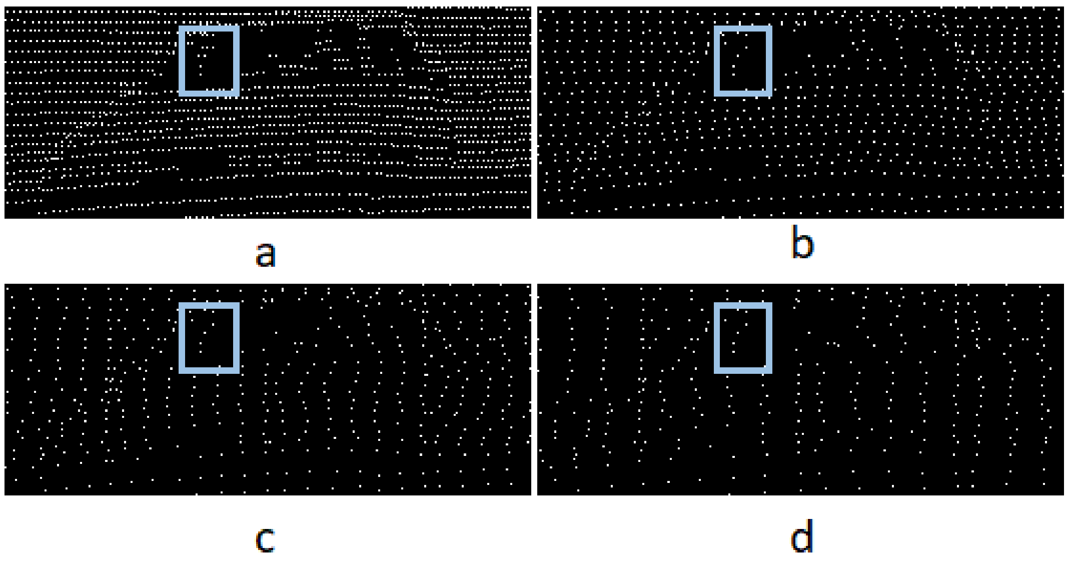

Figure 4 Ranging data projected after filtering with different distances: a) 1 pixel, b) 5 pixels, c) 10 pixels, d) 15 pixels.

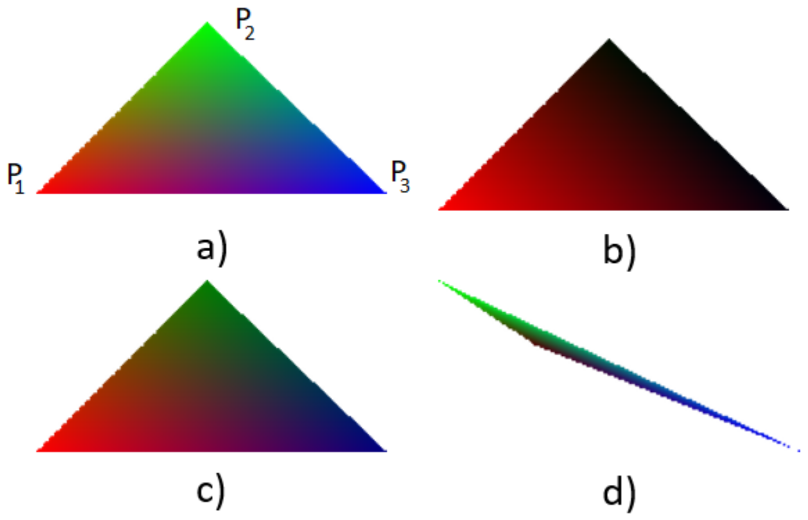

Figure 5 Samples of rendered triangles mimicking possible interpolations cases.

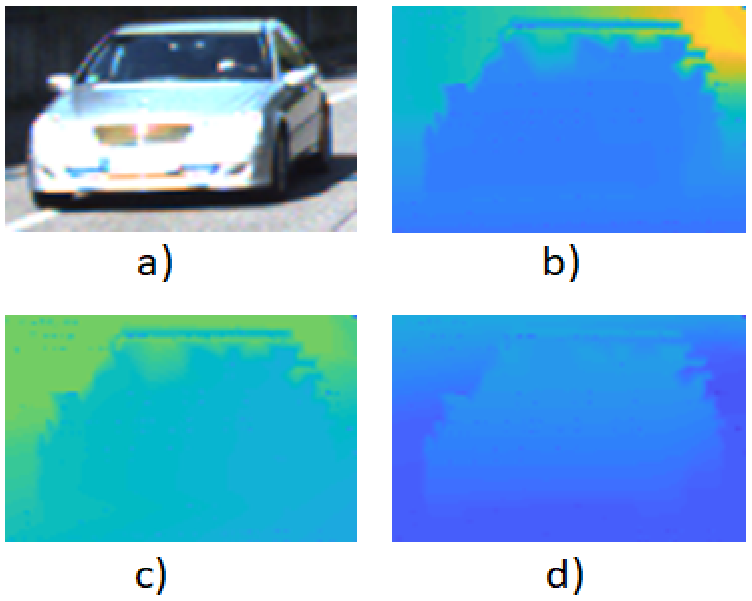

Figure 6 Up-sampled ranging data: a) the original image, b ,c and the respective up-sampled x, y and z coordinates.

Each example in Figure 5 represents a possible case.

In this example P1, P2, and P3 are the delimiting points of the triangle, each point has a different color to mark its contribution to the interpolation. In 5.a), the three points have the same value to simulate points on the same surface. The three points contribute equally to the interpolation. In the case of 5.b), P2 and P3 are 0.05 the value of P1 to simulate the case of a distant point that is connected to close ones, it can be observed that P2 and P3 do not contribute. Case 5.c) simulates the case where the three points belong to the same object with a slight difference in the value. We bring the reader's attention to the differences between this case and case In 5.a) for comparison. Case In 5.d) present a randomly shaped triangle.

5 Improving ML

Machine Learning (ML) based algorithms is becoming popular where a large set of data must be processed in a reduced time period. The three most researched tasks in ML are classification, detection, and segmentation. With the proposed representations in this paper, detection and segmentation tasks are targeted for AV. We present two possible improvements to accelerate and train machine learning algorithms.

Detection and segmentation algorithms are used to extract object shape, delimiters, and position from the provided images. In contrast to existing classification algorithms that have to differentiate between two visually different instance of the same object. We propose to use the generated representations as a replacement to camera provided images. The idea is that object in a scene can be detected using their silhouettes instead of their texture. Figure 6 shows the image of a car and the corresponding x, y and z representations. The reduced visual complexity produced by textures, illumination, and shadows allows the reduction of filters number inside layers, thus compressing the size of the model and accelerating the processing time.

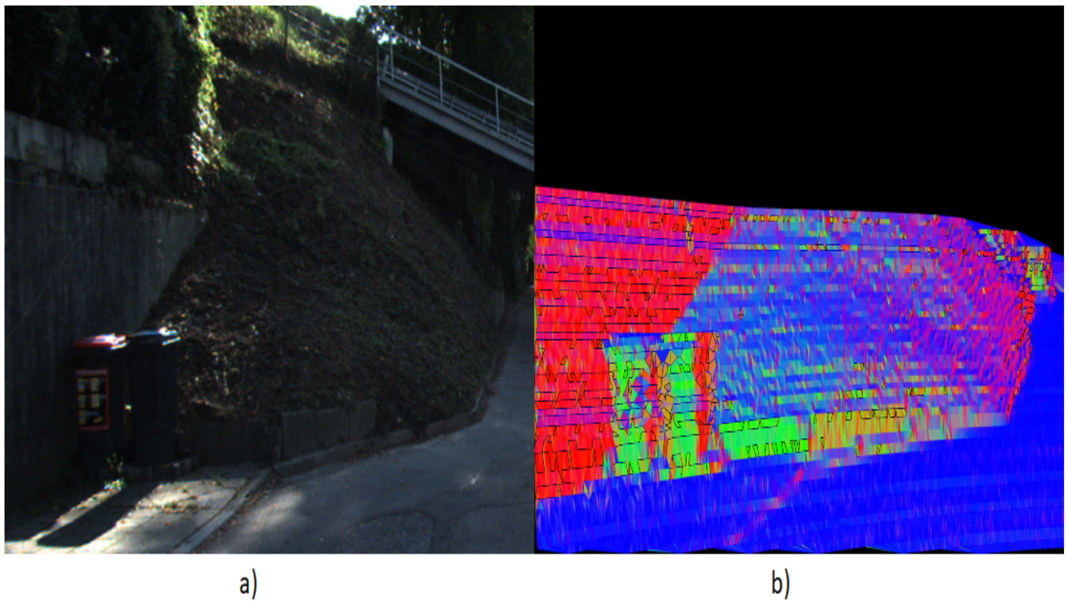

Figure 7 Generated normal map for a complex environment: a) The original image b) The generated normal map.

Furthermore, the generated normal map enables the clustering of points into surfaces and ease the differentiation between the multiple objects in a scene. In Figure 7 we present a scene that contains the normal that can be found in a complex environment.

As a second improvement, we propose to train ML algorithms using synthesized representations using modeling software. In fact, the representations do not require any realism and can be used in real-world applications.

6 Experiments

To evaluate the accuracy of the generated meshes, a ground-truth mesh is needed. However, this information is unavailable, as an alternative, the evaluation was carried using the generated representations. A 100 scene was chosen at random from different sets, and for each scene we generated the X, Y and Z representations with filtering distances in the range [5 - 25] pixels between the points, we found through experiments that a distances less than 5 pixels will create small triangles that cannot be rendered in the flowing step. The results are compared using the Root of Mean Squared Error (RMSE), Mean of Absolute Error (MAE) to the ranging data, and the number of triangles. The RMSE will give an insight on the standard deviation of interpolation error, whereas the MAE will reflect the common error value. Finally, the number of triangles in the mesh present us with the impact of rendering on the hardware. In Figure 7, the MAE and RMSE metric is plotted for the height (Z) representation, the results are in relation to pixels distance between projected points, in addition to triangles count. The result shows that with a distance of 5 pixels introduces a 1 cm absolute error and a 10 cm deviation compared to 3 cm absolute error with 25 pixels distance, it can be observed that standard deviation is only about 2 cm more for 25 pixels distance for 45% of the triangles counts. Thus, enabling the possibility to choose a balance between the desired accuracy and the dedicated processing power or processing time.

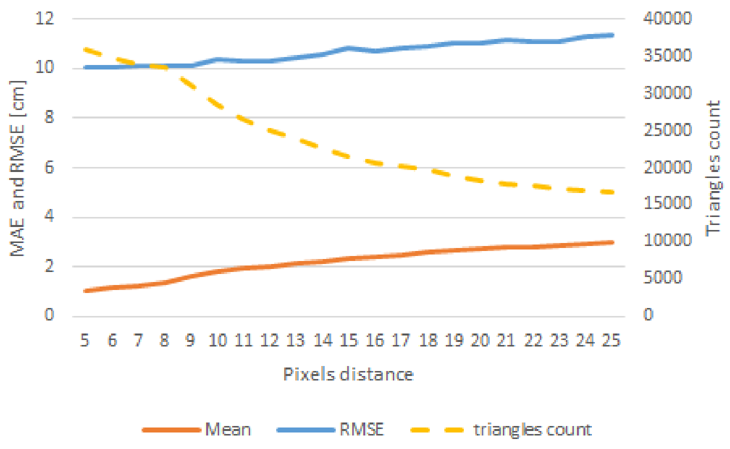

Figure 8 The different fusion architectures.

Table 1 Interpolation error MAE and RMSE in meters.

Error Metric per coord		Filtering distance in pixel			
		5	11	19	25
X	MAE	0.160	0.320	0.443	0.498
	RMSE	1.857	1.902	2.053	2.119
Y	MAE	0.045	0.088	0.153	0.168
	RMSE	0.763	0.768	0.807	0.824
Z	MAE	0.009	0.019	0.026	0.029
	RMSE	0.100	0.102	0.109	0.113

In Table 1, the interpolation error is presented, in relation to the distance by which the number of points is reduced. From the table, it can be observed that the error is strongly connected to the range. For example, values for the X coordinates range from 0 to 60 meters, where the mean absolute error increase from 16 cm to about 50 cm the more the filtering distance increase, which is not the case for the Z coordinates that range from -2m to 1m relative to LIDAR position.

7 Conclusion and Future Works

This paper presents an ongoing work on generating multiple representations from LIDAR ranging data. Our aim is to introduce a novel approach to reduce the size of ML architectures and augmenting the training set. Currently, we are implementing the generation of the representation to run in parallel on a GPU. In the next steps, we will evaluate the impact of reducing the number of points on the processing time and results in accuracy. We will also validate our hypothesis of compressing ML architecture by testing it on a variety of detection and segmentation architectures.

References

1 Henrik Andreasson, Rudolph Triebel, and Achim Lilienthal. Non-iterative vision-based interpolation of 3D laser scans. In *Autonomous Robots and Agents*, pages 83–90. Springer, 2007.

2 Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2040–2049, 2017.

3 Zhe Chen and Zijing Chen. RBNet: A Deep Neural Network for Unified Road and Road Boundary Detection. In *International Conference on Neural Information Processing*, pages 677–687. Springer, 2017.

4 James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, pages 291–298, 2006.

5 Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Learning Morphological Operators for Depth Completion. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 450–461. Springer, 2018.

6 Rex A Dwyer. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2(1-4):137–151, 1987.

7 Herbert Edelsbrunner and Nimish R Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241, 1996.

8 Leonidas J Guibas, Donald E Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(1-6):381–413, 1992.

9 Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

10 Yuhang He, Long Chen, Jianda Chen, and Ming Li. A novel way to organize 3D LiDAR point cloud as 2D depth map height map and surface normal map. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1383–1388. IEEE, 2015.

11 Zixuan Huang, Junming Fan, Shuai Yi, Xiaogang Wang, and Hongsheng Li. HMS-Net: Hierarchical Multi-scale Sparsity-invariant Network for Sparse Depth Completion. *arXiv preprint*, 2018. `arXiv:1808.08685`.

12 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

13 Jason Ku, Ali Harakeh, and Steven L Waslander. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. *arXiv preprint*, 2018. `arXiv:1802.00036`.

14 Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint*, 2017. `arXiv:1712.02294`.

15 Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera. *arXiv preprint*, 2018. `arXiv:1807.00275`.

16 Nick Schneider, Lukas Schneider, Peter Pinggera, Uwe Franke, Marc Pollefeys, and Christoph Stiller. Semantically guided depth upsampling. In *German Conference on Pattern Recognition*, pages 37–48. Springer, 2016.

17 Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement. *arXiv preprint*, 2018. `arXiv:1811.03818`.

18 Peter Su and Robert L Scot Drysdale. A comparison of sequential Delaunay triangulation algorithms. *Computational Geometry*, 7(5-6):361–385, 1997.

19 Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018.

20 Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint*, 2017. `arXiv:1711.06396`.