3D Object Detection based on a Point Cloud obtained from a Stereo Camera*

Tomás Costa[1] and João Marques[2]

***Abstract*— The basis for autonomous driving is reliable 3D Object detection. A robust detection of objects such as: cars, pedestrians, cyclists and other objects that are typically present in roads, can lead the car to avoid collisions. The issue that comes with this problem is the price of LiDAR sensors [1], making it an unviable solution for a real world application. Therefore, this work aims to find an alternative by providing a path that relies on the use of two 2D stereo images. This allows for the creation of depth maps and, subsequently, generate point clouds with better depth estimation. In this particular work, the accuracy was measured visually and we achieved very good results, since we could clearly visualize the object of the car in the generated point cloud.**

I. INTRODUCTION

Currently, the most used method for 3D Object Detection relies on LiDAR (Light Detection And Ranging) [2] scans. These sensors provide highly accurate 3D point clouds of the surroundings [3], since they can capture the whole environment around them. This leads to very precise point clouds, which can be analyzed with PointNet [5]/PointNet++ [6] algorithms. However, LiDAR sensors have two major disadvantages. First, they are very expensive, which makes it impractical in most scenarios. And second, we should not rely on a single point of failure for such a complex matter as autonomous driving, as it involves real risks of the drivers and other users of roads.

To deal with this impracticality, finding an alternative for LiDAR is necessary, and the obvious first candidate are stereo cameras, since they are more affordable, while still being able to provide depth information about the scene. The problem starts here, as previous work on car detection using images from a stereo camera provided an average precision of a mere 10% [7], which is incredibly low and could never be deployed on a real scenario. That's where the Pseudo-Lidar [4] enters, as to vastly increase this average precision (AP), we join stereo image gathering and run in through depth estimation algorithms (available in the PseudoLidar paper) [4] to create a point cloud, which we can then still evaluate with a PointNet algorithm [5].

Our goal was, then, to provide mechanisms for better evaluation and testing of Pseudo-LiDAR solutions in a real world scenario.

*This work was written in the context of Topics of Machine Learning, and the code is publicly available on GitHub [16]

[1]Tomás Costa is with the Department of Eletronics Telecomunications and Informatics, Engenharia Informática, University of Aveiro, Portugal https://github.com/TomasCostaK

[2]João Marques is with the Department of Eletronics Telecomunications and Informatics, Engenharia Informática, University of Aveiro, Portugal https://github.com/joao-p-marques

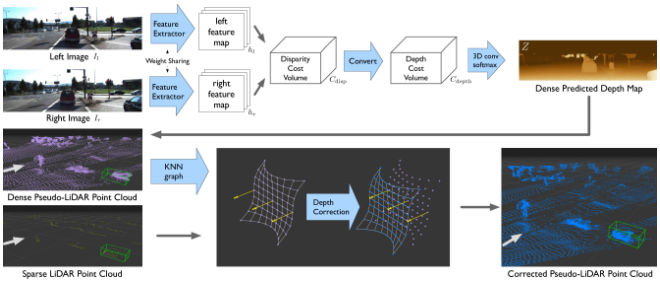

Fig. 1. The whole pipeline of improved stereo depth estimation: (top) the stereo depth network (SDN) constructs a depth cost volume from left-right images and is optimized for direct depth estimation; (bottom) the graph-based depth correction algorithm (GDC) refines the depth map by leveraging sparser LiDAR signal. The gray arrows indicates the observer's view point. We superimpose the (green) ground-truth 3D box of a car. The corrected points (blue; bottom right) are perfectly located inside the ground truth box. Taken from [8]

In this work, we chose to use a simulator close to reality in which the world had some realism and could capture the stereo images. We had the choice of going for Carla Simulator [14] or GTA V, but since one of the contributors had already done some work in the field of extracting stereo images in GTA V [10], we chose to simplify the process by going with the GTA V videogame.

We initially conducted some investigation on both PointNet and PointNet++ as it was planned in the starting objectives. But, after some discussion with the supervisors, we were instructed to focus on **depth estimation and PointCloud generation**. We still made some *State of the Art* research as it was important for us before developing anything, so we've included our research in Section II.

Our objective was to try and validate an average precision that would be practical in a real life scenario. Our methods and results are detailed below.

II. STATE OF THE ART

A. LiDAR

In the world of Autonomous driving, LiDAR detection is already a well accepted method. As it provides good position and depth estimation for common algorithms, many systems are based around this technology.

As reviewed and summarized by Ying Li *et. al.* [3], there is a lot of work developed in this area, and the results prove it is a viable path. As explained before, feeding this sort of information into PointCloud generation algorithms can provide acceptable information for an autonomous driving algorithm.

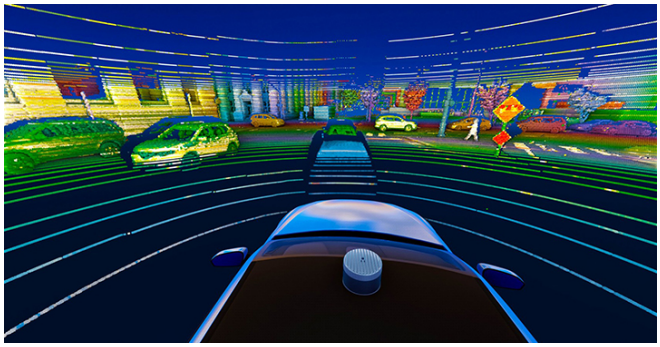

Fig. 2. Example of pointcloud generated by a LiDAR. Taken from [9]

However, as mentioned before, the problem arises with the high cost of LiDAR sensors for a real-world application. As of right now, this seems to be an unobtainable option for consumer grade solutions.

Given this restraints, the problem with different options seems to be the difficulty in obtaining a good depth estimation out of other kinds of sensors, or setups (such as regular cameras).

B. Pseudo-LiDAR

Attempting to solve the previously mentioned issue, a viable solution seems to be the use of a Pseudo-LiDAR sensor, i. e., a machine learning model that allows similar recognition mechanisms through the use of stereo cameras. [12]

Several projects have also been developed in this field, proving that this is a viable path for replacing real LiDAR sensor.

C. Depth Estimation using Pseudo-LiDAR

Given the previous, the most difficult step still seems to be the provisioning of good depth estimation without a real LiDAR sensor. As regular cameras can't provide such estimation, some work has been developed, in particular, by Yurong You *et. al.* [12], in order to provide said depth estimation thanks to the conversion of the images captured by the camera into Depth maps, and later into Point Clouds. In particular, this algorithm relies on the use of 2 images, instead of a single one, in order to improve the generated Depth map, and provide a more accurate depth estimation using the Point Cloud.

D. PointNet & PointNet++

As explained above, we've made some research on PointNet [5] and PointNet++ [6] for object detection after the PointCloud has been created. Both these algorithms accomplish our goals, but after discussing with the supervisors, we chose to go with PointNet++, since they had tested PointNet and didn't come up with very good results.

Both algorithms consume a raw point cloud (set of points) without rendering, then uses a neural network to help classify, or segment parts.

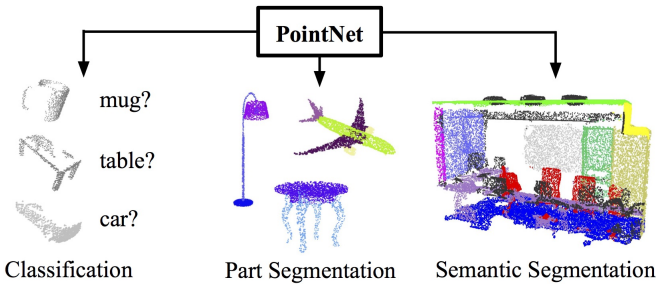

Fig. 3. Example of pointnet classification. Taken from [5]

However, PointNet++ introduces a hierarchical neural network that applies PointNet recursively on a nested partitioning input point set. This means that it is able to learn local features with increasing scales by exploitation of metric space distances. The authors claim that is able to learn deep point set features efficiently and robustly, and can achieve better than state-of-the-art results on 3D point cloud benchmarks.

III. APPROACH

A. Stereo Image Generation

The initial step to the whole 3D Objection Detection is based off getting good stereo image captures, so we can test if the LiDAR substitute is valid. To achieve this we used the supervisor's *ScriptHook altered mod* [10]. His mod was limited in what we wanted to do, but offered the tools needed to achieve it. So, as described in the project proposal, we initially had to follow this flow:

1) Position the camera in the character
2) Have the camera face the car
3) Push the camera away from the car
4) Rotate to the `right` and take a picture
5) Re-position camera in the `center`
6) Rotate to the `left` and take a picture

The images we generated followed the same standards of the figure below. III-A.

Fig. 4. Images taken with initially proposed camera setup. Left and Right camera respectively. Screenshots from in-game footage of GTA V

We tried to run the depth estimation on these images and were getting depth maps that were impossible to visualize or comprehend, so we decided to further investigate the KITTI Dataset [11] and decided to change the cameras setup, so we would fit the standards defined in the dataset. After these changes, we started to get a more frontal view of the car, since the cameras were less spaced out and the angles were cut to 0º. Below is an example of the images we captured with this new method.

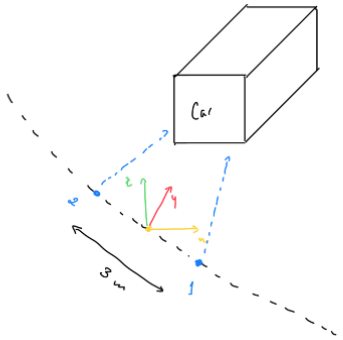

Fig. 5. Initial setup of cameras and orientation for the screenshots of the stereo cameras

Fig. 6. Images from GTA V to simulate KITTI Dataset standard. Left and Right camera respectively. Screenshots from in-game footage of GTA V

Fig. 7. Final setup of cameras and orientation for the screenshots of the stereo cameras (Standardized to KITTI)

B. Depth Estimation

Given the previously mentioned *State of the art* work, the use of stereo camera images would allow for a more accurate depth estimation on the captured scene.

With this in mind, we approached the problem by tweaking the previously available GTA mod code to capture the images according to the KITTI dataset standard, which allows for a more accurate generation of the depth map, with the input from the stereo images, relying on the use of previously developed and tested algorithms[4] to generate said depth map.

Depth estimation is one of the major problems with systems not using LiDAR, so we developed our solution around tests involving realistic situations with complex backgrounds.

C. 3D Object Detection

Since the problem's goals were changed midway, instead of running the pointclouds through an algorithm like PointNet [5], we discussed with the supervisors and decided to visually classify the difference in pointclouds. Even though this method wasn't very reliable early on (since the pointclouds were difficult to visualize), once the pointclouds got clearer, it proved to be a good method of evaluating them, as we can easily compare in this figure III-C below:

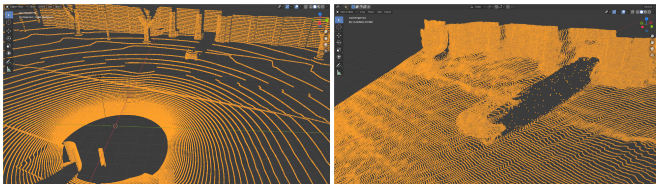

Fig. 8. Left: Pointcloud generated from in-game LiDAR (code provided in [10]); Right: PointCloud generated using our Pseudo-LiDAR approach

IV. EXPERIMENTS

A. Image pre-processing

After capturing the images, as described above, we needed to process them, so we would be able to fit them in the standards of the KITTI Dataset and our camera calibration files would be correctly configured. So, we needed to assure two things:

- Convert the BMP images to PNG.
- Correctly size the images to fit the 1242x375 resolution

Without this data pre-processing, the depth map estimation algorithm wouldnt work correctly. To conver the images from BMP we used simple online converters. But, for the resizing we needed to maintain consistency, so together with the supervisors, we wrote a script that would allows us to resize all images in the same area. This script is present in the `repository` [13].

After this processing being applied, we need to place the images in the file tree according to the KITTI dataset standards, and follow the workflow defined by the creators of the Pseudo Lidar++ algorithm [4], which is also documented in our code repository [15].

B. Results

Given the established base points, we obtained different results, according to the image we fed into the model. On the various interactions we made, we tried to increasingly improve our results.

On the first iterations, we used the images represented in 9.

Fig. 9. First version of stereo GTA capture

As we can see, these no longer have the same problem as mentioned before, and are correctly spaced between each

other. However, after generating the point cloud, we found another issue (IV-B).

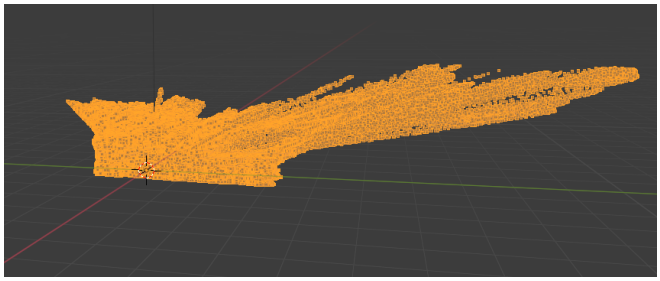

Fig. 10. Point cloud generated by first image set

As we can see, because of the small distance to the vehicle, the point cloud was not able to reflect the environment, and not even detect the vehicle.

To address this issue, we changed the cameras distance to the car (from 8 to 15 meters) and spacing in between cameras, to follow the KITTI standard of a 0.54m spacing. The images shown in 11 reflect those changes.

Fig. 11. Second version of stereo GTA capture

As seen above, the cameras are now correctly spaced, and the result is the point cloud seen in IV-B.

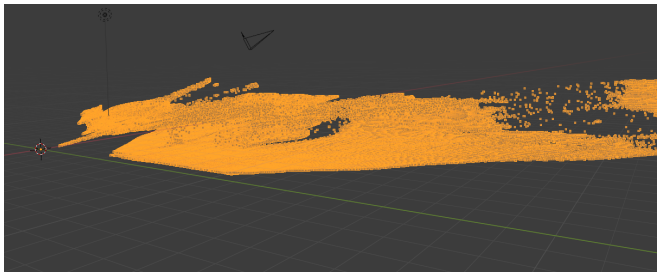

Fig. 12. Point cloud generated by second image set

This point cloud allows a bigger scene visualization, and represents a good improvement, as we can easily identify objects and obstacles.

However, in the generated point cloud, we can see a group of collision points on the top portion, which makes it harder to view the the desired content. With further research and testing, we attribute this error to a possible limitation of the algorithm when dealing with images generated in a video game with specific backgrounds (ocean, skies or clouds).

Indeed, we were able to improve results when using pictures with a solid background, such as seen in 13, and with the respective point cloud (14 and IV-B).

With further discussion with the supervisor, we also established the possibility to still use the point clouds with the top collision "noise", though additional processing would be required to "clean" the output.

Fig. 13. Third version of stereo GTA capture

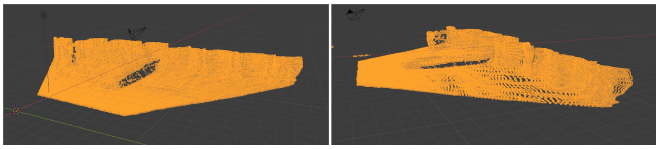

Fig. 14. Point cloud generated by third image set

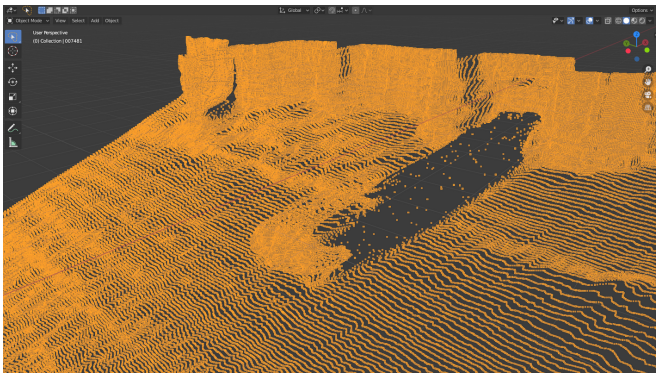

Fig. 15. Better visualization of the third image set's point cloud

C. Automation of detection

Given the large number of different analysis we made, as well as the fact that we relied on external servers and ephemeral services, like Google Colab [1], we felt the need to automate the process as much as possible.

The workflow consisted, mainly, on placing the captured images inside the training dataset structure of KITTI, and then running several scripts with pretrained models in order to generate depth maps, point clouds, and convert them to a usable format.

Given those needs, and with help from the supervisors, we developed a set of scripts to help with this problem. The most advanced of them[18] takes the left and right images, placed inside individual folders, themselves placed in a *GTADataset* folder, and automatically places them inside the correct paths. Besides this, the only thing necessary is initializing the environment[17], by cloning the upstream repository[12], installing all the dependencies, and uploading the pretrained model and configuration files.

[1] https://colab.research.google.com/

V. RESULTS ANALYSIS

Skybox	Time of Day	>1 Car	Visible[2]	Precision[3]
YES	DAY	NO	8/8	75%
YES	NIGHT	NO	0/2	0%
NO	DAY	NO	5/5	100%
NO	DAY	YES	5/5	100%

[1]All cars were visible to both authors but, visibly classifying a car is not very reliable as it depends on the human eye.

[2]Even though all cars are visible in the pictures we tested, the ones that include a skybox have a lot of noise in the image, which made the average precision of the image decrease.

For this analysis we took several parameters into consideration, such as Skybox (did the image have sky showing in it or not), whether it was night or day and if it had only one car or multiple. After creating this testing dataset, we ran the script according to the automation mentioned above and got some pretty good results when it comes to visibly identifying a car, but the images with skybox in it, lacked precision.

VI. DISCUSSION

In retrospective, we can see a big improvement in using real, yet environmentally controlled, images from GTA V, and adapting them to autonomous driving research.

We can see the improvements clear in the transitions between the different image sets used.

Also, in comparison to the point clouds generated using the KITTI Velodyne dataset, we were capable of adjusting our image capturing mechanisms to match the expected performance and results, as seen if III-C.

Finally, we have established a limitation in this process, related to the bad accuracy in the case of skybox backgrounds on the images.

Overall, we believe to have obtained good results, achieving the proposed solution to the initial problem, and providing a good basis for future Pseudo LiDAR based systems, with an accurate Point Cloud generation from GTA V environments.

VII. CONCLUSIONS

As stated above, we come to the conclusion that this algorithm is valuable and can work on a real life scenario as obtained from the KITTI Dataset, but, due to the limitations on the GTA Skybox we cannot accurately generate the pointcloud on every location (ones that have images of the sky) and is therefore impractical to detect a vehicle on a continuous flow as of right now.

In the future, we would like to run the pointclouds throught a PointNet classification algorithm to better understand the accuracy of our pointclouds and determine what parameters we have to tweak. This work is an important piece of Diogo's thesis and therefore these results are a good guideline on how he should advance with his work.

This work and report were developed and written by both authors equally.

ACKNOWLEDGMENTS

We would like to thank the people behind the supervision of the project, for all the help provided in this project.

Supervisors:

- Diogo Guedes - diogoguedes@ua.pt
- Prof. Pétia Georgieva - petia@ua.pt
- Miguel Drummond - mvd@ua.pt

REFERENCES

[1] https://www.technologyreview.com/2017/11/28/147443/lidar-just-got-way-better-but-its-still-too-expensive-for-your-car/
[2] https://www.neonscience.org/lidar-basics
[3] https://arxiv.org/pdf/2005.09830v1.pdf
[4] https://arxiv.org/pdf/1812.07179.pdf
[5] https://arxiv.org/pdf/1612.00593.pdf
[6] https://arxiv.org/pdf/1706.02413.pdf
[7] http://openaccess.thecvf.com/content_cvpr_2018/papers/Xu_Multi-Level_Fusion_Based_CVPR_2018_paper.pdf
[8] https://deeplearn.org/arxiv/80511/pseudo-lidar++:-accurate-depth-for-3d-object-detection-in-autonomous-driving
[9] https://velodynelidar.com/products/alpha-prime/
[10] https://github.com/Diogo525/PC-datasets-generation-gtav
[11] http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d
[12] https://deeplearn.org/arxiv/80511/pseudo-lidar++:-accurate-depth-for-3d-object-detection-in-autonomous-driving
[13] https://github.com/TomasCostaK/autonomous_driving/blob/master/Dataprocessingscripts/GTA_data_samples_processing/GtaView.py
[14] https://github.com/carla-simulator/carla
[15] https://github.com/TomasCostaK/autonomous_driving/blob/master/KITTI_Pseudo_Lidar.ipynb
[16] https://github.com/TomasCostaK/autonomous_driving/
[17] https://github.com/TomasCostaK/autonomous_driving/blob/master/automation_scripts/initialize.py
[18] https://github.com/TomasCostaK/autonomous_driving/blob/master/automation_scripts/mass_generate.py