Bachelor Project Report

ML: Stereo Camera Odometry

Taha Zakariya

Communication Systems

BA6

SCIPER : 288526

8 credit course

Professor: Alexandre Alahi

Supervisor: George Adaimi

June 22, 2021

Abstract

Accurate localization of a vehicle is a fundamental challenge and one of the most important tasks of mobile robots. For autonomous navigation, motion tracking, and obstacle detection and avoidance, a robot must maintain knowledge of its position over time. Vision-based odometry is a robust technique utilized for this purpose. It allows a vehicle to localize itself robustly by using only a stream of images captured by a camera attached to the vehicle.

Stereo visual odometry is an important technique for estimating the parameters of a camera's sensor relative to a scene. It relies on the principle that pixels in images taken by a pair of cameras from two different viewpoints are usually visible in both images, but displaced. This displacement can be used to compute the image's depth map, which can be used to then compute the position of the rover relatively to its previous positions.
To estimate the stereo projection, the passive stereo image is first warped to fix its perspective so that all lines in each eye point along the same direction. For guided stereo reconstruction, this is commonly accomplished by estimating a correspondence function from images of known disparity.

Given the insufficient amount of time, I put my focus on this last task, in other words, I focused on getting the depth map from stereo camera images.

Contents

List of Figures

Introduction

EPFL Xplore is a student association from EPFL whose aim is to develop rovers to participate in Mars Society competitions in Poland and in the United States.
The project is divided in 3 main branches : Communication, Engineering and Finance. Each of these main fields rely on the passion of involved students from various sections of the school.

We are a team of over 40 students working on this project and I have been tasked, in the context of a Bachelor semester project, to develop part of the navigation software, particularly participate in the task of odometry leveraging the power of machine learning and neural networks.

Figure 1: Xplore logo

Motivation

Knowing the accurate position (not using GPS) of the rover is one of the most important and challenging aspect of Xplore Navigation Subsystem. This step is crucial for our path planner, the mapping of the environment and our motor controller. So far we use a IMU and encoder, but those solutions cannot detect slip and output wrong positions estimations. To solve this we have a semester project on 3D Lidar odometry.
Unfortunately 3D LiDar are extremely expensive and it is a known fact that stereo cameras can with some degree replace LiDars (less precise and more prone to perturbations). The goal of this project is to use the stream of stereo cameras image pair to estimate the position of the rover.

Goal

Initially, given a dataset with the stereo cameras stream and the position of each frame, I had to design a model to estimate the position of the rover given a sequence of 360 stereo image pairs.
However I focused on getting the depth map from stereo images which can lead to getting an estimation of the position of the rover.

1 Related works

1.1 AnyNet

AnyNet [1] is an architecture that leverages the power of neural networks and the efficiency of other algorithms to implement theirs which combines time efficiency and accuracy. AnyNet is used to compute disparity maps which can be transformed with some simple formulas to depth maps.

Depth estimation should be accurate for mapping the environment, and real-time, typically for obstacle avoidance, which is an important task for a rover. Current state-of-the-art algorithms can either generate accurate but slow, or fast but high-error mappings, and typically have far too many parameters for low-power/memory devices.

Motivated by this lack of efficiency, AnyNet proposes a new approach, which consists of computing the disparity map in 3 to 4 stages, with ascending resolution, accuracy and time until inference, which gives a good trade off between these metrics: as a larger computational budget is made available, the prediction is refined and becomes more accurate.

Specificities

- Library used: Pytorch
- Camera type: Stereo
- Benchmark datasets: Sceneflow and Kitti

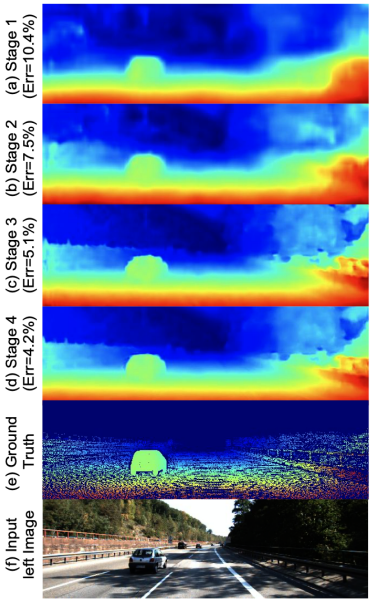

Figure 2: Disparity prediction from 4 stages of AnyNet on KITTI-2015 (On the paper)

1.2 Deeper depth prediction with FCRN (Fully Convolutional Residual Networks)

Deeper depth prediction with FCRN [2] is an algorithm that leverages the power of fully convolutional neural networks to model the ambiguous mapping between monocular images and depth maps. They use the reverse Huber loss that is particularly suited for the task at hand and driven by the value distributions commonly present in depth maps. The model is composed of a single architecture that is trained end-to-end and does not rely on post-processing techniques.

Their main idea is to merge Mask R-CNN with FCRN. The modified FCRN, which can also be regarded as an improvement through Mask R-CNN, is designed on the basis of attention mechanism and optimized on the basis of transfer learning.

Specificities

- Library used: Tensorflow
- Camera type: No paticularity (single RGB image)
- Benchmark datasets: NYU Depth v2 and Make3D

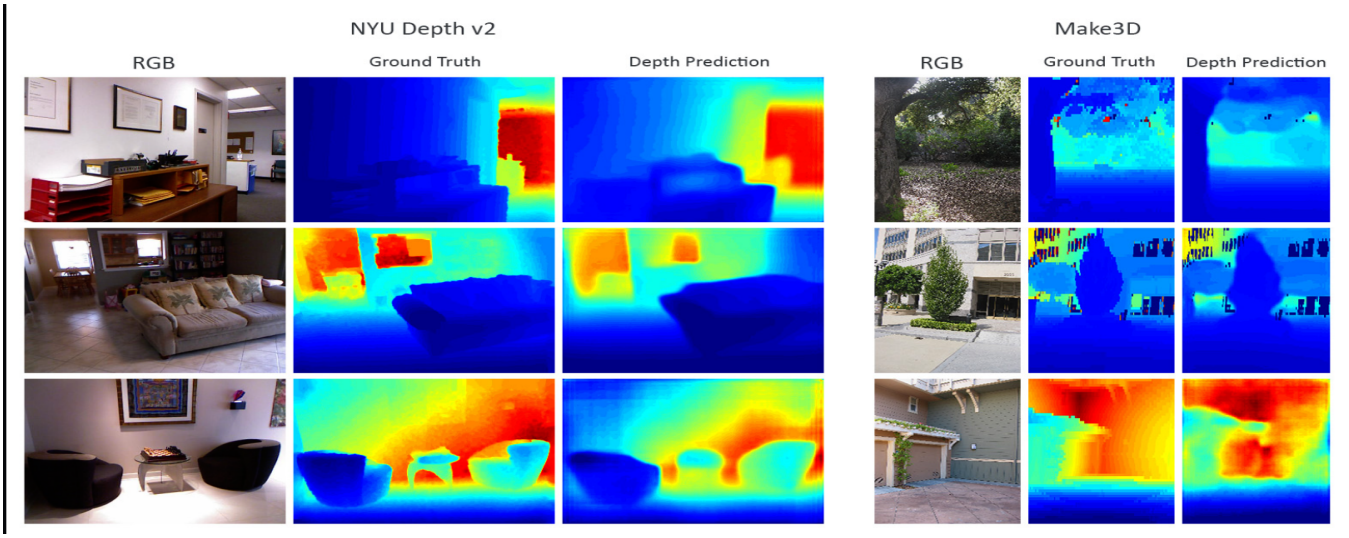

Figure 3: Deeper depth prediction with FCRN qualitative results

1.3 Real-Time Panoramic Depth Maps from Omni-directional Stereo Images for 6 DoF Videos in Virtual Reality

This paper [3] presents an approach for 6 DoF panoramic videos from omni-directional stereo (ODS) images using convolutional neural networks (CNNs). More specifically, they use CNNs to generate panoramic depth maps from ODS images in real-time.

They approach the problem from a learning-based perspective: given an ODS image (represented as a pair of equirectangular images) as input, predict a panoramic depth map to allow for 6 DoF through depth based warping. They introduce a border weighted loss function as well as new error metrics specifically tailored for panoramic images.

Specificities

- Library used: Tensorflow
- Camera type: Stereo and large field view
- Dataset: Generated their own dataset

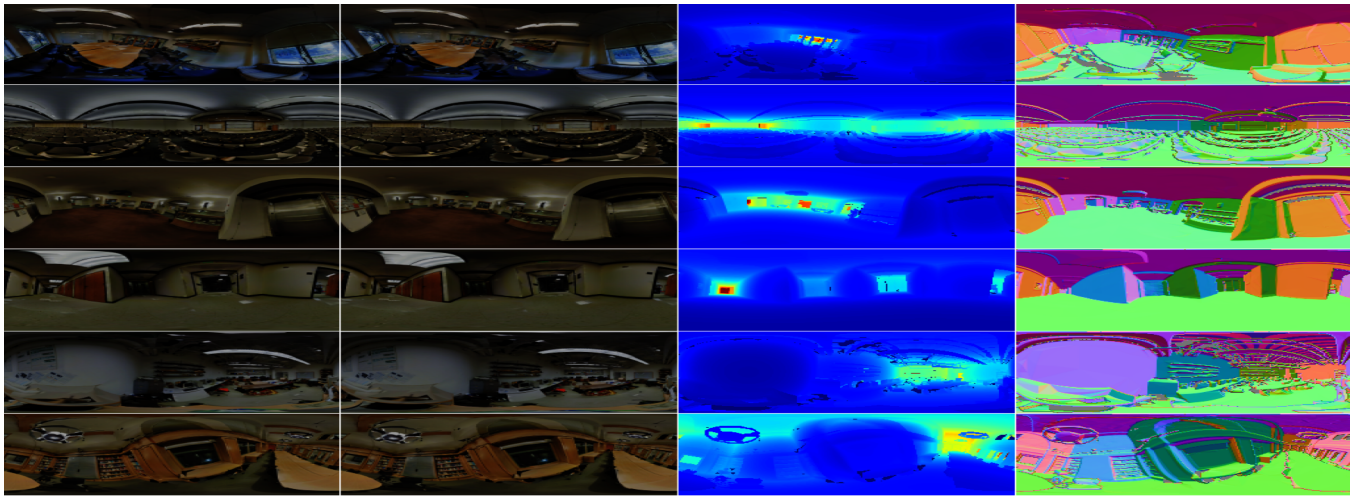

Figure 4: Left: ODS training images, Right: Left depth and normal maps

1.4 FastDepth: Fast Monocular Depth Estimation on Embedded Systems (MIT)

FastDepth [4] addresses the problem of current state-of-the-art algorithms complexity and slowness. In this paper, they try to solve this issue of fast depth estimation on embedded systems. They propose an efficient and lightweight encoder-decoder network architecture and apply network pruning to further reduce computational complexity and latency.

Specificities

- Library used: Pytorch
- Camera type: No particularity (Monocular)
- Dataset: NYU Depth v2

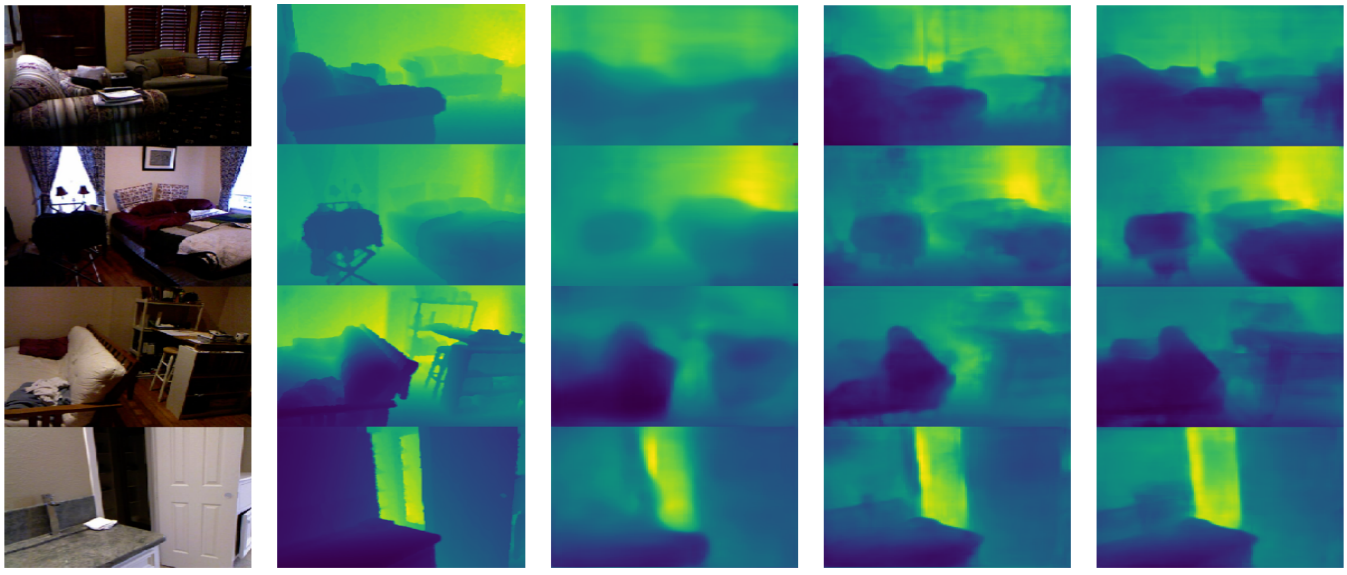

Figure 5: Depth estimation on the NYU Depth v2 dataset. (a) input; (b) ground truth; (c) model, without skip connections, unpruned; (d) model, with skip connections, unpruned; (e) model, with skip connections, pruned

2 The previous work chosen

2.1 Why AnyNet

In addition of looking for papers which give good results, we had some preferences and requirements that we wanted to fulfill.
Our first requirement was to get a paper that gave access to their code. That would help us beforehand test it to see if their results match what they suggest, eventually improve it and adapt it to our needs. Thus we eliminated many good papers which didn't provide that information.
Being beginner friendly, easier to learn and lighter to work with, and hence, is relatively better for passion projects and building rapid prototypes, we also preferred a code that uses pytorch over another that uses tensorflow.
Benchmark datasets for depth map computing are: KITTI, Sceneflow and NYU DEPTH v2, especially KITTI which is very known to evaluate many image-related tasks (odometry, depth, disparity ...). Thus, we would chose a paper that evaluates its algorithm with KITTI over another algorithm that uses their own dataset or other datasets.

It's true that AnyNet outputs disparity maps rather than depth maps. But as claimed in their paper, our researches led to many ways to transform disparity maps to depth maps once we have all the metrics we need from the rover and the cameras, leveraging this algorithm [5] for example.

2.2 AnyNet's algorithm

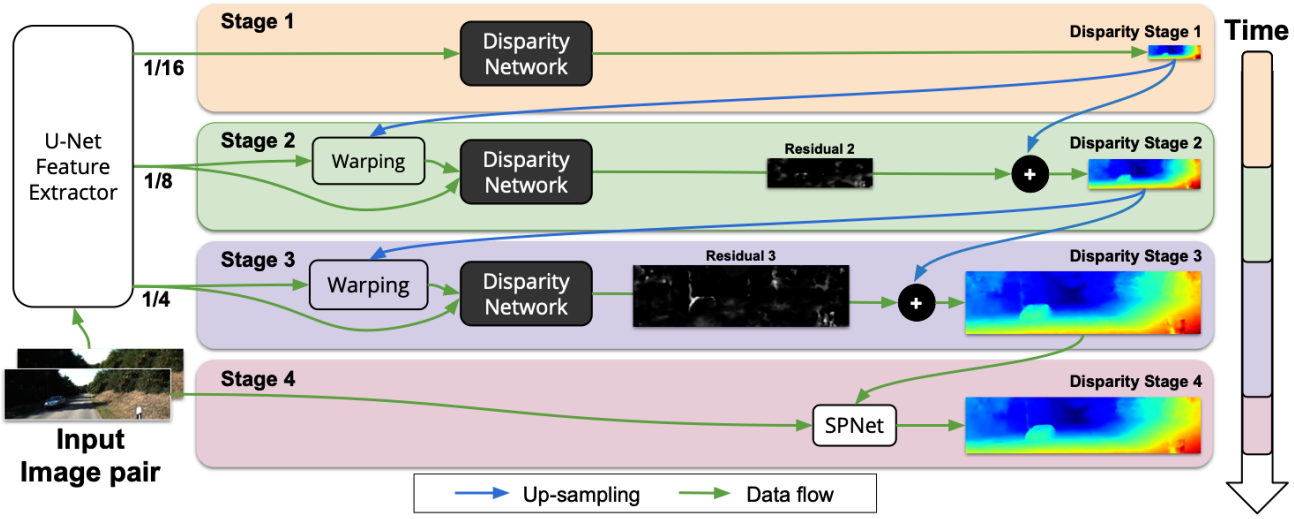

Figure 6: Network structure of AnyNet

In a few words, AnyNet [1] computes the depth map in 4 stages, each with ascending resolution, and leveraging U-Net Feature extractor. In the first stage, only the lowest-scale features (1/16) are computed and passed through a disparity network (fig.6) to produce low-resolution disparity map, which takes only few milliseconds. For the rest of the stages, instead of computing a full disparity map at this higher resolution, the next stage will simply correct the already-computed disparity map from the previous one using a residual map, which contains small corrections that specify how much the disparity map should be increased or decreased for each pixel. Finally, the 4^{th} isn't necessary, it just gives way better results for by sharpening the disparity map for little extra cost.

2.3 Experimental results (Checking the papers' results)

To recall, the result they claim they achieved is the one in (fig.2). To check this result I used this stereo image from KITTI:

Figure 7: Stereo pair (left)

Figure 8: Stereo pair (right)

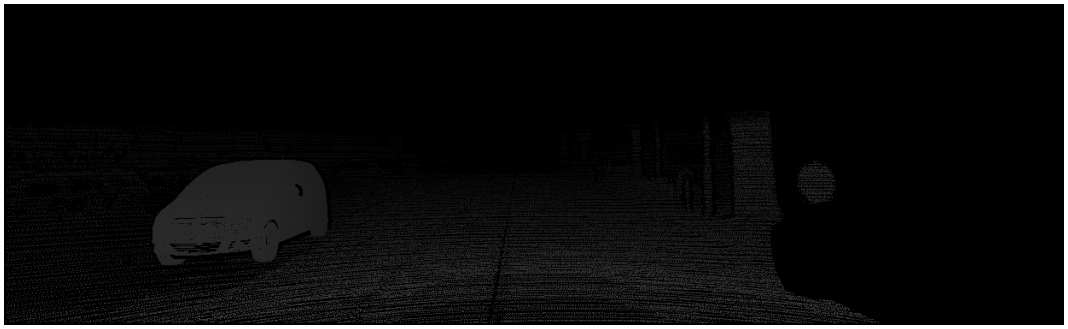

Figure 9: Disparity map ground truth

The first 3 stages disparity maps found by running AnyNet on this stereo pair:

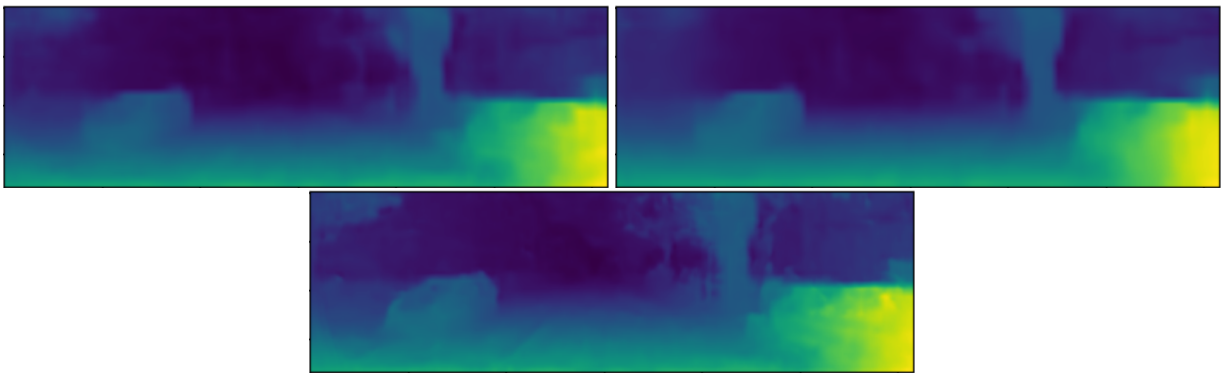

Figure 10: Disparity experimental predictions

The results seem actually pretty good, even in stage 3 without the refinement we could clearly see the shapes, so we were quite optimistic about what was coming.

Unfortunately, when we evaluated the accuracy of the model over the split file provided, the results were :

	Stage 1	Stage 2	Stage 3	Stage 4
KITTI 2015	14%	15.8%	14,5%	13.5%
KITTI 2012	14.7%	20,2%	10%	9%

Figure 11: Three-Pixel error (%) of AnyNet on KITTI-2012 and KITTI-2015 datasets

While the results given in the paper are :

Dataset	Stage 1 28.9ms	Stage 2 48.8ms	Stage 3 87.9ms	Stage 4 97.3ms
KITTI2012	15.1 ± 1.1	9.9 ± 0.6	6.7 ± 0.4	6.1 ± 0.3
KITTI2015	14.0 ± 0.7	9.7 ± 0.7	6.8 ± 0.6	6.2 ± 0.6

Figure 12: Three-Pixel error (%) of AnyNet on KITTI-2012 and KITTI-2015 datasets (from the paper)

It's true that in the initial work, all results are averaged over five randomized 80/20 train/validation splits, so it's expected that it wouldn't exactly be the same outcome. However, the results seem contradictory: the error that increases at the 2^{nd} stage each time doesn't make much of sense: as we can see in fig.10 the second disparity map is sharper than the first one, and AnyNet is supposed to give a better disparity map at each step. We noticed that an issue has been raised about this subject from someone who has similar results to ours after evaluating the model. However, no answer has been provided.

Nevertheless, as the results seemed pretty good visually on KITTI's samples, we decided to move forward and still count on AnyNet to compute our depth map.

3 Dataset

During nearly the whole semester, I only had some indoor quickly recorded dataset to work on in order to test the model. Here's a sample from the first dataset I was first provided with its ground truth depth map (with the LiDar):

Figure 13: Upper camera /Lower camera / Depth map from the first dataset

Then, on the 13th of June, I was provided the new dataset recorded lately. Here's a sample:

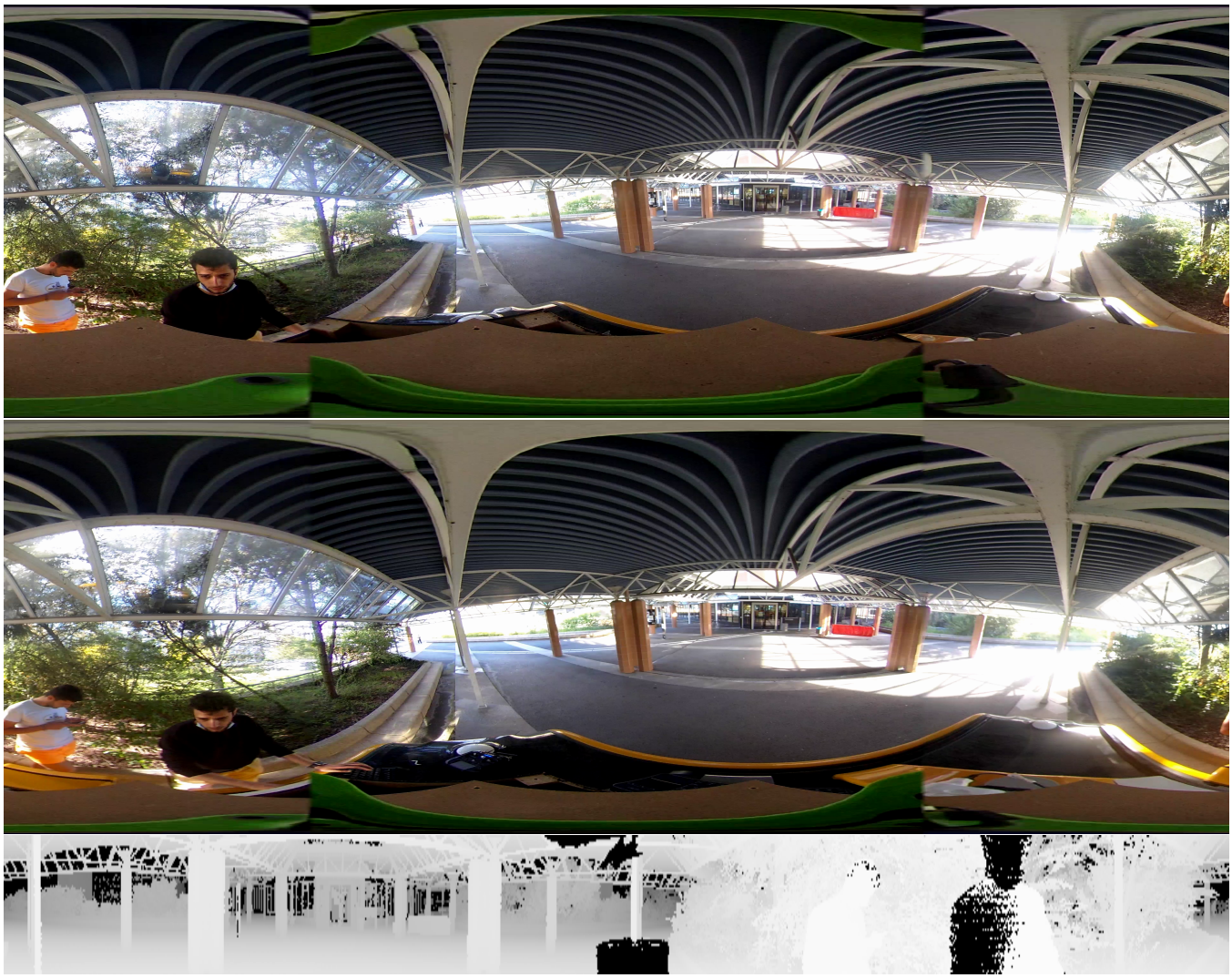

Figure 14: Upper camera /Lower camera / Depth map from the second dataset

4 Depth Estimation

4.1 First dataset

Notice: I specify that for all the tests, we chose to give the depth map as the ground truth to the model as we thought it should be able to adapt given both basically do the same job when we consider a stereo image.

At first, given that the dataset that you can see in fig.13. It was bad, the ground truth was not clear enough to make the necessary shifts and cropping on the stereo images to get the right proportion. In other words, the proportion of the stereo image that matches the Lidar's depth map was nearly impossible to determine as the Lidar captures a very tiny proportion of the landscape vertically, in addition to the fact that the images and the lidar don't start from the same point (images should be shifted by hundreds of pixels to match Lidar's referential).

An important point to know is the fact that AnyNet model doesn't work if the depth map isn't the same size as the input stereo, so given that we couldn't determine the proportion of the depth that matches the stereo, it was nearly impossible to work with properly and determine from where a potential error in the estimation would come from.

To bypass this problem to test out the model, we just resized the depth map to the same size as stereo images, and tried to overfit the model on 1 image only.

One should also know that depth maps shouldn't be given as RGB but should only have 1 layer. Fortunately our depth map is grey-scale so we took 1 layer out to work with as they're all the same.

After trying many parameters and manipulations, here are the results of what we've tested:

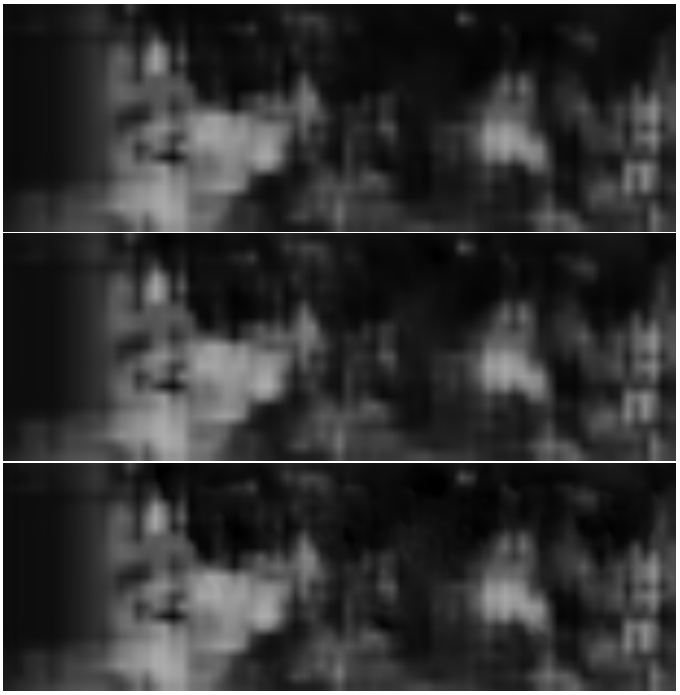

Figure 15: Disparity maps of the stereo image finetuned with KITTI (3 stages)

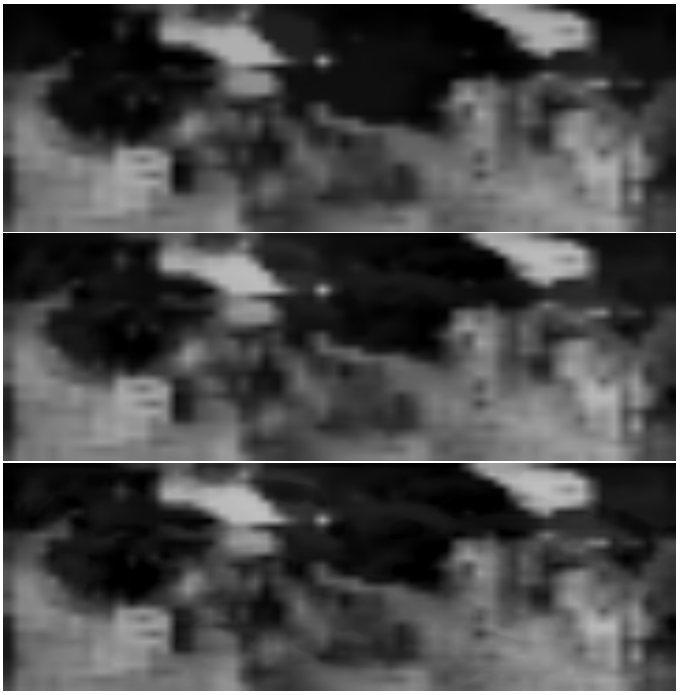

Figure 16: Disparity maps of the stereo image finetuned with the image itself (aim to overfit, 3 stages)

As you can see, the results are quite bad, but at this point we didn't necessarily think that the problem would or wouldn't come from the model not matching our dataset, given that the supposedly ground truth itself was bad, so we asked for a new and cleaner dataset.

4.2 Code modification

In the meantime, while we were waiting for a new dataset to be recorded, we did some analysis on our data and the one provided by KITTI because the results were so far from the supposedly ground truth (we thought that it should be a minimum similar given that we overfitted the model on 1 stereo image only

and tested it on ther same image).

Thus, we noticed that the values of the pixels from KITTI's disparity map go from 0 to 16406 while the pixels' values of the depth map we're provided only go from 0 to 255 while it should be a able to reach higher values given the nature of a depth map.

Considering that we weren't provided the necessary metrics to get these high initial values nor to get a disparity map, we chose to change the dataloader in a way that the values of the depth map are of the same magnitude as KITTI's disparity when using the model (otherwise the model receives values between 0 and 1 which causes bad outputs).

4.3 Second dataset

On 13th of June, we received the new dataset, and I worked with the following stereo pair:

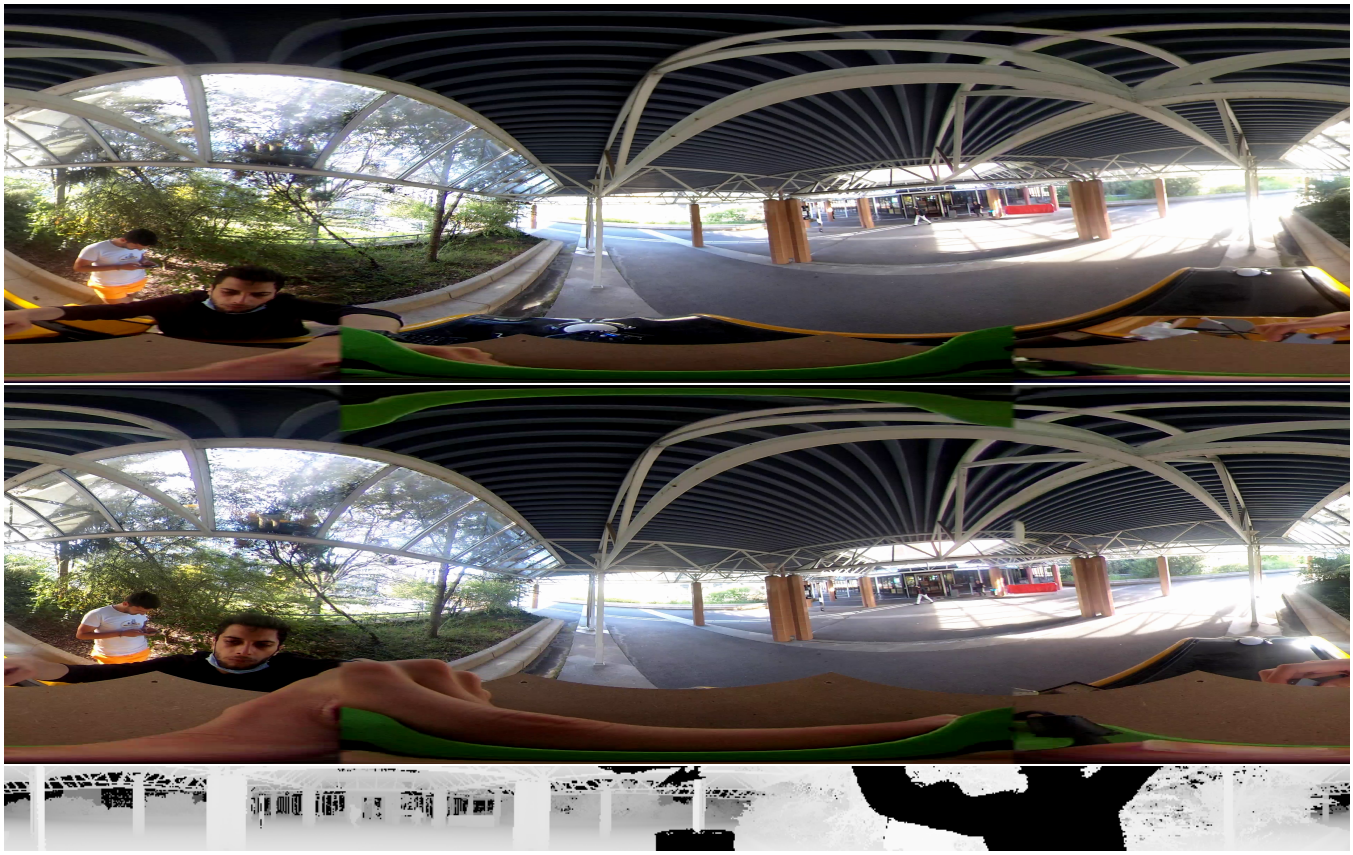

Figure 17: Upper camera/ Lower camera/ Depth map of the new dataset

Now that the depth map is quite clear, we could more or less match a proportion of the stereo with the depth map. Here's the result of the preprocessed pair:

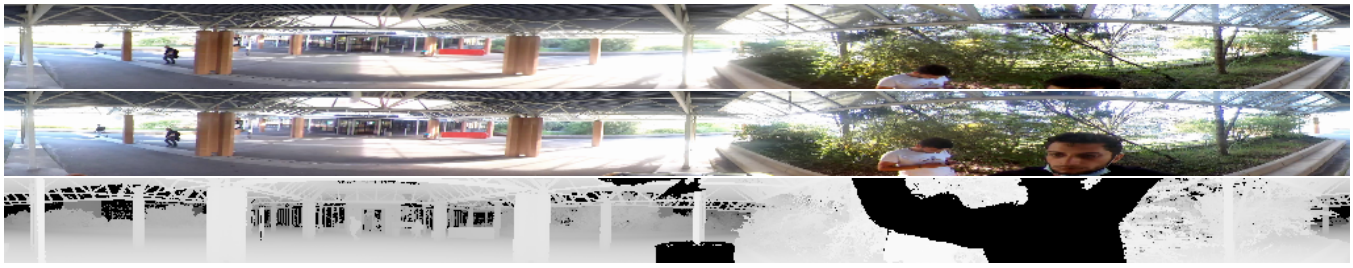

Figure 18: Processed Upper and Lower camera/ Depth map

Given that the cameras and Lidar don't have the same reference (they capture the image at different angles so they're not centered in the same way), we shifted the image in a way that could overcome this issue. Also, the Lidar only captures a small amount of the landscape vertically, so we cropped parts of the image that were irrelevant and didn't match with the actual pair, only taking into account what's farthest points. Finally we resized because the Lidar was 1024x64 while the pair is initially 4k.

We can see that the left part matches quite well the depth map: actually, it's literally impossible to match all the picture with Lidar's depth, so we focused on matching as good as possible with a high focus on the farther points and ignoring the near ones as they're supposed to be impossible to match given that these points are different even in the original unprocessed pair.

We tried to overfit our model with the new pair, changing the parameters in a way that could output a clearer depth map, and we managed to get some improvements:

Figure 19: Model overfit on pair with default parameters

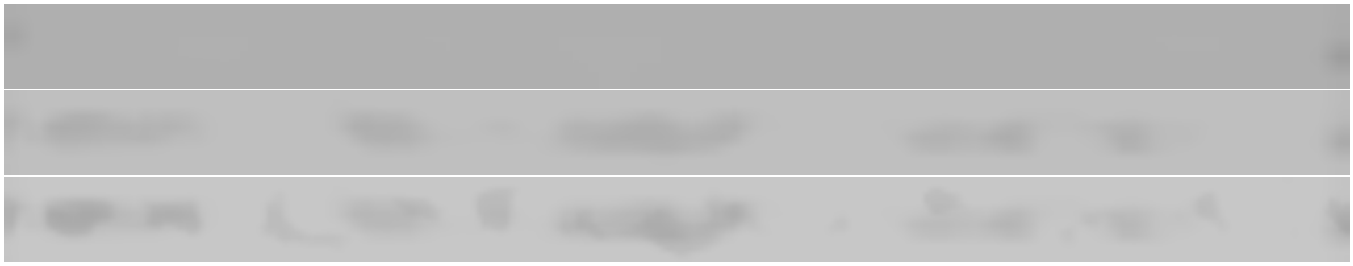

Figure 20: Model overfit on pair with 1000 epochs, maxdisp 255 and learning rate 10^{-3}

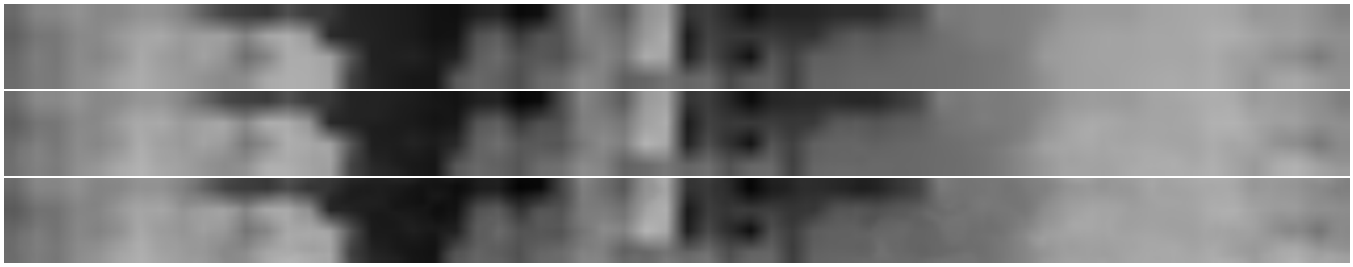

Figure 21: Model overfit on pair with 1000 epochs, maxdisp 192 and doubling maxdisp values

We tried to test the output on KITTI's finetuning, but it's coded in a way that the output is 512x256, which clearly gives worse results given the shape of out depth map.

5 Discussion

Unfortunately, the results aren't that good with respect to what we expected. It could be bad due to many issues:

- Our pair is recorded using upper and lower cameras, where all models that compute the depth maps that we found, including AnyNet, use images that are recorded using left and right cameras.
- During the time we had the dataset, we noticed that when we want to pre-process the pair to match the depth map, we can't implement a simple algorithm to do so: for example in the above pair, we had to shift the pair by 1950 pixels, while in another pair we have to shift by 2400 pixels and so on. Thus there could possibly be a problem in the way we get the data.
- As said in the beginning, we considered that the depth map would work due to the direct relation between the depth map and disparity map. However, it could possibly change everything if we take the time to get the necessary metrics to first compute the disparity map and then use it for the model.

6 Conclusion

This semester project has been a very enriching and rewarding experience. It gave me the opportunity to discover a very important and interesting field, which is that of machine learning, a field that is becoming increasingly popular in the world. I have learned a lot from this project and from my supervisor, and I think it was an essential project for me as it allowed me to explore a field that could give me so many different possibilities, and reassures me in my choice to become a Data scientist.

I would like to thank my project supervisor George Adaimi for providing me with valuable advice and helping me when I felt the need to, Pf. Alexandre Alahi for hosting this project, and the EPFL Xplore team for this first experience.

References

[1] Yan Wang, Zihang Lai, Gao Huang, Brian H Wang, Laurens Van Der Maaten, Mark Campbell, and Kilian Q Weinberger. Anytime stereo image depth estimation on mobile devices. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5893–5900. IEEE, 2019.

[2] Xinyue Li and Samuel Cheng. Human outline reconstruction in depth prediction. In *2019 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 23–28. IEEE, 2019.

[3] Po Kong Lai, Shuang Xie, Jochen Lang, and Robert Laganière. Real-time panoramic depth maps from omni-directional stereo images for 6 dof videos in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 405–412. IEEE, 2019.

[4] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[5] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving, 2020.