XenomatiX
True solid state lidar

INTERNSHIP

A global summary is written for the internship done in "XenomatiX True solid-state lidar" organization.

Nishant Rajpoot
MACS

July-August 2020

Academic Supervisor	**Project Supervisor**
Prof. Dr. Ir. Adrian Munteanu	**Mr. Hung Nguyen-Duc**

TABLE OF CONTENTS

COMPANY OVERVIEW

XenomatiX [1] is a Leuven based company that works in LiDAR-based sensor technology. The company offers a true solid-state lidar sensor based on multi-beam laser and risk-free, scalable semiconductor technology. They are giving solutions in 6D Road scanning, Online road preview 2D & 3D sensing, Lidar as a reference system and Lidar integration.

XenomatiX' sensors stand out thanks to their truly unique approach:

- A built-in camera generating high-density 3D data and 2D images. This rich data set accurately reveal people and objects' position, size and shape, as well as distance and motion ahead of and around a vehicle in any weather conditions.
- Multibeam approach
- Solid-state – no moving parts
- Scalable and affordable thanks to its chip technology

In 2020, XenomatiX has obtained ISO 9001:2015 certification, which further boosts their value in safety, quality management, and rigorous validation processes.

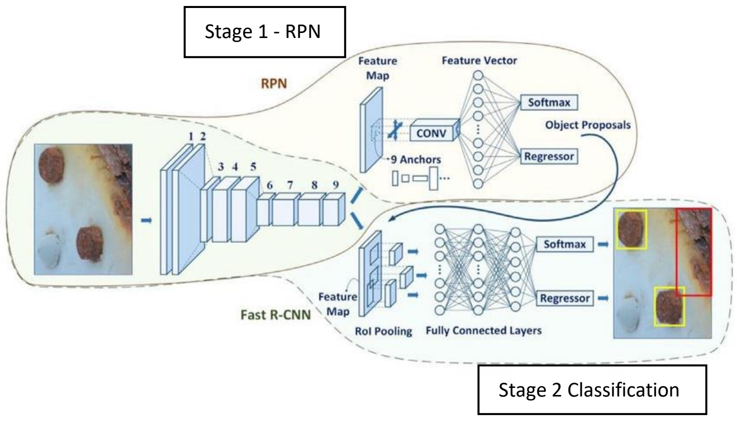
TECHNICAL AND SCIENTIFIC WORK

The internship was a part of my academic curriculum of Master's in Applied Computer Science at Vrije Universiteit Brussels. The duration of the internship was of 2 months from July to August 2020 and was done under the supervision of Professor Adrian Munteanu.

The goal of the internship was to "Find a new approach to annotate the XenoLidar visual Images to effectively train a Neural Network". XenoLidar visual images are 16-bit greyscale images obtained from XenoLidar sensor (mounted on top of the car). This new data annotation approach will specifically work for XenoLidar images and able to detect the instances (discussed later in the report) in it. The 16-bit images cannot be feed to the pre-trained Neural Network; therefore, they have to be converted to the 8-bit greyscale format, which reduces the quality of image significantly. By quality, it means the number of objects visible in an image. There are tools available for annotation, but either they work specifically for RGB images or they are completely manual or sometimes they are just expensive [2][3][4]. Our aim is basically to make a solution for ourselves and reduce the manual job as much as possible. A neural network is used to detect, recognize, and classify the objects in an image. To train a deep neural network, one should need a good sample size of annotated images [5]. However, it depends on the individual dataset, but a few thousand samples per class are recommended to pre-trained the model [6]. Each image can have as low as "No" object and as high as 30 objects (±5). To manually annotate all the objects in all the images can take a few days to weeks. Time is not the only issue; the precision of annotation is also a major problem, a robust image with objects of different size, especially small objects, are very tough to detect while doing the manual annotation. Of many issues there to train a neural network, a bad annotated image set is one of the biggest. A different approach to handle this problem was tried in this internship.

We started with a set of 7 064 XenoLidar Images that were divided among Validation (1 412) and Train (5 652) dataset. Note that these were not the RGB images, but 8-bit grayscale converted from 16-bit greyscale 1 channel format images. The main difference between an 8-bit image and a 16-bit image is the number of tones available for a given colour. The amount of tone for 8-bit is 2 to the exponent of 8, i.e. 256, which means we have 256 tones for black colour and for 16-bit it will be 65,536. Clearly, the difference is huge, and a 16-bit image has more clarity in object detection. The idea was to start from a pre-trained model, then, make a perfect annotated dataset with our solution and finally, refine the model by transfer learning.

The first job was to find a good state-of-the-art algorithm for object detection. As we did not want to start the annotation from scratch. The priority was to have more accuracy than speed. Few state-of-the-art shortlisted were: Faster-RCNN, RCNN, Detectron2, YOLOv4, YOLOv5, FCOS. The comparisons were made based on "what makes each other different", "important things for accuracy" etc. Finally, Faster-RCNN was implemented for object detection. Faster R-CNN is based on RPN (Region Proposal Network) technique. It is a two-stage method, first is finding the object with a square box (RoI) and second is classifying it. To keep the final detected objects, a threshold of confidence score of 0.8 was chosen. All detected objects which have confidence scores lower than 0.8 will be remove.



Working:

Input image > Look for ROI (Region of Interest) > feed each region to CNN > CNN extracts feature from region > prediction of bounding box.

Figure 1: Architecture of Faster R-CNN [7]

Upon running the Faster R-CNN on 8-bit XL Visual Images, it gave quite a good result, especially with small and distant objects. It was discussed in the starting to detect only 7 categories, which were (object ID – object name): 1 – Person, 2 – Bicycle, 3 – Motorcycle, 4 – Car, 5 – Train, 6 – Truck, 7 – Bus, but later "Train" was dropped because of its absence in the dataset. *Figure 2* below shows the example of a good detection by the Faster R-CNN algorithm.

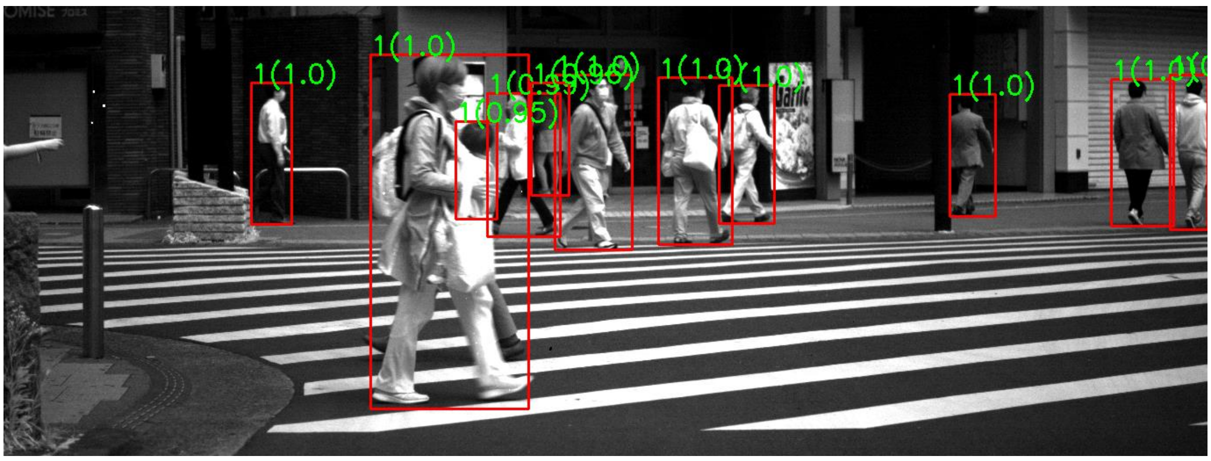


Figure 2: Good example of Faster R-CNN detection on an 8-bit XL Image. The red colour box is the region of interest and the number with green colour on top is the category and confidence score (in bracket).

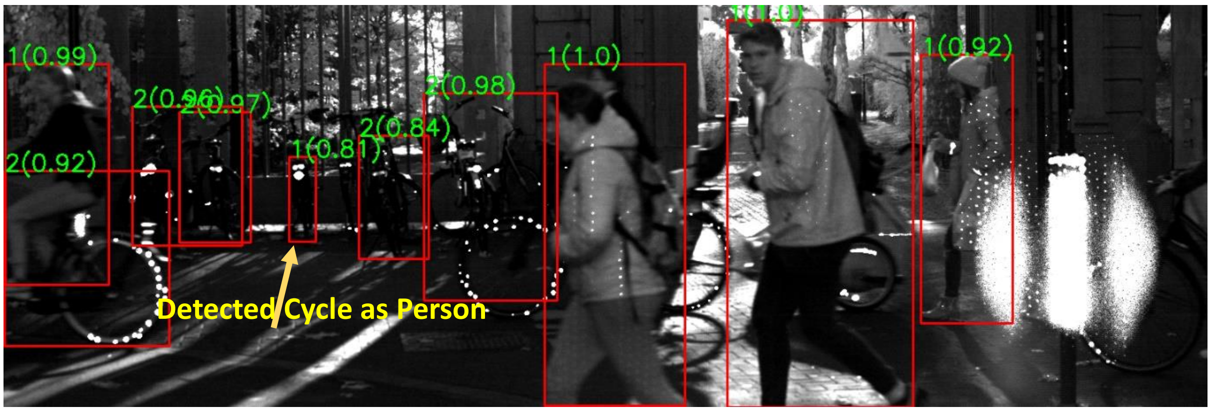


Figure 3: Bad example of Faster R-CNN detection on 8-bit XL Image. A Cycle is classified as a Person.

It took almost 2 hours to do the object detection of 1 412 images on my machine with the specification of Core-i7 8th Gen, 12 GB RAM and 2GB Radeon graphics card. The output of detection was fairly good, almost every image got a detection, but of course, not all the detections were correct. With every image detection, a separate text file is created where each line in the text file corresponds to one object in the detection result. (x1,y1) is the top-left point, (x2, y2) is the bottom-right point of the bounding box, and "id" is the object class id {1: pedestrian, 2: bicycle,3: motorcycle, 4: car, 6: truck, 7: bus}. This, we called a "predication" set. The next aim was to create a "ground truth" set to validate how good or bad was the prediction. To manually annotate the 1 412 images, an open-source data annotation tool name "LabelImg", written in python language, was used. It takes two arguments, an image and its corresponding Pascal VOC file. To do the conversion (from text to VOC) a special script "pascal_voc_io.py" was used. This function helps in converting the ".txt" file in the "prediction" set to the Pascal VOC format(.xml) and vice versa.

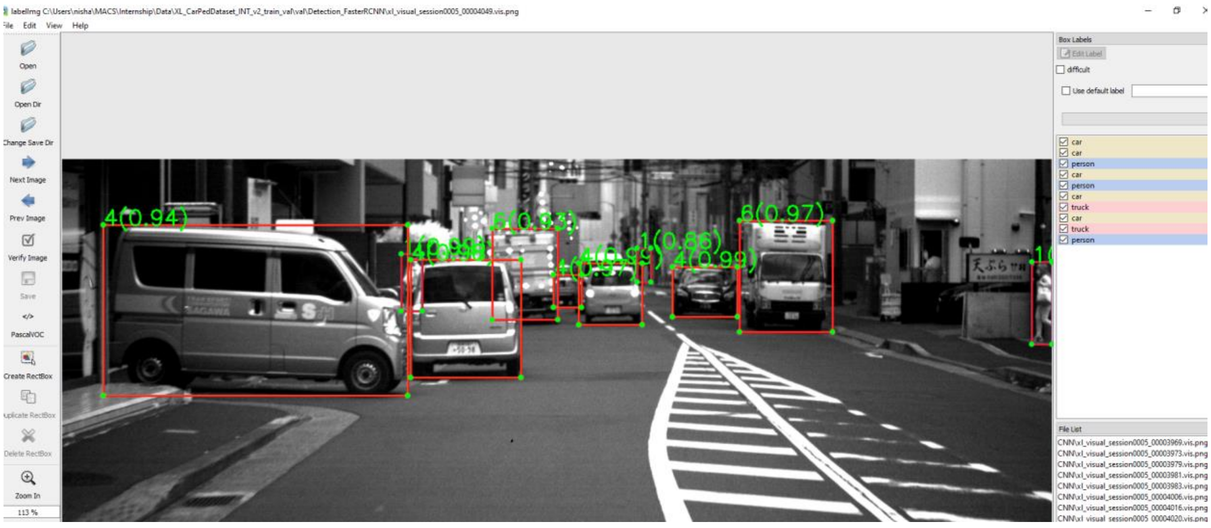

labelImg C:\Users\nisha\MACS\Internship\Data\XL_CarPedDataset_INT_v2_train_val\val\Detection_FasterRCNN\xl_visual_session0005_00004049.vis.png
File
Edit
View
Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
PascalVOC
Create RectBox
Duplicate RectBox
Delete RectBox
Zoom In
113 %
4(0.94)
4(0.98)
6(0.93)
4(0.97)
4(0.99)
1(0.86)
4(0.99)
6(0.97)
Box Labels
Edit Label
difficult
Use default label
car
car
person
car
person
car
truck
car
truck
person
File List
CNN\xl_visual_session0005_00003969.vis.png
CNN\xl_visual_session0005_00003973.vis.png
CNN\xl_visual_session0005_00003979.vis.png
CNN\xl_visual_session0005_00003981.vis.png
CNN\xl_visual_session0005_00003983.vis.png
CNN\xl_visual_session0005_00004006.vis.png
CNN\xl_visual_session0005_00004016.vis.png
CNN\xl_visual_session0005_00004020.vis.png

Figure 4: LabelImg tool for manual annotation of the dataset.

Once the ground truth dataset was created, our next step was to evaluate the prediction set by comparing the "Prediction" boxes and "Ground Truth" boxes. To calculate the result, two parameters were taken into consideration, Precision and Recall.

Precision is basically "How many selected Items are relevant?" and Recall is "How many relevant items are selected?". To quantify these definitions, the formula used was:

$$\text{Precision} = \frac{(True\ Positive)}{(True\ Positive + False\ Positive)} \qquad \text{Recall} \quad = \frac{(True\ Positive)}{(True\ Positive + False\ Negative)}$$

True Positive: Correct bounding box (threshold>0.8) and classification is correct
False Positive: Correct bounding box and classification is not correct OR Have Bounding box but no ground truth
False Negative: Ground truth is present, but the model failed to detect
True Negative: Irrelevant

Category wise Average Precision (AP):
{Person: **97.47%,** Bicycle: **72.09%,** Motorcycle: **63.08%,** Car: **93.29%,** Truck: **65.43%,**
Bus: 70.97%}

mean Average Precision (mAP): **77.58%**

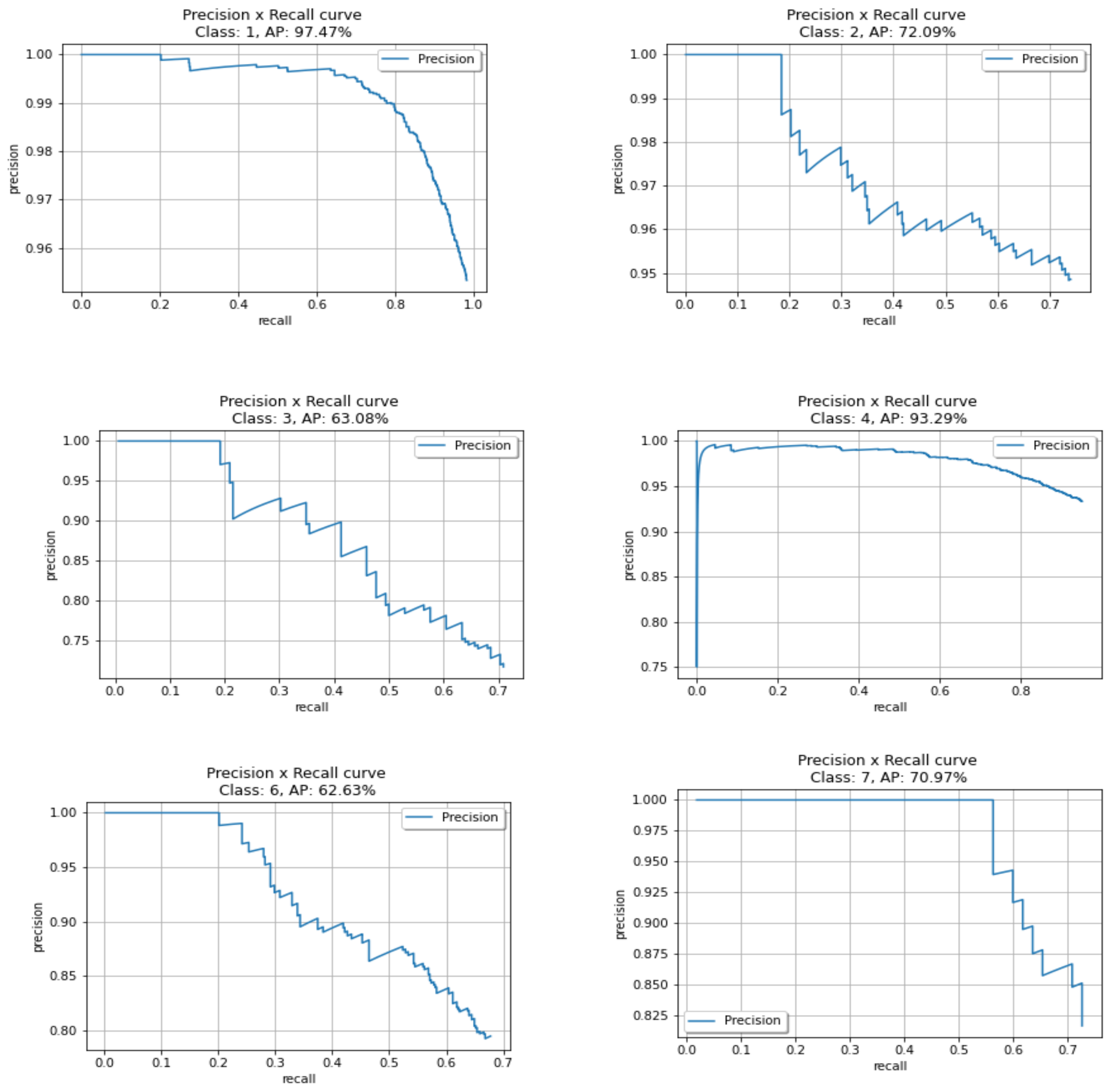


Figure 5: 11-point Precision-Recall curve for all the categories [1-7]. The precision (Y-axis) is plotted as a function of 11 standard Recall (X-axis) levels {0.0,0.1,0.2....1.0}. Precision 1.0 means most relevant and Recall 1.0 means found all the relevant documents. Likewise, Precision 0.0 means least relevant and Recall 0.0 means found no relevant document. [8]

Note that there were no images for category 5: Train, hence no graph. As it can be observed from *Figure 5* that the Average Precision of Class 1 (person) and Class 4 (car) is significantly better than other classes. The reason can be found in the percentage distribution of the categories in our dataset:

Car: 49.15% Person: 38.18% Bicycle: 6.88% Truck: 3.76% Motorcycle: 1.53% Bus: 0.49%

Clearly, the quantity of Car and Person in our dataset is ~90% of all the objects. Hence, it can be one reason why these two object types have higher Average Precision (AP) than others. Also, a script was written to label all the bounding boxes as either True Positives (TP), False Positives (FP), and False Negatives (FN). For labeling the boxes as TP, FP and FN, Intersection over Union (IoU) of "Ground Truth" boxes with "Prediction" boxes were calculated.

Until this point, we just analyzed the 8-bit XenoLidar data given to us and the performance of Faster R-CNN on it. The next step is to make a good annotated "train" dataset, which consists of 5 652 images. First thing first, the Faster R-CNN was run on this "train" dataset and as we did with the "validation" set, text files were generated with the same format and then converted to Pascal_VOC (XML) format. As it was discussed in the starting that our aim here is to reduce the manual annotation as much as possible, so to achieve that, two solutions were discussed and implemented in this internship and both the solutions have an aim to remove the correct False Positives (FP) from the "train" dataset.

In the first solution, we make use of 3D Point cloud data of XenoLidar sensor to have parameters that will uniquely define a bounding box with "label" as True Positive (TP) or False Positive (FP). We were basically finding all the point clouds that lie inside our bounding box. XenoLidar's 3D point cloud is a Nx13 matrix, out of which, first 7 are useful to us, which are: x [mm], y[mm], z[mm], intensity, u [px], v[px] and distance [mm]. Where, x, y, z represents the coordinate in 3 dimensions, u and v correspond to the 2-dimension coordinates and last is 3D distance. A function was created to save the values of each bounding boxes (and the point clouds lie in it) in a text file of every image. There were few cases, with no 3D point cloud in the bounding box, it was replaced by [-1,-1,-1,-1,-1,-1, id], just so we don't skip the boxes. Also, to make the visibility better, an MS Excel file was created from text files and 6 unique parameters were selected for each bounding box and a label column. The **6** parameters were: 3D distance, 2D box area, Number of points, Quadrant, Intensity, and 3D height and a Label column with "TP" or "FP". This was done on the "validation" dataset and "Train" dataset, the only difference was in the "Train" set, we did not have "Label". It is yet to be predicted. To make the prediction, we tried some known binary classification algorithms and found Random Forest was giving the best Result in Predicting correct "FP". The algorithm gave 96% accuracy with cars and 83% accuracy with Pedestrians on the "Train" dataset. For accuracy %, manually all the "False Positives" were checked. Other classes had very few FP, so binary classification wasn't performed on them.

In the second solution, we made use of 3D sensors data of XenomatiX Lidar. The 3D Sensor Coordinates system "Rsensor" is defined as follows: +x points forward, +y points to the right, +z points down as shown in *Figure 6*. The idea here was to distinguish a point cloud as a Ground point/ Non-ground point and then collectively see the number of ground points in a bounding box and if it exceeds a certain threshold then classify it as a False Positive. The solution is explicitly created for the images where the ground surfaces were predicted as objects. However, the function was not very stable but still, it was able to predict the obvious False Positives. An example of "ground point" detected as object has shown in *Figure 7*.

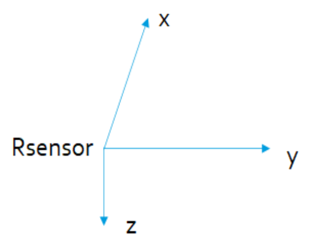


Figure 6: 3D sensor coordinate system

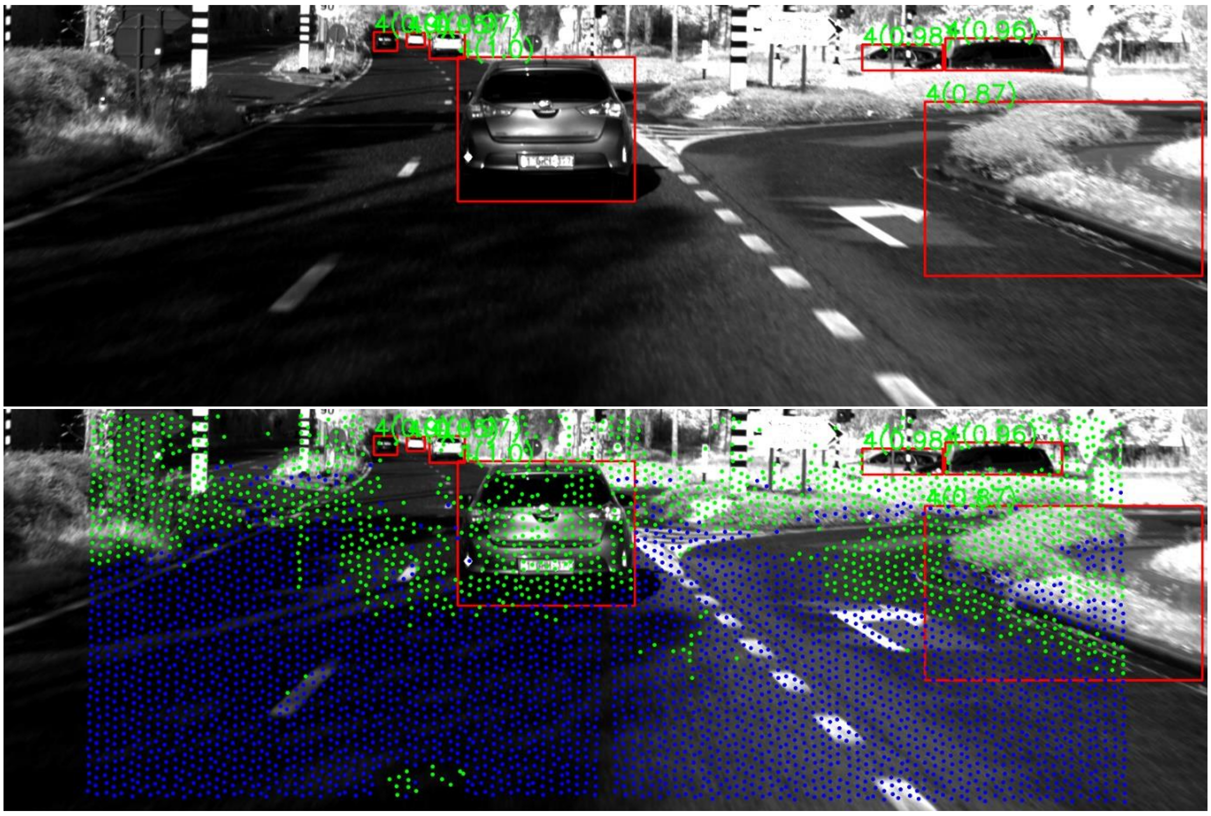


Figure 7: [Top] Bounding Box at the right has been detected as a 'car' (FP) on 8-bit Faster RCNN. [Bottom] Corresponding point cloud is drawn over the image, where the green points are predicted as 'non-ground' and blue as 'ground'. Clearly, the number of 'ground' points in that box are greater than the threshold we set, therefore classified as 'FP' and hence removed according to our solution.

Upon finishing the implementation of both the solutions and removing the False Positives, the last step in the "Train" dataset was to do the manual annotation of the remaining bounding boxes. Once done, we were left with a fully annotated Validation set (1 412) and Train set (5 652) with 6 classes. The last step was to do the transfer learning. For this, we used fine-tuned Faster R-CNN with Resnet 101 backbone. Since we did not have a sufficient number of images to train the Neural network from scratch, we used the pre-trained model. We keep the Backbone and Feature Pyramid Network (FPN) as it is and only changes the last layer of each branch by a new layer corresponding to our new number of classes. We then retrain these new linear layers with our dataset and then refine the backbone and FPN. This training would take more power to run, hence it was done on XenomatiX's machine.

Comparison of the result, accuracy, and precision is not straight forward, since the prediction that was made on the "Test" dataset was an 8-bit image, but the training was done on a 16-bit image. Hence, finding the accuracy and precision for 16-bit images will not be accurate. The new images predicted by the model on 16-bit images might not be present in the original 8-bit ground truth images, therefore, it will count as an FP. Since "False Positive" is present in the denominator of both the Precision and Recall formula, therefore, with the increase of FP in the result will lead to a decrease in Precision and Recall and hence Overall accuracy.

Therefore, to come up with this discrepancy, we divided our result into Quantity and Quality based. For Quantity, we simply calculated the average precision (AP) and a mean Average Precision of the new detection and like explained in the above paragraph, average precision (AP) for each category has decreased (compared to the initial prediction on 8-bit Faster R-CNN, as shown in *Figure 5*), due to the increase of False Positives because was unable to detect those objects in our original ground-truth dataset.

For the Quantity result, the average precision of each category was calculated and compared with the ground truth data.

Category wise Average Precision (AP):
{Person: **73.33%,** Bicycle: **47.12%,** Motorcycle: **77.08%,** Car: **74%,** Truck: **60.47%,** Bus: **65.48%**}

Mean Average Precision(mAP): **66.23%**

For Quality result, we simply compared the 8-bit ground truth images with the final 16-bit image predicted by the Faster R-CNN model. A script was written to run all the predicted images in a slide show. The prediction was accurate and almost all the boxes were accurately predicted by our trained model. Also, it was observed that the new model on 16-bit images was more robust to images like in *Figure 8* as compared to 8-bit ground truth images. That is why, if we look at the object count in 8-bit ground detection {Person - **4277**, Bicycle - **800**, Motorcycle - **199**, Car - **5525**, Truck - **432**, Bus - **55**} and 16-bit final detection { Person - **5762**, Bicycle - **1642**, Motorcycle - **329**, Car - **7749**,Truck - **708**,Bus - **78** }, 16-bit image model has almost "44% more objects" than 8-bit ground-truth dataset. These "44% more objects" are counted as False Positives in our "Quantity" result, which in fact is not correct, because these are the additional boxes that were predicted in 16-bit images.

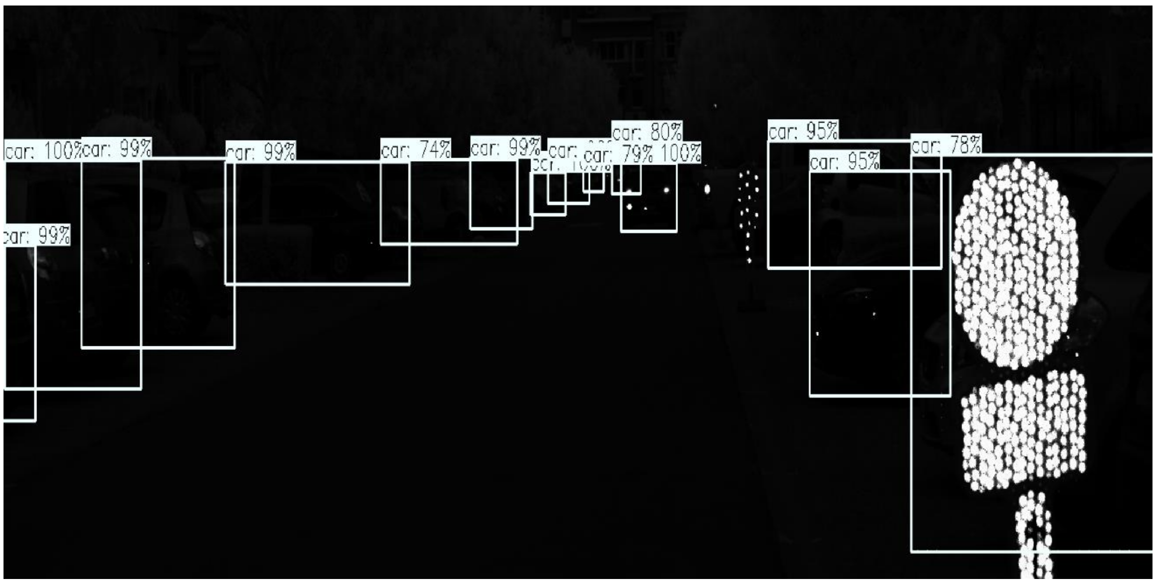
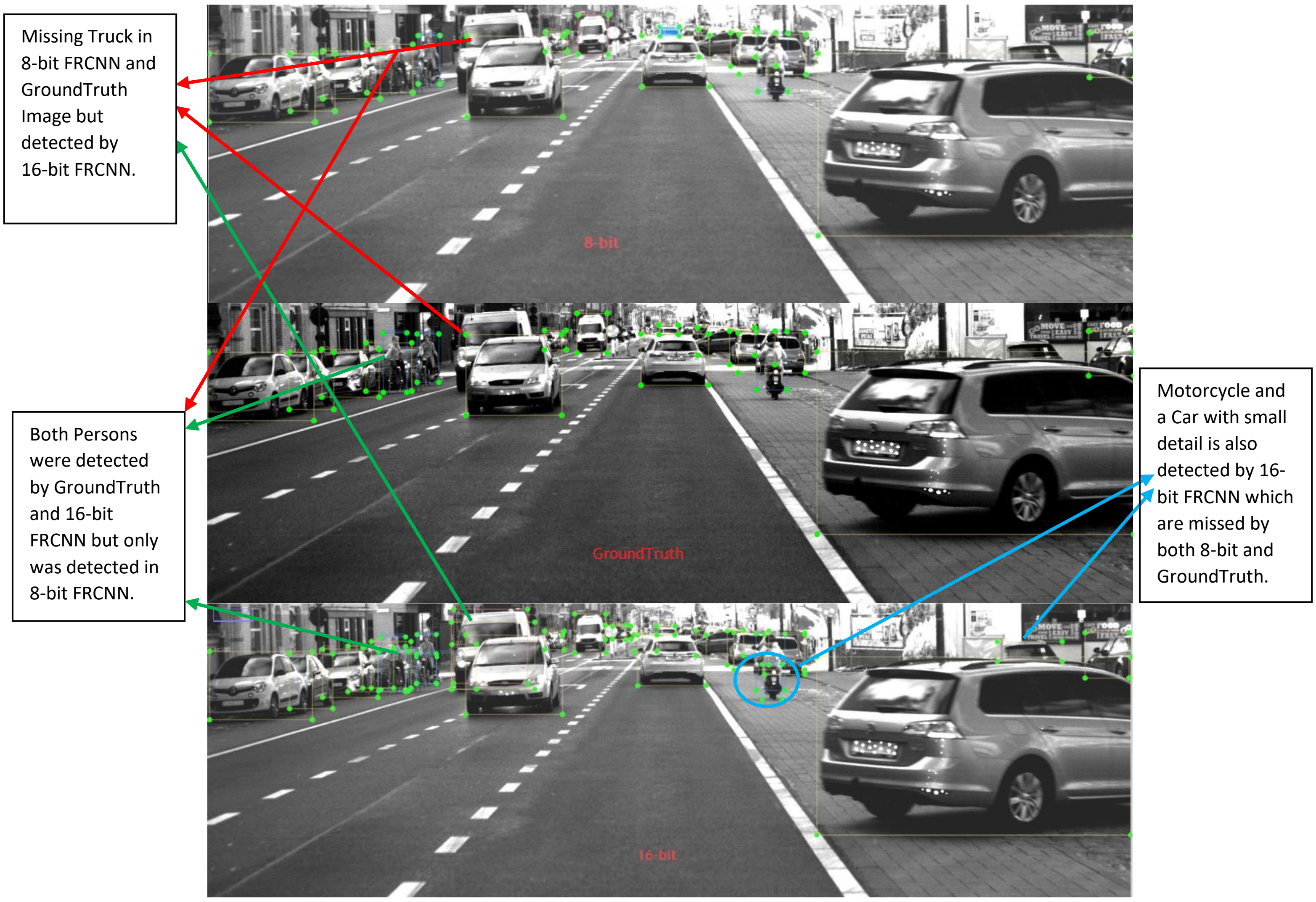


Figure 8: A robust prediction of Image classification on a 16-bit XL-Visual Image



Missing Truck in 8-bit FRCNN and GroundTruth Image but detected by 16-bit FRCNN.
8-bit
Both Persons were detected by GroundTruth and 16-bit FRCNN but only was detected in 8-bit FRCNN.
GroundTruth
Motorcycle and a Car with small detail is also detected by 16-bit FRCNN which are missed by both 8-bit and GroundTruth.
16-bit

Figure 9: A comparison of XL Visual Image on 8-bit Faster R-CNN, Ground Truth, and a 16-bit Faster R-CNN.

Figure 8 shows that how robust a prediction can be made on a 16-bit image which is nearly impossible to detect by human eyes. However, for verification it has to be converted first. *Figure 8* is a comparison made between the XL Images of 8-bit Faster R-CNN, Ground truth, and 16-bit Faster R-CNN. Ground truth is based on 8-bit XL images, hence, there would be more similarity in the result as compared to Ground truth and 16-bit images. In *Figure 9*, it can be observed that 16-bit Faster R-CNN image has more bounding boxes, mostly True Positives, which are absent in Ground truth. In 16-bit faster R-CNN, there are multiple bounding boxes on the same object, which means we have something to do with the non-maxima suppression. Although, in this case, the Truck, Motorcycle was successfully detected in 16-bit faster R-CNN but we need to add more samples in our training set. These more detections explain the reason for the poor accuracy result of 16-bit faster R-CNN.

This leads to the following future work item for the data annotation:

Build a proper tool to read and visualize the 16-bit image. This tool should allow the user to adjust the window level in order to see better objects in the dark zone. This tool should also allow to add/remove/adjust object bounding boxes (similar to labelImg).

WORK ENVIRONMENT

The work environment of the company was really nice and professional. Everybody was friendly to me. In the 2 months of my internship, I mostly worked with my project supervisor, Mr. Hung Nguyen-Duc, who is working as a Vision Software Engineer in the company. Due to the COVID-19 situation, we kept our communication over emails and calls as much as possible. The data was shared through google drive on a common XenomatiX's ID. In the mid-duration of my internship, I gave the presentation of my work to the whole software team in the weekly meeting of the company. I got the answers to my questions well on time and the internship went smoothly, irrespective of the COVID situation.

DEVELOPED SKILLS

The internship has really polished my skills in python coding language. I got familiar with the XenoLidar sensors, its working, and Xenoware software. Since the topic was based on visual image classification, so I got to study various state-of-the-art algorithms in detail. Moreover, I implement the Faster R-CNN algorithm for the internship and learned the working of the Neural Network for object classification. I learned about the difference between 8-bit and 16-bit images and how they can affect the working of the model and its training. I got familiar with the working of 3D point clouds, their connection with sensors, and how to use its different parameters for solving different classification problems. Finally, the techniques of result analysis and data visualization.

REFERENCES

1. XenomatiX N.V. Copyright and Trademark: https://xenomatix.com/disclaimer/
2. Isaac Ronald Ward et al, "RGB-D image-based Object Detection: from Traditional Methods to Deep Learning Techniques".
3. "Intel open-sources CVAT, a toolkit for data labeling". VentureBeat. 2019-03-05. Retrieved 2019-03-09.
4. Example of pricing of a good Data Labeling Tool: https://labelbox.com/pricing
5. Dernoncourt, Franck, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. "De-identification of Patient Notes with Recurrent Neural Networks" arXiv preprint arXiv:1606.03475 (2016).
6. Cireşan, Dan C., Ueli Meier, and Jürgen Schmidhuber. "Transfer learning for Latin and Chinese characters with deep neural networks." The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1-6. IEEE, 2012. http://people.idsia.ch/~ciresan/data/ijcnn2012_v9.pdf
7. Shaoqing Ren et al, "Faster R-CNN: Towards Real-Time Object detection with Region Proposal Networks".
8. Page 160, Book: "Introduction to information retrieval" by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze.