University of Zurich[UZH]

MASTER PROJECT – Communication Systems Group, Prof. Dr. Burkhard Stiller

LiCounter: LiDAR Based Cloud Counting Method

Jiaqi Zhang, Chuqiao Yan
Zurich, Switzerland
Student ID: 20740213, 20740619

Supervisor: Bruno Rodrigues, Eder Scheid, Prof. Dr. Burkhard Stiller
Date of Submission: February 16, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Project
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

LiDAR has been widely used in many fields since it was invented 50 years ago. Nowadays. It is especially popular in the area of autonomous cars to detect objects. However, LiDAR could also provide a new means to improve the way of monitoring objects and analyzing their behaviors. Considering one scenario of an in-store shopping environment, where a camera detects customers and tracks the time that people stayed at some positions. Based on the detected data, the business companies could analyze peoples behaviors and improve business strategies by providing personalized shopping experiences. With the collaboration of Livealytics, this project will bring the two parts together by using two 3D Intel RealSense L515 LiDAR devices and developing a method of tracking in-store customers. The method LiCounter will be developed by combining Yolo and Deep Sort algorithms for detection and tracking, which is evaluated in real locations to collect the data. Further, data is used for classification results by Gaussian mixture clustering method based on business metrics. At the end, it gives a visualization of clustering results for each user.

Acknowledgments

We would like to express our greatest appreciation to our supervisor Bruno Bastos Rodrigues who provides guidance, support, suggestion and comments during our development of this Master Project. We would also like to express our gratitude to Eder Scheid, Professor Burkhard Stiller, and the Communication System Group for the support, providing the equipment and this opportunity for us to work within this interesting field.

Contents

Chapter 1

Introduction

While web user behavior analytics is prevalent, in-store customer behavior analytics is less common due to difficulties in tracking users without disturbing them. With WiFi and LiDAR combined, we can take a passive approach to monitor user movements within stores. Several analogies between online and in-store behaviors can be established, and similar approaches to online customer analytics can be used for in-store performance metrics extractions, as well as machine learning approaches for customer segmentation. We can therefore use the analytics results to optimize marketing campaigns and personalize shopping experiences, which help companies to gain competitive advantages. This will be approached by detecting the positions and duration of customers.

Even though customers shopping experiences have involved many social perspectives, understanding the process could benefit companies to gain competitive advantages by optimizing market campaigns and personalizing the shopping experiences. It will provide insights for researchers about which area has higher shopping density, which path that customers passing by are the most efficient, etc [25]. In-store tracking is one of the major means to capture the activities that customers interact with the environment, as well as observing the flow of customers. However, privacy and prevalence of customers become the biggest concern with the usage of cameras [15]. Not only customers private information will be recorded, but also cameras could disturb customers and affect their shopping experiences.

With a passive approach, LiDAR (Light Detection and Ranging) scanner could be an ideal device for in-door tracking, avoiding privacy concerns raised by use of cameras. Considering the accuracy and area coverage, two 3D-LiDAR (Intel Realsense L515) devices will be used simultaneously in this study.

1.1 Motivation

Understanding customers and their behaviors is a key point of any company. The analysis of wireless signals emitted by portable devices such as smartphones, laptops, and tablets enables the extraction of the positional data from those devices. In public locations,

security measures such as emergency routes can be enhanced by analyzing crowd behavior. Considering the scenario where it is necessary to observe the flow of people in a public environment of passing by, requiring people to associate their mobile devices with an access point is not a feasible strategy. Thus, the passive approach to monitor mobile devices for this use case is ideal. Within the range of passive tracking approaches, it is possible to make the tracking more accurate by combining passively monitored data, with data gathered by a LiDAR (Light Detection and Ranging) scanner.

This project is in collaboration with livealytics [1], a company using tracking measurement solutions to provide insights in order to improve live customer experiences, optimize operational costs and increase sales [2].

A bachelor thesis *LaFlector: Passive Tracking based on LiDAR* by Lukas Mueller explored using a single 2D LiDAR for tracking [16]. This thesis explore the design proposed on the bachelor thesis, further extending the approach to a three dimensional approach using two 3D LiDARs for more reliable tracking results.

1.2 Project Goals

There are four major steps in this project.

Method of detection and tracking Collecting raw distance data by LiDAR, an algorithm of detecting the class of people and tracking the detected object is trained and performed.

Data Storing At each timestamp, the person ID, the center location of the detected person, and the distance from the person to the camera collected from two LiDAR cameras are sent to InfluxDB on a virtual machine.

Method testing A designed scenario based on test results are described for realizing and evaluating the method performance.

Data Analysis Data collected from testing scenarios is used for building a clustering model to predict the interest level of each customer.

1.3 Thesis Outline

Section 2 presents related theoretical work for object detection and tracking, as well as an introduction on hardware preparation, database support and algorithm choices. Section 3 presents the solution of this from architecture design to implementation. Section 4 is the detailed implementations on data collection, localization, tracking, data storage, synchronization, light conditions testing, and data analysis. Section 5 is the evaluation in a designed scenario. Section 6 shows the final considerations, including a summary of this project, conclusions and future work suggestions.

[1] https://www.livealytics.com/

Chapter 2

Background and Related work

This section gives an overview of related theoretical work which represents the background research to realize the goal of this project. The research covers related work about object detection, tracking method, other developed localization methods, classification method, LiDAR background information, InfluxDB, virtual machine and Network Time Protocal.

2.1 Related Work

2.1.1 Object Detection

There are many object detection algorithms available. Deformable parts model (DPM) is one of the most popular algorithms which is capable of handling large variations but speed is a big concern [24]. The classifier of DPM adopts a sliding windows approach which runs through each evenly spaced space over the image [12]. The other algorithm R-CNN stands for regional proposal network, which is accurate but slow only with 5 frames per second [3]. The method of R-CNN is hard to optimise because It runs a classifier on potentially generated bounding boxes in the image, and then refines bounding boxes, eliminating the duplicates [12].

YOLO (You only look once) is a new emerging method based on 53 convolutional neural network layers that can be deployed in consumer products which is fast [12]. For a real-time detection task, YOLO could achieve more than twice the average precision than other detection systems. Also, YOLO processes the entire image instead of regional or sliding window methods to avoid mistakes of background patches. Further, YOLO learns a highly generalizable representation of objects [12].

Reasoning globally is an advantage of YOLO since it uses features from the entire image to predict bounding boxes across all classes simultaneously. For each input image, it will be divided into $S \times S$ grids. Within each grid, bounding boxes and confidence score will be predicted which reflects how accurate the box contains the objects. Bounding boxes contain 5 values, centre position of bounding box x, y; width, height of boxes in the image,

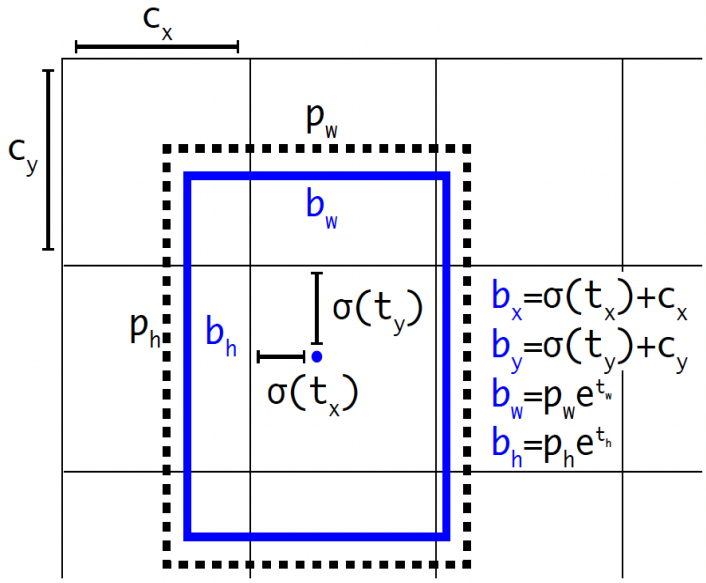


Figure 2.1: Bound box predictions [11]

and confidence. Confidence is the measurement of IOU(Intersection over union), Formula 2.1, between the predicted box and any ground truth box. Conditional class probabilities are also predicted for each bounding box. It represents which classes the detected object belongs to and how accurate the prediction is. Calculation formula is shown in Formula 2.3 [12].

$$IOU = \text{Area of Overlay} \div \text{Area of Union} \tag{2.1}$$

$$\text{Confidence Score} = \Pr(Object) \times IOU_{pred}^{truth} \tag{2.2}$$

$$\Pr(class_i|Object) \times \Pr Object \times IOU_{pred}^{truth} = \Pr(class_i) \times IOU_{pred}^{truth} \tag{2.3}$$

In YOLOv3, updated four predictions for bounding boxes x, y, w and h will be calculated with sigmoid function based on previous version, as shown in Figure 2.1. Binary cross-entropy loss is used for class prediction. The neural network Darknet-53 is performed for feature extractions. The structure of the neural network is shown in Figure 2.2 [11].

2.1.2 Customized object detector

Instead of reinventing the wheel, we referred to the *YOLOv3-Cloud-Tutorial* by the AI guy. [1] This tutorial includes enabling GPU on google colab, cloning and building the darknet,

[1] https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.2: Architecture of Darknet-53 Network [11]

labeling customized dataset, configuring files for training, downloading pretrained weights, training and visualizing the result [19].

We choose to follow this tutorial because first Google Colab offers GPU which makes the training of our model much faster. Secondly, darknet is an open source neural network framework which is fast and easy to install, i.e. darknet13 in this project, and we use a famous implementation from AlexeyAB [2]. In addition, this tutorial suggests some good choices for hyperparameters which work well with our training and testing data.

2.1.3 Object Tracking

Tractor is a framework that was proposed by Bergmann et al. based on the detection module, but it is limited by the appearance of a target which could be uncertain or occluded; as well as the time of tracking which is unable to re-identify of the second appearance of the object [18]. Siamese Track-RCNN contains three branches: track, detect and re-identify of the object, which is accurate but might be slow. Another model, Jointly learns the Detector and Embedding model (JDE), it outputs detected objects and the corresponding embedding of the detected boxes at the same time. This presents nearly real-time detection but is limited by low resolution [23]. However, Deep Sort is an ideal algorithm that could be applied in this case, it is fast with 16 frames per second, accurate and handles occlusion very well.

[2]https://github.com/AlexeyAB/darknet

Deep Sort could handle the occlusion for a longer time, and reduce the number of ID transformations [7]. Deep Sort stands for Extension of SORT (Simple Online Real time Tracker) Algorithm, which combines detector and tracker for multiple object tracking. Detector will use the one trained from YOLOv3. Kalman filter is used in the tracker for prediction. It processes frame-by-frame data to find the object position and velocity of motion by predicting the position of the target in the current frame and updating the tracker parameters [5].

For matching, Deep Sort uses the linear combination of motion information and feature information. Equation 2.4 is the measurement of Mahalanobis distance for motion information, which indicates the distance between the detected result from Kalman filter and predicted result. d_j is the position in j-th frame, y_i is the position by Kalman filter by i-th tracker. A threshold is pre-set for Mahalanobis distance, which means successful matching if the distance is below a threshold [7].

$$d_{1i,j} = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \tag{2.4}$$

$$d_{2i,j} = min\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in \mathbb{R}_i\} \tag{2.5}$$

$$c_{i,j} = \lambda d_{1i,j} + (1 - \lambda) d_{1i,j} \tag{2.6}$$

Cosine distance is the second measurement for matching to solve occlusion and frequent ID conversion in rapid displacement. As Equation 2.5 shown, it calculates the minimum cosine distance between the feature set and descriptor. The final Deep Sort measurement is the linear combination of two distance measurements, refer to Equation 2.6 [7].

2.1.4 Other Localization Methods

For tracking methods of the indoor environment, several related methods have been developed. BluePIL is a passive system for Bluetooth device localization, which delivers results in near-real-time. It uses a path loss model for localization [14]. Similarly, LaFlector is a LiDAR-based indoor tracking system. It is capable of detecting, classifying and tracking several objects simultaneously which can also be used to count and track visitors interest [16]. ASIMOV is another passive tracking method that uses WiFi to identify device information by the correlation of randomized information about the displacement of devices [13].

2.1.5 Clustering

Clustering analysis is an unsupervised machine learning technique commonly used for customer segmentation tasks. To train a clustering model, we do not need to label data.

Sen et al. used mobile sensing data to detect in-store customer behaviors in order to tell customers shopping intentions by performing a 2-stage clustering-cum-classification technique.[17] Kansal et al. explored three different clustering methods, namely k-Means, Agglomerative, and Meanshift to study customer segmentation, and found 5 segments including Careless, Careful, Standard,Target and Sensible customers.[22]

The number for K in K-means is calculated by elbow method. SSE is the cum of square distance between centroids and samples in that cluster. Elbow is the point where as K increases, the most decline in SSE is found. K centroids are initialized, and each sample point is assigned to the closest centroid. Centroids then get updated as the mean of each cluster. Samples are assigned to each new centroid, and centroids get updated. This iterative process continues until the location of centroids no longer changes.

Agglomerative Clustering forms hierarchy using dendrograms. By merging closest data points, clusters form.

2.2 LiDar

2.2.1 Background

LiDAR is an acronym of Light Detection and Ranging. This instrument is used to determine the distance by targeting an object with light pulses and measure the time for the reflected light to return to the receiver. The first application dates back to 1970s to determine the flight height of an airplane. Later, it also applied in the advent of global positioning systems [1]. In this project, Intel Realsense L515 LiDAR devices is used, which is a depth application especially designed for an in-door environment.

2.2.2 Intel Realsense L515 LiDAR Strength and Weakness

The Intel Realsense L515 LiDAR is a small size device, which is convenient and easy for mobility. It is the worlds smallest high-resolution LiDAR camera, with only 100 grams. Without ambient light, It can capture accurate depth data in the range of 0.25m to 9m [6], which is ideal for indoor environment detection use cases. This type of camera has a wide depth field of view, $70° \times 55°$ ($\pm 3°$). For an indoor environment, it is wide enough to cover the detected areas in this project. The detected image by LiDAR has a clear edge based on depth which is beneficial for object detection.

However, drawbacks still exist. The most influential concern is the ambient light which greatly interferes with the detection range. Infrared light from sunshine could degrade the resolution of detected images and affect the measurement results. Thus, the extent of light affection is considered and measured in this project as well.

2.3 InfluxDB

InfluxDB is an open source purpose-built database system to store time-stamped data, it could keep the high resolution of raw data and handle millions of data per second. Currently, it is the most popular time series database system. It supports several programming languages, SQL-like query language; also, it could be easily installed [9]. With the advantage of high throughput ingest and real-time querying, InfluxDB will support high volume of depth data and two LiDAR devices synchronization in this project.

2.4 Virtual Machine

Virtualization is a major technology for cloud computing, which was introduced in the 1960s by IBM. It is a transparent way of sharing both time and resources for multiple users to use hardware concurrently. With the rapid growth of processing and storage power, it becomes more abundant, powerful and cheaper [10]. In this project, a VM is used for a convenient synchronization of both LiDARs.

2.5 Network Time Protocal

Time latency of two LiDAR devices would affect the process of detecting people during the test step, for instance, if one LiDAR responses later than the other one, there will be two records of the same Person in the data. After connecting to the virtual machine, Network Time Protocol, is applied to synchronize two devices to make the individual device's clock accurate. The Switzerland time pool is selected to connect two devices.

Chapter 3

Architecture and Workflow

This chapter describes the detailed scope of this project from a general perspective. Section 3.1 and 3.2 give the requirements and assumptions of this project which lead to the design of the architecture in Section 3.3. Section 3.4 describes all the components in each phase, which is also the list of implementations introduced in Chapter 4.

3.1 Requirements

Based on the objectiveness of this project, feature requirements need to be satisfied. These requirements also relate to the design of the architecture and workflow of each component.

Color images for training model: The data read from LiDAR are depth data which are visualized as grayscale images. Images are applied with colormap in order to train the detection model to recognize customized people. A pool of images needs to be collected.

Valid Localization method: The method developed for detecting and tracking should be valid when LiDAR starts to detect people. It should be able to detect, track and count people when they are in the frame, including different postures of people and multiple people in the same frame.

Synchronization and data storage: Two LiDARs need to be synchronized. They send data to the same data storage space with synchronized timestamps in order to add up the correct number of people at each timestamp. When a person is near the border of two cameras, the person is detected twice, once by each camera, but should be counted as one.

Metrics for classification: Each detected person should have a class label of interest level. The classification results should be visualized as a graph.

3.2 Assumptions

The designed method are performed based on an ideal conditions, which is assumed as follows:

Usability of LiDAR: There is no functional problem of LiDAR performance, which returns very clear images with sharp edges of all objects.

Position of LiDAR: Two LiDARs are always placed horizontally at the same height or have the same setup in the scenario. They both see the same objects and not disturbed by other devices.

3.3 Architecture

To meet the requirements mentioned on Section 3.1, the architecture of this solution contains mainly two parts, local machine and virtual machine. cf Figure 3.1. Local machines are used for first steps, including collecting training images, training a detector and tracker. Images are collected with only one camera. Also, the classification is performed on a local machine with a visualized result returned. To analyze the data locally, it is easier to define metrics and change the measurements.

The other part is a virtual machine, which is aiming for synchronization of two LiDAR devices. Data collected locally is sent to the virtual influxDB for storing and querying, and then the queried data is sent back to the local machine. In this way, more LiDAR devices could be added if it is needed, since they all connect to the Virtual Machine and data can be stored in the cloud instead of the usage of a local socket.

3.4 Components of Workflow

The workflow of this project contains three phases: The first phase is to design a method for localization; the second phase is synchronizing two LiDARs and preparing data for segmentation; and the last phase is data analysis.

A detailed illustration of the workflow is defined in the Figure 3.2. Three phases in total. In phase 1, only one LiDAR will be used to collect images in different scenarios, these images will be labeled manually in order to train the detector of YOLOV3 and the tracker of Deep Sort. Virtual machine will be set up with InfluxDB installed, which is able to receive data from both LiDARs. Based on the result from phase 1, the detected distance of LiDAR with light is tested for a scenario setup. Two LiDARs are used in a designed mock scenario of shopping, where location, light and setup would be measured to test the method. Data are collected from the scenario, which is used for segmentation in the next phase. Phase 3 is implemented locally, containing data cleaning, defining metrics for classification and user segmentation. Finally, a clustered result is returned containing whether the user is interested in an object or not.

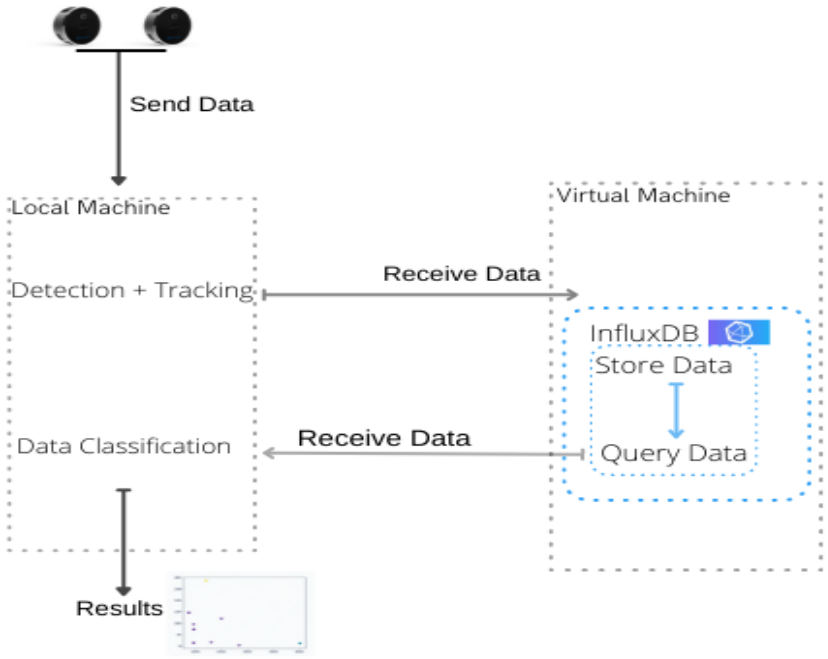


Figure 3.1: Architecture of the solution

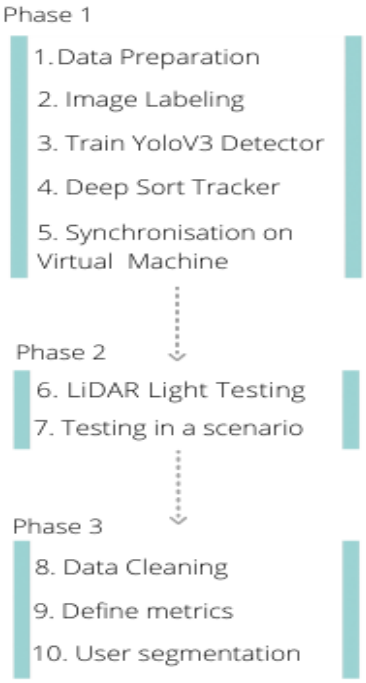


Figure 3.2: Workflow of the solution

Chapter 4

Implementation

This chapter describes the details of components to realize the method of this project, as illustrated in the previous chapter. Section 4.1 is about how the data is collected, which is a basic step of further implementation. Section 4.2 and 4.3 describe the training process of the YOLOv3 detector and the Deep Sort tracker, these two sections. Section 4.4 is aiming to describe the process of storing and querying data for further steps. Section 4.5 mentions the implementation of the virtual machine for synchronization of both LiDAR devices. Section 4.6 is the step of light and distance testing, which helps to define the testing scenario setup. Section 4.7 is the step of data analysis, in specific classification.

4.1 Data Preparation

4.1.1 Data collection theory

The SDK is an open-source and cross-platform library supported by multiple common programming languages [6]. Package *pyrealsense2* is used to access the official Intel RealSense SDK 2.0 in python.

Pyrealsense2 package supports streaming data from the LiDAR sensor and converting the 640*480 objects to a NumPy array of depth data stored as a variable *depth image*. Further, *openCV* library (stands for Open Source Computer Vision Library) converted the depth data into a color map to use existing computer vision algorithms. To prepare a training dataset, data are saved as images with people passing by and different positions at several locations, Bin entrance, train station, and lab. Example images are shown in Figure 4.1.

4.1.2 Data collection setup

Data are collected from four locations and divide into two categories regarding the light conditions. Different settings are applied to different locations. Figure 4.2, Figure 4.3 and Figure 4.4 show the real setup environment of data collection.

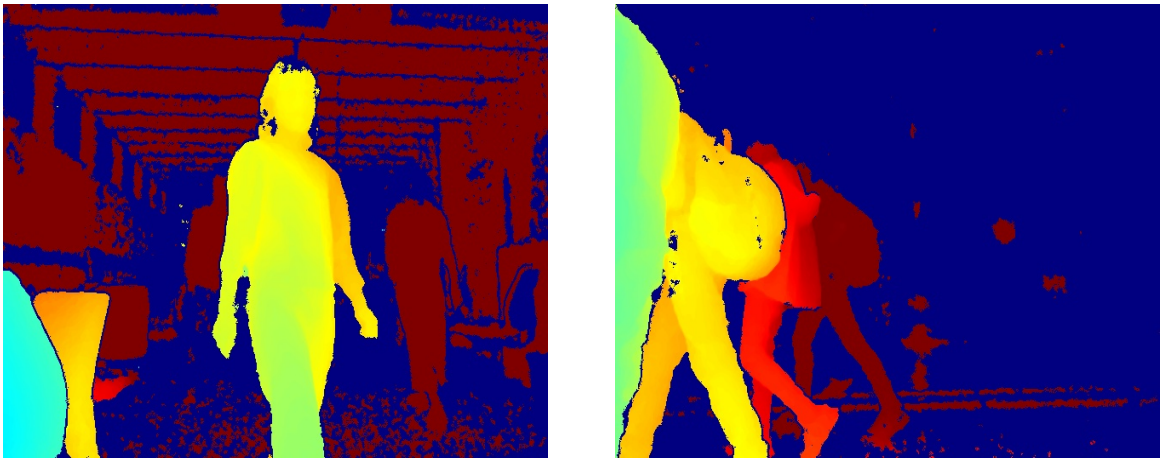


Figure 4.1: Example images of LiDAR collected

Daytime Condition with natural light.

- Lab: It is the most convenient location considering the light, but the number of people are limited. Different angles are considered and tested. *cf* Figure 4.2
- Train station: An ideal location to have more people and more light influences. LiDAR is settled underground. The farthest distance detected is around 3.5 meters. *cf* Figure 4.3
- Bin entrance: The LiDAR is set on a chair on the side of the entrance, facing people when they pass by. The distance is set as 2 meters. *cf* Figure 4.4

Night Without natural light.

- Mock condition at home: There is no ambient light, and a minimal number of people passing by. This scenario allows long-time data collections and more postures.

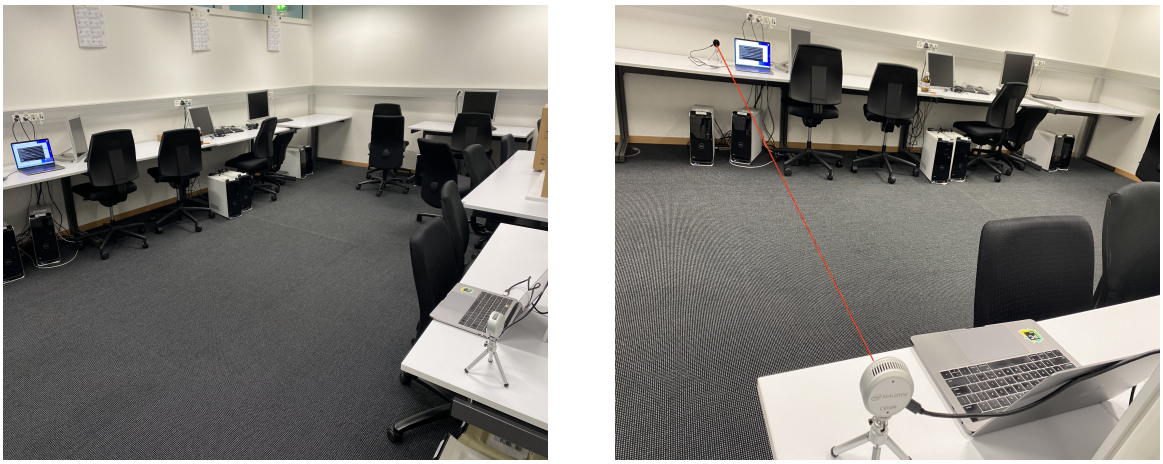


Figure 4.2: Data collecting at the lab

(a) Train station setup

(b) Train station setup distance

Figure 4.3: Data collecting at Train station

(a) Bin Entrance

(b) Bin entrance distance

Figure 4.4: Data collecting at Bin entrance

4.2 Object Detection

4.2.1 Image Labelling

Labeling [1]is an image annotation tool for generating text files with object location information that the YOLO algorithm can use [21]. To label images for customized object detection, in our case, a Person, we manually draw rectangles around the Person as closely as possible and set the class name as *person*, shown in Figure 4.5. We use the system timestamps as image names to avoid repeated names. The resulting data is a text file with the coordinate, x, and y of each object in the image and index of the class name.

[1]https://github.com/tzutalin/labelImg

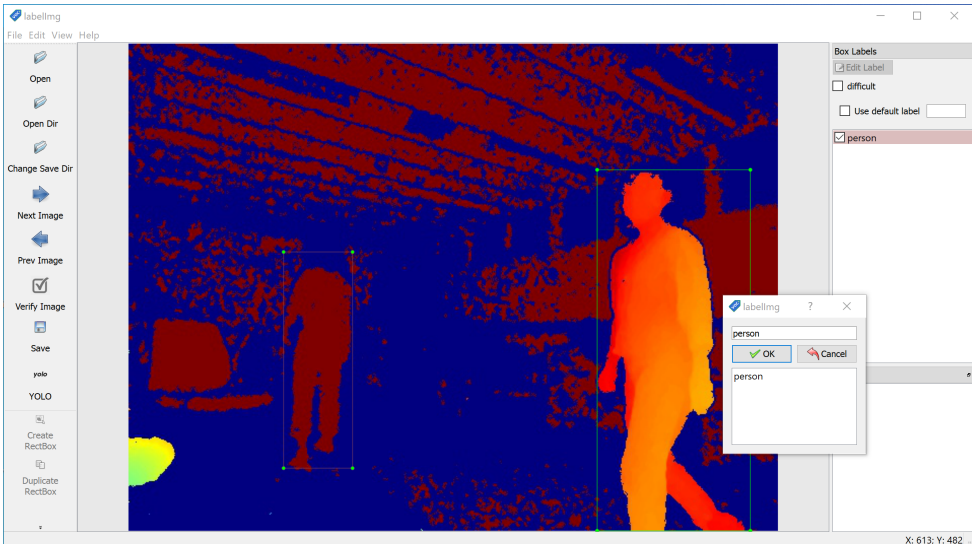


Figure 4.5: Labeling images with the tool *LabelImg* for customized object

4.2.2 YOLO for custom object detection

The color images used in detection are converted from depth data and applied with colormap, which is different from the normal RGB images. In other words, the images do not show details of the person and the pretrained detector can not recognize the person in this type of image. Thus, a customized object detector needs to be trained. The image together with the text file generated by *LabelImg* is our dataset for training. Data is split into Train and Test as well. With YOLOv3, a pre trained Darknet-53 model weights is used for the customized object detector in order to achieve a faster and more accurate training [19].

4.2.2.1 Colab setup and preparation

The training is processed with Google Colab GPU. We clone darknet from AlexeyAB's repository[2], adjust the Makefile to enable OPENCV and GPU for darknet, and then build darknet.

4.2.2.2 Train test data split

As mentioned in Section 4.1.2, 120 images are collected for each condition. Training and testing data were split with a ratio of 8:2.

To know how the model generalizes to different environments, we use different training and testing data combinations and compare the model loss. 9 models are compared by mixing the data from two conditions, as Table 4.1 shows.

[2]https://github.com/AlexeyAB/darknet

Train/Test	Daytime	Night	Mix
Daytime	Model1	Model2	Model3
Night	Model4	Model5	Model6
Mix	Model7	Model8	Model9

Table 4.1: Different combinations of training models based on data collection conditions.

4.2.2.3 Training

With labeled training and testing data prepared, several parameters are specified in configuration. Parameters [3] are suggested as follows: *batch = 64, subdivisions = 16, max_batches = 6000, steps = 4800, 5400, width = 416, height = 416. classes = 1* in the three YOLO layers and *filters = 18* in the three convolutional layers before the YOLO layers [19].

In the file *obj. Names*, only one class *person* contained. File *obj. Data* is created that includes information of a number of classes, the path for *train.txt*, *test.txt*, *obj. Names*, and a backup path for saving training weights throughout training. Finally, the Weights file is created that contains the weights of our trained custom object detection model.

4.2.2.4 Comparison of model Loss

Table 4.2 below shows the average loss of all 9 models and their comparisons. Training figures are attached below to show the trend of training loss through iterations.

Train/Test	Daytime	Night	Mix
Daytime	0.1899	0.1697	0.1856
Night	0.1025	0.1069	0.0861
Mix	0.1534	0.1610	0.1423

Table 4.2: Average loss comparisons over 9 models

4.3 Deep Sort Tracking

This section gives the detail of how to implement a Deep Sort tracker [4] based on the YOLOv3 detector trained before. Specifically, detector helps to detect people from a real time video and tracker works to track them until people are out of frame. It gives an ID and object center position of each detected person, which are used in the classification.

[3] https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial
[4] https://github.com/theAIGuysCode/yolov3_deepsort

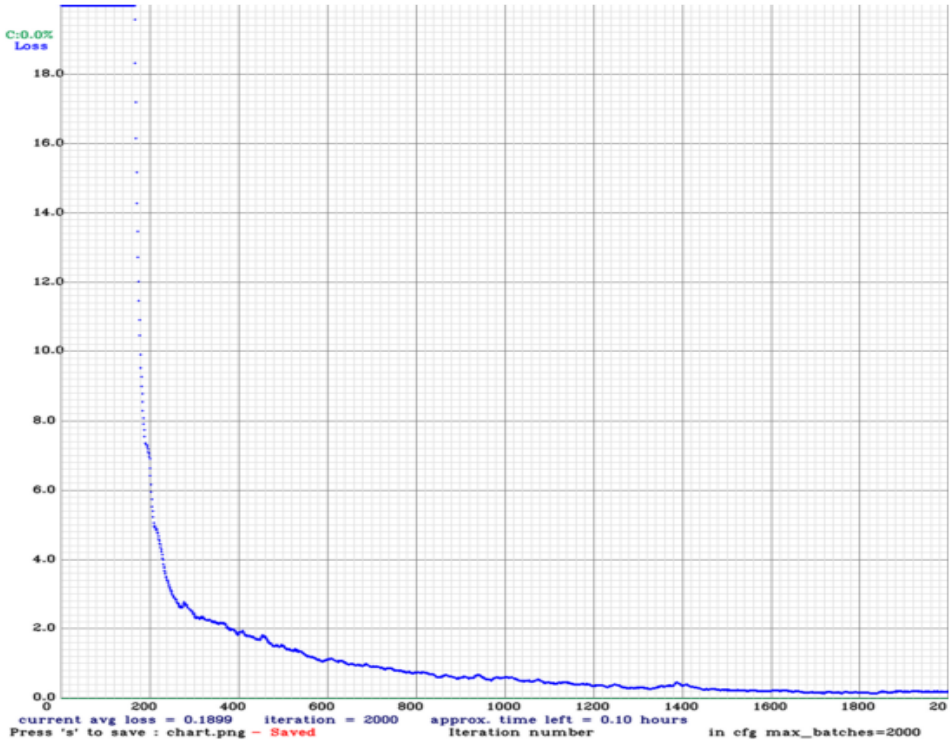


Figure 4.6: Model 1 Daytime training loss

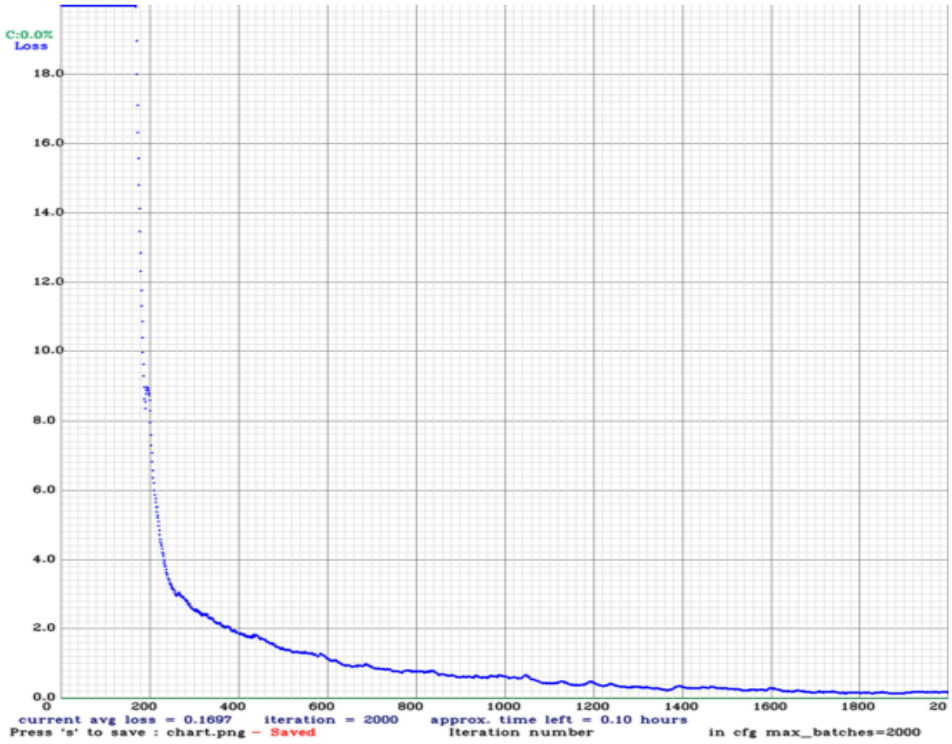


Figure 4.7: Model 2 Daytime and night combined model training loss

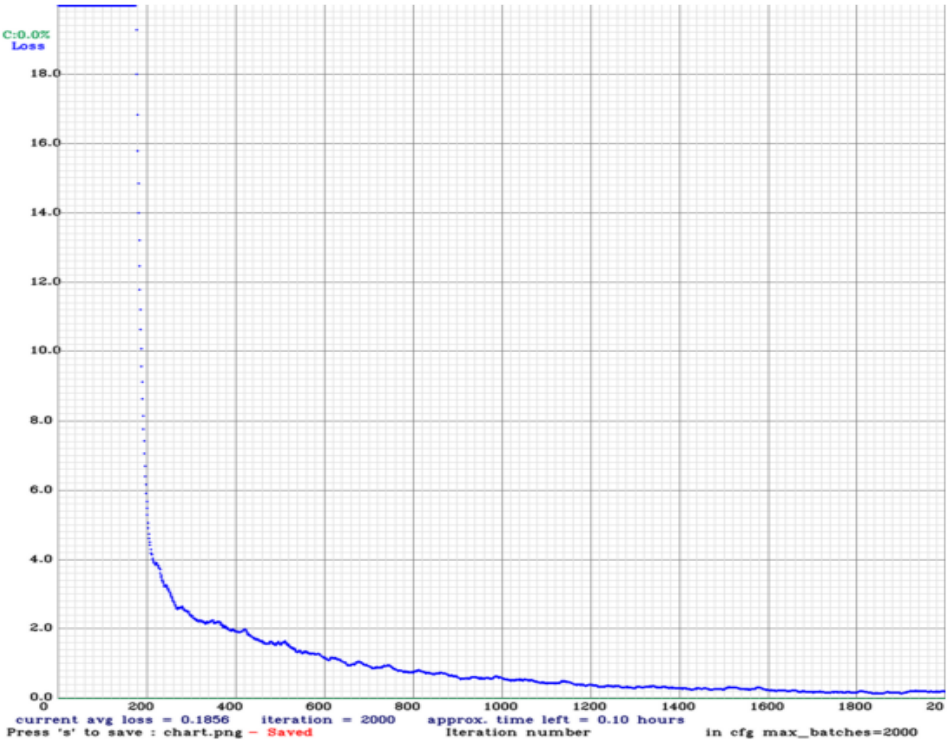


Figure 4.8: Model 3 Daytime and night combined model training loss

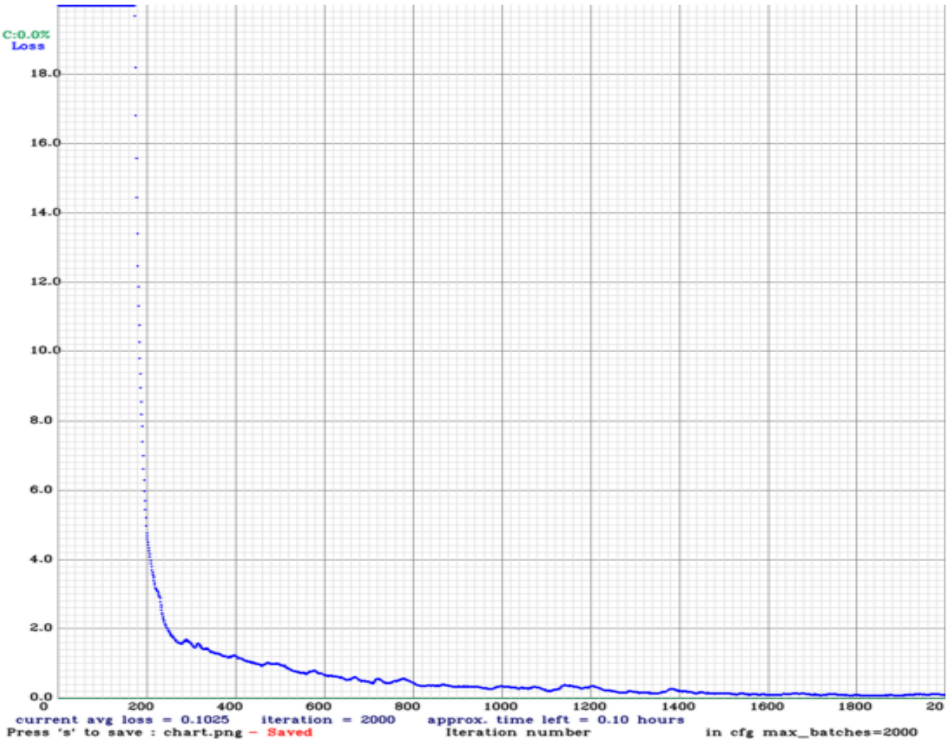


Figure 4.9: Model 4 Night and daytime combined model training loss

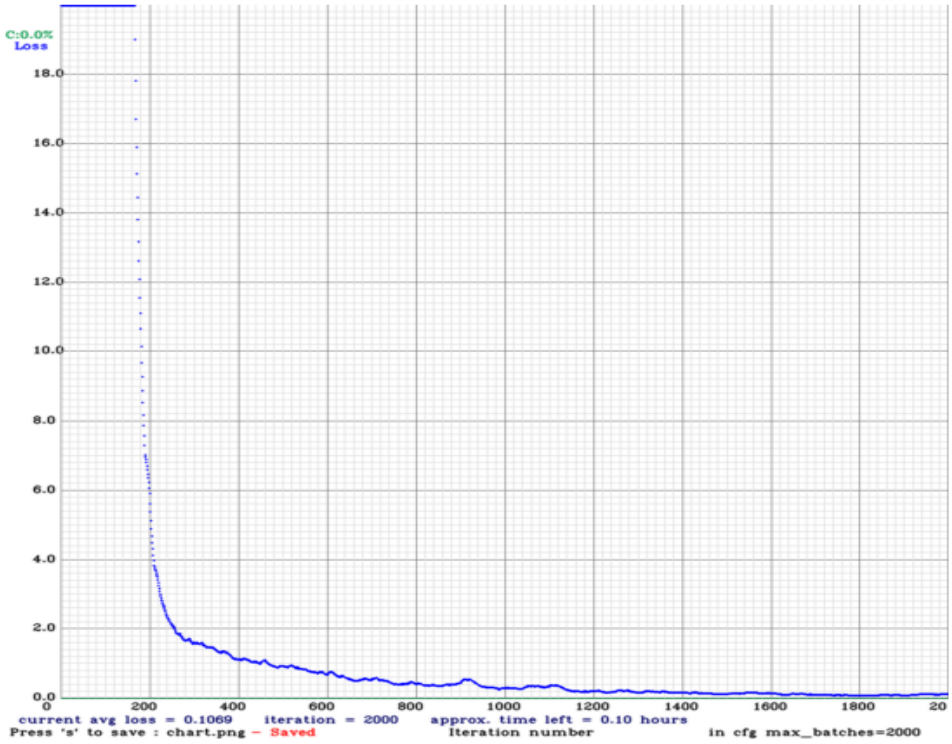


Figure 4.10: Model 5 Night and daytime combined model training loss

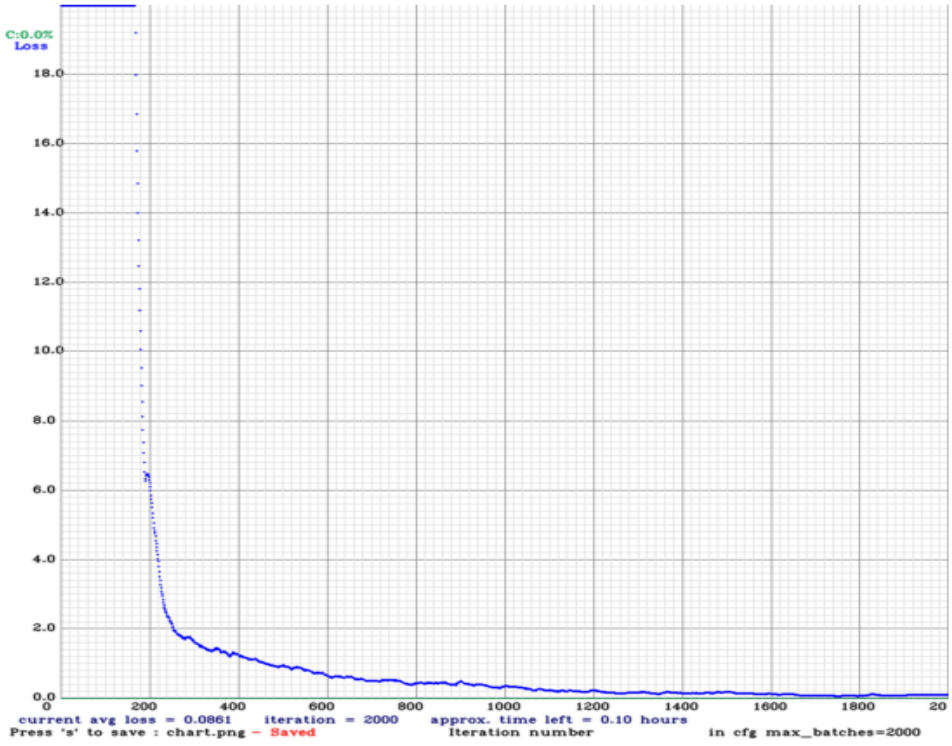


Figure 4.11: Model 6 Night model training loss

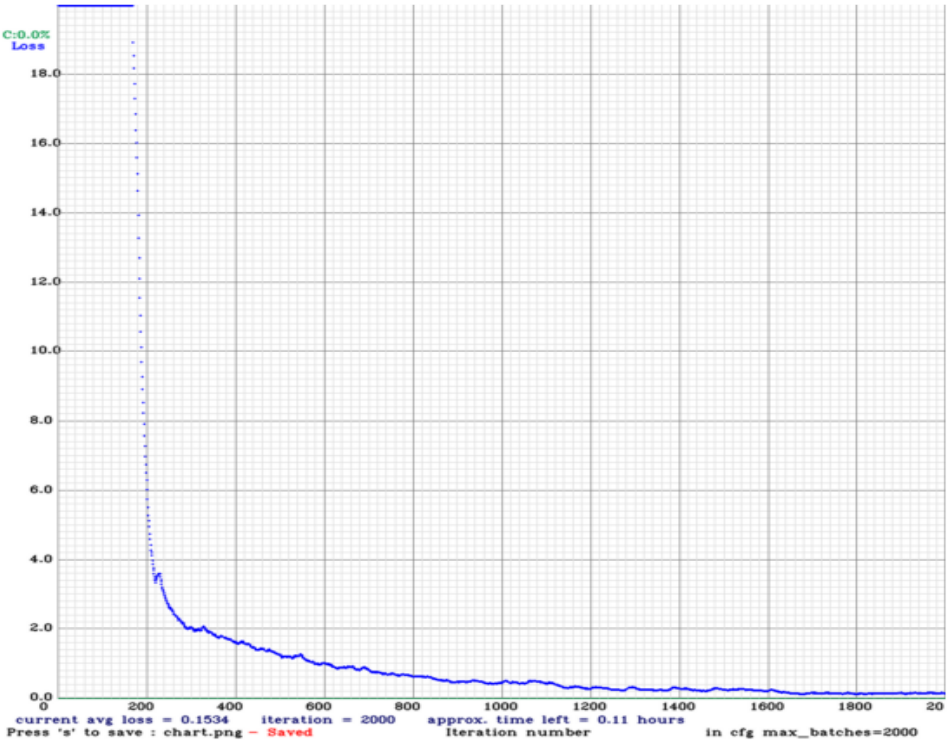


Figure 4.12: Model 7 Mixed and daytime combined model training loss

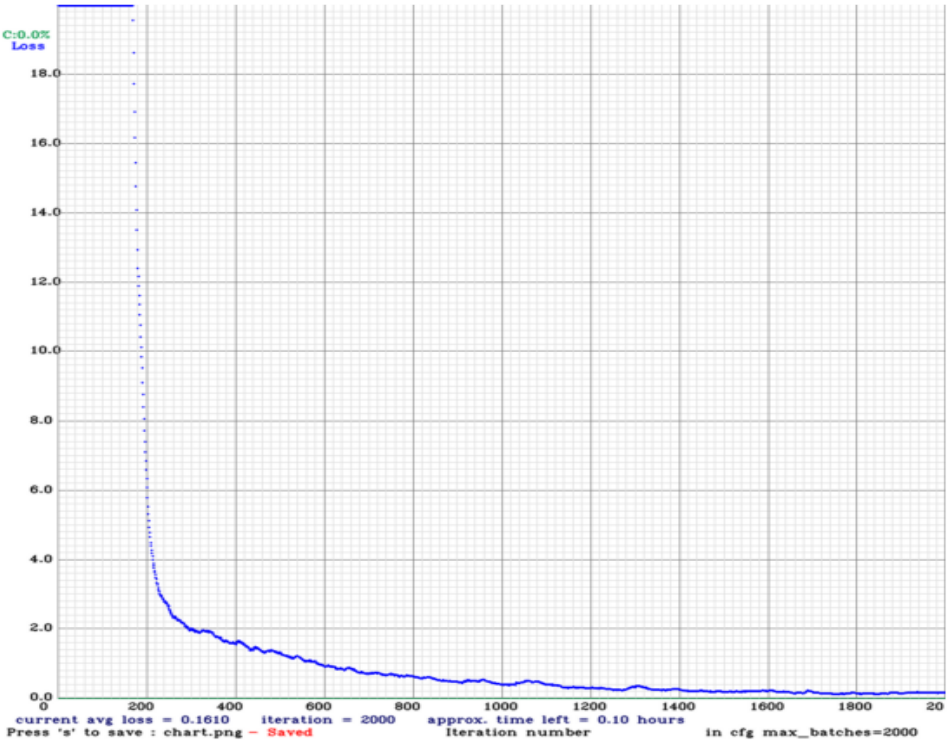


Figure 4.13: Model 8 Mixed and night combined model training loss

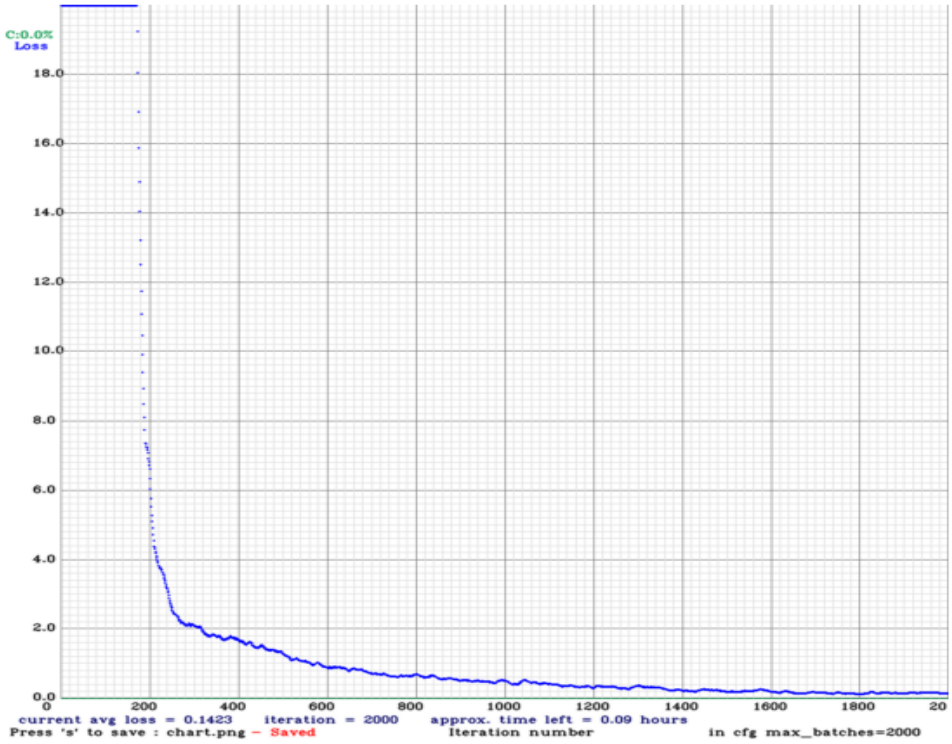


Figure 4.14: Model 9 Mixed combined model training loss

4.3.1 Setting

There are some functions that need to be defined to run the Deep Sort tracker. Recording the ID of each tracking box, the coordinate of each tracking box center, the distance from camera to the center point, and current number of people in the frame.

Function *yolo.predict(img_in)* yields boxes, scores, classes, nums. Using *track.to_tlbr* to get the predicted bounding box in format of (min x, min y, max x, max y). The center of detected object is the diagonal center of the bounding box, which can be calculated using: *cx = int((bbox[0] + bbox[2])/2)* and *cy = int((bbox[1] + bbox[3])/2)*.

In the file of *realsense_camera.py*, we returned the depth frame matrix minus depth, using method *rs.get_frame_stream()*; depth of the object center can then be obtained directly from the matrix using function *depth_mm = depth[cy, cx]*.

To run the Deep Sort tracker, converting the YOLOv3 weights into TensorFlow model files.[5] [20].

4.3.2 Running

To run the Deep Sort tracking algorithm in real time, command line as below: *"python object_tracker.py –output ./data/video/results.avi weights ./weights/YOLOV3-custom.tf num_classes 2 classes ./data/labels/obj.names"*.

[5]https://github.com/theAIGuysCode/YOLOV3_deepsort

4.4 Data Storage

The Figure 4.15 shows the setup step of InfuxDB, the file *config_cloud.ini* stores information that could connect to the InfluxDB cloud.

Function *write_api.write* is used to write data from package *influxdbclient*, and function *query_api.query_data_frame_stream* is used to query data from InfluxDB. Figure 4.16 shows the process to clean queried data directly from InfluxDB. When streaming data is collected from two LiDAR devices, it writes to the InfluxDB cloud. The information about detected person ID, position X and position Y, distance to camera, and a timestamp are queried later, and saved to local devices for further analysis.

```
bucket = 'lidar'
with InfluxDBClient.from_config_file('config_cloud.ini') as client:
    # delete
    delete_api = client.delete_api()
    start = "1970-01-01T00:00:00Z"
    stop = "2021-08-01T00:00:00Z"
    delete_api.delete(start, stop, None, bucket=bucket, org=client.org)
```

Figure 4.15: InfluxDB Setup

```
df_stream = query_api.query_data_frame_stream(query)
_dataFrames = list(df_stream)

if len(_dataFrames) == 0:
    df_required = pd.DataFrame(columns=[], index=None)
elif len(_dataFrames) == 1:
    df_required = _dataFrames[0]
    print("888")
    print(df_required)
    print(type(df_required))
else:
    df_requireds = _dataFrames
    # there are dataframes for multiple data!!!
    for df_required in df_requireds:
        if df_required.columns[-1] == 'depth_mm':
            df_required = df_required.drop(['result','table','_time','_start','_stop','_measurement'],axis=1)
            print(df_required.index)
            print(df_required)
            print(type(df_required))
            csv_data = df_required.to_csv('result_test.csv', index=False)
```

Figure 4.16: InfluxDB Query Data

4.5 Synchronization

4.5.1 Virtual Machine

As introduced in the related work in Section 2.4, a virtual machine is used to collect data from two LiDARs by running the developed method from phase 1. With the advantage of VM, no physical router is needed for connecting two LiDARs devices in the same network.

Also, by installing the InfluxDB on the VM directly, the query step is finished on the VM as well.

4.6 LiDAR Light Testing

In order to find the best setup of LiDAR, the app "LightMeter" is applied to test the factor of lighting in an indoor environment. In the lab, there are 4 types of light condition modes: all lights on; first half light on; second half lights on and all lights off.

With a phone camera facing the laptop, and starting the LiDAR streaming and visualizing, moving away the phone camera until LiDAR is unable to detect the object; and then recording the light measurement metrics with a picture. As shown in the Figure 4.17, the darker the condition is, the further LiDAR could detect; however, the distance does not change too much for all four conditions, approximately 30 centimeters difference. The furthest distance detected is around 0.5 meters when all lights turn off.

With this step of measurement, the detected distance are not be affected too much in an indoor environment. This measurement also explains the best training model loss is with the condition of night.

4.7 Testing Scenario Design

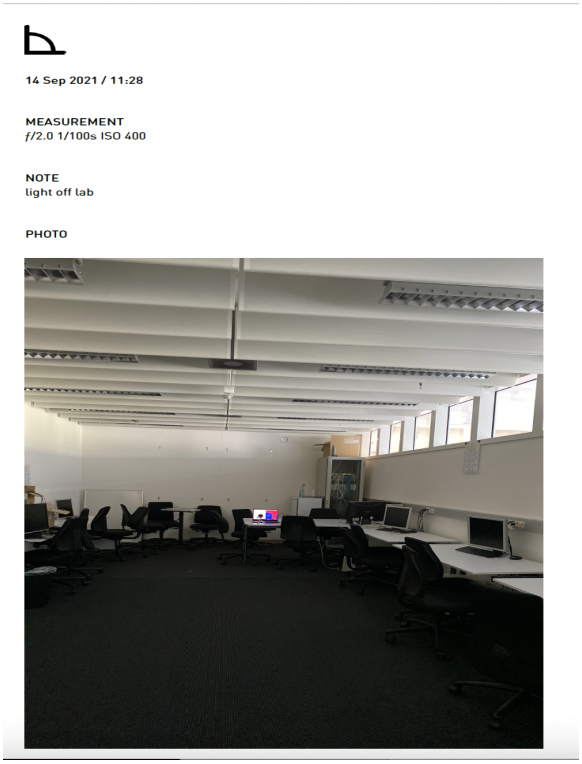
Due to the limitations of measurements in a real case, a scenario is made for testing the developed method with a mock shopping scenario. A product is chosen as the point of interest, in this case, it is the scooter. Testers need to pass through this scenario by performing as they are shopping, some of them are interested in the scooter by standing close to the scooter and staying longer; and some of them are not interested, thus passing by the scooter quicker. The data are trying to collect as many different user cases as possible.

A home basement location is chosen with complete darkness. Model 5 weights are used. Scenarios are designed as the Figure 4.18 shows. Two LiDARs are facing each other from 4.6 meters away, the object is located in the middle(scooter), 2.3 meters from 2 cameras.

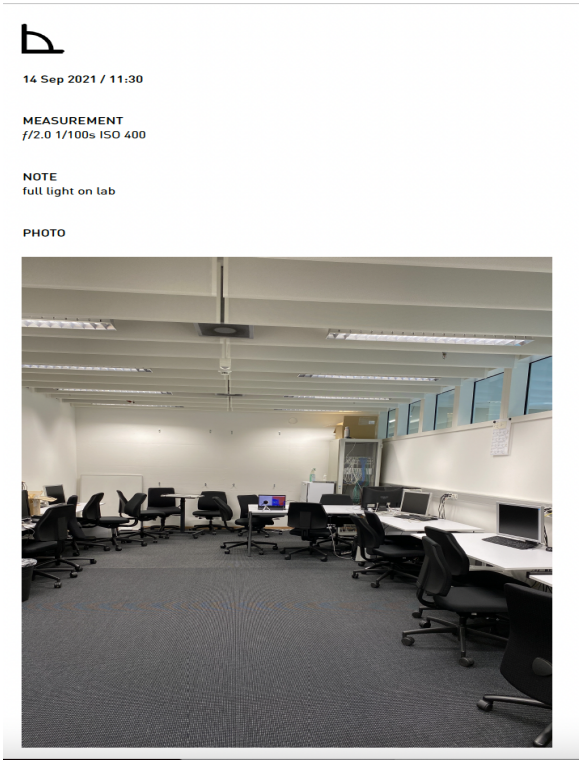
4.8 Classification

4.8.1 Data Preprocessing

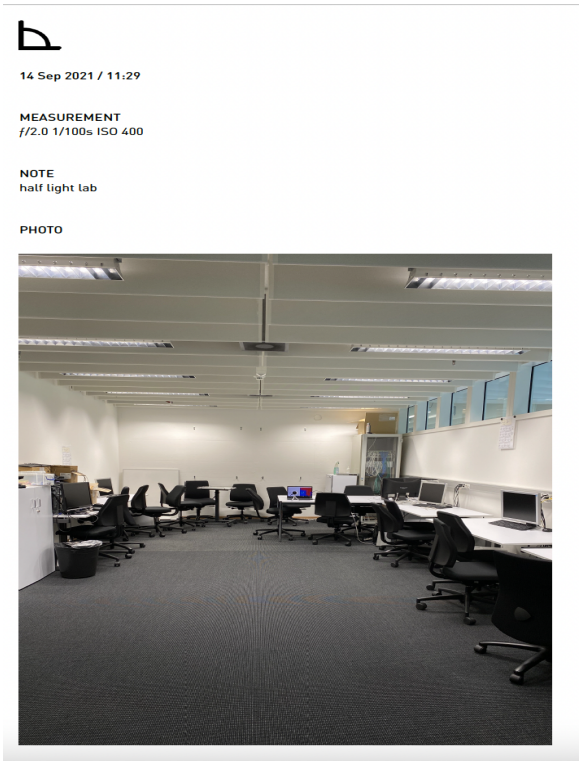
This sub-section includes how to clean the queried data from the designed scenario. Also, measurement metrics are defined in this section.



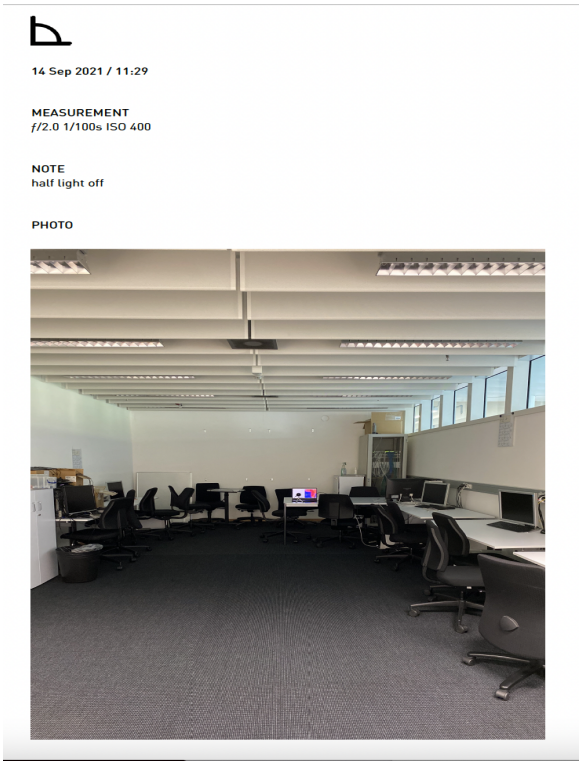
(a) Full lights off



(b) full lights on



(c) right half-lights off



(d) left half-lights off

Figure 4.17: Four lights condition testing in lab

Figure 4.18: Testing scenario setup

4.8.1.1 Data Cleaning

From the queried data of the VM, only one feature needs to be cleaned.Timestamps appear as strings in format *2022-01-19 14:51:28.156420* is convert into datetime object in the format of *%Y-%m-%d %H:%M:%S.%f*.

Outliers exist in depth measurements. Since the range of detection is only 9 meters in an ideal environment, any depth data above 9 meters(9000 mm) is an outlier. Some measurements only appear in one timestamp, having a duration of zero. Since a person cannot pass through the scenario setting within one frame, the zero duration data are also considered as outliers.

4.8.1.2 From Camera Frame to World Frame

This LiDAR has Depth field of view $70° \times 55°$ ($\pm 3°$. Camera front glass with -4.5mm and focal length is 1.88mm [6]. L515 camera outputs depth, which is the distance from the object to the image plane, as Figure 4.19.

The center of the detected object is recorded as (u, v) coordinates in the Image plane. There is ambiguity in projecting the points from image frame into world frame, but knowing the focal length and the depth help to disambiguate the points in the world frame, as Figure 4.20.

Two different coordinate systems are used for two cameras.In Figure 4.21, LiDAR2 is an example of mapping the 2D image coordinates to 3D world coordinates. The point of interest is defined as (0, 2) from top view. Only considering the horizontal distance between the detected object center and the point of interest because this tells how near the object is to the point of interest.

Here are two steps of how to convert from camera frame to the world frame:

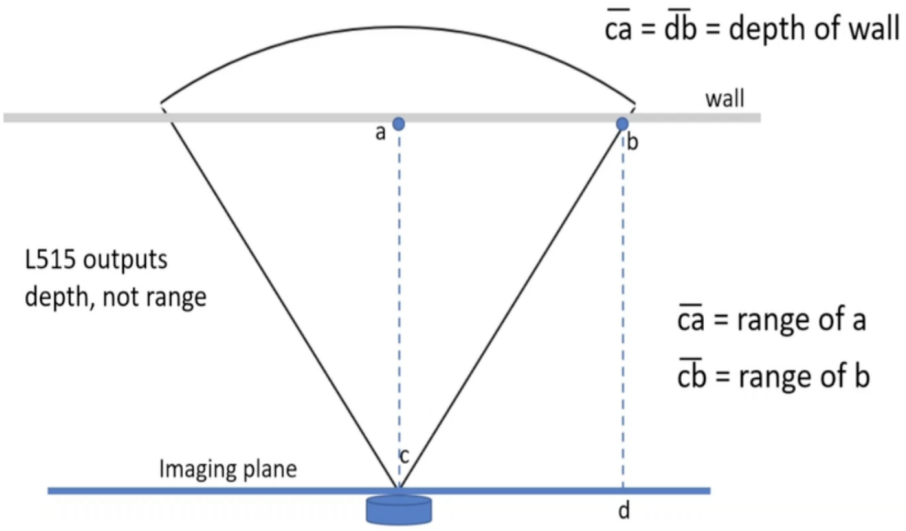


Figure 4.19: Converting Camera frame to world frame[6]

1. Conversion factor: Given the field of view of 70 degree, focal length of 1.88 mm, and image plane width of 640 pixels, we can calculate the focal length in pixels. The conversion factor alpha is defined as mm per pixel.

2. Distance from object center to point of interest: We already know the depth d from LiDAR, the x from image plane coordinates, and the focal length f in pixels. We convert d to pixels using the conversion factor from step 1. Using similar triangles, we can derive the X which is the horizontal distance from object center to the point of interest in the top view using:

$$X/u = (d + f)/f \tag{4.1}$$

   The X is presented in pixels. We convert it to mm using the conversion factor. Since we place the cameras 2 meters away from each other, we can easily calculate the euclidean distance between the object center and the point of interest (marked in blue in Figure 4.22) as $\sqrt{X^2 + (2 - d)^2}$.

4.8.2 Features

This section gives the list of features in the cleaned data that are used for classification.

4.8.2.1 Total number of people

This feature is defined as the number of distinct IDs in the frame within a defined time range. To count this value, data are grouped by ID.There are two cased to count.

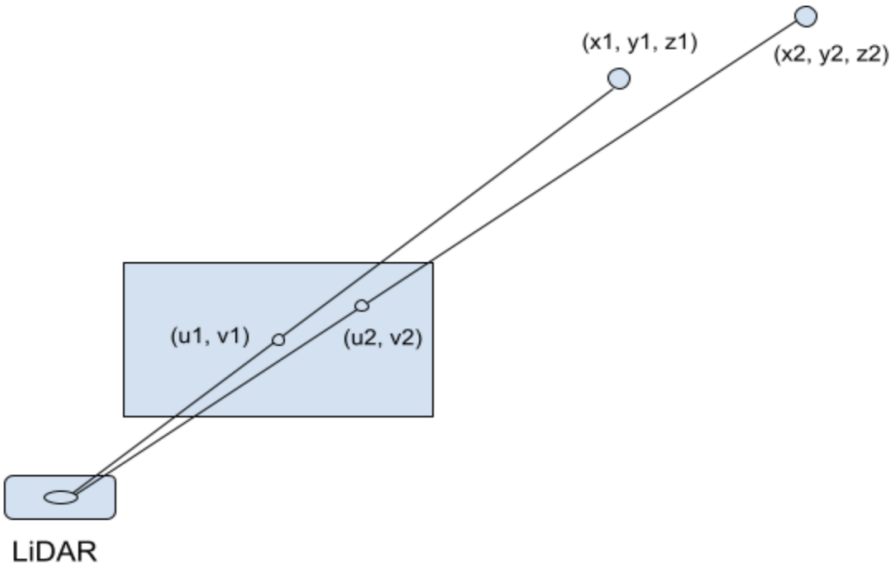


Figure 4.20: Define coordinates

1. Number of people at each timestamp: Since we cap the depth of each camera at 2 meters, there is no overlapping area. The number of people at each timestamp is the sum of counts in both cameras. There are no repetitive counts in this case.

2. Total number of people: Repetitive counts can occur in this case. People are assigned a different ID when going out of the frame and coming back. This also applies to people going from one camera field to the other one. One person would be assigned two IDs when crossing from one camera to the other camera field, and be counted as two people.

   To solve this problem, we merge the IDs by replacing the two IDs of the same person who is detected by both cameras with a one new ID. This can be done because whenever a person is crossing the border, it is within the detection range of both cameras, and is assigned two different IDs.

   A person can be assigned a third ID when going outside the range of one camera and coming back, and a fourth ID when the same happens for the other camera, so on and so forth. Therefore, we need to keep track of all the IDs that represent the same person and merge them as one ID to avoid repetitive count.

   We do this by analyzing neighboring points according to timestamp. We identify the samples as should be merged when they satisfy all three criterias 1) happen at the same time: the time difference between the neighboring points below 1 second. 2) at the border: the sum of depth from both cameras between 16000 to 18000mm. 3)taken by two cameras: one ID with the '_1' flag and one without. We build a dataframe called 'merge_ID' that collects all the IDs and their next IDs which should be merged. Then we build a graph *cf* Figure 5.1that has all the IDs as nodes and the neighboring relation as edges. All the IDs that belong to a connected graph are the same person that has been detected multiple times. In our original dataframe, we replace all the IDs that belong to the same graph with a new ID starting with 'm_' followed by an index number of that graph.

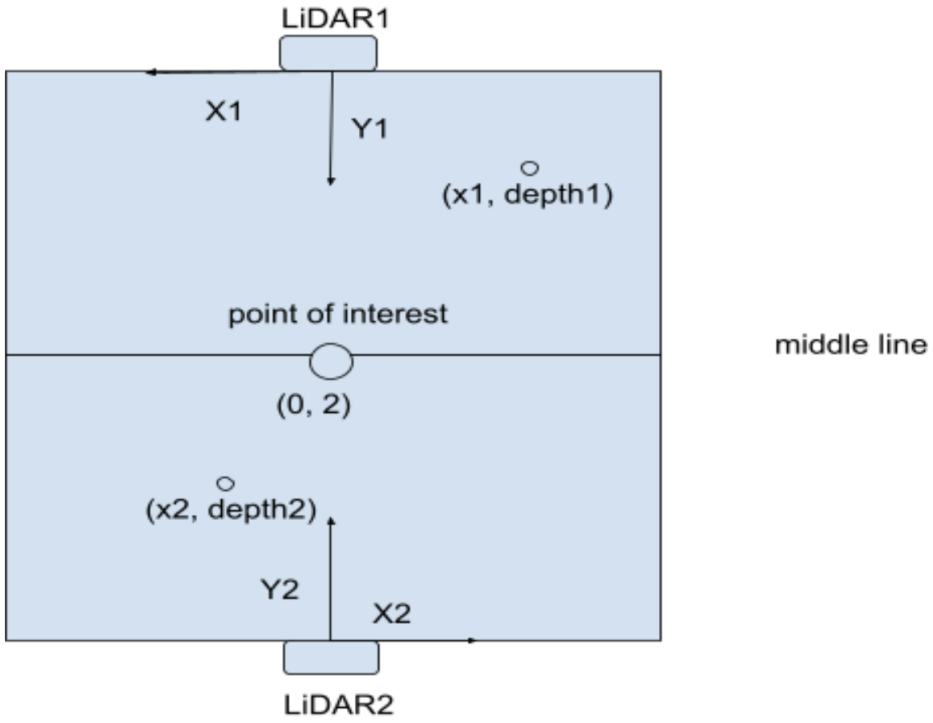


Figure 4.21: Designed figure of testing scenario

4.8.2.2 Distance to point of interest

Given horizontal field of view h_fov = 70, focal length f = 1.88, focal length in pixels $f\_p = 320/(tan((h\_fov/2) * pi/180))$, we can get conversion factor alpha $\alpha = f/f\_p$.

Using similar triangle, we have $(x\_mm/\alpha)/(x - 320) = (d + f)/f$. Therefore, distance in mm is $\sqrt{x\_mm^2 + (2000 - d)^2}$.

For each timestamp and corresponding coordinate we can calculate the distance to point of interest. To analyze the distance from each person to the point of interest, we can group by IDs, and use aggregation measurements like min, max, mean, and median to describe the distance from person to point of interest.

In code: *distance = df.groupby('ID')['dist_mm'].agg(['min', 'max','mean','median'])*

4.8.2.3 Duration

Duration is the time length of each ID appearing in the frame. In the data cleaning step, we get the timestamp in the right datetime format, so that the time difference can be calculated by the maximum of timestamp minus the minimum of timestamp.

To get the duration of each ID, we group data by ID, and convert the time difference into seconds for easier comparisons. In code:

duration = df.groupby('ID')['timestamp'].agg(['min', 'max'])

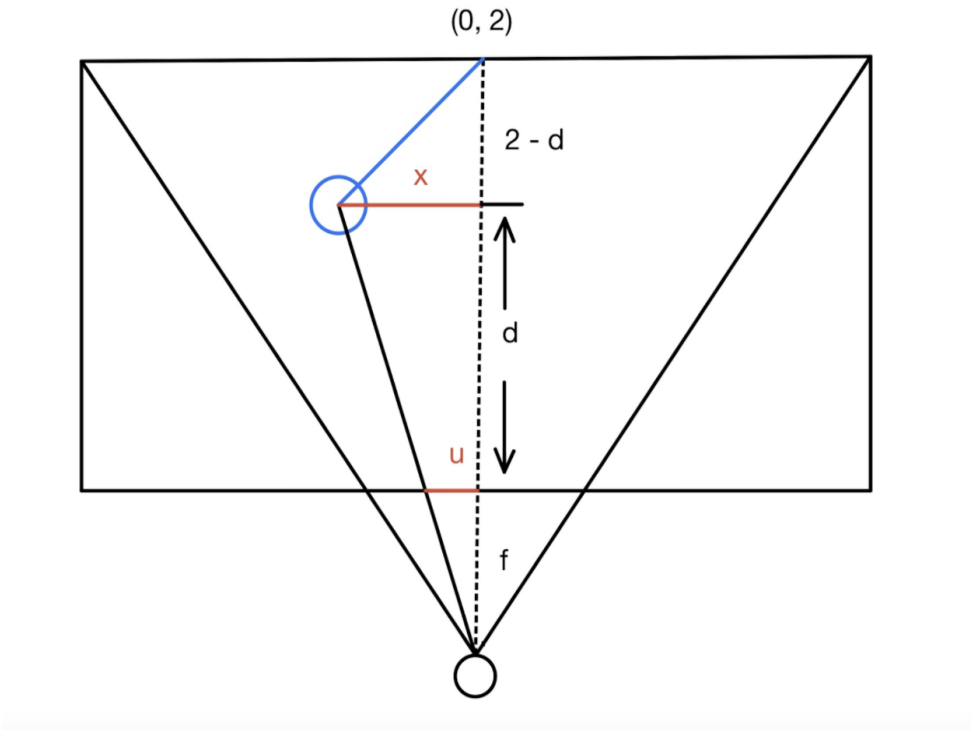


Figure 4.22: Calculating Distance

duration['duration'] = duration['max'] - duration['min']

duration['duration_s'] = duration['duration'].apply(lambda x: x.seconds)

4.8.3 User Segmentation

The Gaussian mixture clustering model is performed to segment users into different interest levels. Different groups of features can be taken into consideration when applying clustering algorithms.

Duration and distance are the most intuitive two features for measuring user interests, and can be plotted on a 2D image with high explain-ability. Several variations of duration and distance can be applied.

- **Duration and mean distance**: Mean is usually the default choice, but the distribution of distance might be skewed, making the mean distance biased.The mean distance can also be considered as duration weighted distance. This measurement is useful because time spent far away from the point of interest can not represent users' interest. Some users might be standing and chatting or looking at other points.

- **Duration and median of distance**: Using median, we are less sensitive to extreme values, but we may lose the track of small distances which we are interested in.

- **Duration and min distance**: We can track the closest distance between people and the point of interest, but if one person spends most of the time away from the point and pass by closely, it is likely to be clustered as someone with high interest.

4.8.4 Metrics

Metrics give an overview of the performance, and can lead to generation of actionable insights. There are some general properties of a good metric (STEDI) [4].

- **Sensitivity**: can move, detect a statistically-significant change. A/A experiment. Robustness.
- **Trustworthiness**: correct, avoid selection bias.
- **Efficiency**: easy to compute at scale, low cost of measurement.
- **Debuggability and Action-ability**: debug why.
- **Interpretability and Directionality**: understand, movement is good or bad.

We can encourage users' behavioral change by intervening in the system, in order to optimize the metrics. Optimizations can be done manually, automatically, or by combining both. The manual optimization refers to the Hypothesis-Experiment-Evaluation Cycle, which is widely known in internet companies as A/B test, and in academia as controlled experiments[8].

Below are metrics that we developed for the LiDAR counting application, focusing on measuring user number and user interest by measuring duration and distance.

- **Total user number**. Total number of distinct users appeared in the scene.
- **Accumulated interested user number**. Total number of distinct users that have spent more than 10 seconds within 0.5 meter range of the point of interest.
- **Percentage of interested users**. It is calculated by the Interested users / total users. Measuring how attractive the object is to the group of visited users. This is especially useful when we have multiple points of interest placing in different places that might have different user flow.

  This metric is not affected by the total user number. For example, a product is visited by 10 users, 10 interested is considered to be more attractive than an object visited by 100 users but only 15 users are interested.
- **Total time spent on an interesting object**. When the distance between the user and the point of interest is below 0.5 meter and the time spent is more than 3 seconds, the time spent is counted as time spent on the interested object.

- **Variation of interest**. Variation of user flow calculated on interested users. It is a product-centered metrics, which gives insights about which product is more likely to be interested, even more likely to be bought and which does not.

- **Variation of user flow**. Divide the entire time into windows of 10 seconds. The number of people that have appeared in each 10-second window is one record. The variation of user flow is the variance of the record. A high total user number and high variation of user flow might be caused by visitor groups. A high total user number and a low variation indicates a stable user flow. This metric gives insights about the venue instead of products. For instance, it can be used to compare two retail stores at different locations, etc.

Chapter 5

Evaluation

This chapter presents the result of clustering from data that the LiCounter method developed. Section 5.1 gives the logic behind clustering, as well as visualization. Section 5.2 discusses the chosen model for tracking, designed scenario setups, and evaluates projects requirements are satisfied. Section 5.3 lists the limitations.

5.1 Results

5.1.1 Raw Data

First, there are different possible interpretations from the raw data. It has column names *ID*, *position_x position_y*, *Timenow*, *depth_mm*. Three levels of interest are listed below:

1. Level of 'Not interested': Table 5.1 is an example of people who stay 1 meter away from the point of interest for 10 seconds. The last column is the distance to the camera in mm. Since the depth measurement is inaccurate, and is around 4 times larger than the actual depth, 4000 mm means about 1 meter away from the camera.

   Table 5.2 is an example of people passing by near the point of interest.The person is around 1.75 meters from the camera, 0.25 meter to the point of interest, and simultaneously detected by two cameras.

2. Level of 'Interested': This is an example of people staying near the point of interest for more than 10 seconds. Notice that when the two cameras are both detecting people, they can be the same person or different persons. It is possible to infer the simultaneous detection case by adding up the depth in two cameras. If they add up to around 16000mm, we consider it as detecting the same person. Below is an example of simultaneously detecting two people because the depths add up to only around 14000 mm, *cf* 5.3.

3. Unclear level: People staying within 1 meter for long time period: *cf* Table 5.4. User 16 stays for 25 seconds, at 0.75 meter from the point of interest. Within 0.5 meter but short time: User 18 stayed within 0.5 meter, for 8 seconds. *cf* Table 5.5.

ID	Position_x	Position_y	Timenow	depth_mm
33	207	265	20220126 18:11:17.695467	4377
33	201	265	20220126 18:11:19.558252	4454
33	205	264	20220126 18:11:21.314639	4400
33	193	265	20220126 18:11:23.314639	4400
33	149	268	20220126 18:11:25.181597	3328
33	133	269	20220126 18:11:27.028346	2925

Table 5.1: Raw data of people stay less than 10 seconds

5.1.2 Merged ID

To avoid repetitive counts, IDs of people that are simultaneously detected by both cameras are merged according to section 4.8.2.1. Below Figure 5.1 shows the graph of merging 43 IDs into 7 new IDs.

5.1.3 Aggregated data

Duration and distance to point of interest can be aggregated according to ID. By seeing the duration and distance data together, we can tell whether a person is interested or not. User 10_1, 26196_1, and 26340_1 have a duration of 0 second. Since we assume a person cannot pass through the frame within 1 second, we consider them as outliers. Examples are given in the following tables. *cf* Table 5.6

Interest
- User 12 stays for 17 seconds, is as close as 0.1 meter to the point of interest, and always within 1 meter.
- User 1_1 stays for 10 seconds, with a mean distance below 1 meter, and the minimum distance of 0.5 meter.
- User m_4 stays for the longest duration, almost 5 minutes, with minimum distance below 0.2 meter, and on average below 1 meter.

Not Interest
- User 26318 stays for 124 seconds, which is a relatively long time. The distance is however mostly above 1 meter.
- User 17_1 is also always more than 1 meter away, staying for 13 seconds.

Unclear
- User 26349_1 stayed for 36 seconds, but mostly more than 1 meter to the point of interest.

ID	Position_x	Position_y	Timenow	depth_mm
84_1	673	192	2022-01-26 17:43:47.716286	10381
58	330	254	2022-01-26 17:43:48.912462	7266
84_1	678	193	2022-01-26 17:43:50.284571	10537
84_1	679	192	2022-01-26 17:43:50.782269	10565
58	324	262	2022-01-26 17: 43:52.304258	7103
84_1	679	187	2022-01-26 17:43:52.304258	10500
58	319	265	2022-01-26 17:43:53.819235	7041
84_1	679	191	2022-01-26 17:43:53.838248	10519
58	318	269	2022-01-26 17:43:54.743203	7025
84_1	679	196	2022-01-26 17:43:55.619163	10493
58	319	267	2022-01-26 17:43:56.703451	7096

Table 5.2: Raw data of people passing by

5.1.4 Clustering

We compare the duration to the different aggregated measures of distance. 16 distinct persons are detected in this testing scenario.

1. Duration and mean distance

   Mean distance is distance weighted by duration since the longer duration the more appearance, thus more weight. Only two clusters are identified. This clustering result ignores the distance to the point of interest as a horizontal line at around 125 seconds can well separate the two clusters. *cf* Figure 5.2

2. Duration and median distance

   Three clusters are identified in Figure 5.3. Cluster 1 (yellow) with duration below 60 seconds, regardless of the distance, Cluster 2 (purple) with duration above 150 seconds and distance above 1.2 meter, and cluster 3 (blue) with duration above 200 seconds, distance below 1 meter.

   Cluster 1 is a mix of interested and uninterested people. Those who stay below 10 seconds, no matter how long the distance, are not likely to be interested. Those who stay longer time with distance below 1 meter are likely to be interested. This also fits our description of an unclear group.

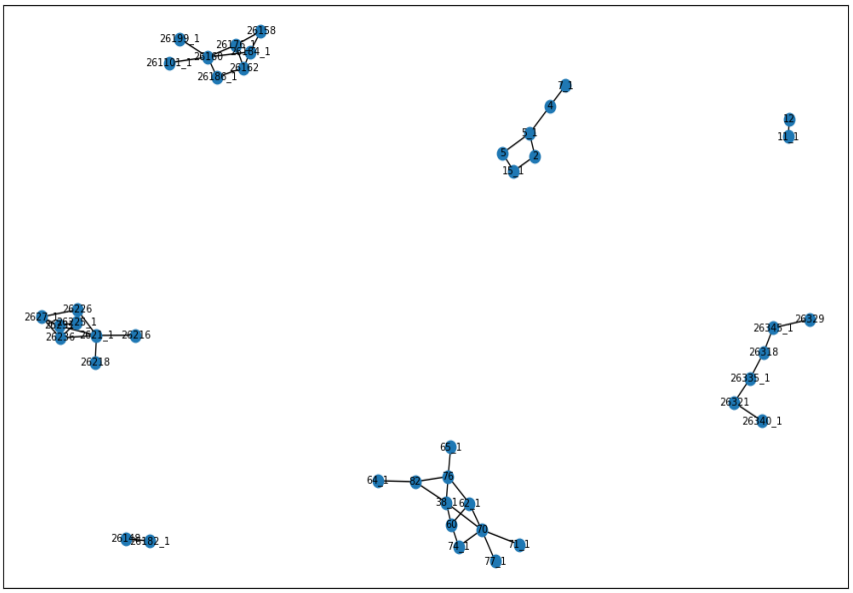


Figure 5.1: Graph of Merged ID

Cluster 2 represents people who stay long but far away from the point of interest. We consider them as not interested since 1.2 meters is too far to see the point of interest clearly. People might be just staying somewhere and chatting.

Cluster 3 is more likely to be an interested group, since they stay very long, and relatively near (below 1 meter).

3. Duration and min distance

   Min distance shows more clearly the group of people who are not interested. Cluster 2, the yellow cluster in Figure 5.4, shows people who have never been within 0.9 meter to the point of interest.

5.2 Discussion

5.2.1 Model Training and Selection

Nine models are trained for YOLOv3 detector. Training using daytime dataset always performs worse than training using night time dataset, and the mixed one is in between. This can be explained by the influence of ambient light at daytime that greatly reduces image quality, which makes it harder for training the model. The result of comparison suggests using less ambient light scenarios for training.

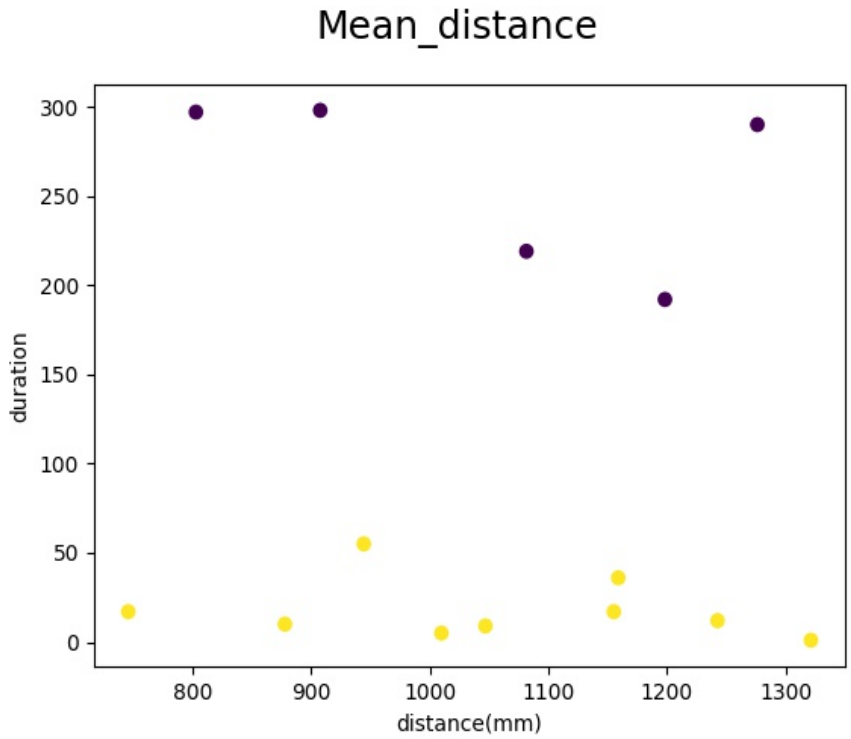


Figure 5.2: Duration and mean distance

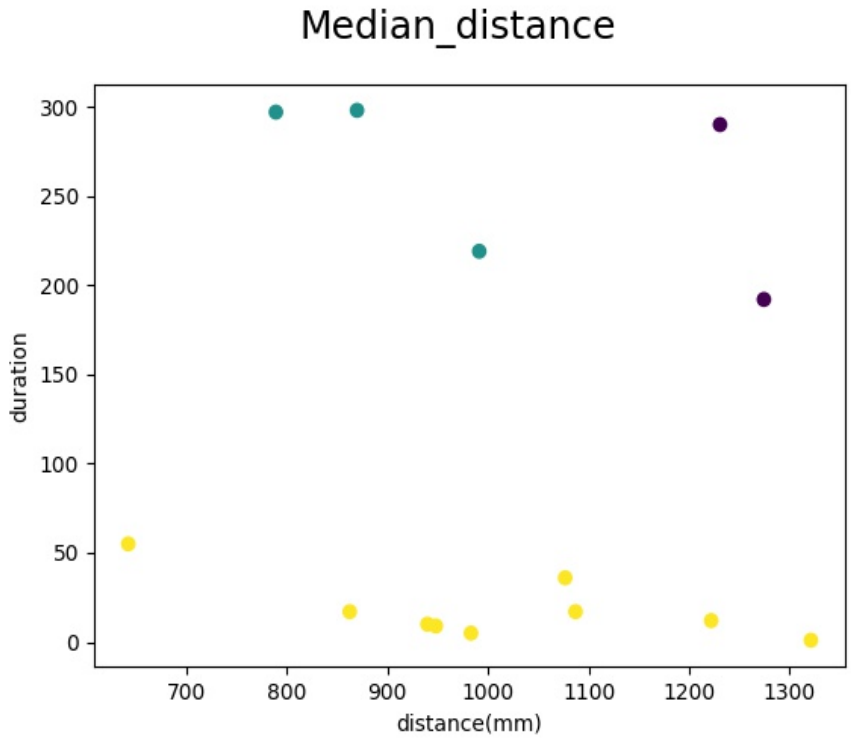


Figure 5.3: Duration and median distance

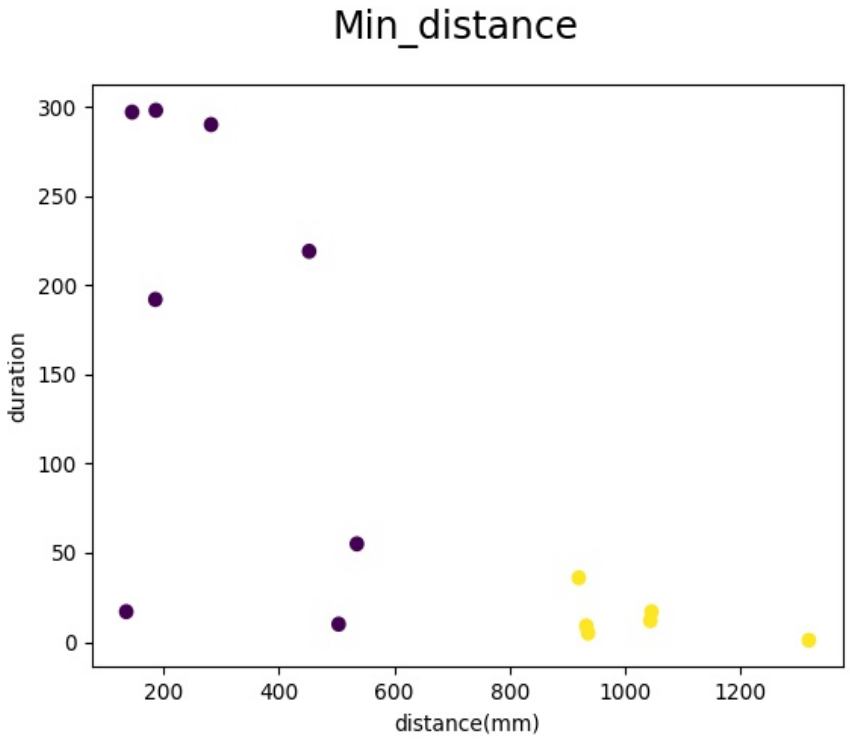


Figure 5.4: Duration and min distance

Night training scenario has the lowest loss, thus we use the corresponding weights for our mocking shopping scenario at the home basement, where it does not have any light influences. Testing image of model 5 is illustrated in Figure 5.5, it has a very accurate detection.

5.2.2 Scenario Design

In the designed scenario, the goal is to collect data from different people who behave differently. Specifically, people who are Staying within 0.5 meter to the scooter for more than 10 seconds means they are interested. People who are staying more than 1 meter away from the scooter for an arbitrary amount of time means they are not interested. People who are staying within 0.5 meter to the scooter, but for less than 10 seconds are defined as random actions. Doing any random stay to enrich the variety in the dataset.

The ground is flat enough to ensure that two LiDAR cameras are at the same height, and facing each other to both detect the people. Since the space is limited at the evaluation scenario, only one point of interest the scooter is placed in the test, which cannot represent a shopping case. There is also a limitation with the number of testers. Only two testers are in the testing scenario, and try to simulate a scenario of 40+ people by walking out of the frame and coming back again. However, the Deep Sort algorithm can sometimes identify the tester as the same person that has been tracked and assign the same ID. If there are more than 40 different people testing, is it not likely for Deep Sort to assign the same ID to different people. If more objects could be placed and more testers could join, the testing scenario will better represent a real shopping scenario

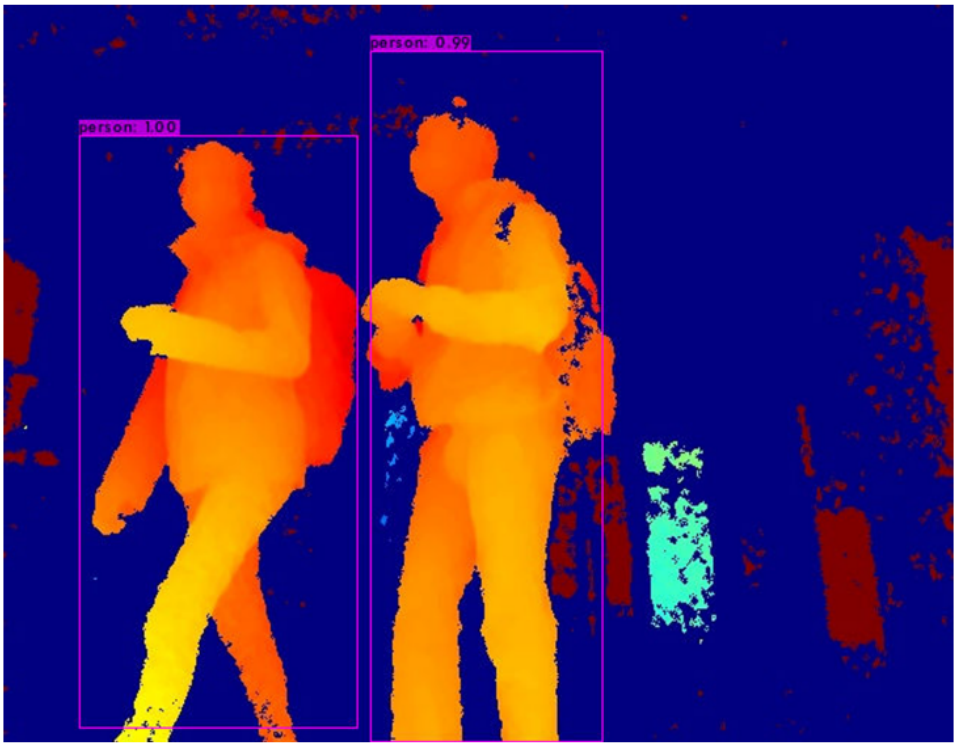


Figure 5.5: Testing Image from Model5

5.2.3 Data Analysis

In the step of analyzing data, a clustering algorithm is demonstrated to segment users into three groups based on duration and distance. During the analysis, the challenge is the repetition of counting when people are crossing the boundary of two cameras. The solution used for counting people at each timestamp is adding the number of people detected in each camera; and counting the total number of people is done by subtracting repetitive counts from total number of users.

Due to data collection limitations, the data set is not large enough, which may not be ideal. Only one scenario is presented, so some metrics that we designed for multi scenarios cannot be used.

5.2.4 Evaluation of Requirements

This subsection evaluates whether the results satisfied the requirements for this project from Section 3.1.

- Color images for training model. The images collected from LiDAR in the Phase 1 are valid, and objects have clear edges in each picture, which is easy for the detector.
- Valid Localization method. The methods developed for localization are realized with YoLoV3 and Deep Sort combined. The method is implemented whenever LiDAR

starts to capture images, it detects and tracks people, and writes the timestamped data to the InfluxDB cloud.

- Synchronization and data storage. The two parts of synchronization include synchronizing the time of two LiDARs to ensure there are no time latency between each other; the other part is the synchronization of detected users in classification, this is achieved by analyzing the data based on different scenarios.

5.3 Limitations

The results can reflect the limitation of the solution for this project, which can be used for further development.

- Inaccurate LiDAR. The detection of depth from LiDAR is inaccurate, and this type of camera does not offer any tool for calibration so far. The inaccuracy in depth measurement is the limitation of the device that cannot be overcome by data analysis methods currently.
- Light Measurement. The detected distance based on light is measured from an app on mobile phone, which only measures the lighting condition indoors. It is not professional to measure the influence from daylight. If it can be solved, the measurements and set up of scenarios can be more accurate.
- Limited Testing Space. Due to the coronavirus, it is not easy for lots of people gathered together to test. Thus, we lack data to represent the real case. Also, the lack of an adequate physical space to implement the camera and evaluate the method.
- Limited Data. Due to the lack of data, we are unable to calculate the metrics as mentioned. For example, we do not have the data from different venues and points of interest, which is unable to get results about percentage of interested users, variation of user flow and variation of interest.
- Computation limitation. Data gathered is collected by running CPU on the laptop. It is slow and FPS does not achieve a high level, some people passing by may not be entirely captured.

ID	Position_x	Position_y	Timenow	depth_mm
86_1	462	237	2022-01-26 17:46:04.267133	7957
86_1	476	234	2022-01-26 17:46:05.924858	7879
60	606	269	2022-01-26 17:46:06.175569	6474
86_1	475	239	2022-01-26 17:46:07.487082	7844
60	586	265	2022-01-26 17:46:08.151662	6293
86_1	472	242	2022-01-26 17:46:01.062778	7854
60	628	263	2022-01-26 17:46:10.059983	6024
86_1	464	244	2022-01-26 17:46:10.603107	7888
60	643	261	2022-01-26 17:46: 12.12.049374	6207
86_1	473	242	2022-01-26 17:46:12.177930	7898
86_1	472	239	2022-01-26 17: 46:13.757630	7850
60	646	262	2022-01-26 17:46:14.011700	9012
86_1	471	240	2022-01-26 17:46:15.373585	7865
60	671	264	2022-01-26 17:46:16.007585	6188
86_1	472	246	2022-01-26 17:46:16.917737	7857
60	708	268	2022-01-26 17:46:17.914525	5430

Table 5.3: Raw data of people interested

ID	Position_x	Position_y	Timenow	depth_mm
16	274	269	2022-01-26 17:53:50.628949	5265
1_1	167	269	2022-01-26 17:53:50.798043	9659
1_1	164	269	2022-01-26 17:53:52.294529	9537
16	302	273	2022-01-26 17:53:52.567748	5343
16	137	273	2022-01-26 17:54:11.725321	5286
1_1	128	269	2022-01-26 17:54:12.522806	8666
16	130	277	2022-01-26 17:54:13.618995	5329
1_1	128	269	2022-01-26 17:54:14.046111	8686
16	165	272	2022-01-26 17:54:15.535175	3114

Table 5.4: Raw data of people with unclear behaviors 1

ID	Position_x	Position_y	Timenow	depth_mm
18	451	322	2022-01-26 17:53:14.572150	7762
1_1	159	267	2022-01-26 17:53:15.982252	9403
18	454	325	2022-01-26 17:5316.467847	7766
1_1	134	267	2022-01-26 17:53:17.479598	8877
18	458	327	2022-01-26 17:53:18.378997	7787
1_1	151	268	2022-01-26 17:53:18.999464	9245
18	464	332	2022-01-26 17:53:20.282187	7772
1_1	157	268	2022-01-26 17:53:20.518660	9386
1_1	136	268	2022-01-26 17:53:22.005160	8915
18	456	332	2022-01-26 17:53:22.184480	5535

Table 5.5: Raw data of people with unclear behaviors 2

ID	duration_s	min_distance	max_distance	mean_distance	median_distance
10_1	0	333.548	333.548	333.548	333.548
12	17	135.754	986.894	746.168	862.349
17_1	12	1043.73	1696.26	1242.42	1221.86
1_1	10	503.734	1195.82	878.085	939.682
21_1	55	535.317	2000	944.518	642.119
26148	5	935.945	1137.72	1009.91	982.993
26196_1	0	2000	2000	2000	2000
26318	124	969.066	2000	1426.55	1420.23
26333	9	932.863	1303.73	1046.97	948.048
26340_1	0	2000	2000	2000	2000
26349_1	36	919.834	2000	1158.82	1076.66
27_1	17	1045.88	1406.73	1155.09	1086.91
29_1	1	1318.52	1323.64	1321.08	1321.08
62_1	197	1202.48	2149.75	1361.29	1224.29
71_1	6	986.309	1127.39	1065.69	1072.92
m_0	192	186.17	2000	1198.11	1274.3
m_1	290	282.726	2040.44	1269.23	1267.13
m_2	295	424.273	2000	886.104	799.889
m_3	124	145.922	1029.47	608.459	714.834
m_4	298	187.018	2000	907.82	869.639
m_5	94	595.206	1011.2	886.911	939.012
m_6	94	452.63	2000	1011.86	990.608

Table 5.6: Table of aggregated data

Chapter 6

Final Considerations

6.1 Summary

In this project, A method LiCounter is presented, which is to detect, track, and count people within an area. Yolo version 3 is the algorithm used for detection, and Deep Sort is used for tracking. Further, a dataset is collected based on the method developed, and classification is used to categorize people who are near the region of interest or not. Gaussian mixture clustering method is used for user segmentation based on defined metrics measurement, including distance to the object, duration time of staying. The returned results are a visualization of classification for each user.

6.2 Conclusions

The results of this project provides details for commercial companies about a general view of user flow and the number of users for a period of time. Especially in a retail store, companies can know which products are more likely to be bought and which are not, where should the product be placed to attract consumers, and other insights.

The LiCounter approach is used typically for customized objects and indoor environments. The advantage of using LiDAR is straightforward, which can preserve passengers' privacy and not disturb them. However, LiDAR is very sensitive to daylight conditions; the darker the condition is, the farther it can detect. Based on light testing, all data were chosen to collect in a complete dark condition without daylight influences, including data for training Yolo tracker and testing scenarios. However, the limitations of the LiDAR and method developed both existed, and should be reconsidered to apply them in a real case.

6.3 Future Work

This project still can be further developed, and there are some future works suggested.

- Computation power. the tracker could work better with GPU for each LiDAR camera. The FPS could be improved to 14 and more data could be collected.
- LiDAR Measurement. Due to the inaccurate measurement of Intel RealSense L515 LiDAR, other devices can be considered, which may have more accurate measurement.
- Light Testing Tool: Professional light measurement tool can be used for more accurate measurement. The data of LiDAR sensitivity are helpful to identify the placement of devices in a real case.
- Physical Space: It is better if the developed method can be measured in an adequate physical place where more people can be gathered, and different scenarios can be mocked.

Bibliography

[1] Abdullah Emin Akay et al. “Using LiDAR technology in forestry activities”. In: *Environmental monitoring and assessment* 151.1 (2009), pp. 117–125.

[2] *Analytics and IOT services to measure visitor engagement.* URL: `https://www.livealytics.com/`.

[3] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[4] Somit Gupta et al. “Challenges, Best Practices and Pitfalls in Evaluating Results of Online Controlled Experiments”. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 877–880.

[5] Kristina Host, Marina Ivasic-Kos, and Miran Pobar. “Tracking Handball Players with the DeepSORT Algorithm.” In: *ICPRAM*. 2020, pp. 593–599.

[6] *Intel realsense Lidar Camera L515*. 2021. URL: `https://www.intelrealsense.com/lidar-camera-l515/`.

[7] Yang Jie et al. “Ship Detection and Tracking in Inland Waterways Using Improved YOLOv3 and Deep SORT”. In: *Symmetry* 13.2 (2021), p. 308.

[8] Mounia Lalmas, Heather O’Brien, and Elad Yom-Tov. “Measuring user engagement”. In: *Synthesis lectures on information concepts, retrieval, and services* 6.4 (2014), pp. 1–132.

[9] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. “Time series databases and influxdb”. In: *Studienarbeit, Université Libre de Bruxelles* 12 (2017).

[10] Mostafa Noshy, Abdelhameed Ibrahim, and Hesham Arafat Ali. “Optimization of live virtual machine migration in cloud computing: A survey and future directions”. In: *Journal of Network and Computer Applications* 110 (2018), pp. 1–10.

[11] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[12] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[13] Rafael Hengen Ribeiro et al. “ASIMOV: a Fully Passive WiFi Device Tracking”. In: (2021), pp. 1–3.

[14] Bruno Rodrigues et al. “BluePIL: a Bluetooth-based PassIve Localization Method”. In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2021, pp. 28–36.

[15] Bruno Rodrigues et al. “CCount: Correlating RFID and Camera Data for High Precision Indoor Tracking”. In: *University of Zurich, Department of Informatics, Tech. Rep* 1 (2022).

[16] Bruno Rodrigues et al. “LaFlector: a Privacy-preserving LiDAR-based Approach for Accurate Indoor Tracking”. In: *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE. 2021, pp. 367–370.

[17] Sougata Sen et al. “Accommodating user diversity for in-store shopping behavior recognition”. In: *Proceedings of the 2014 ACM International Symposium on Wearable Computers*. 2014, pp. 11–14.

[18] Bing Shuai et al. “Multi-object tracking with siamese track-rcnn”. In: *arXiv preprint arXiv:2004.07786* (2020).

[19] theAIGuysCode. *TheAIGuysCode/Yolov3-Cloud-tutorial: Everything you need in order to get yolov3 up and running in the cloud. learn to train your custom yolov3 object detector in the cloud for free!* URL: `https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial`.

[20] theAIGuysCode. $TheAIGuysCode/yolov3_deepsort : Objecttrackingimplementedwithyolov3, deepsortandtensorflow..$ URL: `https://github.com/theAIGuysCode/yolov3_deepsort`.

[21] Tzutalin. *Tzutalin/labelimg: LabelImg is a graphical image annotation tool and label object bounding boxes in images*. URL: `https://github.com/tzutalin/labelImg`.

[22] Jesus Vargas, Nelson Alberto, and Oswaldo Arevalo. “Algorithms for Decision Making Through Customer Classification”. In: *Proceedings of International Conference on Intelligent Computing, Information and Control Systems*. Springer. 2021, pp. 535–542.

[23] Zhongdao Wang et al. “Towards real-time multi-object tracking”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer. 2020, pp. 107–122.

[24] Junjie Yan et al. “The fastest deformable part model for object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2497–2504.

[25] Xiaoling Zhang et al. “An examination of social influence on shopper behavior using video tracking data”. In: *Journal of Marketing* 78.5 (2014), pp. 24–41.

Abbreviations

DPM	Deformable Parts Model
FoV	Depth field of view
IOU	Intersection Over Union
JDE	Detector and Embedding Model
LiDAR	Intel Realsense L515 Light Detection and Ranging Scanner
NTP	Network Time Protocol
OpenCV	Open Source Computer Vision Library
SDK	Software Development Kit
SORT	Simple Online Real time Tracker
SSD	Sum of Square Distance
VM	Virtual Machine
YOLO	You Only Look Once
YOLOv3	You Only Look Once Version 3

List of Figures

List of Tables