Detection of Objects and Trajectories in Real-time using Deep Learning by a Controlled Robot

Adil Sarsenov, Aigerim Yessenbayeva[a], Almas Shintemirov[b] and Adnan Yazici[c]
Department of Computer Science, Nazarbayev University, 53 Kabanbay Batyr Ave, Nur-Sultan, Kazakhstan

Keywords: Deep Learning, Object Detection, Trajectory, Depth Camera, LIDAR.

Abstract: Nowadays, there are many different approaches to detect objects as well as to determine the trajectory of an object. Each of these approaches has its advantages and disadvantages in terms of real-time use for various applications. In this study, we propose an approach to detect objects in real-time using the YOLOv3 deep learning algorithm and plot the trajectory of an object using 2D LIDAR and depth cameras on a robot. The laser rangefinder allows us to find distances to objects from a certain angle, but does not provide accurate object detection of the object class. In order to detect the object in real-time and discover the class to which the object belongs, we formed YOLOv3 deep learning model using transfer learning on several classes from data sets of publicly accessible images. We also measured the distance to an object using a depth camera with LIDAR together to determine and estimate the trajectory of objects. In addition, these detected trajectories are smoothed by polynomial regression. Our experiments in a laboratory environment show that YOLOv3 with 2D LIDAR and depth camera on a controlled robot can be used fairly accurately and efficiently in real-time situations for the detection of objects and trajectories necessary for various applications.

1 INTRODUCTION

In recent years, the robotization of all spheres of human activity is gaining momentum. A technological breakthrough in robotics and machine learning allows people to build autonomous vehicles and various robots to work autonomously. Mobile robots typically focus on solving a wide range of diverse applications to collect heterogeneous information and to be able to perform technological operations in extreme environments. Almost every robot needs input sensors like a laser scanner (LIDAR) or a camera to perceive the environment that surrounds it. LIDAR is a device that measures distances to targets at specific angles and is used for light detection and ranging. The rapid evolution of the technological level tends to lower the prices of the various sensors including the 2D and 3D LIDARs. 2D LIDAR can only estimate the distance to objects. For this reason, the data that can be extracted from LIDAR is the only array of ranges to the objects.

The main objective of this study is to recognize the object in the image frame and to estimate its position to trace the trajectory of the object, which is supplemented by the use of the Jaguar mobile robot. The robot is equipped with 2D Hokuyo LIDAR, an IP camera, a depth camera and a Raspberry Pi, which are the main components of our system architecture. The Raspberry Pi will serve as a processor and data transmitter, whose data collected from LIDAR and cameras are sent to the PC via Wi-Fi for further processing (Lu, 2018). In this study, we propose the use of a deep learning algorithm on LIDAR data to detect the bounding boxes in the images to increase awareness around the environment and the use of the distance to a specific angle of the object obtained from LIDAR which can help locate an object and retrace its trajectory. The distances and angles returned by LIDAR are used with a distance to an object using the depth camera to construct the trajectory of an object.

Object detection is performed using the You Look Only Once v3 (YOLOv3) algorithm. In order to reduce the size of the training data, the transfer learning approach with pre-trained weights is applied. The final step is to locate the object by returning its Cartesian coordinates and plotting the trajectory path. The source of a dataset is images (Kuznetsova and at. al.,

[a] https://orcid.org/0000-0002-0128-3634
[b] https://orcid.org/0000-0002-6969-8529
[c] https://orcid.org/0000-0001-9404-9494

2018) for our deep learning algorithm for the detection of objects requiring big data. The transfer learning (Tan and at. al., 2018) is a technique that avoids training the model from scratch by training your own rather smaller data size using pre-trained weights. It therefore applies the previous knowledge acquired by the huge amount of data to new small data specific to an application.

The main contributions of the study are as follows.

- We propose accurate and robust solutions for the detection and localization of objects in real time on a system architecture composed of a 2D Hokuyo LIDAR, an IP camera, a depth camera and Raspberry Pi. In addition, we have tested and validated this proposed architecture in a laboratory environment on a robot equipped with these components.
- We train YOLOv3 deep learning algorithm on open-source datasets and use obtained parameters for real-time object detection instead of using traditional object detection approaches, which are relatively slow.
- 2D Hokuyo LIDAR is used to find the coordinates of the object and follow the movement of the detected object. We use polynomial regression on these generated coordinates to find the line that best fits and smooths the trajectory.
- In addition, the distances measured from the depth camera are used with the LIDAR angles for the trajectory plot for an accurate estimation of the trajectory.
- With our experiments we show that YOLOv3 with 2D LIDAR and depth cameras on a controlled robot is used fairly precisely in real-time situations for the detection of objects and the estimation of trajectories necessary for various applications.

The rest of the article is organized as follows. The second section includes related work. Section 3 briefly explains the hardware configuration. The fourth section describes the deep learning algorithm used for object detection. Section 5 presents trajectory estimation including localization techniques for measuring distances to objects. The experimental results are presented and discussed in Section 6. Finally, we give the conclusion and future work.

2 RELATED WORK

This section provides a review of the literature related to our study. Viola Jones et al (Wang, 2014) proposed initial attempts to detect objects on the images using the features of Haar wavelet and Adaboost cascading algorithm. Later in 2005, Dalal and Triggs (Dalal and Triggs, 2005) proposed Histograms of Oriented Gradients (HOG), the HOG feature was more discriminative than Haar-cascade features. As already mentioned, deep learning algorithms for the localization and detection of objects as well as the use of these algorithms in the field of robotics are the subject of active research. With the increasing popularity of deep learning models, HOG functionality has been replaced by models of convolutional neural networks (CNN). Nowadays, there are several states of algorithms for the detection and localization of objects in images. Some examples are the Single Shot MultiBox Detector (SSD) method (Liu and at. al., 2016), Faster RCNN (Ren and at. al., 2015), YOLOv3 (Redmon and Farhadi, 2018), etc.

For real-life applications, there is no straightforward answer to the question of which of them is the best. The following sources of literature are more related to applications that involve object detection in combination with the various sensors. (Wei and at. al., 2018) proposes the LIDAR camera and data fusion using fuzzy logic for beacon detection as part of the multi-sensor collision avoiding system. The study places LIDAR vertically in order to extract points correlated to the beacon from different angles and applied support vector machines in order to extract characteristics which are later combined with object detection using fuzzy logic. Insu Kim and Kin Choong Yow in (Kim and Yow, 2015) propose an estimate of the location of objects from a single camera. They use HOG to detect an object and estimate the distance to that object using stereo vision. The state-of-the-art deep learning classification algorithms are provided in (Ciaparrone and at. al., 2020).

The study in (Krizhevsky et al., 2012) introduces AlexNet that contains eight layers, five convolutional and three fully-connected layers and as an activation function it uses ReLU. After the success of the ResNet model (He and at. al., 2015), in 2016 the Inception-v4 and InceptionRes were introduced (Szegedy and at. al., 2016). The main idea of SENet (Hu and at. al., 2017) is to learn a weight tensor that provides different weights for feature maps for each channel (activation).

3 HARDWARE CONFIGURATION

In this section, we present the preliminary hardware configuration, including the fundamental context of the mobile robot, the sensors, the IP camera, the depth cameras and the Raspberry Pi 3.

The design of a mobile robot is often completely determined by the environment in which it is used and based on various parameters (Rincon and at. al., 2019). In our study, we focus on the Jaguar 4x4 wheeled mobile robotic platform. This robot is mainly designed for indoor and outdoor navigation. One of the advantages of this mobile platform is that it has faster maneuverability and movement capacity on a vertical stage with a maximum step of 155 *mm* and a variety of speed between 0 and 14 *km/h*.

Sensors are used to convert a certain physical quantity into an electrical signal. The main task of any sensor is to respond to external influences and provide the system with data on changes in the environment. There are different types of sensors used in the hardware configuration of our testbed. In particular, for this study, the sensors detect movement and measure the distance to objects.

An IP camera is a digital video camera and the transmission of a video stream through it is done in digital format on an Ethernet and TokenRing network using the IP protocol (Cabasso, 2009). Specialized IP cameras often transmit video in an uncompressed form. IP cameras are often powered by Power over Ethernet (PoE). Higher resolutions, including megapixels, can be used in IP cameras, as there is no need to transmit an analog signal in Phase Alternating Line (PAL) or National Television Standards Committee (NTSC) format. The typical resolution for network cameras is 650 × 840 pixels. Generally, IP cameras can be classified as webcams.

Intel RealSense Depth cameras shoot video, but in each pixel instead of brightness, there is a point of depth for each corresponding pixel. These cameras have been gradually increasing the resolution, depth accuracy and stability of the output signal, but they are still relatively imperfect. In our study, we use the Intel RealSense D435 depth camera (Keselman and at. al., 2017). The camera has the highest possible viewing angle and it minimizes the risk of blind spots appearing and is equipped with a shared shutter, which guarantees the highest quality and clear perception of data. The camera has a set of video sensors that can identify differences in images with resolutions up to 1280 x 720 pixels. One advantage of the RealSense camera that comes with its own Intel-supported SDK. The main objective of RealSense technology is to enable a new type of communication and to give the possibility of interacting with the outside world.

The Raspberry Pi 3 is a single board computer launched in industrial production in 2012. Initially, it was intended as an affordable solution to introduce the basics of programming and global high technology. It contains the ARM processor, RAM chips, a slot for a micro-SD card, as well as an Ethernet port, HDMI, a 3.5 mm audio output and USB ports for connecting peripherals. The majority of the components required for our mission are already integrated into the Jaguar mobile platform. It is a good choice for small storage and data transmission. Due to the absence of LIDAR among the components of the Jaguar, we were able to manually mount the LIDAR on top of the robot hood. The main disadvantage of such a LIDAR location from the point of view of the building map and obstacle avoidance is that it does not see certain objects located below the LIDAR. Other components such as the Wi-Fi router and the integrated IP camera are included in the components of the mobile platform. Finally, the version fully connected to the robot's sensors is visible in Figure 1. We connect the Raspberry Pi which is inside the Jaguar robot to the source $5V$. In addition, we also connect the LIDAR to another source $5V$ and connect it to the Raspberry Pi. We have made the appropriate configuration and configured the IP addresses and masks of the global system. This configuration is required so that the Raspberry Pi can transmit data from LIDAR directly to the PC via Wi-Fi. The main computer which acts as a server receives and then processes the data obtained from the client which, in our case, is Raspberry Pi. This simple client-server architectural model allows data to be processed and transmitted over the network.

The data received from LIDAR is in the form of an array. The size of each array is 1080 points. The points represent the distance to each object within the angle step of the LIDAR. The video stream from the IP camera is also sent via Wi-Fi. It is possible to simply connect to the camera via its IP address. The video frame can be adjusted to the specific size, the default is 680 x 480 pixels.

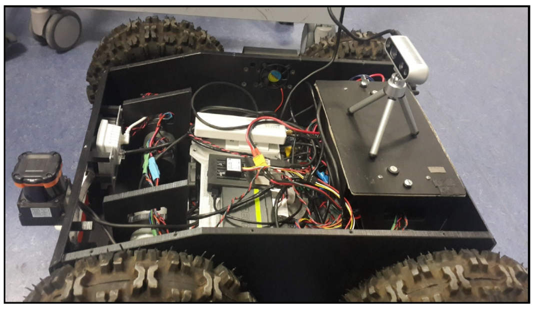


Figure 1: The hardware setup.

4 OBJECT DETECTION

The main objective of object detection is to find objects: pedestrians, bicycles, buildings or any other class on images and videos and to annotate them with the bounding box (Khalifa et al., 2020). In other words, the output of the object detection is *(x, y, width, height)* from the bounding box that contains the object. On the other hand, object classification or recognition indicates what the object is in the bounding box. Thus, the output of the classification is a class label of the particular bounding box. We use the YOLOv3 deep learning model for object detection. Generally, algorithms related to object detection and its localization within an image consist of two groups:

1. Algorithms built on the classification.
2. Algorithms built on the regression.

The algorithms consist of two phases. First, the algorithm selects distinct regions of an image. In addition, these regions are classified using CNNs. Prediction is made for each region selected. Since the algorithm runs for each selected region, it can be very slow. The region-based CNN (RCNN) and its modifications (Faster-RCNN) are considered as this type of algorithms (Girshick et al., 2013),(Yang and at. al., 2020). The algorithms based on the second type do not select the regions offered in an image. Instead, it involves predicting bounding boxes and related classes in a single execution of the algorithm for the entire image. The YOLO algorithm belongs to the second category.

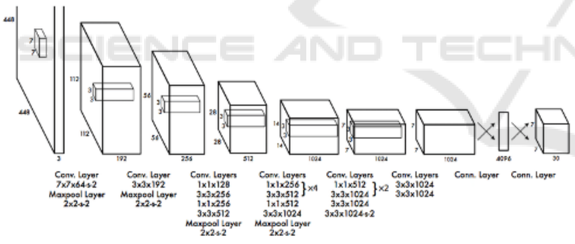


Figure 2: Architecture of CNN (Li and at. al., 2018).

The YOLO algorithm was introduced at the end of 2015 (Redmon and at. al., 2016). YOLO allows fast image processing compared to CNN, but with lower accuracy. As mentioned earlier, the YOLO solves the problem of object detection as a regression problem and due to the high-speed image processing, it is suitable for use in real-time systems.

The CNN model takes the entire image as input and gives the coordinates of the bounding boxes and the probability of belonging to classes. The CNN architecture is shown in Figure 2. The output of the CNN is a tensor with coded predictions, and its size is $S \times S$ $(B \times 5 + C)$. Here, S denotes the dimension of a grid and B denotes the number of bounding boxes. The value C is the number of classes that a NN is able to recognize.

The YOLOv3 algorithm in terms of mean average precision (*mAP*) is superior to the Single Shot MultiBox Detector (SSD) method (Liu and at. al., 2016), but inferior to the Faster R-CNN method (Ren and at. al., 2015). However, the Faster R-CNN frame does not process more than 5 frames per second. Using the NVIDIA Titan X graphics processing unit, the YOLOv3 algorithm processes images at 30 frames per second, good for real-time video processing systems. Therefore, we have chosen the YOLOv3 model for this study. The pseudo code of object detection using YOLOv3 is given in Algorithm 1.

Algorithm 1: Real–time Obj. Detection With YOLOv3.

```
function detectObject (V, $t_d$);
Input: V : video stream
       $t_d$ : threshold
Output: Object market with bounding box
for each frame f in V do
    //resize frame
    f = f.size(416, 416)
    //detect objects in frame above threshold
    detections = detect (f, td = 0.5)
    //draw bounding boxes on detected objects
    image = drawBoundingBox(detec–s, frame)
    //display image
    show(image)
end
return image
```

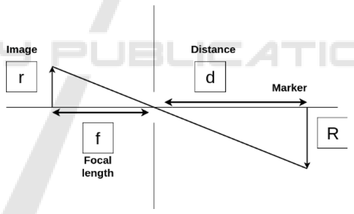


Figure 3: Triangle similarity principle.

5 TRAJECTORY ESTIMATION

This section presents the methods of distance and trajectory measurement. Estimating the distance between a camera and an object is often offered as an inexpensive solution to alternative methods such as the use of laser scanners or radars. In general, there are several basic methods for estimating the distance between a particular object and a simple monocular camera (Kim and Yow, 2015).

We determine the distance to an object using information about its size. If the real dimensions of an object (height or width) are known, then, knowing the coordinates of an object in the image, using the formulas of projective geometry, we can calculate the distance to it (Tian and at. al., 2018). The disadvantage of this approach is the increase in error when the object is further away. In addition, for the algorithm to work properly, the length and width of the object must not change over time. Also, to use this method, you need to know the focal length of the camera (Oztarak and at. al, 2016) in advance. Finding the distance from a camera to a certain object is a very well-studied problem in image processing (Kim and at. al., 2020). One of the main and straight-forward methods is the triangle similarity. The monocular camera produces a one-to-one relationship between the image and the object located inside the image. Using this principle, it is possible to derive a relationship between known parameters: focal distance *(f)*, width in the image plane *(w)*, and width of the object in the object plane *(W)* and unknown parameter, distance from camera to object *(d)*. Using the principle described in Figure 3 of triangle similarity, the following formulas can be obtained:

$$d = f \times \frac{R}{r} \tag{1}$$

For example, let's measure the distance to the pedestrian 2.75 meters from the camera. To do this, we adjust a pedestrian with a known width *(W* to a certain distance *D* from the camera. In our experiments, our known width is 34cm and we adjust a person to 3.8 meters from the camera. Then, we capture an image of an object and measure the apparent width in pixels, which allows us to estimate the focal distance:

$$f = 380cm \times \frac{346px}{34cm} \tag{2}$$

As an object (marker) continues to move closer and farther away from the camera, it is possible to use the triangle similarity to determine the distance from a camera. Now, an object is moving 2.75 meters from the camera and the bounding box returns a perceived width which is 435 pixels.

$$d = f \times \frac{34cm}{435px} = 302cm \tag{3}$$

Finally, using the principle of triangle similarity, the distance at 2.75 meters is estimated at around 3.02 meters. Another way to get the focal length of the camera is the process of calibrating the camera using the chessboard. Camera calibration consists of obtaining internal and external parameters, namely the camera matrix and the camera distortion coefficients from the available photos or videos taken by it. Camera calibration is often used at the initial stage of solving many computer vision problems, especially in augmented reality. In addition, calibrating the camera corrects the distortion of photos and videos. In our study, we use the method which was developed by Zhengyou Zhang and is based on the use of a flat calibration object in the form of a chessboard (Zhang, 2000). We use the principles of triangle similarity to find the distance to an object at certain ranges, as shown in Figure 4. The experimental results are given in the next section.

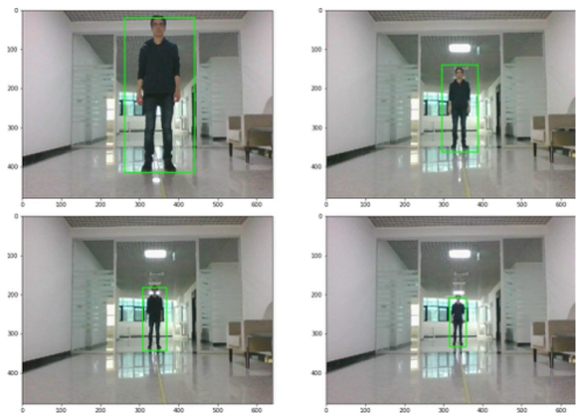


Figure 4: Person at certain ranges.

5.1 Distance Measurement using Depth Camera

This section describes another and more accurate method to estimate the distance to an object using a depth camera (Draelos and at. al., 2015). The depth camera is working using the following principle:

1. First, it has an Infra-Red(IR) projector that pins the scene with invisible markers. This allows projecting Infra-Red lights which pin the object like a cloud of dots representing the depth for each pixel. The output of this is called the depth map.
2. The distance triangulation applied between IR projector and camera.

In simple words, a depth map is an image in which, for each pixel, instead of color, its distance from the camera is stored. In our study, we take the depth map from the Intel depth camera, but it could also be constructed from a stereo pair of images.

The quality of the depth of an image is highly correlated with the resolution of a camera. Our depth camera supports a few defined depth resolution presets, which can be selected in advanced camera mode. We have tried several of them and decided to use 848×480. Generally, the lower resolutions can also be used, but it will decrease the depth precision.

The overall process of finding the distance to an object using the depth camera is pretty straight forward. The method that we used consists of several steps. Firstly, we capture the object by accessing its color component RGB frame and depth frame. The next step is to align both frames because they must be accessible from the same physical viewport. Figure 5c illustrates the RGB data and depth data combined into a single RGB-D image and it can be seen that images is properly aligned. The process of the depth alignment further allows using the depth data as any of the other image channels.

After the process of the depth alignment, it is possible to use any deep learning algorithm on the RGB frame to find specific objects. In our case, we use pre-trained state-of-the-art object detection model YOLOv3 to recognize and localize the object in the RGB image and use additional depth data to find the distance to an object (Henry et al., 2012).

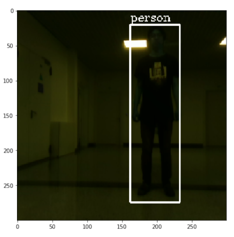
It can be seen in Figure 5a, the object is found using object detection model and marked with the bounding box. Since we have aligned RGB-D image it is possible to project the bounding box on the depth frame as shown in Figure 5b. To be more concrete, we draw the bounding box on the depth frame using its coordinates from the RGB frame.

Generally, there exist several approaches to calculate the distance to an object. As one of the approaches, the distance to the object can be measured by averaging the depth data inside the bounding box. The number of experiments shows that this is not a reliable method, because some depth pixels are too far away and can significantly affect the result of overall distance measurement. For this reason, we propose first to find the coordinates of the center pixel of the cropped bounding box. In our case, the coordinates of the center pixel of the bounding box are already returned by the YOLOv3 model. There also exists another popular format for the bounding box, instead of returning *x, y, h, w* it returns the coordinates of the corners of the box. The last step is to extract the depth using the obtained coordinates. The output is the final distance to an object. The results of the distance measurement are provided in the next part.

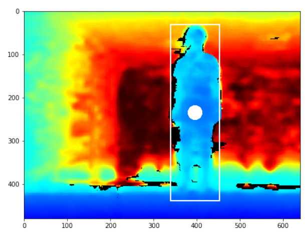
5.2 Trajectory Estimation

This subsection describes how to estimate the trajectory of an object and the process of getting the coordinates of this object to plot the distance travelled. The trajectory is the path that the object follows under the given reference frame (Ciaparrone and at. al., 2020). As we know in classical mechanics the trajectory is represented as a series of the coordinates. The trajectory vector can be represented as:

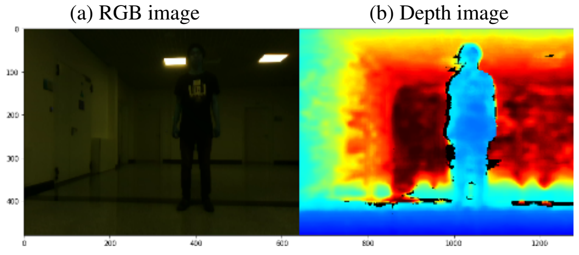
$$T_1 = ((x_1,y_1),(x_2,y_2),(x_3,y_3)...(x_n,y_n)), \quad (4)$$



(a) RGB image



(b) Depth image



(c) Aligned image (RGB - D)

Figure 5: Depth estimation process.

where the vector T_1 contains the coordinates of an object.

In our case, to plot the trajectory, we need to find the distance and the angle of an object with respect to the LIDAR. As was mentioned, the laser rangefinder (LRF) returns the distance in the form of scans. If the distance and the angle are known, it is possible to get the rectangular coordinates of an object. Obviously, if the coordinates of an object are known it is possible to plot its trajectory. We used Hokuyo UTM-30LX scanning laser rangefinder. The LIDAR has a 270° area scanning range with 0.25° angular resolution. In order to obtain the coordinates of an object, we convert Polar's LIDAR data into Cartesian coordinates. Generally, we must solve a simple problem of a right triangle problem with a known long side and angle. This allows us to localize an object on the Cartesian plane. Below are standard formulas for converting a Polar Coordinates (r,θ) to Cartesian coordinates (x,y).

$$\alpha_{radius} = \alpha_{degrees} \times \frac{\pi}{180} \quad (5)$$

$$x = r \times \cos(\alpha_{radius}), y = r \times \sin(\alpha_{radius}) \quad (6)$$

In the previous sections, the process of detecting an object and estimating its distance is described. Now, if the distance to an object and the angle to the object are known, it is possible to localize the object using the conversion formula. For example, if the person stays at 2 meters from LIDAR at an angle of 90°. To be more accurate, we convert meters to millimetres and multiplied the result to the re-scaling coefficient 0.05. Then, the coordinates of the person are:

$$x = (2000 \times 0.05) \times \cos(\alpha_{radius}) = 0.079 \quad (7)$$

$$y = (2000 \times 0.05) \times (-\sin(\alpha_{radius})) = -99.99 \quad (8)$$

As mentioned before, our LIDAR has a scanning range of 270°, and its angular resolution is 0.25°, so in total, we have 1080 laser scans, one scan at each step of angle. In order to simplify the visualization of the laser beams and to draw a trajectory, we pre-computed some useful values for plotting trajectory using the formula below:

$$\alpha_{radius} = (\frac{-270}{2} + \frac{value}{1080} \times 270) \times \frac{\pi}{180} \quad (9)$$

where *value* is the number in the range of 1 and 1080.

Finally, we have obtained an array of radians with the length of 1080 for the angles between (-135, 135) which allows us to convert the coordinates by multiplying the radians obtained at each corresponding laser scanning range. We visualize the laser beams using the precalculated values and the canvas library. The experimental results of the object trajectory are provided in the following section.

6 EXPERIMENTS AND RESULTS

In this section, we present the datasets used for our experiments. We then describe the experimental results of the distance measurement. Finally, we present the results of LIDAR and object detection for the trajectory estimation and the fusion of these two. All the experiments related to object detection were carried out on a machine with an Intel Core-i7 8570H processor, an NVIDIA GeForce 1080Ti GPU and on an Ubuntu 16.04 LTS operating system.

6.1 Dataset for Object Detection

There are many publicly available image datasets used for training models for object detection (Deng and at. al., 2009). Google Open Images-v4 and Pascal Visual Object Classes (VOC) 2007 and 2012 are two of the most popular ones. The Google open images-v4 dataset contains around 600 classes and 1,743,042 training images, as well as validation 41,620 and 125,436 test images (Kuznetsova and at. al., 2018). We choose four classes (Person, Box, Chair, Mechanical Fan), which are relevant to our research from Google's open image dataset. For testing, we use 10% of each class and all are high-quality images with different image sizes. The other dataset that we use in our experiments is the Pascal VOC dataset. The complete dataset contains 20 classes. We merged the Pascal VOC 2007 dataset with 2012. We extracted 12 classes relevant to our study. These classes are airplane, bicycle, bus, car, cat, cow, dog, motorcycle, person, horse, sofa and train.

The training process is done on NVIDIA 1080 Ti GPU. Instead of training model from scratch, we use pre-trained convolutional weights that have been trained on the ImageNet dataset. Using these convolutional weights helps us to train our model faster. Additionally, the mean average precision of the model increases compared to training from scratch.

We trained several YOLO models on our datasets. The training process for the Pascal VOC dataset took approximately 5 days for YOLOv3 with an ImageNet backbone that contains an additional 53 convolutional layers. The next model to be trained is YOLOv3-tiny. The training process for the tiny version took about 2 days with the backbone of an ImageNet that contains 15 additional convolutional layers. The average accuracy per class is computed every 5000 iterations. In order to test the algorithm in real-time, we choose the trained weights with the highest *mAP*.

6.2 Distance Measurement to the Object

The trained checkpoints of the model allow us to detect and identify the objects in real-time. After the object is marked with a bounding box, it is possible to find the distance using the methods described earlier. To evaluate the distance measurements, we conduct experiments with the class person at 2.75, 5.5, 7.25 and 9 meters from the camera and LIDAR. We compare our results obtained for distance measurement with similar studies. The results are presented in Table 1. The abbreviations used are AD (actual depth), MD (measured depth) and DERR (depth error rate). DERR, where Z is the actual depth and Z' indicates the estimated depth, is calculated as follows.

$$DERR = \frac{|Z - Z'|}{Z} \quad (10)$$

The results show that the depth camera has accurate results up to 5 meters. The depth precision start fluctuates at longer distances. The main advantage of the depth camera is that it is suitable for the real-time distance estimation while the monocular real-time camera fluctuates enormously even at a closer range. Conventional methods work better at lower ranges, while LIDAR can give a very precise distance up to 30 meters with a small deviation in millimeters. Table 2 provides accuracy of our approach for distance measurement and compared to other existing methods. Algorithm 2 includes the details of distance measurement using depth camera.

Table 1: Comparison of distance measurement methods.

	CMOS Camera		Depth Camera		2D LIDAR	
AD (m)	MD (m)	DERR (%)	MD (m)	DERR (%)	MD (m)	DERR (%)
2.75	3.02	9.8	2.745	0.18	2.75	0
5.50	4.99	9.2	5.86	6.36	5.50	0
7.25	6.5	10.3	8.1	11.72	7.25	0
9.00	8.01	11	10.2	13.33	9.00	0

6.3 Object Detection and Trajectory Plot

Now, we give a detailed account for the combination of the object recognition with LIDAR localization. The main steps of the algorithm used in our experiments are given below.

1. Detect and recognize an object using YOLOv3.
2. If an object is recognized, then start saving the coordinates in CSV file.
3. Plot the trajectory scatter plot of coordinates.
4. Apply polynomial regression in order to smooth the trajectory path.

The first step requires the execution of YOLOv3 in real-time for object detection, since we have already trained the YOLOv3 model, we can use our trained weights in order to more precisely detect objects, which is called transfer learning.

Algorithm 2: Dist. Measurement Using Depth Cam.

```
function distanceMesurementDepth (V, D, $t_d$);
Input: V : video stream
       D : depth stream
       $t_d$ : threshold
Output: distance : distance to an object
for each frame $f_v, f_d$ in V, D do
    //detect objects in frame above threshold
    detections = detect (f, $t_d$ = 0.5)
    //draw bounding boxes on detected objects
    image = drawBoundingBox(detections, $f_v$)
    //reproject bounding boxes on depth frame
depthBoundBox=reprojectBoundBox (detec–s, $f_v$)
    //evaluate distance
    distance=takeCenterPixel(depthBoundBox)
end
return distance
```

It is possible to take the minimum range value at LRF from the ranges at each index of an array of the scans to verify the closest object at a particular angle. When the object is marked with the bounding box and begins to move around, its coordinates are stored continuously in the CSV file, for further processing. Additionally, it is possible to apply a polynomial regression in order to smooth the trajectory.

Table 2: Comparison with the other works.

System	Error(%)(Accuracy)
M. Sereewattana et al.(Only for stationary object below 3m)	3.9 ~12.4
Insu Kim and Kin Choong Yow (Kim and Yow, 2015)	0.4 ~14.5
CMOS Camera Using Principle of Similar Triangles	9.2 ~11
Depth Camera	0.18 ~13.33
Laser rangefinder	0

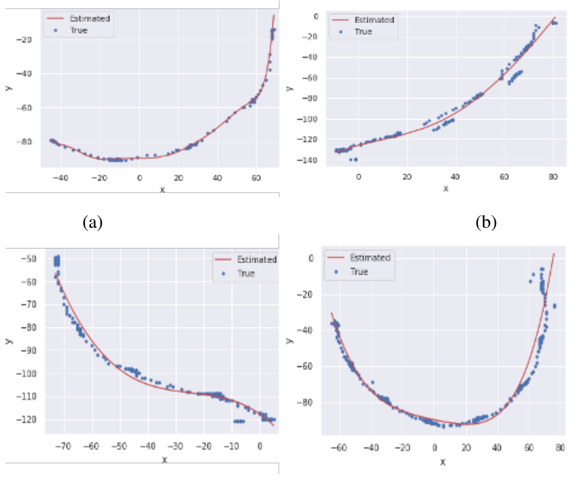
Table 3: RMSE, R^2 scores for trajectories.

Figure 6	RMSE	R^2	Polynomial Degree
a	1.62	0.99	10
b	4.9	0.98	5
c	1.95	0.99	4
d	10.92	0.65	3

Polynomial regression finds the line that best fits our generated data. Figure 6 shows the various trajectories for objects and the results of applying polynomial regression. Table 3 includes corresponding root mean squared error (RMSE), R^2 scores and the degree of each line that best fits to the points.

We have done some experiments to plot the trajectory of a person using the angles obtained from the LIDAR and the distances obtained from the depth camera. In addition we compared the trajectories using distances obtained from the LRF and depth camera in the same angles extracted from the LRF. For this experiment, a person is moving with a stable 0.5 m/s straight ahead to the LRF and camera, covering the distance of the about 6.5 meters. Furthermore, we consider the trajectory obtained from the LRF as the ground truth. The distances from the depth camera are sufficiently accurate and the distribution of the points is almost the same compared to LRF in a real-time system. The Y-axis values of the trajectory using the depth camera are slightly higher than the trajectory using the LRF. It can be concluded that distances from the depth camera can also be used for detection of trajectories.

The trajectory is detected by using an ordinary Complementary Metal Oxide Semiconductor (CMOS) camera. The distribution of coordinates is very huge because the distances obtained with the CMOS camera are highly fluctuating. It can be concluded that the single CMOS camera using the principle of similar triangles is not suitable for accurate trajectory plotting in real-time. Then we aggregated trajectories of the three sensors. The series of coordinates obtained from the depth camera are close to the ground truth, while the coordinates obtained from



(a) (b) (c) (d)

Figure 6: Depth estimation process.

Table 4: Comparison of trajectories.

Ground Truth Sensor	Sensors	RMSE	Spearman Correlation
2D LIDAR	Depth Camera	52.03	0.98
	CMOS Camera	101.5	0.67

the CMOS camera have a high deviation from rest. Table 4 describes the results of the trajectory comparison using the RMSE and Spearman's rank correlation coefficient. As a ground truth value, the coordinates obtained from 2D LIDAR are used.

6.4 Trajectory using Controlled Mobile Robot

In the previous sections, we traced the trajectory of a person moving straight relative to the fixed 2D LIDAR. We now look at the trajectory plotting with the same method but with an extension of a trajectory. By extension, we mean moving our robot to the particular checkpoint and continuing to calculate the trajectory of a person with the same technique. The checkpoint is where the person disappears from the view of the camera. The pseudo-code of object localisation and detection of trajectories is given in Algorithm 3.

The experiments have been carried out inside the building, the same as the previous experiments. In this specific experiment, a person is moving with a stable 0.5 m/s straight ahead from the LRF and the camera, covering the distance of about 6.5 meters. Towards the end of the 6.5 meters, the person is moved to the right and disappears from the view of the camera. Then our robot moves to this position where the person disappears and continue to calculate the trajectory again. The trajectory is shown in Figure 7.

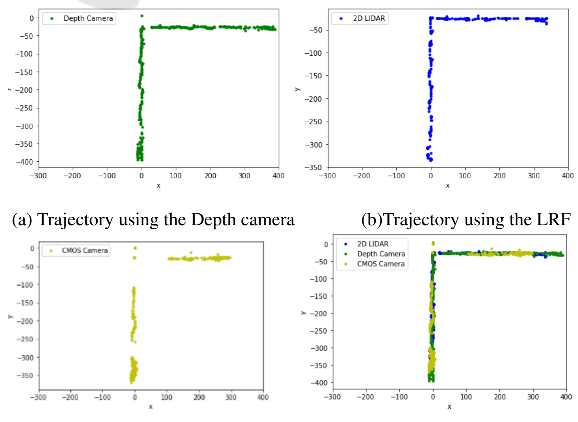
As it is possible to see from this plot, the person moves straight and turns right, again covering the distance of about 6.5 meters. Table 5 describes the results of the trajectory comparison for the case with the relocation of the mobile robot. It can be concluded that the trajectory of an object can be estimated using a mobile robot in a controlled manner.

Table 5: Comparison of trajectories.

Ground Truth Sensor	Sensors	RMSE	Spearman Correlation
2D LIDAR	Depth Camera	33.37	0.90
	CMOS Camera	78.94	0.61

Algorithm 3: Object Localisation and Trajectory Plot.

```
function plotTrajectory (V);
Input: V : video stream
Output: $coordinates_{plot}, regression_{plot}$: Trajectory plots
for each frame f in V do
    //Start saving coordinates of detected object
    if Object is in f then
    lidar = turnOn()
    //get converted coordinates
    x, y = lidar.saveCoordinates()
    //Stop prog., apply regr–n and plot the traj.
    if button “ESC” isPressed then
    lidar = turenOff()
    $coordinates_{plot}$=plotTrajectory(x, y)
    $regression_{plot}$=polynomialRegression(x, y);
end
return $coordinates_{plot}, regression_{plot}$
```



(a) Trajectory using the Depth camera (b)Trajectory using the LRF

(c) Trajectory using the CMOS camera (d)Aggregation of trajectories

Figure 7: Trajectories of a pedestrian.

7 CONCLUSION AND FUTURE WORK

The main objective of this study was to implement the method for the object trajectory estimation using the YOLOv3 object detector algorithm an 2D laser rangefinder. We analysed the YOLOv3 and YOLOv3-tiny deep learning models on various classes from datasets including Pascal VOC and Google Open Images. We used additional pre-trained convolutional weights to increase the capability of the model to detect the objects. The combination of the object detection and 2D LIDAR helps the trajectory estimation of an object. In addition, we tried to plot the trajectory by using the distances from the depth camera, CMOS camera and LRF angles. We also estimated the trajectory of an object using a mobile robot in a controlled fashion. Furthermore, we used polynomial regression with the purpose of smoothing trajectory path but only for suitable cases. The experiments show that our approach is feasible and robust to obtain the object location and further draw the trajectory.

As a possible future work, we plan to investigate different algorithms for the objects trajectory prediction, as well as methods related to object tracking using mobile robot control.

REFERENCES

Cabasso, J. (2009). Analog vs. ip cameras. *Aventura Technologies*, 1(2):1–8.

Ciaparrone, G. and at. al. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of CVPR'05*, pages 886–893. IEEE Computer Society.

Deng, J. and at. al. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248 – 255.

Draelos, M. and at. al. (2015). Intel realsense=real low cost gaze. pages 2520 – 2524.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. pages 98–136.

He, K. and at. al. (2015). Deep residual learning for image recognition.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

Hu, J. and at. al. (2017). Squeeze-and-excitation networks.

Keselman, L. and at. al. (2017). Intel realsense stereoscopic depth cameras.

Khalifa, A. B., Alouani, I., Mahjoub, M. A., and Amara, N. E. B. (2020). Pedestrian detection using a moving camera: A novel framework for foreground detection. *Cognitive Systems Research*, 60:77–96.

Kim, D. H. and at. al. (2020). Real-time purchase behavior recognition system based on deep learning-based object detection and tracking for an unmanned product cabinet. *Expert Systems with Applications*, 143.

Kim, I. and Yow, K. C. (2015). Object location estimation from a single flying camera. In *UBICOMM 2015*, page 95.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings NIPS'12*, pages 1097–1105, USA.

Kuznetsova, A. and at. al. (2018). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale.

Li, Z. and at. al. (2018). Large-scale retrieval for medical image analytics: A comprehensive review. *Medical Image Analysis*, 43(10).

Liu, W. and at. al. (2016). Ssd: Single shot multibox detector.

Lu, Z. (2018). *Client- server system for web-based visualization and animation of learning content*. PhD thesis, Darmstadt Univ. of Tech., Germany.

Oztarak, H. and at. al (2016). Efficient active rule processing in wireless multimedia sensor networks. *I. Journal of Ad Hoc and Ubiq. Computing*, 21:98–136.

Redmon, J. and at. al. (2016). You only look once: Unified, real-time object detection. 32(3):779–788.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement.

Ren, S. and at. al. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. In *Proceedings of NIPS'15*, pages 91–99.

Rincon, L. and at. al. (2019). Adaptive cognitive robot using dynamic perception with fast deep-learning and adaptive on-line predictive control. *Advances in Mechanism and Machine Science*, 73:2429–2438.

Szegedy, C. and at. al. (2016). Inception-v4, inception-resnet and the impact of residual connections on learnings.

Tan, C. and at. al. (2018). A survey on deep transfer learning.

Tian, R. and at. al. (2018). Novel automatic human-height measurement using a digital camera. *2018 IEEE BMSB*, pages 1–4.

Wang, Y. (2014). Tan analysis of the viola-jones face detection algorithm. *An Analysis of the Viola-Jones Face Detection Algorithm*, 4:128–148.

Wei, P. and at. al. (2018). Lidar and camera detection fusion in a real time industrial multi-sensor collision avoidance system.

Yang, Y. and at. al. (2020). A trajectory planning method for robot scanning system using mask r-cnn for scanning objects with unknown model. *Neurocomputing*.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE*, 22(11):1330–1334.