Distance Estimation using Tensorflow Object Detection

Faizan Amin, Syed Ali Mohd Murtaza Mehdi, Ahmad Danish Khan

Department of Computer Engineering, Aligarh Muslim University

faizanamin135@gmail.com

ali_mehdi_1996@yahoo.com

danish21752@gmail.com

Abstract— **Today, as a civilization we produce an unprecedented amount of data, in the form of audio, images, video and so forth. One application of this data-driven approach is autonomous vehicles. The current technologies have made it possible to happen. The top car companies like Hyundai, Kia, Ford motors, Tesla motors have been working on the self-driving cars projects and they have achieved it to some extent. But self-autonomous vehicles are not only limited to the self-driving cars, but the UAVs (Unmanned Aerial Vehicles) are also part of it. While the applications of the self-driving cars are somewhat limited to the usage of normal public, the UAVs have applications that vary from surveillance to patrol to enemy reconnaissance, in short, the UAVs have more applications for the military than the normal public. These autonomous vehicles require an understanding of the environment they operate in. As these vehicles are used to travel in cities (in case of self-driving cars) and also might be used in forests or mountains (in case of UAV use by for reconnaissance), they require to detect obstacles in order to avoid them. This is often achieved through scene depth estimation, by various means. We propose an approach which not only requires a minimum amount of space but also consumes far less power. Our approach is based on Obstacle Detection and calculating distance using the disparity estimated. These represent highly desirable features, especially for micro aerial vehicles.**

Index Terms - Tensorflow object detector, disparity, depth, bounding box, stereo images.

I. INTRODUCTION

Estimating depth is very important if we want to understand the geometry of a scene. For autonomous vehicles like self-driving cars and UAVs, estimating depth is the most important step in the process of their obstacle avoidance. In the case of UAVs, obstacle collision avoidance is usually carried out based on obstacle perception [1]. Also nowadays since the UAVs are improved, more and more tasks are being carried out using multiple UAVs. Thus, in this case, cooperative control of UAVs is needed to perform obstacle detection and avoidance for all the UAVs [1]. While when we talk about the self-driving cars, Google car and Benz-S class are the ones who are ahead in the race of intelligent cars [2], [3]. In these cars along with depth estimation, speed estimation is also important to avoid obstacles and this can be done using text detection of the number plate of the vehicle, using which the vehicle can be tracked and so its speed can be estimated [4]. At the same time, object detection is important as to know what kind of obstacle is in the way of the vehicle. It is important because obstacles are usually of two types: they may be static or dynamic. Static obstacles are those objects that do not move, like the trees, traffic lights, or even buildings. Whereas dynamic obstacles are those which are in motion. Although detecting both kinds of obstacles are important, the detection of the dynamic ones is more important and more challenging. It is so because at one instance these objects might not be in the view of the sensor or the camera or laser used, and they might come in the view suddenly and so in some situations, if the detection happens slightly slowly, then the obstacle avoidance becomes more difficult for the vehicle; but if detection happens timely, it helps the vehicle to make a decision about the obstacle avoidance. For example, if a car is moving in front of a vehicle or even if a person is crossing the road in front of a vehicle, the vehicle might not need to avoid them if they are at a safe distance. Thus along with depth estimation, object detection is also important, as it gives the vehicle idea about what kind of obstacle it is facing and also helps it to decide whether to avoid that specific object or not. In our approach, we use stereo images. The stereo images captured are then used for object detection. We then calculate the disparity for the detected objects. Using the calculated disparity we further calculate the distance of those objects from the camera which can simply be also termed as "depth".

II. RELATED WORKS

Several approaches have been proposed in the past for the same problem, i.e. finding the depth or distance of an object from the viewpoint.

A. Pollefeys et al., 2008, used a method where a single camera for the sensor system was used for the existing image-based 3D reconstruction [5]. This method had several limitations which were addressed by the use of a stereo camera set (Fathi & Brilakis, 2014) [6].

B. In the stereo vision applications, the configuration of the two-camera calibration setup is important. Each camera in this setup captures the image which has the 2D representation of the view. 3D information is being extracted by processing the two 2D representations captured by the two cameras. This extraction creates a map, which describes which point in the two 2D images corresponds to the 3D one (Calin & Roda, 2007) [7].

C. Santoro, AlRegib, and Altunbasak (2012) proposed that the natural disparity between stereo views is incorporated into a constrained motion estimation framework [8]. Their method accurately estimates synthetic misalignments due to translation, rotation, scaling, and perspective transformation. This helped in reducing the depth errors because of camera shifts.

D. The comparison of stereo visions could be done by digital laser range-finder and this was proposed by Mahammed [9].

E. Setyawan et al., 2018 aimed to find a correlation between the accuracy of distance measurement and the baseline distance between the two cameras [10]. Setyawan et al. had proposed a trigonometric formula with which they would calculate the distance after evaluating the disparity. Also, they had used Semi Global Block Matching called SGBM to minimize the Global matching [11].

Our approach is different from the ones mentioned above. Our main objective is to detect the objects in the stereo pair of images captured and then find the distance of those specific objects from the camera. To achieve this, we find the distance by initially evaluating the disparity between the objects captured in images of both the camera.

III. Model Development

Our proposed model architecture is shown in the figure 1. We kept our research limited to images in which Tensorflow object detector can detect only one object [12]. According to TensorFlow official website it is an open source library based on the design Google Brain Team. Its purpose is mathematical computations using a graph model. In this graph, nodes are operations and edges are multidimensional arrays (tensors). We fixed the two cameras at a baseline distance *B*. It is also important to ensure that both cameras are in a perfect parallel position [13]. First, the images captured from the left and the right camera are used as an input by the object detector. The left and the right images are captured at the same focal length and resolution. The object detector detects the object in the images and creates a bounding box around the target object.

After this step, the bounding box central x-coordinate calculator calculates the x-coordinate of the center of bounding box of the target object for both the pair of images. This operation is carried out by first computing the Xmin and Xmax coordinates of the bounding box and then averaging them. The XL and XR coordinate of left and right images respectively, obtained are passed to the disparity calculator where the disparity d in the target object is calculated using the formula:

$$d = XL - XR \text{ --- (1) [14]}$$

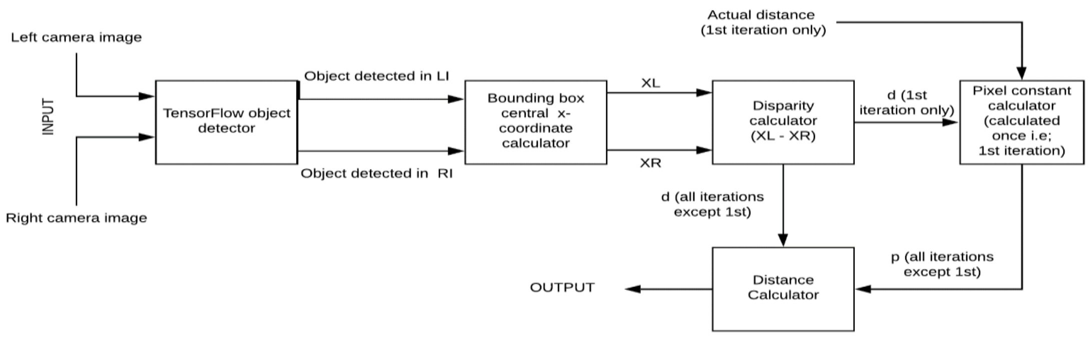
(Note that we are considering disparity due to horizontal shift in the position of the object.). Both the cameras are aligned in a way such that the line joining them is parallel to the x-axis (baseline) to minimize the lateral shift in the position of the target object. Now the value of disparity either goes for the calculation of pixel constant p or skips this part and directly jumps to the distance calculator depending upon the iteration number.

- In the first iteration, p is calculated. We want to calculate p because disparity depends upon the image width and height which is in pixel. Therefore each time we change the image resolution, the difference between XL and XR would change and as a result, the disparity changes. This is why we multiply disparity by pixel constant so that each time there is a change in disparity due to resolution, we can adjust it in the formula in a way such that we obtain the actual distance. Therefore we used equation (3) instead of equation (2) (gives relative distance).

$$D = \frac{B \times f}{d} \cdots (2)\ [15], [16]$$

$$D = \frac{B \times f}{d \times p} \cdots (3)$$

Using equation (3), we calculate p in the first iteration. We put the actual value of distance in the equation along with baseline distance *B*, focal length *f*, and disparity *d*.



Actual distance (1st iteration only)
Left camera image
INPUT
Right camera image
TensorFlow object detector
Object detected in LI
Object detected in RI
Bounding box central x-coordinate calculator
XL
XR
Disparity calculator (XL - XR)
d (1st iteration only)
Pixel constant calculator (calculated once i.e; 1st iteration)
d (all iterations except 1st)
p (all iterations except 1st)
Distance Calculator
OUTPUT

Fig. 1: Model architecture

- For every iteration except the first one, the value of disparity directly passes to the distance calculation section since p is already known by now. **Also, it is important to note that pixel constant p is to be evaluated just once during the first iteration and after that, we need not calculate it for other iterations. For example, in the case autonomous vehicles, it will be the manufacturer who will evaluate the pixel constant p while setting up the stereo cameras in the vehicle. Afterwards, it is ready to calculate the actual distances from the obstacles without any need to re-evaluate the pixel constant (as it was already evaluated and is being used to calculate the distance).**

Finally, the distance calculator section calculates distance using the formula:

$$D = \frac{B \times f}{d \times p}$$

(Note that we already used the same formula for calculating pixel constant once, but now we use it for the calculation of distance as by now other quantities are known.)

IV. Experimental Setup

In this section, we discuss the experiment that we conducted after creating our proposed model. For this, we used a deodorant bottle as a target object which was detected easily by the object detector. We used two Samsung galaxy A50 mobile cameras and placed them at a baseline distance of 30 cm from each other.

First, we took an image of the bottle from the left camera and the right camera at a resolution of 3024x4032 pixel and a focal length of 3.93mm. As shown in figure 2(a) and figure 2(b), the object detector detected the bottle in both the images captured from the left camera and the right camera. Using the information of the target object produced by the object detector, our proposed method estimated the central *XL* and *XR* coordinate of the bounding boxes of the left and the right images respectively and subtracted *XL* and *XR* to get the disparity in pixel.

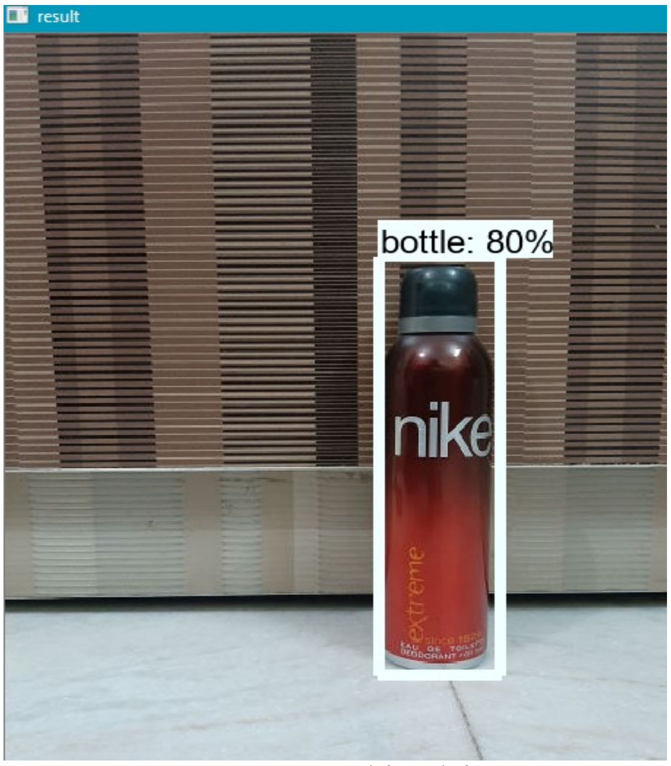


Fig. 2(a): Image captured from left camera.

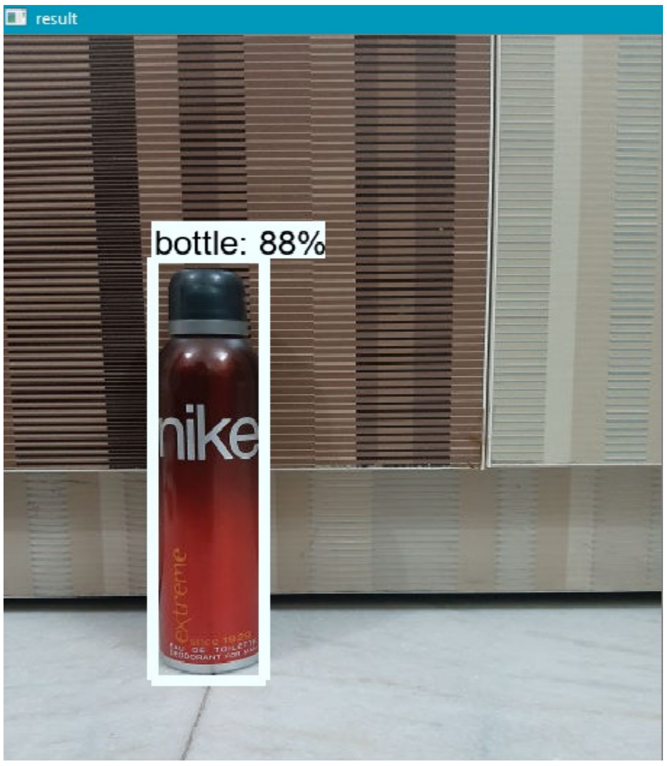


Fig. 2(b): Image captured from right camera.

a) Calculation

In the first iteration, the model used the actual distance between the object and cameras to estimate the value of pixel constant *p*. The calculation performed by the model is shown below:

$$D = \frac{B \times f}{d \times p}$$

$$50.80 = \frac{(0.393 \times 30)}{1863.50 \times p}$$

$$p = 1.24543393707 \times 10^{-4} cm/pixel$$

Here 50.80 cm is the actual distance provided by us, 1863.50 pixel is the calculated disparity by our model and we kept cameras at a distance of 30 cm from each other.

For the second iteration, the model estimates the distance using the same formula for another pair of images. The model uses the same value of pixel constant calculated by the first iteration, assuming that the resolution of the second pair of images is kept the same (3024x4032 pixels). Also this time the cameras were placed at a baseline distance of 10 cm from each other instead of 30 cm.

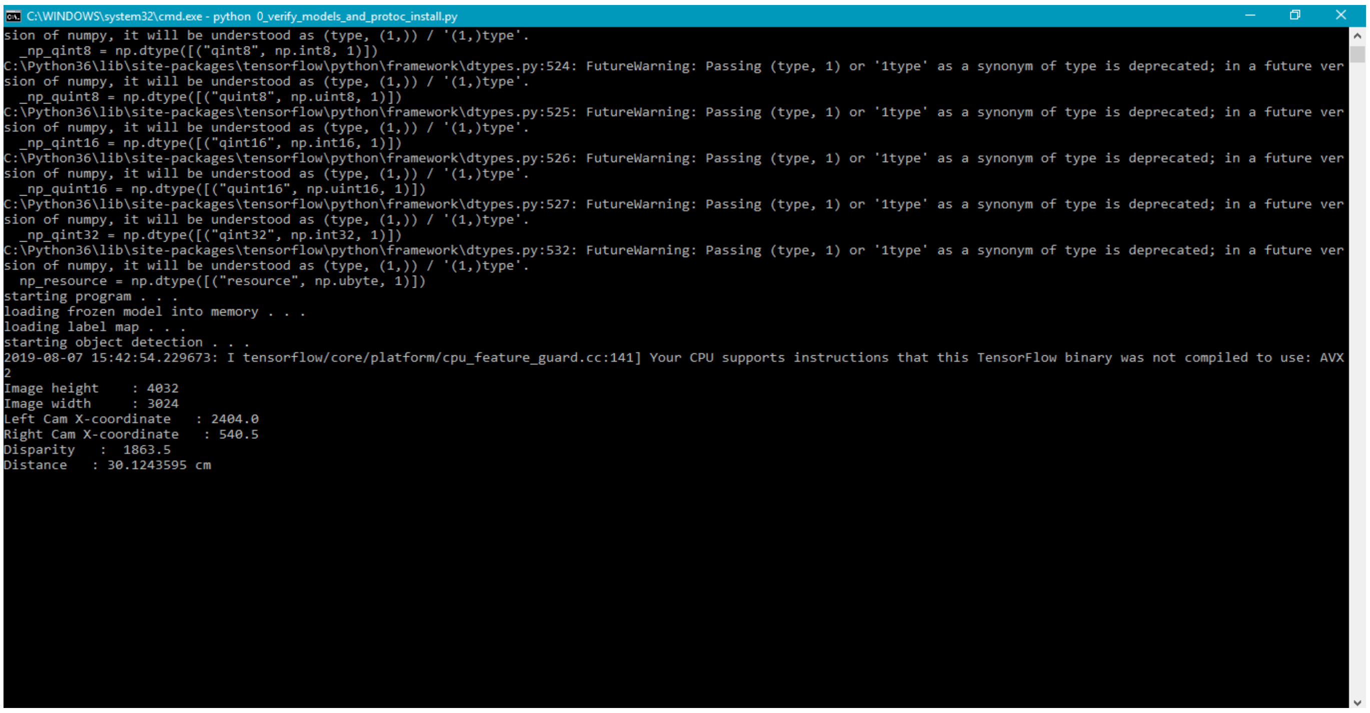
$$D = \frac{B \times f}{d \times p}$$

$$D = \frac{(0.393 \times 10)}{1863.50 \times (1.24543393707 \times 10^{-4})}$$

$$D = 30.12\, cm$$

b) Performance

The results we obtained after our method computed the distance were shockingly good. The calculated distance was 30.12 cm, which is only 0.12 cm more than the actual distance (In the second iteration we placed the target object at an actual distance of 30 cm from cameras). We also changed the baseline distance in the second iteration and it wasn't affecting our results which shows the adaptability of our model.


C:\WINDOWS\system32\cmd.exe - python 0_verify_models_and_protoc_install.py

```
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Python36\lib\site-packages\tensorflow\python\framework\dtypes.py:524: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future ver
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Python36\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future ver
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Python36\lib\site-packages\tensorflow\python\framework\dtypes.py:526: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future ver
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Python36\lib\site-packages\tensorflow\python\framework\dtypes.py:527: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future ver
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Python36\lib\site-packages\tensorflow\python\framework\dtypes.py:532: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future ver
sion of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
starting program . . .
loading frozen model into memory . . .
loading label map . . .
starting object detection . . .
2019-08-07 15:42:54.229673: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX
2
Image height    : 4032
Image width     : 3024
Left Cam X-coordinate   : 2404.0
Right Cam X-coordinate   : 540.5
Disparity   :  1863.5
Distance   : 30.1243595 cm
```

Fig. 3: Distance evaluation.

Cases	Resolution	Pixel constant (p) in cm/pixel	Actual distance	Calculated distance	Baseline distance (B)	Error %
1	3024x4032	1.24543×10^{-4}	15.00 cm	15.05 cm	5 cm	0.33
2	3024x4032	1.24543×10^{-4}	30.00 cm	30.12 cm	10 cm	0.4
3	3024x4032	1.24543×10^{-4}	30.00 cm	30.15 cm	30 cm	0.5
4	453x605	8.14338×10^{-4}	51.10 cm	50.09 cm	30 cm	1.97
5	453x605	8.14338×10^{-4}	60.96 cm	61.80 cm	30 cm	1.38
6	3024x4032	1.24543×10^{-4}	91.44 cm	93.20 cm	30 cm	1.92
7	453x605	8.14338×10^{-4}	121.93 cm	125.00 cm	30 cm	2.51
8	3024x4032	1.24543×10^{-4}	182.00 cm	185.60 cm	30 cm	1.98
9	3024x4032	1.24543×10^{-4}	250.00 cm	263.15 cm	30 cm	5.26
10	453x605	8.14338×10^{-4}	300.00 cm	309.90 cm	30 cm	3.3
11	3024x4032	1.2454×10^{-4}	350.00 cm	362.25 cm	30 cm	3.5
12	453x605	8.14338×10^{-4}	400.00 cm	434.00 cm	30 cm	8.5
13	453x605	8.14338×10^{-4}	450.00 cm	492.20 cm	30 cm	9.37
14	3024x4032	1.2454×10^{-4}	500 cm	576.50 cm	30 cm	15.5

Table 1: Actual and calculated object distances and the percentage error of the results.

V. Results

To evaluate the performance of the proposed method, we conducted some tests to find out the distance of the target object from the cameras. Although we were conducting tests on those images in which the object detector was able to detect only one object, we captured many different objects (one at a time) and selected them as the target object to make sure that the method was efficient. We found out that when the object was placed at a fairly close distance, we were getting a very minute error in the calculated distance.
We conducted some more tests to determine the farthest distance up to which our proposed method would yield fairly accurate results and our observation showed that it could predict precisely up to 3.5 m. Figure 3 shows the output distance estimated by the model in the command prompt screen.

The specifications of the apparatus we used for testing are as follows:

Camera used – Samsung galaxy A50 cell phone camera (as left camera and right camera).
Image resolution – 3024x4032 pixels and 453x605 pixels
Focal length – 3.39 mm
Calculated pixel constant –1.2454×10^{-4} cm/pixel (3024x4032)
8.14338×10^{-4} cm/pixel (453x605)

Table 1. shows the actual distance, calculated distance and the percentage error in distance calculation. It shows that our method provides highly accurate results up to a distance of 3.5 m after which our results start deviating from the expected value. Moreover, we also changed the baseline distance between both the cameras keeping other parameters fixed and observed that the calculated distances in both cases were almost the same (case 1 and case 2). This proves that our model is flexible to a great extent. Furthermore, we also observed a pattern in the estimated value of errors. As shown in figure 4, the magnitude of error is increasing with an increase in the target object distance exponentially.

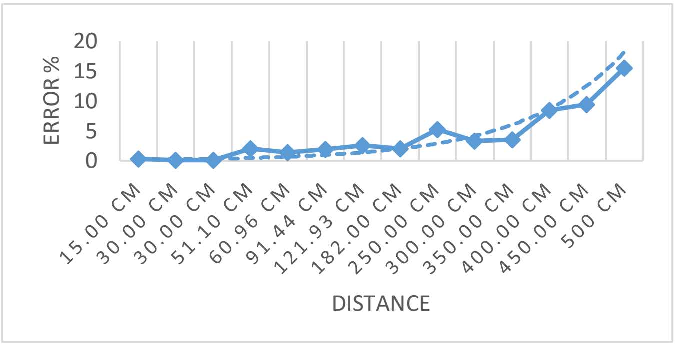


Fig. 4: Percentage error in distance calculation of the target object.

VI. Limitations

Our model comes with two main limitations, they are as follows:

- Firstly our model gives near to accurate results up to a distance of 3.5 meters. But after that we see error to be increasing nearly exponentially.
- Also, our model will only work when only one object has been detected. If more than one object is being detected, then ambiguity is being faced as to which bounding box is supposed to be selected. One solution for this can be we use the label as the differentiating factor, but that can also lead to ambiguity if the objects have the same label. The same problem is with the confidence score (shows certainty level of the object detector for the detected object) because that too can be the same for the objects detected.

VII. Conclusion And Future Works

As shown in the results and also explained in the experiment done, our model does get near to accurate and favorable results until a distance of 3.5 meters. The images were captured at the same time by both cameras and were captured in a way such that the object detector can only detect one object. We did that because after detecting several objects, their labels might be the same, the confidence score (which is the accuracy by which an object is detected), can be the differentiating factor. But in certain situations, even the confidence score of detection can be the same and in this case, ambiguity arises. Coupling the object detection with distance estimation will help in determing the type of obstacle present and will help in making better decision regarding obstacle avoidance. In our future work, we are hopeful to find a way to address the problem of multiple objects detection. We are also planning to expand our work by trying to find the distance of the objects from the camera using "Video" captured. Also, here we used stereo images, but using a monocular image for the same will be more cost-efficient while also being challenging. We are also planning to expand this method to the monocular image where we would reconstruct the stereo pair from the monocular image and then apply the same method.

References

[1] F. Bin, F. XiaoFeng and X. Shuo, "Research on Cooperative Collision Avoidance Problem of Multiple UAV Based on Reinforcement Learning," 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, 2017.

[2] P. E. Ross, "Robot, You Can Drive My Car - Autonomous Driving will Push Humans into the Passanger Seat," IEEE Spectrum, June, 2014.

[3] S. Salahat, A. Al-Janahi, L. Weruaga and A. Bentiba, "Speed Estimation from Smart Phone In-Motion Camera for the Next Generation of Self-Driven Intelligent Vehicles," 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), Sydney, NSW, 2017.

[4] D. C. Luvizon, B. T. Nassu and R. Minetto, "Vehicle Speed Estimation By License Plate Detection And Tracking," in IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP), Florence, Italy, 4th – 9th May, 2014.

[5] Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., . . . Merrell, P. (2008). Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision.

[6] Fathi, H., & Brilakis, I. (2014). Multistep explicit stereo camera calibration approach to improve euclidean accuracy of large-scale 3D reconstruction. Journal of Computing in Civil Engineering.

[7] Calin, G., & Roda, V. (2007). Real-time disparity map extraction in a dual head stereo vision system. Latin American applied research

[8] Santoro, M., AlRegib, G., & Altunbasak, Y. (2012). Misalignment correction for depth estimation using stereoscopic 3-d cameras. Paper presented at the Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on.

[9] M. a Mahammed, A. I. Melhum, and F. a Kochery, "Object Distance Measurement by Stereo Vision," International Journal of Science and Applied Information Technology. 2013.

[10] R. A. Setyawan, R. Soenoko, P. Mudjirahardjo and M. A. Choiron, "Measurement Accuracy Analysis of Distance Between Cameras in Stereo Vision," 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Batu, East Java, Indonesia, 2018.

[11] R. A. Setyawan, R. Sunoko, M. A. Choiron, and P. M. Rahardjo, "Implementation of Stereo Vision Semi-Global Block Matching Methods for Distance Measurement," vol. 12, no. 2, pp. 585–591, 2018.

[12] www.tensorflow.org,'Object Detection'. [Online] https://www.tensorflow.org/lite/models/object_detection/overview [Accessed : 13-07-2019].

[13] W. Sankowski, M. Włodarczyk, D. Kacperski, and K. Grabowski,"Estimation of measurement uncertainty in stereo vision system ☆," Image Vision Computing. 2017.

[14] 'Stereo and 3D Vision',06-03- 2009. [Online]. Available: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect16.pdf. [Accessed : 07-07-2019].

[15] Godard C., Aodha O. M & Brostow G. J. 2017. Unsupervised Monocular Depth Estimation with Left-Right Consistency.

[16] S. Solak and E. D. Bolat, "Distance estimation using stereo vision for indoor mobile robot applications," 2015 9th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, 2015.