Detection of Real Time Objects Using TensorFlow and OpenCV

Ajay Talele, Aseem Patil, Bhushan Barse
Department of Electronics Engineering, Vishwakarma Institute of technology, Pune, Maharashtra, India.
Email: ajay.talele@vit.edu, aseem.patil16@vit.edu, bhushan.barse16@vit.edu

Abstract: **Detecting and recognizing objects in unstructured as well as structured environments is one of the most challenging tasks in computer vision and artificial intelligence research. This paper introduces a new computer vision-based obstacle detection method for mobile technology and its applications. Each individual image pixel is classified as belonging either to an obstacle based on its appearance. The method uses a single lens webcam camera that performs in real-time, and also provides a binary obstacle image at high resolution. In the adaptive mode, the system keeps learning the appearance of the obstacle during operation. The system has been tested successfully in a variety of environments, indoors as well as outdoors, making it suitable for all kinds of hurdles. It also tells us the type of obstacle which has been detected by the system.**

Keywords – Object Detection, Image Edge Detection, Image Segmentation, object recognition, computer vision

I. INTRODUCTION

Obstacle detection is an important task for many mobile technological applications. Most mobile applications rely on range data for obstacle detection. Popular sensors for range-based obstacle detection systems include ultrasonic sensors, lasers, radar, stereo vision, optical flow, etc. Because these sensors measure the distances from obstacles to the robot, they are unavoidably suited for the tasks of obstacle detection and obstacle avoidance. But, none of these sensors are perfect. Ultrasonic sensors are cheap but suffer from optical and spectral reflections and usually from poor angular resolution. Lasers and radars provide better resolution but are more complex and more expensive. Moreover, stereo vision and optical flow are nearly computationally expensive. In addition to their individual deficiency, all range based obstacle detection systems have difficulty detecting small or flat objects on the ground. Accurate and reliable detection of these objects require high measurement accuracy and hence precise calibration. Range Sensors have a difficult time in detecting obstacles on the ground surface. This is a problem especially outdoors, where range sensors are usually unable to differentiate between the sidewalks and the adjacent flat grassy areas. While small objects and different types of ground are difficult to detect with range sensors, they can be easily detected with color vision. For this reason, we have developed a new appearance-based obstacle detection system that is based on the technology of artificial intelligence. The heart of the algorithm consists of detecting pixels different in appearance than the ground and classifying them as obstacles. The algorithm performs in real-time, providing a high-resolution of the obstacle image, and operates in a variety of environments. The fundamental difference between the range-based and thee appearance-based obstacle detection systems is the obstacle criterion. In range-based systems, obstacles are objects that bulge out a minimum distance from the ground. In appearance-based systems, obstacles are objects that are distinct in appearance from the ground.

II. LITERATURE SURVEY

There are two main important ways of detecting obstacles and objects in a system. They are namely Pytorch and TensorFlow.

PyTorch is an open source in the machine learning library for Python, based on the language Torch, used for applications such as natural language processing (NLP). It was primarily developed by Facebook's artificial-intelligence research group, and Uber's research group namely "Pyro" for probabilistic programming. Whereas, TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a representative math library, and is used for machine learning applications such as convolutional neural networks. TensorFlow follows the idiom ‘Data as Code and Code is Data’.

In TensorFlow you define the graph inertly before a model can run. All its communications with the outer world are performed with two sensors namely, tf.Session object and tf.Placeholder, which are tensors that will be substituted by an external data at its runtime. In PyTorch things are way more essential and dynamic, you can define, change and execute nodes as you edit and process the code, no special session interfaces or placeholders are required. Overall, the framework is slightly more tightly integrated with Python programming language and it feels more native most of the times. When you code in TensorFlow sometimes you feel that your model is behind a brick wall with several tiny apertures to communicate over. From the following points we can conclude that using TensorFlow the process is more efficient and is more reliable than Pytorch.

So we shall implement our project using TensorFlow.

III. APPEARANCE-BASED OBJECT DETECTION

Appearance-based object recognition methods have recently demonstrated better performance on a variety of problems. The system demonstrates extraordinarily good recognition of a variety of 3-D objects, ranging from cars and planes to snakes and lizards with full orthographic invariance. It has performed a number of large-scale experiments, involving over 2000 separate test images, that evaluate performance with increasing number of items in the database, in the presence of background change and occlusion, it is also the results of some the generic classification experiments where the system is tested on the objects never previously seen or modeled.

The basic idea is to represent the visual appearance of an object as the loosely structured combination of a number of local context regions keyed by distinctive key features, or fragments. Now under the different conditions the feature extraction process will find some of these distinctive keys, however, in general not all of them. Even with the local contextual verification, such keys may well be consistent with a number of global hypotheses. But, the fraction that can be found by existing feature extraction processes is frequently sufficient to identify objects in the scene, once the global evidence is assembled. This is one of the principle problems of object recognition, which is that, in any but rather artificial conditions, it has so far proved impossible to reliably segment whole objects on a bottom-up basis. In a current system, local features are based on automatically extracted boundary fragments are used to represent multiple 2-D views (aspects) of rigid 3-D objects, but the basic idea can be applied to other features and other representations.

IV. IMPLEMENTATION

A. *YOLO Weights in real object-detection:*

You only look once (YOLO) is an object detection system used for real-time image processing.

YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

Figure 1.1 Types of YOLO weights

1. *Advantages of YOLO:*

- ❖ Speed (45 frames per second) , faster than real-time
- ❖ Network understands generalized object representation
- ❖ It is a faster version with a smaller architecture of 155 frames per sec , but is less accurate.

2. *Limitations of YOLO:*

YOLO imposes strong spatial constraints on the bounding box predictions since each of the grid cells only predicts two boxes and can have only one class. This spatial constraint then limits the number of nearby objects that our model can predict. The model struggles with the small objects that appear in groups. Since the model learns to predict bounding boxes from data, it however struggles to generalize objects in new or unusual aspect configurations.

B. *Webcam/Camera installment:*

- For this module, we are going to be using OpenCV.

```
import cv2

video_capture = cv2.VideoCapture(0)
while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()
    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

```

Figure 1.2 Using OpenCV for installing camera

- We need to call the function cv2.VideoCapture(), and read the incoming frames. The .read() method is a blocking operation, so the main thread of our Python script is completely blocked until the frame is read from the camera device and returned to our script.

This is a problem, as it is critical for our system to run in real time. We can improve the FPS (frames per second) simply by creating the new thread that does nothing but pulls the camera for new frames while our main thread handles processing over the current frame.

```
class VideoStream:

        def __init__(self, src=0):
                self.stream = cv2.VideoCapture(src)
                (self.grabbed, self.frame) = self.stream.read()
                self.stopped = False

        def start(self):
                Thread(target=self.update, args=()).start()
                return self

        def update(self):
                 while True:
                        if self.stopped:
                                return
                        (self.grabbed, self.frame) = self.stream.read()

        def read(self):
                 # Return the latest frame
                return self.frame

        def stop(self):
                self.stopped = True
```

Figure 1.3 Connecting the video stream of the system by FPS

C. *Advantages:*

- The system can be used both indoor and outdoor surveillance..
- The optical flow calculation result is accurate because the GDIM base method is used and the gradient operator in polar-log coordinate is employed
- **"Excessively Smooth"** can be resolved by optical flow computation only used in moving area. And this method can be used for multiple objects tracking and real time object tracking. Finally, the experimental results prove that the proposed method in the paper is efficient to multiple objects detection.

Figure 1.4 Output of the system

V. EVALUATION

The system works well as long as the three assumptions about the environment stated in section three are not violated. We have experienced only few cases where the obstacles violate the first assumption by having an appearance similar to the ground. Including the additional visual clues that might decrease the rate of false negatives to an even lower number. A small inclination of a sidewalk introduces small errors in a distance estimate. This is not really a problem as the errors are small for obstacles that are close to the system. Our system however overestimates the distance to overhanging obstacles, which violates the third assumption. This distance can estimate error easily and lead to collisions, even when the pixels are correctly classified as obstacles .The simplest way to detect tabletops is probably to combine our system with a range-based sensor. For example, by adding one or two low-cost wide-angle ultrasonic sensors for the detection of tabletops would make our system much more reliable in office environment.

.

A. *Block Diagram*

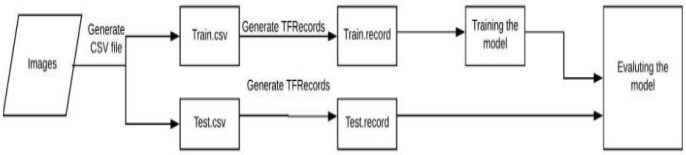

Figure 1.5 Block Diagram of training the model

B. *Algorithm of the System with VideoCapture*

Step 1: Load the required yolo.cfg and yolo.weights depending on its processing speed.

Step 2: Develop the TensorFlow graph and store it locally using TFNet.

Step 3: Capture the Video using OpenCV and break it into frames.

Step 4: While the frame exists predict and perform the suitable triggering actions on it.

Step 5: Once done with all the frames close Video capture.

C. *Applications of the system*

1. *Self-Driving Cars:*
   Self-driving cars square measure the longer term, there's little question in this.
   However the operating behind it's terribly tough because it combines a range of techniques to understand their surroundings, together with radio detection and ranging, optical device lightweight, GPS, odometry, and laptop vision.

Advanced management systems interpret sensory info to spot acceptable navigation methods, similarly as obstacles and once the image sensing element detects any sign of a living being in its path, it mechanically stops. This happens at a awfully quick rate and could be a massive step towards Driverless Cars.

2. *Security:*

Object Detection plays a awfully vital role in Security. Be it face ID of Apple or the membrane scan employed in all the sci-fi movies.

It is conjointly employed by the government to access the protection feed and match it with their existing information to search out any criminals or to observe the robbers' vehicle. The applications square measure limitless.

3. *Industrial Quality Check:*

Object detection is additionally utilized in industrial processes to spot product. Finding a particular object through visual review may be a basic task that's concerned in multiple industrial processes like sorting, inventory management, machining, quality management, packaging etc.

Inventory management is terribly difficult as things area unit arduous to trace in real time. Automatic object tally and localization permits rising inventory accuracy.

VI. RESULT

TABLE I.
Object Detection Results are as follows:

		Pixel level			Object level		
		P	**R**	**F**	**P**	**R**	**F**
Overall	Our	76.9	80	78.4	63.5	73.4	68.1
		81.9	73.7	77.6	61.9	66.9	64.3
Dataset1	Our	68.3	73.6	70.9	51.7	56.2	53.8
		69.9	66.9	68.3	46.8	54.9	50.6
Dataset2	Our	85.6	**85.9**	85.8	75	**92.5**	82.8
		94.9	**80.3**	87	81.8	**80.1**	81

(P) Precision (%), *(R)* Recall (%), *(F)* F-Measure (%)

Figure 1.6 Pixel level vs. Object level table

VII. CONCLUSION

TensorFlow has far better and easier support for saving and loading models across totally different environments and even programming languages. If loading and saving models is a priority, then TensorFlow comes in handy.

This paper presents a new method for obstacle detection with a single webcam camera. It also presents a new method of vision-based surveillance robot with obstacles avoidance capabilities for general purposes in indoor and outdoor environments. The algorithms of neural networks for obstacle detection and framing the image has been used. The camera used has a very poor resolution because it uses only 5MP. For future work, we will improve this system, to act as a surveillance system in robotic as well as future applications.

REFERENCES

[1] Moon, J. H. Kim and W. S. Lee, "Advanced Lane Detecting Algorithm for Unmanned Vehicle," Proceeding of the ICCAS2003, pp. 1130-1133, 2003.

[2] H. Park, Y. J. Son and J. H. Kim, "Design of Advanced Tele-operated Control System for Unmanned Vehicle," Proceeding of the ICCAS2005, pp. 915-919, 2005.

[3] 3, Cosmin D. Pantilie, Silviu Bota, Istvan Haller, Sergiu Nedevschi, "Real-time Obstacle Detection Using Dense Stereo Vision and Dense Optical Flow", 2010, ISBN 978-1-4244-8230-6/101\$26.00.

[4] Ryuzo Okada, Yasuhiro Taniguchi, Kenji Furukawa, Kazunori Onoguchi, "Obstacle Detection Using Projective Invariant and Vanishing Lines", IEEE International Conference in Computer Vision (ICCV, vol. 2, 2003.

[5] Jungwon Kang, Myung Jin Chung, "Stereo-Vision Based Free Space and Obstacle Detectionwith Structural and Traversability Analysis Using Probabilistic Volume Polar Grid Map", 2012 IEEE 5th International Conference on Robotics Automation and Mechatronics (RAM).