Wayfinding for Blind Persons

CS419/619 Computer Vision Course Project

Submitted by

Ashutosh Anshul - 170001011
Gumpili Sai Pranav - 170005010
Prayag Jain - 170001037

Under Supervision of

Dr. Surya Prakash
Associate Professor
Discipline of Computer Science and Engineering
Indian Institute of Technology Indore

Introduction

The work presents a system that aims to guide visually impaired people from source to destination. The system uses computer vision techniques to locate markers placed in surrounding areas, get suitable information, detect any obstacle in path and get some idea about the environment the user is in. The system is implemented in Python and mainly uses OpenCV for processing local scenes and suitable text-to-speech and speech-to-text libraries to convey information to the visually impaired user.

Background

One of the major difficulties that the visually impaired people experience the most is efficient and correct mode of an automated navigation from a source to a destination. This one big problem is actually a combination of multiple smaller problems existing simultaneously in a scene. These problems include, determining one's location, finding the direction from the current location to a destination location of choice, getting information about the surrounding the user is present in like objects present around, and obstacle detection and avoidance while navigating.

The aforementioned problems can be tackled using suitable and problem-specific computer vision algorithms and image processing techniques. A physical prerequisite for solving this problem includes installing QR code markers at selected locations to hold information regarding locations and directions. QR code seems to be the most optimal solution for selecting a coloured marker considering the ease of generation, uniqueness to information coded inside it, cost of installation and ease of coding information in form of images.

In our work we have tried to solve most of the problems using the QR code markers and a pipeline of the above group of existing algorithms efficiently. We have tried our pipeline to be dynamic and robust regarding different scene set ups, and different times of the day. The dynamicity of the problem statement made quantitative analysis improper for system testing and so we performed qualitative, real-time check of our model. This test includes a check over individual components of the model and an overall analysis of model performance.

System description (Work done)

The system consists of the following three major components, each working independently to solve one sub problem of the actual problem statement:-

1. QR code marker detector and decoder
2. Obstacle detector
3. Room Scanner

QR Code Marker Detector and Decoder

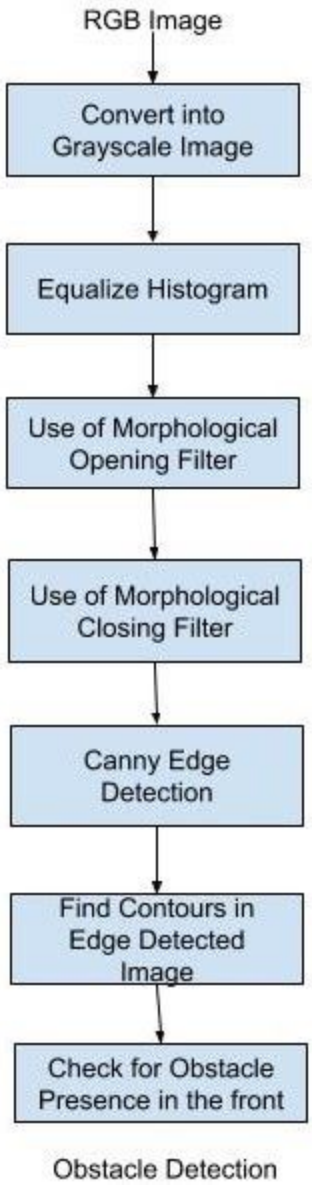
This component is actually meant to notify the user about his/her current location and the direction to the next marker in his path to destination. The QR code as mentioned above is an optimal answer to code multiline textual information in the form of images that are unique for each text encoded into them also supported by inexpensive and easy installation.

The QR code detection and decoding can be done using OpenCV functions and upon the decoding suitable text-to-speech libraries can be used to convey the information to the user in the form of audio. The QR code used here is a 7cmX7cm image. Practically we can even use bigger QR codes that increase the distance and angle range over which it can be easily detected and decoded. It solves the positioning and wayfinding problem.

Obstacle Detector

This section of the system, as the name suggests, is used to detect any obstacle present in front of the user while navigation. It runs after a fixed interval of frames so as to reduce the complexity of the algorithm. A continuous frame-by-frame check is unnecessary as there is no drastic change in the scene while the user is walking and in other words will lead to redundant checking of the frames.

This problem is solved using edge detection based segmentation techniques. The process includes converting input color image into grayscale image, followed by morphological opening and closing operations. Then Canny edge detection is used on the scene. At this



Obstacle Detection

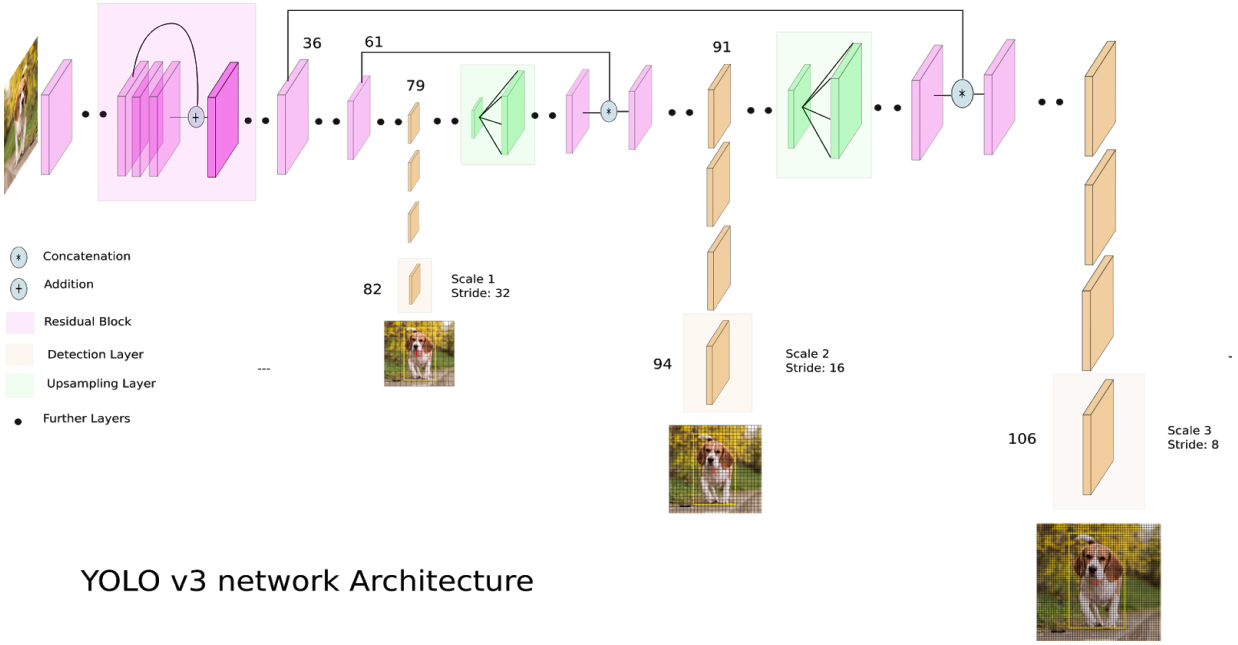
point we had two options, either we can go through region growing segmentation technique where we could have used flood fill given a seed point. But some missing part of the edge would have caused the flood fill to give incorrect region filling. So, we moved with a second way of finding all the contours in the detected edges and using the contours to detect objects. A flow of the stated method is given on the left side.

Once all the contours are detected, we can look at the specific front region window of the image and look for the amount of white pixels present in the window. The presence of a large amount of white pixels would indicate the presence of obstacles and would inform the user using text-to-speech. It solves obstacle detection and avoidance during navigation.

Room Scanner

This part of the system runs to detect the objects that are present in the surrounding environment, and their direction of placement relative to the user. This actually solves the problem of localization. It uses the YOLO version 3 object detector model. You Only Look Once(YOLO) is one of the faster real-time object detection algorithms. Since the YOLO model uses heavy neural networks architecture, we have restricted the use of YOLO to only when the user initiates it using speech i.e. says “SCAN”. When Yolo runs since we have the locations of objects recognised we also use TTS (Text To Speech) to tell the blind whether the object is in **left**, in **front** or the **right** of the person by speaking it out loud.

Given below is the architecture of YOLO v3. It includes the Darknet 53 model followed by more convolution and fully connected layers for detection giving a total of 106 layers.



YOLO v3 network Architecture

Datasets used

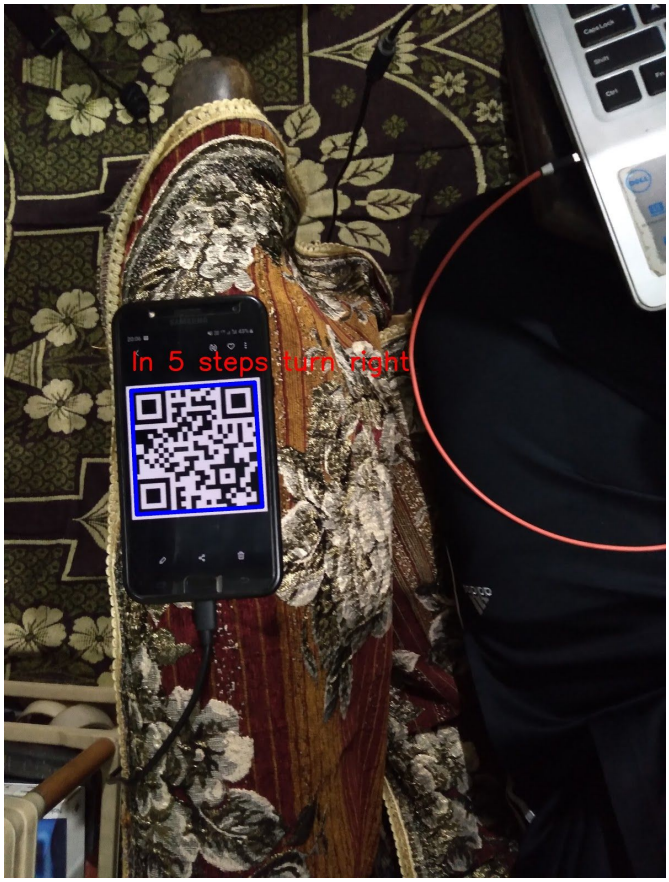
There already exists a predefined function for QR code detection in OpenCV, so we need not to train any such model for the same.

For checking the performance of our obstacle detection, we created our own dataset of around 10 images taken at our homes. The images were of different locations in two different houses, and were taken at different times of the day in order to maintain the dynamicity and generalization of the dataset and ensure robust performance.

The pre-trained YOLO model used was trained using the standard COCO dataset. It is a vast dataset that can be used for large-scale object detection, segmentation, and captioning. It consists of 80 object classes, 91 stuff categories, 330K images in which more than 200K are labeled, 1.5 million object instances, 5 captions per images and 250,000 people with keypoints.

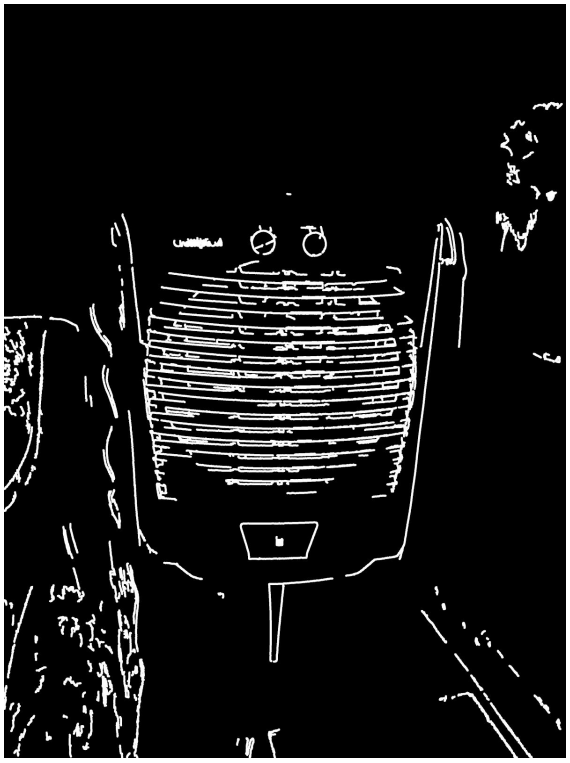
Results and Analysis

In order to test the performance of the model, we have used a qualitative analysis of two kinds. One, checking the performance of each of the components individually over the generated dataset, and trying the whole model over a real time scenario. The images are taken at night and the video in the day, showing the robustness of our model during different times of day in presence of different illuminations.

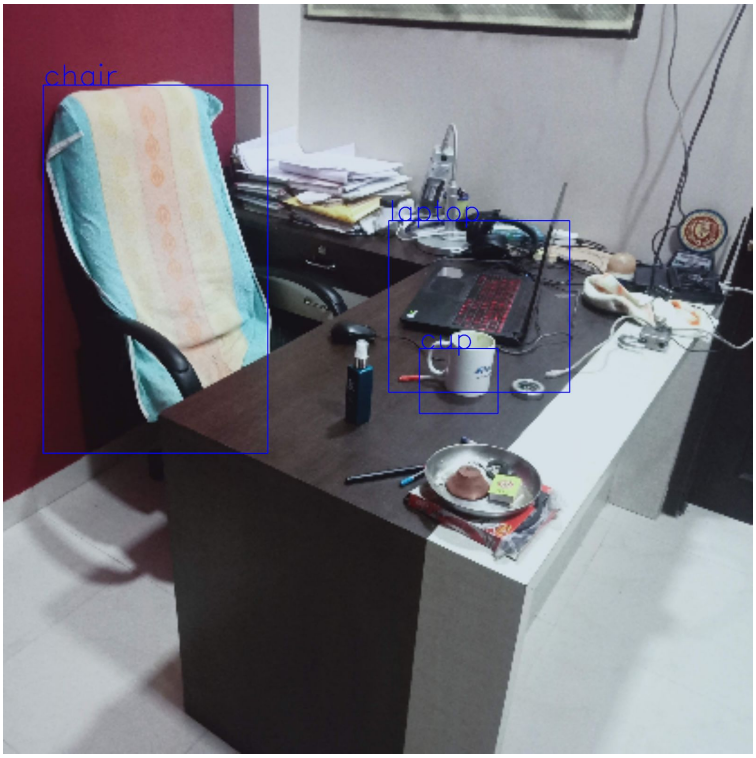


The figure shows the detection of the QR code bound in the blue box along with its decode text.

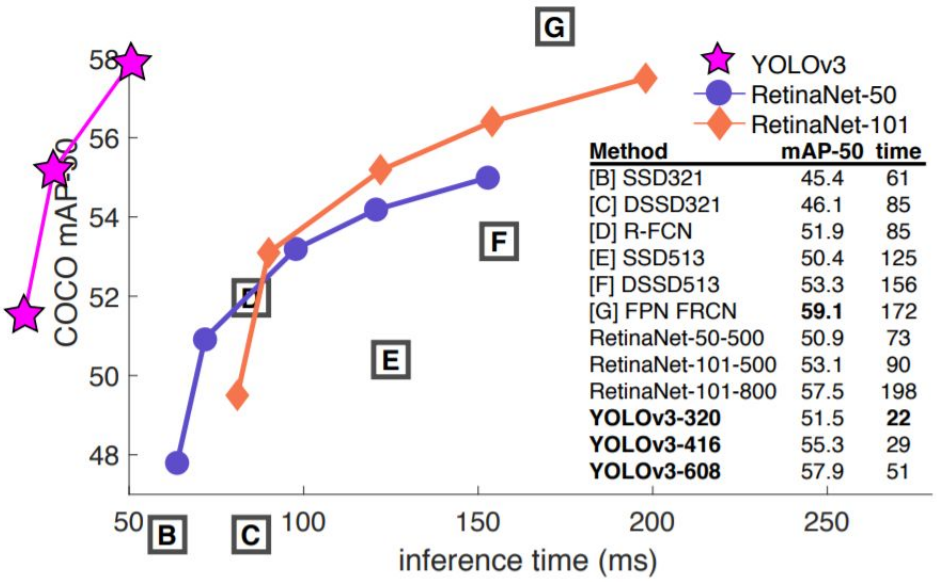
The above images show scenes with no obstacles in the front and their detected contours after applying canny operation. Notice that the white pixels are very less in the central and lower middle portion of the image



The above images show scenes with obstacles in the front and their detected contours after applying canny operation. Notice that the white pixels are in high to very high range in the central and lower middle portion of the image as compared to last case



Objects detected in the surrounding using YOLOv3 when the user says “SCAN”



The above graph compares various object detection models using their Speed(Inference time)-Accuracy(mean Average Precision- mAP) tradeoff curve. It shows that YOLOv3 is best among them as in terms of time it is far to left and in terms of precision, it is higher. So this supports choosing YOLOv3 for object detection.

Conclusion

The above results show that each of the individual components are working up to their expectations and are robust with respect to the location and illumination. The model takes real time video feed, performs necessary computations and gives efficient and appropriate outputs making our model end-to-end. It tackles all the major problems faced in blind navigation and primarily uses speech guidance. Similar solutions to our Problem Statement use Different markers but we opted for QR code since it's more easily detectable and more information can be encoded. Future works include distance calculation of objects using markers or other methods, and also make the model robust in unconstrained environments i.e without QR code or other noises. The QR detection performance can also be increased to ensure higher distance and angle range.

References

1. Manduchi, Roberto & Kurniawan, Sri & Bagherinia, Homayoun. (2010). Blind guidance using mobile computer vision: A usability study. 241-242. 10.1145/1878803.1878851.
2. A.Nayak, Ankitha & S, Venugopala & H., Sarojadevi & Chiplunkar, Niranjan. (2015). An Approach to Improvise Canny Edge Detection using Morphological Filters. International Journal of Computer Applications. 116. 38-42. 10.5120/20368-2575.
3. Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement.
4. Liu, D. and J. Piersol. “Real-Time Segmentation , Tracking , and Coloring of Walls Using iOS.” (2012).
5. https://docs.opencv.org/master/de/dc3/classcv_1_1QRCodeDetector.html
6. https://github.com/experiencor/keras-yolo3

Notes

1. Please keep the Sound ON(headphones recommended) as it involves text-to-speech for the person to hear instructions.
2. The output video seems to be a bit slow. This is due to the screen recorder running in the background. In general the system runs smoothly and efficiently over the video feed.