Designing a navigation system for the visually Impaired

Computer Vision CSCI-B657

Github code link: https://github.iu.edu/natarajr/BlindAssistance

Nagarajan, Ganesh
gnagaraj@umail.iu.edu
Nataraj,Raghavendra
natarajr@umail.iu.edu
Sankaranarayanan,Arun Ram
arunsank@umail.iu.edu

Objective

The objective of this project is to design a navigation model that could benefit the visually impaired to navigate safely in an unknown environment. This project aims to alleviate the usage of a comparatively less effective equipment like a walking stick that could serve the purpose of navigation for the visually impaired. The scope of this project is limited to indoors at the moment.

Motivation

The following are the major causes that influenced us to develop an effective navigation system that could improvise and help the purpose of navigation of a visually challenged person

- WHO survey of 2014 says that about 285 million people around the world are visually impaired
- Among the visually impaired people 90 % of people are in low income settings and about 82 % of people are aged above 50
- With usual equipment people mostly depend on their accumulated memories.
- The aim to create an affordable and most convenient wearable device that could help the visually challenged people.

Background

Related Work

Door Detection Algorithm

There were numerous amount of research that were being carried out to detect doors. We found [1] to be more related to our work in an indoor environment. The research paper discusses about how the geometry of a door can be used to better detect doors.
The paper makes use of the rectangular property of the door and carries out door detection by following the below steps sequentially

1. Calculate horizontal and vertical line counts after dilation
2. Detect corners using Harris Corner Detector
3. Overlay corners on image to get better intuition of door corners
4. Rectangular region detection
5. Estimate Canny Edge map

6. Extract dominant line features and filter out the rest
7. Apply 8-level otsu thresholding
8. Region counting to look for features with the size of knobs
9. Use fuzzy logic to calculate confidence
10. Adjust over various thresholds of length and width to obtain a higher confidence

Our Work

Door Detection Algorithm

Based on the paper [1] discussed above we designed a door detection algorithm to satisfy our needs. The constraint was to stick to indoors. With our approach we modified and added more constraints to the above approach to better detect doors. We applied Canny edge detector on the door image. We then performed corner detection and contour detection using OpenCV. We then implemented necessary heuristics based on the geometry of doors to effectively identify doors.

Requirements

1. Efficient Lighting
2. A scene with usual Indoor objects and Doors
3. Stereo Camera - Minoru 3 D webcam
4. Optimal orientation mechanism

Dependencies

1. Quality of the stereo camera
2. OpenCV

Assumptions

1. We assume that the environment is indoor. This limits us with the number of obstacles and depth view of the camera.
2. We set the viewing angles of the camera to a constant value to detect the doors and obstacles. This is to make sure only obstacles are being detected, as there are possibilities of patterns in doors and walls being detected as objects. Even though currently no obstacles are detected in floors in the environment we tested, there are possibilities that objects may be detected on floors. So we assume the viewing angles of the camera to certain angles only.

Preprocessing

Calibrating the stereo camera was an important preprocessing step. Without calibration the depth map was extremely noisy. We used the Camera Calibrator Application for calibrating the left and right cameras. The steps involved were

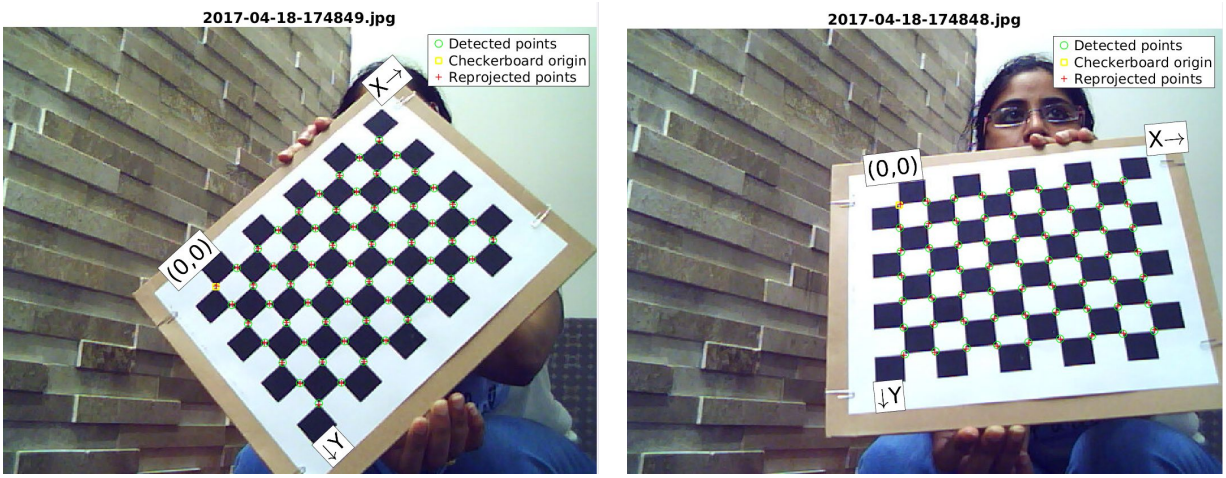


Fig 1 : Images from left camera of the StereoCam with MATLAB calibrations

1. Take a checkerboard image(Vector based image for straight lines)
2. Take multiple images with both left and right camera with different orientations and different translation.
3. Use the Camera Calibrator Application for each left and right images.

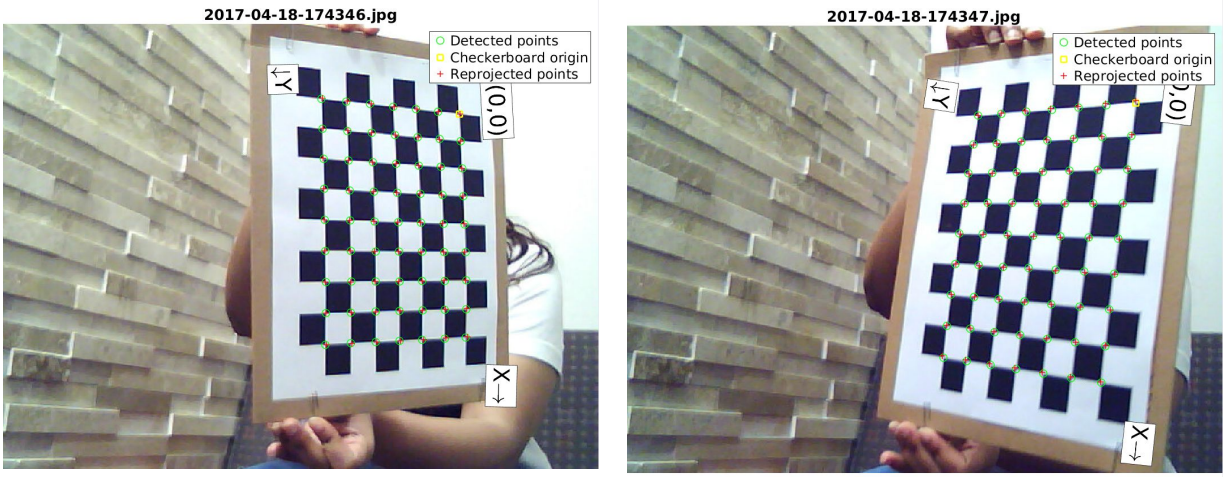


Fig 2 : Images from Right camera of the StereoCam with MATLAB calibrations

4. Remove the images which introduces a lot of error.
5. After removal calculate all the calibration parameters
6. The Calibration parameters include Intrinsic and Extrinsic parameters.
7. The intrinsic parameters includes the camera center(x,y) and focal length(x,y)
8. The extrinsic parameter includes radial and tangential distortions.
9. The calibration sessions and calibrations parameters for both cameras are stored as mat files and is available in the git repository

Implementation

The implementation mainly involves detection of doors and obstacles. Since we are looking to feed only the information about the objects and the location of the door to the visually impaired user, we are have followed the approach of detecting doors and obstacles so that the user can think and decide on an optimum navigation path

Detection of doors

We have use two methods for door detection

Fig 3 : Door detection process flow

Method One

The following are the sequence of steps that we followed in our first method for door detection. This method was our best method amongst our methods for door detection with less or no false positives given that the image scene is with optimum lighting.

1. Estimate edges using Canny edge Detector
2. Estimate corners using Harris Corner detector

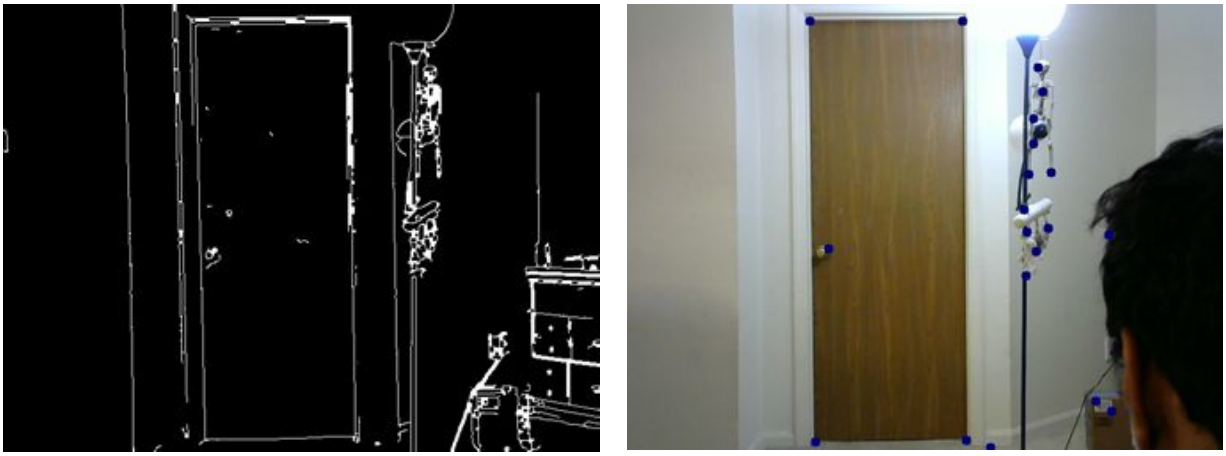


Fig 4 : Canny edge detection and Harris corner detection

3. Estimate contours using opencv
4. Compare Contours with corner points to look for optimal number of corner points which could lie within the threshold that we set for a rectangle

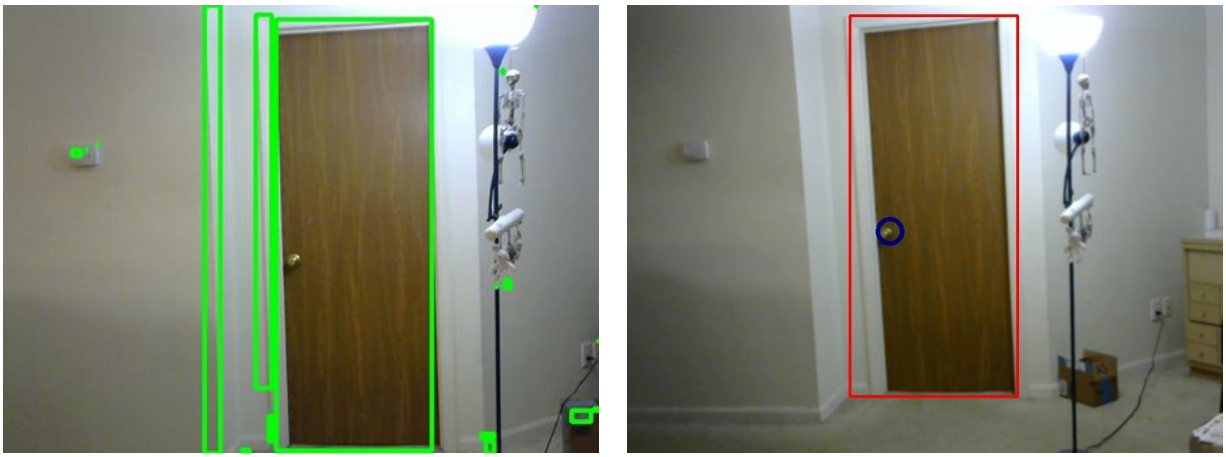


Fig 5 : Contour detection and rectangle overlay using heuristics

5. Overlay rectangles which satisfy the width and height threshold set.
6. Search for sub contours within the rectangles that were created
7. Use Euclidean distance to estimate the distance between points between these contours
8. Set thresholds between h/3 and h, where h is the height of the rectangle.
9. If the sub contour lies within this threshold then it is detected as a door knob.
10. Filter other rectangles as non-door candidates

Method One was significant in finding doors however we had couple of reasons to move for another methodology. First is open door detection. Method one uses knob as one of its heuristics to detect doors. So this methodology would not hold if doors are opened. Also development of method one was done in windows and when we migrated to linux however the same did not work.The contours were the same that were detected in windows and linux. We are still trying to identify the issue regarding portability between windows and linux.

Method Two

1. Identify rectangular contours in the frame.
2. Reject contours based on height and width.
3. Find Harris corners in the image.
4. Based on the geometry and assumptions that doors have knobs we choose regions within doors.
5. We try to identify if Harris corners are there in at least five of the regions .
6. We classify it as a door.

Open/Close Door

The next objective is to detect whether the door is open or closed so that we could get a better intuition of the state of the door. For this we decided to use the depth information we got with the help of our stereo camera. The results of the disparity were satisfactory, but if a better camera was used we could probably get better results of disparity.
The following are the sequence of steps that were followed when trying to determine the open and closed state of the door

1. Once door is found we pick a random point within the door

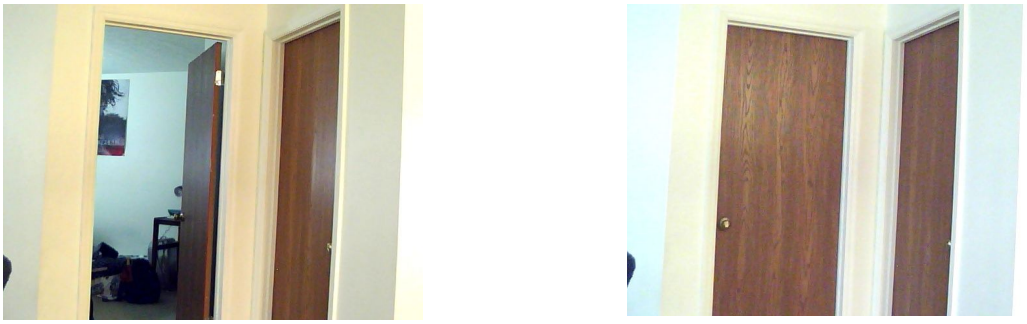
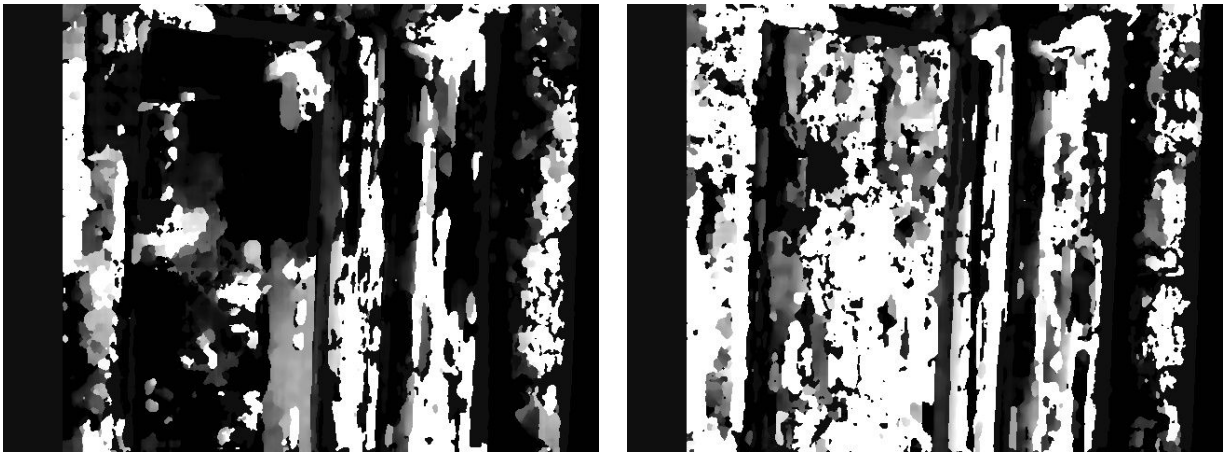


Fig 6: Door open and door closed frame that was captured

2. Average the depth in those point.
3. Closed doors have greater values than open doors



.

Fig 6: Disparity map for open and closed

4. As seen above there are lots of black pixels in the center on the left meaning the depth of the door is high and on the right there are a lot of white pixels at the center meaning that the door is closed and has a lesser depth
5. Based on a threshold we decide if the doors are opened or closed.

Detection of obstacle/Non-door elements

Obstacle Detection

Fig 7: Process flow for obstacle detection

1. K means segmentation based on color of the frame instance.

Fig 8: K-means segmented Image and feature points(FAST)

2. Find Contours in each color segment.
3. Find FAST descriptors in the Image frame.

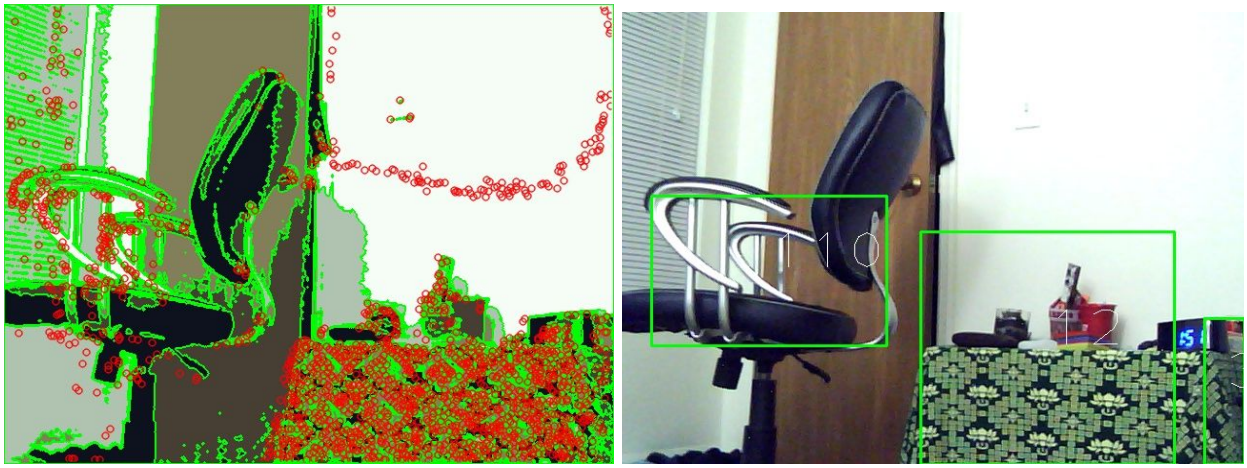


Fig 9: Contour detections , filtering and grouping

4. Reject Contours based on width and Height threshold.
5. Reject Contours based on descriptor density in each contour.
6. After filtering out contours, combine contours based on proximity in image.
7. Combined image is an obstacle.

Obstacle distance Detection

1. Once obstacle is identified we use the depth map to calculate distance.
2. Using the formula D = Bf/d we calculate the distance of the obstacle from the camera.
3. B is the distance between cameras which is 30cm
4. f is the focal length . I averaged the fx and fy for both cameras we found during calibration.

5. d is the disparity which we get from the depth map.
6. Since we have a noisy disparity , we randomly choose 10 points within the obstacle
7. We calculate distance using the formula mentions above.
8. Reject outlier distance
9. Average the other distance, This will be the distance between the obstacle and the camera.

Results and Limitations

Results

Door Detection

Using method 1 for door detection we were successfully able to detect doors and door knobs, there were less to no false positives in this technique. This has been our best method for door detection, The below image shows us the bounding box that detects the door and a door knob that is shown using a bounding circle over the door knob

Fig 10: Door and Door knob detected

Door Open/Close

Fig 11: Door closed and Door open detection

The above figure shows the detection of open and close doors using depth information that we acquired from our stereo camera

Obstacle detection

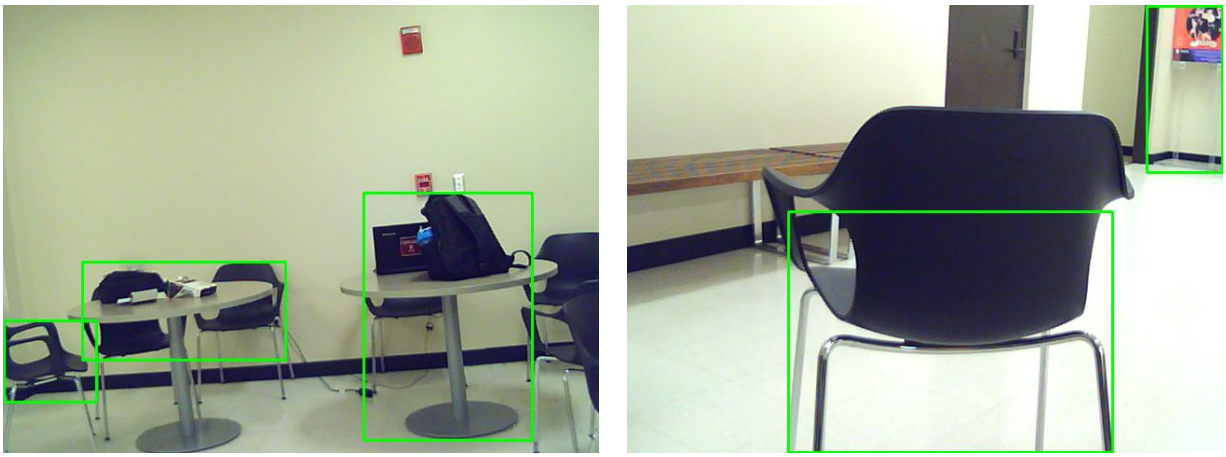


Fig 12: Obstacle detection

The above images shows the results that we got using our obstacle detection method. The above method was considerably effective with a few amount of false positives that were detected in different indoor environments.

Limitations

When detecting doors using method 2 some of the non door objects were detected as doors. This might be due to objects that might match the same geometry as doors. Implementing effective thresholds and adding more heuristics might improve this method's accuracy.

Single obstacle are identified as multiple obstacle. This happens when we have a obstacle of multiple colors we recognize it as two objects, because we use color based segmentation. We tried to do color and location based segmentation , with lower values of k we had a equally divided image. Only higher values of K has made sense. Increasing the K had an impact on the performance. So we decided to use lower K with color based segmentation.

Multiple objects are shown as one object. In case of multiple objects in the scene, It is shown as a single large obstacle. The reason for this is because of the idea we implemented of combining contours of close proximity in the image. We should actually combine contours of close proximity in the image and the depth. Since we did not have clear depth map we were not able to implement this. If we have a clear depth map we can combine contours close in real world to get a better detection.

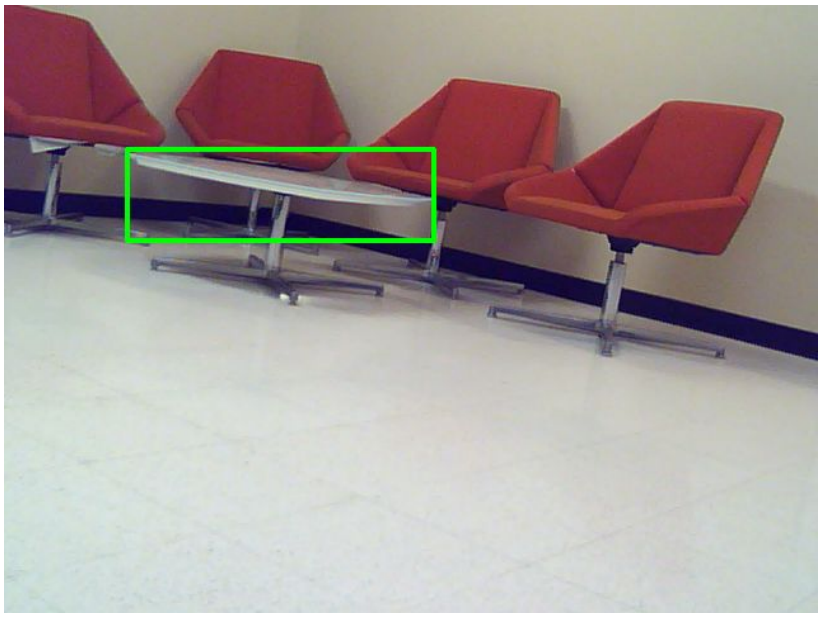


Fig 13: Background obstacles not identified

Some obstacles are not identified. If the images are plain and do not have enough features then they may go undetected. The filtering magic parameters like height, width and threshold features.

Finding doors in ground. We are not sure of the angle of the camera view. If the camera is viewing the floor based on the heuristics we defined for finding the doors, we could end up with doors within obstacles or the other objects. We can improve this with accelerometer , using which we can find the pitch and roll of the camera and decide the viewing angle of the camera and decide whether to find doors or obstacles.

Future Work

- Identifying an effective way of communicating the information about the surroundings to the visually impaired. The communication could be voice commands or through actuators
- We also would try to integrate the accelerometer to the camera to find the viewing angle of the camera. This would help to decide if door or obstacle detection is required.
- We would like to improve the obstacle tracking mechanism. At present we are just using feature based detection. If we are able to find a mechanism to identify the motion of camera/obstacle we could use probabilistic method to track the obstacle better.
- Improving the depth map would be of high priority as it would help to distinguish obstacles and measure the distance of the obstacle from the camera.
- Use Cascade based detection to identify faces and use neural networks to identify the the face to help them identify the person standing in front of them.
- Try to build the dynamic map using SLAM and help the visually impaired to navigate to different location of the house.
- We need to incorporate all these features keeping performance in mind. Since the environment is dynamic so we have to be fast in out detection.

Extending beyond Indoor doors

In a preliminary study for generalizing doors, we attempt to detect outdoor doors. We loosen few assumptions we initially had, and we intend to detect outdoors. Since outdoors come in various shapes, we approached the vision problem from machine learning perspective.

We took several images (~50) from imagenet containing doors and selected several images to represent several door shapes like as follows.

The intention is to sample as many shapes of doors as possible to fairly represent real life variations. These represent “positive” results. We used object detection based on Viola-Jones object detection method that uses haar-based features.

We used a script from Naotoshi Seo[4] for generating about 1500 samples with 40 x 40 windows from about 50 positive images and 100 background images tilted to multiple angles. The generated vec files was joined using the script from Blake Wulfe [5] and a cascaded classifier was trained to detect doors. Multiple cascaded classifiers were trained with different (Stage 5, Stage 20) to detect the doors. As expected, 20 stages performed better than 5 stages, however had much more false positives.

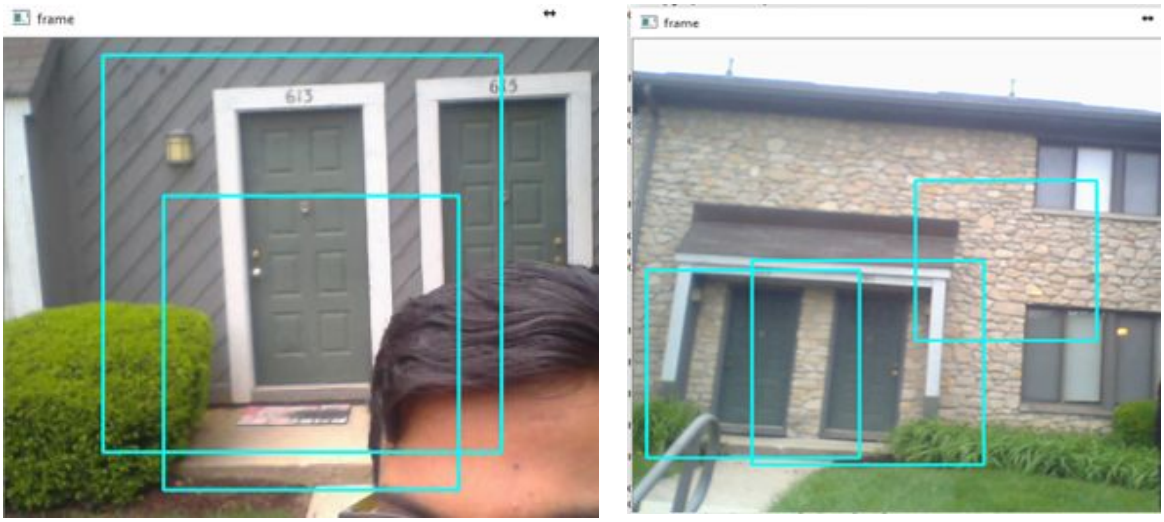


Fig 10 : Cascade classifier using Haar like features

In above images, we show few positive examples of correct detections. However there were a lot of false positives detected when we implemented this method. We are working on improving it.

Conclusion

The problem of creating a system that can assist the visually impaired has been an inspiring challenge that has been improving over the years. The main motivation behind this is creating a system that could effectively improve the lives of the visually impaired people around the world. Since this problem closely aligns with the medical domain, the influence of false positives and false negatives can be critical and can have adverse effects on human lives. Hence it is important to come to a consensus on the scope of this computer vision based system. With extensive research and improvements this problem is sure to have a linear progress over time.

References

[1] ROBUST DOOR DETECTION , Christopher Juenemann, Anthony Corbin, Jian Li
[2] http://robocv.blogspot.com/2012/01/minoru-3d-webcam-for-real-time-stereo.html
[3] Open CV open source codes
[4] Naotoshi Seo, script for generate multiple samples for open CV,
https://raw.githubusercontent.com/mrnugget/opencv-haar-classifier-training/master/bin/createsamples.pl
[5] Merge vec generated by opencv_trainsamples, Blake Wulfe,
https://raw.githubusercontent.com/wulfebw/mergevec/master/mergevec.py