Indian Institute of Information Technology Vadodara
Government Engineering College, Sector-28, Gandhinagar,
Gujarat - 382028

Design Project Report

on

Obstacle Avoidance for Mobile Robots

Submitted by

Anagha Mittal - 201851020
Kavya Tripathi - 201851058
Vraj Paresh Mistry - 201851148

under the supervision of

Dr. Pratik Shah

Abstract- The object detection system for mobile robots is a device that uses YOLO and OpenCV's contour methods to detect an object and calculate the distance between it and our device. The main purpose of this model is to clean the floor as and when it is moving as it has a mop attached to it that can be detached, cleaned and re-attached. The device detects an obstacle in front of it at a suitable distance and turns from its current path to avoid collision. Not only is this model very useful but ours is comparatively very cost efficient. It is much simpler and affordable compared to its contemporaries already present in the market.

I. Introduction

How about saving some time and getting our homes cleaned as we do some other work? Obstruction Avoidance for Mobile Robots is a mobile cleaning device that uses YOLO model to detect the objects in front of it and has a cleaning fabric attached to it so that it cleans the floor as and when it is moving. The main aim of this project was to build an automated cleaning device by using a detachable mop along with an object detection model. The model uses a camera to capture the images in front of it, calculates the distance of any obstruction in front of it and changes its direction if required. We have used Raspberry pi for automating this system and YOLO to train our model.

II. Literature Survey

For building this project, we started with the study of models like regression, decision trees, LSTM, Convolutional Neural Networks, Ensemble Trees etc. Since there is an extensive usage of CNN and Ensemble trees in object detection systems, we narrowed down to the following two training models; CNN and YOLO.

CNN: Computers read images as pixels and it is expressed as a matrix (NxNx3) — (height by width by depth). Images make use of three channels (RGB), so that is why we have a depth of 3. CNN uses (MxMx3) with M<N. called learnable filters. With depth remaining the same, this filter takes a dot product to give an activation map. Different filters which detect different features are convolved on the input file and a set of activation maps is outputted which is passed to the next layer in the CNN. These activation maps form a stack to form a new image of smaller dimension than the input image.

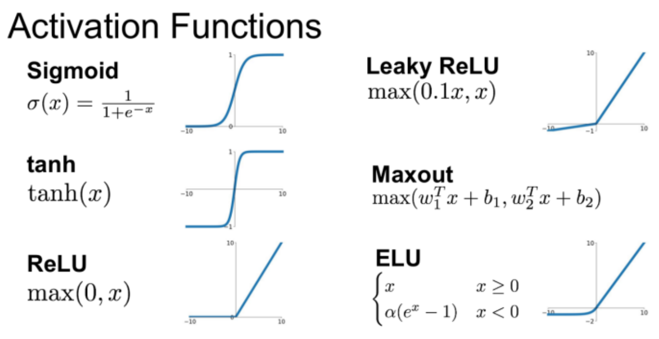


Fig -: Activation Functions

YOLO: You Only Look Once (YOLO) technique is better than CNN, R-CNN or Faster R-CNN as it looks at the whole image at once unlike the CNN models that only process that part of the image which has a high probability for objects. The processing speed of YOLO is 45 frames per second which is much better than real-time. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

For the hardware survey, we studied the working of Arduino UNO, Arduino Mega and Raspberry pi. Raspberry pi functions 40 times faster than Arduino in terms of clock speed. Also, Raspberry pi can handle complex functionality better than Arduino. Arduino programming is easy but for a large scale application, raspberry pi is more suitable and reliable.

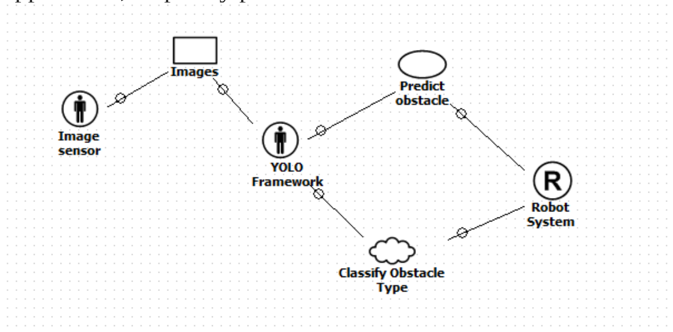


Fig -: SD diagram for System Design Model

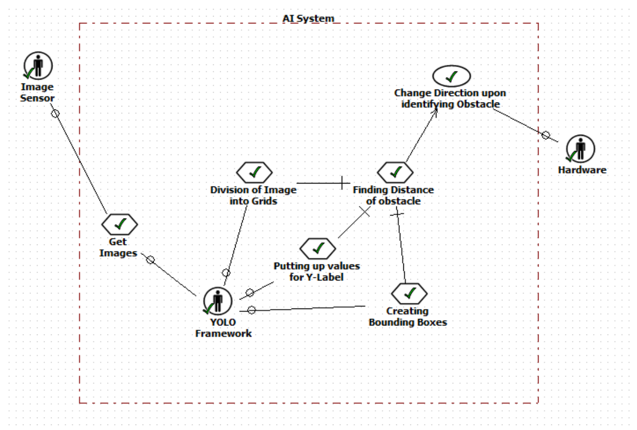

AI System
Image Sensor
Change Direction upon identifying Obstacle
Hardware
Division of Image into Grids
Finding Distance of obstacle
Get Images
Putting up values for Y-Label
Creating Bounding Boxes
YOLO Framework

Fig -: SR diagram for System Design Model

III. THE PRESENT INVESTIGATION

The final model is built on the YOLO (You Only Look Once) model as it can classify the objects in the entire image at once and can identify the classes of objects in that. Then we calculate the distance of the nearest object using OpenCV's contour methods. Finally, we feed all these codes in Raspberry pi for connecting our hardware and software.

- First of all, we started by training our model to classify the objects in images and identify them. For this we used the YOLO model as it was easily available and much better than the CNN, R-CNN or Faster R-CNN models in terms of speed and accuracy. The model has been trained with 80 classes of objects which include dining table, chair, car, animals, mouse, certain food items etc.

  The model identifies these objects in the image and creates a bounding box around them for their classification.

- After detecting the object by YOLO, we identify the closest obstacle in the image by OpenCV's contour methods. When the closest obstacle is identified, we retrieve the average width of the detected object, for example, the camera detects a ping pong ball in the path, then the ball's width would on an average be 1.6 inches. Using the formula

  Distance from the camera = (knownWidth * focalLength) / perWidth

  perWidth is the marker width calculated from the image
  focalLength is the calibrated focal length of the camera.
  knownWidth is the average width of the closest object detected in path.

  By this we find the distance of the object in the path. Then we set a limit that would be our pre threshold Distance that is maybe about 15 cm, that would be enough for our vehicle to turn to any of the sides and an another limit that would be our threshold Distance that would then reverse the vehicle than steering on sideways.

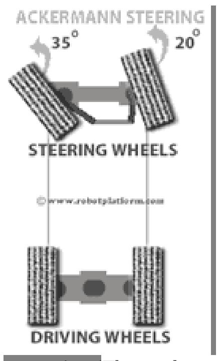


Fig --: The Ackermann steering (The angle mentioned in the image is just to make it easier to understand. We deviate our device by a 60 degrees angle to avoid collision)

- For connecting our hardware devices, we have used Raspberry pi. The camera is connected to it to capture the images and then send to the software to detect and classify images.

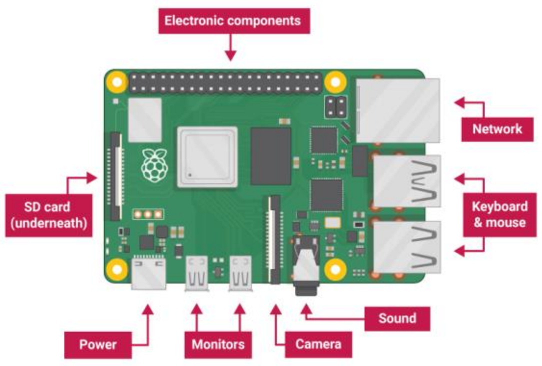


Fig -: Raspberry pi board for connections of the hardware

SD card: It has all the codes (YOLO and OpenCV codes) and the Raspberry pi OS (Linux)

Camera: To connect the camera for capturing images

Network: It is the Ethernet port

Electronic components: To connect Servo motor and other hardware components

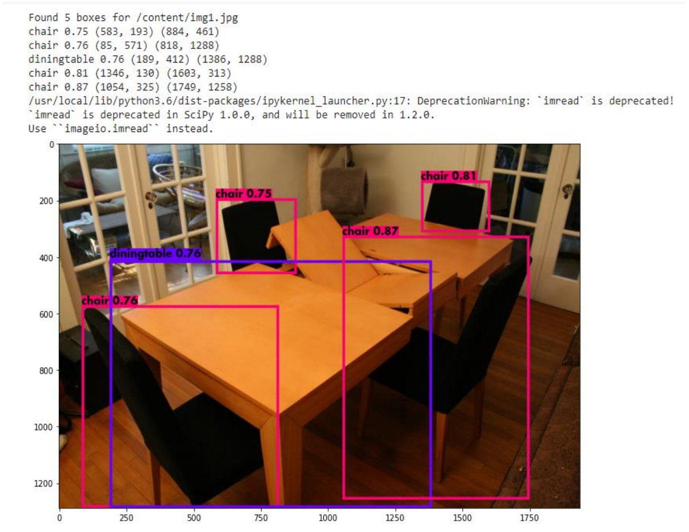
Power: For inputting a power source.

Fig -: Camera to be attached to the Raspberry pi

The hardware model also includes a Servo motor that will help the device turn from its original path when needed. The Servo Motor will be connected to the Raspberry pi and will rotate by a certain angle when it is necessary for the robot to turn. Also it includes a detachable mop that can be clipped to the mobile robot for cleaning the floors. The mop can be removed, washed and again attached to the mobile robot.

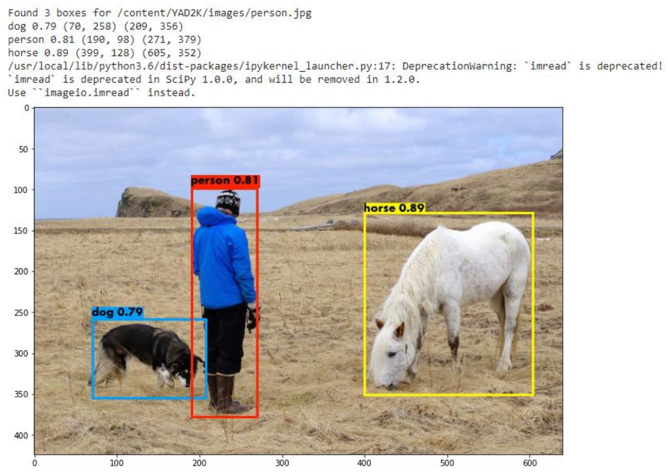
IV. Results and Discussions

The model is trained with 80 classes of objects and is able to classify the things in the images fed by us as test cases. Following are some of the images tested by us along with image classification/identification:



```
Found 5 boxes for /content/img1.jpg
chair 0.75 (583, 193) (884, 461)
chair 0.76 (85, 571) (818, 1288)
diningtable 0.76 (189, 412) (1386, 1288)
chair 0.81 (1346, 130) (1603, 313)
chair 0.87 (1054, 325) (1749, 1258)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

Fig 1: classifies the tables and chairs



```
Found 3 boxes for /content/YAD2K/images/person.jpg
dog 0.79 (70, 258) (209, 356)
person 0.81 (190, 98) (271, 379)
horse 0.89 (399, 128) (605, 352)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

Fig 2: classifies person, dog and horse

```
Found 10 boxes for /content/img.jpg
car 0.63 (998, 325) (1090, 388)
car 0.64 (0, 279) (79, 339)
car 0.68 (139, 278) (219, 322)
car 0.69 (401, 314) (537, 413)
car 0.72 (541, 308) (669, 405)
car 0.73 (1005, 544) (1200, 765)
car 0.79 (549, 417) (750, 562)
car 0.79 (414, 544) (762, 778)
car 0.83 (13, 415) (301, 630)
car 0.85 (912, 408) (1126, 577)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

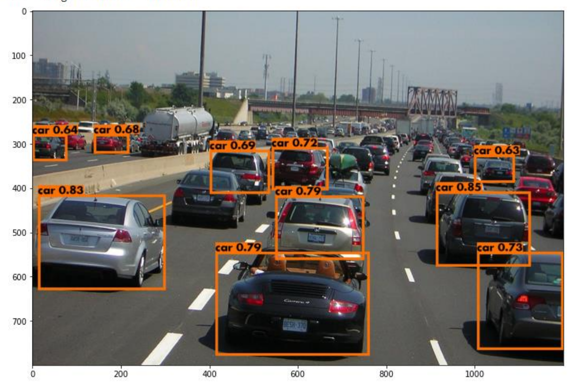
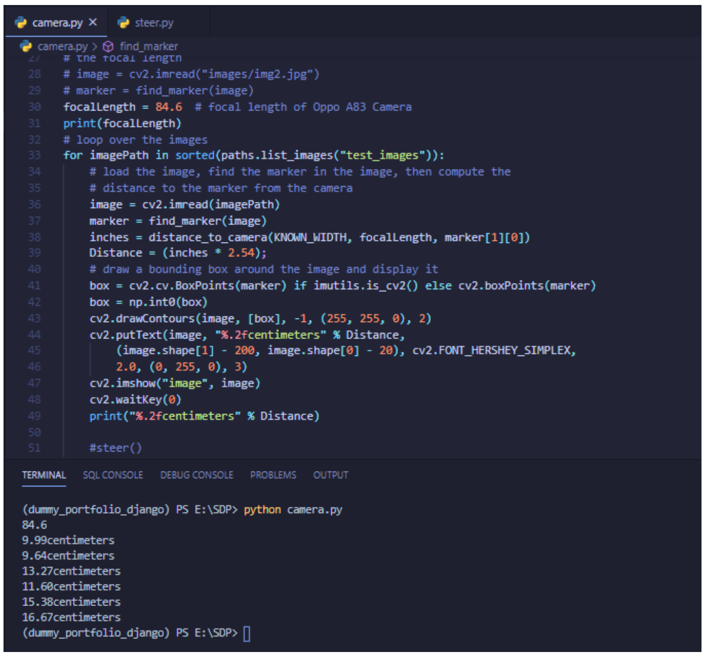


Fig 3: identifies all the cars creates a bounding box around each



```python
# the focal length
# image = cv2.imread("images/img2.jpg")
# marker = find_marker(image)
focalLength = 84.6  # focal length of Oppo A83 Camera
print(focalLength)
# loop over the images
for imagePath in sorted(paths.list_images("test_images")):
    # load the image, find the marker in the image, then compute the
    # distance to the marker from the camera
    image = cv2.imread(imagePath)
    marker = find_marker(image)
    inches = distance_to_camera(KNOWN_WIDTH, focalLength, marker[1][0])
    Distance = (inches * 2.54);
    # draw a bounding box around the image and display it
    box = cv2.cv.BoxPoints(marker) if imutils.is_cv2() else cv2.boxPoints(marker)
    box = np.int0(box)
    cv2.drawContours(image, [box], -1, (255, 255, 0), 2)
    cv2.putText(image, "%.2fcentimeters" % Distance,
        (image.shape[1] - 200, image.shape[0] - 20), cv2.FONT_HERSHEY_SIMPLEX,
        2.0, (0, 255, 0), 3)
    cv2.imshow("image", image)
    cv2.waitKey(0)
    print("%.2fcentimeters" % Distance)

    #steer()
```

```
(dummy_portfolio_django) PS E:\SDP> python camera.py
84.6
9.99centimeters
9.64centimeters
13.27centimeters
11.60centimeters
15.38centimeters
16.67centimeters
(dummy_portfolio_django) PS E:\SDP> 
```

```
Found 6 boxes for /content/img2.jpg
diningtable 0.61 (0, 10) (562, 399)
sandwich 0.63 (9, 46) (159, 177)
sandwich 0.71 (326, 180) (460, 305)
pizza 0.80 (0, 270) (298, 406)
sandwich 0.82 (175, 62) (301, 172)
sandwich 0.86 (178, 147) (285, 249)
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

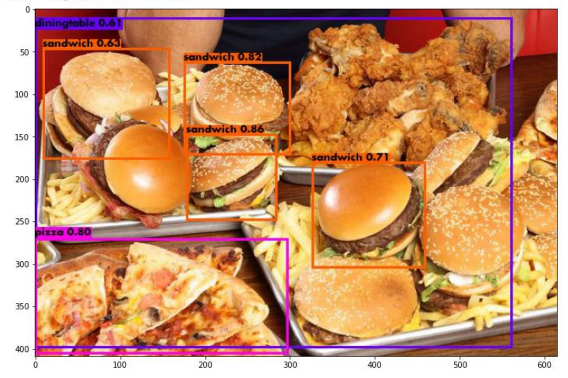


Fig 4: is able to identify the table even when it is not visible and is completely covered with food items

Further our model also predicts the distance of the nearest object and takes a 60 degrees turn from a distance of 15 cm from the object (as the standard size of such devices is 20 cm).

V. CONCLUSIONS AND FUTURE WORK

- The model is able to identify and classify images with a high accuracy hence it is perfectly suitable for our device whose main purpose is to avoid any obstacle in front of it and keep on moving without colliding with anything.
- The nearest object in the image is also recognized and we are getting an approximate distance of it from the object.
- The object is made to take a 60 degrees turn from its path on detecting an object at a distance of 15 cm from it.
- The mop attached to the device will clean the floor and can be detached from the device for washing.

For future development, the model can be trained with larger data sets for better accuracy and scaled up to the industry level. It can be built to a larger setup to cover bigger areas. It can also be linked with cloud to store the map of the house/room it is being used in. Also, IoT can be included in this system to make it automated and get better results. Further, self-charging mechanism can be added to this device to make it more user-friendly.

ACKNOWLEDGMENT

We are extremely thankful to IIIT Vadodara for giving us this wonderful opportunity to work on a design project and the Design Project Coordinator for the instructions. We would like to thank **Dr. Pratik Shah**, our mentor for the project who helped us throughout and it was for his guidance that we could build this project successfully. Also, we want to thank our colleagues from the institute who helped us whenever we needed them.

REFERENCES

[1] https://github.com/allanzelener/YAD2K

[2] https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/

[3] https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17#:~:text=The%20activation%20function%20is%20a,neuron%20would%20fire%20or%20not.&text=We%20have%20different%20types%20of,Rectified%20Linear%20Unit%20(ReLU).

[4] http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html

[5] Ma, Li Yong, Wei Xie, and Hai Bin Huang. "Convolutional neural network based obstacle detection for unmanned surface vehicle." *Mathematical biosciences and engineering: MBE* 17.1 (2019): 845-861.

[6] Du, Juan. "Understanding of object detection based on CNN family and YOLO." *Journal of Physics: Conference Series*. Vol. 1004. No. 1. IOP Publishing, 2018.