Contents

Path finding algorithm using OpenCV

1 Introduction

1.1 Title
Path finding algorithm using OpenCV

1.2 Brief Description of the Problem
Briefly, the problem is detecting the course markers to help an autonomous vehicle find an optimal trajectory to complete the circuit. The idea is to develop an algorithm that will help a controller, which will first be demonstrated on an autonomous car that will navigate a closed course in minimal time. An embedded device(Raspberry-Pi) will be used to take live video input and run the algorithms on it.

1.3 Brief Description of the Solution
I will use deep learning and computer vision(OpenCV) algorithms to detect the course markers. To find an optimal path for the vehicle, we need to detect markers present in the circuit; thus, the coloured cone-like object will be spread throughout the course, both ML and CV algorithms will be deployed on Raspberry-Pi4. A camera connected to the Raspberry-Pi4 will take the circuit's live video and feed it to Raspberry-Pi4. The algorithm will run on video input. It will find the detected markers' locations and use that; various information of the markers (distance, angle, etc.) relative to the vehicle will be calculated.
The words model and ML algorithm are used interchangeably.

2. Project Goals

2.1 Project Objectives
The objective is to detect and return course markers' locations to help an autonomous vehicle find optimal trajectory using OpenCV and machine learning.

2.2 Expected Deliverables

- Train an efficient, accurate and faster TensorFlow Lite object detection model that will be deployed on embedded devices such as Raspberry-Pi4 and detect and return the course markers' locations.
- An OpenCV algorithm that will detect the course markers and return its locations
- A function which will calculate distance range of the object w.r.t camera.
- Deployment of both the algorithms on Raspberry-Pi4.

2.3 Current State of the Problem
The current state of the project does not include marker detection in the circuit. I propose to add object detection algorithms using ML and CV, which will find the locations and types of markers.

2.4 Future work based on the project
This algorithm can be extended to detect vehicles and other objects such as pedestrians, which can be very helpful for planning the optimal trajectory. This information will be used to plan an optimal trajectory for the vehicle.

3. Implementation

3.1 Project Methodology

- **Deep Learning Algorithm**
  To detect markers(cones, etc.) and identify the location in real-time and with reasonable accuracy, an object detection algorithm is needed. I propose using SSD (Single Shot Detector) for object detection since it is the most compatible algorithm and works with the fastest speed rate. I will use transfer learning to train the model provided by TensorFlow in their model detection model zoo. Transfer learning is a technique that uses a model's weight trained on a large dataset and finetunes it for retraining on a different dataset.

- **Markers Detection Using CV algorithm**
  I will use different CV algorithms to detect the markers. I will start with a matching feature matching algorithm and proceed further with colour segmentation and contour detection. Using testing data, I will do experimentation to finetune the algorithm. I will use OpenCV for this.

- **Deployment of models on Raspberry-Pi4**
  I will use TensorFlow Lite to deploy the algorithm on Raspberry-Pi. It is a framework that is optimised for deploying lightweight deep learning models on resource-constrained edge devices. It requires less processing power and faster inference time, so it can be used to obtain speedier performance in real-time applications. Before deploying, the model will be converted into the TensorFlow Lite model. Then the ML model will be deployed.

  I will deploy the OpenCV algorithm on Raspberry-Pi after comparison on the test data.

3.2 Project's Technical Elements
The following are technical elements of the project:

- Dataset preparation for marker detection
- Training of ssd-mobilenet model
- Preparation of Testing data and Testing
- Improvement of model and re-training of model
- Blob detection using the different CV algorithms
- Testing of the CV algorithm
- Calculation of distance and angle of marker w.r.t. vehicle
- Deployment of algorithms on Raspberry-Pi4

3.3 Proposed Solutions

- **Dataset Creation**

  1. **Collecting Images**
     Training a good deep learning model requires a vast dataset. The training images should have random objects in the image and the objects(cones of different colours), resulting in a robust classifier's training. Dataset will contain cones of different colours. Using more than one colour will result in the training of a good classifier. I will collect at least 300 to 400 images for each class.
  2. **Labelling Images and Making Data in Trainable Form**
     For the training of an object detection model, the images in training data must-have objects with labels and their locations, i.e., a bounding box for each object. For the annotation of the images, I will use a great open-source tool LabelImg.

     TensorFlow uses tfrecords(a simple format for storing a sequence of binary records) to train an ML model. I have written the python function for doing this task. By running the functions on data, I will generate tfrecords.

- **Training and Exporting SSD-mobilenet model**

  1. **Configure Training**
     Before the training, I have to configure the parameters for transfer learning and other hyperparameters, which is the last step before running training. I will use the config and checkpoint files provided by TensorFlow.
  2. **Training**

During the training process, I have to track the model's progress, which is measured by the loss function. I will be using TensorBoard to track the loss of the model. Tensorboard creates a webpage on the local computer that will provide information and graphs that show how the training process is progressing. I will apply early stopping on the model, which controls the training process if our model's improvement is halted (using validation data). I will use Checkpoints to save the weights of the model.

3. **Convert into TensorFlow Lite models**

   To use the trained model, I have to convert the model weights into the TensorFlow lite model, which is a two-step process. The first step is to export the model weights, using the python script open-sourced by TensorFlow itself; this will generate the inference graph. The next step is to use TOCO, which converts models into an optimized format that allows them to run efficiently on TensorFlow Lite. The execution of this will result in the final model.

- **Testing and Improvement of Object Detection Model**

1. **Preparation of Testing Data**

   To evaluate the object detection model, I will prepare a testing dataset in the form of video. The data will be generated using a Raspberry-Pi4 camera. As discussed, the dataset's complexity will increase as the first video streams will only have a single cone of different colours, and then the number of cones with varying colours in a frame is increased. Finally, the compilation of all the video streams will be our final testing data.

2. **Test the Model on Locally System**

   Before deploying the model, I will test it on my local system. I will create a jupyter notebook for this task. The parameters to test the model are

   A) Accuracy: I will use IOU to measure this parameter

   B) Speed of detection: I will track the fps at which detection is correctly done.

   IOU metric can be computed using numpy:

```
intersection = np.logical_and(target, prediction)
union = np.logical_or(target, prediction)
iou_score = np.sum(intersection) / np.sum(union)
```

3. **Improving Accuracy of the Model**

I will use the following well known methods to improve the identification speed and accuracy of the learning algorithm.

1. Data Augmentation
2. Visually Coherent Image Mix-up
3. Training Scheduler Revamping

- **Marker Detection Using CV**

  I will start with a matching feature matching algorithm and proceed with colour segmentation and contour detection and obtain the results. I will do experiments and explore other algorithms as well. I will use OpenCV's great library for the implementation. I will use the same video testing data to test the algorithms. For this, I will have to write a python code so that the algorithms will work on video input.

- **Deploying TensorFlow Lite Model on Raspberry-Pi**

  The model is ready for use. I will deploy the model on Raspberry-Pi. Then, I will create the necessary python scripts that use the TensorFlow Lite model and run on video input, cloned in Raspberry-Pi. The final model will identify the locations of the markers.

- **Deploying OpenCV algorithms on Raspberry-Pi**

  I will deploy the OpenCV algorithms on Raspberry-Pi. For this, I have to install OpenCV on Raspberry-Pi. Then, the codes will work.

- **Measurement of Distance and Angle of Markers relative to the Vehicle**

  I will be using triangle similarity relation to calculate the approximate distance between the vehicle and the marker.
  The relation is given as:
  Distance = (known height * focal length) / pixel height
  The object detection model will return the object's height and width, which will be fed into the function. Finally, these results will be used to obtain the map of the course.

4. Project Timeline

4.1 Project plan and Deliverables Schedule

I expect to complete the following goals in order.

Community Bonding Period (17 May - 7 June)

- I will be discussing this with my mentor in further detail.
- Preparation of the dataset for training and setting up the TensorFlow environment in my local system.

First and Second Week (7 June - 20 June)

- Complete the dataset annotation
- Train the ML model with some experimentations(In transfer learning and hyperparameters)
- Create an inference graph using the weights of the trained model for Testing

Third and Fourth Week (21 June - 4 July)

- Write python scripts for testing the model on the live video inputs
- Preparation of Testing dataset.

Fifth Week (5 July - 11 July)

- Complete the preparation of testing data
- Prepare the First term report for evaluation

Sixth and Seventh Week (12 July - 25 July)

- Start testing the model on video input on the local system (Linux)
- Experiment with ML models for better results
- Final Testing of models and convert the best one into TensorFlow Lite
- Deployment of the ML model on Raspberry-Pi4

Eighth and Ninth Week (26 July - 8 August)

- Preparation of different CV algorithms for Marker detection
- Testing the algorithms on live video input
- Experiment with algorithms for better result Implement the function for measuring the distance of object relative to the vehicle
- Deployment of CV algorithm on Raspberry-Pi4

Tenth week (9 August - 15 August)

- Buffer week for unexpected delays
- Complete the documentation
- Showcase the final demonstration and complete the final report

4.2 Commitments outside project that might impact work

I do not have any commitments outside the project during summer. I understand that this is a serious commitment and equivalent to a full-time paid summer internship or job. I will work with my full dedication and I am very excited to work on this project.

4.3 Areas of project you expect to be most/least challenging

I think that the testing of the ML algorithm is a challenging task. I will have to prepare the dataset first; then, I have to improve the model after testing. It is the most exciting and difficult part of the model, where I get to apply the tricks I have learnt. Other areas of the project aren't that challenging since I have implemented them in various previous projects. So, they are relatively straightforward.

5. Biographical information

5.1 Relevant experience / previous work

Implementation of object detection for visually impaired person GitHubLink

- The project aims to develop a compatible device for aiding vision for blinds.
- Implemented two states-of-the-art object detection algorithms(Faster RCNN and YOLOv3).
- Implemented the paper" Show and Tell" for implementing image captioning.
- Deployed algorithm on Raspberry-Pi and its camera is used to take images of live feed as an input and provides captions as output to the user through speech.
- Hosted the algorithm on AWS, which communicated through the Flask API.
- Developed an Android app for the task.

Rock Mass Analysis using Image Processing GithubLink

- Analysis of Rock-Mass such as Joints and Faults using image processing.
- The Hough Line Transform implemented in OpenCV is used to detect faults and joints.

5.2 Relevant Education Background

I am a dual degree (B.Tech + M.Tech) student of Mining Engineering at the Indian Institute of Technology, BHU. My area of interest is machine learning and computer vision. In my course of study, I have done various projects in the field of mining, with the help of computer vision. My B.Tech project was classification of minerals through images using deep learning and image processing. Currently, for my Master's thesis I am working on the detection of geological structures based on images using object detection.

5.3 Programming interests and strengths

I have been programming for the last four years at college. I have been a part of many projects, and I have developed a flair for tackling challenges. For the past few years, I have developed an interest in deep learning and its real-life applications.I have participated in various inter-college hackathons and applied my deep learning skills there. I am proficient in coding in Python and C++ and also regularly participate in competitive programming contests on different online judges(Expert on Codeforces and 4-star on codechef). (My resume)

5.4 Contact information

Name:	Rupal Sharma
Country:	India
Email:	**rupal.sharma.min17@itbhu.ac.in**
Phone:	+919695223215
Github:	**https://github.com/rsdel2007**
Linkedin:	**https://www.linkedin.com/in/rupal-sharma/**

6. References

1) Object detection using TensorFlow Lite
2) Single Shot Detectors
3) LabelImg for data annotation
4) Paperspace for data augmentation
5) TensorFlow
6) OpenCV
7) Model Zoo