VISUAL ODOMETRY

(Autonomous Bot)

ALIABBAS MERCHANT, JASH RATHOD, SUYASH MORE, SWAPNIL PAREKH

COMPUTER AND IT DEPARTMENTS

SRA VJTI MUMBAI-400019

merchantaliabbas@gmail.com	9892875512
swapnilbp100@gmail.com	9833026405
suyash.21.more@gmail.com	9860467544
jashrathod0@gmail.com	9920691795

CONTENTS

ABSTRACT

Today is the age of self-driving cars. Various MNCs around the world are working hard to accomplish the dream of self-driving (autonomous) cars.

An autonomous car (also known as a driverless car, self-driving car, and robotic car) is a vehicle that is capable of sensing its environment and navigating without human input. Autonomous cars combine a variety of techniques to perceive their surroundings, including odometry and computer vision.

The potential benefits of autonomous cars include reduced mobility and infrastructure costs, increased safety, increased mobility, increased customer satisfaction, and reduced crime. These benefits also include a potentially significant reduction in traffic collisions; resulting injuries; and related costs, including less need for insurance. Autonomous cars are predicted to increase traffic flow; provide enhanced mobility for children, the elderly, disabled, and the poor; relieve travelers from driving and navigation chores; lower fuel consumption; significantly reduce needs for parking space.

Taking up this challenge in our Eklavya 2018 project, we have designed an autonomous robot. The robot is instructed where to go. It then uses dual cameras in a stereo setup, in order to analyse its surroundings and detect obstacles in its path. The robot then makes an internal map of the destination and any potential obstacles in its path. Artificial Intelligence is then used to detect the best possible route to the destination, while avoiding the obstacles. The bot then navigates to its destination.

KEYWORDS - Computer Vision, Artificial Intelligence, Autonomous Car/Robot, Stereo Cameras

INTRODUCTION

In robotics and computer vision, visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images. It has been used in a wide variety of robotic applications.

Algorithm:

1. Acquire input images
2. Image correction (apply image processing techniques for noise removal, etc.)
3. Feature detection (define interest operators, and match features across frames) (Use of feature extraction and correlation to establish correspondence of two images).
4. Estimation of shortest path to destination and issuing instruction set for robotic maneuver.

TECHNICAL BACKGROUND

Computer Vision

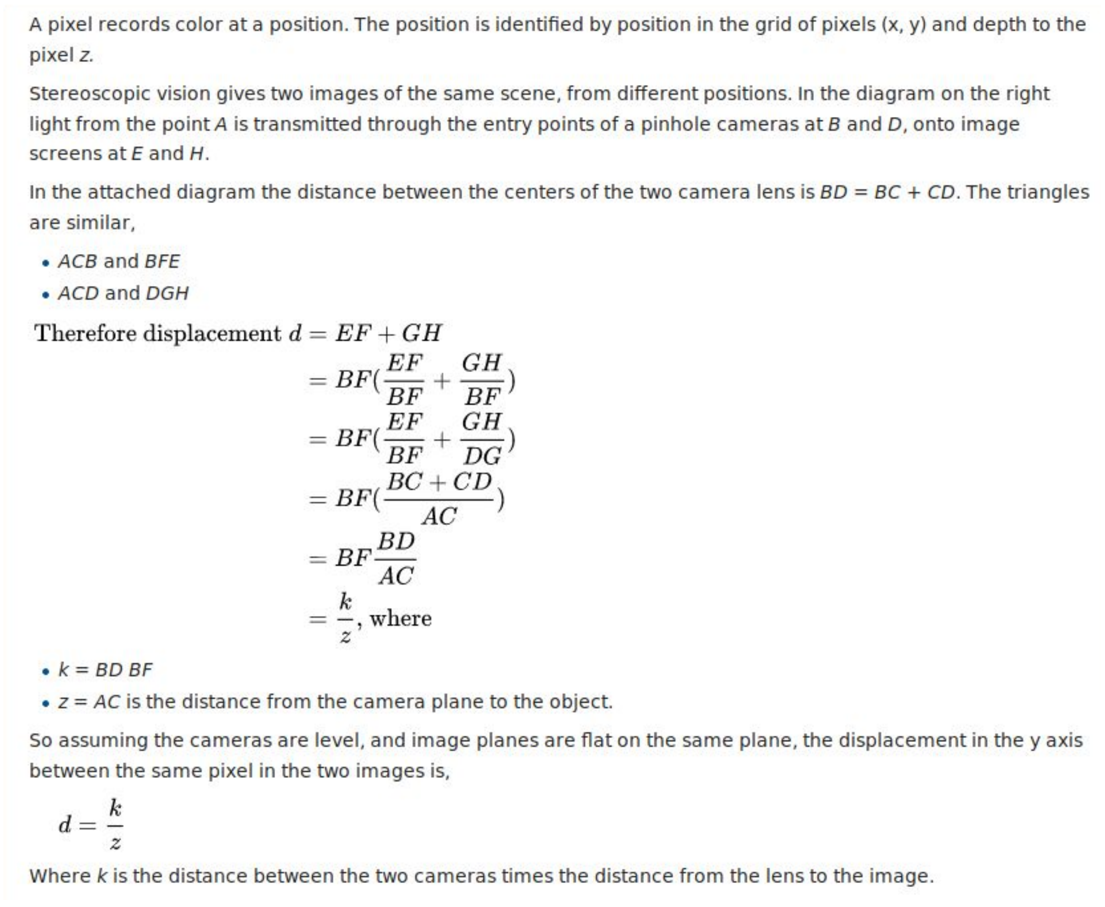
Exploration is an important and active area of research in field robotics, as vehicles capable of autonomous exploration have the potential to significantly impact a wide range of applications such as search and rescue operations, environmental monitoring, and planetary exploration. For this work, we define the *exploration problem* as 1) covering an unknown environment,2) mapping the area, and 3) detecting objects of interest. Therefore the key component of robotic exploration are vision systems. Several ways can be used to achieve computer vision, like visual odometry with image processing, machine learning, convolutional neural networks, etc. In the scope of this project we used the former for the tasks mentioned above.

There are three main challenges present in a complete solution to the exploration problem. First, the approach should maintain a globally consistent map over long distances with mainly relative measurement information and intermittent absolute measurements, like magnetometers. Second, the solution should reliably identify potential objects of interest at as great a range as possible to minimize the time spent sweeping an environment for candidate objects, as well as identify objects of interest in varying lighting and environmental conditions. Finally, a method to plan an efficient search path over a terrain with unknown obstacles and contours is required. The first two are discussed next and third is discussed in a separate section.

Stereo Cameras and Depth Perception

Using the aforementioned techniques of image processing, the tasks of detection and mapping of an unknown area were achieved. Stereo Vision refers to 3D vision. In a traditional sense, the images taken by both eyes combined by the mind is referred to as a stereo image which helps in depth perception. This process is called 3D reconstruction.

Depth perception using stereo cameras is done in the following way:



A pixel records color at a position. The position is identified by position in the grid of pixels (x, y) and depth to the pixel *z*.

Stereoscopic vision gives two images of the same scene, from different positions. In the diagram on the right light from the point *A* is transmitted through the entry points of a pinhole cameras at *B* and *D*, onto image screens at *E* and *H*.

In the attached diagram the distance between the centers of the two camera lens is *BD* = *BC* + *CD*. The triangles are similar,

- *ACB* and *BFE*
- *ACD* and *DGH*

$$\begin{aligned}\text{Therefore displacement } d &= EF + GH \\ &= BF(\frac{EF}{BF} + \frac{GH}{BF}) \\ &= BF(\frac{EF}{BF} + \frac{GH}{DG}) \\ &= BF(\frac{BC+CD}{AC}) \\ &= BF\frac{BD}{AC} \\ &= \frac{k}{z}\text{, where}\end{aligned}$$

- *k* = *BD BF*
- *z* = *AC* is the distance from the camera plane to the object.

So assuming the cameras are level, and image planes are flat on the same plane, the displacement in the y axis between the same pixel in the two images is,

$$d = \frac{k}{z}$$

Where *k* is the distance between the two cameras times the distance from the lens to the image.

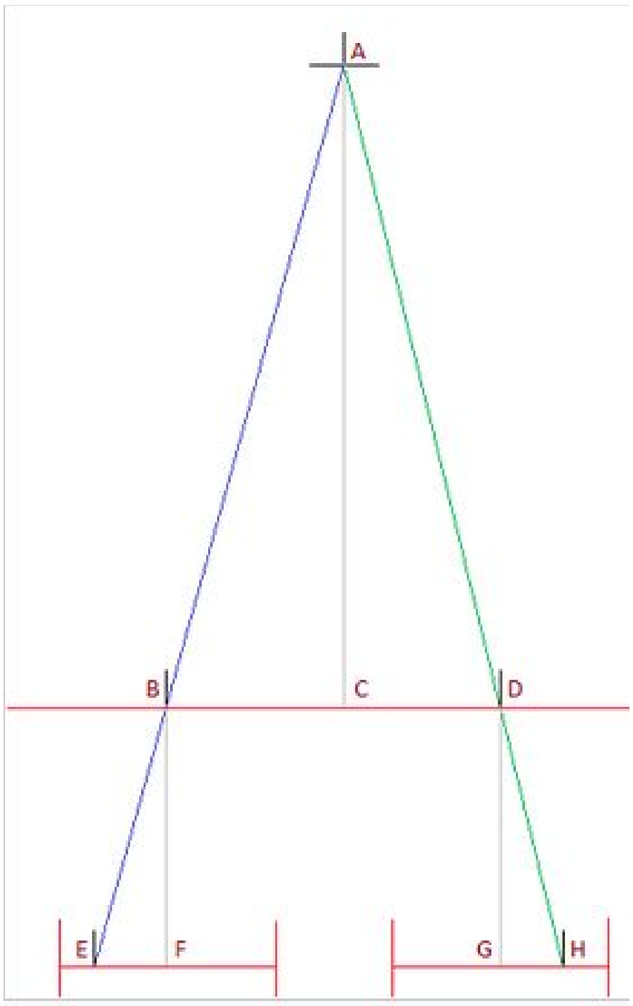


Diagram describing relationship of image displacement to depth with stereoscopic images, assuming flat co-planar images.

(Source: https://en.wikipedia.org/wiki/Computer_stereo_vision)

Path Finding

The aforementioned approach needs to accomplish the objective to reach the goal from the current position. Autonomous navigation using visual odometry utilizes path-planning algorithms for finding shortest path from source to destination. Algorithms used for path-planning tasks include dijkstra, depth-first search and a-star(which we have used in this project).

The algorithm works by expanding the so-called '*frontier*' of the map being parsed node-wise from the starting position until the destination is reached; by looking at its neighbours. The number of neighbouring moves considered valid depends on the use-case, and in ours, they number *4*, with diagonal moves being illegal.

Any neighbours not visited yet, are added to the '*nodes_to_visit*' priority-based queue. To find the shortest route we add the concept of path cost to the design. The first cost is the weight associated with traversing a node, for example, blocked paths have a weight of *infinity* for disincentivization. The standard weight for all other nodes was assumed as *one*. An additional cost was introduced to find the path faster, i.e. a *heuristic cost*, which helps us know if we are en route towards the destination.

The heuristic we chose was the *Manhattan norm* which is, given nodes a(to be visited node) and b(destination) is:

heuristic_cost = (a.x - b.x) + (a.y - b.y)

Hence the final cost of any node traversal is :

cost = cost_so_far + weight_of_node

And hence the priority of the neighboring node in the queue structure nodes_to_visit is:

priority = cost+heuristic_cost

This node is then pushed into the queue according to priority.

This process happens in a loop until the number of nodes to be visited is zero or we have reached the goal index; and the resulting path from start to goal is the shortest path.

The next step was to issue actionable instructions for the actual robotic motion. To solve this, given the shortest path coordinates and an initial orientation, the legal orientations were mapped and correspondingly, the directives to reach the next node were added to a list of instructions. The first term was a boolean value signifying whether to move or not, and the second being the angle of rotation needed to reach the next node. Thus after parsing through the shortest path, a complete set of instructions to reach the goal was ready.

Artificial Intelligence

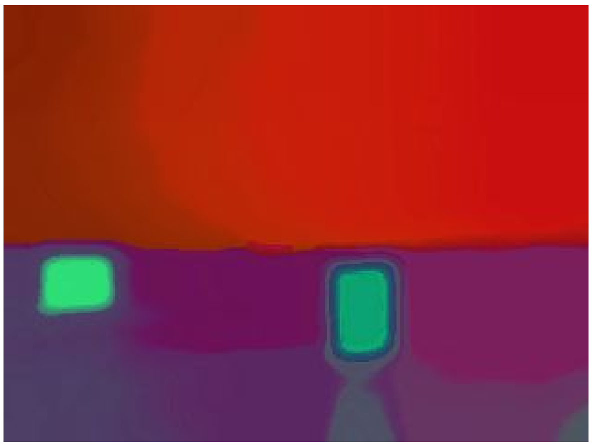
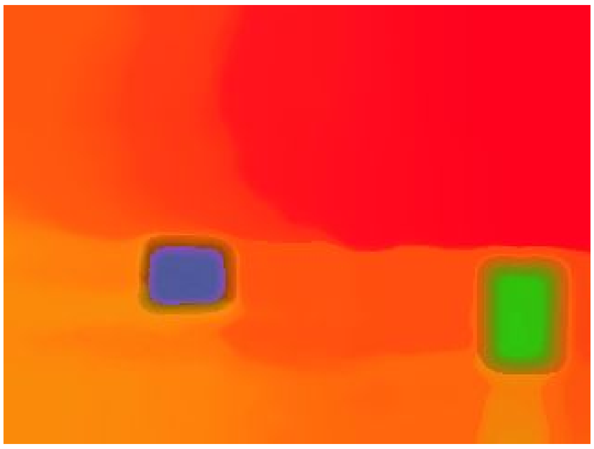
The above mentioned approach can be encapsulated under the domain of AI since the approach allows the robot to travel autonomously without human intervention. The subject of AI is very broad, including various sub-disciplines of Machine Learning, Deep Learning, Heuristic models, Visual Odometry etc. The purpose is to make machines more intelligent for automating tasks that humans are bad at, hence increasing accuracy and decreasing costs. AI is used in diverse areas like medicine, economics, and of course robotics. Extensive research has been directed towards developing newer, faster algorithms for accomplishing tasks like classification, prediction and detection. The most prominent fields of AI in recent times are Machine Learning and Deep Learning, which works by teaching machines the task with enormous amounts of data. Other fields like heuristics are also rising with semi-rule-based algorithms making the trade-off between speed and accuracy.

PROPOSED SOLUTION/IMPLEMENTATION/WORKING

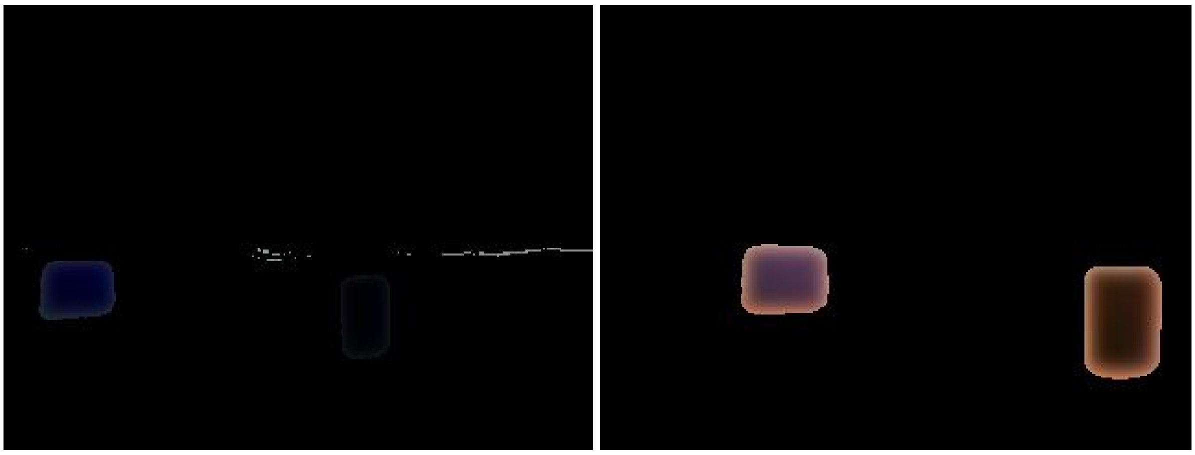
The bot works in the following way:

0. The destination coordinates are given to the robot, in cms. (along +ve X-axis and +ve Y-axis, just like in normal graphs.)
1. It takes a picture of its surroundings using dual cameras and applies noise reduction techniques to filter out the noise. The below 2 images are the ones which have been captured by the cameras and processed.

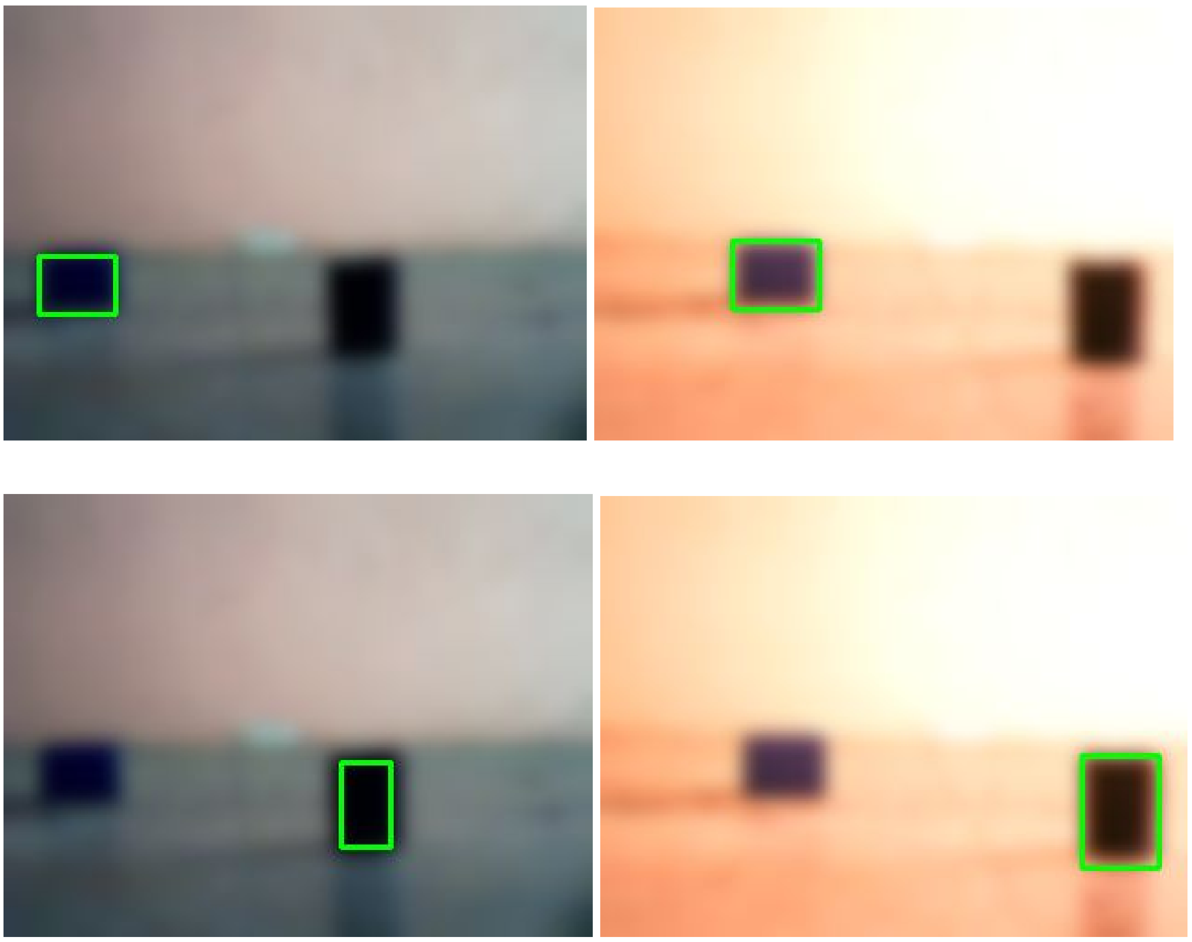
2. The bot (using Raspberry Pi and OpenCV via Python) then converts the 2 images from BGR to HSV format.

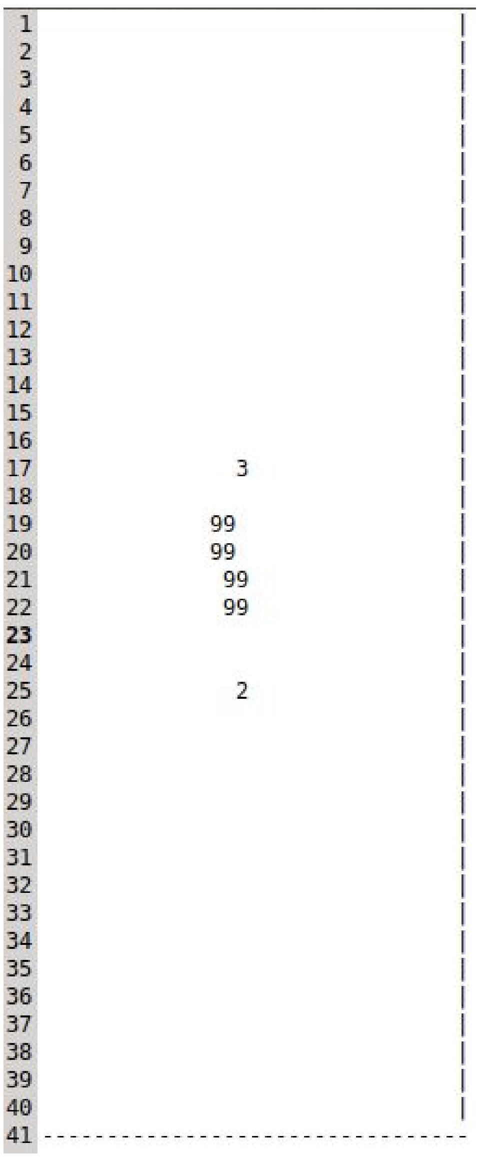
3. Assuming ideal conditions (uniform ground and uniform background), the background and ground are then removed from the image. The resulting images are:



4. Contours are detected in the resulting images and the contours are matched, so as to match the images of the same object in both the cameras.



5. The distance, height, and other properties of the object are then calculated using the detected object contours.
6. Next, using all the detected objects, an internal map (having block size 30 cm x 30 cm) is created, consisting of the objects(denoted by 9), bot position(denoted by 2) and the destination(denoted by 3). (Please note that we have assumed a depth of 30cm for each object. Hence, there is no gap between the detected objects. Also, the detected dimensions are 60 cm, as the objects lie on the boundaries of the blocks of the map, and hence occupy not 1, but parts of both the blocks.)



```
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
               3               |
                               |
            99                 |
            99                 |
             99                |
             99                |
                               |
                               |
               2               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
                               |
--------------------------------
```

7. Now, using the A-Star algorithm, the best route from the bot(denoted by 2) to the destination(denoted by 3) is calculated. (The first term is a boolean value signifying whether to move or not, and the second value is the angle of rotation(anticlockwise positive) needed to reach the next node.)

Detected Path: [True, 0], [True, 0], [False, -90], [True, 0], [False, 90], [True, 0], [True, 0], [True, 0], [True, 0], [True, 0], [True, 0], [True, 0], [False, -270] ,[True, 0]

8. Each action in the path of the bot is then executed. The movement/rotation command is then sent from Raspberry Pi to Arduino, using UART. For each movement command, the bot wheels rotate for a constant amount of time, so that the bot can cover 30 cm(the block size of the internal map). For rotation (spot-rotation), a magnetometer and I2C communication have been used, so that the bot can be rotated the desired number of degrees.
9. The bot finally reaches its destination.

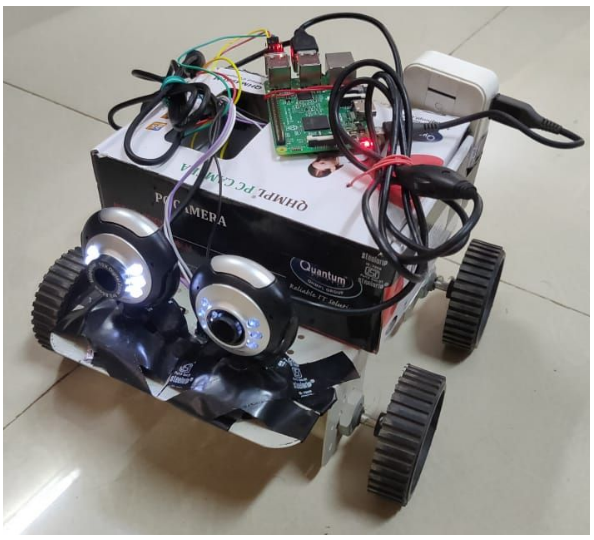
[A sample video of the bot navigating to its destination, (in the presence of 1 object), has been attached along with this document.]

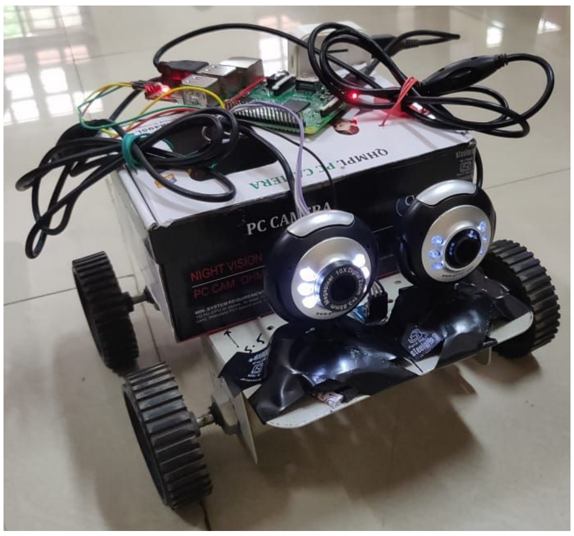
<u>CHALLENGES, RESTRICTIONS AND LIMITATIONS</u>

1. The processing on Raspberry Pi is very slow, so we can’t process the camera feed live. Hence, we take the images, only once, at the start, and then we follow the map so generated.
2. The objects are properly detected only under ideal conditions, i.e. when we have uniformly coloured background and uniformly coloured ground.
3. The distance mapping is not 100% accurate.
4. Since it is not possible to calculate the depth of the objects using just the images from the front, we have to assume a standard depth (say, 30 cm.) for each object.
5. The angle of rotation of the bot is sometimes not accurate, because of slipping of the bot wheels.

HARDWARE USED

1. Raspberry Pi 3 B
2. SRA Development Board
3. Magnetometer
4. 4x GearMotors
5. 2x USB Camera
6. Li-ion battery
7. Powerbank
8. CP2102
9. 4-wheel generic bot chassis





CONCLUSION

This project enabled an extremely fruitful utilization of our summer and also helped further our understanding of autonomous vehicles. The team understood the incredible value of precision in the design of these vehicles and the real-life implications of its lack thereof. Even as such vehicles may be safer than human drivers, even the slightest miscalculation can cause severe loss of life and property, not to mention the loss of faith of the general public in the industry.

The primary deliverable of this project, an autonomous delivery robot, was successfully realized. The team would love to pursue several improvements in the both the software and hardware design of the bot and have it tested in real scenarios to automate the delivery of small to medium sized articles in confined places like restaurants and offices.

REFERENCES

https://www.redblobgames.com/pathfinding/a-star/introduction.html

http://intermediate-and-advanced-software-carpentry.readthedocs.io/en/latest/c++-wrapping.html

https://www.geeksforgeeks.org/a-search-algorithm/

http://opencv-python-tutroals.readthedocs.io/en/latest/

https://docs.opencv.org/3.0-beta/index.html

https://www.pyimagesearch.com/

https://en.wikipedia.org/wiki/Computer_stereo_vision

https://github.com/keepworking/Mecha_QMC5883L