Aerial View Based Guidance System

Abhijan Wasti [1], Saugat Tripathi [1], Sandeep Regmi[1], Sanjeev Yadav[1], and Dinesh Baniya Kshatri[1], Assistant Professor

[1]Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, Tribhuvan University, Kathmandu, Nepal

Corresponding author: Abhijan Wasti (e-mail: tha072bex301@tcioe.edu.np).

ABSTRACT With the limited amount of resources and roads available, traffic congestion seems to be the current leading problem for urban areas. Reliance on the traditional vehicular navigation systems seems to a huge contributing factor, which simply cannot meet the present demand for better traffic management. While the existent navigation system relies on the Global Navigation Satellite System (GNSS) and predefined historic navigation data, a real time optimal pathfinding navigation system has been lacking. We have proposed a close to a real-time system that calculates an optimal path from start to destination using floodfill algorithm and provides guidance to ground vehicles. A camera hovering over the top provides a bird's eye view and via the Hyper Text Transfer Protocol (HTTP) on top of 802.11n protocol, transfers aerial view from the airborne device to Unmanned Ground Vehicle (UGV) where the actual computer vision processing is done in a local processor. The use of floodfill algorithm although requires an image that contains the whole maze but makes the system far more efficient. This makes it real time and efficient which may be further improved and integrated with actual traffic scenarios.

INDEX TERMS Floodfill, Image Processing, Maze Solving, Navigation

I. INTRODUCTION

Traffic congestion is not only a problem in Nepal but in the whole world. According to TomTom's website, there is always the presence of traffic congestion somewhere in the world at this very moment [1]. The main cause behind these congestions is the increase in the number of vehicles. As it accounts for more than 30% of traffic congestion in an urban area, the lack of proper navigation is a serious problem [2]. The main problem of these cases arises around an urban area with a narrow and congested road and tall skyscrapers. The number of vehicles is growing in rapid numbers which will not be decreasing any time soon. Sustenance of a large number of vehicles and to minimize the problem of proper navigation by emergency vehicles is of high priority in the near future. Being dependent only on the Global Navigation Satellite System (GNSS) and the historical data for finding an optimal path around a city is now somewhat obsolete. Thus, the need for smart navigation is an absolute necessity now and in the near future.

Micromouse has been in the forefront to test out algorithms and design for maze solving where localized sensory data are used to navigate the test robot to the final destination [3]. Similar is the case for the vehicles running in an urban route with the knowledge about the route being localized to the driver alone. Micromouse has been used for maze solving and to test out algorithms and their efficiency. It easily relates to the roads and the skyscrapers which form a maze-like structure so they both share a common solution. Each algorithm performs in a different approach to find an optimal solution on the maze, most of them only depends on the input from the subject's point of view. So, instead of using just the subject's point of view having an additional view might give a better approach to find an optimal solution. So, in this project, the use of aerial images, that gave an extra perspective, was taken into the account.

This proposed system can help us sustain a large number of vehicles running and to minimize the problems of proper navigation of emergency vehicle. The proper navigation of the ground vehicle is also needed for the military defense purposes to scout areas ahead of the battalion. Similar applications are seen in commercial unmanned home delivery systems. The proper road navigation of the emergency vehicle and law enforcement agencies can also help minimize precious time during time critical scenarios.

The first part of this paper gives a general introduction and application about the paper and its proposal as well as the problem definition. The second section provides the review of the related works that have guided the proposed system. The

third section defines the methodology used as well as the proposed system architecture. The fourth section provides the implementation details of the system which is then followed by the results, analysis of our work and consequently conclusion and future enhancement of the system

II. RELATED WORKS

For the maze solving, two approaches are used, one without the prior knowledge of the maze and based on the local data collected from the sensor onboard the ground vehicle and one that has prior knowledge of the maze [4]. For the latter, several algorithms have been tested out for finding an optimized path. Dang *et.al.* proposed an algorithm which was based on the flood-fill algorithm but improved by reducing certain repetitive steps. As there are certain parts in the maze where the robot can go only straight forward, when the robot is inside these parts, it does not need to perform all four steps of maze solving which is updating the wall, flooding the maze, determining which turn to be taken and moving to the next cell. This proved to be more efficient than the traditional approach of just using the flood-fill algorithm [5]. Whereas, Patel *et.al.* demonstrated the use of flood-fill algorithms for mobile robotic application. The object detection was done by using pixel-to-pixel scanning of the image capturing the top view of the maze. Then, the flood-fill algorithm finds the shortest path from a defined section of start to finish from user input. Through their research, it was found that the flood-fill algorithm can be effective in finding an optimal path while eliminating the faulty paths which increased functionality efficiency in paths feasibility examination [6]. Aqel *et al.* proposed a maze solving robot system based on image processing. This system captures images, stores it in a computer, preprocesses it, applies suitable graph theories, generates a path and returns a set of commands for an autonomous robot to follow. The process wasn't fully autonomous; human intervention was required to load the relevant image files on to the computer. Graph theory algorithms Breadth First Search (BFS), Best First Search and A* were used, and compared for efficiency. The image coordinates were then converted into actual coordinates for the autonomous bot to follow. Also, the possibility of infinite looping and dead ends were considered. The Breadth First Search algorithm was found to be more accurate and was employed in the final maze solving program [4].

Harik *et.al.* described a system on an industrial area that would operate an Unmanned Ground Vehicle (UGV) according to the commands given by Unmanned Aerial Vehicle (UAV). A UAV is used to provide global coverage of the area allowing the leader to navigate safely in the transportation area through waypoints selected by a human operator. It navigates using a predictive vision-based target tracking controller taking the leader as a target, which makes the communication and computational requirements at a lower level [7]. Similary, Vandapel *et.al.* used aerial lidar sensors to capture the data of the ground terrain. The ground vehicle also consisting of a lidar sensor collected the data which was used to localize the ground vehicle in the absence of Global Positioning System (GPS) coverage. The main challenge that was faced in this study was the presence of the vegetation on the ground which had been difficult to separate out form the rest of the terrain [8].

The approach proposed in our paper is based on having the prior knowledge of the maze i.e. aerial view of the maze over which the flood-fill algorithm can be employed. And using that result of the optimized path the UGV traverses the maze.

III. METHODOLOGY

This section discusses the methods employed for the design and implementation of the system. The data flow diagram of the system is presented in fig. 2. The system consists of an UAV with a downward facing camera, an UGV vehicle with dedicated processors for maze solving and path traversal. Communication between two vehicles was via the 802.11n protocol.

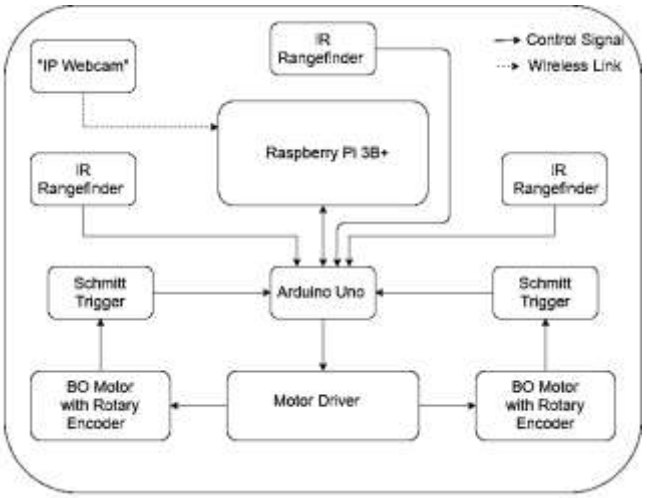

Figure 1: System Block Diagram

While traditional methods of maze solving, most notably those demonstrated in *Micromouse*, do converge to a solution over time, the solution may not necessarily calculate the shortest path in the maze. To calculate the shortest path, the UGV must traverse every possible path to map a complete layout of the maze and only then calculate the shortest path to traverse. This is time consuming and resource intensive. Instead, a top down bird's eye view provides a complete picture of the maze beforehand thus eliminating much of the time taken for initial maze mapping while also guaranteeing the shortest path solution every time.

The proposed system of maze solving is elaborated below:

A. ACQUISITION OF MAZE IMAGE

The top down aerial view of the maze is captured by the UAV camera and sent to the UGV for further processing. The communication occurs via Wireless Fidelity (Wi-Fi) using the 802.11n protocol because of its simplicity in use and cost

effectiveness. The image file is sent using the Transmission Control Protocol (TCP) with an additional option of Real Time Streaming Protocol (RTSP) for dynamic viewing.

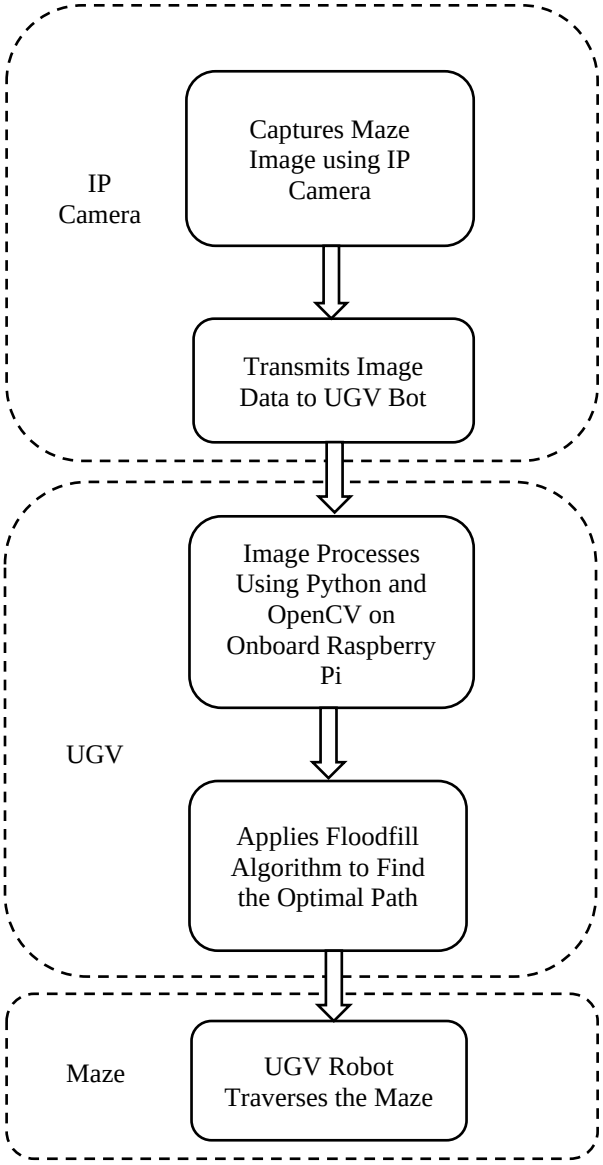

Figure 2: Data Flow Diagram

A. PROCESSING OF IMAGE

After receiving the image from the UAV, the UGV passes the image of the maze for detection of starting and end point of the maze. This is done via Quick Response (QR) codes; QRs placed accordingly denote the starting and ending point of the maze. After this is done, the UGV resizes the image to a manageable format, applies basic image preprocessing steps, such as filtering, converts the image into grayscale and applies appropriate threshold. The result is a binary image with white areas denoting valid paths and black areas denoting walls of maze.

B. MAZE SOLVING

Most image-based search algorithms extract a graphical representation of the maze for application of maze solving applications. In our system, we opted to operate directly on the base image because of ease of use. We employ the floodfill algorithm to "*flood*" the maze from the starting point and lead us to the end point of the maze.

In floodfill, the entirety of the maze is treated as a grid consisting of cells. Initially, an entity is created at the starting cell and added to the stack. This entity checks whether neighboring cells are walls or visited cells. If neither of those are true, a new entity is created in the neighboring cell, the direction of the originating cell is noted and the new entity is added to the stack. This process continues on until the end cell is reached.

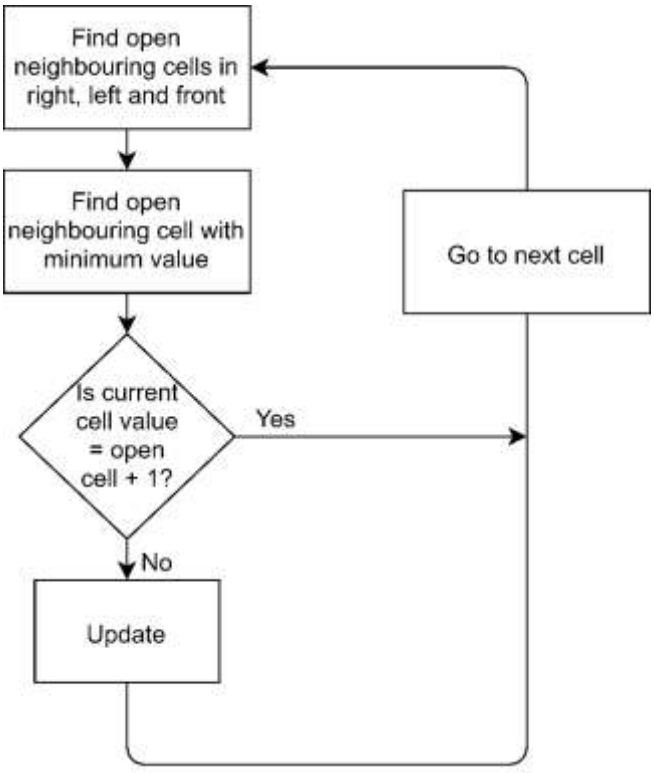

Figure 3: Floodfill algorithm

Upon completion, starting from the end cell, the direction of originating cell of each entity in corresponding cell is checked and added to a list of directions. This list of directions is then reversed and used as a guide for path traversal.

C. TRAVERSAL OF PATH

Based on the set of direction acquired from the previous step, the UGV now starts traversing the maze. The sensors in the UGV check whether there is more than one possible direction to travel i.e. intersection point of multiple paths. If multiple paths are found, the UGV references the list of directions and reads which direction to travel. This process then continues on until the end point is reached.

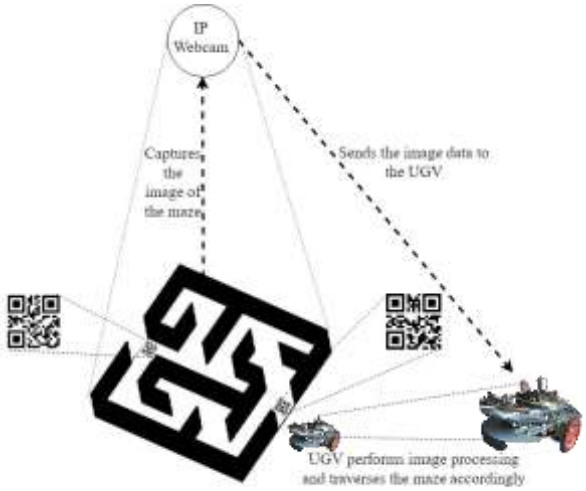

Figure 4: System Overview

IV. INSTRUMENTATION AND SOFTWARE USED

The UGV robot which physically runs on the maze consists of two dedicated processors; one for image acquisition and processing and another for driving robot through the maze. The two processors were supplied with isolated power supply so as to improve battery efficiency.

The Raspberry Pi 3B+, which was responsible for image processing was powered via a buck converter and the Arduino Uno responsible for driving the robot i.e. reading sensor data, driving the robot and traversing the maze, was powered via LM7805 fixed voltage regulator. The two motors onboard the UGV had encoders on them. To receive better signal from motor encoders, Schmitt triggers were used. IR Rangefinders were used for wall detection sensors for maze traversal.

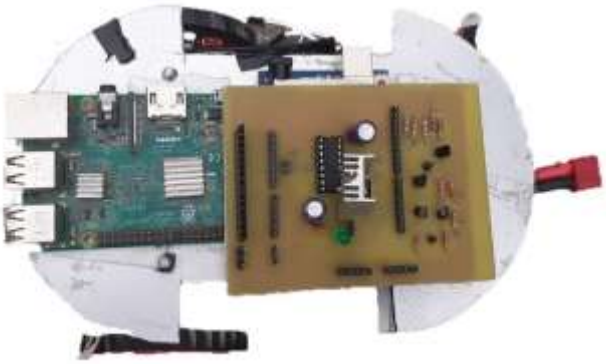

Figure 5: Top View of UGV

The core image processing coding was done in Python language with the OpenCV library. The maze traversing algorithm was coded onto the Arduino Uno using Arduino IDE in C++.

V. IMPLEMENTATION DETAILS

For the purposes of the demonstration an actual maze was made on which an UGV would travel. The maze of size 1590 mm × 1590 mm was made out of Styrofoam pieces cut precisely with 150mm as height of the wall. The maze architecture should be clearly visible from aerial view. So, for this purpose distinct color contrasting to that of the maze i.e. black color cardboard cut into correct sizes were glued to the Styrofoam's top view part.

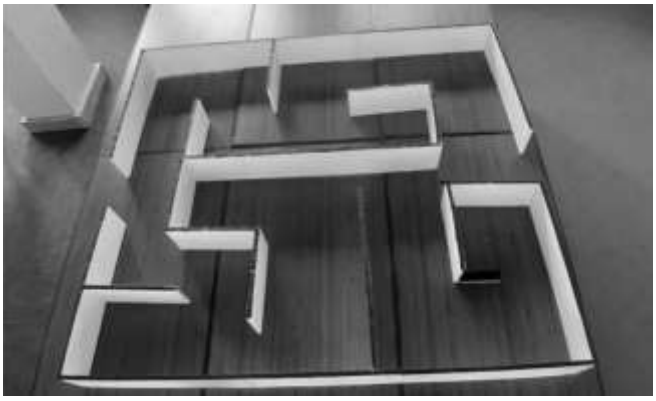

Figure 6: Prototype Maze

The UGV base frame was made using aluminum composite material which had Raspberry Pi and Arduino on board. The Raspberry Pi 3B+ was on the same network as Internet Protocol (IP) camera which is required for the automated transfer of picture. The optical encoder on the motors of UGV produced a signal represented in fig. 7.

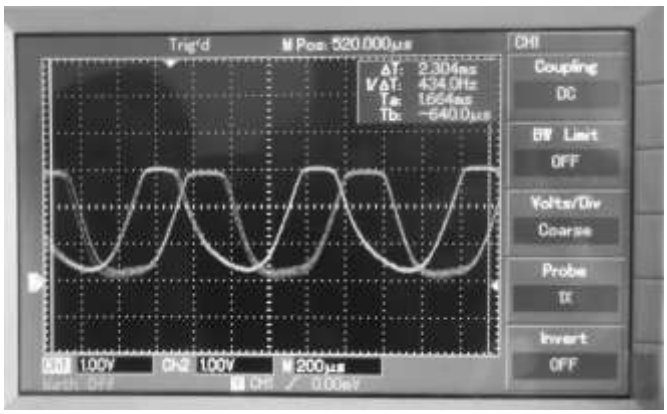

Figure 7: Raw Output of Optical Encoders

The slight roll off at each cycle would produce faulty results. To avoid this, a Schmitt trigger was used. Fig. 8 illustrates the output waveform after passing through a Schmitt trigger.

Figure 8: Corrected Output after using Schmitt Trigger

VI. RESULTS

To test the image preprocessing algorithm and its efficiency, first a 2D maze was created for real life testing. The maze outline was printed onto a flex and initial testing and debugging was done. After successful testing with 2D maze, walls of maze were constructed with Styrofoam and QR codes in entry and exit points. Contrasting colors were used for the wall and floor of maze for ease of processing.

A prototype of the two system was made and was ready for testing. For aerial view, an android application called IP Webcam was used, which would then capture image and send to the Raspberry Pi 3B+ onboard the UGV.

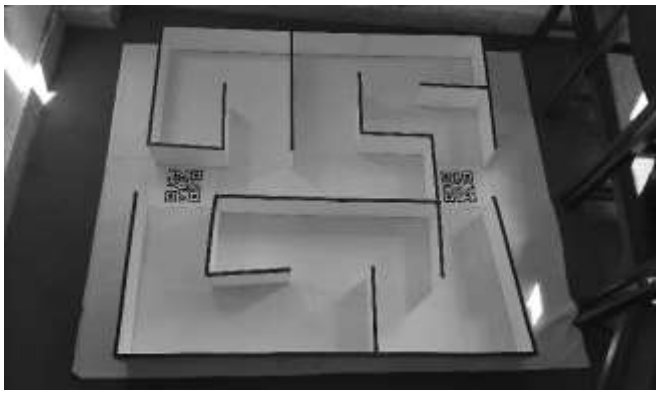

Figure 11: Aerial View of Maze

For application of maze solving algorithm, the QR codes were read, image scaled and preprocessing steps were performed. Before the processing of the image for floodfill the QR code had to be removed as it would result in the imperfection of the maze.

Image preprocessing steps included blurring to removing small imperfection in the image, conversion to grayscale, conversion to binary image and morphological transformations such as erosion and dilation. Thereafter, floodfill was applied; starting from start point, new entities were created based on neighboring cells, the path traced back and the final set of direction was derived.

The path information is then sent to the Arduino Uno which was responsible to drive the UGV on the maze. The UGV would read sensor data to ascertain whether or not a particular cell was an intersection i.e. point with multiple possible paths to travel. When such an interaction was encountered, the UGV would then refer to the array of directions sent from the Raspberry Pi to move in the correct direction.

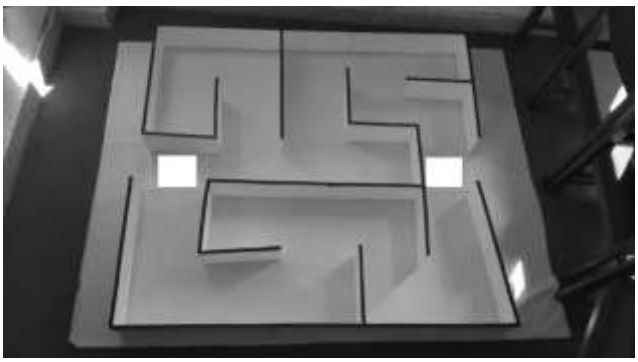

Figure 12: View of Maze After QR Removal

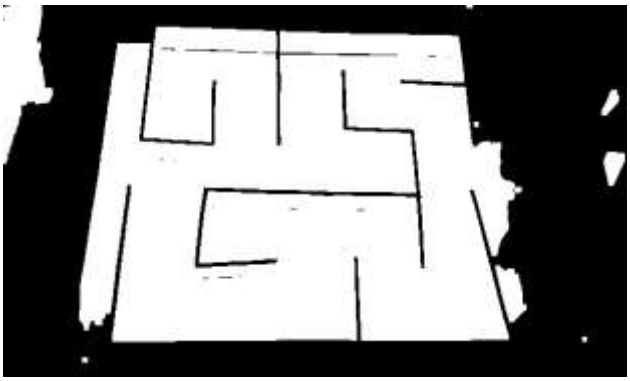

Figure 9: Binary Image Before Floodfill

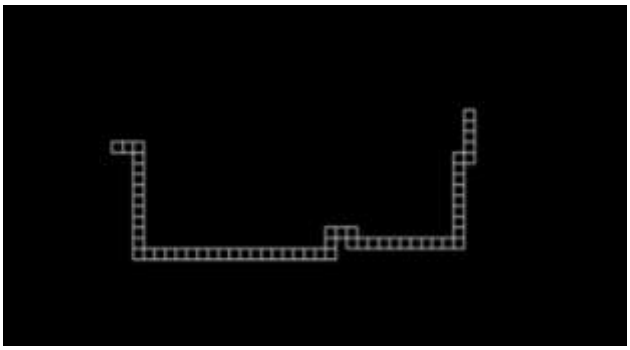

Figure 10: Extracted Path After Floodfill

VII. ANALYSIS AND DISCUSSION

Application of different steps along the process and their analysis revealed the following:

A. DIFFICULTIES WITH IMAGE PROCESSING

When applying image preprocessing steps, notably thresholding and morphological operations, it was non-trivial to ascertain a universally ideal value to isolate just the walls of the maze and not influence the binary image with other image artifacts such as a shadows or imperfection on the path. While better solutions have been created in the real world, for our isolated test case, we opted for manual calibration whenever the test apparatus was set against a new setting.

Based on the height and perspective of camera, the size of the maze appears to be of different size and with slight perspective deformity. By affixing the camera directly above the maze in a horizontal position facing downward, both the size of the maze and the perspective deformity could be avoided.

Since the size of maze varies with placement of camera, the size of individual cells in the grid must vary accordingly to ensure the most efficient use of time and resources when floodfilling. The processing of floodfilling is $O(n^2)$ complex. While smaller cell size ensure meticulous traversal through closely places cells, they take larger time to converge to a solution and are largely affected by imperfections in the image. On the other hand, while larger cell size would be efficient in terms of time and processing power, they take larger strides and might miss smaller details or even entire sections of walls. Thus, size of grid and in turn the cell is an important factor to consider when floodfilling.

B. OUTPUT OF IR RANGEFINDER

The voltage reading from IR Rangefinder was discovered to follow a non-linear curve. As the distance increased, the voltage measure did not scale at the same rate. Fig. 13 illustrates this relationship.

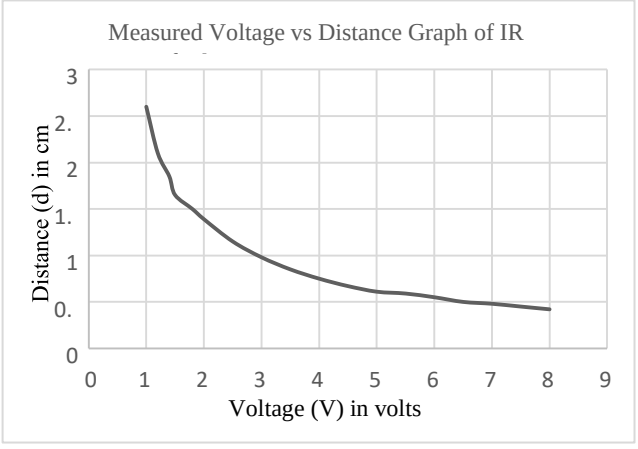

Figure 14: Measure Voltage vs. Distance Graph of IR Rangefinder

C. OUTPUT OF ROTARY ENCODER

The readings from the encoder was quite precise for our purposes. However, due to inertia and friction the actual distance measures deviated from the theoretical distances. Their relationship is shown in the fig. 14.

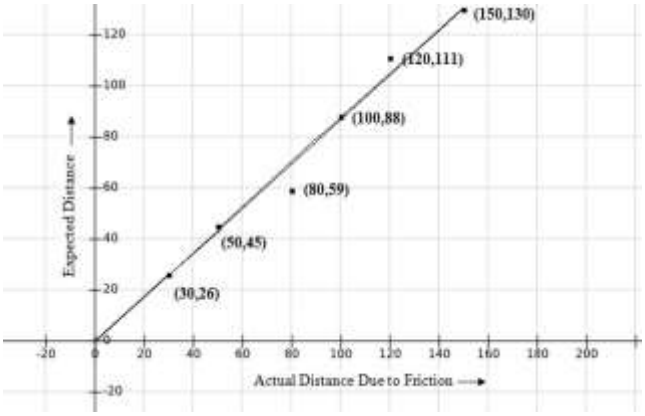

Figure 13: Expected vs Actual Distance Measure by Optical Encoders

VIII. FUTURE ENHANCEMENT

A major part in this project that has been lacking is in the use of real UAV for image acquisition and transmission. Use of real UAVs (a quadcopter) can help implement this idea out in the real world. Similarly, swarm of drones can be used for larger area coverage and efficient route planning. For such a connection of swarm of drones MQTT protocol can be implemented in order to communicate between drones and send the data to the required ground station or directly over to the internet for access by ground vehicle through the internet. The standard of 802.11p can be quite effective for actual vehicular connection as the standard, as it defines a long range communication with lower transmission bandwidth.

For real world implementation, image preprocessing steps can be made more robust against variable lighting changes. Similarly, detection of static as well dynamic obstacles is definitely the right step. The image processing algorithm can be made more robust by the use of machine learning for image segmentation for differentiation of roads and buildings in the real world. Another detail that can be implemented in future works is the use of some heuristic.

IX. CONCLUSION

While aerial view can provide long distances view clearly, ground vehicles are suited for transportation of heavier loads. With the proper coordination of these two units that excel in different fields, they can perform task that neither could perform alone. Despite the inherent disadvantage of using an unlicensed frequency band of 802.11n that is susceptible to interference from the environment, the transmission of image file from the aerial camera to the UGV still performed in a span close to real time. Processing of image file in a general

propose processor used in the project yielded a result that was a good tradeoff between accuracy and time. Under controlled conditions, the entire process from image acquisition to the complete traversal of maze was completed successfully and has proved to be a viable option for future applications. Several factors still come into play that need to be addressed in a real world scenario, however the basic proof-of-concept holds true.

ACKNOWLEDGMENT

The authors would like to express their deepest gratitude to the Department of Electronics and Computer Engineering of the Institute of Engineering, Thapathali Campus for providing access to laboratory equipment and financial support to conduct this study. The authors are also indebted to the Robotics and Automation Center also belonging to the Institute of Engineering, Thapathali Campus for the sharing of technical experience and facilitating the procurement of hardware instruments required to conduct this study.

X. REFERENCES

[1] TomTom, "TomTom," TomTom, 26 05 2018. [Online]. Available: https://www.tomtom.com/en_gb/trafficindex/list?citySize=LARGE&continent=ALL&country=ALL. [Accessed 20 12 2019].

[2] R. Arnott, T. Rave and R. Schob, Alleviating urban traffic congestion, MIT Press, 2005.

[3] L. Wyard-Scott and Q.-H. Meng, "A Potential Maze Solving Algorithm for a Micromouse Robot," in *IEEE*, Victoria, BC, 1995.

[4] M. O. Aqel, A. Issa, M. Khdair, M. ElHabbash, M. AbuBaker and M. Massoud, "Intelligent Maze Solving Robot Based on Image Processing and Graph Theory Algorithms," in *nternational Conference on Promising Electronic Technologies*, Deir El-Balah, 2017.

[5] H. Dang, J. Song and Q. Guo, "An Efficient Algorithm for Robot Maze-Solving," in *IEEE*, Nanjing, 2010.

[6] A. Patel, A. Dubey, A. Pandey and S. Choubey, "Vision guided shortest path estimation using floodfill algorithm for mobile robot applications," in *IEEE*, Allahabad, 2012.

[7] E. H. C. Harik, F. Guérin, F. Guinand, J.-F. Brethé and H. Pelvillain, "UAV-UGV Cooperation For Objects Transportation In An Industrial Area," in *IEEE*, Séville, 2015.

[8] N. Vandapel, R. Donamukkala and M. Hebert, "Unmanned Ground Vehicle Navigation Using Aerial Ladar Data," *International Journal of Robotic Research*, vol. 25, no. 1, pp. 31-51, 2006.