Automated Route Traversal with Simple Collision Avoidance Using a Raspberry Pi, GoPiGo, and A*

Love Bennett, Josh Burks, Greeta Joshi, Ayça Küçükdurmaz, Jacob Morgan
Computer Science, Concord University, Athens, WV.

Abstract

The proliferation of mapping technology has created a market for vehicles that are capable of self-navigation. In fact, what started as the sole enterprise of the single largest data aggregator in the world has evolved into an ever changing, progressing field. Our project aims to take the concepts being developed by companies like Google and Uber and implement them on a much smaller, safer scale while simultaneously serving as a method to learn about these algorithms as well as how to interact with the world using a computer system. In this context, we developed a small-scale self-navigating car using a Raspberry Pi and the GoPiGo bots, capable of finding an optimal path and adjusting its path to avoid obstacles.

Introduction

The Raspberry Pi Foundation is a United Kingdom based organization whose mission statement it is to further open the platform of computer development to people of all ages. In aligning themselves with this goal, they have produced a micro-computer called the Raspberry Pi, a sub-$40 computer with output pins to allow the computer to interact with other electronic devices. This has made it a huge part in the Maker community, offering hobbyists and newbies alike an affordable way to experiment and create new methods of human-computer interactive systems. With this further proliferation, many technologies have been released specifically tailored to the Pi, and among these is the GoPiGo platform, a method of creating a mobile, drivable unit. Using the combination of these two technologies, along with the standard platform in computer vision technology – OpenCV – as well as many popular algorithms in graph searching and edge/object detection, we've developed a methodology for emulating the experimental self-driving cars of Google, Uber, and the like.

Technologies Used

•Raspberry Pi
•GoPiGo Bot
•Python
•OpenCV
•A* Search Algorithm
•Canny Edge Detection

OpenCV and Obstacle Detection

Having a computer "see" is a very complex and computationally expensive process. To help mitigate this problem, we utilized a technology called "OpenCV", where the CV stands for "Computer Vision". This is an open source library to contains many common operations used in computer vision tasks, which are optimized much more than we would be able to produce at an undergraduate level.

One of these algorithms that we used is known as "Canny Edge Detection". This is an algorithm that takes an image as an input and returns an image that is black everywhere except where the algorithm has detected an edge. In general, it works by applying a Gaussian Filter to the image in order to blur the image (within a certain threshold to simplify the image's color complexity), calculating intensity gradients (to quantify how quickly the image is changing colors), applying a threshold (to identify lines along which the colors significantly change fast enough to likely indicate an edge), then pare down the results in order to more clearly define where the edges are.

Using this algorithm along with some probabilistic inferencing, we can predict a collision with a visible object. When a collision seems imminent, the GoPiGo will then take measures to ensure that it will not collide with the object. In this implementation, the bot randomly decides to take a right or left path way around the object, and will adjust its path accordingly.

A* Search

In looking for an optimal algorithm for path-finding, researchers have developed the A* algorithm, a double-valued graph search algorithm that is guaranteed to return an optimal path granted that it meets a specific requirement.

A* takes a graph that has a "cost function", $f_c(x)$ (or simply $f(x)$) - a numerical "cost" to travel from one node to another – as well as a "heuristic function", $f_h(x)$ (or simply $h(x)$) – a function that measures, in some way, the "closeness" of a node to the desired final state. By minimizing the value of $f(x) + h(x)$, A* is guaranteed to return a path that is optimal, given that the heuristic function does not *over-estimate* the "closeness" to completion. Others have used A* to evaluate an optimal way of completing the classic sliding-picture puzzle by using $f(x) = 1$ and $h(x) = \Sigma\, d_i$, where d_i = the Manhattan Distance of puzzle piece i from its final position.

Given our purpose in our experiment (as well as research published by Google on the topic), we defined our functions as follows:

$$f(x) = \sqrt{(x_n - x_c)^2 + (y_n - y_c)^2}$$

$$h(x) = \sqrt{(x_f - x_n)^2 + (y_f - y_n)^2}$$

where (x_i, y_i) is the x and y coordinates of the i-th node, and c, n, f designates the current, next, and final nodes, respectively.

In layman's terms, the cost function is equal to the Euclidean Distance between the current node and the next node (or, the distance between two positions on the map), and the heuristic function is equal to the Euclidean Distance between the next node and the final node (or, the distance between the next position on the map and the final position on the map). By defining these functions in this manner, our A* algorithm is guaranteed to return an optimal path from one point to another.

Process

The project's iterative nature lends itself to a smooth development process where each function contributes to the next. For example, once the GoPiGo robot receives a map to navigate, it's trivial to apply Canny Edge Detection and Computer Vision such that the robot can now avoid obstacles. Following this design process we're able to transform the GoPiGo robot from a naive device that follows a user's direct commands into a robust machine that can make decisions for itself.

Conclusion

Combining A* Search with Canny Edge Detection and applying these algorithms to our GoPiGo robot, we are able to emulate the functionality of Uber or even Google's very own self-driving cars on a smaller and safer scale. This is invaluable data capable of demonstrating concepts relating to shortest path algorithms and artificial intelligence and applying those concepts to emerging technologies. One of our goals in choosing to work on this project was to contribute our findings back to the community so that greater strides can be made in this area of focus.

References

• "Teach, Learn, and Make with Raspberry Pi." Raspberry Pi. N.p., n.d. Web. 05 Apr. 2017.
• "GoPiGo." Dexter Industries. N.p., n.d. Web. 05 Apr. 2017.
• "Welcome to Python.org." Python.org. N.p., n.d. Web. 05 Apr. 2017.
• "OpenCV Library." OpenCV Library. N.p., n.d. Web. 05 Apr. 2017.
• "A* Search Algorithm." Wikipedia. Wikimedia Foundation, 04 Apr. 2017. Web. 05 Apr. 2017.
• "Canny Edge Detector." Wikipedia. Wikimedia Foundation, 03 Apr. 2017. Web. 05 Apr. 2017.