Robotic E-Car for Autonomous Navigation

Alex Benjamin and Evan Nixon, under supervision of Sandeep Gogineni, Ed Richter, and Arye Nehorai
Preston M. Green Department of Electrical and Systems Engineering
Washington University in St. Louis

Abstract

In this project we focus on integrating sensors into a small electrical vehicle to enable it to navigate autonomously. Over the semester we installed a camera on our vehicle and have developed circuitry to allow a computer to control it. We then wrote LabVIEW and MATLAB code to determine pathway edges and generally make turning decisions. This vehicle is currently capable of staying on straight outdoor pathways. We hope to have the vehicle fully capable of autonomous navigation by the end of the semester. Our additional future plans involve expanding the vehicle's sensors to include GPS and compass systems.

Project Objectives

Vehicle Movement

This semester we have primarily been working on algorithms that use camera images to center the vehicle on the road while driving forward.

Future goals include autonomous handling of intersections and irregular pathways.

Obstacle Avoidance

Use camera data to identify people and other obstacles in the roadway. Wait or drive around these obstacles.

GPS Localization

In the future we plan on adding a GPS receiver to the vehicle for mapping applications. We could use the GPS to:

- Store pathway data to implement shortest-pathway algorithms
- Improve handling of irregular intersections
- Deliver packages autonomously

Error Weighting and Outlier Elimination

Error Weighting

We found that the outside points of captured images are more likely to be close to the trend line than points closer to the center – this is a result of greater chance that the middle points have moved onto the road when the car is misaligned. Our weighting function varied from 1 on the outside to 0 on the inside linearly.

Outlier Elimination

Our outlier elimination process is an iterative algorithm which follows these steps:

1) Check if each point is within a specified tolerance from the trend line. If all points are within the tolerance, exit the routine.
2) If a point or points are outside that distance, find the point that is the farthest distance from the trend line.
3) Remove that point and re-run a linear fit.
4) If there are only three points left, stop. Otherwise go to step 1.

Pathway Edge Detection and Decision Making

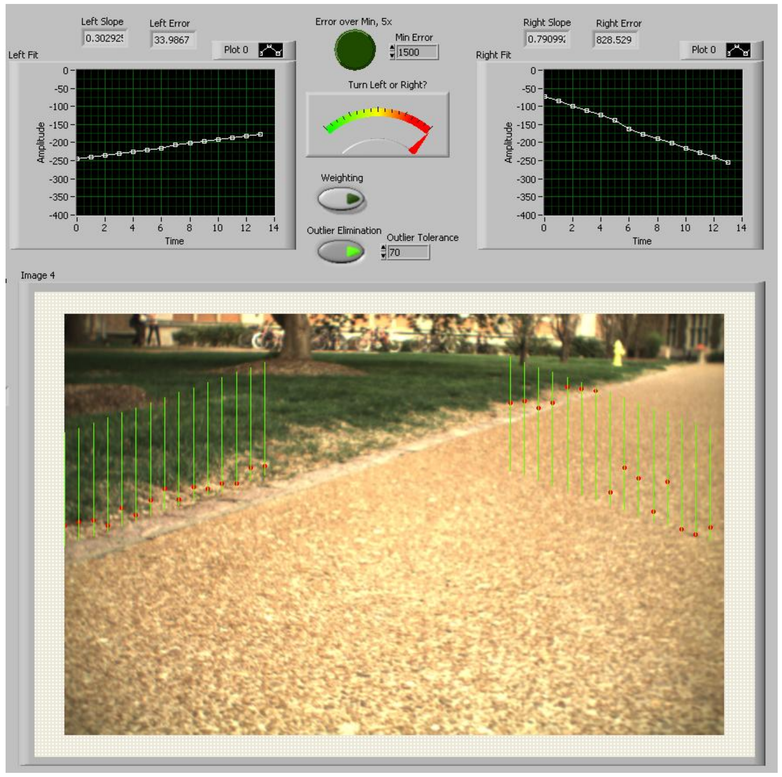


Figure: LabVIEW GUI showing detected edges and turning direction

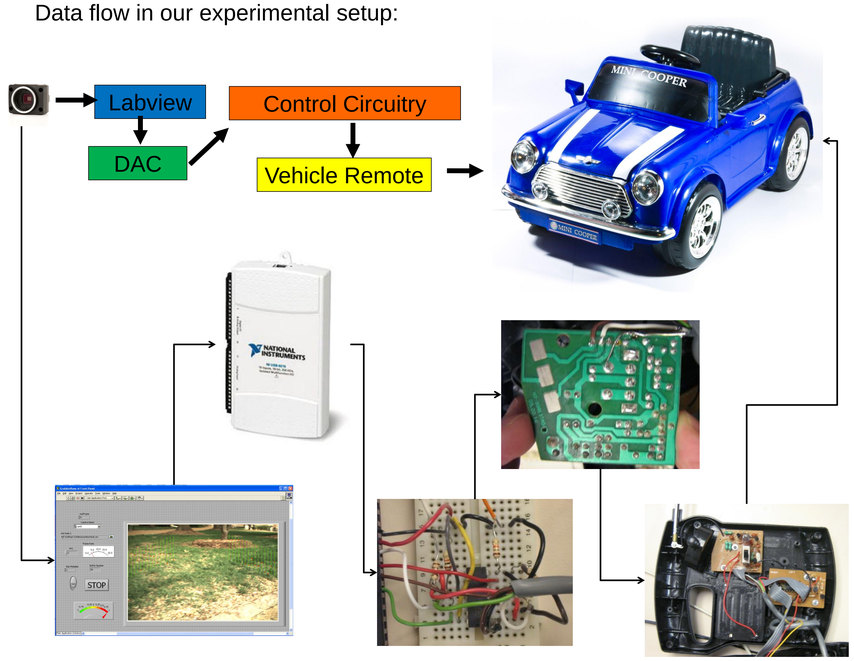
Movement Algorithm

The current decision making process operates as follows:

1) Draw 15 pre-specified lines on the right and left third of a captured image.
2) Find the point of greatest intensity change along that line.
3) Apply a linear fit to the points on the right and left side (separately).
   a. (If desired) Apply pre-weighting to points before linear fit to favor those less likely to contain error.
   b. (If desired) Apply iterative outlier-elimination method to remove points abnormally far away from the trend line.
4) Compare slopes of trend lines and move towards the steeper slope.
   a) Override this calculation if linear fit mean-squared error is above pre-set threshold. Head towards side with overly high error.
   b) (If desired) Do not turn if the slopes are within a certain tolerance of each other
5) Send turning information to the vehicle, specifying the duration of the turn signal sent to the servo motor controlling the wheels.

Experimental Setup and GUI

Data flow in our experimental setup:



Design Considerations and Future Plans

Adaptive Edge Detection

In order to improve the accuracy of our edge detection techniques and allow for more robust navigation algorithms, we would like to implement an adaptive edge detection algorithm. Rather than finding edges in each image frame in an inverted V pattern (as seen in the center image), the new algorithm will take a number of horizontal and vertical slices of the image, and aggregate the best linear fit. Using this methodology, we should be able to reduce the influence of outliers, as well as linear boundaries that are not the correct path (for example, intersecting sidewalks).

Sensor Integration

While the image processing from the camera provides a framework for navigation, fusion of multiple sensors will allow the vehicle to gain a more accurate picture of it's surroundings. For example, a GPS and compass system will provide real-time direction-of-travel information. This will be useful both in staying on the path, and navigating to a specific point. The GPS would also be a necessary component to dynamically recording a map as the car travelled.

Obstacle Avoidance / Reactivity

A Microsoft Kinect camera is able to provide information on depth of objects in it's field of view, as well as distinguish human forms and specific movements. Using the Kinect (in addition to our Firefly camera), the car would be able to dynamically react to changes in it's environment. This would include detecting pedestrians, and then waiting or navigating around them.

The Kinect product is released with an open source driver for passing sensory information. A significant task would include programming the software interface to send and receive information from the device.