A Report On: Design and Implementation of (an) Autonomous Car using (a) Raspberry Pi

William Mollenkopf
November 03, 2015

Abstract:

The paper I researched is titled: "Design and Implementation of Autonomous Car using Raspberry Pi" by Gurjashan Singh Pannu, Mohammad Dawud Ansari, and Pritha Gupta. It was written in the International Journal of Computer Applications Vol.113-No. 9 (March 2015).

The paper concerns their project which aimed to create a monocular vision autonomous car prototype using Raspberry Pi as a processing chip. Along with an HD camera, ultrasonic sensors, they set out to create an autonomous car that can detect obstacles, roads, and perform the autonomous tasks necessary for the car.

The purpose of their research is to research cars capable of reaching a given destination safely, and help avoid the risk of human errors during driving.

1. Introduction

As you the reader may be aware, Google in recent years has developed a highly advanced autonomous car. However, when you begin to take the to question what exactly is allowing their car to behave autonomously you begin to realize that under the advanced algorithms exist conceptually simple components to make such a thing possible. For example, object detection can be performed many different ways, through several different types of sensors - such as Ultrasonic sensors for example. And obtaining images of the road is nothing more than a camera taking a photo and then performing image processing algorithms through software.

So, as like myself, the thought occurs that perhaps something as simple as a Raspberry Pi and some simple sensors can be used to help create, learn and evolve an autonomous car that can detect obstacles, roads and drive along unknown roads on it's own - this is what the authors of the research paper that this paper focuses about has done exactly. With the intent to help lead to a future where human driving error can help be eliminated and avoid costly accidents, injury or deaths.

2. Hardware

The hardware used consisted primarily of a Raspberry Pi Model B, Rev. 2 as presented in Fig. 1 below.

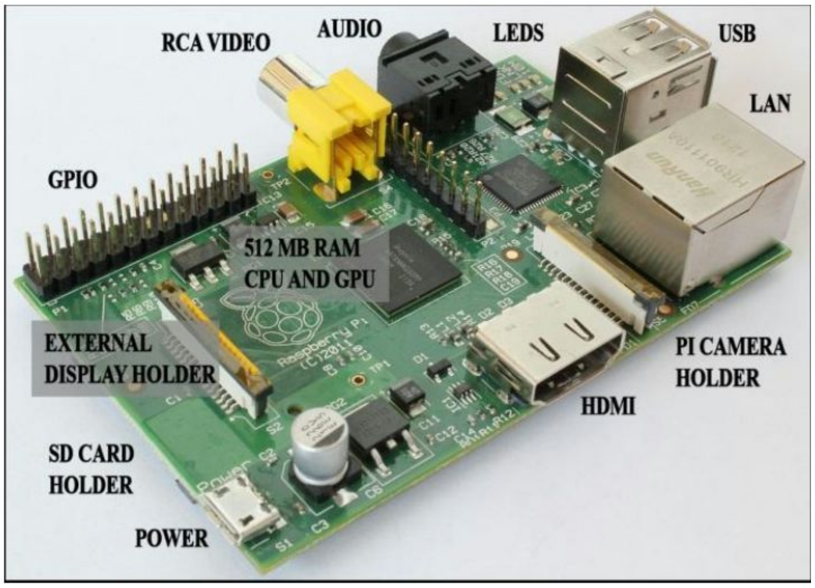

Fig. 1. a Raspberry Pi B Revision 2

The Raspberry Pi in itself is a very simple device with a series of input and output pins known as the GPIO pins. These pins, as well as the Pi Camera interface port (labeled as “PI CAMERA HOLDER”) - help to control various sensors connected to the Raspberry Pi.

Other hardware used includes: A Pi Camera for HD Photos, A USB Wi-Fi 802.11n adapter, Ultrasonic sensors and an L293D Motor driver.

3. Software

The operating system they decided to use was the Raspbian OS for it’s ease of use and optimized performance specifically for the Raspberry Pi (as opposed to other operating systems such as Arch Linux).

Other software used includes the RPi.GPIO Python Library which can be used for manipulating the GPIO pins extremely easily compared to other methods[1] and was a favorable choice among the authors due to that ease of use and implementation.

The OpenCV library was also used, which is a C++ program, but their implementation used Python wrappers in order to make use of it with the RPi.GPIO Python library.

The OpenCV library consists of over 2,500 optimized computer vision algorithms and is an open source library. Essentially the OpenCV stands for “Open Source Computer Vision (library)”. This library can perform such computer vision algorithms such as: image processing, detection, facial recognition, object identification, classification actions, traces, and other functions as well. It's used extensively by companies like Google, Yahoo, Microsoft, IBM, Intel, Sony, Honda, Toyota, and many startup companies as well. This library helps to detect the roads and guide the car on unknown roads.

4. Hardware Implementation

The Raspberry Pi and Pi Camera are attached to the “top shelf” of the car. The L293D motor driver is capable of synchronously controlling two motors at the same time. There exists one motor for each wheel, and two motor drivers control two motors each. One motor driver controls the left wheels, and the other motor driver controls the right wheels. In this way the car is able to move forward, backwards or make turns as well using only two pin signals from the Pi.

Using this scheme, the car is able to make turns as well, by for example, having the left two wheels move forward, and the right two wheels move backwards, the car will turn towards the right. This is how the car is steered and primarily how it’s direction is controlled as well.

5. The Car

The exact details of the car were never mentioned in much detail within their paper. The most mentioned is that it’s simply a “A pre-built four wheel drive (4WD) chassis” leaving much to the imagination. Based on their hardware implementation scheme of the motor drivers and motors, I strongly believe the car is nothing more than a small RC sized car. I did some searching through the resources the paper listed and found one that was very similar to the research they conducted and noticed a car used that sounded very similar (if not exact) to what the authors of this paper had used. This car can be seen below in Fig. 3.

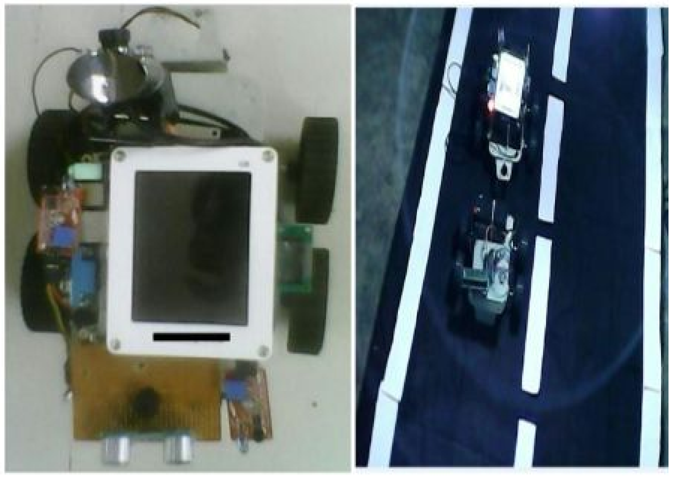

Fig. 2. An suspected image of the car that is similar to what was used by the authors of this paper.

6. Lane Detection Methodology

For the interest in length of this paper, this section has been reduced to simply state that the lane detection methodology used is simply a combination of both feature and model based lane detection algorithms that will be explained in detail in the main algorithm section below.

7. The Main Algorithm

The main algorithm runs in 7 separate steps beginning in order to render the image obtained by the Pi Camera and make decisions about the road itself (if it should turn, continue forward, and so forth).

7.1. The Main Algorithm: Step 1

The first step of the algorithm converts the image obtained into a binary image by extracting the color range for the road (directly in front of the car itself) and the areas that are assumed to not be the road. Determining the color range is only performed when no color range exists, or the color range is deemed incorrect by step 6 of the algorithm.

7.2. The Main Algorithm: Step 2

The second step in the algorithm generates a region of interest that is usually no more than half of the image since much of the upper half of the image is not necessary (sky and other other horizon features). This Region of Interest (ROI) starts from the bottom up due to the position of the car being located closer to the bottom part of the image and is essentially a trapezium shape.

7.3. The Main Algorithm: Step 3

The third step is to use approximations and generate contours of the binary image, within the region of interest. This simplifies the image greatly and reduces a lot of noise as well.

7.4. The Main Algorithm: Step 4

The fourth step draws Hough lines on the image generated by step 3 and creates well defined straight lines for possible road edges (some of which are actually just noise).

7.5. The Main Algorithm: Step 5

The fifth step reduces the Hough line noise by removing lines that are surely not road edge lines. For example if a line on the left side of the image and is tilted to the left, then that line is surely not an edge for the left side of the road, since the line edges will actually tilt inwards (from bottom up) similar to the left and right edges in a trapezium itself actually.

7.6. The Main Algorithm: Step 6

The sixth step implements a fault tolerance for calculating the road edges. Since the roads are usually very smooth and not abrupt, if the lane edges that are detected do not line up properly with the previous line edges, the lane edges calculated are discarded. If this occurs 3 times in succession, the color range from step 1 is re-calculated to allow for this step to actually succeed. This sort of event can be caused by temporary irregularities in the road that may occur. Such as patches of dirt that match the road color range, thus throw off where the edge of the road is believed to be, three frames of this sort of error can be ignored since within 3 frames the same path should still be valid, but after 3 frames, the color range needs to be recalculated.

7.7. The Main Algorithm: Step 7

The final step of the algorithm is to take the region of interest and divide the portions into 20%, 30%, and 50% regions (from the bottom up). Each region is compared to determine essentially if there's some change in the road such that the road may contain a turn. If a turn does exist, the car's motion is changed to allow for the car to follow the turn ahead.

8. Development And Implementation Phases

Development for the car was divided into three separate sections to slowly implement portions of the car necessary for autonomous functionality.

8.1. Development And Implementation Phase 1

The first phase was to implement a way to remotely control the car manually. To do this the authors created an Android App that directly connects to the Raspberry Pi via Wifi. The application was a very simple forward, backward, left and right turning control button app.

8.2. Development And Implementation Phase 2

The second phase was to implement the Ultrasonic sensors to perform object detection for flat objects, and to have the car stop and rotate itself around an obstacle if one is found within 1 meter or less of the car.

8.3. Development And Implementation Phase 3

The third phase was to put everything developed and created together and put the car for a test drive.

9. Results

The results were that the car was able to perform basic autonomous actions including obstacle avoidance, and recognizing an unknown road, and driving down that road on it's own.

10. Conclusion

The conclusion of the paper was that that attempts to create an autonomous car were successful. My personal conclusion is that as well, but with the hopes that more research can be done to simplify the implementation of autonomous driving on actual human drivable automobiles - more similar to what Google and Tesla have and are doing currently.

12. References and Relevant Literature

[1] Gurjashan Singh Pannu, Mohammad Dawud Ansari, Pritha Gupta. "Design and Implementation of Autonomous Car using Raspberry Pi" in International Journal of Computer Applications Vol.113-No. 9 (March 2015) <http://research.ijcaonline.org/volume113/number9/pxc3901789.pdf>

[2] Narayan Pandharinath Pawar, Minakshee M.Patil. "Driver Assistance System based on Raspberry Pi" in International Journal of Computer Applications Vol.95-No. 16 (June 2014) <http://research.ijcaonline.org/volume95/number16/pxc3896794.pdf>

[3] Raspberry Pi Foundation. "GPIO: Raspberry Pi Models A and B" <https://www.raspberrypi.org/documentation/usage/gpio/>

[4] Python Software Foundation. "RPi.GPIO 0.5.11 : Python Package Index" <https://pypi.python.org/pypi/RPi.GPIO>

[5] Wikipedia. "Hough transform" <https://en.wikipedia.org/wiki/Hough_transform>

[6] Wikipedia. "Ultrasonic transducer" <https://en.wikipedia.org/wiki/Ultrasonic_transducer>