Proceedings of the Annual General Donald R. Keith Memorial Conference
West Point, New York, USA
April 29, 2021
A Regional Conference of the Society for Industrial and Systems Engineering

Prototyping a System of Cost-Effective Autonomous Guided Vehicles

Yuhan Bae, Eduardo Gomez, Alexis Haywood, Julio Lazo, Peter Whitson, and Yong Wang

SUNY - Binghamton University
Department of Systems Science and Industrial Engineering
Binghamton, NY

Corresponding author's Email: ybae6@binghamton.edu

Author Note: Our team of Industrial and System Engineers worked on this project for several months. The project included entirely new programs and with it, a large learning curve. The team spent a great deal of time researching and asking questions. We could not have done it without the help of Kanishkan Tamilarasan, Qianxiu Ai, James Henenlotter, and Dr. Mark Poliks. We greatly appreciate their support.

Abstract: The objective of this project is to create a cost-effective system of autonomous guided vehicles (AGV) that utilizes deep learning. TensorFlow and OpenCV will be utilized for image processing, lane tracking, and integrating robotic car communications with a Raspberry Pi board. This may be implemented in environments such as warehouses, hospitals, and more. There are five main steps in achieving autonomous navigation which include image processing, lane tracking, and object detection. This AGV will use an onboard camera to accomplish these steps within the system. Our success will be measured by assessing the behaviors of the robotic car including object detection and accuracy of movements.

Keywords: Artificial Intelligence, Autonomous Navigation, Object Detection, Path Tracking

1. Introduction

As manufacturers and distributors continue to grow in an era of technological development, Industry 4.0 has shaped improvements to industry such as cloud computing, internet of things, robotics and more. The introduction of this technology has improved the overall quality of life, efficiency, and work productivity (Li et al., 2011). It has influenced methods to reduce labor costs, human error, and increasing safety within facilities. Industry 4.0 has brought in a new wave of "smart technology" using artificial intelligence (AI). AI technology has evolved significantly over the last decade and many companies have taken advantage. Large companies have implemented autonomous guided vehicle (AGV) systems similar to one this team is developing. Amazon currently owns its own AGV system, and it is blocking other companies from using its technology (Edwards, 2020). AGVs have a high capital cost and require physical changes to the work environment since they depend on landmarks and sensors on the ground to achieve navigation and positioning. As per its cost and labor, these systems are not yet financially viable options for smaller companies, although necessary to remain competitive.

The aim of this project is to prove and demonstrate the use of autonomously guided vehicles at an economical level for smaller companies. To do this, the team first constructs a simulation of a small warehouse consisting of three sources, four servers, a workstation, and a sink. This simulation models four main pathways that these vehicles would follow. The simulation model runs for 8 hours, comparing human labor and robotic cars. The results show that the robotic car saves an average of 1.5 hours and completes 81 more observations compared to human simulation. These vehicles are able to run continuously without breaks. Assuming that workers are paid $15/hr, the autonomous vehicles would save $112.5 a week totaling $5,850 in yearly savings. This preliminary evidence supports that the addition of autonomous vehicles are more cost effective.

This project creates a prototype of an autonomous robotic car, which has the potential to be scaled to multiple cars. An on-board camera is used to collect data from the environment and relay this data to the Raspberry Pi 3B+ located on the robotic car. The Pi is able to utilize image processing, lane tracking, behavior cloning, and object detection using TensorFlow and OpenCV (Tian, 2020). A camera is placed on the robotic cars allowing for commands to be transferred from the image processing software to a Pi board. The Pi board communicates with the driver board. The drive board directs and powers the movement of the robotic cars. The image processing is able to detect lines, using coded modules, similar to Lane Keep Assist Systems (Tian, 2020). The robotic car uses this system to drive and maintain a position within the lanes. With further implementation, the robotic car is able to perform given functions using the navigation from this image processing software. Using the navigation system, the robotic car will be able to travel throughout multiple environments such as warehouses and hospitals to complete tasks.

2. Method

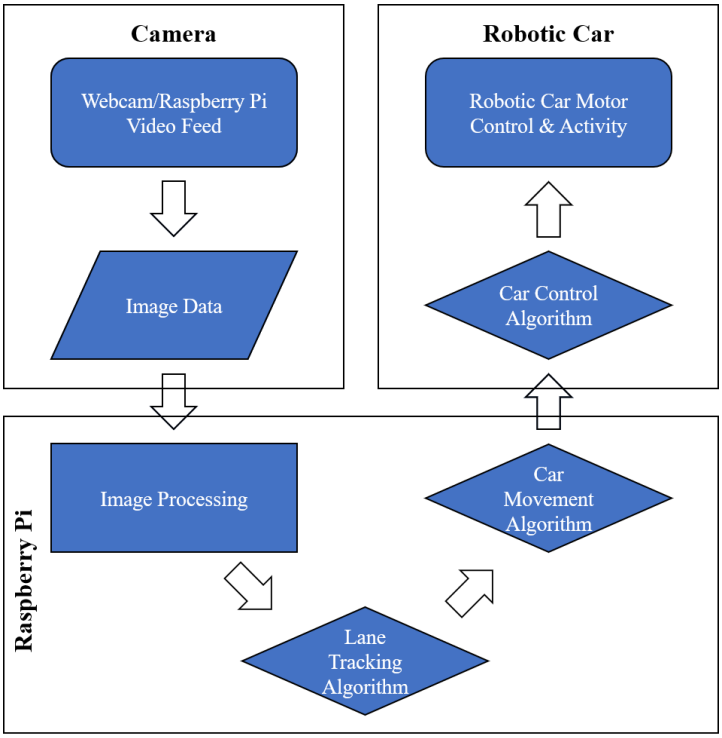


Figure 1. System Structure

2.1 Hardware

In order to create a viable prototype for small/medium businesses, hardware design and cost was a priority. The robotic car is built on a plate set. Two DC gear motors deliver power to back wheels. Front wheels are moved by a SF006C servo that controls direction. In the front of the car, a camera is connected to two servo motors that provide x and y movements, allowing for the car to utilize various angels once autonomous driving is engaged. The camera is connected and provides live video to a Raspberry Pi 3B+. The Pi board performs all algorithms and connects to a PCA9685 PWM Driver and TRA9118A Motor Driver. The entire robotic car is powered by two 18650 Li-ion 3.7V batteries. With this specification of Raspberry Pi, it can omit 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless communication. The robotic car has dimensions of 10”x 7.2”x3.4” and weighs 1.1lbs. When fully configured and assembled, the car costs roughly $200. All parts are sourced within the United States and can be replaced. The Raspberry Pi boards utilize open-source software, which allows for job specific programming to be uploaded. Overall, the design is an operational model that is modular. Additional components can be added to further enhance the functionality and improve system operations.

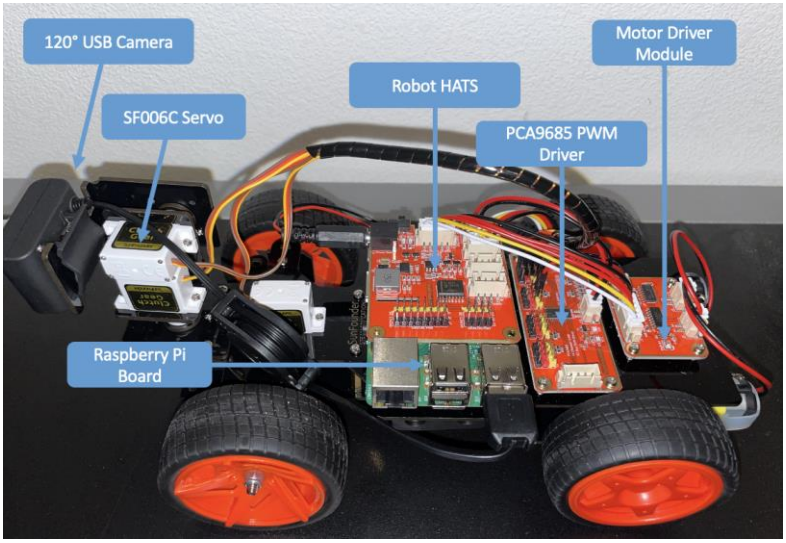


Figure 2. Robotic Car Schematic

2.2 Image Processing

This project relies heavily on the use of machine learning (ML) applications such as TensorFlow, OpenCV, and Python to obtain image processing, object detection, and lane navigation.

2.2.1 TensorFlow Object Detection

TensorFlow, created by the Google Brain team, allows for ML and numerical processing. The open-source program combines many ML and neural networking libraries to create a comprehensive program for users (Models & Datasets, 2021). The TensorFlow object detection API contains pre-trained models referred to as the Model Zoo. The project utilizes the Common Objects in Context (COCO) dataset due to its large-scale object detection, segmentation, and captioning dataset (COCO, 2021). Pre-trained models from the MobileNet SSD V1 set are utilized in addition to novel objects such as "Robotic Car", "Box", and "Book" (Shi, 2020). This is done to specify certain objects that might appear in the project environment excluding the objects found in the pre-trained models. More than 50 images of the novel objects were individually trained, added to the library of pre-trained objects, and tested. The newly trained TensorFlow software would be the backbone of the project's object detection capabilities.

2.2.2 Lane Tracking

OpenCV is an open-source computer vision library that aids in the development of machine vision applications and interfaces. Its numerous algorithms can be applied to interfaces such incorporating C++, Python and Java, making it an ideal program to use on a low powered computer such as a Pi (Bradski & Kaehler, 2008). OpenCV utilizes the live feed streamed from the onboard camera to capture the image frames and process the environment. In this project, the path environment is created with the use of blue tape and white paper. Blue colored tape allows lane detection software to isolate color and enhance edge detection along with the white paper surrounding it to prevent any lined floor patterns from interfering with the edge detection. In order to perform lane navigation, OpenCV is programmed to first isolate all blue colors within its camera frame. Canny edge detection, a multi-stage OpenCV algorithm, then reduces the noise to detect the edges of the color isolated frame (Iqbal et al., 2019). Since the output from Canny is simply a collection of pixels, Hough Transform determines the pattern and distance of pixels to output the shape which in this case is lines. In the case of a curved path, the car is coded to detect one, two, or no lanes. Max deviation for two lines detected is set at 5 radians and for one-line detected set at 1 radian with the car stopping upon detecting no lanes at all. Working together with the pre-trained or manually trained models of TensorFlow the system can detect the data collected and classify them based on the TensorFlow model in place (Talele et al., 2019).

2.2.3 Testing

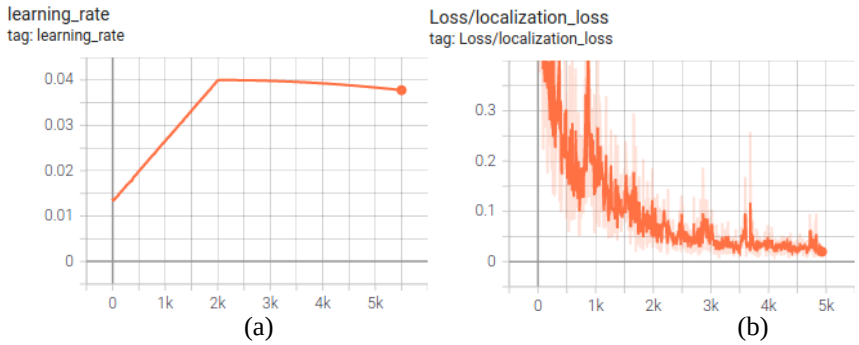
For object detection, the trained TensorFlow model testing is performed by placing objects in front of the robotic car

and determining the accuracy of the recognized objects.

For lane navigation, three test environments were created: straight lane on tiled floor, straight lane with white paper, curved lane with white paper. The control environment contained tiled, colored flooring and to determine if there is a difference in the lane edge detection, white paper was placed in the middle of the lane as well as 8 inches on either side of the lanes.

3. Results

As the model trains, TensorBoard is used to visualize different types of data for the model after each epoch. Training of the model continues until a satisfactory loss is reached (loss approaches 0) at which point the training process terminates. In the current model, the loss and learning rate stabilized after 4,000 epochs. As seen in Figure 3(a), the learning rate begins its downwards trend, indicating the model has already improved upon the speed and accuracy of its object detection capabilities since the start of training. The robotic car is designed to operate based on previous trained commands, which allows it to detect and plan more efficiently.



(a) (b)

Figure 3. TensorBoard Learning Rate (a) and Localization Loss (b) Adjustments in the Scope

Pre-trained objects such as “cell phone”, “person”, and “keyboard” were detected with accuracy percentages of 76%, 56%, and 62% respectively. The frames per second (FPS) was also determined to be 5.10 (Figure 4a) and 5.08 (Figure 4b). Factors including lighting, background, and noise within the camera image impacted the accuracy and box detection.

(a) (b)

Figure 4. TensorFlow Lite v1.15 Object Detection (a & b)

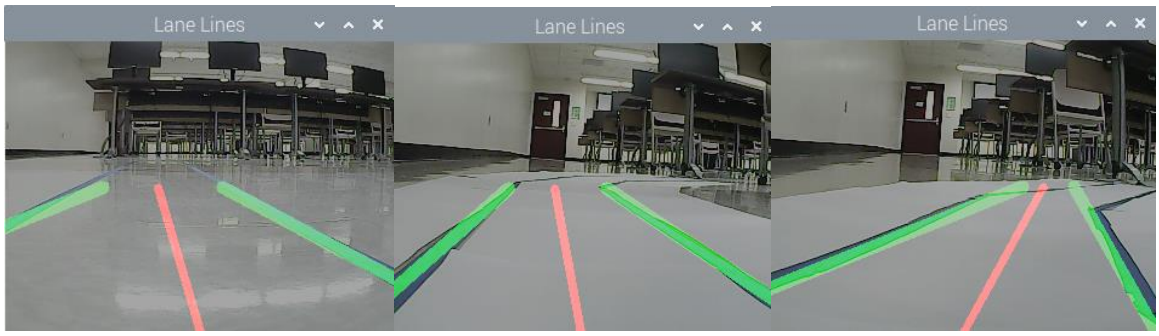


Figure 5. Control Environment (a), White Paper (b), White Paper Curved Lane (c)

In Figure 5a, the green lines show the boundaries of the lanes while the red line determines the estimated path the robotic car takes. Of the 10 trials, at 0.166 m/s, the robotic car veered passed the left lane on 4 occasions. Figure 5b shows an isolated path environment with the white paper encompassing the camera's vision. Of the 10 trials conducted, at 0.166 m/s, the car strayed off to the left 1 time. The improvement in lane navigation allowed the team to conduct the final testing environment with white paper on the ground. Of the 10 trials conducted with white paper and curved lanes, at 0.166 m/s, the car strayed off 3 times.

4. Discussion

The project calls for the use of high-computational applications such as TensorFlow and OpenCV. Unfortunately, due to the processing speed and RAM capacity these programs individually require the full usage of the Raspberry Pi's computational power. The team looked at accelerators to assist with this issue, however, is unable to obtain them due to global unavailability. Thus, tests were conducted for object detection and lane tracking individually. In the future, the project would contain an accelerator to perform these two processes simultaneously.

Lane navigation is improved upon adding white paper on top of the colored tile flooring. However, due to the fact that the entirety of the environment is not consistent with the white paper background, the robotic car attempts to follow the edge of the paper instead. In the future, consistent flooring would be used so as to not confuse the robotic car on edges other than lanes given.

AGVs worldwide are proven to reduce costs and improve efficiency. They can autonomously perform tasks within warehouses or facilities. These tasks are also able to be performed without violating safety standards. The AGVs can detect objects and adjust accordingly, after successful training. However, implementation of these systems come at a high cost and are limited to certain companies. Our team is attempting to create an AGV system that completes similar tasks using existing infrastructure at a fraction of the cost. A prototype of a lane tracking, object detecting robotic car was created. Future works will allow the onboard camera to be able to perform collision avoidance to improve upon its safety standard, improve the accuracy of object detection and lane navigation, and to determine carrying capacities all to further simulate real-life scenarios.

ISBN: 97819384962-0-2

5. References

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly Media, Inc.

COCO - Common Objects in Context. (2021). COCO. Retrieved March 10, 2021, from https://cocodataset.org/ #homeEdjeelectronics/tensorflow-lite-object-detection-on-android-and-raspberry-pi. (n.d.). GitHub. Retrieved April 19, 2021, from https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi

Edwards, D. (2020, January 21). Amazon now has 200,000 Robots Working in its Warehouses. Retrieved March 01, 2021, from https://roboticsandautomationnews.com/2020/01/21/amazon-now-has-200000-robots-working-in-its-warehouses/28840.

Iqbal, B., Iqbal, W., Khan, N., Mahmood, A., & Erradi, A. (2019). Canny edge detection and Hough transform for high resolution video streams using Hadoop and Spark. Cluster Computing, 23(1), 397–408. https://doi.org/10.1007/s10586-019-02929-x

Li, Q., Adriaansen, A., Udding, J., & Pogromsky, A. (2011). Design and Control of Automated Guided Vehicle Systems: A Case Study. *IFAC Proceedings Volumes, 44*(1), 13852-13857. doi:10.3182/20110828-6-it-1002.0123

Models & Datasets. (2021). TensorFlow. Retrieved March 12, 2021, from https://www.tensorflow.org/resources/models-datasets.

Shi, Y. S. (2020). tensorflow/models. GitHub. https://github.com/tensorflow/models/blob/master/research/object_detection/ g3doc/ tf1_detection_zoo.md

Talele, A., Patil, A., & Barse, B. (2019). Detection of Real Time Objects using TensorFlow and OpenCV. *Asian Journal For Convergence In Technology (AJCT)*.

Tian, D. (2020, March 19). Dctian/deeppicar. Retrieved March 29, 2021, from https://github.com/dctian/DeepPiCar