PyRoboCar: A Low-cost Deep Neural Network-based Autonomous Car

Gordon Hendry [gih5305],[1] Seth Harris [sh5144], [1] Ryan Goodwin [rmg4626], [1] Leonid Neverov [ln4227], [1]
Yunkai Xiao [yxiao28], [2] David Tian [dctian], [3] Yang Song [songy] [1]
1. University of North Carolina at Wilmington, Wilmington, North Carolina 28407, USA [@uncw.edu]
2. North Carolina State University, Raleigh, North Carolina 27695, USA [@ncsu.edu]
3. Google LLC, Mountain View, California 94043, USA [@google.com]

***Abstract*—In this innovative practice WIP paper, we present PyRoboCar, a low-cost deep neural network-based autonomous car project. PyRoboCar is a small-scale replication of a real self-driving car using a deep convolutional neural network (CNN), which takes images from a front fisheye camera as input and produces car steering angles as output. PyRoboCar uses a similar network architecture as industry-level autonomous cars and can drive itself in real-time using a camera, an additional Tensor Processing Unit, and a Raspberry Pi 4 platform. We have also made this project open online, including the code and instructions.**

Keywords—Autonomous car, educational robotics, deep learning

I. INTRODUCTION

In recent years, the use of educational robotics has increased in STEM education as a creative process based on a hands-on learning experience. By participating in the construction, programming, and manipulation of a robot, students have opportunities to apply their knowledge of science, technology, and social skills. Although it does not imply a pedagogy by itself, educational robotics can play a crucial role of being an exciting tool in the lab to teach entry-level concepts, and a comprehensive science lab setup to teach upper-level topics like physics [1] or artificial intelligence (AI) [2].

Although educational robotics is still not considered as a required component in STEM education, the recently published robotic projects envision a series of positive contributions on: 1) academic performance [3], [4], 2) interest/motivation [5], 3) teamwork/social skills [5], 4) creativity [6], and 5) problem-solving skills [7].

In this paper we present the PyRoboCar as a compelling, realistic, and extensible educational robotics project. This project primarily targets high school and college-aged students by providing an end-to-end project, from hardware assembly to deploying a deep learning model for autonomous driving. All autonomous behavior is implemented onboard in Python using computer vision and deep learning framework.

II. RELATED WORK

In a robotics project for education purposes, the project should be of 1) reasonable price (affordable by schools, or even students), 2) provide sufficient computational power (real-time processing and decision making), and 3) extensible (to new ideas, new hardware such as light sensors). In this section, we review some of the recent robotics projects for educational purposes or have the potential to be adopted by educators. We especially focus on reviewing the hardware, the cost, and the extensibility (reviewed project summarized in TABLE I).

DeepPicar (the authors are from University of Kansas and Indiana University; in this paper, we call it **DeepPicar by KU and IUB** since there are two projects that used similar names) is a robotic car project completed in 2018 based on a deep convolutional neural network (CNN) on a Raspberry Pi 3 [8]. The authors also tested several hardware platforms and compared their performances in supporting CNN-based robotic cars. The hardware was relatively simple - the robotic car used is a Remote Control (RC) car that can drive at only two different speeds, and the potential for future extensions (such as adding sensors, etc.) is also limited. The code is posted on Github, but it has not been updated since June 2019.

DeepPiCar (the author posted it on towarddatascience.com as a series of blogs [2]; in this paper, we call it **DeepPiCar on TDS**) was a newer robotic car project originally published in 2019. The author chose the SunFounder robotic car kit because the producer provides an open source Python API. The deep learning model used in this project is a real-world CNN model developed by Nvidia for self-driving cars and provides a foundation on which the student can build upon by adding new features. However, this project has not been updated since 2019, and many of the frameworks and libraries used in this project (TensorFlow, OpenCV, etc.) have since been updated, causing several compatibility issues which are not addressed.

PiDrone is an autonomous drone project created at Brown University in 2018 [9], and it is utilized as a core tool in an undergraduate introduction to robotics course. The project puts extra emphasis on real-world applications of autonomous drones, aiming to educate students on the use-cases and potential societal impact of UAVs. The project itself involves creating an autonomous drone that can localize and maintain position, utilizing Python and Robotics Operating System (ROS) on a Raspberry Pi 3. The course provides an introduction to robotics, giving students exposure to ROS and controllers. Given how focused this course is on robotics, it may not be as appropriate for introducing K-12 students into STEM, as the learned knowledge via this course is not widely applicable on domains outside of robotics. The course is still offered at Brown University as of April 2021; however, the updated coursework

This project is partially supported by Landfall Foundation at Wilmington, NC.

is unavailable for students not in attendance at Brown University.

The Sustainable City Robotic project was developed in 2020 by researchers from Spain [10]. The project was designed to introduce the concept of "sustainable cities" to primary school students, using robotics as a learning tool. The students are tasked with building a miniature sustainable city, incorporating the main concepts of sustainable cities taught throughout the project. The robotic element includes a line-following robot that the students program using a drag and drop, code-block interface called Bitbloq. This project is a good demonstration of the breadth of subject domains to which educational robotics can be incorporated into the curriculum. By adding an interactive element to the lesson, the researchers were able to introduce an important and complex topic to children aged 10-11. While this project is a good introduction for young children, there is not much room to build on top of and/or extend the project to introduce new and more complex concepts.

The PiBot is an educational robotics tool created in 2019 [11] that targets high-school students and aims to bridge the gap between courses and devices used to introduce young children to robotics and more advanced projects aimed at undergraduate and graduate-level students. PiBot succeeds in its goal of being a low-cost and accessible project, as the hardware components are available for purchase or they can be 3-D printed, adding another educational component to the project. Students can also forgo the hardware all together and operate a 3D PiBot model on simulation software, Gazebo. The authors have also developed exercises that are available online that introduce students to the different capabilities of the PiBot. However, it appears much of the code required for this project is no longer available, as we received a 404 error when attempting to open the code repository included in the article.

III. PyRoboCar Project

This project and paper is based on the earlier work by David Tian. His blogs [2] were the foundation of our PyRoboCar project, but we have upgraded the hardware and software of his project in 2019 and made it friendlier to users. The PyRoboCar is a practical car that can be assembled for under $300, with options to upgrade accessories for additional costs. TABLE II shows the current pricing for the parts of the PyRoboCar project.

In addition to the hardware listed, there are necessary items and materials that users are assumed to have access to in a standard school computer lab, including: colored tape, a USB keyboard and mouse, a monitor that takes HDMI input, a micro-HDMI to HDMI cable, and a lab computer.

Our PyRoboCar project provides a roadmap for students to build their own physical and self-driving robotic car using deep learning and computer vision from scratch. Students are expected to implement various steps on different levels of computing environments, including robotics (physical), Raspbian (single-board computer), PC, and the Google Colab cloud computing environment. See Figure 1 for the outline of this PyRoboCar project.

The first step in the project is assembling the robotic car, mounting the Raspberry Pi, Tensor Processing Unit (TPU), and camera onto the car, and labeling the lane in a lab space. After that, students need to install the operating system, Raspbian, and the necessary software, Sunfounder PiCar-V's python API that provides the driver code for the robotic car. Next, students can set up remote access to the Raspberry Pi via VNC Viewer, which allows the use of remote file transfer and avoids continuous monitor, keyboard, and mouse connections. Following this initial setup, students can run simple code to test the functionality of the robotic car (e.g., move forward/backward, make turns) and camera. This process likely necessitates students return to the robotics level to make the necessary adjustments to the hardware before moving to the next step.

The second step in the project is to make PyRoboCar see and think using computer vision and deep learning software. Students should install OpenCV, Matplotlib, and Numpy. Students are then able to test the video processing by running code to see live video window(s) on PyRobocar. Next, students must install TensorFlow for the CPU and the TPU-Processor. This installation step also includes the installation of Keras, a python deep learning API. Students are then able to test the object detection capabilities on objects in the lab (Figure 2).

The third step in the project is lane-keeping. First, students must test the OpenCV. To do this, the video-recording script must be downloaded, which allows students to record videos and save them to PyRoboCar's file system. The student's first video should be recorded following the labeled lane in the lab space. This video will be used in the PyRoboCar code for training. When running the computer vision code, the program will emerge in multiple windows. These windows will show the

TABLE I. Information of Recent Educaitonal Robotics

Project name	Hardware	Cost	Programming language	Other AI/ML components	Target users
DeepPicar by KU and IUB (2018)	Raspberry Pi 3, Pololu motor driver kit, RC car	$70	Python	CNN, computer vision	N/A
DeepPiCar on TDS (2019)	Raspberry Pi 3, SunFounder robotic car, Google Edge TPU, camera	$323	Python	CNN, computer vision, TensorFlow	High school and college students
PiDrone by Brown University(2018)	Raspberry Pi 3, camera, motors, Skyline32 IR sensor, ESC's, frame/hardware	$220	Python	Computer vision (OpenCV)	Undergraduate college students
Sustainable City Robotic Project (2020)	BQ PrintBot, plastic board, ZUM robotics kit	€180 (approx $217.49)	Bitbloq	N/A	Primary School (10-11 years old)
PiBot (2018)	Raspberry Pi 3, infrared sensors, Raspberry PiCamera, ultrasonic	< €180	Python	Computer Vision (OpenCV)	High school students

TABLE II. COST OF PYROBOCAR

Component	Cost
CanaKit Raspberry Pi 4 4GB	$55
64 GB micro SD Card	$11
SunFounder Smart Video Car Kit V2.0 PiCar-V Robot Kit	$100
Four 18650 batteries and one charger	$20
Google Edge TPU USB Accelerator	$60
Fisheye USB camera	$42
Total (before tax)	$288

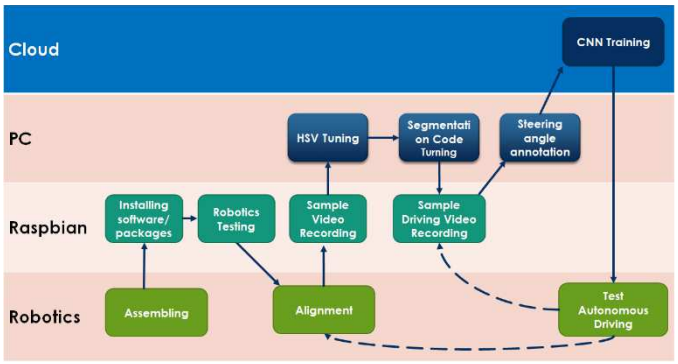

Fig. 1. General project flow of PyRoboCar project.

steering angle, projected steering lane line overlays, and the edges of the color mask. If the steering angles are too sporadic, then the students must go into the PyRoboCar code and adjust the HSV (Hue Saturation Value) values to allow for better detection of lane colors.

In step four of the project, students train the PyRoboCar, create the training data, create the model with the cloud-based CNN deep learning model, and test the performance with the PyRoboCar. To produce the training data, students record videos of the car being manually controlled driving down the labeled lane in the lab space. It is recommended to record at least ten videos with varying light conditions, traveling in different directions, and turning the car back and forth inside the labeled lane while moving down its track. Increasing the amount of training data is associated with better performance. Students test the training data on the code to ensure that the car is reading the data correctly and the tuned computer vision code runs well. Once the training videos are ready, students convert the videos to images that will be fed to the CNN cloud-based model. Further, students must transfer the images to google drive and train the model on the cloud. Once the model has been created, students can then move the model to the PyRoboCar and test the model. At this point, PyRoboCar should be able to use the CNN model and autonomously follow the lanes by itself.

Step five is to review the model's performance. It is possible that the model does not perform as well as expected when initially running. If so, then students must try to diagnose the problem. Firstly, students should check the alignment of the hardware, including the wheels and camera. If the problem is not the hardware, students must improve the training dataset and then retrain the model. Steps four and five should be iterated until the car is running to the optimal level.

IV. PROJECT OUTCOME

In this section, we present the major milestones of the PyRoboCar project and the outcome of each milestone.

A. Robotics testing and computer vision testing

After students assemble the robotic car and install the driver code, they can test the car using some basic scripts in python by calling the pre-defined functions. For example, making the car move forward for one second, turn the wheels to the right side by 45 degrees, etc. This testing can be done without needing to create the lanes on the floor. This step is also crucial for the later hardware alignment process in case there are mistakes in the hardware assembling process.

Another test that can be done in the early phase is computer vision testing. To test the camera and the OpenCV installed on the Raspberry Pi, students can do live object detection using the code from the COCO project [12]. The object detection is not a component that we use in the PyRoboCar project right now, but this is an interesting extension we will implement. In the computer vision tests shown in Figure 2, students can check the installation and the camera, and get some rough idea about 1) the computation power of the Raspberry Pi (10 FPS) and 2) the vision of the fisheye camera (barrel distortion is presented).

Fig. 2. Computer vision testing.

B. OpenCV tuning

In the project instructions of the PyRoboCar, we recommend students to record several short videos (like 5-second long) from the camera when the car is placed in the middle of a straight lane, facing forward in both bright and dark light conditions. Students can use this type of videos to "pick" the HSV color scale that helps the OpenCV separate the lanes from the rest of each frame. The range of the HSV scale for the lane can cause failure in lane segmentation (if the range is too narrow) or too much noise in the lane segmentation (if the range is too large).

The second part of the OpenCV tuning is on the parameters of the lane segmentation. There are three parameters (minThreshold, minLineLength, and maxLineGap) to be tuned to make the car able to detect the lanes from each frame.

Students can use the short videos to test the parameters so that they can produce the lane segments and the current direction of the car from each frame. Figure 3 is an example of the output of the tuned OpenCV code.

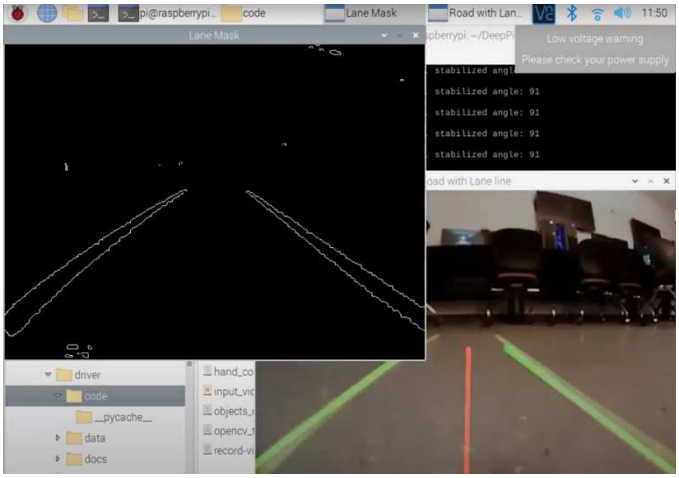

Fig. 3. A screenshot of the tuned OpenCV code. The left window shows the lane segments as polygons, and the right window shows the lane segments as straight lines (green lines), and the facing angle of the car (red line).

C. *Model Training Process*

One of the most significant steps of the project was being able to build a working model to read lane lines and predict an accurate angle for the car to drive. The process of developing a working model involves students in recording a large number of videos, then slicing up each video into individual frames. Next, students need to import the folder of frames to their Google drive to train a CNN model on Google Colab.

The CNN model will take each frame, and pick out the individual pixels that fit in the parameters students set from their chosen HSV values and create the road lines. Then, the program will take the angle of the lanes and average the two to create a steering angle, as shown in Figure 3. After the students complete the model, the program will give some summary statistics. The student should look over the summary to make sure that the model is accurate before downloading the model. We recommend that students choose an area that does not give a lot of background noise and is not overly reflective and a lane color that is not used around the lab to reduce potential problems.

D. *Testing Autonomous driving*

After the students create a successful model, the next step is to test it on the lane (track). Students will need to download their model in a '.h5' format from their Google drive and import it to the PyRoboCar. The students may also need to create a new lane (track) to test the CNN model.

We encourage the students not to be discouraged if the first model did not perform well. This process often takes building and testing several models before finding one that works. Being persistent and continuing to problem solve is the key to get the model to be successful. Figure 4 shows the car running autonomously.

Fig. 4. The robotic car driving by itself on a test lane.

V. PyRoboCar as an Open Repository

As we reviewed the related work, most of the educational robotic projects do not have detailed instructions teaching future students/researchers to reproduce the robotic projects. Though some of the code is available online, building and coding a robot requires tremendous effort beyond the coding part. Robotics-related projects have unique challenges, including tuning the hardware, designing the experiment spaces, debugging the hardware issues, etc. Therefore, other than research papers, there should be additional step-by-step instructions, including the tricks and tips learned by the authors.

Another issue with most of the recent robotic projects is that the authors didn't make use of the existing open-source hosting systems or the project management tools well; thereby, the projects are not properly maintained and supported. Most of the projects have pending questions unaddressed, and the code are likely to become outdated after a major update of the framework used (such as TensorFlow). In the DeepPiCar project, we decided to make the source code and the instructions both available on GitHub. We plan to make this project an extensible open-source educational robotics project. The research team will continue expanding this project and add new content to the Github repository[1] (both instructions and code), and we also welcome collaborators to interact with our team or contribute their code. The extensibility of this project comes from 1) the open-source driver code, 2) the frameworks used (OpenCV, TensorFlow, etc.), and the computational power (Raspberry Pi and TPU).

In our instructions[2], in addition to our code, we also include the photos/screenshots/videos to help readers to follow the process step-by-step. We have also provided the tips and tricks we have learned from this project.

VI. Conclusion

Robotics-based projects can help students develop scientific skills [13]. While accessible educational robotic projects do exist, most of them do not provide a comprehensive roadmap to guide students, nor do they offer high enough extensibility.

In this paper, we present the PyRoboCar project as an end-to-end educational robotics project, from hardware assembly to deploying a deep learning model for autonomous driving. We open our source code and our instructions online, and the extensibility inherited from the framework and the make of this car offers great potential for us to continue expanding this project. Opening our work as an open-source project, we also welcome contributions from the community.

1 https://github.com/larguncw/PyRoboCar
2 https://larguncw.github.io/PyRoboCar/

REFERENCES

[1] W. J. Church, T. Ford, N. Perova, and C. Rogers, "Physics with robotics—using LEGO MINDSTORMS in high school education," 2010.

[2] D. Tian, "DeepPiCar — Part 1: How to Build a Deep Learning, Self Driving Robotic Car on a Shoestring Budget," *Medium*, May 08, 2019. https://towardsdatascience.com/deeppicar-part-1-102e03c83f2c (accessed May 12, 2021).

[3] H. Eck, S. Hirschmugl-Gaisch, A. Hofmann, M. Kandlhofer, S. Rubenzer, and G. Steinbauer, "Innovative concepts in educational robotics: Robotics projects for kindergartens in Austria," presented at the Austrian Robotics Workshop, 2013. Accessed: Apr. 21, 2021. [Online]. Available: https://graz.pure.elsevier.com/en/publications/innovative-concepts-in-educational-robotics-robotics-projects-for

[4] S. Hussain, J. Lindh, and G. Shukur, "The effect of LEGO Training on Pupils' School Performance in Mathematics, Problem Solving Ability and Attitude: Swedish Data," *J. Educ. Technol. Soc.*, vol. 9, no. 3, pp. 182–194, 2006.

[5] P. Janka, "Using a Programmable Toy at Preschool Age: Why and How?," p. 10, 2008.

[6] F. Bennie, C. Corbett, and A. Palo, "Building Bridges, Robots, and High Expectations.," *Odyssey New Dir. Deaf Educ.*, vol. 16, pp. 14–19, 2015.

[7] K. Highfield, "Robotic toys as a catalyst for mathematical problem solving," 2010.

[8] M. G. Bechtel, E. McEllhiney, M. Kim, and H. Yun, "Deeppicar: A low-cost deep neural network-based autonomous car," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 11–21.

[9] I. Brand, J. Roy, A. Ray, J. Oberlin, and S. Oberlix, "Pidrone: An autonomous educational drone using raspberry pi and python," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–7.

[10] F. R. Vicente, A. Z. Llinares, and N. M. Sánchez, "'Sustainable City': A Steam Project Using Robotics to Bring the City of the Future to Primary Education Students," *Sustainability*, vol. 12, no. 22, pp. 1–21, 2020.

[11] J. Vega and J. M. Cañas, "PiBot: An open low-cost robotic platform with camera for STEM education," *Electronics*, vol. 7, no. 12, p. 430, 2018.

[12] "COCO - Common Objects in Context." https://cocodataset.org/#home (accessed May 14, 2021).

[13] S. Turan and F. Aydoğdu, "Effect of coding and robotic education on pre-school children's skills of scientific process," *Educ. Inf. Technol.*, vol. 25, no. 5, pp. 4353–4363, 2020.