Lane Detection using Canny Edge Detection Algorithm for Real-time Racing Game

Sehar Shahzad Farooq[1][0000−0002−2571−9121], Hameedur Rahman[2][0000−0001−8892−9911], Samiya Abdul Wahid[2][0000−0003−4322−4736], Iftikhar Ahmad[1], Jin Ho Lee[1], and Soon Ki Jung*[1][0000−0003−0239−6785]

[1] School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea

[2] Department of Computer Games Development, Faculty of Computing and AI, Air University, Islamabad 44000, Pakistan

{sehar146, skjung}@knu.ac.kr

Abstract. Self-driving cars are not only a current reality, but this technology also gives us a glimpse of what we can expect from complex technology in the future. Many experts from various fields have come together to create the best autonomous vehicles possible. Lane detection is now a common feature in vehicles and requires large amounts of real-life data for training models. To create fully autonomous vehicles, this data must consider all potential weather conditions, road geographies, driver actions, and vulnerable road users. Most companies do not have the financial resources to test their cars over long distances, and it is impractical to recreate every potential scenario in the real world. One solution to train models more efficiently is to use car racing games, which offer realistic driving scenarios and can help avoid real-world security and privacy concerns. The proposed system is to develop a system for detecting road lanes in a car racing game using Python, and OpenCV, with the ultimate goal of using this system to generate data for training models in self-driving vehicles. The system will use Canny edge detection to detect the edges of the lane, along with a masking function that obscures unwanted details in images, such as trees, rocks, and power lines, will be used to allow the model to concentrate on identifying lanes. The Hough Transform is used to identify and draw the lanes in the game. The proposed system accurately detects lanes in the real-time racing car. This approach allows for the generation of a large volume of data for lane detection system in autonomous cars in a cost-effective and efficient manner, while also providing a range of driving scenarios that may be difficult or impractical to replicate in the real world.

Keywords: Lane detection · Canny edge detector algorithm · Car racing game.

1 Introduction

Human beings make numerous choices each day, and these decisions are often based on the sensory information we receive from our surroundings. Most of

this perception is visual when it comes to driving. Autonomous vehicles, also known as self-driving cars, are designed to be able to detect objects in their surroundings and make timely decisions to respond to these objects. This leads to several challenges related to computer vision in autonomous vehicles, such as detecting road signs, traffic signals, pedestrians, other vehicles, and lanes [10]. This paper focuses specifically on lane detection, which is a crucial aspect of planning the movement of a vehicle.

In this research, we propose a lane detection system using the Canny edge detection algorithm for lane detection in a real-time car racing game called Asphalt 8: Airborne. The screen of the game is accessed using the ImageGrab module of the Python Imaging Library (PIL). Canny edge detection, a widely used method in computational vision processing, is applied using OpenCV to detect the edges of the lanes, and a masking function was applied to eliminate distractions such as trees, rocks, and power lines [25]. The Hough Transform was utilized to mark and draw the lanes in the game [6]. The game environment includes realistic physics, graphics, and a variety of environments such as sharp turns, slopes, and different weather conditions which are useful in lane detection system.

The following objectives are intended to achieve through this research:

1. To utilize computer vision algorithm i.e., canny edge detection algorithm for detecting lane edges in a real-time car racing game (Asphalt 8: Airborne).
2. To develop a system which would detect lanes in racing games and develop large volume of datasets for building lane detection system in self-driving cars.
3. To contribute to minimizing the time complexity in terms of collecting real-life dataset for developing lane detection system in self-driving cars.

One of the main issues in lane detection in self-driving cars is finding enough real world data to feed into the powerful machine learning algorithms that are used to perform tasks such as lane detection [15]. Practical applications of self-driving vehicles require a large amount of data to be collected and labeled in order to train the algorithms. It is time-consuming and costly to collect and label real-world data, and it is also not feasible to test every potential scenario, such as crashing a car at high speed into a brick wall, in real life [7]. A lot of work is being done on real-life captured data of cars for building a lane detection system in self-driving cars, however, very less work has been done on lane detection in car racing games for producing large datasets which are realistic, less time-consuming and cost-effective. A lot of work can be done on detecting lanes in car racing games to produce large amounts of training datasets for building effective and efficient self-driving cars.

2 Related Work

Utilizing synthetic data for development of self-driving cars is becoming popular and is gaining a lot of attention of researchers [13]. As large amounts of real-life

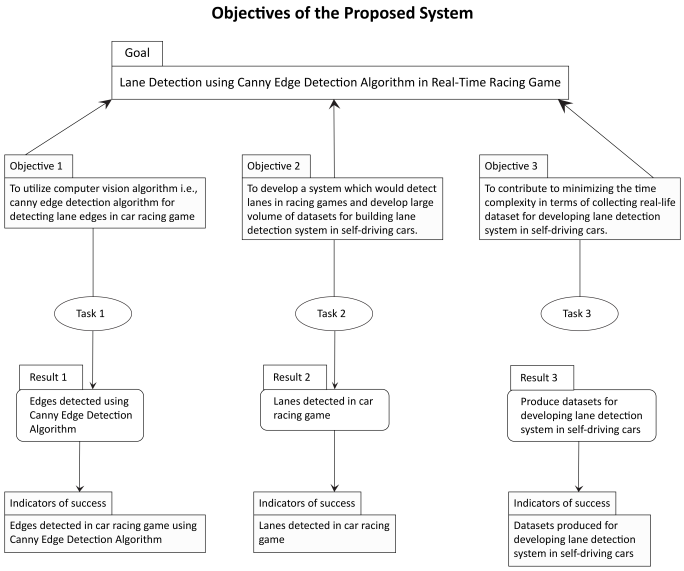

Fig. 1. Research Objective's Breakdown Structure (RBS) of our study

data for training models to create fully autonomous vehicles, this data must also consider all potential weather conditions, road geographies, driver actions, and vulnerable road users. The visual landscapes in numerous video games are so highly detailed and realistic that they can provide data that is just as accurate as real-life images [8]. Readily available computer games that feature realistic graphics and hours of gameplay can offer a more convenient way to collect large amounts of training data. A group of researchers from Intel Labs and Darmstadt University in Germany have devised an innovative method for obtaining valuable training data from the popular video game Grand Theft Auto [18]. The researchers developed a software layer that sits between the game and a computer's hardware, automatically categorizing various objects in the road scenes depicted in the game. This allows for the labeling of data to be fed into a machine learning algorithm, enabling it to identify cars, pedestrians, and other objects shown in the game or in real life. Research conducted by PhD students at the University of British Columbia demonstrates that video games can be utilized to train a computer vision system, sometimes just as effectively as real data [21]. Another research showed that video games also offer a simple way to diversify the environmental conditions present in training data. A team at Johns Hopkins University in Baltimore created a tool that can link a machine learning algorithm to any environment created using the popular game engine Unreal. This includes games like KiteRunner and Hellblade, as well as numerous impressive architectural visualizations [17]. In another research study, asynchronous learn-

ing technique was used to train an end-to-end agent for a realistic car racing game called World Rally Championship 6 (WRC6) [16]. The system does not depend on the game's score, but the agent is trained using only images and speed to determine the best actions, while taking into account real driving conditions. Moreover, in 2013 researchers created and improved a controller for a car in the TORCS open-source racing game. They added advanced collision detection, overtaking strategies, real-time evaluation, and a thorough analysis of the track using the concept of multi-agent artificial potential fields (MAPFs) [19]. In a project, researchers developed a computer program named HORCNN Network that can identify the lanes on a road, detect vehicles and buildings, and control the vehicle's direction based on these inputs, using individual frames from a video game as input [12]. Researchers have also built a system where they present a fuzzy logic-based self-driving car control system and demonstrate its use in the JavaScript Racer game. To achieve this goal, the frameworks of control theory, fuzzy logic, and computer vision were combined in the proposed intelligent driving system [9]. The above research papers and several other papers including [14, 3, 22, 5, 1, 24, 23, 4, 2] show that racing games can be used as an alternative for developing a lane detection system in self-driving cars. However, not a lot of work has been done in this aspect and as result the proposed system detects lanes in a car racing game (Asphalt 8: Airborne) using computer vision algorithms I.e., canny edge detection, Hough transform and a filtering mask.

3 Methodology

In the present system we proposed a lane detection system in a real-time car racing game named Asphalt 8: Airborne. The approach adopted in for this system is explained in the following subsection of Lane detection system.

3.1 Lane Detection System

The purpose of the lane detection system is to detect lanes in a real-time car racing game (Asphalt 8: Airborne) to provide synthetic dataset for autonomous cars to make better decisions while driving. For an instance, autonomous cars can be trained on this synthetic dataset for detecting lanes in an efficient and timely manner. As discussed in Section 1 will access the real-time screen of the game using ImageGrab module of Python Imaging Library (PIL) for detecting the lanes in game. It will then use Canny edge detection to detect the edges of the lane, along with a masking function that obscures unwanted details in images, such as trees, rocks, and power lines, and will be used to allow the model to concentrate on identifying lanes. The Hough Transform is used to identify and draw the lanes on the frame in python output window. The following section of Experimental Results gives a step by step details about the approach of the lane detection system. The main flow process of the system is demonstrated in Figure 2.

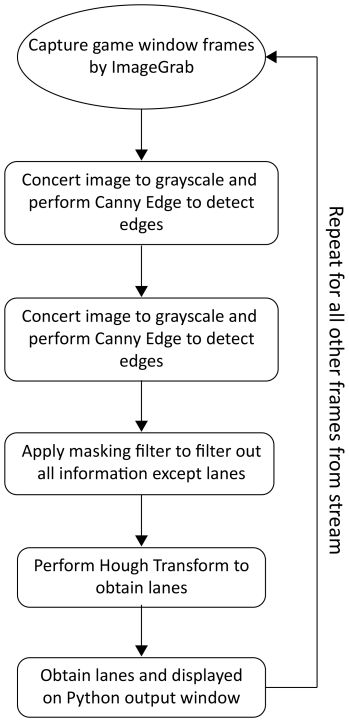

Fig. 2. Process flow chart of detecting lanes and displaying on Python output window

Figure 3 given below displays the layer by layer pipeline of detecting and displaying the lanes.

3.2 Selection of Asphalt game

We chose "Asphalt" as the game for our research because it provides a high level of realism and a range of driving scenarios that can be used to test and train the lane detection system. We believe that the realism and variety of driving scenarios provided by "Asphalt" make it an ideal platform for generating data for autonomous vehicle research.

3.3 Restricted region of interest (ROI)

The reason we chose to use a restricted ROI was to minimize the impact of other elements in the game, such as trees and rocks, on the lane detection system. However, we understand about the potential impact of the restricted ROI on the accuracy of the system, and we plan to further evaluate the impact of this choice in further experiments. we also evaluate the lane detection system under

noisy and varied situations. We plan to perform experiments in different weather conditions and with various road geographies to better understand the robustness and reliability of the system. We will also compare the results with other lane detection methods to provide a comprehensive evaluation of the system.

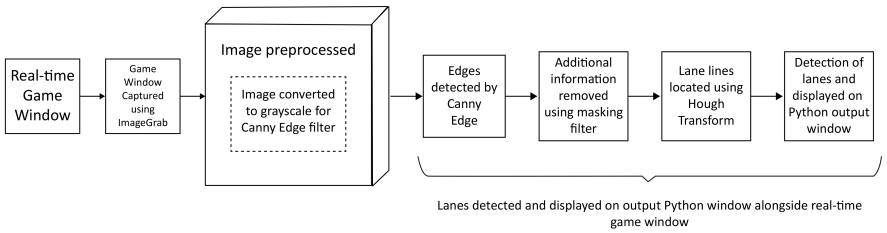

Fig. 3. Layer by layer pipline of detecting and displaying the lanes

4 Experimental Results

4.1 Capturing the Game Screen in Real-Time

The first step in the developed system is performed by accessing and capturing the racing game screen in real-time. This is done using the ImageGrab module of PIL library [20]. The game screen is captured in real time and is illustrated in Figure 4. The image on the left is real-time game window and the image on the right shows the python window which has captured the game screen using ImageGrab module.

Fig. 4. Real-time game window running alongside game screen capture in python window

4.2 Image Processing for Edge Processing Using Canny Edge Detection Algorithm

We used a technique called Canny edge detection, which is implemented in the OpenCV library, to extract and retain useful structural information from images while significantly reducing their size [26]. The image is first first converted to grayscale and then Canny edge filter is applied to obtain the edges. This process simplifies the image significantly and we obtained the edges of the image on the screen. Edges are detected which is illustrated in the Figure 5 given below.

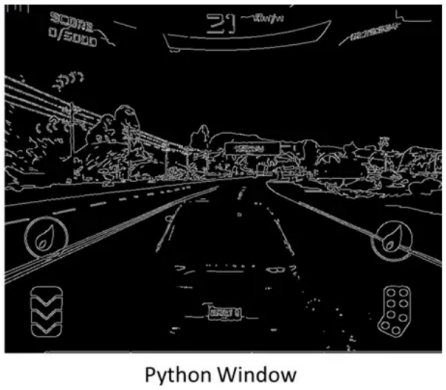

Fig. 5. Edges detected using Canny edge detector

4.3 Applying Mask Filter to Filter all the Additional Information from the Image Other Than the Lanes

For applying masking filter, we defined a masking function. The purpose of this function was to create a polygon shape that hid all the information from the image except for the region that contains the lanes. The resulting mask allowed the algorithm to focus on the lanes in the image. The additional information which is removed from the image is the area other than the red marked area as illustrated in the Figure 6 given below.

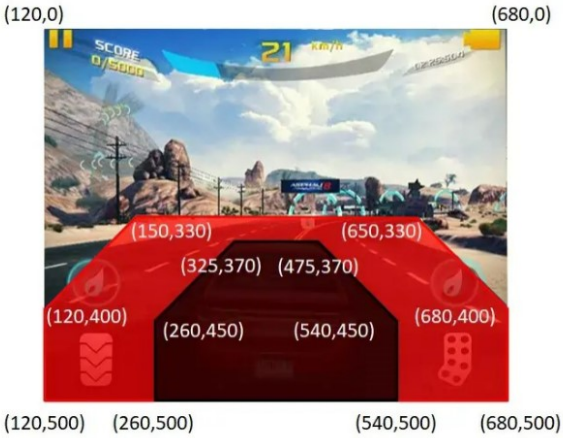

Fig. 6. Masking all the area other than area marked with red color.

The result of the masking filter removed all the additional information other than from the image except for the lanes which are illustrated in Figure 7 given below.

Fig. 7. All additional information filtered out using masking

4.4 Applying Hough Transform to Locate Lane Lines

To locate the actual lane lines in the image, we applied Hough Transform implemented in OpenCV to the edge-processed image [11]. To use this effectively, it is necessary to blur the image slightly before applying the Hough Transform. We used the Gaussian Blur function from OpenCV for this purpose. The lane lines

are located and displayed on the screen which is demonstrated in the Figure 8 shown below.

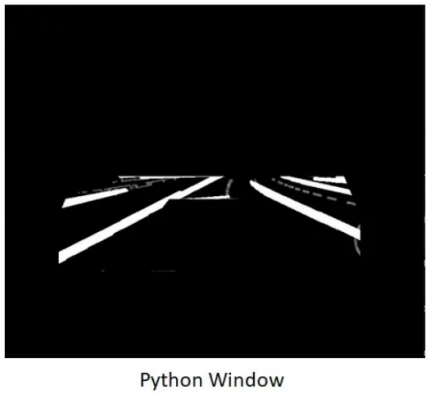

Fig. 8. Hough Transform implemented for overlaying lane lines

4.5 Detecting and Drawing Lanes in Python Output Window

For obtaining the lanes and drawing them in the output window, we created a function. The desired output was obtained which detected the lanes in the output python window. The lanes are detected when game console is running in real-time. The lane detection is shown simultaneously in a python window as shown in the Figure 9 given below.

Fig. 9. Lanes detected in python window in real-time alongside the original game window running

The results of the proposed system are satisfactory as it detected the lanes in the car racing game (Asphalt 8: Airborne). We applied Canny edge detection algorithm for detecting the edges in the captured screen of the game. Canny

edge successfully detected the edges in the captured screen of the game. After the edge detection masking filter was applied for filtering out all the additional information other than the lane edges to focus on the lanes in the image. The Hough Transform located the actual lane lines in the image successfully, which was implemented in OpenCV following the Gaussian blur function. The location of the actual lane line was followed by the successful detecting and drawing of lanes in the python output window alongside the game console window running in real-time.

5 Conclusion

The proposed system successfully detected lanes in a real-time car racing game named Asphalt 8: Airborne with the help of computer vision algorithm i.e., Canny Edge Detection. The screen of the game was accessed using the ImageGrab module of the Python Imaging Library (PIL). Canny edge detection was applied using OpenCV to detect the edges of the lanes, and a masking function will be applied to eliminate distractions such as trees, rocks, and power lines. The Hough Transform was utilized to mark and draw the lanes in the game. The prosposed system can be successfully deployed in generating synthetic dataset for training artifical intelligence models for the development of effective and efficient self-driving cars. It can also provide a wide range of driving scenarios that may be difficult or impractical to replicate in the real world. The generated data can be used for training models in various fields, including computer vision, machine learning, and deep learning. In future, several other training algorithms can be implemented for training autonomous vehicle lane detection by using the data generated using the proposed method. It can also be used as a benchmark study in near future for detecting lanes as well as other objects in real-time. Further studies can be done on the above mentioned idea to improve the quality of the model as well as data collection using diverse games.

6 Acknowledgement

This study was supported by the BK21 FOUR project (AI-driven Convergence Software Education Research Program) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (4199990214394).

References

1. Farooq, S.S., Aziz, A., Mukhtar, H., Fiaz, M., Baek, K.Y., Choi, N., Yun, S.B., Kim, K.J., Jung, S.K.: Multi-modality based affective video summarization for game players. In: International Workshop on Frontiers of Computer Vision. pp. 59–69. Springer (2021)

2. Farooq, S.S., Baek, J.W., Kim, K.: Interpreting behaviors of mobile game players from in-game data and context logs. In: 2015 IEEE Conference on Computational Intelligence and Games (CIG). pp. 548–549. IEEE (2015)
3. Farooq, S.S., Fiaz, M., Kim, K., Jung, S.K.: Experience modeling for candy crush game player using physiological data by means of homogeneous transfer learning
4. Farooq, S.S., Rahman, H., Raza, S.A.N., Raees, M., Jung, S.K.: Designing gamified application: An effective integration of augmented reality to support learning. IEEE Access **10**, 121385–121394 (2022)
5. Farooq, S., Kim, K.: Game player modeling, encyclopedia of computer graphics and games (2016)
6. Gabrielli, A., Alfonsi, F., Del Corso, F.: Simulated hough transform model optimized for straight-line recognition using frontier fpga devices. Electronics **11**(4), 517 (2022)
7. Givan, D., Hanshaw, N., Choi, H.: Application of lane navigation and object detection in a deep-learning self-driving car. In: Future of Information and Communication Conference. pp. 906–917. Springer (2022)
8. Knight, W.: Self-driving cars can learn a lot by playing grand theft auto (Sep 2016), https://www.technologyreview.com/2016/09/12/157605/self-driving-cars-can-learn-a-lot-by-playing-grand-theft-auto/
9. Korkmaz, B., Etlik, U.B., Beke, A., Kumbasar, T.: Fuzzy logic based self-driving racing car control system. In: 2018 6th International Conference on Control Engineering & Information Technology (CEIT). pp. 1–6. IEEE (2018)
10. Lee, D.H., Liu, J.L.: End-to-end deep learning of lane detection and path prediction for real-time autonomous driving. Signal, Image and Video Processing pp. 1–7 (2022)
11. Marengoni, M., Stringhini, D.: High level computer vision using opencv. In: 2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials. pp. 11–24. IEEE (2011)
12. Motupalli, C., Mohinth, R., Gaur, S., Mittal, V., Prakash, S.: Supervision of video game car steering implementing horcnn network. In: 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence). pp. 291–294. IEEE (2022)
13. Nowruzi, F.E., Kapoor, P., Kolhatkar, D., Hassanat, F.A., Laganiere, R., Rebut, J.: How much real data do we actually need: Analyzing object detection performance using synthetic and real data. arXiv preprint arXiv:1907.07061 (2019)
14. Oda, O., Lister, L.J., White, S., Feiner, S.: Developing an augmented reality racing game. In: 2nd International Conference on INtelligent TEchnologies for interactive enterTAINment (2010)
15. Pattanashetty, V.B., Mane, V., Hurkadli, S.S., Iyer, N.C., Kore, S.: Lane detection for visual assistance of self-driving vehicles for structured and unstructured roads. In: Information and Communication Technology for Competitive Strategies (ICTCS 2020), pp. 271–279. Springer (2022)
16. Perot, E., Jaritz, M., Toromanoff, M., De Charette, R.: End-to-end driving in a realistic racing game with deep reinforcement learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 3–4 (2017)
17. Qiu, W., Yuille, A.: Unrealcv: Connecting computer vision to unreal engine. In: European Conference on Computer Vision. pp. 909–916. Springer (2016)
18. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European conference on computer vision. pp. 102–118. Springer (2016)

19. Salman, M.: Collision detection and overtaking using artificial potential fields in car racing game torcs using multi-agent based architecture (2013)
20. Sathaye, N.: Python Multimedia. Packt Publishing Ltd (2010)
21. Shafaei, A., Little, J.J., Schmidt, M.: Play and learn: Using video games to train computer vision models. arXiv preprint arXiv:1608.01745 (2016)
22. Shahzad Farooq, S., Fiaz, M., Mehmood, I., Kashif Bashir, A., Nawaz, R., Kim, K., Ki Jung, S.: Multi-modal data analysis based game player experience modeling using lstm-dnn. Computers, Materials and Continua **68**(3), 4087–4108 (2021)
23. Togelius, J., Lucas, S.M., Nardi, R.D.: Computational intelligence in racing games. Advanced Intelligent Paradigms in Computer Games pp. 39–69 (2007)
24. Tognetti, S., Garbarino, M., Bonarini, A., Matteucci, M.: Modeling enjoyment preference from physiological responses in a car racing game. In: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games. pp. 321–328. IEEE (2010)
25. Wu, F., Zhu, C., Xu, J., Bhatt, M.W., Sharma, A.: Research on image text recognition based on canny edge detection algorithm and k-means algorithm. International Journal of System Assurance Engineering and Management **13**(1), 72–80 (2022)
26. Xu, Z., Baojie, X., Guoxin, W.: Canny edge detection based on open cv. In: 2017 13th IEEE international conference on electronic measurement & instruments (ICEMI). pp. 53–56. IEEE (2017)