BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ
AUSUD
JITSA

Research Article

Lane detection for autonomous driving in a video game environment

Ahmet Onur Giray[1,*], Hatice Doğan[2]
[1] Department of Electrical and Electronics Engineering, Faculty of Engineering, Dokuz Eylül University, Izmir, Turkey
[2] Department of Electrical and Electronics Engineering, Faculty of Engineering, Dokuz Eylül University, Izmir, Turkey
*Correspondence: ahmetonurgiray@gmail.com
DOI: 10.51513/jitsa.1200774

Abstract: To ensure comfortable and safe driving, the automotive industry has accelerated the development of autonomous vehicles in recent years. In the design of autonomous vehicles, challenging problems such as lane detection need to be solved. Convolutional neural networks, which show superior performance in many fields, have also been used in the lane detection problem. The datasets required to train CNN models are too large to be collected and labeled by manual effort. In this study, a method is proposed to automatically collect a labeled data set from the video game environment to be used in the detection of highway lanes. Different CNN models such as ResNet50, VGG16, Xception, and InceptionV3 networks are trained using the Transfer Learning method with 745,823 collected images. The images captured by the front vehicle camera are used as input, the coordinates of the points in the left and right lane and the center of the lane in the 2D plane in front of the vehicle and the angle of the vehicle are used as outputs. The performances of these models are tested and compared on the images collected from a road not used in the training set. According to the performance comparisons, ResNet50 performs best.

Keywords: convolutional neural networks, autonomous driving, road lane detection

Video oyunu ortamında otonom sürüş için şerit tespiti

Özet: Konforlu ve güvenli sürüşü sağlamak için otomotiv sektörü son yıllarda otonom araçların gelişimini hızlandırmıştır. Otonom araçların tasarımında şerit tespiti gibi zorlu problemlerin çözülmesi gerekmektedir. Birçok alanda üstün performans gösteren evrişimli sinir ağları şerit tespit problemlerinde de kullanılmıştır. CNN modellerini eğitmek için gerekli olan veri setleri, manuel çaba ile toplanıp etiketlenemeyecek kadar büyüktür. Bu çalışmada, otoyol şeritlerinin tespitinde kullanılacak etiketli bir veri setinin video oyunu ortamından otomatik olarak toplanması için bir yöntem önerilmiştir. ResNet50, VGG16, Xception ve InceptionV3 ağları gibi farklı CNN modelleri, toplanan 745,823 görsel ile Transfer Öğrenme yöntemi kullanılarak eğitilmiştir. Araç ön kamerası tarafından yakalanan görüntüler girdi olarak kullanılmış, aracın yol merkezine olan açısı ile birlikte aracın önündeki iki boyutlu düzlemde bulunan sol, sağ ve merkez şerit koordinatları çıktı olarak kullanılmıştır. Bu modellerin performansları eğitim setinde kullanılmayan bir otoyoldan toplanan görüntüler üzerinde test edilerek karşılaştırılmıştır. Performans karşılaştırmalarına göre en iyi performansı ResNet50 modeli vermektedir.

Anahtar Kelimeler: evrişimli sinir ağları, otonom sürüş, yol şerit tespiti

ORCID: 0000-0003-4774-3349; 0000-0003-0420-592X (in hierarchical order)
Received 07.11.2022; accepted 13.08.2023
Peer review under responsibility of Bandirma Onyedi Eylul University.

1. Introduction

According to the World Health Organization (2018), 1.35 million people die in traffic accidents annually, and related studies show that human factors cause 97.5% of accidents (Pakgohar et al., 2011). In this regard, autonomous vehicles are expected to eliminate the human factor and significantly reduce the number of casualties. The automotive industry is continuously moving towards autonomous vehicles as more and more companies invest in the development of self-driving vehicles. Lane detection is one of the many challenges that the industry and researchers are trying to solve using various data collected by sensors such as RADAR, LIDAR, and high-resolution cameras. Developing, testing, and deploying lane detection in real-world scenarios on public roads is costly, time-consuming, and can be dangerous.

This study aims to provide a cheaper and more efficient alternative to this problem by using a popular video game called Grand Theft Auto V (2022), developed and published by Rockstar Games, which provides high-quality, realistic visuals and traffic environments. GTA V offers a wide variety of scenes with different lighting conditions, day and night cycles, weather conditions, and numerous vehicle and pedestrian models. The game's virtual city, Los Santos, is based on real-life Los Angeles with amazing detail. As a result, the scenery, road types, and traffic patterns are replicas of the real city. Examples of different driving scenarios are shown in Figure 1.

The aforementioned features of video games have led to their use by many researchers. Shafaei et al. (2016) showed that video games can be used to train and evaluate computer vision models and that trained models can be evaluated on real-world data. Using the same video game, Zou et al. (2020) evaluated the generalization performance of a model trained on video game data for remote sensing image segmentation, with promising results.

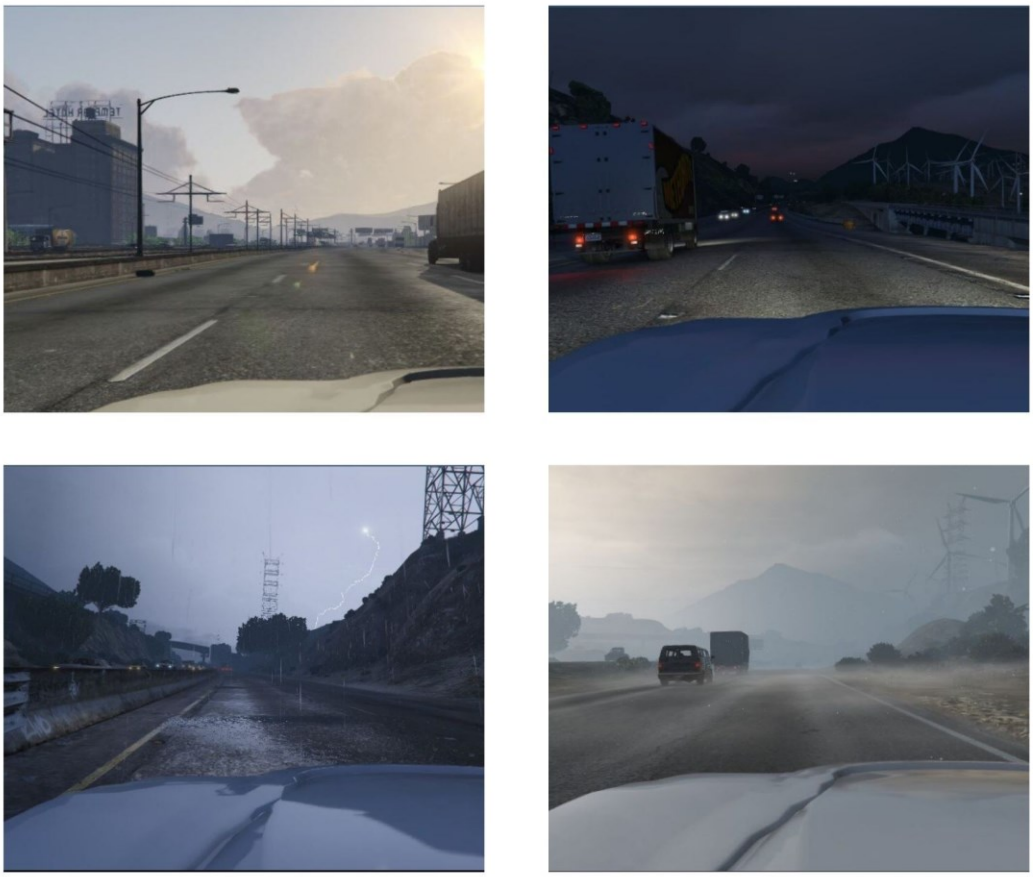

Figure 1. Variety of driving scenarios in the video game.

Filipowicz et al. (2017) presented an AlexNet model to learn stop signs from the dataset generated in GTA V and investigated the success of the developed model on real-world data. It was concluded that it is possible to transfer the models trained in the game to real-world applications. Using GTA V, Filipowicz (2016) demonstrated how to extract video game data, such as vehicle and pedestrian bounding boxes and vehicle distance to lane markings, for model training and the creation of tools to automate the process. Richter et al. (2016) proposed a semantic labeling tool to extract ground truth from GTA V without accessing the source code. Martinez et al. (2017) showed that it is possible to generate datasets from GTA V that include the distances of the agent's car to the lanes and other vehicles. Cai et al. (2021) used the same video game to generate a large-scale 3D human dataset with a variety of subjects, actions and scenarios for 3D human recovery. The study concluded that the game data was surprisingly useful and provided an important complement to real-world data. Novello et al. (2021) also used GTAV in an end-to-end approach, developing a model that converts inputs from a hood camera image and a set of speed values into three driving commands, such as steering angle, accelerator, and brake pressure. In another research, Jaladi et al. (2022) proposed an end-to-end learning and testing framework based on GTAV for autonomous highway driving. The study proposes to use the video game to train a neural network model capable of learning from human driving data. Schöning et al. (2023) used the video game's sophisticated traffic environment to create a simulator for evaluating traffic-based cognitive enhancement technology. Park and Yun (2021) used GTAV to virtualize self-driving algorithms for a digital twining autonomous vehicle, and in their study they have used the video game for object recognition, collision avoidance and lane recognition. Sekkat et al. (2020) used GTAV to generate omnidirectional images for robotics and automotive applications, which is challenging to do in the real world.

This study focuses on the detection of lanes on a highway by a camera placed on the dashboard of a vehicle. The main goal is to create a framework that generates labeled visual data from the video game that can be used to train convolutional neural networks, which have been successfully used in autonomous driving tasks (Muller et al., 2005; Huval et al., 2015). For the lane detection task, classical computer vision methods can be used. The main challenge with these methods is adapting them to different environments, such as different types of road markings, damaged road surfaces, discontinuous road lines, weather and lighting conditions. Classical methods mainly consist of detecting edges or matching them to predefined templates. These methods introduce noise into the extracted lane information (Duong et al., 2014). Using CNNs for lane detection has several advantages over traditional methods. Traditional methods can be prone to error depending on the lighting, brightness, and contrast of the captured images, or when road lanes are partially visible. Deep neural networks can generate interpretable features from time-series data, which can lead to better classification results (Gallitz et al., 2019). In addition, semantic segmentation networks have been widely used in lane detection tasks, and have shown significant improvements in network performance. Although the effectiveness of these networks is not yet high enough to be practically used in real life, they are one of the key methods that will soon be practically used (Shi and Zhao, 2021). Fast and accurate lane detection is critical when using such algorithms in real-world applications. An example of such a network is Z-Net, which is specifically designed for lane detection (Zhang, 2020). Other studies using deep learning for lane detection tasks include LaneNet, end-to-end lane detection using differentiable least squares fitting, and robust lane detection from continuous driving scenes using deep neural networks (Neven et al., 2018; Van Gansbeke et al., 2019; Zou et al., 2019).

Training CNNs requires a large amount of labeled data. The datasets generated by the methods described in this study do not require a manual labeling process, allowing the collection of approximately eight thousand image-label pairs during an hour of driving. Data is generated using third-party tools that leverage source code functionality. Lane labeling is done directly using game engine data and functions, so the data is collected exactly as the game generates it. This increases the accuracy of the dataset.

Using the methods described in this study, a dataset of 745,823 frames was collected over approximately 90 hours of driving. From each frame, the angle of the vehicle, the coordinates of the center point of the lane, and the coordinates of the left and right lane boundaries are extracted.

Driving is also automated using third-party tools, and manual input is only required if the vehicle is to be driven to a predefined point in the world. Otherwise, the vehicle will follow the defined parkour

without the need for any manual input. The collected data is used to train InceptionV3, Xception, VGG16, and ResNet50 models (Szegedy et al, 2016; Chollet, 2017; Simonyan and Zisserman, 2014; He et al, 2016).

This study demonstrates the potential of using a realistic simulation environment for large-scale data collection and training of AI models using transfer learning methods. The dataset generation methods presented do not require manual labeling and are highly accurate. These methods can be used to generate more complex and larger datasets to increase the success rate of AI models under different conditions. The proposed methods can be reused with minimal effort for AI models developed for tasks other than lane detection.

The rest of this paper is organized as follows: In Section 2, the main tools used to extract game data and the CNN models used are presented. The performance comparison of the models is shown in Section 3, the shortcomings of the proposed data extraction methods are discussed in Section 4, and the conclusions are given in Section 5.

2. Materials and methods

While collecting images from a video game is easier, manually labeling lane boundaries on each image is nearly impossible. To streamline the labeling process, the initial idea was to use more traditional and widely used methods such as Canny Edge Detection and Hough Transform (Canny, 1986; Hough, 1962). To partially automate the labeling process, a tool that can modify the parameters of these methods in real-time is used (Gale, 2018). However, visual inspection revealed that under different lighting conditions and when lanes are only partially in the scene, the parameters of these methods need to be readjusted to produce acceptable results. In most complicated traffic scenes, the lanes could not be detected by these methods.

Third-party tools such as Script Hook V and Rage Plugin Hook make it possible to extract information directly from the video game engine (Blade, 2022b; Url1, 2022). The Rage Plugin Hook connects to the game instance and acts as an interface between the game's engine and custom user code. It also provides an in-game menu to facilitate the data collection, such as teleporting the player character to a specific point in the world, changing weather and time conditions, and spawning a vehicle. This menu can also be used to hide the game's HUD (Head-Up Display) for more realistic images. Script Hook allows the use of any .NET language for the development of scripts that run in the game. It provides a huge database containing APIs (Application Programming Interface) of the game engine that have been researched and reverse-engineered by the game's fanbase (Blade, 2022a). Native APIs from the game engine can provide critical information about the location of lanes and their 3D world coordinates within the game. By converting 3D world coordinates to 2D screen coordinates, a huge dataset consisting of images and coordinates of four key points can be collected. These points are:

- The angle of the vehicle to the road
- Left lane boundary
- Right lane boundary
- Center of the lane.

2.1. Extracting game data

The virtual world of GTA V is filled with different types of AI-driven vehicles, including cars, buses, trucks, and motorcycles. Hundreds of such entities can surround the player's character in virtual traffic at any given time. These vehicles behave similarly to real life, they can go slower or faster and they can change lanes. They are controlled by in-game path-finding algorithms, and such functions can be used to understand how the roads are detected for the AI vehicles. The two main pieces of information used are nodes and links. Nodes are individual points scattered throughout the virtual world. They are placed in the middle of roads and connected by entities called links. The links contain information about the type of road (highway, urban, seaway, etc.) and other information such as the number of lanes on the road and the width of each lane. A game file called "paths" holds the information about nodes and links, along with their 3D world coordinates. By parsing this file, one can determine how path-finding information is distributed in the virtual world.

Auxiliary functions such as GET_NTH_CLOSEST_VEHICLE_NODE_ID can be used to determine where the player's character is in the virtual world and how close the character is to a vehicle node or vehicle link. Since nodes and links are labeled with unique IDs, it is possible to determine the properties of the road the player is currently traveling on. The exact positions of the left, right, and center lane points are determined in real time from the road type, lane width, and number of lanes. Using vector calculation APIs such as GET_ANGLE_BETWEEN_2D_VECTORS, even the angle of the vehicle's forward vector to the road center can be calculated. Calculated ground truths can be displayed in the virtual world for debugging and visual inspection.

By default, the video game does not provide a 'dashboard camera'. To provide this view to the player another third-party tool called Hood Camera is used (Url2, 2022). To further speed up the data collection process, another script called Self Drive Plugin is used (Url3, 2022). This script allows placing a waypoint on the game's map, and when the start button is pressed, the car drives to the waypoint at an adjustable speed. The script uses in-game functions similar to the developed dataset collection tool, and the player's vehicle behaves like any other AI-controlled vehicle in the game.

The video game is run at the highest possible graphics settings for more detailed and realistic visuals, but at 800x600 resolution to reduce the size of the dataset.

The dataset collected in this study focused on highway driving in the middle of two lanes on a predefined parkour. Using the developed and above-mentioned complementary scripts and tools, a data collection procedure is defined. After starting the game, a vehicle is spawned using the Rage Plugin Hook tool. The view is switched to the dashboard camera. Both the time and the weather are set to dynamic mode, which means that they change randomly over time. The vehicle is teleported to the starting point of the parkour on the highway. A waypoint is placed at a point on the parkour and the Self Drive Plugin is initiated, each time at a different speed, so that the captured frames are as varied as possible. The developed dataset collection script is initiated from the in-game console provided by the Script Hook V tool. Once the waypoint is reached, another is automatically placed on the parkour, and so on. A folder named 'dataset' is automatically created at the beginning of the process, containing the image files in jpg format and corresponding text files containing the labels.

The flow chart of the developed dataset collection script and an example of its output are shown in Figure 2 and Figure 3, respectively. The functions used in the flowchart are explained in the following.

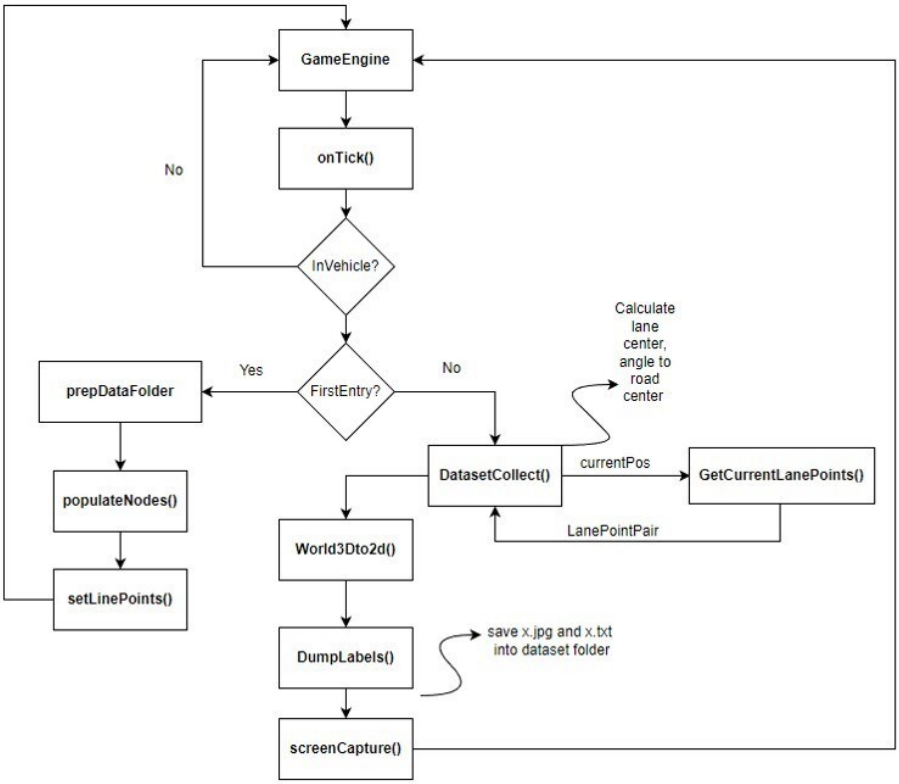

Figure 2. Flowchart of the developed dataset collection script.

- OnTick: A periodic function that is called periodically from the game engine via Script Hook.
- prepDataFolder: Get the last filename in the dataset folder, so that an interrupted process can pick up where it left off.
- populateNodes: Parse the entire "paths" game file for nodes and links to create a database to be used for path detection.
- setLinePoints: Calculate the lane points of links using the width and number of lanes on the road.
- DataSetCollect: Get the current coordinates of the vehicle on the game world, find the closest nodes and links using the database. Also calculates the angle of the vehicle to the road.
- GetCurrentLanePoints: Get lane points for the current position of the car.
- World3Dto2D: Convert 3D world coordinates of lane points to 2D screen coordinates.
- DumpLabels: Save lane coordinates and road angle to a txt file.
- screenCapture: Take a screenshot of the game and saves it as a jpg file.

$Angle = 88{,}24182$

$Center = \{X = 394, Y = 477\}$

$Point0 = \{X = 662, Y = 476\}$

$Point1 = \{X = 114, Y = 478\}$

Figure 3. Example of image-label pair from the dataset. Point0 and Point1 represents the right and left lane boundary coordinates respectively.

2.2. Data analysis

Using in-game entities such as markers, as shown in Figure 4, and the game engine's subtitle or help displayer functions, an analysis of the developed scripts is performed. By manually driving the vehicle in the virtual world, visual inspections are performed for placed markers, and the angle of the vehicle to the road is displayed as a subtitle. When driving straight on the highway, it is seen that the script can generally detect 4 or 5-point pairs (left-right lane boundaries) in front of the vehicle. By increasing the range of nodes and links considered as input to the script, it is possible to determine more pairs of points that belong to lanes. However, this results in incorrect lane detection when the vehicle passes near a connector road or when the node-link density within the vehicle's range is lower. This is because the game has a larger number of nodes and links on the connecting roads to provide more paths for AI-driven vehicles. Since the developed script only considers a predetermined number of nodes and links from the nearest to the farthest, having many nodes and links clustered on the path causes confusion, and makes lane detection difficult. Therefore, it was decided to use only one pair of lane points in front of the vehicle, which is detected correctly in almost every label file.

Figure 4. Road lanes marked in-game.

Since GTA V is not a research tool, debugging and developing the script is a task that must be done using in-game assets such as the markers shown in Figure 4. As a result, the developed scripts have some bugs that cause some nodes-links to be missed or not detected at all, especially when the vehicle

is turning on a highway and the nearest node is not visible on the screen. Sometimes nodes are registered multiple times, which can be caused by the existence of multiple links providing the same lane position information. Depending on the position of the vehicle, the developed script may detect one lane point (left or right boundary) earlier (registered as Point0 in Figure 3) than another one. This can cause confusion during training because Point0 may be a right lane boundary in one label file and a left lane boundary in another.

To reduce the effects of inconsistencies in the training, a label post-processing tool is developed in Python. The tool removes multiple detected points and compares X-axis information within detected points and rearranges left-right lane boundaries so that Point0 output is always the left lane boundary and Point1 is the right lane boundary. If the nearest node is not visible on the screen and therefore not logged in the label file, it is removed from the dataset. Finally, post-processing converts the txt files in the dataset folder into a single csv file, including the ID (which corresponds to the file name), the angle to the road center, the left and right lane points, and the center lane point.

2.3. Model training

Four different models are selected for lane point detection, including ResNet50, VGG16, Xception, and InceptionV3. These models were trained on the ImageNet dataset and the Xception model showed the best Top-5 accuracy. In this study, transfer learning methods are applied to these models to provide a comparison and to select the most suitable model for the lane detection task. To fit the purpose of lane detection, some modifications are made to the models. The input layer is rescaled to 224x224x3 pixels, and a fully connected layer consisting of seven neurons is added as the output layer of the model.

The outputs are, the angle of the vehicle to the road, the coordinates of the center, the coordinates of the left lane boundary, and the coordinates of the right lane boundary. The sigmoid activation function is used at the output layer to scale the values between 0 and 1. The parameters of the last layer are updated during the training phase.

All images are rescaled from 800x600x3 to 224x224x3 and all labels are rescaled with the same ratio before normalization. The angle to the road is divided by 180 degrees for normalization. Therefore, all the output values are between 0 and 1 before training starts.

Training is performed on all models by freezing all base model layers and updating weights on the newly added, fully connected layer on top. It is done using the Adam Optimizer with a preset learning rate of 0.00001 and 60 epochs. Training is performed on the RTX 3080 graphics card in a total of 18646 batches for one epoch. Since the batch size of 32 images does not fit into the 10 GB video RAM of the graphics card, a manual input generator is implemented, which reads the data from the disk and mixes the inputs.

3. Results

Data collected by using the mentioned tools and methods are used to train Xception, InceptionV3, VGG16, and ResNet50 models. The Mean Squared Error is used as the loss function and the results of the first training are shown in Table 1. The validation and training loss graphs are shown in Figure 5. The results showed that the best performing model in terms of low validation loss is ResNet50, so it is selected to be used after a second training for fine tuning.

Table 1. Results of the first trainings.

Model	Training Loss (e-3)	Validation Loss (e-3)	Training Duration
ResNet50	8.7169	8.8481	14h 24m
Xception	9.4252	9.5401	14h 8m
InceptionV3	9.6134	9.7153	12h 26m
VGG16	9.9814	9.932	13h 36m

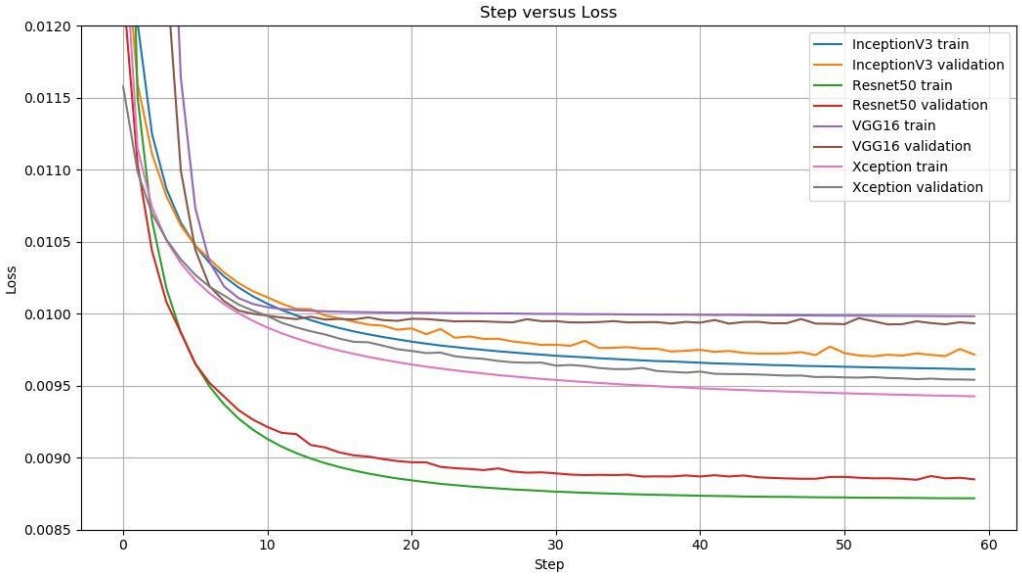

Figure 5. Training and validation losses for all models.

The first 165 layers of the ResNet50 model are frozen and the remaining last block of the base model is joined in the training. Similarly, the Adam Optimizer is used, but this time the learning rate of 0.00001 is decreased in every 10 epochs by multiplying by $(0.85)^{epoch/10}$. The second training is performed for 50 epochs and took approximately 12.5 hours and resulted in a validation loss of 3.4737e-3, which is significantly lower than the first training. Graphs of the training and validation losses of the fine-tuning training are shown in Figure 6.

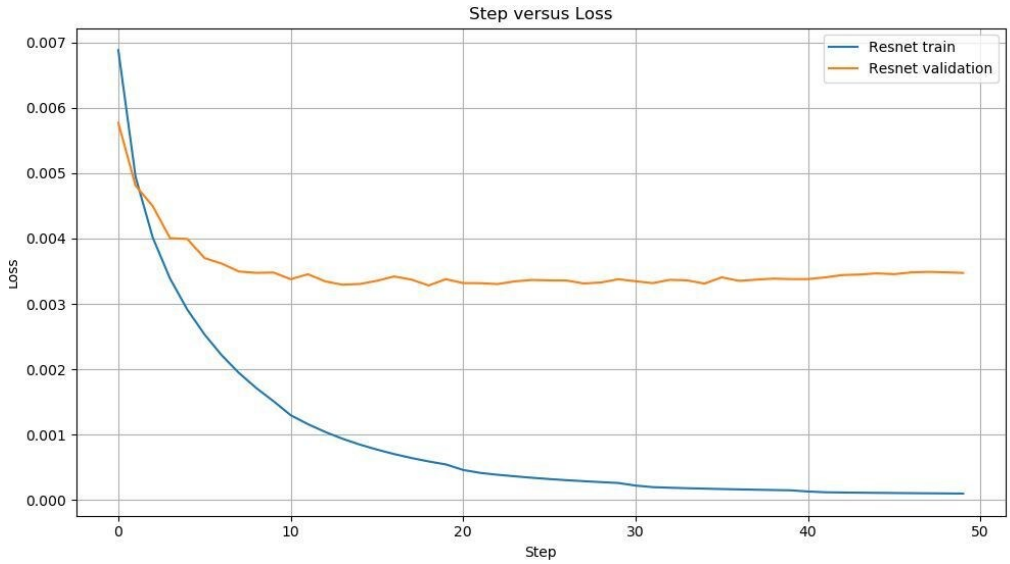

Figure 6. Training and validation losses for ResNet50 fine-tune training.

Using the developed tools, a test dataset consisting of 754 images is collected on a completely different highway than the one used for the training and validation datasets. When the trained model is evaluated on the test dataset, the loss is calculated to be 0.01231.

To visually evaluate the output of the trained network, another tool based on Python's MSS module is created to capture live video from the game window. Sample images of the test set and detected lane points are shown in Figure 7 through Figure 9. The respective ground truth and predicted values of the seven outputs are shown in Table 2, Table 3, and Table 4.

Figure 7. Three points marked by the model.

Table 2. Predicted and actual results for Figure 7.

Parameter	Actual	Predicted
Angle to road	89.85872	91.25470
Center-X	450	466
Center-Y	376	385
Point0-X	331	332
Point0-Y	376	385
Point1-X	569	612
Point1-Y	376	385

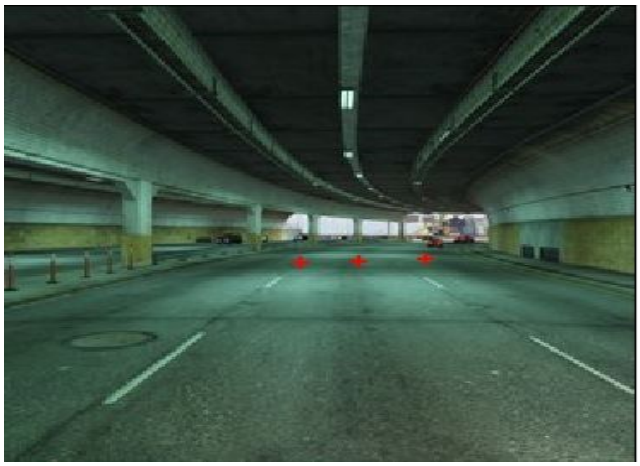

Figure 8. Three points marked by the model.

Table 3. Predicted and actual results for Figure 8.

Parameter	Actual	Predicted
Angle to road	90.26978	88.40917
Center-X	427	490
Center-Y	355	364
Point0-X	340	409
Point0-Y	355	367
Point1-X	514	583
Point1-Y	356	362

The results show that the model can find lane points when driving straight on a highway it hasn't seen before.

Figure 9. Detection of lane boundaries while lanes are partially visible.

Table 4. Predicted and actual results for Figure 9.

Parameter	Actual	Predicted
Angle to road	92.12427	92.27007
Center-X	506	434
Center-Y	449	417
Point0-X	262	247
Point0-Y	449	417
Point1-X	743	637
Point1-Y	449	418

The positive aspect of extracting data from the game is that the lane points can be marked even when a larger vehicle blocks the agent's view, as shown in Figure 9. Such cases are not a problem, because the dataset contains many similar cases where there is traffic and lanes are only partially visible or not visible at all.

Additionally, the trained ResNet50 model is tested on real-world video footage obtained from open source projects (Gautam, 2019; Udacity, 2017). Visual inspection showed similar behavior to in-game footage. The output of the model on two different videos is shown in Figure 10.

Figure 10. Evaluation of the trained ResNet50 model on real world images

4. Discussion

The presented data collection method shows how quickly large amounts of data can be collected for various purposes using the in-game engine and some third-party tools. A drawback of the presented solution is that the developed scripts are difficult to debug while the game is running. This can be overcome by using in-game assets to visualize the information needed for debugging.

The prediction error of lane points when a vehicle is turning on a connection road is higher than when driving straight resulting in an increase in MSE. This is due to the fact that the parkour selected during the training contains only several connection roads that the vehicle needs to turn. Also, because the AI drives the vehicle in an almost perfectly straight line during data collection, the training data is largely dominated by road angles around 90 degrees, making it difficult for the model to learn different angle scenarios.

Model prediction errors can be reduced by using human players to collect data instead of AI scripts. This will increase the randomness in recorded scenes as well as scenarios where the car is not being driven perfectly.

5. Conclusion

The study demonstrates the ability to collect large amounts of data and train artificial intelligence models in realistic simulation environments. Using transfer learning methods, ResNet50, VGG16, Xception and InceptionV3 models, which are typically trained on real-world data, were trained to detect lane points and road angles in a virtual world. ResNet50 was shown to outperform the comparison models for lane and road angle detection tasks on the collected dataset. The training time and computational requirements for transfer learning are significantly less than the training of a network from scratch.

The dataset generation methods presented in this study do not require manual labeling operations and are robust in accuracy since the data is extracted directly from the game engine. Using the methods presented, more complex and larger datasets can be generated to further increase the success rate of the developed model under different conditions.

Researchers' Contribution Rate Statement

The authors' contribution rates in the study are equal.

Acknowledgment and/or disclaimers, if any

This work has been supported by the Dokuz Eylül University Scientific Research Projects Coordination Unit grant 2021.KB.FEN.005

Conflict of Interest Statement, if any

There is no conflict of interest with any institution or person within the scope of the study.

References

Blade, A. (2022a). *GTA V Native DB***.** Retrieved October 28, 2022 from http://www.dev-c.com/nativedb

Blade, A. (2022b). *Script Hook V*. Retrieved October 28, 2022 from https://www.dev-c.com/gtav/scripthookv

Cai, Z., Zhang, M., Ren, J., Wei, C., Ren, D., Lin, Z., Zhao, H., Yang, L., Loy, C. C., & Liu, Z. (2021). Playing for 3D human recovery. *arXiv preprint arXiv:2110.07588.*

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679-698.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

Gautam, D. (2019). *Video file for lane detection project*. Retrieved May, 2023 from https://www.kaggle.com/datasets/dpamgautam/video-file-for-lane-detection-project

Duong, T. H., Chung, S. T., & Cho, S. (2014). Model-based robust lane detection for driver assistance. *Journal of Korea Multimedia Society, 17*(6), 655-670.

Filipowicz, A. (2016). "Driving School II Video Games for Autonomous Driving," M. Eng. thesis, Princeton University, New Jersey, USA.

Filipowicz, A., Liu, J., & Kornhauser, A. (2017). Learning to recognize distance to stop signs using the virtual world of Grand Theft Auto 5. In *Transportation Research Board 96th Annual Meeting*, 17-05456.

Gale, J. W. (2018). *GTA Advanced Lane Finding.* Retrieved October 28, 2022 from https://github.com/Will-J-Gale/GTA-Advanced-Lane-Finding

Gallitz, O., De Candido, O., Botsch, M., & Utschick, W. (2019, October). Interpretable feature generation using deep neural networks and its application to lane change detection. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (pp. 3405-3411). IEEE.

GTAV. (2022). *Grand Theft Auto V*. Retrieved October 28, 2022 from https://www.rockstargames.com/gta-v

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Hough, P. V. (1962). U.S. Patent No. 3,069,654. Washington, DC: U.S. Patent and Trademark Office.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., ... & Ng, A. Y. (2015). An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716.*

Jaladi, S. R., Chen, Z., Malayanur, N. R., Macherla, R. M., & Li, B. (2022, October). End-To-End Training and Testing Gamification Framework to Learn Human Highway Driving. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 4296-4301).

Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., & Kornhauser, A. (2017). Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. *arXiv preprint arXiv:1712.01397.*

Muller, U., Ben, J., Cosatto, E., Flepp, B., & Cun, Y. (2005). Off-road obstacle avoidance through end-to-end learning. *Advances in neural information processing systems, 18*.

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., & Van Gool, L. (2018, June). Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)* (pp. 286-291). IEEE.

Novello, G. A. M., Yamamoto, H. Y., & Cabral, E. L. L. (2021). An end-to-end approach to autonomous vehicle control using deep learning. *Revista Brasileira de Computação Aplicada, 13(3), 32-41.*

Pakgohar, A., Tabrizi, R. S., Khalili, M., & Esmaeili, A. (2011). The role of human factor in incidence and severity of road crashes based on the CART and LR regression: a data mining approach. *Procedia Computer Science, 3*, 764-769.

Richter, S. R., Vineet, V., Roth, S., & Koltun, V. (2016, October). Playing for data: Ground truth from computer games. In *European conference on computer vision* (pp. 102-118). Springer, Cham.

Schöning, J., Kettler, J., Jäger, M. I., & Gunia, A. (2023). Grand Theft Auto-based cycling simulator for cognitive enhancement technologies in dangerous traffic situations. *Sensors, 23*(7), 3672.

Sekkat, A. R., Dupuis, Y., Vasseur, P., & Honeine, P. (2020, May). The omniscape dataset. In *2020 IEEE International conference on robotics and automation (ICRA)* (pp. 1603-1608).

Shafaei, A., Little, J. J., & Schmidt, M. (2016). Play and learn: Using video games to train computer vision models. *arXiv preprint arXiv:1608.01745.*

Shi, J., & Zhao, L. (2021). A Review of Lane Detection Based on Semantic Segmentation. *International Journal of Advanced Network, Monitoring and Controls, 6*(3), 1-8.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Udacity (2017). *Finding lane lines on the road.* Retrieved May, 2023 from https://github.com/udacity/CarND-LaneLines-P1

Url-1 <*https://rageplugin*hook.net>, date retrieved 28.10.2022.

Url-2 < *https://tr.gta5-mods.com/scripts/hood-camera>,* date retrieved 28.10.2022.

Url-3 < *https://tr.gta5-mods.com/scripts/self-drive-plugin>,* date retrieved 28.10.2022.

Van Gansbeke, W., De Brabandere, B., Neven, D., Proesmans, M., & Van Gool, L. (2019). End-to-end lane detection through differentiable least-squares fitting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.

World Health Organization (WHO). (2018). *Global Status Report on Road Safety 2018.* Accessed: 28.10.2022. Retrieved from https://www.who.int/publications/i/item/9789241565684

Yun, H., & Park, D. (2021). Virtualization of self-driving algorithms by interoperating embedded controllers on a game engine for a digital twining autonomous vehicle. *Electronics, 10*(17), 2102.

Zhang, Z. (2020, November). Z-Net: A Novel Way of Lane Detection. In *Journal of Physics: Conference Series* (Vol. 1682, No. 1, p. 012013). IOP Publishing.

Zou, Q., Jiang, H., Dai, Q., Yue, Y., Chen, L., & Wang, Q. (2019). Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology, 69*(1), 41-54.

Zou, Z., Shi, T., Li, W., Zhang, Z., & Shi, Z. (2020). Do game data generalize well for remote sensing image segmentation?. *Remote Sensing, 12*(2), 275.