International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 6 (December 2019)
www.ijemr.net https://doi.org/10.31033/ijemr.9.6.6

Autonomous Vehicle and Augmented Reality Usage

Dr. Yusuf UZUN[1] and Mehmet BİLBAN[2]
[1]Assistant Professor, Department of Computer Engineering, Necmettin Erbakan University, TURKEY
[2]Lecturer, Department of Computer Technologies, Necmettin Erbakan University, TURKEY

[1]Corresponding Author: yuzun@erbakan.edu.tr

ABSTRACT

With the development of autonomous development technology, the need for additional applications to be used inside and outside the vehicle is increasing. As a result of the literature review, many applications have been developed to display vehicle data directly on the monitor, with reflections on glass, and on hardware devices. These applications have been developed only for a defined problem and for a particular autonomous system. In this study, a basic autonomous vehicle software infrastructure and mobile Augmented Reality application that can work on Android devices have been developed. The Mobile Augmented Reality app serves inside and outside the vehicle. In addition, this application has been shown to support multiple autonomous system infrastructures.

***Keywords*—** Augmented Reality, Deep Learning, Mobile Application, Autonomous Vehicle

I. INTRODUCTION

Autonomous systems are defined as self-navigating vehicles capable of detecting the outside world using data from sensors. Autonomous systems include industrial robots, UAVs, driverless automobiles, unmanned underwater vehicles, unmanned agricultural vehicles, unmanned aerial-space vehicles. As the automobile industry has more consumer markets than other areas, there has been a further increase in the developments in the automotive sector. Autonomous vehicle technology has progressed rapidly over the last decade as a result of competition in artificial intelligence technology and automotive technologies [1].

Augmented Reality (AR) technology goes back to the 1960s. The first system infrastructure was developed for both AR and Virtual Reality (VR). It is known that the idea of how two-dimensional objects can be displayed in three dimensions to the user in the real world is tried with a limited number of computers [2].

AR is a technology used by many public or private companies before developing autonomous systems. This technology is used as a compass to guide many areas, such as designed algorithms, system integration, communication protocols, and display mode.

Autonomous vehicles will become the indispensable technology of personal transportation in the near future. Based on the question of how the human eye perceives the real world, autonomous vehicle developers sought ways to show how an autonomous vehicle perceives the outside world, and decided that AR is one of the technologies that can be used [3]. With this technology, it is aimed to combine and display data from sensors and cameras with real world objects. AR technology is a technology that can easily show all complex situations about speed, energy, road conditions and vehicle.

The fact that autonomous vehicles have attracted great interest and the studies carried out by many state and private institutions in this field has played a major role in making autonomy the main subject in this study. In this study, for an autonomous tool, the basic software architecture was developed and the AR application was built on it.

In this study, the developed AR application has the infrastructure that can be used in the following autonomous systems.

I. Self-Driving Vehicles
II. Unmanned Aerial Vehicles
III. Unmanned Aerial – Spacecraft
IV. Unmanned Submarine Vehicles
V. Unmanned Agricultural Vehicles
VI. Thermal and Nuclear Power Plants
VII. Internet of Things Applications

An autonomous software architecture has been developed using MIT-RACECAR, which MIT offers developers as open source. All the software and hardware infrastructure we have developed is designed to work on this tool.

Mobile Augmented Reality application is an Android based mobile application. This mobile application is designed to work on all Android devices.

For Server-Client Architecture, software running on the server has been developed. This software sends the data it receives from the autonomous vehicle to the mobile device. It also processes the data received from the peripherals of the autonomous vehicle and shares it with the mobile AR.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 6 (December 2019)
https://doi.org/10.31033/ijemr.9.6.6

II. METHODOLOGY

MIT-RACECAR is an open source 1/10 scale autonomous vehicle platform for robotics research and education. This platform has sensors and computer hardware installed on the Traxxas RC chassis. The design and development of RACECAR was done by the Lincoln Laboratory's BeaverWorks Initiative, the Department of Aeronautics and Space Science, and the Information and Decision Systems Laboratory at the Massachusetts Institute of Technology [4]. RACECAR and its hardware components are shown in Figure 1 [5].

Figure 1: MIT RACECAR

2.1 Converting RACECAR to an Autonomous Vehicle

Since a vehicle cannot perform humanoid driving actions on its own, systems in which hardware and software work together are used. The need analysis was based on the question of how to make human driving actions in a real world a robot. To enable autonomous movement of a vehicle, different software and hardware systems must be integrated. All components are shown in Figure 2.

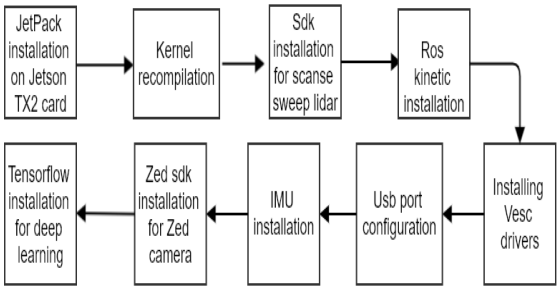

Figure 2: Autonomous Vehicle Components

2.2 ROS Architecture and Concepts

ROS is a framework for writing robot software. It is used in many projects for research and commercial purposes. Today, robotics application developers can develop robot software with ROS quickly and flexibly [6]. Especially if more than one application wants to communicate with each other, ie to send data, it will benefit from the framework provided by ROS. When sending and receiving data, it is more convenient to use ROS instead of socket programming [7]. In order to create nodes on ROS, we also need ROS client libraries to broadcast services and messages.

ROS client libraries consist of code collections with functions to implement ROS concepts. In this study, the rospy client library developed for ROS was used.

The Rospy client library allows Python programmers to quickly interface with ROS Threads, Services, and Parameters. Rospy's design supports application speed (ie developer time) based on runtime performance. Thus, algorithms can be quickly prototyped and tested [8].

2.3 Data Collection

In order to move the vehicle without a driver, data must be collected on the road to be driven. Data were collected in the northern part of Necmettin Erbakan University Seydisehir Ahmet Cengiz Engineering Faculty 1st Floor. To collect data, a code was developed with the Python programming language. This developed code takes speed, angle and 30 visual / sec data from the vehicle driving in the corridor and writes to the file. The area where the data is collected has a large exterior window, lights at different angles, and reflections from tiles on a bright floor. All the collected data is used as a database for end-to-end learning in later stages.

2.4 Deep Learning Network Structure

While Machine Learning techniques give good results in grouping or clustering variables, accuracy decreases as data size increases. Therefore, Deep Learning is one of the most important developments in the field of artificial intelligence. Deep learning is a field of study that involves machine learning algorithms such as artificial neural networks and the like, trying to model abstraction in data. In other words, Deep Learning algorithms are an ANN set that can better represent large-scale data sets.

In this study, the end-to-end learning model developed by NVIDIA for autonomous vehicles is used [9]. The CNN structure is shown in Figure 3 [10]. This network consists of the Keras Lambda function for image normalization, three 5x5 convolution layers, two 3x3 convolution layers, and fully connected layers. This model also includes conversion from RGB to YUV color space, 2x2 shifting on a 5x5 convolution layer.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 6 (December 2019)
https://doi.org/10.31033/ijemr.9.6.6

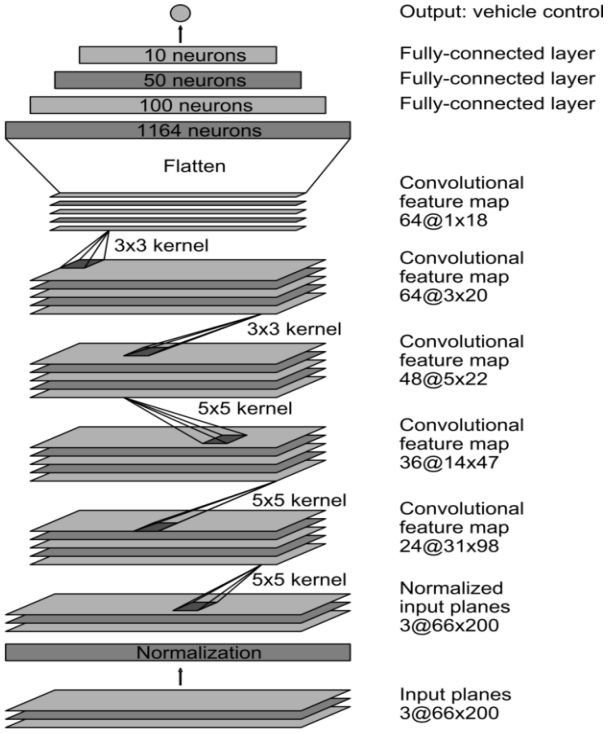

Figure 3: NVIDIA Convolutional Neural Network [13]

CNNs have made a radical change in image recognition [11], [12]. Prior to the widespread adoption of CNNs, most image recognition operations were performed by manual feature extraction followed by a classifier. Together with the invention of CNNs, the features are automatically learned from training examples. The CNN approach is particularly powerful in image recognition tasks because convolution captures the 2D nature of images [9].

2.5 Augmented Reality Tracking and Tracking System Architecture

In 1997, Azuma defined AR as a system that fulfills 3 basic criteria [14]:

I. Real and virtual combination,
II. Real-time interaction,
III. System where 3D real and virtual objects are together

Today, AR is defined as the technology that allows the virtual images created in computer environment to be placed and used in the real world. According to the actions of the users, the positioning of the graphical objects according to the most appropriate place is very important for AR applications. Today, AR applications can be divided into two groups according to their usage areas: 1) Marker-based technology, 2) Markesless.

Marker-Based AR System: Marker-based tracking is achieved by physically adding markers to the object to be monitored. The markers must have a suitable design according to the area to be used. Pointers placed on real objects have already been introduced to the system. The position of markers on real objects must be determined in advance for quick and easy detection.

Markerless AR System: The Markerless method is a method that performs object recognition without the need for additional placement or pointers on the media. This method performs object recognition by inference from models created by computer-aided design (CAD) or from a set of points representing an existing 3D object [15]. Markerless recognition ensures minimum process readiness for users. Markerless object recognition requires more complex operations than marker object recognition. This method has not proved to be better than a marker-based method [16].

2.6 Object Recognition with Android Neural Networks and Tensor Flow Lite

Android Artificial Neural Networks Application Programming Interface (UPA) is an Android C UPA designed to perform compute-intensive operations for machine learning on mobile devices. UPA is available on all devices with Android 8.1 (UPA Level 27) or higher. Android Artificial Neural Networks support the ability to extract from previously trained and built-in models with UPA.

The runtime of Android neural networks is determined by optimally allocating the available workload based on hardware features and application requirements on the device, graphics processing units (GPUs), and signal processors. Figure 4 shows the architecture of the Android neural networks [17].

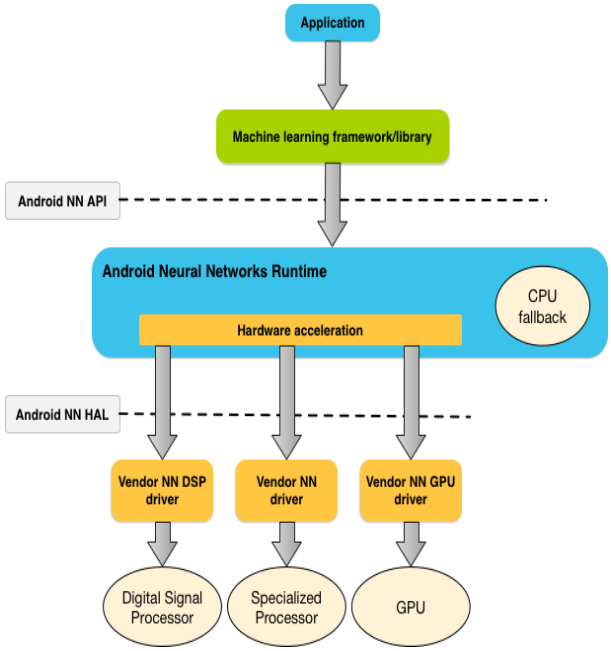

Figure 4: Android Artificial Neural Networks Application Programming Interface [17]

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 6 (December 2019)
https://doi.org/10.31033/ijemr.9.6.6

Smartphones can take advantage of inference operations on the device for machine learning tasks. In general, it is common for mobile devices to use machine learning models in the cloud. TensorFlow Lite enables extraction in the mobile device. TensorFlow Lite supports Android and iOS platforms. Figure 5 illustrates the mobile platforms that TensorFlow Lite supports. The first step involves converting a trained TensorFlow model to the TensorFlow Lite file format (.tflite) using the TensorFlow Lite converter. The converted model file is used in the application [18].

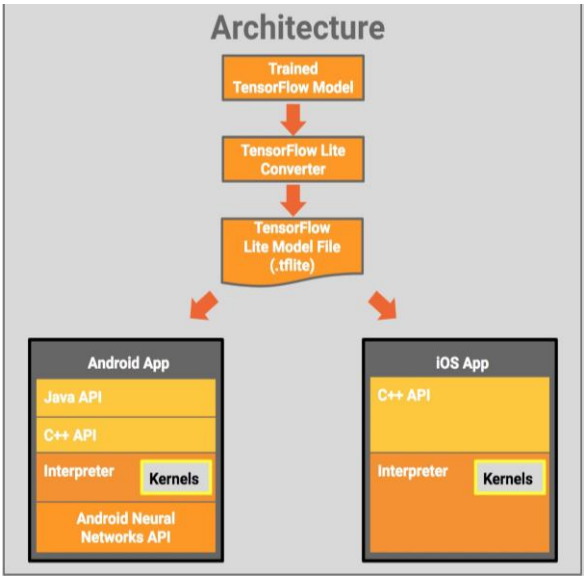

Figure 5: TensorFlow Lite architecture [19]

In this study, TensorFlow Lite was added to the Android application. At the same time, the previously trained COCO SSD MobileNet V1 model was incorporated into the autonomous Android mobile AG application. At the same time with this provided model, 10 objects can be recognized and located in an image. This model is trained to recognize 80 object classes [20].

2.6 Developing a Mobile AR Application for an Autonomous Vehicle

We can control how an autonomous vehicle perceives the outside world remotely or through in-vehicle applications. Traditional mobile applications often operate in a server-client architecture to provide an interactive service. Mobile AR application developed in the study, instant vehicle speed value change, simultaneous energy consumption and real-time steering angle change on the screen to show the data must be taken from the vehicle. Data acquisition through the vehicle is provided by a developed server. Figure 6 represents the communication architecture between a server and mobile AR developed on RACECAR.

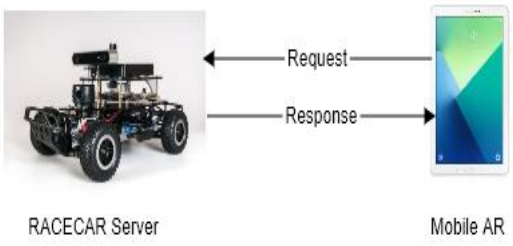

Figure 6: RACECAR Server – Mobile AR

RESTful data communication architecture is used between server and client. REST is expressed as representative state transfer. Using the HTTP protocol, GET and POST requests, such as requests to respond to these requests in various formats is a flexible way of communication.

IV. RESEARCH RESULTS

Mean square error (MSE) loss function was used as the performance evaluation criterion during the training phase of the autonomous vehicle. Figure 7 shows how the values of the MSE loss function change as the number of steps (epochs) for training and validation data sets increases. The loss of MSE decreased rapidly after the 1st epoch and was decreased by 5-8. epoch remains constant between. 8-9. The epochs show a change but the error rate is neglected as a result of the test operations.

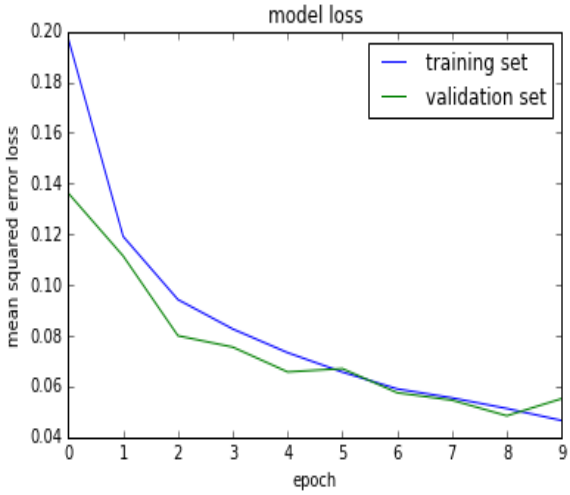

Figure 7: MSE loss function graph

Figure 8 shows one of the objects recognized by real-time mobile AR.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 6 (December 2019)
https://doi.org/10.31033/ijemr.9.6.6

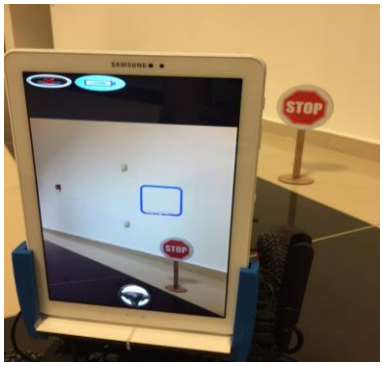

Figure 8: Identifying the Stop Sign

V. CONCLUSION

The Mobile AR application was designed and developed for use on devices with Android operating systems. This application has a substructure for internal and external use. In the same way, it is shown how the outside world is perceived with AR technology to make the passengers traveling in the vehicle feel safer. The energy, speed and steering changes of a vehicle performing autonomous driving were monitored remotely by mobile AR without driver and traveling passenger.

This infrastructure, which we have developed, is intended to be useful in multiple autonomous systems, to be modular, to be mobile and to contribute to the literature by shedding light on some future problems.

As a result of the tests, the vehicle needs continuous high battery values and unstable behaviors at low voltage values (for 12.6V and below values) were observed. In order not to put an extra burden on the vehicle's power consumption, the electricity requirement of the mobile device on which the mobile AR application operates was not taken over the vehicle and was provided from the mobile device's own battery.

REFERENCES

[1] V. Rastogi. (2017). *Virtual reality based simulation testbed for evaluation of autonomous vehicle behavior algorithms*. A Thesis Presented to the Graduate School of Clemson University in Partial Fulfillment of the Requirements for the Degree Master of Science Computer Science. Clemson University.

[2] I. E.Sutherland. (1968). A head-mounted three dimensional display. *In Proceedings of the AFIPS Fall Joint Computer Conferance*, pp. 757–764.

[3] J. Fredriksson, B. Kulcsar, & J. Sjöberg. (2015). *Proceedings of the 3rd international symposium on future active safety technology towards zero traffic accidents*. Available at: http://publications.lib.chalmers.se/records/fulltext/222422/local_222422.pdf.

[4] MIT Racecar. (2017). *MIT racecar mobile platform*. Available: http://racecar.mit.edu.

[5] Openzeka. (2019). *Openzeka online*. Available at: https://openzeka.com/.

[6] ROS. (2019). *Robot operating system*. Available at: http://www.ros.org/.

[7] L. Joseph. (2018). *Robot operating system for absolute beginners_ robotics programming made easy-apress*. Available at: https://www.apress.com/gp/book/9781484234044.

[8] http://wiki.ros.org/rospy.

[9] M. Bojarski *et al.* (2016). *End to end learning for self-driving cars*. Available at: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf.

[10] Y. LeCun *et al.* (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computer*, pp. 541–551.

[11] A. Krizhevsky, I. Sutskever, & G. E. Hinton. (2015). ImageNet classification with deep convolutional neural networks. *Journal of Geotechnology and Geoenvironmental Engineering*, *12*, 04015009.

[12] K. Z. and D. D. T. Mariusz Bojarski, Ben Firner, Beat Flepp, Larry Jackel, & Urs Muller. (2016). *End-to-end deep learning for self-driving cars*. Available at: https://devblogs.nvidia.com/deep-learning-self-driving-cars/.

[13] R. T.Azuma, (1997). *Survey of augmented reality*. Available at: https://www.cs.unc.edu/~azuma/ARpresence.pdf.

[14] J. Paulo Lima *et al.* (2017). Markerless tracking system for augmented reality in the automotive industry. *Expert Systems with Applications*, *82*, 100–114.

[15] P. Khandelwal, P. Swarnalatha, N. Bisht, & S. Prabu. (2015). *Detection of features to track objects and segmentation using grabcut for application in marker-less augmented reality*. *Procedia Computer Science*, *58*, 698–705.

[16] Android Developers. (2019). *Android neural networks API*. Available at: https://developer.android.com/ndk/guides/neuralnetworks.

[17] Arun Mani Sam. (2019). *Developing SSD-Object detection models for android using tensorflow*. Available at: https://www.inspirisys.com/objectdetection_in_tensorflowdemo.pdf.

[18] L. Moroney. (2018). *Using tensorflow lite on android*. Available at: https://miro.medium.com/max/1573/0*Bt9qwKDjd1xi5RDd.

[19] TensorFlow. (2019). *TensorFlow lite object detection*. Available at: https://www.tensorflow.org/lite/models/object_detection/overview.