MANTECH Publications

International Journal of Robotics and Autonomous Systems
Volume 7, Issue 2, July-December, 2022

Use of Autonomous Vehicles with Augmented Reality

Dr. Balveer Dohare[1], Rahul Sahu[2], Hansraj Thakur[3]

Professor[1], Students[2, 3]

Department of Mechanical Engineering

Bhopal Institute of Technology & Management

Corresponding Author's Email: - *rahul08sahhu@gmail.com[2]*

Abstract

The need for more apps to be employed inside and outside the car is growing as autonomous development technology develops. As a consequence of the literature research, several programmes have been created to show vehicle data directly on the monitor, with reflections on glass, and on hardware devicesprogrammes have been created to show vehicle data directly on the monitor, with reflections on glass, and on hardware devices. These apps have only been created for a certain autonomous system and a defined issue. In this study, an Android-compatible mobile augmented reality application and a rudimentary autonomous car software infrastructure have been created. Both inside and outside of the car, the mobile augmented reality software is useful. Additionally, it has been demonstrated that this programme supports a number of autonomous system infrastructures.

Keywords: *- Augmented Reality, Deep Learning, Mobile Application, Autonomous Vehicle*

INTRODUCTION

Autonomous systems are referred to be self-driving cars that can sense their surroundings utilising information from sensors. Industrial robots, UAVs, driverless cars, unmanned underwater vehicles, unmanned agricultural vehicles, and unmanned aerial-space vehicles are examples of autonomous systems. There has been an upsurge in advances in the automotive sector since it has larger consumer markets than other industries. The battle between artificial intelligence and automotive technologies has

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

accelerated the development of autonomous vehicle technology during the past ten years [1].

Technology for augmented reality (AR) has existed since the 1960s. The first system infrastructure for both augmented reality and virtual reality was created (VR). It is well known that a small number of computers have been used to test the concept of showing users two-dimensional items in three dimensions in the actual world [2].

Before creating autonomous systems, numerous public or private enterprises exploited augmented reality technologies. In many areas, including created algorithms, system integration, communication protocols, and display mode, this technology serves as a compass.

In the not-too-distant future, autonomous cars will replace other transportation technologies as necessary ones. Autonomous vehicle engineers looked for ways to demonstrate how an autonomous car senses the outside environment based on the question of how the human eye perceives the actual world, and they realised that AR is one of the technologies that can be employed [3]. This technology aims to mix and show data from sensors and cameras with actual items from the real world. The use of augmented reality (AR) technology makes it simple to see complicated problems, including speed, energy, road conditions, and vehicles.

The popularity of autonomous cars and the numerous studies in this area that have been conducted by both public and commercial organisations have greatly contributed to making autonomy the main topic of this study. In this study, the foundational software architecture for an autonomous tool was created, and the AR application was constructed on top of it.

application was constructed on top of it.
The infrastructure for the built-in AR application in this study may be applied to the following autonomous systems.

1. Self-Driving Vehicles
2. Unmanned Aerial Vehicles
3. Unmanned Aerial – Spacecraft
4. Unmanned Submarine Vehicles
5. Unmanned Agricultural Vehicles
6. Thermal and Nuclear Power Plants
7. Internet of Things Applications

Using MIT-RACECAR, which MIT makes available to developers as open source, an autonomous software architecture has been created. Our whole

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

hardware and software infrastructure has been created with this tool in mind.

An Android-based smartphone application for augmented reality is available. All Android smartphones are compatible with this mobile application.

There is software that runs on the server in a server-client architecture. The mobile device gets the data that this software receives from the autonomous car. Additionally, it analyses and transmits to the mobile AR the data obtained from the autonomous vehicle's peripherals.

METHODOLOGY

An open-source, 1/10-size autonomous car platform called MIT-RACECAR is used for robotics research and instruction. The Traxxas RC chassis of this platform includes sensors and computer hardware installed. The Massachusetts Institute of Technology's Information and Decision Systems Laboratory, the Department of Aeronautics and Space Science, and the BeaverWorks Initiative of the Lincoln Laboratory all contributed to the design and development of RACECAR [4]. Figure 1 [5] depicts RACECAR and the hardware elements that make it up.



Figure 1: MIT RACECAR

Converting RACECAR to an Autonomous Vehicle

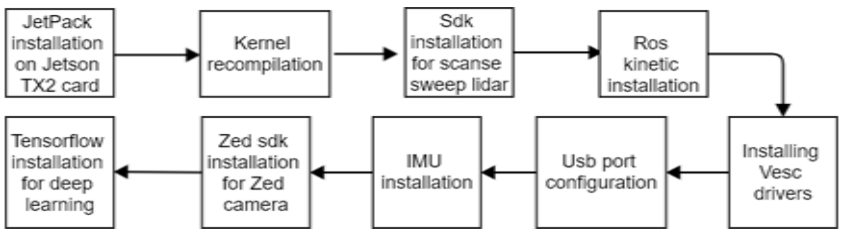


Figure 2: Autonomous Vehicle Components

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Systems that combine hardware and software are utilised since a vehicle is unable to conduct humanoid driving movements on its own. The requirement analysis was based on the query of how to automate real-world human driving operations. Different hardware and software technologies must be combined in order to allow autonomous movement of a vehicle. Figure 2 displays every component.

ROS Architecture and Concepts

The ROS framework may be used to construct robot software. It is used in a number of initiatives with both commercial and academic goals. Today, ROS enables robot software developers to produce robot code quickly and adaptably [6]. The infrastructure provided by ROS will be beneficial, particularly if numerous applications need to exchange data. It is more feasible to provide and receive data using ROS as opposed to socket programming [7]. To develop ROS nodes, we also need ROS client libraries that can broadcast services and messages.

The ROS client libraries are collections of code that implement ROS concepts. In this study, the Rospy client library developed for ROS was used.

Thanks to the Rospy client library, Python programmers may easily communicate with ROS (i.e., services and parameters). The Rospy design encourages application speed based on runtime performance (i.e., developer time). As a result, algorithm prototypes and testing may be carried out very quickly [8].

Data Collection

Before leaving without a driver, information regarding the intended route must be acquired. Data were collected on the northern half of the first floor of the Seydisehir Ahmet Cengiz Engineering Faculty of Necmettin Erbakan University. The Python programming language was used to develop a piece of code that gathered data.

The code of this software creates a file containing the speed, angle, and 30-second visual data from the moving automobile. The area where the data is obtained has a large glass wall on the exterior, variously placed lighting sources, and reflections from the glittering tiles on the floor. End-to-end learning's latter stages use the complete collected data as a database.

Deep Learning Network Structure

Although machine learning approaches perform well when grouping or clustering

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

variables, accuracy declines with increasing data volume. As a result, deep learning is one of the most significant advancements in artificial intelligence declines with increasing data volume. As a result, deep learning is one of the most aims to model abstraction in data. In other words, deep learning algorithms are a subset of ANNs that are better suited to handling massive amounts of data. The end-to-end learning approach created by NVIDIA for autonomous cars is utilised in this work [9]. Figure 3 [10] depicts the CNN structure in detail. Three 5x5 significant advancements in artificial intelligence.

Artificial neural networks and other machine learning algorithms are used in the research of "deep learning," which convolution layers, two 3x3 convolution layers, the Keras Lambda function for picture normalisation, and fully linked layers make up this network. Additionally, this model has 2x2 shifting on a 5x5 convolution layer and RGB to YUV colour space conversion.

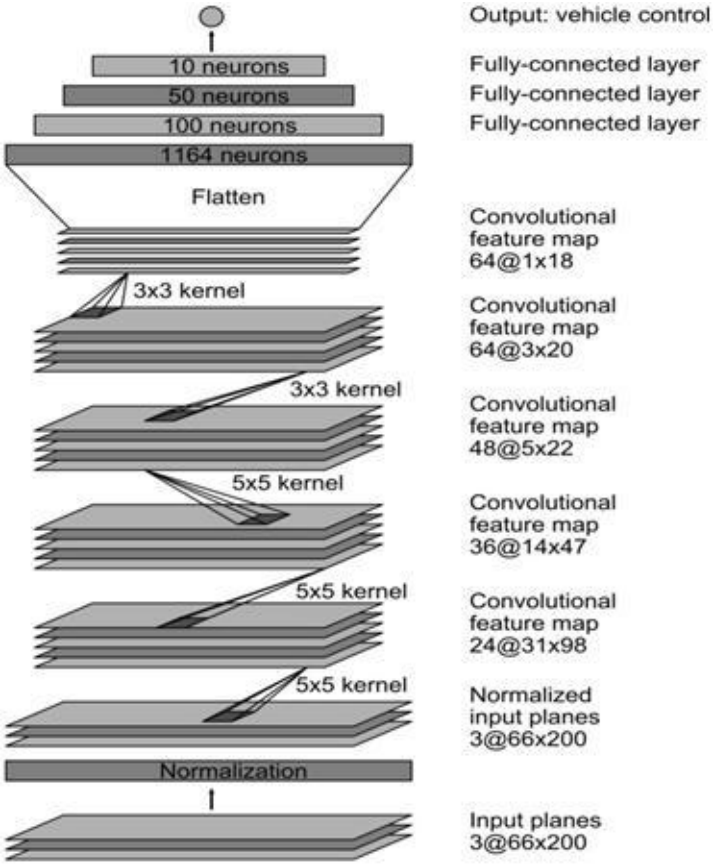


Figure 3: NVIDIA Convolutional Neural Network [13]

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Image recognition has undergone a significant transformation thanks to CNNs [11], [12]. The majority of picture identification tasks were formerly carried out by human feature extraction followed by a classifier until CNNs became widely used.

The characteristics are automatically learned from training examples, along with the development of CNNs. Because convolution can take into account how pictures look in 2D, the CNN technique works well in image identification applications [9].

Augmented Reality Tracking and Tracking System Architecture

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

1. Real and virtual combination,
2. Real-time interaction,
3. System where 3D real and virtual objects are together

The definition of augmented reality today is the technology that enables the placement and usage of virtual pictures generated in a computer environment outside of it. The placement of visual items in the most appropriate location in accordance with user actions is crucial for AR applications. According to their intended uses, AR apps nowadays may be split into two categories: 1) marker-based technology; and 2) markerless

Marker-Based AR System

Markers are physically attached to the item to be tracked in marker-based tracking. The markers need to be designed appropriately for the space they will be used in. The system already uses pointers that are placed on actual objects. So that markers can be found quickly and easily, they need to be placed on real objects in a certain way.

Markerless AR System

The markerless approach is a technique that can identify objects without placing additional markers or pointers on the media. By drawing conclusions from CAD models or a group of points that together describe an actual 3D item, this technique recognises objects [15].

Markerless recognition guarantees users a minimal level of process readiness. More complicated processes are needed for markerless item recognition than marker-based object recognition. This approach hasn't been shown to be superior to a marker-based approach [16].

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Object Recognition with Android Neural Networks and Tensor Flow Lite

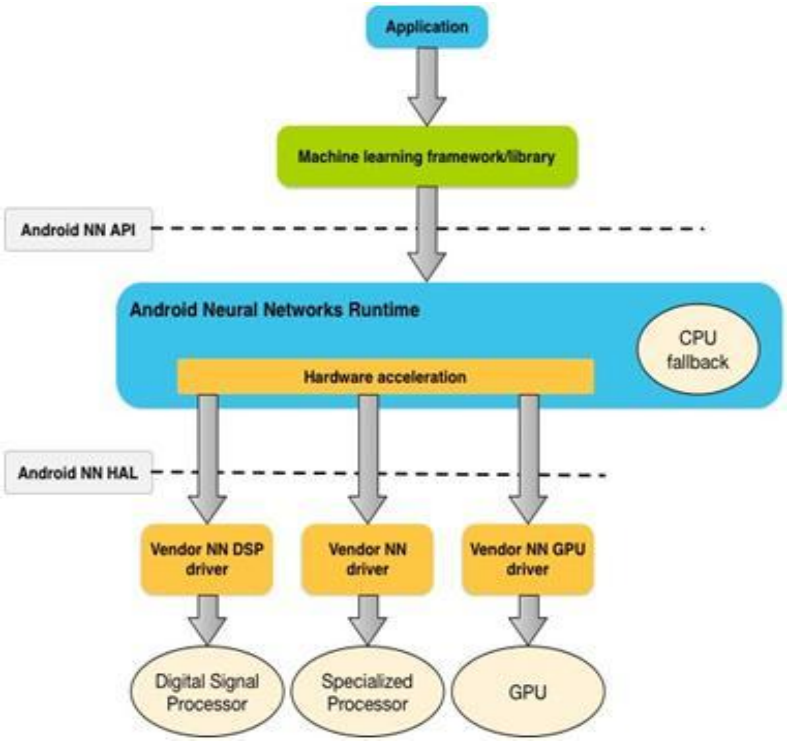


Figure 4: Android Artificial Neural Networks Application Programming Interface [17]

An Android C UPA called Android Artificial Neural Networks Application Programming Interface (UPA) was created to carry out computationally demanding machine learning tasks on mobile devices. All smartphones running Android 8.1 (UPA Level 27) or above support UPA. Android artificial neural networks enable the use of UPA to extract data from built-in and previously trained models. The graphics processing units (GPUs), signal processors, and hardware requirements on the device are used to efficiently distribute the workload in order to calculate the runtime of Android neural networks. The architecture of the Android neural networks is depicted in Figure 4 [17]. Smartphones can use inference processes on the device to do machine learning tasks. In general, machine learning models in the cloud are commonly used by mobile devices. TensorFlow Lite allows for extraction on mobile devices. TensorFlow Lite is available for both Android and iOS. TensorFlow Lite supports the mobile systems depicted in Figure 5. The first step is to use the TensorFlow Lite converter to convert a trained TensorFlow model to the TensorFlow Lite file format (.tflite). The programme [18] makes use of the transformed model file.

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

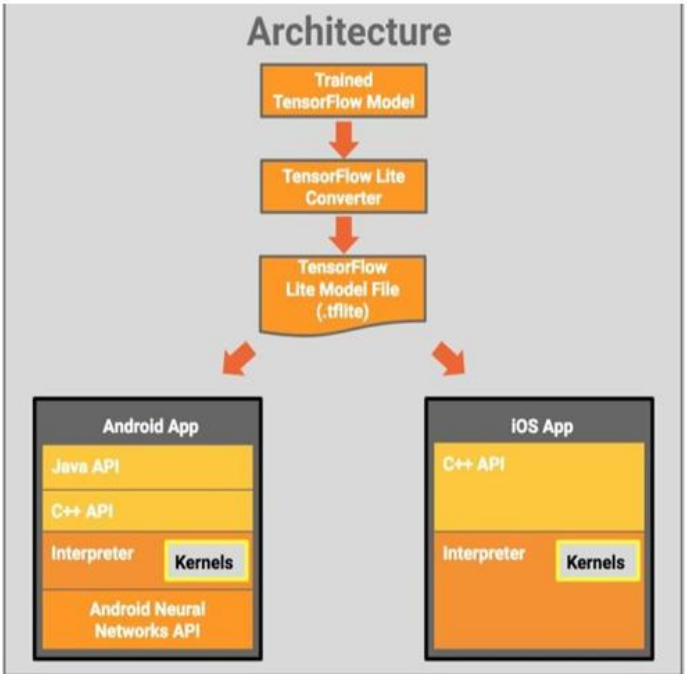


Figure 5: TensorFlow Lite architecture [19]

TensorFlow Lite was incorporated into the Android application for this investigation. Simultaneously, the previously trained COCO SSD MobileNet V1 model was integrated into the self-contained Android mobile AG application. With the model supplied, 10 things in a picture may be identified and found at the same time. This model has been trained to detect 80 different object types [20].

Developing a Mobile AR Application for an Autonomous Vehicle

We can influence how an autonomous car sees its surroundings remotely or through in-vehicle apps. To deliver an interactive service, traditional mobile applications frequently use a server-client architecture. The data must be gathered from the car using the mobile AR application created for the study, which shows instant vehicle speed value change, simultaneous energy consumption, and real-time steering angle change on the screen.

A created server provides data gathering via the vehicle. Figure 6 depicts the communication architecture between a server and RACECAR-based mobile AR.

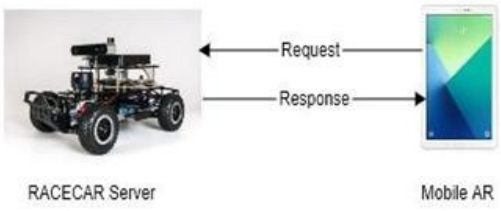


Figure 6: RACECAR Server – Mobile AR

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

Between the server and the client, a RESTful data exchange architecture is employed. REST stands for representational state transfer. Using the HTTP protocol, GET and POST requests, as well as requests to react to these requests in a variety of ways, is a versatile method of communication.

RESEARCH RESULTS

During the autonomous vehicle's training phase, the MSE loss function was employed as the performance assessment criterion. Figure 7 depicts how the values of the MSE loss function fluctuate as the number of training and validation data set steps (epochs) increases. MSE loss fell significantly after the first period and was reduced by 8–9. Between 8 and 9, the epoch remains unchanged. The epochs vary, but the error rate is ignored due to the test operations.

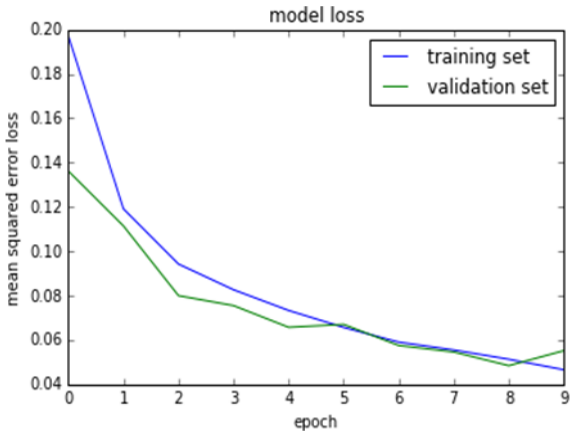


Figure 7: MSE loss function graph

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



Figure 8: Identifying the Stop Sign

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

CONCLUSION

The Mobile AR application was created and built for usage on Android-powered smartphones. This application has an internal and external substructure. Similarly, it is demonstrated how the outside environment is interpreted using AR technology to make passengers in the vehicle feel safer. The energy, speed, and steering adjustments of a vehicle conducting autonomous driving were remotely monitored by mobile AR without the presence of a driver or a travelling passenger.

This infrastructure, which we created, is meant to be helpful in a variety of autonomous systems, to be modular and movable, and to contribute to the literature by shining light on certain future difficulties.

As a result of the testing, the vehicle requires constant high battery values, and unstable behaviours at low voltage levels (for 12.6V and lower values) were noticed. The electricity demand of the mobile device on which the mobile AR application functions was not taken over by the car and was delivered from the mobile device's own battery in order to not place an additional stress on the vehicle's power consumption.

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.

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

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.

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved

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_tensorflowde mo.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/over view.

Page 82-93 © MANTECH PUBLICATIONS 2022. All Rights Reserved