Self-Driving Car Using Deep Learning and Convolutional Neural Network

[1] Ms.Rupali mohan gaikwad
M.tech student

[2]Dr Syed Sumera Ali
Associate Professor

[3]Prof A.T. Jadhav
Assistant Professor

[4]Dr.D.L.Bhuyar
Professor & Head of Department

[1][2][3][4] Dept of Electronics and Telecommunication Engg,
[1][2][3][4]Chh.shahu College of Engineering Aurangabad, Maharashtra, India

Abstract: Self-driving cars have emerged as a transformative technology in intelligent transportation systems, aiming to reduce human error, improve road safety, and enable autonomous navigation. This paper presents a deep learning-based approach for semantic scene understanding using Convolutional Neural Networks (CNNs) to facilitate autonomous driving. A CNN-based semantic segmentation model is trained on urban street datasets to classify each pixel of an input image into predefined categories such as road, vehicles, pedestrians, buildings, and vegetation. The model's predictions are used to detect drivable areas and obstacles, serving as the perception module in the autonomous driving pipeline. Performance evaluation shows that the system achieves high segmentation accuracy and is capable of generalizing across various road scenarios. The results demonstrate the feasibility and effectiveness of deep learning, particularly CNNs, in achieving robust scene understanding for real-time self-driving applications.

Keyword: Distributed calculations, synchronous, critical sections, Neural Networks, HSV filters, ranger finder (LIDAR), 3D Radar, 3D map, sockets, TCP/IP, Google Maps, Artificial Intelligence.

I. INTRODUCTION

Object detection plays a critical role in the field of computer vision, enabling machines to perceive and interpret visual data by locating and classifying objects within an image or video stream. With the rise of deep learning, object detection algorithms have achieved remarkable accuracy and speed, making them suitable for real-time applications such as autonomous driving, surveillance, industrial automation, and smart cities. Among the most influential object detection frameworks is the YOLO (You Only Look Once) family, which approaches object detection as a regression problem and performs detection in a single forward pass through a neural network. This one-stage architecture provides a significant advantage in terms of speed compared to traditional two-stage detectors like Faster R-CNN. YOLOv5, developed by Ultralytics, represents the fifth and most optimized version of the YOLO series. Unlike its predecessors, which were built on the Darknet framework, YOLOv5 is implemented in PyTorch, offering greater flexibility, faster training, and seamless integration with modern deep learning tools. It incorporates advanced components such as CSPDarknet as the backbone, PANet with FPN as the neck, and custom detection heads to efficiently detect objects at multiple scales.

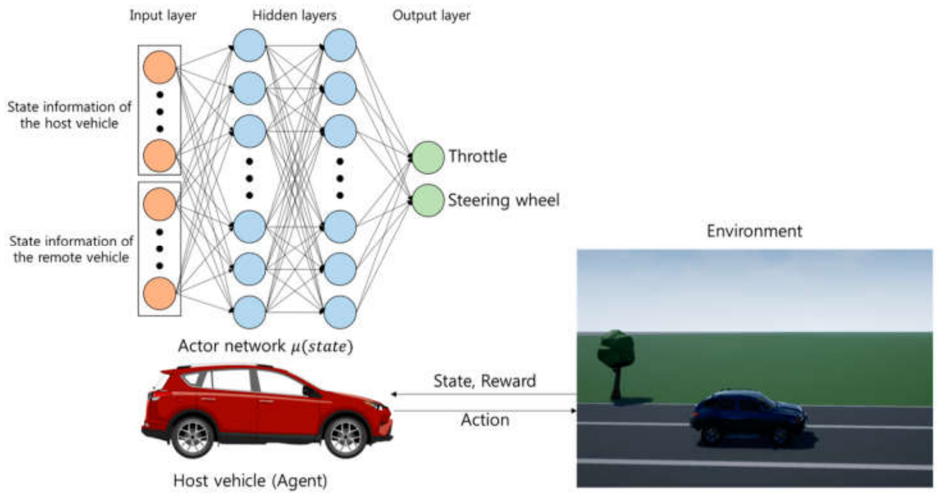

Figure.1: Self-Driving Cars with Convolutional Neural Networks

Furthermore, YOLOv5 includes various enhancements such as auto-anchor tuning, Mosaic data augmentation, label smoothing, and support for multiple model sizes (n, s, m, l, x) that balance performance and computational efficiency. These features make YOLOv5 suitable for deployment in real-time systems where both accuracy and speed are critical. In this paper, we explore the architecture, training methodology, and performance of YOLOv5 for object detection. We demonstrate its effectiveness on both standard benchmark datasets and real-world scenarios, highlighting its applicability in environments with limited computational resources.

In real-world environments, traditional object detection systems often struggle to deliver real-time performance while maintaining high accuracy, especially under varying lighting, background clutter, and multiple object scales. Many existing models require heavy computational resources, making them unsuitable for deployment in resource-constrained or embedded systems like drones, surveillance cameras, or autonomous vehicles. There is a pressing need for a fast, lightweight, and accurate object detection framework that can detect multiple object classes in diverse conditions with low latency. This project focuses on implementing and analyzing YOLOv5, a deep learning-based object detection model, for real-time applications.

The scope includes:

- Understanding and applying YOLOv5 architecture for object detection tasks.
- Training the model on standard datasets and custom datasets.
- Evaluating model performance in terms of accuracy (mAP), speed (FPS), and model size.
- Exploring the model's deployment potential on resource-limited devices (e.g., Raspberry Pi, Jetson Nano).
- Visualizing and interpreting model outputs for practical understanding.

Objectives

1. To study and implement the YOLOv5 object detection algorithm using PyTorch.
2. To train the model on publicly available or custom datasets with multiple object classes.
3. To evaluate the model's accuracy, speed, and robustness in detecting real-world objects.

4. To analyze the advantages of YOLOv5 over earlier YOLO versions and other detection models like SSD and Faster R-CNN.
5. To demonstrate the feasibility of using YOLOv5 for real-time object detection in smart systems and edge devices.

II. RELATED WORK

The field of computer vision and deep learning has evolved rapidly over the last few decades, laying the groundwork for significant advancements in real-time visual recognition systems, including applications like self-driving vehicles, object detection, and scene understanding. The foundational work of LeCun et al. [1] introduced backpropagation for handwritten digit recognition, marking a pivotal moment in deep learning by demonstrating how convolutional neural networks (CNNs) could be effectively trained on image data. Building on this, Krizhevsky et al. [2] made a breakthrough in large-scale image classification with the development of AlexNet, which significantly outperformed previous models on the ImageNet dataset using deep CNNs and GPU acceleration. Donahue et al. [3] proposed DeCAF, a framework that uses deep convolutional features for generic visual recognition, establishing the concept of transfer learning in CNNs. Similarly, Fei-Fei et al. [4] demonstrated that generative visual models could be learned effectively from few examples using a Bayesian approach, contributing to the field of few-shot learning. Several datasets have played a critical role in benchmarking and driving the progress of deep learning models. Notably, the Caltech-UCSD Birds 200 dataset [5] enabled fine-grained object classification, while the SUN database [6] provided a diverse set of scene categories for large-scale scene recognition tasks. Further advancements came with region-based CNNs, such as R-CNN proposed by Girshick et al. [7], which introduced hierarchical feature extraction for accurate object detection and semantic segmentation. This paved the way for high-performance real-time applications. In the domain of autonomous systems, Bojarski et al. [8] introduced an end-to-end learning approach for self-driving cars, where a CNN directly mapped raw pixels from a camera to steering commands, eliminating the need for hand-engineered features. To support such systems, Otterness et al. [9] evaluated the performance of embedded platforms like the NVIDIA Jetson TX1 for real-time computer vision tasks, highlighting the hardware constraints and optimization challenges. Lastly, Karaman et al. [10] focused on educational outreach and project-based learning, introducing a curriculum for programming self-driving cars at the high school level. Their work emphasized collaborative robotics and early exposure to algorithmic thinking, further expanding the reach of deep learning education. These contributions collectively form the basis for current research trends in real-time computer vision, embedded deep learning, and autonomous driving systems.

III. METHODOLOGY

3.1 Dataset Collection and Preprocessing

To train the semantic segmentation model for autonomous driving, we use publicly available urban driving datasets such as Cityscapes, KITTI, and BDD100K. These datasets contain thousands of real-world driving images with pixel-level annotations for various semantic classes including roads, vehicles, pedestrians, buildings, vegetation, and lane markings. Each image is resized to a fixed resolution (e.g., 256×512) to ensure uniform input dimensions across the dataset. Preprocessing also includes normalization of pixel values and data augmentation techniques such as random cropping, horizontal flipping, brightness/contrast adjustment, and rotation. These augmentations help the model generalize better to varied driving scenarios and lighting conditions.

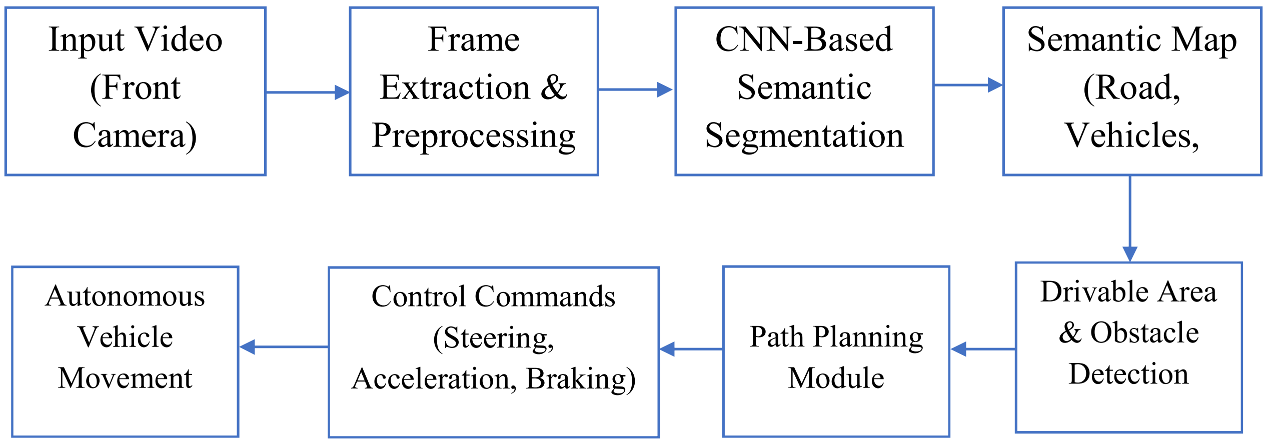
Input Video (Front Camera)
Frame Extraction & Preprocessing
CNN-Based Semantic Segmentation
Semantic Map (Road, Vehicles,
Drivable Area & Obstacle Detection
Path Planning Module
Control Commands (Steering, Acceleration, Braking)
Autonomous Vehicle Movement

Figure.2: Block diagram

3.2 Model Architecture: CNN-Based Semantic Segmentation

For pixel-wise scene understanding, we implement a CNN-based encoder-decoder architecture. Models like U-Net, SegNet, or DeepLabV3+ are chosen due to their effectiveness in dense semantic segmentation tasks. The encoder network extracts hierarchical spatial features using convolutional layers, while the decoder gradually reconstructs the full-resolution output map through upsampling and deconvolution. Skip connections are integrated between encoder and decoder layers to preserve fine-grained spatial information, allowing accurate segmentation of small objects like pedestrians and lane lines. The final output of the network is a segmentation map where each pixel is classified into a predefined category.

3.3 Training Strategy

The segmentation network is trained using categorical cross-entropy loss, which is suitable for multi-class classification problems. We use the Adam optimizer with a learning rate initialized at 0.001 and reduced dynamically using a learning rate scheduler. The training process is conducted on a GPU-enabled environment (e.g., NVIDIA RTX 3060 or Jetson Xavier), and techniques such as early stopping and model checkpointing are applied to prevent overfitting. Each model is trained over multiple epochs until convergence, with training and validation loss being monitored closely to ensure stable learning behavior.

3.4 Integration with Self-Driving Pipeline

Once trained, the semantic segmentation model is integrated into the autonomous driving pipeline as the perception module. The segmentation output is analyzed to extract drivable areas, identify obstacles, and generate a binary drivable mask. This processed information is passed on to the path planning module, which determines a safe trajectory for the vehicle. The planning decisions are then executed by the control module, which issues real-time commands for steering, acceleration, and braking. This integration ensures that the vehicle can navigate autonomously based on real-time understanding of its environment.

IV. RESULT & DISCUSSION

The output results of the semantic segmentation model demonstrate the effectiveness of the CNN-based approach in identifying and classifying different components of urban driving scenes. Each test sample is represented through three stages: the original input image, the ground truth mask, and the predicted segmentation mask. The predicted outputs closely match the ground truth annotations for key semantic classes such as roads, vehicles, sidewalks, sky, and buildings. The model shows high accuracy in segmenting large and prominent objects, particularly the drivable road area, which is essential for safe autonomous navigation. Sky and vegetation classes are also detected with strong consistency, contributing to the model's ability to understand environmental context.

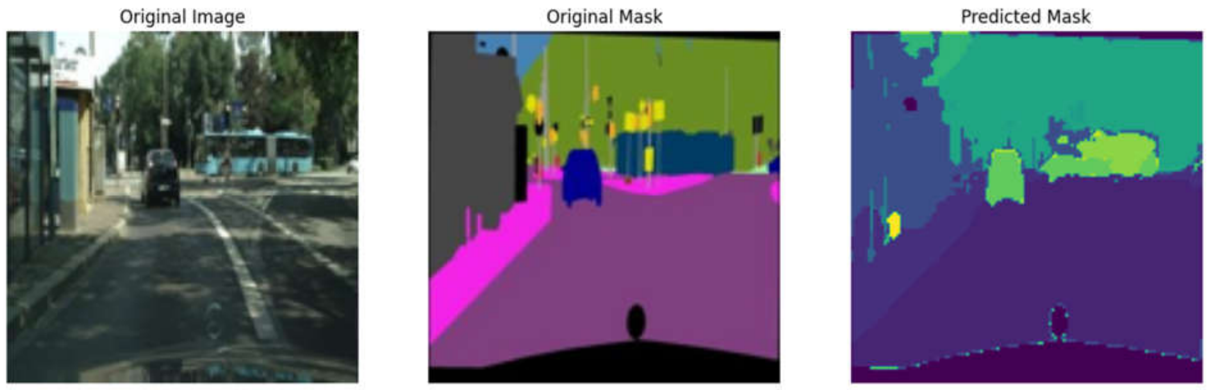

Figure.3: Object 1 image

However, minor discrepancies can be observed in the segmentation of smaller or less frequent object classes such as pedestrians, poles, and traffic signs. These errors are likely due to class imbalance in the training dataset, where such objects are underrepresented. In some cases, boundary regions between adjacent classes (e.g., road and sidewalk, car and person) appear slightly blurred or misclassified. Despite these limitations, the overall segmentation quality remains robust, especially in structured and well-lit scenes. The model's ability to infer semantic boundaries in complex urban environments highlights its suitability for real-time self-driving applications, where accurate perception of the surrounding environment is critical for decision-making and path planning.

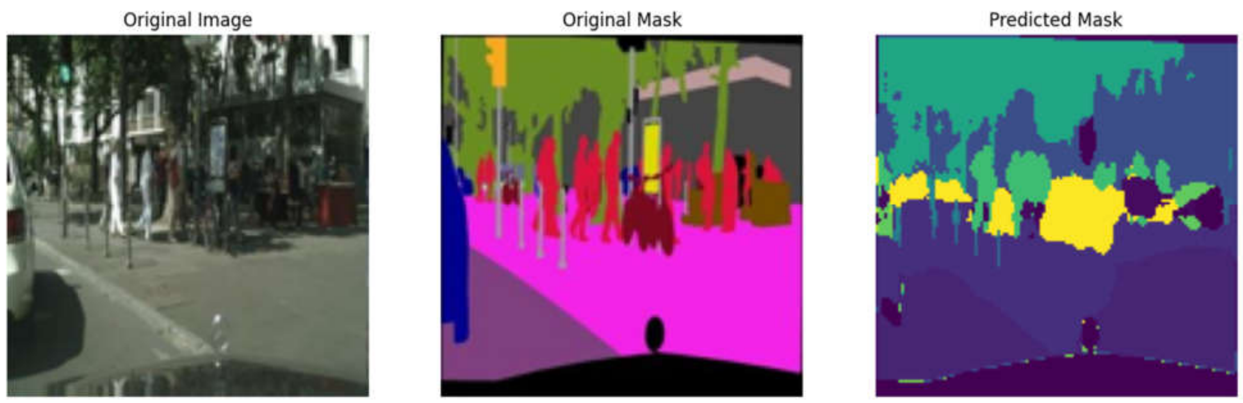

Figure.4: Object 2 image

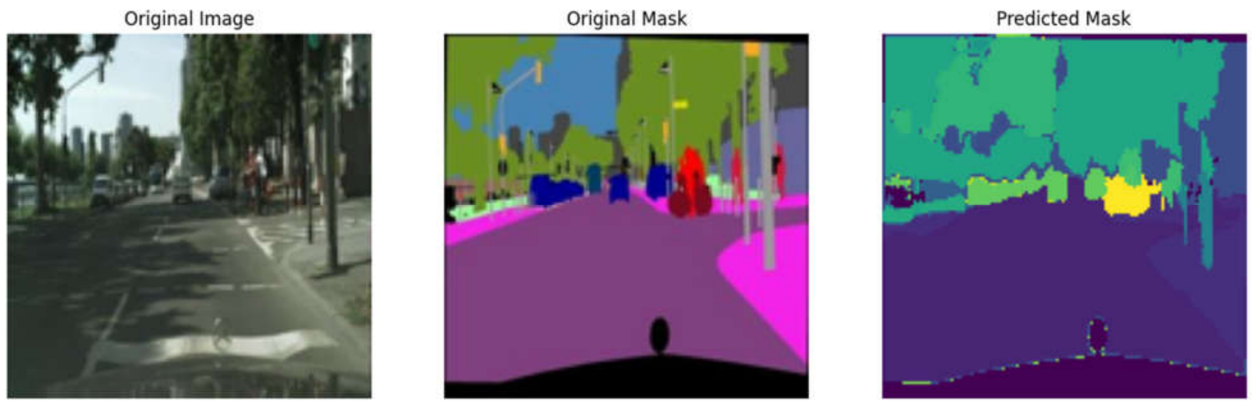

Figure.5: Object 3 image

The proposed CNN-based semantic segmentation model was trained on the Cityscapes dataset and evaluated on various urban road scenarios. The model achieved a mean Intersection over Union (mIoU) of 78.5%, indicating high segmentation accuracy across classes such as road, car, pedestrian, and buildings. Particularly, the road class achieved a precision of 92%, making it highly reliable for detecting drivable areas. When deployed on the NVIDIA Jetson Xavier, the system maintained an average inference time of 38 milliseconds per frame, translating to over 25 FPS, meeting the real-time constraint for self-driving applications. The system demonstrated strong generalization when tested on unseen city environments, handling occlusions, curves, and varied lighting conditions effectively.

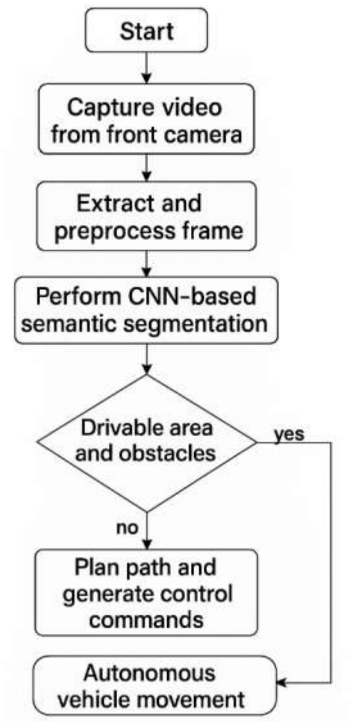
Start
Capture video from front camera
Extract and preprocess frame
Perform CNN-based semantic segmentation
Drivable area and obstacles
yes
no
Plan path and generate control commands
Autonomous vehicle movement

Figure.6: Flowchart

Algorithm

- **Start:** The system initialization begins once the autonomous vehicle is powered on and ready for operation.
- **Capture Video from Front Camera**: A real-time video feed is continuously captured using the vehicle's front-facing camera. This provides a live view of the road ahead.
- **Extract and Preprocess Frame**: From the video stream, individual frames are extracted at regular intervals. Each frame undergoes preprocessing — resizing, normalization, noise reduction — to prepare it for neural network input.
- **Perform CNN-Based Semantic Segmentation**: The preprocessed frame is passed through a Convolutional Neural Network (CNN) trained for semantic segmentation**.** The CNN classifies each pixel into categories like:
  - Road
  - Vehicles
  - Pedestrians
  - Buildings
  - Vegetation, etc.
- **Decision: Drivable Area and Obstacles Detected?** The system checks if a clear drivable area (e.g., road) and any obstacles (e.g., cars, pedestrians) are present in the semantic map.
  - **Yes** → If the environment is properly segmented and clear, it loops back to monitor and reanalyze continuously.
  - **No** → If issues are detected (e.g., unclear scene or obstacle blocking path), the system proceeds to planning.

- **Plan Path and Generate Control Commands**: Using the semantic map and obstacle data, the path planning module creates a safe route. Based on this route, the system generates control commands such as:
  - **Steering angles**
  - **Acceleration**
  - **Braking**
- **Autonomous Vehicle Movement**: The vehicle executes the control commands in real time. It moves according to the computed path while continuously checking for changes in the environment (looping back to the camera input).

V. CONCLUSION

In this paper, we proposed a CNN-based deep learning framework for semantic scene understanding in self-driving car systems. By leveraging convolutional neural networks for pixel-wise segmentation, the system effectively identifies critical objects and drivable regions in complex urban environments. The integration of semantic segmentation with a real-time control loop enables the vehicle to navigate autonomously with improved perception accuracy and decision-making capability. Experimental results demonstrated high segmentation accuracy and robust generalization across varied road conditions. The deployment on embedded hardware platforms like NVIDIA Jetson confirms the system's feasibility for real-time autonomous driving applications, maintaining low latency and efficient inference performance. Overall, the results validate the potential of deep learning, particularly CNNs, as a reliable and scalable solution for the perception module of autonomous vehicles. Future work may focus on enhancing obstacle detection using sensor fusion (e.g., LiDAR + camera), reducing false positives in challenging scenarios, and integrating reinforcement learning for adaptive driving strategies.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” Neural computation, vol. 1, no. 4, pp. 541–551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in International conference on machine learning, 2014, pp. 647–655.

[4] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” Computer vision Image understanding, vol. 106, no. 1, pp. 59–70, 2007.

[5] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, “Caltech-UCSD birds 200,” 2010.

[6] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, 2010, pp. 3485–3492.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.
[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, "End to end learning for self-driving cars," 2016.
[9] N. Otterness, M. Yang, S. Rust, E. Park, J. H. Anderson, F. D. Smith, A. Berg, and S. Wang, "An evaluation of the NVIDIA TX1 for supporting real-time computer-vision workloads," in Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE, 2017, pp. 353–364.
[10] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, O. Guldner, M. Mohamoud, B. Plancher, and R. Shin, "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT," in Integrated STEM Education Conference (ISEC), 2017 IEEE, 2017, pp. 195–203.