A Multimodal Perception-Driven Self Evolving Autonomous Ground Vehicle

Jamie Roche, Varuna De-Silva, and Ahmet Kondoz, Senior Member, IEEE

Abstract—Increasingly complex automated driving functions, specifically those associated with free space detection (FSD), are delegated to convolutional neural networks (CNNs). If the dataset used to train the network lacks diversity, modality, or sufficient quantities, the driver policy that controls the vehicle may induce safety risks. Although most autonomous ground vehicles (AGVs) perform well in structured surroundings, the need for human intervention significantly rises when presented with unstructured niche environments. To this end, we developed an AGV for seamless indoor and outdoor navigation to collect realistic multimodal data streams. We demonstrate one application of the AGV when applied to a self-evolving FSD framework that leverages online active machine-learning (ML) paradigms and sensor data fusion. In essence, the self-evolving AGV queries image data against a reliable data stream, ultrasound, before fusing the sensor data to improve robustness. We compare the proposed framework to one of the most prominent free space segmentation methods, DeepLabV3+ [1]. DeepLabV3+ [1] is a state-of-the-art semantic segmentation model composed of a CNN and an autodecoder. In consonance with the results, the proposed framework outperforms DeepLabV3+ [1]. The performance of the proposed framework is attributed to its ability to self-learn free space. This combination of online and active ML removes the need for large datasets typically required by a CNN. Moreover, this technique provides case-specific free space classifications based on the information gathered from the scenario at hand.

Index Terms—Active learning, assistive robots, autonomous ground vehicles (AGVs), autonomous vehicles, depth sensing, free space detection (FSD), online learning, sensor data fusion.

I. INTRODUCTION

INHERENT complexities and diverse environments prevent autonomous ground vehicle (AGV)s from being programmed with a fixed set of rules that govern the policy that controls them [2]. Therefore, AGV’s need to learn to make decisions independently based on the scenarios they face and the objects perceived. Only in this manner can adequate driver policy be derived, where the AGV self-evolves over time, depending on the encounters they make.

Regarded as one of the most fundamental elements of perception, free space detection (FSD) experiences a lack of research in unstructured niche environments [3]. In structured environments like carriageways, FSD has been extensively researched [4].

Traditionally free space is detected using color [5] or texture segmentation [6], deduced from stereovision-based obstacle detection [7], or a combination of both [8]. Recently, however, FSD has, for the most part, been delegated to the convolutional neural network (CNN) [1]. In principle, given unlimited data, CNN should be able to classify traversable space under any circumstances. However, there is a question about whether CNNs can access diverse data quantities to achieve this task in practice. While results with large datasets are impressive, on a wide range of potential representations where data are sparse or lack diversity, a CNN will always encounter difficulties [9].

This article presents an opensource experimental AGV for data gathering, sharing, and experimental validation of driverless vehicle technology. Our target is to provide access to multimodal data to enable researchers to test control algorithms on the prototype using a unified interface. Toward this end, we have developed an autonomous platform equipped with several sensors and real-time control through a high-performance computer and Web-enabled interface. As proof of concept, we demonstrate a self-evolving FSD framework that self-learns using a combination of online and active ML. Online learning is a supervised ML paradigm, where an agent learns data as they become available. Active learning is a semisupervised ML paradigm, where an agent queries a human oracle as the data become available. The proposed self-evolving FSD pipeline is shown in Fig. 1, while Fig. 2 shows the architecture for the autonomous platform.

The contribution of this research is as follows.

1) An experimental platform that can navigate both indoor and outdoor unstructured environments.

2) An extensive opensource dataset that is suitable for multiple avenues of research into autonomous vehicle technology.

3) A novel self-evolving FSD model that utilizes a robust sensor stream to self-learn traversable space.

We chose the semantic segmentation of free space to demonstrate the autonomous platform’s functionalities as a research utility. To that end, we have organized this article as follows: Section II provides an overview of recent developments in FSD. Section III describes the experimental setup.

Manuscript received 16 June 2020; revised 22 March 2021 and 1 July 2021; accepted 8 September 2021. Date of publication 8 October 2021; date of current version 18 August 2022. This work was supported by the Engineering and Physical Sciences Research Council of the United Kingdom through MIMI+: Multimodal Imitation Learning in Multiagent Environments under Grant EP/T000783/1. This article was recommended by Associate Editor W. X. Zheng. (Corresponding author: Jamie Roche.)

Jamie Roche is with the Institute of Technology Sligo and the Institute for Digital Technologies, Loughborough University, London E15 2GZ, U.K. (e-mail: roche.jamie@itsligo.ie).

Varuna De-Silva and Ahmet Kondoz are with the Institute for Digital Technologies, Loughborough University, London E20 3BS, U.K.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3113804.

Digital Object Identifier 10.1109/TCYB.2021.3113804
Section IV describes a self-evolving FSD framework for the AGV. Section V presents the experimental results, followed by a discussion in Section VI before concluding this article in Section VII.

II. RELATED WORK

For the most part, people involved in teaching machines to segment traversable space have relied on a combination of camera sensors as in [10] and [11], radar-based FSD as in [12] and [13], and fusion-based FSD using a camera, light detection and ranging (LiDAR), and/or Radar [14], [15]. Generally, researchers use a combination of these data-gathering devices to generate some representation before making a classification.

In the most basic form, FSD algorithms solve problems using two different steps: 1) preprocessing and 2) classification [16]. The preprocessing stage can be as simple as a threshold technique using a texture-based analysis [6]. This sort of preprocessing is generally used to represent the data captured by the sensor. It can help reduce noise or mitigate issues related to shadows. In other cases, the image is transformed to produce a birds-eye view, generating an OGM representing the free space in the image [17].

The final step of most contemporary FSD pipelines is to extract features from the preprocessed image before sorting the space into the associated class. Features are derived from the pixels in the image, using different descriptors such as histogram of orientated gradient (HOG) or Hue saturation variance (HSV). By and large, this process of learning the features has been delegated to a CNN [18]–[20]. Generally, when using supervised ML paradigms for image processing, a CNN obtains superior results compared to traditional ML algorithms. While some have succeeded in using unsupervised ML methods to distinguish between lanes and traversable surfaces [21], most use a CNN. For example, CNNs are prevalent in tasks, such as road boundary detection [22], lane detection [23], and semantic segmentation [24].

More recently, researchers in [25] presented a pooling module using a pyramid structure to aggregate background data. The module links a CNN’s feature map to the output of the unsampled layer [26]. In addition to the unusual pooling module structure, Zhao et al. [25] reported a new loss function to solve mismatched relationships, confusing categories, and inconspicuous classes.

In [27], researchers reported on a dense upsampling convolution (DUC) network and a hybrid dilated convolution (HDC) network. Both the DUC and HDC networks solved upsampling and dilated convolution problems by dividing the label map into a section with the same size as the input feature map. Expanding on the DUC and HDC networks used in ResNet [26], DeepLabV3+ [1] combines an encoder–decoder with atrous separable convolution layers to semantically segment free space.

Atrous convolution is a formidable tool used to control the resolution of features computed by a CNN. Achieved by adjusting the filter’s field-of-view, atrous convolution facilitates the capture of multiscale information and generalizes the standard convolution operation. Using atrous separable convolution layers DeepLabV3+ [1], an incremental extension on its predecessors, reported one of the lowest mean intersection over union (IoU) when compared to other semantic segmentation networks [1].

From the review material, we understand that most contemporary FSD algorithms classify pixels using large datasets to learn the features that describe traversable surfaces. Most of these techniques do so in a supervised manner. Although they have demonstrated relatively high accuracy, there are difficulties in classifying surfaces when data is lacking. Therefore, FSD needs to work with little or no data in all environments under all lighting conditions and all surface types. In addition, it would be prudent for FSD algorithms to learn by querying image data against data from a reliable sensor stream, such as ultrasound or LiDAR. This approach should borrow from both supervised and semisupervised ML paradigms and take advantage of sensor data fusion. Given the diverse scenarios within which an autonomous platform will operate, the algorithm needs to be robust, self-calibrate, and function in all environments with relative ease.

III. METHOD

The technology and sensors integral to vehicle autonomy already impact the way humans drive. While it is possible to distinguish between the different systems—AGV or intelligent mobile platform—they are all robots [28]. To that end, we use the terms interchangeably throughout this research. This section is organized into three sections: 1) the platform specifications; 2) the experimental setup; and 3) the collected data to validate the proposed framework.

A. Platform Specifications

The autonomous platform architecture comprises stackable layers, consisting of a Sensing layer, a Data Analysis layer, a Multilayered Context Representation, and an Application layer. The platform utilizes seven different sensors, including cameras, LiDAR, Radar, and an ultrasonic sensor array. The inertial measurement unit (IMU) logs speed, power, and
orientation using a rotary encoder, a voltage divider, and a Magnetometer, respectively. A more extensive description of the autonomous platform and the design’s motivation can be found in [29]. Fig. 2 shows the architecture of the autonomous platform. The platform collects data autonomously with the possibility of human intervention. While human intervention is not desired, it was a prerequisite of the license.

Furthermore, when designing the autonomous platform, there was some discussion about implementing reinforcement learning. While reinforcement learning is beyond this article’s scope, it is regarded by some as an essential contribution to intelligent mobility [30].

Table I summarizes the optical sensor used during the development of the self-evolving FSD algorithm. While 360Fly and Wansview cameras also collected data, they are not detailed in Table I as they were considered outside the scope of this research.

The Theta V 360° camera has two fisheye lenses: 1) front and 2) rear-facing (360°). Cameras with a wide-angle lens were chosen to increase the possibility of capturing the entire scene into a frame. In addition, because the autonomous platform was designed to operate indoors and outdoors, space manipulation was crucial to gathering context information about the scene. Therefore, using a camera with a standard prime, zoom, macro, or telephoto lens would fail to capture context over various scenarios.

Table II summarizes the proximity sensor used during these experiments. While LiDAR and Radar also collected range data, they were not detailed in Table II as they were considered outside the scope of this research. The ultrasonic sensor array represents data on a single plane (2-D) with a horizontal field of view (HFOV) of 90°, a vertical field of view (VFOV) of 30°, and a max range of 5 m. The ultrasonic array supports multiple channel data streams and takes six measurements per second utilizing six emitter/detector pairs with a spatial resolution of 20°. The collision avoidance policy is implemented depending on the proximity of objects relative to the platform. Fig. 3 shows the conditions and reactions that the platform makes depending on an obstacle’s location. The ultrasonic sensor array consists of six HC-SR04 sensors positioned at 5°, 25°, and 45° on either side of the centerline. Objects within range are logged and acted upon, influencing the platform driver policy, as depicted in Fig. 3.

**B. Experimental Setup**

Fig. 4 shows the autonomous platform indicating the location of the proximity and optical sensors. Except for the
Fig. 4. Loughborough London autonomous platform showing the location of the range and optical sensors used to collect the LboroLdnAV dataset reported in the supplementary data.

Fig. 5. Platform setup indicating each sensor’s location relative to the ground and the platform’s front axle. Except for the Wansview cameras, all sensors were on the centerline.

global positioning satellite (GPS), all the sensors used in these experiments are commonly found in AV technology [31], [32]. The metric dimensions of the sensor position relative to the ground plane and front axle of the autonomous platform are presented in Fig. 5. All sensors were positioned along the platform’s center, except for the Wansview cameras, which were positioned 15 cm on either side of the centerline.

C. Data Collection

The self-evolving AGV requires a specific type of dataset so it can self-learn free space. One that can fulfill the requirements of multimodality while providing optical and range data from at least two sensor streams of a known location. As of 30 January, 2020, the Loughborough London autonomous vehicle (LboroLdnAV) dataset consists of 45.6 h of Video, LiDAR, and Ultrasound data collected over 1.2 km, under a variety of scenarios from unstructured indoor and outdoor environments.

The dataset comprises 2.5 million frames captured by four cameras; 672k frames captured by the 360Fly Wide-angled Camera; 1.2 million frames captured by the Ricoh Theta V 360° Camera, and 624k frames captured by the two Wansview IP cameras. Both the LiDAR and Ultrasonic sensor array captured a total of 252k and 220k scans, respectively. The Radar was not used during this data collection period.

Data collection is an ongoing project to assist in developing multimodal ML algorithms for use in intelligent mobility. Not all sensors were in use during the data collection period. Likewise, not all sensors were used during the development of the self-evolving FSD framework. The release reported in this article is the first part of a more extensive project covering data collected on the Queen Elizabeth Olympic Park. The project has since been expanded to include unstructured outdoor environments in Sri Lanka. Although the data collection lasted for extended periods, it was impossible to collect data every day due to the management company’s restrictions. The LboroLdnAV dataset is detailed in the supplementary data.

IV. SELF-EVOLVING FRAMEWORK FOR AGV

Many of the problems faced by cybernetics systems are complex and difficult to solve [33], [34]. For problems where it is impossible to apply heuristic ML algorithms, self-evolving algorithms are a sensible choice [35]. Inspired by natural selection, a self-evolving algorithm simulates evolution to solve complex real-world problems like identifying traversable space [34], [36]. We demonstrate this and the functionalities of the platform with a self-evolving FSD algorithm. Mimicking biological evolution, the proposed framework adapts to its environment using active ML. Typically, active ML methods consist of three parts: 1) identification of outliers; 2) human intervention feedback; and 3) model update [37]. With this in mind, we define a self-evolving framework as one that eliminates the need for human intervention and uses a reliable sensor stream to self-label data as they become available.

The self-evolving FSD framework that drives the AGV can be broken down into three components. The three components are depicted in Algorithm Component 1, Algorithm Component 2 and Algorithm Component 3. Using this approach improves the framework’s ability to recognize free space with little information to start while also allowing the framework to perform the retraining process on board and update the kernel function automatically.

When this process is used in conjunction with sensor fusion, it returns a case-specific result to space just encountered by the autonomous platform. Although it is possible to use any sensor that generates range data, we chose ultrasound because of its short-range reliability. It is also possible to use an alternative ML method, such as a neural network. However, the time to train an alternative ML method needs to be considered when the practical application of learning on the go is considered.

A. Image-Based FSD Component

The first component of the proposed framework is an image-based classifier using the SVM. The SVM is trained on a small amount of both HOG and HSV patches. Each patch set is assigned to a class as free space or not free space. The pseudocode for the first component of the self-evolving FSD framework is displayed in Algorithm Component 1. 1500-pixel patches of size $8 \times 8$ are collected from the CamVid dataset.
Algorithm Component 1 Image-Based FSD

Input: x and y loaded with labelled training data and the input image.
Output: Image-Based FSD Prediction.

1: \( \text{Train the free space classifier()} \)
2: repeat
3: for all \( \{x_i, y_i\}, \{x_j, y_j\} \) do
4: \( \text{Optimize } \alpha_i \text{ and } \alpha_j \)
5: endfor
6: until no change in \( \alpha \) resource-constrained criteria met
7: \( \text{import image} \)
8: \( \text{for each pixel patch, do} \)
9: \( \text{Extract features from an image patch} \)
10: \( \text{Apply the free space Classifier()} \)
11: if free space Classifier() \( \neq +1 \) then
12: \( \text{The pixel patch is "free space."} \)
13: endif
14: endfor

Ensure: Re-train only the support vector when \( \alpha_i > 0 \)


The HOG features are calculated for every \( 4 \times 4 \) blocks within the \( 8 \times 8 \) patch, while HSV values are chosen for the entire patch. A Radial Basis function is used as the kernel for the SVM and \( K \)-fold cross-validation for model selection.

In this case, the SVM learns a basic understanding of free space from the original sample, randomly partitioned into ten equal-sized subsamples. Of the ten samples, a single subsample is retained for validation, with the remaining is used for training. In Algorithm Component 1, lines 1–6 describe training the SVM, and lines 7–14 describe the process of classifying the pixel patches in the image. Finally, these images are passed onto the second component to be queried against the ultrasound data.

B. Ultrasound-Based FSD Component

The second component of the proposed framework uses an OGMap generated from the ultrasonic sensor array and geometrically aligned to the image data. We use the ultrasound OGMap to label the camera data before adding it to the dataset and then finally retraining the classifier.

When geometrically aligning two modalities, we need to know the relative location of different sensors. A plan view of the autonomous platform and sensor setup is graphically illustrated in Fig. 6.

Fig. 7 shows a side elevation of the platform. Metric dimensions for the platform relative to the location of the individual sensors are presented in Fig. 5. For the process of geometric alignment, consider an object \( O \) at a range \( (a) \) of 140 cm from

the focal point of the ultrasonic sensor array. In this case, angle \( B \) for object \( O \) identified to the right front of the ultrasonic sensor array can be determined as \( B = 180^\circ - 45^\circ = 130^\circ \).

The longitudinal \( (c) \) and vertical \((c')\) distances between the Camera and ultrasonic sensor array are 45 and 65 cm, respectively. The ultrasonic sensor array is positioned \( (H_0) \) 33 cm above the ground, and \( a \) is the distance between the object \( O \) and the ultrasonic array. If the angle between vector \( a \) and \( c \) is \( B \), we can describe the distance \( b \) between the Camera and object \( O \), as

\[
b = \sqrt{a^2 + c^2 - 2 \times a \times c \times \cos(B)}.
\] (1)

In turn, angle \( A \) between vectors \( b \) and \( c \) can be described as

\[
A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2 \times b \times c}\right).
\] (2)

Knowing the azimuth angle \( A \) and range \( b \), we can use the resultants from (1) and (2) to solve for the range \((c')\) between the object \( O \) and the Camera

\[
c' = \sqrt{a^2 + b^2}.
\] (3)

From (3), we can calculate the corresponding elevation angle \( \Omega \) for the object \( O \) relative to the Camera, as per

\[
\Omega = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2 \times b \times c}\right).
\] (4)

The pseudocode for the second component of the self-evolving FSD framework is displayed in Algorithm Component 2. At this point, the camera data are queried against the aligned OGMap, before merging with the annotated data of the first component, increasing the size of the dataset by 150-pixel patches.

The merged data are passed onto the final component. This process is repeated each time the ultrasonic sensor array takes one full scan of the environment. This part of the framework aims to find the corresponding pixel in the camera output for each data point output from the ultrasonic sensor array. We assume that the Camera’s longitudinal axis and the ultrasonic sensor array are aligned; however, an offset can be accounted for should it be required. When generating the OGMap, we use an inverse measurement model [38] using ultrasound and IMU data. The IMU gathers pose data in the form of coordinates \((x, y)\) and orientation \((\theta)\) from the rotary encoder, and
Algorithm Component 2 Ultrasonic-Based FSD Using OGMap
Input: Metric dimensions of sensor locations. Ultrasonic range and azimuth angle. Pose data from IMU. Image-based predictions from <Algorithm Component 1>.
Output: Newly labelled free space pixel patches
1: Import Image-Based FSD Prediction, ultrasound and IMU data
2: OG = occupancy_grid_map (US, IMU)
3: for each OG do
4: \( b = \sqrt{a^2 + c^2 - 2 \times a \times c \times \cos(B)} \)
5: \( a = \cos^{-1}\left(\frac{b^2 + c^2 - d^2}{2 \times b \times c}\right) \)
6: \( c' = \sqrt{d^2 + b^2} \)
7: \( \Omega = \cos^{-1}\left(\frac{a^2 + c'^2 - d^2}{2 \times a \times c'}\right) \)
8: \([\text{OG}_{\text{aligned}}] = \text{spherical_to_cartesian} (B, \Omega, c')\)
9: if \( \text{OG}_{\text{aligned}} = 1 \) then
10: The pixel patch is "free space."
11: endif
endfor

Function occupancy_grid_map \([l_{i-1,i}], x_i, z_i\):
12: for all cells \( m_i \) do
13: if \( m_i \) is in perceptual field of \( z_i \) then
14: \( l_{i,i} = l_{i-1,i} + \text{inverse_sensor_model} (m_i, x_i, z_i) - l_0 \)
15: endfor. return \( l_{i,i} \)

Function inverse_sensor_model \((i, x_i, z_i)\):
16: Let \( m_i \) be the Cartesian coordinates centre point
17: \( r \leftarrow \sqrt{(x_i - x)^2 + (y_i - y)^2} \)
18: \( \theta \leftarrow \tan^{-1}\left(\frac{y_i - y}{x_i - x}\right) \)
19: \( k \leftarrow \text{argmin} \{\theta - \theta_{j,\text{sens}}\} \)
20: if \( r > \text{min}(z_{\text{max}}^k, \frac{z_i}{\sin^2 \beta}) \) then \( \theta_{j,\text{sens}} > \beta \) return \( l_0 \)
21: if \( z_i \times \frac{r}{\sin^2 \beta} < \frac{z_{\text{max}}^k}{\sin^2 \beta} \) return \( l_{\text{free}} \)
endfor

Magnetometer, respectively
\[
l_0 = \log \frac{p(m_l = 0)}{p(m_l = 1)} = \log \frac{p(m_l)}{1 - p(m_l)} \quad (5)
\]
\[
l_{i,i} = \log \frac{p(m_{i|z_{1:i}, x_{1:i}})}{1 - p(m_{i|z_{1:i}, x_{1:i}})} \quad (6)
\]

In Algorithm Component 2, lines 1–10, we call the OGMap function before geometrically aligning it to the predictions generated in the first component. In Algorithm Component 2, lines 10–23, we apply the inverse model [38] using the beam index \( k \) and the range \( r \) for the center cell \( m_i \). The thickness of the obstacle and the width of the sensor beam are represented as \( \eta \) and \( \beta \), and the logarithm of the ratio of probabilities, frequently called log-odds ratio \( l_0 \) and \( l_{i,i} \) are defined in (5) and (6), respectively [38].

C. Self-Evolving FSD Component

Contextual information is vital for accurate path planning in dynamic environments [39]. Therefore, information on all sides of an autonomous platform carries equal importance for scene interpretation and autonomous navigation [40]. Although modern-range sensors, such as LiDAR, can provide a broad FoV over an extended range, the FSD framework uses low-cost ultrasonic sensors with a short-range FoV. While there have been ample contributions to the field of grid mapping using range sensors [41]–[43], these techniques cannot derive contextual information from range measurement in the environment to interpret the scene adequately. Unlike range data, image data is rich in context, providing a large amount of information over a broad area. Most contemporary FSD techniques focus on CNN and large quantities of data [18], [44]. When annotated data lack, a CNN will not generalize adequately, overfit the model, and misclassifying traversable space.

The pseudocode for the final component of the self-evolving FSD framework is displayed in Algorithm Component 3. The final component of the proposed framework fuses the ultrasound OGMap with the prediction of the retrained classifier. Retraining the classifier occurs each time the ultrasonic sensor array makes a new scan, doing so in the same manner as in the first algorithm component of the framework.

Outside the FoV of the ultrasonic sensor array, the algorithm cannot query new data. However, inside the FoV of the ultrasonic sensor array, the algorithm uses the fused data to self-learn and make a more conservative prediction about traversable space. It is important to note that the longer the algorithm is in operation, the better it becomes at predicting a paches class. At each point when the framework retrains, any knowledge the algorithm has learned is lost in the retraining process, catastrophically forgetting all it has learned. This makes for a classifier that can detect free space easily but is not cost effective on resources.

Retraining of the SVM with the new dataset occurs in Algorithm Component 3, lines 1–6. In Algorithm Component 3, lines 7–14, we apply the classifier before fusing data streams between lines 17 and 26. While this process does away with calibration, this geometric alignment method cannot be considered 100% accurate. Even if it works very well, the sensor assembly’s imperfections and variations in lens manufacturing processes can cause the sensor to deviate from the ideal geometry. Finally, it should be noted that the resolution of the ultrasonic sensor array is lower than that of the Camera. Although the resolution could be increased, it will never meet that of camera data, and therefore is an intrinsic shortcoming of the process.
V. EXPERIMENTAL RESULTS

This section is organized into three parts. The first section summarizes the data used during training and validation of the framework. The second and third sections report on the comparative performance of the framework driving the AGV.

A. Dataset

To test the ability of the proposed self-evolving FSD framework and DeepLabV3+ [1] to generalize, we trained them using the CamVid dataset [45] and tested them using the LboroLdnAV dataset. The CamVid database was collected using a Panasonic HVX200 camera mounted to the dashboard of a vehicle driven for 2 h around Cambridge. Data was collected at 30 fps with a resolution of 960 × 720 pixels using a standard prime lens. The CamVid dataset provides ground truth labels that associate each pixel with 32 semantic classes from 701 semantically labeled images [45].

We used a pretrained ResNet-18 [26] to initialize the weights of DeepLabV3+ [1]. We reduced the number of classes in the CamVid dataset from 32 classes into two superclasses. For example, Free Space is combined from Sidewalk, Road, Road Shoulder, Drivable Lane Markings, and Nondrivable Lane Markings. The remaining classes were grouped into the superclass “Not Free Space.” For the proposed self-evolving FSD framework, a small subset of 1500 pixel patches was used to train the classifier to start. After that, the framework self-learns as data become available. Testing the proposed self-evolving FSD framework and DeepLabV3+ [1] was done using an LboroLdnAV dataset.

The supplementary data appended to this article details the dataset used during the experiments. The data used for evaluating the different frameworks were not part of the data used to train the proposed framework.

Testing was done in this manner to demonstrate the generalizability of both frameworks. This subset of the dataset indicates multiple different surface types from indoor and outdoor unstructured environments.

B. Comparative Performance of FSD Frameworks

Comparative evaluation describes a mechanism where the proposed framework’s performance is evaluated using a set of metrics. It is not always clear how to do this, as what works for one system will not necessarily work for another. Typically, in semantic segmentation tasks, the accuracy, the fScore, and the weighted IoU suffice as metrics used to measure performance. When scrutinizing FSD frameworks, two ways of thinking regarding metrics should be kept in mind: 1) how well the classifier works on the test data or dataset metrics and 2) how well the classifier works on the individual class or class metrics. To that end, we report the performance of both the dataset and the class metrics of the proposed self-evolving FSD framework and DeepLabV3+ [1].

Dataset metrics describe metrics that rank the response of the proposed framework to the test data. They aggregate the algorithm’s response and provide detail as to how well the framework performs. The class metrics indicate the response of the framework to specific classes. While dataset and class metrics show different things, they both utilize similar techniques. For example, the global average indicates the percentage of correctly identified pixels for each class. Defined as the ratio of correctly classified pixels to the total number of pixels in that class. For the aggregate dataset, the mean accuracy is the average accuracy of all classes in all images. Consequently, class accuracy is typically used in conjunction with IoU for a complete evaluation of the segmentation results.

The IoU is the most used metric in semantic segmentation and object detection. For each class, IoU is the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in that class. For the entire dataset, the mean IoU is the average IoU score of all classes in all images. Consequently, class accuracy is typically used in conjunction with IoU for a complete evaluation of the segmentation results.
false positives. This metric is used if the ratio of classes is disproportionately sized.

Like accuracy, the bfScore or boundary F1 Score considers both the precision and the recall of the classifier to determine the advantage of one system over another. Typically, the bfScore is a metric that correlates better with human qualitative assessment than the IoU. For each class, the mean bfScore is the average bfScore of that class in all images. For the aggregate data set, the mean bfScore is the average bfScore of all classes in all images.

1) Performance of the Self-Evolving FSD Framework: Table III reports on the global average, mean accuracy, mean IoU, weighted IoU, and mean bfScore for the proposed online active self-evolving FSD framework.

These metrics report on the response of the framework to all the test data. In this case, the proposed framework performs quite well for most metrics and reasonably well for the mean bfScore.

Table IV reports on the accuracy, IoU, and mean bfScore for the proposed self-evolving FSD framework. These metrics report on the response of the self-evolving FSD framework to the individual classes in the dataset. Interestingly, when considering the individual class metrics, the proposed self-evolving FSD framework performs better on the “not free space” class compared to the “free space” class. Overall, the metrics in Tables III and IV and the confusion matrix in Fig. 9(a) show that the proposed framework generalizes exceptionally well to environments never encountered.

2) Performance of the DeepLabV3+ FSD Framework: Table V reports the global average, mean accuracy, mean IoU, weighted IoU, and mean bfScore for the DeepLabV3+ framework.

These metrics report on the response of DeepLabV3+ [1] to all the test data. It should be noted the dataset metrics for DeepLabV3+ [1] lagged behind the proposed FSD framework.

Table VI reports the accuracy, IoU, and mean bfScore for the DeepLabV3+ framework. These metrics report on the response of DeepLabV3+ [1] to the individual classes in the dataset. Similar to the proposed self-evolving FSD framework, DeepLabV3+ [1] reports a higher error on the “free space” class than the “not free space” class. It should be noted that this is a reoccurring event for both frameworks. Fig. 9(b) shows the confusion matrix for the DeepLabV3+ [1]. When comparing class metrics, dataset metrics, and the confusion matrix, the proposed self-evolving FSD model outperforms DeepLabV3+ [1]. While DeepLabV3+ [1] performs relatively well, it lags the online active ML method for generalizing.

3) Visual Comparison of FSD Frameworks: We benchmarked the proposed self-evolving FSD framework against DeepLabV3+ [1]. In consonance with the results presented in Fig. 10 Scenario 1 (a), 2 (a), 3 (a), and 4 (a), the proposed self-evolving FSD framework is better at detecting free space than DeepLabV3+ [1]. While DeepLabV3+ [1] is capable, there are several misclassifications. For example, in Fig. 10 Scenario 1 (a), the area in front of the platform is correctly classified, whereas in Fig. 10 Scenario 1 (b), DeepLabV3+ [1] misclassifies the area as occupied space. This corresponds to a situation where DeepLabV3+ [1] has not generalized from the training data.

Again, in Fig. 10 Scenario 2 (b), DeepLabV3+ [1] fails to detect a portion of free space to the front of the platform. Whereas in Fig. 10 Scenario 2 (a), the proposed framework outperforms DeepLabV3+ [1] and accurately classifies the area to the front of the platform. Interestingly, the proposed framework performs poorly on the concrete paving on either side of the autonomous platform. Conversely,
DeepLabV3+ [1] classifies the paving stones correctly but mistakenly classifies the sky as free space.

Almost a repeat of the results is found in Fig. 10 Scenario 2 (b), in Fig. 10 Scenario 3 (b), DeepLabV3+ [1] miss classifies the area to the front and correctly classifies the area on either side of the platform. Conversely, in Fig. 10 Scenario 3 (a), the self-evolving FSD framework outperforms DeepLabV3+ [1] except for a small area to the right front of the autonomous platform. Likewise, in Fig. 10 Scenario 4 (a), we have a similar situation where the proposed framework classifies almost all the free space in the image correctly. Conversely, in Fig. 10 Scenario 4 (b), DeepLabV3+ [1] misclassifies the area to the platform’s front and side.

VI. DISCUSSION

The self-evolving AGV relies on an ultrasonic sensor array to self-label camera data as they become available. The objective is to identify traversable space with little or no training data to start. The comparative framework, DeepLabV3+ [1], uses many training images to identify patterns in the data that indicate the different classes in the dataset. If the dataset used to train DeepLabV3+ [1] is annotated with two classes (Free Space and Not Free Space), it will only semantically segment these elements in an image. Since FSD can be regarded as the most fundamental element of perception, the need to identify additional classes can be regarded as unnecessary for autonomous navigation. [4].

Not being able to generalize is an issue relating to the dataset. Herein lies the problem; if the data used to train a CNN are lacking, the network will not generalize. Since the data needed to train a network adequately vary immensely, having a large and diverse dataset is challenging. Although CNN has attained high degrees of accuracy, there will always be difficulties when data lack. Therefore, it makes sense to branch out to a ML paradigm that can self-learn. Only when machines are learning from the data they are presented with, will they be able to handle safety-critical decisions alone.

The framework’s fundamental principle is the querying of optical data against the robust sensor stream as they become available. In effect, the framework classifies on a case-by-case basis, relearning a new understanding of free space each time. Consequently, any free space knowledge it has gained is lost each time retraining occurs. This phenomenon is known as Catastrophic forgetting and is a common issue with online ML.

Furthermore, the proposed framework requires an ML method that can retrain quickly when the practical applications of retraining an algorithm on the go are considered. Regardless, this form of self-evolving FSD will outperform a supervised ML paradigm when data are insufficient. This is evident when empirical metrics are considered.

It is important to note that both frameworks were trained on data captured by a camera using a stand prime lens and tested on data captured by a camera using a wide-angle lens. Although some arguments regarding the underperformance of DeepLabV3+ [1] could rotate around the use of wide-angle lenses during testing, it can be disregarded since DeepLabV3+ [1] performs poorly at the center of the image.

The center of images captured by a camera using a wide-angle lens resemble the data captured by a camera
using a standard prime lens. Unlike a standard prime lens, information toward the perimeter of an image captured using a wide-angle lens is distorted. This is important to note when using supervised ML methods like DeepLabV3+ [1] that rely on identifying the features. When an image is distorted, a supervised ML method will have difficulty identifying the features it has been trained to detect. Unfortunately, the self-evolving FSD framework requires a specific type of dataset to fulfill the multimodality requirements while providing optical and range data from at least two sensor streams of a known location. None of the datasets we reviewed for this research fulfilled these requirements. Consequently, the comparison between both frameworks could be regarded by some as unfair. However, due to the proposed framework’s constraints, we could not identify a superior means of comparative evaluation.

VII. CONCLUSION

This research presented an opensource experimental framework for data gathering, sharing, and experimental validation of driverless vehicle technology. The primary objective is to provide access to a multimodal dataset and facilitate the development and testing of ML algorithms for AGVs. As a use case, we demonstrated a self-evolving FSD framework that self-learns using a combination of online and active ML. We chose online ML over other ML paradigms, like incremental ML, because it processes data as they become available in a sequential order. The advantages of doing so are prevalent in the metrics we reported on and show how the self-evolving FSD algorithm can outperform a static state-of-the-art deep learning segmentation algorithm.

This research’s implications are a multimodal perception-driven, self-evolving autonomous ground vehicle that can self-learn free space with virtually no data to start. As new surfaces present themselves, the AGV learns by querying a robust sensor stream. Thus, allowing the platform to work in all environments, under all lighting conditions, and on all surface types. We chose an SVM based on HOG/HSV features as a classifier because of the little time required for online training. While other ML methods such as deep neural networks also work. Training time was a major factor when considering the algorithm’s application. Should the time required to train such networks, or the computational cost reduce, it would be possible to train on the go and utilize a more capable ML method.

The planned future work includes the expansion of the sensor fusion framework to include multiple cameras and LiDAR. Combining these additional sensors with the ultrasonic sensor data will improve classification in both the near and midfield range. Of course, this will also increase the rate at which the dataset grows, which can be regarded as an intrinsic shortcoming. However, this also provides parallel research objectives, and should attempt to answer the question of what do we remember and why do we remember it? Answering these questions will allow us to optimize the computing costs required by the framework. After all, if we know what features we remember some things over others, we can decide what features to keep when retraining the system.

REFERENCES

[1] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 835–851.
[2] V. De Silvio, X. Wang, A. D. Aladaghi, A. M. Kondoz, and E. Ekmeckicoglu, “Agent-based modeling for driving policy learning in connected and autonomous vehicles,” in Proc. Intell. Syst., 2018, pp. 113–125.
[3] E. D. Farner, J. E. Ball, and A. C. Gurbuz, “Extending free-space mapping to unstructured, off-road environments,” in Proc. Auton. Syst. Sens. Process. Security Veh. Infrastruct., 2020, Art. no. 1141509.
[4] R. Fan et al., “Learning collision-free space detection from stereo images: Homography matrix brings better data augmentation,” IEEE/ASME Trans. Mechatronics, early access, Feb. 22, 2021, doi: 10.1109/TMECH.2021.3061077.
[5] J. D. Crisman and C. E. Thorpe, “UNSCARF-a color vision system for the detection of unstructured roads,” in Proc. IEEE Int. Conf. Robot. Autom., vol. 3, 1991, pp. 2496–2501.
[6] J. Zhang and H.-H. Nagel, “Texture-based segmentation of road images,” in Proc. Intell. Veh. Symp., 1994, pp. 260–265.
[7] A. Broggi, C. Caraffi, R. I. Fedriga, and P. Grisleri, “Obstacle detection with stereo vision for off-road vehicle navigation,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2005, p. 65.
[8] N. Soquet, D. Aubert, and N. Hautiere, “Road segmentation supported by an extended v-disparity algorithm for autonomous navigation,” in Proc. IEEE Intell. Veh. Symp., 2007, pp. 160–165.
[9] G. Marcus, “Deep learning: A critical appraisal,” 2018. [Online]. Available: arXiv:1801.00631.
[10] Y. Hallakov, H. Behnner, and S. Ilie, “Scene understanding from a moving camera for object detection and free space estimation,” in Proc. IEEE Intell. Veh. Symp., 2012, pp. 105–110.
[11] R. Grewe, A. Hohn, S. Hegemann, S. Lueke, and H. Winner, “Towards a generic and efficient environment model for ADAS,” in Proc. IEEE Intell. Veh. Symp., 2012, pp. 316–321.
[12] M. Schreier and V. Willert, “Robust free space detection in occupancy grid maps by methods of image analysis and dynamic B-spline contour tracking,” in Proc. IEEE Conf. Intell. Transp. Syst., 2012, pp. 514–521.
[13] C. Lundquist, T. B. Schön, and U. Orguner, “Estimation of the free space in front of a moving vehicle,” in Proc. IEEE Intell. Veh. Symp., 2009, p. 47.
[14] Q. Li, L. Chen, M. Li, S. L. Shaw, and A. Nüchter, “A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios,” IEEE Trans. Veh. Technol., vol. 63, no. 2, pp. 540–555, Feb. 2014.
[15] V. De-Silva, J. Roche, and A. Kondoz, “Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots,” Sensors, vol. 18, no. 8, p. 2730, 2018.
[16] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: A survey,” Mach. Vis. Appl., vol. 25, no. 3, pp. 727–745, 2014.
[17] M. Felisa and P. Zanì, “Robust monocular lane detection in urban environments,” in Proc. IEEE Intell. Veh. Symp., 2010, pp. 591–596.
[18] F. Pizzati and F. Garcia, “Enhanced free space detection in multiple lanes based on single CNN with scene identification,” in Proc. IEEE Intell. Veh. Symp., 2019, pp. 2536–2541.
[19] S. Tsutsui, T. Kerola, S. Saito, and D. J. Crandall, “Minimizing supervision for free-space segmentation,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, 2018, pp. 988–997.
[20] K. Gholias and I. E. Vlahogianni, “Convolutional neural networks for on-street parking space detection in urban networks,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 12, pp. 4318–4327, Dec. 2019.
[21] J. Liu, L. Lou, D. Huang, Y. Zheng, and W. Xia, “Lane detection based on straight line model and K-means clustering,” in Proc. IEEE Data Driven Control Learn. Syst. Conf., 2018, pp. 527–532.
[22] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, “Towards end-to-end lane detection: An instance segmentation approach,” in Proc. IEEE Intell. Veh. Symp., 2018, pp. 286–291.
[23] S. Lee et al., “VPGNNet: Vanishing point guided network for lane and road marking detection and recognition,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops, 2017, pp. 1965–1973.
[24] X. Zhao, Q. Zhang, D. Zhao, and Z. Pang, “Overview of image segmentation and its application on free space detection,” in Proc. IEEE Data Driven Control Learn. Syst. Conf., 2018, pp. 1164–1169.
[25] H. Zhao, J. Shi, X. Qin, X. Wang, and I. Jia, “Pyramid scene parsing network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6230–6239.
[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[27] P. Wang *et al.*, “Understanding convolution for semantic segmentation,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2018, pp. 1451–1460.

[28] S. Thrun, “Toward robotic cars,” *Commun. ACM*, vol. 53, no. 4, pp. 99–106, 2010.

[29] V. De-Silva, J. Roche, X. Shi, and A. Kondoz, “IoT driven ambient intelligence architecture for indoor intelligent mobility,” in *Proc. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (CyberSciTech)*, 2018, pp. 451–456.

[30] B. R. Kiran *et al.*, “Deep reinforcement learning for autonomous driving: A survey,” 2020. [Online]. Available: arXiv:2002.00444.

[31] G. Pandey, J. R. McBride, and R. M. Eastic, “Ford Campus vision and lidar data set,” Int. J. Rob. Res., vol. 30, no. 13, pp. 1543–1552, 2011.

[32] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[33] K. De Jong, *Evolutionary Computation: A Unified Approach*. Cambridge, MA, USA: MIT Press, 2006.

[34] A. M. Andrew, “Introduction to evolutionary computing.” *Kybernetes*, vol. 33, nos. 5–6, pp. 1064–1065, 2004.

[35] C.-F. Juang, C.-Y. Chou, and C.-T. Lin, “Navigation of a fuzzy-controlled wheeled robot through the combination of expert knowledge and data-driven multiobjective evolutionary learning,” *IEEE Trans. Cybern.*, early access, Jan. 5, 2021, doi: 10.1109/TCYB.2020.3041269.

[36] C. Borg, M. Rosner, and G. Pace, “Evolutionary algorithms for definition extraction,” in *Proc. 1st Workshop Definition Extraction*, 2009, pp. 26–32.

[37] A. Chakrabarty, C. Danielson, S. D. Cairano, and A. Raghunathan, “Active learning for estimating reachable sets for systems with unknown dynamics,” *IEEE Trans. Cybern.*, early access, Jul. 22, 2020, doi: 10.1109/TCYB.2020.3000966.

[38] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents Series). Cambridge, MA, USA: MIT Press, 2005.

[39] P. Zhou, W. Zhao, J. Li, A. Li, W. Du, and S. Wen, “Massive maritime path planning: A contextual online learning approach,” *IEEE Trans. Cybern.*, early access, Feb. 26, 2020, doi: 10.1109/TCYB.2019.2959543.

[40] J. E. Ha, J. B. Shim, and K. H. Choi, “Retrieval of contextual information on Korean road sign,” in *Proc. Int. Conf. Control Autom. Syst.*, 2008, p. 1481.

[41] S. K. A. Nair, S. Joladarashi, and N. Ganesh, “Evaluation of ultrasonic sensor in robot mapping,” in *Proc. Int. Conf. Trends Electron. Informat.*, 2019, pp. 638–641.

[42] S. Kodagoda, E. A. S. M. Hemachandra, P. G. Jayasekara, R. L. Peiris, A. C. De Silva, and R. Munasinghe, “Obstacle detection and map building with a rotating ultrasonic range sensor using Bayesian combination,” in *Proc. Int. Conf. Inf. Autom.*, 2006, pp. 98–103.

[43] S. C. Wei, Y. Yagi, and M. Yachida, “Online map building based on ultrasonic and image sensor,” in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 1996, pp. 1601–1605.

[44] S. Patra, P. Maheshwari, S. Yadav, S. Banerjee, and C. Arora, “A joint 3D-2D based method for free space detection on roads,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2018, pp. 643–652.

[45] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2009.