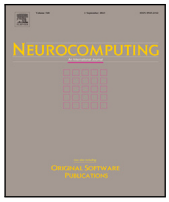
Neurocomputing 600 (2024) 128161

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Enhancing end-to-end control in autonomous driving through kinematic-infused and visual memory imitation learning

Sergio Paniego *, Roberto Calvo-Palomino, JoséMaría Cañas

Robotics Lab, Universidad Rey Juan Carlos, Madrid, Spain

ARTICLE INFO

Communicated by X. Gu

Keywords:
End-to-end autonomous driving
Imitation learning
Deep learning
Lane-following

ABSTRACT

This paper presents an exploration, study, and comparison of various alternatives to enhance the capabilities of an end-to-end control system for autonomous driving based on imitation learning by adding visual memory and kinematic input data to the deep learning architectures that govern the vehicle. The experimental comparison relies on fundamental error metrics (MAE, MSE) during the offline assessment, supplemented by several external complementary fine-grain metrics based on the behavior of the ego vehicle at several urban test scenarios in the CARLA reference simulator in the online evaluation. Our study focuses on a lane-following application using different urban scenario layouts and visual bird-eye-view input. The memory addition involves architectural modifications and different sensory input types. The kinematic data integration is managed with a modified input. The experiments encompass both typical driving scenarios and extreme never-seen conditions. Additionally, we conduct an ablation study examining various memory lengths and densities. We prove experimentally that incorporating visual memory capabilities and kinematic input data makes the driving system more robust and able to handle a wider range of challenging situations, including those not encountered during training, in terms of reduction of collisions and speed self-regulation, resulting in a 75% enhancement. All the work we present here, including model architectures, trained model weights, comparison tool, and the dataset, is open-source, facilitating replication and extension of our findings.

1. Introduction

Making cars and robots capable of driving by themselves has been a topic of interest in both research and industry for the past years and seems to be a topic that will have a wide impact on day-to-day life in the coming years too [1]. The potential benefits of autonomous driving cars and robots are enormous, including improved traffic safety and security, as well as more optimized mobility for individuals and globally.

Typically, this autonomy is divided into 6 levels, as described by the SAE J3016 Standard, from no automation (0) to full automation (5), where human intervention is not needed in any situation. Some current commercial solutions (e.g., Tesla, Waymo, Cruise, etc.) implement levels 2–3 and even some 4 autonomy capabilities, but most benefits come at levels 4 and 5, which still need more progress in research and industry. Some competitions are also helping to advance in this research topic, including DuckieTown [2], AWS DeepRacer [3], and F1TENTH [4].

Although research examples in autonomous driving appeared several years ago [5], interest in this field has risen significantly thanks to the latest advancements in deep learning, especially with the inclusion of specialized hardware (GPUs), the development of large open datasets [6], and specialized simulators like CARLA [7] or TORCS [8], and advancements in previous techniques, such as CNN [9]. Deep learning and artificial intelligence-based solutions help to improve the results of this perception and control problem, although their application area expands across multiple domains [10–12].

The methodologies for autonomous driving control solutions are usually divided between modular and end-to-end pipelines. The first one includes several modules [13] whereas the end-to-end [14–16] pipeline generates control decisions based on the direct understanding of the input. Vision-based end-to-end autonomous driving has gained significant traction in recent literature but still presents some limitations [17] and challenges [18]. It is an integral segment of accumulating exteroceptive information about the surrounding environment of the vehicle [19–21]. Using vision for end-to-end imitation learning [22] control of autonomous vehicles was proposed by NVIDIA's PilotNet [23] framework where the goal was to deliver steering control commands based on raw frontal images with the help of a CNN. This was further analyzed by augmenting fully connected layers to

* Corresponding author.
E-mail addresses: sergio.paniego@urjc.es (S. Paniego), roberto.calvo@urjc.es (R. Calvo-Palomino), josemaria.plaza@urjc.es (J. Cañas).

https://doi.org/10.1016/j.neucom.2024.128161
Received 12 January 2024; Received in revised form 21 June 2024; Accepted 29 June 2024
Available online 6 July 2024

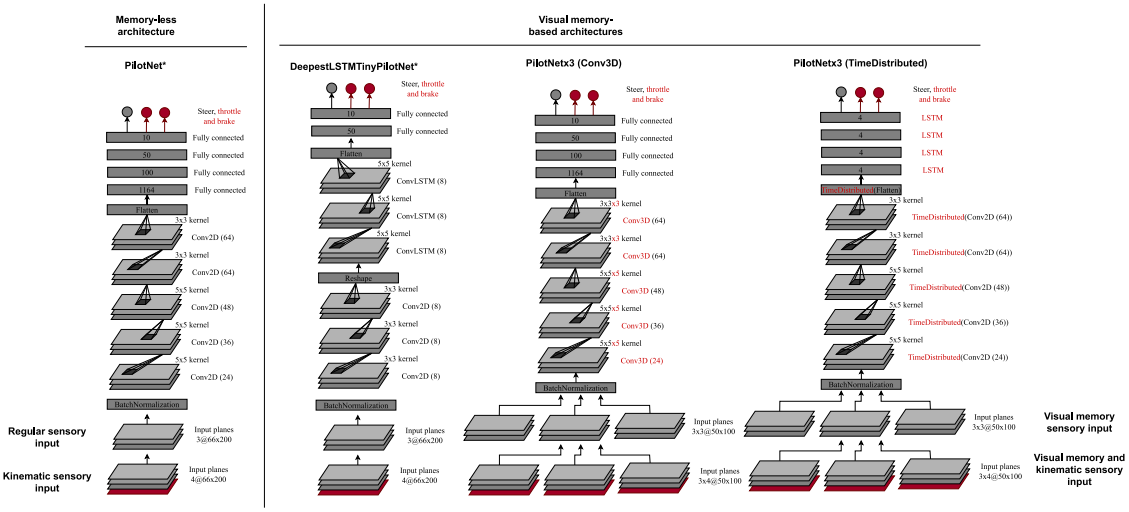


Fig. 1. Details of the deep learning architectures compared in this work. One of them is memory-less and the other three are visual memory-based. A variation of each of them also receives kinematic data input. Layers and input data marked in red are the modifications proposed in this work based on baseline architectures.

a CNN [24] to achieve performance comparable to a human driver. Simulated images have also been used by [25] to study the importance of road-related features in the images. Even input commands have been considered for managing vehicle behavior [26], as well as solutions based on reinforcement learning [27,28]. While all the above contributions considered utilizing the spatial features of the camera image inputs, [29] proposed the addition of temporal analysis using memory-based deep neural networks, examining the importance of LSTMs and convolutional layers with LSTMs respectively.

This use of memory-based solutions has also been explored previously in different research publications. For instance, in [30], researchers combined CNN and LSTM layers in their architecture, and in [31,32], they used analogous approaches combining ConvLSTM modules for capturing the temporal information of the data input for generating the control commands of the vehicle. Long Short-Term Memory networks (LSTMs) [33] are a special type of recurrent neural network (RNN), that are specialized in learning temporal dependencies that are present in the dataset. A variation of the LSTMs are convolutional LSTM (ConvLSTMs) [34], architectures that include convolutions and can learn these temporal dependencies from sequences of images. 3D convolutions (Conv3D) are another commonly used architectural layer when the temporal dependencies need to be learned.

When understanding and assessing the performance of different solutions for end-to-end imitation learning control of an autonomous car/robot, loss metrics are very important. Some commonly used metrics include mean absolute error (MAE) and mean squared error (MSE). These metrics show the similarity of the model outputs to the supervised outputs on a test set of the dataset extracted from the expert. In addition to these, it is also important to measure the actual performance of the model when driving the car in test scenarios as these systems ultimately need to operate in real-time on actual vehicles [35]. While it is useful for better understanding the system, it can be difficult to quantify. When considering urban scenarios, these external metrics could include whether the model commits a traffic violation (collisions or lane invasions) or the effective interval completed distance in a fixed experimental time-lapse. Previous research has extensively examined this question, yielding a plethora of metrics and evaluation techniques customized to distinct elements of autonomous driving systems and varying operational contexts [36–39].

There is a diverse range of datasets available for various tasks in autonomous driving. For instance, comma AI [40] and Udacity datasets [41] are dedicated to the lane-following problem and provide car speeds. Other datasets focused on visual perception for autonomous driving, such as BDD100K [42] or nuScenes [43], are also available. However, since the focus of this manuscript is lane-following in urban scenarios using CARLA simulator, we cannot utilize these datasets. In Section 2, we provide a detailed explanation of a newly generated dataset that utilizes expert data.

The research hypothesis for this paper is that integrating visual memory capabilities into the deep learning architecture and kinematic input data improves the quality of the generated robot control behavior. Specifically, it aims to enhance system robustness in previously unseen situations and improve speed self-regulation. We introduce several contributions.

- First, we explore and compare various alternatives to enhance end-to-end autonomous driving capabilities by adding visual memory and kinematic input data. These enhancements include deep learning architectural modifications and different sensory input types for the memory additions and sensory input modification for the kinematic data input.
- Second, we perform an offline evaluation of each approach using fundamental error metrics (MAE, MSE), proving that these metrics are not enough for a reliable comparison.
- Third, we conduct an online evaluation of each approach in regular conditions similar to the ones in the dataset using the state-of-the-art autonomous driving simulator CARLA in urban scenarios, targeting a follow-lane application. We employ Behavior Metrics [44], a software tool that facilitates the assessment of different approaches to end-to-end imitation learning, which we also describe.
- Fourth, we evaluate the models in never-seen situations to understand their generalization capabilities, demonstrating that models with visual memory and kinematic input present improvements in these settings and understanding how and how much the addition of memory and kinematic data enhances driving behavior and when it can have a significant impact.
- Fifth, we conduct ablation studies of the visual memory length and density to identify the optimal combinations.
- Sixth, we release all the materials including models, architectures, and datasets, as open-source, along with the comparison software.

2. Kinematic-infused and visual memory end-to-end control based on imitation learning

This section introduces the system developed, which consists of different deep learning architectures (see Fig. 1), some of them with inner memory capabilities. These architectures are trained using imitation

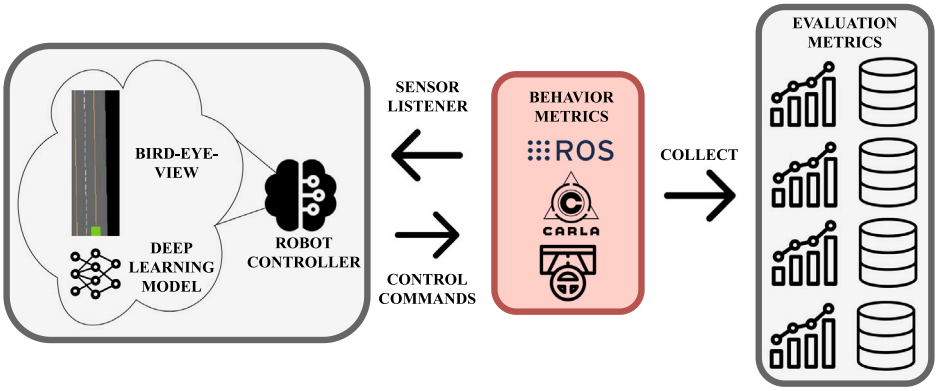


Fig. 2. End-to-end autonomous driving pipeline using Behavior Metrics software and a robot controller based on a deep learning model that controls the vehicle based on its sensory data.

learning for a lane-follow problem and with a range of sensory input data. We have used 8 deep learning architectures based on an end-to-end approach. They use as input at least a sensory image and generate motor control commands for an ego vehicle in a reactive control loop.

We focus our work on the implications in the final behavior of the robotic system of the addition of visual memory and kinematic data to the models. The perception data used as input is a simplified processed data from the sensory data. While these adjustments have been previously introduced in other studies, as detailed in Section 1, our current work focuses on studying and assessing their impact on the ultimate control behavior of the vehicle through comprehensive experimental validation. Instead of directly using the frontal camera of the car, for this work we used a bird-eye view of the scenario, removing part of the complexity that the system needs to perceive (shadows, weather, different textures...). The bird-eye view used is a segmented image including only the key components of the scene (see Fig. 2 for an example). In this case, we only need the car position information and the lanes that surround the vehicle. The approaches are trained and tested in the variety of towns that CARLA includes.

An imitation learning approach is used for training the neural architectures. In this approach, an expert agent drives along the towns while the sensory information and the behavior are recorded (behavior cloning). The sensory data in this scenario includes the bird-eye view, current vehicle speed (kinematic data), and the behavior includes normalized information about control commands (throttle, steer, and brake). Using the information extracted from the expert agent as supervised output for training, the final trained agent should mimic the behavior of the former, if the dataset is varied enough. To give the models an understanding of different situations, four urban environments were used for training the models: *Town01*, *Town03*, *Town05* and *Town07* (see Fig. 3). These towns include various common scenarios that a vehicle could be exposed to, including urban scenarios with different numbers of lanes, turn layouts, and road types like urban or highways. *Town02* is used for testing the trained models, as explained in depth in Section 4. The expert agent used is the rule-based autopilot included in the simulator, which can drive in different urban scenarios and has access to privileged simulation information. We used only one vehicle model to maintain a similar visual structure and physics behavior when driving.

We have explored four different deep learning architectures: one without memory, and three with visual memory. To augment our investigation, we have introduced additional kinematic input data to the baseline models, resulting in eight distinct models (see Fig. 1 for a detailed view of each architecture and group). We introduce modifications to two architectures (PilotNet and DeepestLSTMTinyPilotNet) from previous literature, while the remainder represents evolutionary developments from the baselines. The chosen deep learning architectures are characterized by their shallow, visual-based, end-to-end design, prioritizing simplicity and efficiency, as highlighted in prior research [45]. Our strategy for modifying these architectures is centered on preserving simplicity rather than embarking on a complete redesign to formulate entirely novel models.

2.1. Memory-less deep learning architecture

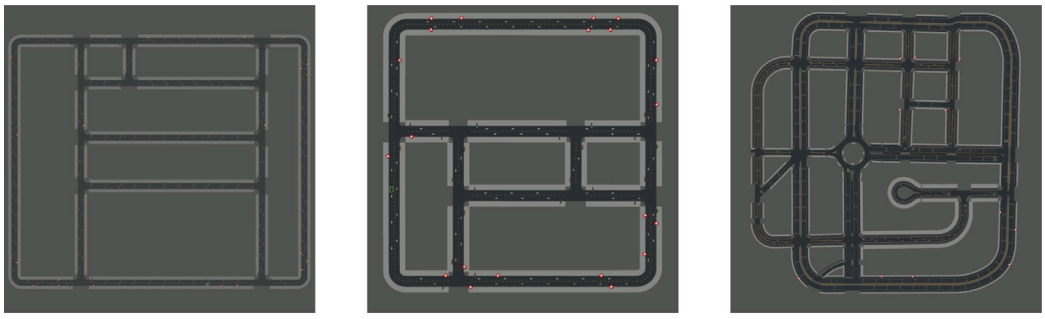
In the first group, we consider the architectures whose input only includes the visual sensory information at the current time, a single image, and no architectural modules that could be considered as memory, such as LSTM cells. We include here PilotNet*. PilotNet [46] was proposed in a previous work on end-to-end imitation learning for steering control. In this case, we have extended it to support throttling and braking (PilotNet*), considering that this information is also available from the expert agent. This architecture is specialized in understanding the context of an input sensory image and generating control commands for the ego vehicle.

- **PilotNet***: a powerful network that combines a convolutional backbone with some connected layers and an output of the control commands.

2.2. Deep learning architectures with visual memory

In the second group, we describe three architectures with visual memory. We include here DeepestLSTMTinyPilotNet* and two architectures especially created for this work. DeepestLSTMTinyPilotNet is an architecture that was proposed in a previous work on end-to-end imitation learning for steering control where they only used one image as input and two architectures created for this work that are extensions of PilotNet*. Again, we have extended them to support throttling and braking (DeepestLSTMTinyPilotNet*), considering that this information is also available from the expert agent. The two architectures created for this work, namely PilotNet*x3 (Conv3D) and PilotNet*x3 (TimeDistributed), receive visual sensory information from the current time instant and additionally, information from preceding instants. We study whether this additional information helps vehicle control in some scenarios or situations. We explore two different variations for extracting knowledge from the visual data input, *Conv3D* and *TimeDistributed*.

- **DeepestLSTMTinyPilotNet***: update of PilotNet architecture model making it smaller, reducing the number of convolutional and fully connected layers. It has some ConvLSTM layers that add some memory information. It only uses an image as input, instead of taking full advantage of the LSTMs modules using several images as input.



(a) Town01 (b) Town02 (c) Town03

(d) Town05 (e) Town07

Fig. 3. Set of urban environments in CARLA used.

- **PilotNet*x3 (Conv3D)**: based on PilotNet*. In this variation, the convolutional part is replaced by a 3D convolutional backbone that extracts the temporal information and understands the content of the visual data. The second part of the architecture maintains the PilotNet* structure and again generates the collection of control commands for the vehicle.
- **PilotNet*x3 (TimeDistributed)**: based on PilotNet*. In this variation, the model extracts information from the provided visual data from each image, combining the extracted features after the convolutional backbone and using LSTM modules for extracting the temporal information. The final fully connected layers included in PilotNet* are removed from this approach since the LSTM modules are enough for finally generating the control commands for the vehicle and understanding the global context.

2.3. Deep learning architectures with kinematic data as input

The previous four architectures have been also extended with an additional variation of sensory input using *kinematic data*. In this variation, the same architectures are used with a modification in the input sensory data, including information about the current ego vehicle velocity that we named kinematic data (e.g. 20 km/h). This exploration is motivated by the widespread availability and easy access of this data in vehicles. We also consider the high safety standards needed for an autonomous driving system and recognize the pivotal significance assigned to the system's speed in this matter. The additional velocity data is included as an additional channel to the visual data, without modifications to the described layers. The extra channel is uniformly filled with the normalized speed, scaled between 0 and 1. For the architectures with visual memory that receive more than one image as input, the ego vehicle speed is added as a new channel in the images, including the current velocity at the specific instant of each frame. We opt for this data input format to maintain the architecture under study without alterations, ensuring a fair comparison can be conducted.

2.4. Training

The training procedure varies slightly between the approaches considering their data structure, although we use the same amount of data for all the presented models. For the architectures considered memoryless, all the data collected from the expert is divided and shuffled as in a common machine learning workflow. We collected data at a rate of 20 images per second, so (t, $t-5$, and $t-10$), which is the input for models that receive more than one frame, is half a second. For the memory-based architectures, we first generated mini-sequences using a sliding window of three data points with temporal relationships before shuffling.

A collection of image data augmentations is included in the training procedure. We include modifications of brightness, contrast, gamma channel, hue saturation, PCA color augmentation, gaussian blur, and horizontal affine transformations. This augmentation is conducted using Albumentations [47]. We found horizontal affine augmentations to be important in the final behavior of the model and its generalization. Using imitation learning, the model can learn the behavior displayed in the dataset but struggles when the test data differs even a little bit from the training data distribution, as empirically proved in previous works [48]. With horizontal affine augmentation, the model can generalize better and provide good behavior in test scenarios that are slightly different from the rest of the data distribution. For example when approaching a turn a few centimeters away from the center of the lane (see Fig. 4). We use mean squared error as the loss function during training.

3. Measuring end-to-end imitation learning for robot control

For measuring the quality of robot behavior in autonomous driving, the common metrics used in machine learning are not enough to understand whether a model behavior is proficient or not. Typically, MSE or MAE are commonly utilized to calculate the loss of the model during training. They are good indicators, but not enough to assess robot application quality as they only measure the similarity of the

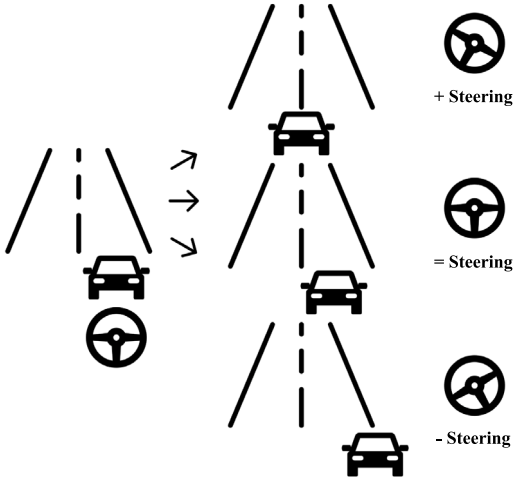


Fig. 4. Affine image date augmentation example. From the training example on the left, new examples are generated modifying the steering command accordingly.

model output and the supervised output at each instant. They do not take into account the future effects of control decisions, so a neural model with low loss may still exhibit poor robot behavior. In an end-to-end control system, previous decisions and the current vehicle situation play an important role in the next evolution of the situation. A previous inadequate decision a few seconds ago could lead to a very difficult situation now.

For assessment of such control systems, the external global holistic metrics do take into account such effects and others, and provide a more reliable indicator of the quality of the robot behavior. They are typically application-specific.

In addition to the common metrics with supervised data, we use as part of our experimental methodology a complete software tool for experimental validation of the models in simulation, called Behavior Metrics (see Fig. 2 for a view of its architecture), with a set of complementary global holistic metrics.

Behavior Metrics [49] is a software tool built on top of an autonomous driving simulator, in this case, CARLA, for running experiments and comparing different robot controllers fairly. It uses a ROS bridge for the communication between the tool and the simulator. It includes support for CARLA and Gazebo simulators. The tool allows the implementation of experiments in batches and easy comparisons of different robot controllers with the same metrics. Moreover, it facilitates assessment not only based on supervised data but also on holistic metrics. The robot controller is the master brain of the vehicle, which generates control commands based on the sensory input following an end-to-end approach. In this case, the sensory data is limited to bird-eye view images and kinematic states (vehicle speed). Inside the controller, a deep learning model, an explicitly programmed controller, or a reinforcement learning algorithm performs the decision-making. For this work, deep learning models control decision-making. The software provides the sensory data to the controller and manages the communication module that sends the control commands to the vehicle (throttle, steer, and brake).

The software tool runs the experiments in batches, so we have compared the different models in different urban environments and situations at the same time, with a common framework and in the same hardware. This configuration allows easy and fair comparison of the models with holistic metrics complementing the common machine learning metrics and the ones provided by the simulator itself.

Behavior Metrics supports the standard vehicle sensors compatible with CARLA. However, for this project, we have only utilized the bird-eye view sensor and the kinematic data (vehicle speed). We have also taken advantage of the varied range of simulation towns, vehicles, and settings that CARLA provides to test our models. The measurements we have taken include both common values provided by the simulator, such as collisions and lane invasions, as well as additional data that further illustrates the vehicle's behavior. This is particularly important, as the specific metrics used in the CARLA Autonomous Driving Leaderboard[1] do not suit our objectives. In the experiments described in Section 4, we have used some of the most relevant metrics, which are:

- **Effective completed distance**: distance navigated during the experiment that covers the town lanes. If the car drives outside a lane, that distance is not counted.
- **Position deviation mean per km**: positional deviation from the center of the lane per km, considering it to be the best possible trajectory for the vehicle.
- **Controller iterations frequency**: this number gives an intuition about the computation load of the robot controller and the system in general.
- **Collisions per km**: number of collisions per km of the vehicle with other elements in the environment.
- **Lane invasions per km**: number of lane invasions per traveled km.
- **Vehicle jerk in control commands per kilometer**: metric for understanding how much the control commands differ between time steps. If this number is low, it shows a smooth control behavior.
- **Vehicle jerk in velocity per kilometer**: metric for understanding how much the velocity changes per time step. It indicates whether the conduction is aggressive of smooth.
- **Average speed**: average speed driven by the ego vehicle.

4. Experiments

In this section, we present a series of experiments for the validation of the models and understanding of the implication of memory in the behavior of the robot in different situations. As described in previous sections, we have four different deep learning models trained on an imitation learning basis with two variations for each one. We use Behavior Metrics with CARLA as software for experimental validation of the models, along with the typical loss metrics discussed previously.

All the different experiments are easily reproducible, with the model's weights, architectures, and the tool for simulation and experiment available open-source [50]. Tensorflow has been used for programming and training the different deep learning architectures. 2 Nvidia GeForce RTX 3090 GPUs were used as hardware when running the experiments.

The number of control decisions per time stamp is important in this type of robotic scenario but we do not study its implications experimentally in the present work and we leave it for other studies. We consider a scenario where the number of iterations of the controller is high enough for the correct control of the vehicle.

We provide six experiments, where the eight models are evaluated to understand their differences. The models are tested using a never-seen scenario (*Town02*). In addition, we explore a lane-following scenario with no other vehicles or obstacles involved. Traffic lights and signals are also ignored in these experiments since they remain out of the scope of this study, the implications of including memory in the robot control. In the case of an intersection, the vehicle learns to follow a policy of going straight through it, based on the dataset provided by the expert agent and the imitation learning policy. The starting position is random among a set of points for each town, considering that the vehicle can drive lane-following for at least a few hundred meters

[1] https://leaderboard.carla.org/

Table 1
MAE and MSE metrics comparison for each trained model using test data from the dataset. Four different architectures are tested with visual memory and kinematic input (vehicle speed). ✔: supported. ✘: unsupported.

Model	Visual memory	Kinematic input	MAE test	MSE test
PilotNet*	✘	✘	0.0507	0.0177
PilotNet*	✘	✔	0.0332	0.0086
DeepestLSTMTinyPilotNet*	✔	✘	0.0662	0.0196
DeepestLSTMTinyPilotNet*	✔	✔	0.0456	0.0094
PilotNetx3* (Conv3D)	✔	✘	0.0295	0.0082
PilotNetx3* (Conv3D)	✔	✔	0.0074	0.0079
PilotNetx3* (TimeDistributed)	✔	✘	0.0289	0.0077
PilotNetx3* (TimeDistributed)	✔	✔	0.0069	0.0086

without interfering with situations that differ widely from the training dataset. We provide results for each described architecture to display its behavior in different situations and to extract a deeper insight into how each architectural modification or change in input data affects the final performance.

We do not provide experimental results comparison with the original PilotNet or DeepestLSTMTinyPilotNet architectures from the papers where they were proposed since our architectures feature distinct output configurations, as elaborated in Section 2. Conducting a fair comparison is not feasible due to these inherent differences. We also omit a comparison with prior baseline methods as our research inquiry focuses on whether the incorporation of kinematic input and visual memory to shallow visual-based end-to-end deep learning models, employing imitation learning, can lead to improvements in control performance in certain situations. While existing state-of-the-art models are engineered for a broad spectrum of tasks, they frequently integrate numerous sensors, employ deeper and more intricate architectures, and rely on extensive datasets. Previous studies emphasize the importance of compact and efficient deep learning networks for autonomous driving, which are adaptable for deployment across diverse devices with varying hardware capabilities. Our methodology prioritizes the simplicity of our models.

4.1. Comparison of models using common ML metrics

In Table 1, the best values obtained for MAE and MSE in the test set of the supervised dataset for each of the models are displayed. These values are better for the models whose input is kinematic sensory data, with a 90% reduction in MAE and 60% in MSE comparing the best and worst obtained results for each metric. This suggests that models with kinematic data are better at imitating the supervised datasets. They adeptly encapsulate the correlation between inputs and outputs.

These results are relevant indicators, but not enough for a reliable comparison of the models' final control behavior due to the high variety of situations that the vehicle could encounter at simulation test time and other variables that are involved in its quality such as the number of control commands per second that the model can generate. Inferring a single non-ideal control command to the robot actuators in a key moment may have worse consequences in robot performance than many non-ideal commands in harmless moments. To understand the complete behavior of the robot controller, a comparison in simulation should be conducted. Even so, we include this comparison with the supervised dataset since it is standard in machine learning research although for robotics applications these metrics are not conclusive to indicate good performance, good robot behavior. We need further experimental validation for the validation of the models, which is conducted in the following subsections.

4.2. Behavior in test scenario with top speed regulation

In this experiment, all the models are evaluated in the test scenario (*Town02*), as a lane-following vehicle. Each experiment is conducted five times starting from a random position in the test scenario. This is the most simple and common scenario, where the vehicle follows a lane using a reactive controller which calls the inference of a neural model on each iteration to generate the motor commands. The expert agent used for recording the supervised samples included in the training dataset has a top speed of 30 km/h. In this experiment, the models without kinematic sensory data also include a limit of 30 km/h to make a fair comparison, the vehicle speed is truncated when this limit is reached. In Section 4.3 the comparison without this top speed limitation is also studied.

In Table 2, we can see the results for the different models (columns) and the measured metrics (rows) in a test circuit, *Town02*. The *Successful experiments* metric is the most informative. It represents the number of experiments completed by the vehicle without collisions and without exceeding the maximum speed (30 km/h) out of the five runs. We also require that the agent reaches the average speed of the expert agent which is between 25 and 30 km/h to be considered successful.

The differences in this point are small. All the architectures can complete the experiments successfully, without collisions, and with an adequate average speed. The *Effective completed distance* is similar for all of them and the *Positional deviation mean per km* from the center of the lane is low. The *Vehicle jerk* for both control commands and velocity is also low, which translates to smooth and safe driving. The *Controller iterations frequency* is also adequate for the experiment. Although some of the models experiment some lane invasions, they are not problematic since the numbers are low and they do not cause collisions.

4.3. Studying the model without top speed limitation

In the previous experiments, some robot controllers needed a maximum speed limit like the one provided by the expert agent (30 km/h) to drive correctly. In this new experiment, we explore how removing that top boundary affects the behavior of the models. The rest of the experimental setup remains unchanged.

In Table 3, we can see the results of this experiment on *Town02*. We exclusively account for experiments without collisions in the table for all metrics, except for *Collisions per km* and *Lane invasions per km*. We can observe that models without memory or with only visual memory capabilities are more prone to collisions when the speed limit is not controlled. Looking at their *Max. speed* or *Average speed*, we can understand that they are not able to learn how to maintain a safe speed (30 km/h), self-regulating it, which leads to failure in all the experiments. For the models with kinematic sensory data input, the results are the same as in the experiment in Section 4.2, since the robot controllers using these models were already able to drive without a top speed limit. The interpretation of these results is that the kinematic data input is key for the robot to understand its state precisely and must be included in the model as input for proficient behavior.

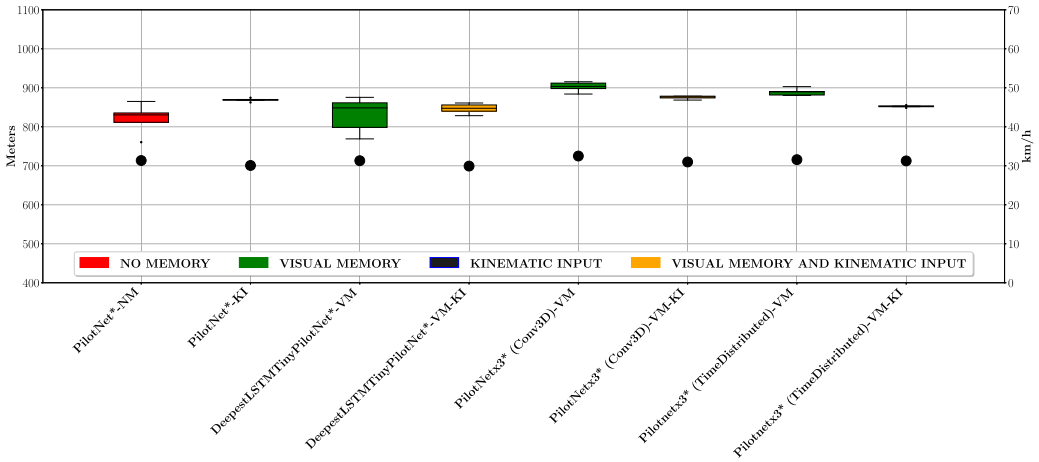
In Fig. 5, we provide the detail of *Effective completed distance* and *Max. speed* metrics for each model for the case with (Experiment 4.2) and without top speed restriction (Experiment 4.3). We can see that models with visual memory and kinematic input can traverse a bit longer effective distances maintaining a safe top speed. We can also see that the standard deviation is small and the number of atypical values is very low. The models' behavior is always similar. The black dots represent the mean *Max. speed* in the experiments. It remains similar for the cases where the top speed is controlled (top graph) whereas it generates extremely high value for models without kinematic input when it is not controlled (bottom graph).

Table 2
Comparison of models (columns) in different test environments considering some measured metrics (rows) provided by Behavior Metrics. Values in **bold** highlight the most interesting results. ✔: supported. ✘: unsupported.

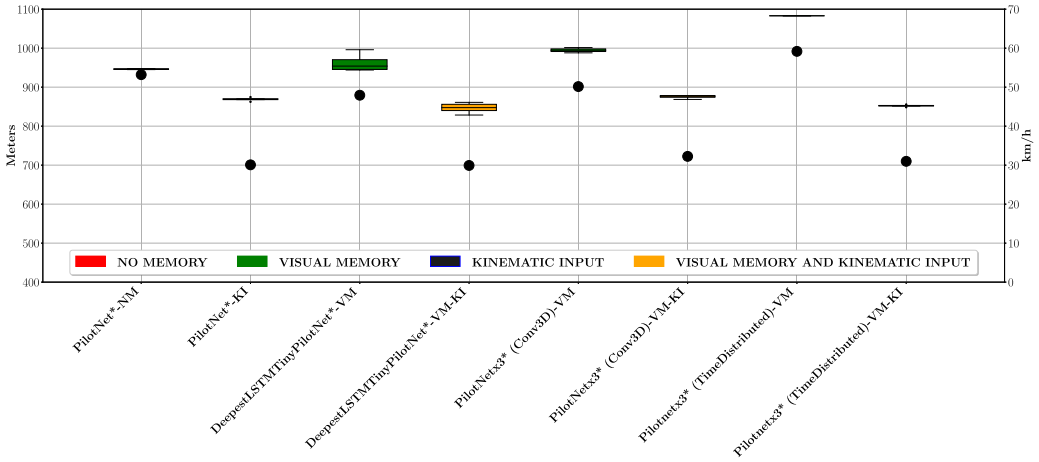
Map	Town02							
Model	Pilotnet*		DeepestLSTMTinyPilotNet*		Pilotnetx3* (Conv3D)		Pilotnetx3* (TimeDistributed)	
Visual memory	✘	✘	✔	✔	✔	✔	✔	✔
Kinematic input	✘	✔	✘	✔	✘	✔	✘	✔
Effective completed distance (m)	820.6	868.9	830.6	846.6	902.6	875.2	889.0	852.3
Position deviation mean per km (m/km)	0.25	0.26	0.22	0.33	0.24	0.25	0.29	0.29
Controller iterations frequency (Hz)	18.32	18.40	18.25	18.05	17.10	17.05	17.65	17.51
Vehicle jerk in control commands per kilometer	0.31	0.19	0.16	0.15	0.18	0.12	0.19	0.12
Vehicle jerk in velocity per kilometer	0.34	0.33	0.30	0.31	0.33	0.49	0.34	0.51
Average speed (km/h)	24.71	26.78	25.01	26.41	27.51	26.77	27.15	26.21
Max. speed (km/h)	31.35	30.08	31.30	29.92	31.57	31.25	31.50	30.98
Experiments with collisions	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
Collisions per km	0	0	0	0	0	0	0	0
Lane invasions per km	0	0.46	0	3.08	0	0.46	0	0
Successful experiments	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**	**5/5**

Table 3
Comparison of models in different test environments without top speed limit considering metrics from Behavior Metrics. **Bold** values (excluding *Successful experiments*) indicate changes in results from previous experiment results. Values in **red bold** and **bold** for *Successful experiments* highlight the most interesting results. ✔: supported. ✘: unsupported.

Map	Town02							
Model	Pilotnet*		DeepestLSTMTinyPilotNet*		Pilotnetx3* (Conv3D)		Pilotnetx3* (TimeDistributed)	
Visual memory	✘	✘	✔	✔	✔	✔	✔	✔
Kinematic input	✘	✔	✘	✔	✘	✔	✘	✔
Effective completed distance (m)	**946.3**	868.9	**962.0**	846.6	**994.8**	875.2	**1083.0**	852.3
Position deviation mean per km (m/km)	**0.23**	0.26	**0.21**	0.33	**0.28**	0.25	**0.28**	0.29
Controller iterations frequency (Hz)	**17.34**	18.40	**16.11**	18.05	**17.09**	17.05	**17.65**	17.51
Vehicle jerk in control commands per kilometer	**0.28**	0.19	**0.15**	0.15	**0.18**	0.13	**0.17**	0.12
Vehicle jerk in velocity per kilometer	**0.26**	0.33	**0.22**	0.31	**0.25**	0.49	**0.21**	0.51
Average speed (km/h)	**30.37**	26.78	**30.13**	26.41	**31.72**	26.77	**38.82**	26.21
Max. speed (km/h)	**53.20**	30.08	**47.93**	29.92	**50.15**	31.25	**59.18**	30.98
Experiments with collisions	**5/5**	0/5	**1/5**	0/5	**3/5**	0/5	**4/5**	0/5
Collisions per km	**2.09**	0.0	**0.32**	0.0	**0.95**	0.0	**1.45**	0.0
Lane invasions per km	**2.24**	0.46	**0.71**	3.08	**0.87**	0.46	**1.05**	0.0
Successful experiments	**0/5**	**5/5**	**0/5**	**5/5**	**0/5**	**5/5**	**0/5**	**5/5**



(a) Effective distance completed with 30 km/h restriction (Section 4.2)



(b) Effective distance completed without 30 km/h restriction (Section 4.3)

Fig. 5. Effective distance completed with 30 km/h restriction (top) and without (bottom). The right *y*-axis shows the vehicle's maximum speed (represented using black dots). NM: no memory. VM: visual memory. KI: kinematic input.

4.4. Taking the control of a fast-moving car

In this experiment, the robot controller with the deep learning model is suddenly connected to a vehicle that is already running at high speed, in this case, 50 km/h and 70 km/h. Once the vehicle reaches that speed, the model starts generating control commands for the car (throttle, steer, and brake) and we test whether the model can control that unseen extreme situation or not. In this situation, the vehicle should react fast to regain control over the car. This situation falls outside of the training dataset distribution.

In Table 4, the results of the experiment are shown. We only consider architectures with kinematic data as input, since we have already proved it to be necessary for proficient control. We can see that the only models able to gain back control and reduce the speed are the ones with visual memory and kinematic input for the 50 km/h case and only one of them for the 70 km/h case. This is a clear sign of the advantages of adding both types of features. In certain scenarios, both visual memory and kinematic input data help take back control of the car. In the experiments, the vehicle reduces the speed to the learned behaviors (less than 30 km/h) and then starts driving as usual, as shown in Fig. 6. In the depicted figure, we illustrate both distinct behaviors observed in our experiment. Firstly, we showcase the behavior of a model with both visual memory and kinematic input, demonstrating its ability to recover from perturbations and maintain forward progress. Conversely, we present another scenario featuring a model solely relying on visual memory where it is unable to recover and reduce the speed and it finally generates a collision. We can also see that a model with both memory types can take back the control, reducing the speed to the known point (30 km/h). On the contrary, for the model without memory, the ego vehicle continues driving at this top speed and control is not possible, quickly causing a collision. For the extreme case of 70 km/h, we can see that only PilotNetx3*(Conv3D) can control the car and reduce the speed to a known state. This could be attributed to the enhanced memorization capabilities that Conv3D provides. Considering the metric of *Average Speed*, favorable outcomes similar to the expert agent are observed when the model operates without incurring collisions.

4.5. Robustness to sensory manipulation

The next experiment tests the robustness of the models to a series of perturbations in the sensory data. In this case, we imagine a case where input data suffers some alterations, as it could occur in a real-world situation, and test how the memory helps in the control problem. Since the common input data for all the models is the visual data, we alter that information and study its implications considering the memory capabilities. In this case, we drop out randomly some parts of each of the visual data used as input. We again only consider architectures with kinematic data input since we have already proved it to be necessary.

In Table 5, the results of these experiments are displayed for *Town02* for a percentage of dropout of 50% and 90% (see Fig. 7 for an example).

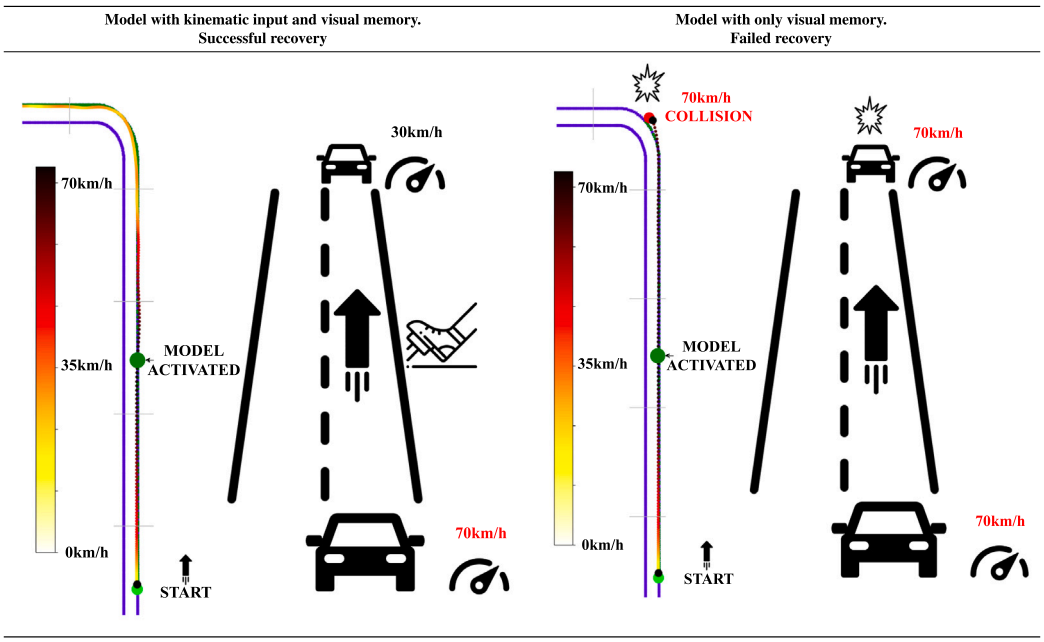


Fig. 6. Example of activating the vehicle autonomous driving system at a high-speed situation. On the left, the model with visual memory and kinematic input can restore the speed to the known point and continue driving. On the right, the model with only visual memory is not able to restore the speed and it collides due to the high speed.

Table 4
Comparison of models in a high-speed scenario where the model takes control when the ego vehicle is already at a speed of 70 km/h. For the *Average speed*, we only consider experiments without collisions. This experiment is tested in Town02. Values in **bold** highlight the most interesting results. ✔: supported. ✘: unsupported.

Map	Town02			
Model	Pilotnet*	DeepestLSTMTinyPilotNet*	Pilotnetx3* (Conv3D)	Pilotnetx3* (TimeDistributed)
Visual memory	✘	✔	✔	✔
Kinematic input	✔	✔	✔	✔
Speed	**50 km/h**			
Experiments with collisions	5/5	0/5	0/5	0/5
Average speed	–	27.18	27.07	26.82
Collisions per km	46.51	0.0	0.0	0.0
Successful experiments	0/5	**5/5**	**5/5**	**5/5**
Speed	**70 km/h**			
Experiments with collisions	5/5	5/5	0/5	5/5
Average speed	–	–	29.92	–
Collisions per km	27.25	29.95	0.0	26.95
Successful experiments	0/5	0/5	**5/5**	0/5

We again only consider *Succesful experiments* those without collisions and with an average speed close to the one provided by the expert agent. We can see that with a percentage of 50% of dropout, only two models are successful, PilotNet* and PilotNetx3*(TimeDistributed). The explanation for these results could be that DeepestLSTMTinyPilotNet* and PilotNetx3*(Conv3D) are reliant on the visual data that they receive (the first one includes *ConvLSTM layers* and the second one *Conv3D layers*) whereas the successful models while relying on visual data, they are more prone to consider kinematic input. For the extreme case of 90%, only PilotNetx3*(TimeDistributed) is still successful, which can be attributed to the visual memory capabilities. Each model with visual memory includes different ways of introducing it, and some of them are more important in certain extreme scenarios such as this one.

4.6. Visual memory length and density comparison

In this experiment, we evaluate the model's memory capabilities in terms of the length and density of the visual input data. Memory length refers to the amount of information that the model receives. In this case, we evaluate the implications of adding more visual input data. In Section 2.4, we describe that the visual memory used for training the

Table 5
Comparison of model performance modifying the input sensory information. For the *Average speed*, we only consider experiments without collisions. Values in **bold** highlight the most interesting results. ✔: supported. ✘: unsupported.

Map	Town02			
Model	Pilotnet*	DeepestLSTMTinyPilotNet*	Pilotnetx3* (Conv3D)	Pilotnetx3* (TimeDistributed)
Visual memory	✘	✔	✔	✔
Kinematic input	✔	✔	✔	✔
Percentage	**50%**			
Experiments with collisions	0/5	1/5	5/5	0/5
Average speed	26.12	22.54	–	25.96
Collisions per km	0.0	1.52	72.75	0.0
Successful experiments	**5/5**	0/5	0/5	**5/5**
Percentage	**90%**			
Experiments with collisions	5/5	0/5	5/5	0/5
Average speed	–	5.19	–	25.18
Collisions per km	23.68	0.0	12.31	0.0
Successful experiments	0/5	0/5	0/5	**5/5**

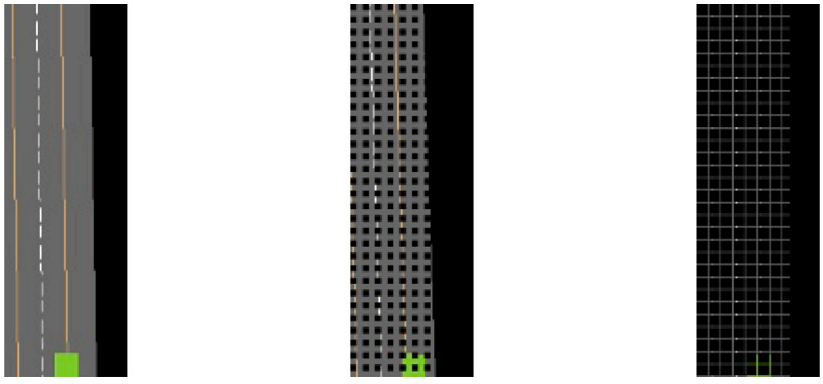


Fig. 7. Example of input data: normal (left), broken 50% (middle) and broken 90% (right).

models is (t, $t-5$, and $t-10$) considering that the sensors and controller run using 20 frames per second. Considering that the vehicle receives 20 frames per second, the default visual memory used is 0.5 s long. In this experiment, we evaluate other possible visual lengths (5 and 9 frames). 5 frames is 1 s of memory and 9 frames is 2 s if we consider that the frame rate is the same. Similarly, memory density refers to the time gap between frames. For the default configuration, we use (t, $t-5$, and $t-10$). For the experiments, we test (t, $t-1$, and $t-2$), (t, $t-10$, and $t-20$) and (t, $t-20$, and $t-40$). The experiments are conducted using the models with visual memory that receive several frames and considering the top speed boundary (PilotNetx3*(Conv3D) and PilotNetx3*(TimeDistributed)).

In Table 6, we present the results for the different proposed memory lengths. We can see that adding extra frames does not help for these models and they start failing when adding them. They are more prone to collisions and if we look at the *Position deviation mean per km*, we can see that greater frame counts correspond to heightened position deviation. Since the models are simple and based on PilotNet, we can attribute these results to models that are simple and that cannot understand a lot of frames. While it is conceivable that further architectural modifications could enhance their capacity to comprehend a wider array of frames, such enhancements are deemed unnecessary for the present scenario.

In Table 7, we present the results for the different studied memory densities. A memory with a small density (t, $t-1$, and $t-2$) can drive successfully and we can see that when the space between frames is widened, the models are more prone to collisions and its *Position deviation mean per km* is deteriorated. Similar behavior is observed for the *Position deviation mean per km* when the space between frames is excessively narrow (t, $t-1$, and $t-2$). The optimal configuration for these models and scenarios appears to be (t, $t-5$, and $t-10$).

Finally, after the six presented experiments, a summary of the results obtained in the experiments is displayed in Table 8 with further elaboration for *Successful experiments* metric provided in Fig. 8. As we can see, deep learning models including integrated visual memory and kinematic input excel in completing experiments, proving their superiority over other architectures. Although kinematic input contributes positively in certain situations (third column), the combined effect with visual memory proves most advantageous. Remarkably, the addition of visual memory alone (second column) shows no discernible improvements from the baseline, at least within the range of experiments conducted in this study for the metric examined, which

Table 6
Comparison of model performance with different visual memory lengths. For the *Average speed* and *Position deviation mean per km*, we only consider experiments without collisions. Values in **bold** highlight the most interesting results. ✔: supported. ✘: unsupported.

Map	Town02					
Model	Pilotnetx3* (Conv3D)			Pilotnetx3* (TimeDistributed)		
Visual memory	✔	✔	✔	✔	✔	✔
Kinematic input	✘	✘	✘	✘	✘	✘
Memory length (frames)	3	5	9	3	5	9
Collisions	0	0	1.0	0	0.8	1.0
Average speed	25.18	25.64	–	26.08	26.65	–
Position deviation mean per km	1.13	1.75	–	1.12	2.02	–
Collisions per km	0.0	0.0	1.30	0.0	0.0	6.25
Successful experiments	**5/5**	**5/5**	0/5	**5/5**	1/5	0/5

Table 7
Comparison of model performance with different visual memory densities. For the *Average speed* and *Position deviation mean per km*, we only consider experiments without collisions. Values in **bold** highlight the most interesting results. ✔: supported. ✘: unsupported.

Map	Town02			
Model	**Pilotnetx3* (Conv3D)**			
Visual memory	✔	✔	✔	✔
Kinematic input	✘	✘	✘	✘
Memory densities (frames)	t, $t-1$, $t-2$	t, $t-5$, $t-10$	t, $t-10$, $t-20$	t, $t-20$, $t-40$
Collisions	0.0	0.0	0.4	0.6
Average speed	24.96	25.18	26.59	26.07
Positions deviation mean per km (m/km)	1.39	1.13	1.35	1.89
Collisions per km	0.0	0.0	5.13	5.12
Successful experiments	**5/5**	**5/5**	3/5	2/5
Model	**Pilotnetx3* (TimeDistributed)**			
Visual memory	✔	✔	✔	✔
Kinematic input	✘	✘	✘	✘
Memory densities (frames)	t, $t-1$, $t-2$	t, $t-5$, $t-10$	t, $t-10$, $t-20$	t, $t-20$, $t-40$
Collisions	0.0	0.0	0.2	0.2
Average speed	25.94	26.08	26.63	26.01
Positions deviation mean per km (m/km)	1.12	1.12	1.47	1.59
Collisions per km	0.0	0.0	2.97	3.01
Successful experiments	**5/5**	**5/5**	4/5	4/5

could be a limitation. However, this observation could potentially change in more complex scenarios including moving obstacles or racing situations, where visual memory might offer enhanced performance. Nevertheless, it showcases its importance when combined with the kinematic input. Notably, successful modifications to the model lead to consistent success across all trials, whereas unsuccessful modifications result in complete failure across the experiment. We have proved with our research and experiments that adding kinematic input (vehicle speed) data and visual memory improves the general behavior and robustness of the deep learning model in the end-to-end control of an autonomous car for a lane-follow application. This augmentation offers advantages in navigating previously unseen and complex scenarios while adeptly regulating the vehicle's speed. Significantly, the most robust models have been obtained by combining both visual memory and kinematic input data.

5. Conclusions

In this paper, we have presented and studied four different deep learning architectures and a proposed variation for each of them for end-to-end robot control based on imitation learning for an autonomous driving problem. We have studied and proved how adding visual memory and kinematic input data to the models enhances the quality of the final control behavior for following the lane. These architectures are PilotNet*, DeepestLSTMTinyPilotNet*, PilotNetx3* (Conv3D), and PilotNetx3* (TimeDistributed), with their variation with also kinematic input. Specifically, we have studied how adding visual memory and

Table 8
Comparison summary of model performance across presented experiments. The addition of at least kinematic input data improves the final behavior and adding both types generates gains in certain scenarios. ✔: successful. ✘: failure.

Experiment \ Type	Visual	Visual memory	Kinematic input	Visual memory and kinematic input
Behavior in test scenario with top speed regulation (Section 4.2)	✔	✔	✔	✔
Studying the model without top speed limitation (Section 4.3)	✘	✘	✔	✔
Taking the control of a fast-moving car (Section 4.4)	✘	✘	✘	✔
Robustness to sensory manipulation (Section 4.5)	✘	✘	✘	✔

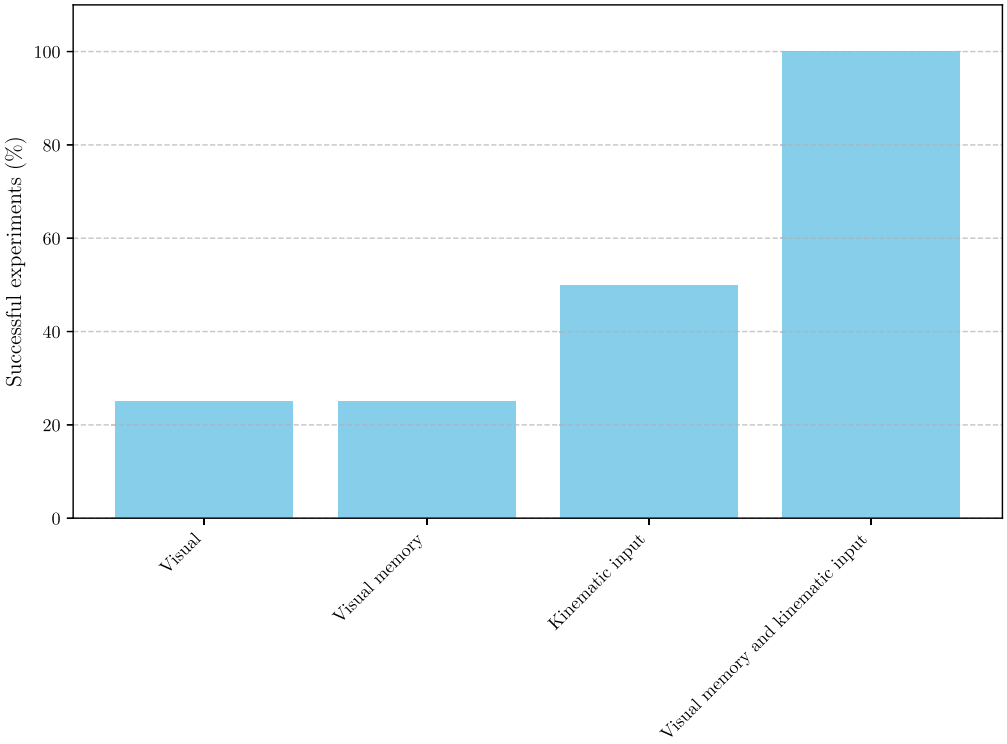


Fig. 8. Comparison summary of model performance enhancements across presented experiments focusing on *Successful experiments* metric.

kinematic input data to the models improves the performance in certain situations (by 75% in *Successful experiments*), such as taking control of a fast-moving car in never seen before high-speed scenarios or self-regulating the vehicle speed correctly. The models have been tested extensively in diverse simulated urban scenarios with varying layout designs, proving the research hypothesis widely.

Incorporating kinematic input data enhances speed control within the system. Moreover, when adding both visual memory and kinematic input data, the vehicle can drive in even more situations and is more robust to sensor failure or controlling never-seen situations like high-speed experiments. We have proved that adding at least kinematic sensory data can help in the final system compared to the situation where only instant visual perception is used, which leads to a less complete context understanding.

We have also studied different visual memory lengths and densities for extracting insight into how it affects the control system even for these simple deep learning architectures. We have proved that a lower density of frames can cause failed experiments when using simple architectures and that the length of the memory generates a comparable influence, causing failures when adding an excessive number of frames.

All the materials are provided as open-source, including the dataset, model weights and architectures, and the comparison software tool to make all experiments reproducible and extendable.

In future lines of work and addressing possible limitations of the present work, the ideas presented and tested in this paper need to be experimentally validated in physical vehicles. The models should also be tested in higher complexity settings, closer to real-world scenarios. For instance including other agents in the environments that the ego vehicle has to consider, such as a car driving directly in front of it in the same lane.

CRediT authorship contribution statement

Sergio Paniego: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Roberto Calvo-Palomino:** Writing – review & editing,

Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization. **JoséMaría Cañas:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by (GAIA) Gestión integral para la prevención, extinción *y* reforestación debido a incendios forestales, Spain, Proyectos de I+D en líneas estratégicas en colaboración entre organismos de investigación *y* difusión de conocimientos TRANSMISIONES 2023, Spain. Ref PLEC2023-010303 (2024–2026) by Agencia Estatal de Investigación de España, Spain.

References

[1] T. Litman, Autonomous vehicle implementation predictions implications for transport planning, 2022.

[2] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y.F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I.F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, A. Censi, Duckietown: An open, inexpensive and flexible platform for autonomy education and research, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, 2017, pp. 1497–1504, http://dx.doi.org/10.1109/ICRA.2017.7989179.

[3] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, D. Karuppasamy, DeepRacer: Educational autonomous racing platform for experimentation with Sim2Real reinforcement learning, 2019, arXiv preprint arXiv:1911.01562.

[4] M. O'Kelly, H. Zheng, D. Karthik, R. Mangharam, F1TENTH: An open-source evaluation environment for continuous control and reinforcement learning, in: H.J. Escalante, R. Hadsell (Eds.), Proceedings of the NeurIPS 2019 Competition and Demonstration Track, in: Proceedings of Machine Learning Research, vol. 123, PMLR, 2020, pp. 77–89, URL https://proceedings.mlr.press/v123/o-kelly20a.html.

[5] D.A. Pomerleau, ALVINN: An autonomous land vehicle in a neural network, in: Proceedings of the 1st International Conference on Neural Information Processing Systems, NIPS '88, MIT Press, Cambridge, MA, USA, 1988, pp. 305–313, URL https://dl.acm.org/doi/10.5555/2969735.2969771.

[6] Y. Cabon, N. Murray, M. Humenberger, Virtual KITTI 2, 2020, arXiv preprint arXiv:2001.10773.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, V. Koltun, CARLA: An Open Urban Driving Simulator, in: 1st Annual Conference on Robot Learning, CoRL 2017, in: Proceedings of Machine Learning Research, vol. 78, PMLR, 2017, pp. 1–16, URL http://proceedings.mlr.press/v78/dosovitskiy17a.html.

[8] E. Espié, C. Guionneau, B. Wymann, C. Dimitrakakis, R. Coulom, A. Sumner, TORCS, The open racing car simulator, 2005.

[9] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551, http://dx.doi.org/10.1162/neco.1989.1.4.541.

[10] S. Al-Janabi, A.F. Alkaim, Z. Adel, An innovative synthesis of deep learning techniques (DCapsNet & DCOM) for generation electrical renewable energy from wind energy, Soft Comput. 24 (14) (2020) 10943–10962, http://dx.doi.org/10.1007/s00500-020-04905-9.

[11] Y. Chen, R. Xia, K. Yang, K. Zou, MICU: Image super-resolution via multi-level information compensation and U-net, Expert Syst. Appl. 245 (2024) 123111, http://dx.doi.org/10.1016/j.eswa.2023.123111, URL https://www.sciencedirect.com/science/article/pii/S0957417423036151.

[12] Y. Chen, R. Xia, K. Yang, K. Zou, DNNAM: Image inpainting algorithm via deep neural networks and attention mechanism, Appl. Soft Comput. 154 (2024) 111392, http://dx.doi.org/10.1016/j.asoc.2024.111392, URL https://www.sciencedirect.com/science/article/pii/S1568494624001662.

[13] C. Gómez-Huélamo, A. Diaz-Diaz, J. Araluce, M.E. Ortiz, R. Gutiérrez, F. Arango, A. Llamazares, L.M. Bergasa, How to build and validate a safe and reliable autonomous driving stack? A ROS based software modular architecture baseline, in: 2022 IEEE Intelligent Vehicles Symposium, IV, 2022, pp. 1282–1289, http://dx.doi.org/10.1109/IV51971.2022.9827271.

[14] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, Y. Qiao, Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline, 2022, arXiv preprint arXiv:2206.08129.

[15] L. Han, L. Wu, F. Liang, H. Cao, D. Luo, Z. Zhang, Z. Hua, A novel end-to-end model for steering behavior prediction of autonomous ego-vehicles using spatial and temporal attention mechanism, Neurocomputing 490 (2022) 295–311, http://dx.doi.org/10.1016/j.neucom.2021.11.093, URL https://www.sciencedirect.com/science/article/pii/S0925231221018051.

[16] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, H. Li, Planning-oriented autonomous driving, 2023, arXiv preprint arXiv:2212.10156.

[17] F. Codevilla, E. Santana, A.M. López, A. Gaidon, Exploring the limitations of behavior cloning for autonomous driving, 2019, arXiv preprint arXiv:1904.08980.

[18] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, H. Li, End-to-end autonomous driving: Challenges and frontiers, 2023, arXiv preprint arXiv:2306.16927.

[19] A. Riboni, N. Ghioldi, A. Candelieri, M. Borrotti, Bayesian Optimization and Deep Learning for steering wheel angle prediction, 2021, arXiv preprint arXiv:2110.13629.

[20] H. Xu, Y. Gao, F. Yu, T. Darrell, End-to-end learning of driving models from large-scale video datasets, 2016, arXiv preprint arXiv:1612.01079.

[21] J. Kocić, N. Jovičić, V. Drndarević, An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms, Sensors 19 (9) (2019) http://dx.doi.org/10.3390/s19092064.

[22] A.O. Ly, M. Akhloufi, Learning to drive by imitation: An overview of deep behavior cloning methods, IEEE Trans. Intell. Veh. 6 (2) (2021) 195–209, http://dx.doi.org/10.1109/TIV.2020.3002505.

[23] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, Explaining how a deep neural network trained with end-to-end learning steers a car, 2017, arXiv preprint arXiv:1704.07911.

[24] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, J.K. Hedrick, Learning a deep neural net policy for end-to-end control of autonomous vehicles, in: 2017 American Control Conference, ACC, 2017, pp. 4914–4919, http://dx.doi.org/10.23919/ACC.2017.7963716.

[25] S. Yang, W. Wang, C. Liu, W. Deng, J.K. Hedrick, Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach, in: 2017 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2017, pp. 1033–1038, http://dx.doi.org/10.48550/arXiv.1703.09744.

[26] F. Codevilla, M. Müller, A. Dosovitskiy, A.M. López, V. Koltun, End-to-End Driving Via Conditional Imitation Learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 1–9, http://dx.doi.org/10.48550/arXiv.1710.02410.

[27] M. Toromanoff, E. Wirbel, F. Moutarde, End-to-end model-free reinforcement learning for urban driving using implicit affordances, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 7151–7160, http://dx.doi.org/10.1109/CVPR42600.2020.00718.

[28] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, D. Rus, Learning robust control policies for end-to-end autonomous driving from data-driven simulation, IEEE Robot. Autom. Lett. 5 (2) (2020) 1143–1150, http://dx.doi.org/10.1109/LRA.2020.2966414.

[29] L. Chi, Y. Mu, Deep steering: Learning end-to-end driving model from spatial and temporal visual cues, 2017, arXiv preprint arXiv:1708.03798.

[30] R. Zhao, Y. Zhang, Z. Huang, C. Yin, End-to-end spatiotemporal attention model for autonomous driving, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, Vol. 1, ITNEC, 2020, pp. 2649–2653, http://dx.doi.org/10.1109/ITNEC48623.2020.9085185.

[31] J. del Egio, L.M. Bergasa, E. Romera, C. Gómez Huélamo, J. Araluce, R. Barea, Self-driving a car in simulation through a CNN, in: R. Fuentetaja Pizán, A. García Olaya, M.P. Sesmero Lorente, J.A. Iglesias Martínez, A. Ledezma Espino (Eds.), Advances in Physical Agents, Springer International Publishing, Cham, 2019, pp. 31–43, http://dx.doi.org/10.1007/978-3-319-99885-5_3.

[32] H.M. Eraqi, M.N. Moustafa, J. Honer, End-to-end deep learning for steering autonomous vehicles considering temporal dependencies, 2017, arXiv preprint arXiv:1710.03804.

[33] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[34] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS '15, MIT Press, Cambridge, MA, USA, 2015, pp. 802–810, URL https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[35] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, W. Shi, Computing systems for autonomous driving: State of the art and challenges, IEEE Internet Things J. 8 (8) (2021) 6469–6486, http://dx.doi.org/10.1109/JIOT.2020.3043716.
[36] M.N. Sharath, B. Mehran, A literature review of performance metrics of automated driving systems for on-road vehicles, Front. Future Transp. 2 (2021) http://dx.doi.org/10.3389/ffutr.2021.759125, URL https://www.frontiersin.org/articles/10.3389/ffutr.2021.759125.
[37] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, E. Böde, Criticality metrics for automated driving: A review and suitability analysis of the state of the art, Arch. Comput. Methods Eng. 30 (1) (2023) 1–35, http://dx.doi.org/10.1007/s11831-022-09788-7.
[38] D. Paz, P. jung Lai, N. Chan, Y. Jiang, H.I. Christensen, Autonomous vehicle benchmarking using unbiased metrics, 2020, arXiv preprint arXiv:2006.02518.
[39] P.S. Chib, P. Singh, Recent advancements in end-to-end autonomous driving using deep learning: A survey, 2023, arXiv preprint arXiv:2307.04370.
[40] E. Santana, G. Hotz, Learning a driving simulator, 2016, arXiv preprint arXiv:1608.01230.
[41] Udacity's Self-Driving Dataset contributors, Udacity's self-driving dataset, 2022, https://github.com/udacity/self-driving-car. (Online; Accessed 10 January 2024).
[42] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, BDD100K: A diverse driving dataset for heterogeneous multitask learning, 2018, arXiv preprint arXiv:1805.04687.
[43] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, in: CVPR, 2020, http://dx.doi.org/10.48550/arXiv.1903.11027.
[44] Behavior Metrics contributors, Behavior metrics, 2022, https://github.com/JdeRobot/BehaviorMetrics. (Online; Accessed 10 January 2024).
[45] S. Paniego, N. Paliwal, J. Cañas, Model optimization in deep learning based robot control for autonomous driving, IEEE Robot. Autom. Lett. 9 (1) (2024) 715–722, http://dx.doi.org/10.1109/LRA.2023.3336244.
[46] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, 2016, arXiv preprint arXiv:1604.07316.
[47] A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A.A. Kalinin, Albumentations: Fast and flexible image augmentations, Information 11 (2) (2020) 125, http://dx.doi.org/10.3390/info11020125.
[48] S. Ross, G.J. Gordon, J.A. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, 2011, arXiv preprint arXiv:1011.0686.
[49] S. Paniego, R. Calvo-Palomino, J. Cañas, Behavior metrics: An open-source assessment tool for autonomous driving tasks, SoftwareX 26 (2024) 101702, http://dx.doi.org/10.1016/j.softx.2024.101702, URL https://www.sciencedirect.com/science/article/pii/S2352711024000736.
[50] Open-source paper resources contributors, Open-source paper resources, 2023, https://roboticslaburjc.github.io/publications/2023/enhancing_end_to_end_control_in_autonomous_driving_through_kinematic_infused_and_visual_memory_imitation_learning. (Online; Accessed 10 January 2024).

Sergio Paniego Blanco is a Ph.D. student at the Department of Teoria de la Señal y de las Comunicaciones y Sistemas Telemáticos y Computación at Universidad Rey Juan Carlos. He completed his Masters in Artificial Intelligence at Universidad Politécnica de Madrid, Spain in 2019 and his B. in Computer Engineering and Software Engineering at Universidad Rey Juan Carlos in 2018. His main research areas are Autonomous Driving, Deep Learning, Model Optimization and Robotics.

Roberto Calvo-Palomino currently serves as an Assistant Professor at Rey Juan Carlos University, where he teaches in the Software Robotics Engineering program. Previously, he was a post-doctoral researcher at the Madrid Institute for Advanced Studies (IMDEA Networks), focusing his research on collaborative systems for electromagnetic spectrum monitoring. He holds a Bachelor's degree in Computer Engineering from Rey Juan Carlos University and a Ph.D. in Telematic Systems from Carlos III University of Madrid and IMDEA Networks.

Throughout his academic and professional career, Roberto has published over 30 articles in various journals, 11 of which are indexed in the Journal Citation Reports (JCR). He has actively participated in 7 European and 4 national projects, and has led 13 technology transfer projects with both national and European companies. He has completed significant pre-doctoral internships; at the University of Ljubljana in Slovenia, he developed new methods for the precise estimation of aircraft signal flight times, and at the Swiss Defense Department, he worked on distributed signal decoding using RTL-SDR receivers, focusing on the security of IoT devices through artificial intelligence.

His current research interests include:

- Deep-learning-based autonomous navigation algorithms for autonomous driving.
- Deep-learning-based algorithms for driver state detection and prevention.
- Embedded systems and security in IoT.

José-María Cañas-Plaza is tenured associate professor at Universidad Rey Juan Carlos, where he leads the RoboticsLabURJC. He received the M.S. and Ph.D. degrees in telecommunications engineering from the Universidad Politécnica de Madrid. He has done research in robotics at Carnegie Mellon University and the Georgia Institute of Technology. His research interests include AI/ML driven Robotics, Computer Vision, and the engineering education of those disciplines.