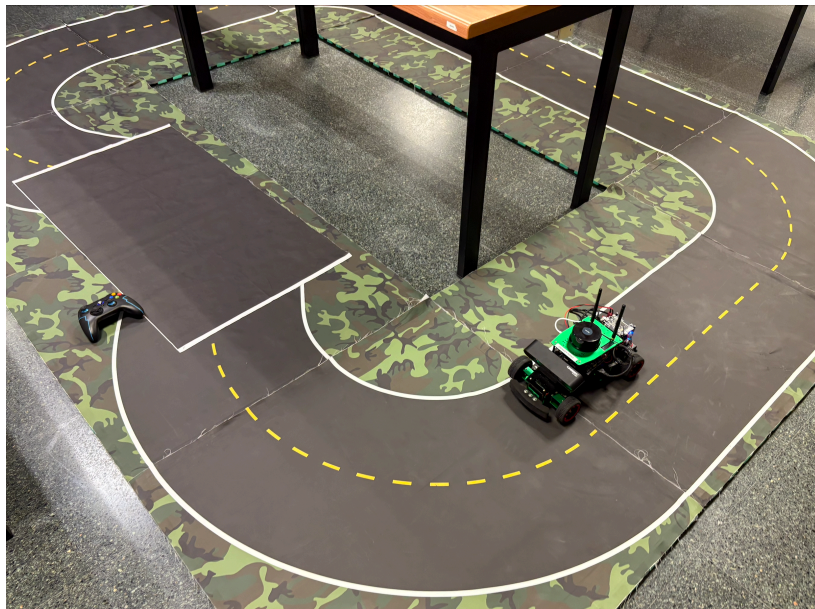


Physical Car Report

Joan Lafuente, Maria Pilligua and Xavi Soto



Contents

1	Introduction	3
2	Model	3
2.1	CIL++ Architecture	5
3	Sensors	5
3.1	LiDAR Postprocessing	6
4	Data	7
4.1	Circuits	7
4.2	Lighting Conditions	9
4.3	Locations	10
4.4	Obstacles	10
4.4.1	Stop	10
4.4.2	Avoid	11
4.5	Other details	11
4.6	Data characteristics	12
5	Config file Params	13
5.1	RGB Camera Model	13
5.2	RGB Camera + LiDAR Model	13
5.3	Augmentations	14
5.4	Qualitative Results	14
5.4.1	RGB Camera Model	15
5.4.2	RGB Camera + LiDAR Model	15
6	Main problems	16
7	What would we do different	17
7.1	What would we change if we started from scratch	17
7.2	What would we change if the vehicle had to drive around UAB	18
8	Conclusions	18

1 Introduction

The aim of this project was to implement a complete end-to-end pipeline for training an autonomous driving model using a physical robotic vehicle. The process included from the data acquisition in a real-world environment to the final deployment of a trained imitation autonomous driving model.

The core learning paradigm used for the model was imitation learning, where the system learns to replicate human behavior by observing examples. In this case, the robot was manually driven using a joystick while various types of sensor data, including visual information from a front-facing camera, distance data from a LiDAR sensor, and the control signals such as steering and acceleration. The joystick was used not only as control device, but also as a source to obtain labeled data reflecting the driving decisions of the drivers.

The most critical aspect of the project was the creation of a robust and diverse dataset. Since there was no autopilot available, all training data had to be collected manually. This required covering a variety of driving scenarios, such as staying within lanes, recovering from off-track positions or stopping at obstacles. The collected data was then used to train a CIL++ model, which is designed to predict low-level control commands directly from the input from the sensors.

In this report we will explain the model and sensors used and provide a detailed explanation of the data collection process analyzing the results and how the models were improved based on its flaws highlighting how important is having good data to improve the model performance.

2 Model

The model that we used is CILv2 as recommended in class. This model was released in 2023 in the paper *Scaling Vision-based End-to-End Autonomous Driving with Multi-View Attention Learning* et. al. Yi Xiao. CIL++ was introduced as a improvement to previous existing autonomous driving models.

The beginning was CIL, the first model that processed high-level commands (such as "turn left" or "go straight") as input. This model is train using imitation learning which consists of giving the model data of correct driving examples and making the model predict what the car should do in each situation. Then you have losses that penalize the model if the guess is not close to the actual movement done by the real driver (Ground truth). That is where the name of CIL comes from, as it is using conditional imitation learning.

The original CIL model does this conditioning by activating various branches in the network, each trained for a particular maneuver. However, even though the idea of high level commands is good, the way of introducing it (using the branches) was not too straightforward, that is why CILRS appeared.

CILRS streamlined the architecture by eliminating the distinct branches and instead considering the command as input to a unified network, resulting in more stable and efficient training. Nonetheless, CILRS depended on just one front-facing camera with a limited field of view and did not incorporate any attention mechanism, restricting its capacity to comprehend the entire driving environment, specially in complex situations.

CIL++ was created to address these issues by integrating multi-view data from three cameras (ensuring 180° coverage), incorporating an attention-based transformer for enhanced spatial

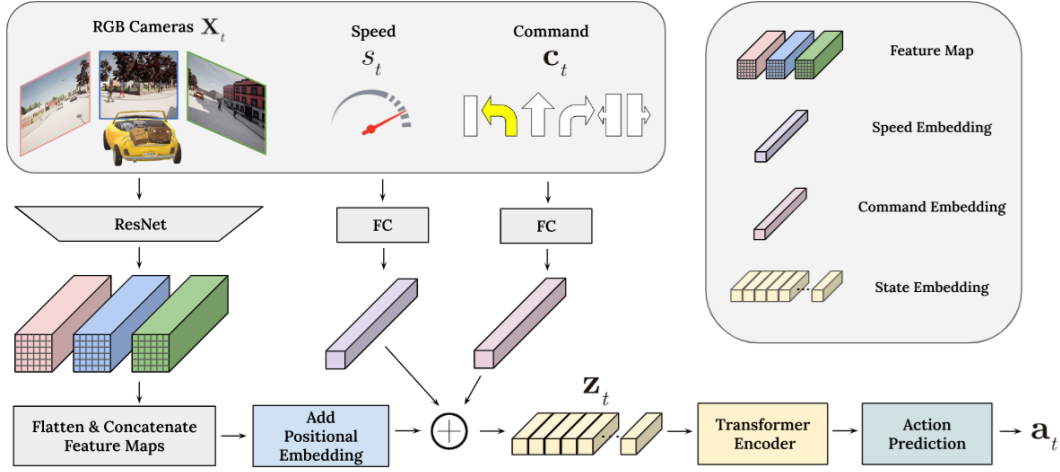


Figure 1: Architecture of CIL v2

context processing across views, and refining training through a more authentic expert driver. The outcome gives a model that, as they say in the paper is "processing higher-resolution images using a human-inspired HFOV as an inductive bias and incorporating a proper attention mechanism." Moreover, achieves competitive performance compared to models which are more costly to develop.

Here there are 2 things worth mentioning. First, CIL, CIRLS and CIL++ are all vision based, meaning that it has no extra sensors that may help during the inference. There are no depth sensors, no lidars, no infrared sensors, just a camera. The main point is that they defend that the camera alone is enough to be able to infer everything you need to drive properly.

The second point to notice is that they called their method end-to-end. This is a concept which is not strictly defined and has created some confusion in the literature. In this paper, what they mean by end-to-end is that the model directly maps raw sensor inputs (in this case, just RGB images, along with speed and a navigation command) to the final control outputs: steering and acceleration.

This contrasts with traditional autonomous driving systems, where the process is split into multiple stages like perception, localization, planning, and control. In those classical pipelines, intermediate representations such as object detection, semantic segmentation, depth estimation, and HD maps were required at each step to make the final decision.

These intermediate steps often needed separate models, human-annotated data, and additional sensors like LiDAR or radar. What makes end-to-end approaches like CIL++ relevant is that they greatly simplify the pipeline by skipping these hand-crafted stages and learning everything in a single neural network, trained only from human driving demonstrations. This makes the system cheaper, more scalable, and closer to how humans actually learn to drive, by observing the world through our eyes and acting accordingly.

2.1 CIL++ Architecture

In 1 you can see the original architecture of CIL++. This model is based on a transformer. It takes as input a one to three images of the scenario as well as the current speed and high-level command. As output it gives the action that the car should take in that moment by specifying the direction and acceleration it should have.

The model works as follows, the speed and the command pass each through its own fully connected layer, the input images pass through a resnet block which has been pretrained on ImageNet, the output features of each of the images is then flattened and concatenated. A positional embedding is added and then the result is added with the output of the fully connected layers of the speed and the command. The result of this is called Z_t . This goes into the transformer encoder, which has the self-attention mechanism.

In the paper they say that they use this hybrid approach between CNNs and transformers to have the best of both worlds. Keep the inherent properties of CNNs (i.e., exploiting locality and translation equivariance), as well as leveraging the attention mechanism of transformers.

The transformer encoder used is composed of 4 layers, each implementing multi-head attention. Specifically, every layer applies multi-head self-attention, followed by layer normalization and a feed-forward neural network. After passing through all layers, the outputs from the attention heads are concatenated and projected through a linear layer. This resulting representation is then passed on to the action prediction block which consists of an average pooling layers and an MLP composed of three layers followed by ReLu activations. The output of this is just a 2 dimension vector which accounts for the predicted control signals: steering angle and acceleration (which also accounts for throttle and brake, depending on the sign of the value).

3 Sensors

To be able to do autonomous navigation, the vehicle is equipped with several sensors that provide data about its environment, motion and control state. These inputs were essential both for real-time inference and for training models to mimic human driving behavior. Below, we detail the main sensors and input sources used:

- **RGB Camera:** A forward-facing RGB camera was mounted on the vehicle to capture visual data from the driver's point of view. The camera provided high-resolution images of the environment, including the road, lane markings or obstacles among others. This visual data was crucial for the model to learn to stop when having objects in front, to maintain its trajectory through the right lane, to come back to the correct lane when not being at it or even driving to the track even when starting from outside it. This visual data stream was the main input for the CIL++ model.
- **Steering angle:** The angle of the steering wheel, the orientation of the front wheels, was used to record the direction of the car when being controlled manually. This input indicates the driver's steering decision in response to the observed environment, either to evade some obstacle or to turn on curves. It was used as ground truth for the model indicating how a human would steer given a particular image.
- **Vehicle speed and Acceleration:** Speed and acceleration are treated as distinct data points that have different origins. The vehicle's current speed is read directly from its inner sensors, being used as input feature to the AI model. The acceleration however is

captured directly from the joystick, representing the driver's action and also being used as ground truth. During the autonomous driving, the model predicts an acceleration value which then is translated into a concrete desired speed command for the motors. This conversion is done by integrating the predicted acceleration over the time elapsed since the last control loop (Δt) and calculates the speed based on the Uniformly Accelerated Motion. The specific implementation uses different multipliers depending whether the car is accelerating or braking as shown in the formula.

$$\Delta v = a_{predicted} \cdot M \cdot \Delta t$$

$$v_{desired} = v_{current} + \Delta v$$

In this formula, $v_{current}$ is the car's currently measured speed and $a_{predicted}$ is the model's output. The variable M is a multiplier set to an ACCELERATION_MULTIPLIER of 5.0 for positive acceleration or to a BRAKE_MULTIPLIER of 20.0 for braking. Finally, $v_{desired}$ is limited to a safe range ensuring it does not exceed a maximum speed of 0.2 m/s before being sent to the car.

- **Joystick:** Was used as primary manual control for the data collection. It helped generate the data the steering and speed data which was recorded and used to train the model. The joystick also allows to transition from manual to AI mode, enabling the control of the car based on the predictions of the model, while keeping the manual control available in case of requiring to make an emergency stop.
- **LiDAR sensor:** A 2D LiDAR sensors was also used, although only for few models, to gather the spatial information about the vehicle surroundings. By emitting laser pulses and measuring their return time, the LiDAR can generate a dense point cloud that is able to compute distances to nearby objects. It was useful to enhance the model's awareness of obstacles, especially for evasion maneuvers, to correctly return to the lane only after fully passing an obstacle.

3.1 LiDAR Postprocessing

The LiDAR sensor produces a point cloud, which cannot be directly used as input for the CIL++ model without either modifying its architecture or transforming the point cloud into a compatible format.

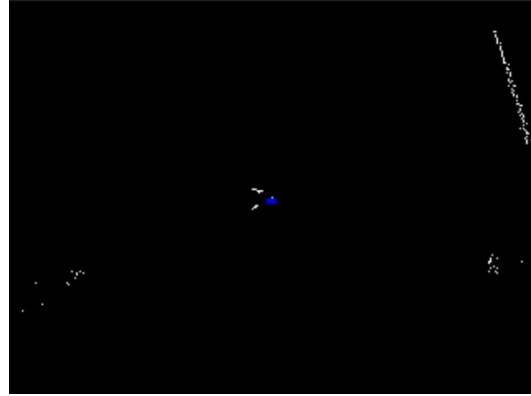
We opted for the second approach, transforming the point cloud into an image, since it avoided changes to the model architecture and leveraged the model's existing ability to process image inputs.

To achieve this, we adapted the code originally used for the live visualizer of the camera and LiDAR. Our main modification was to generate an image representing a bird's-eye view of the car, extending the view to include not only the front but also the rear of the vehicle (see Figure 2).

This change allowed the LiDAR to provide additional context, particularly valuable for obstacle avoidance. While the camera captures forward-facing information, the inclusion of the rear in the LiDAR image helps the model determine whether an obstacle is still alongside the vehicle. This is crucial for safely deciding when it is possible to return to the original lane after avoiding an obstacle.



(a) Camera view of the scene, where can be seen two boxes.



(b) Corresponding LiDAR bird's-eye view. The blue dot indicates the car's position.

Figure 2: Comparison between the camera (a) and LiDAR (b) views of the same scene.

4 Data

The most critical component of this project is the data collection process. This is because we rely on imitation learning, which involves teaching the model to replicate the behavior of an expert under diverse conditions. As a result, it is essential to capture a wide range of scenarios during training, including variations in lighting, circuit layouts, and backgrounds. As with any machine learning model, the quality of the data directly affects the model's performance. In particular, undesirable behaviors recorded during training will likely be learned and reproduced by the model.

For example, if all the data were collected under the same lighting conditions or in a single environment, the model might overfit to specific visual cues, such as reference points or background elements, instead of learning generalizable driving behavior. This would significantly reduce its performance in real-world situations.

Before starting collecting data, we dedicated one day to becoming familiar with the remote control car. During these initial tests, we found that the default joystick control script was overly sensitive for steering. Even small inputs caused sharp turns, making it difficult to smoothly control the car. To address this, we set the sensitivity to the lowest possible level, bringing it down from 5 to 3, which despite it does not seem a big change, it is a exponential, so it actually reduced a lot the sensitivity. This adjustment allowed for smoother and more consistent driving, reducing the need for frequent corrections. This is especially important in imitation learning, where unintended behaviors in the training data might negatively impact the model's performance.

In the following sections, we describe the different setups and conditions under which data was recorded, including the various circuits, lighting conditions, and obstacles.

4.1 Circuits

We used six different circuit layouts to collect training data, each designed to introduce variability in driving scenarios. This diversity should allow the model to generalize better to new

environments. For every circuit, we recorded data while driving in both directions, always using the right lane.

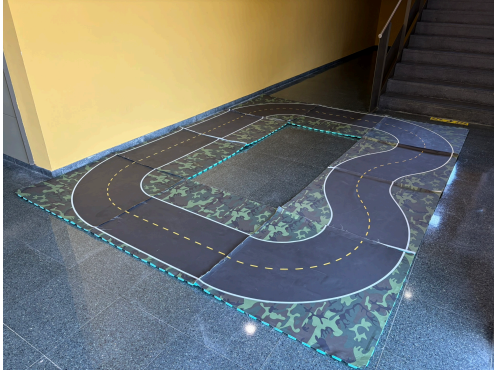


Figure 3: Provided circuit.



Figure 4: Provided circuit with small degree curves at both sides.



Figure 5: Provided circuit without short straights.

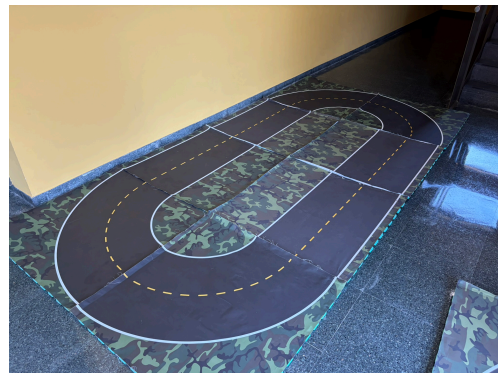


Figure 6: Circuit with two straight segments between two sharp turns.



Figure 7: Circuit with small degree curves between sharp turns.

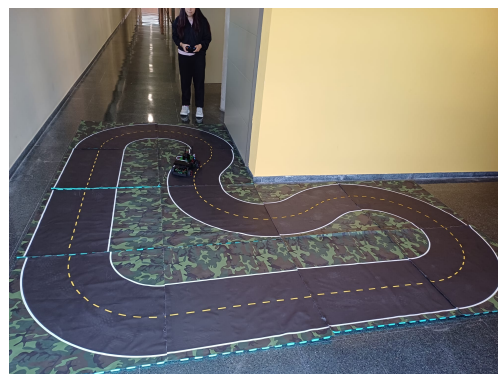


Figure 8: Large custom circuit.

The first circuit we used was the default configuration provided (see Figure 3). This layout includes a wide range of curves and scenarios suitable for training a base model. We also created a small variation of this default circuit by replacing the long straight segment with two small-degree curves from another circuit (see Figure 4).

Building on the provided circuit, we created three additional variations that eliminated both short straight lines, introducing sharp 180° turns not present in the original circuit. These variations were:

1. The provided circuit with the short straight lines removed (Figure 5).
2. A simplified layout with only straight segments between sharp 180° turns (Figure 6).
3. A layout with continuous small degree curves on both sides between sharp turns (Figure 7).

These modified circuits emphasized turning behavior, allowing us to collect more diverse data for cornering scenarios. This was particularly useful in training the model to handle turns more effectively. Additionally, these configurations required the model to interpret some of the curves differently than in the standard circuit.

Lastly, we built a larger, more complex circuit using pieces from two different ones available in class (see Figure 8). This layout was significantly different from the others and introduced entirely new situations for the car to handle. By expanding the diversity of driving conditions, this circuit helped improve the robustness and generalization capabilities of our model.

4.2 Lighting Conditions

To enhance the model's behavior and generalization capabilities, training data was collected under a range of lighting conditions. These scenarios were designed to expose the model to diverse visual environments, improving its robustness in real-world applications. The lighting conditions can be grouped into five categories:

1. **Naturally well-lit, without artificial lighting:** The shutters were fully opened, and all interior lights were turned off. This configuration provided strong natural illumination and introduced reflections when the vehicle faced windows. These samples were important for improving the model robustness to high-glare conditions.
2. **Naturally well-lit, with artificial lighting:** With the shutters open and the room lights turned on, lighting conditions became more balanced. This reduced window glare and shadows, helping the model adapt to well-lit but complex environments.
3. **Partially lit by natural light, without artificial lighting:** The shutters were partially closed and all interior lights were turned off, resulting in reduced ambient lighting. This setup introduced stronger window reflections and shadowed regions, simulating worse visibility conditions. Including this data contributed to improving the model's robustness in less optimal lighting environments.
4. **No natural light, without artificial lighting:** The shutters were completely closed, and no lights were turned on. Although the room was mostly dark, some residual light from adjacent corridors allowed to correctly see the track. This data helped the model learn to navigate in low-light conditions.

5. **No natural light, with artificial lighting:** This was the most controlled setup. The shutters were closed to block all external light, and the interior lights provided consistent, even illumination.

4.3 Locations

The different circuits were placed in various locations to enhance the model capabilities and prevent it from relying on external reference points, such as specific furniture or walls, for navigation, which could introduce bias unrelated to the circuit itself.

Two driving locations were used inside the class, with recorded data collected on the two circuit locations farthest from the whiteboard. This ensured that chairs, walls, and table legs appeared in different positions relative to the circuit. Additionally, the circuit was sometimes rotated 180 degrees to further vary the environment. Only the provided circuit was used in these cases, as the others did not fit the available space.

Furthermore, data was recorded in the corridor using only the narrower circuits. Finally, nearly all circuits were used at the end of the corridor, where a wider space allowed to assemble all of them. This area introduced new conditions, including a floor-level window that caused stronger reflections, adding complexity to the recorded data.

4.4 Obstacles

Effective obstacle handling is crucial for autonomous driving models, as the vehicle, in this case the remote control car, must avoid collisions when encountering obstacles on the track. In imitation learning, the model's behavior is shaped directly by the examples it observes during training. Therefore, it is essential to provide clear examples of how the model should behave when facing specific objects.

In this context, a decision was needed: when encountering an obstacle, the model could either avoid it or stop. We chose to provide examples of both behaviors depending on the object. Ultimately, the model was trained to avoid only a specific bottle, tall enough to be detected by the LIDAR, while for all other obstacles, the model was expected to stop.

Regarding the amount of data collected for this task, no obstacles were introduced during the first two hours of driving data collection. Only after the model demonstrated confident navigation of the circuit did we begin adding obstacles in subsequent recording sessions.

4.4.1 Stop

We recorded data for the stop action using a variety of objects. Initially, we used simple obstacles such as the car's box, the charger, and the joystick. Once the model reliably stopped in front of these objects, we gradually increased the difficulty by introducing more complex and irregularly shaped obstacles.

These included our feet in different positions and orientations, backpacks, small toy cars, and a plastic construction-type vehicle (see Figures 9 and 10). We found that using these obstacles was sufficient, as the model was able to generalize and stop for other objects, even when they were significantly different from those used during training.

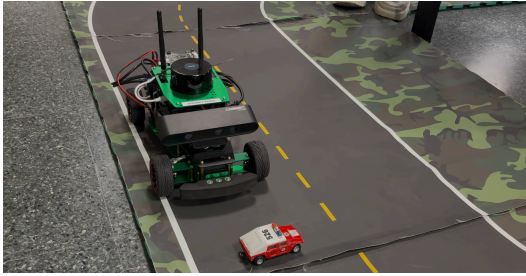


Figure 9: One of the small toy cars used for training the stop behavior.



Figure 10: Plastic construction-style vehicle used.

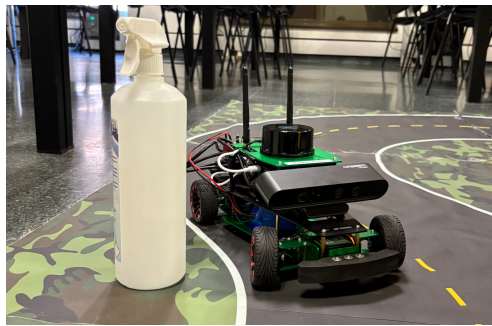


Figure 11: Image showing the remote-controlled car and the bottle used to train the avoidance behavior.

4.4.2 Avoid

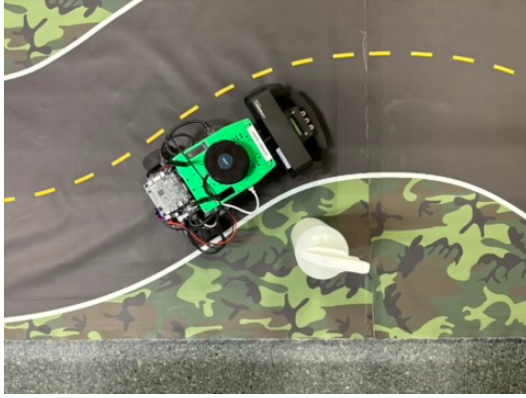
We recorded data for obstacle avoidance using two types of objects: small human figures and a tall bottle (see Figure 11). However, after encountering several issues, discussed in Section 6, we decided to not use of the small human figures.

The tall bottle was visible within the LIDAR plane, which provided a key advantage for obstacle avoidance. This allowed the model to determine whether the bottle was still alongside the car, helping it decide whether it was safe to return to the correct lane (see Figure 12). Without this LIDAR input, the car's camera would not detect any visible obstacle, making it difficult for the model to understand why it was not driving in the right lane, thus introducing misleading data during training.

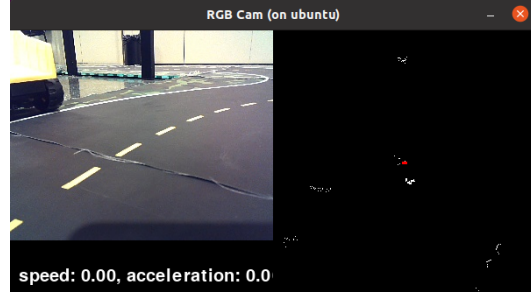
4.5 Other details

To improve the model's ability to recover from poor initial positions, the car was intentionally started outside the track, or even outside the circuit pieces themselves, in a significant portion of the recorded RosBags. In these cases, the circuit was always visible in the camera view, allowing the model to learn how to return to the track when beginning from suboptimal locations.

To further improve the robustness against variations in the track layout, A4 sheets of paper were placed at various points such as corners, mid-track positions, on lines, or in random locations. Additionally, the large black plastic/paper available in class was sometimes used to cover all



(a) Bird's-eye view of the scene.



(b) LiDAR and car camera view of the same scene.

Figure 12: Comparison between visual (a) and LiDAR (b) perspectives of the scene.

the track lines. In these situations, we mimicked the actions we would take if we could only see what the car's camera was capturing, to ensure the data reflected realistic and consistent behavior rather than introducing misleading examples.

Finally, all team members contributed equally to data collection, each driving the car for approximately the same amount of time

4.6 Data characteristics

To conclude this section, Table 1 summarizes the main characteristics available for training our models. The lower number of LiDAR images is due to the camera capturing more samples per second than the LiDAR sensor.

Based on the CALLBACK_TIME of 0.05s set for the camera node, we can conclude that the camera saves images at 20 FPS. Based on this FPS values and the total number of RGB images we can conclude the amount of hours of training data.

$$\text{Total Hours} = \frac{\text{Total Images}}{\text{Frames Per Second} \times \text{Seconds Per Hour}}$$

and if we substitute the values:

$$\text{Total Hours} = \frac{669,306}{20 \times 3600} \approx 9.3 \text{ hours} = 9 \text{ hours and } 18 \text{ minutes}$$

Characteristic	Value
Hours of data collection	9h 18min
Number of RGB images	669.306
Number of LiDAR images	548.236
Hours of data with LIDAR	7h 36min
Number of drivers	3
Number of circuits	6
Number of locations	4
Number of lighting conditions	5

Table 1: Collected data characteristics.

5 Config file Params

In this section we explain the best performing models (with and without LiDAR) and their params used.

5.1 RGB Camera Model

This model was trained only with the RGB camera sensor but did not contain all the data available, instead, some rosbags were discarded as they were misleading for the model. We removed the data where we avoided or stopped in presence of people (small figurines of people not real people). Additionally, the data collected to evade obstacles was also removed due to the problems mentioned in 6. The model was trained with 41 different rosbags with varying circuits, light, obstacles and many other variations mentioned in 4. The model was trained for 60 epochs, and the best performing model was the 45. Prior to this model, we normally trained for 30 epochs which already achieved good results, but increasing the epochs helped the model further improving its driving skills. After testing the different epochs of the model, we saw that after 50 epochs the model was clearly overfitting not being able to drive correctly.

In the end, we achieved a robust autonomous driving model that was able to stop to object's it had never seen before, drive in a high variation of circuits, including ones that it had never driven before, being able to come back to lane even when starting from outside the circuit.

5.2 RGB Camera + LiDAR Model

For this model we used the exact same configuration as for the previous one (the best performing non-lidar model) and included the LiDAR information, which was always being recorded despite not being used. The LiDAR was included to solve the object's evasion problems faced when only using the RGB camera as mentioned in 6. This model was again trained for 60 epochs. However, we observed that after epoch 40 the model just overfitted massively to the data. That is due to the amount of data that we had. As shown in the Table 1, recoding with the lidar we obtained less images and only certain rosbags did contain actual avoidance events. Consequently, the model overfitted to the small subset we had to learn.

Epoch 35, however, had a great balance between generalization and memorization. Overall, this model managed to learn to drive correctly; we noticed a slight decline in performance

compared with the RGB-only version. Despite this, it showed that it learned to stop reliably for unseen obstacles and to avoid the bottle well.

5.3 Augmentations

As an extra experiment we also tried to add augmentations to help the model work in different lighting conditions. The ideal setup would be to have a lot of data in a lot of lighting conditions to help the model learn to generalize. However, due to the time constraints that we had it was unfeasible to record huge amounts of data of lots of scenarios.

As explained in section 4.2, we actually recorded in as many different scenarios and lighting conditions as we could think of. With all of the combinations that were possible in class playing with the natural and artificial lighting. We also payed a lot of attention to record as much data as we could in low light conditions, that is, with the blinds totally closed and no artificial lighting. However, even though we recorded a lot of data in these conditions, it was not enough for the model to generalize in all situations. That is why we decided to test augmentations.

We used an augmentation that changed the gamma of input images. In fig 13 you can see the effect changing the gamma has over the images. We made it so that the input was augmented with different values of gamma. The values that we used were $[1/10, 1/7.5, 1/5, 1/2.2, 1, 2.2, 5, 7.5, 10]$ and were uniformly selected during training. The main point is that changing the gamma is a synthetic way of simulating different lighting conditions.



Figure 13: Example of the different values of gamma used in on of the images captured by the car.

However, at the end this did not give as good results as we expected. Nonetheless, we strongly believe this would work if more realistic light changes were made to the input images. This can be done by training low-light models in a "reverse" manner, that is, feeding the model a well-lit image and making it predict a worse-lit image that is overly saturated (simulating reflections from windows) or with blocking shadows or poorly lit images.

This is a novelty model that, to our knowledge, does not exist and would be relatively easy to train using existing low-light enhancement datasets and even architectures.

Moreover, the task the model would have would be even easier than when enhancing it, as its task is merely to degrade the image instead of reconstructing the original, well-lit image. However, regarding its simplicity, we believe it would bring a huge improvement if done correctly.

5.4 Qualitative Results

After training the models, it is also important to visualize their behavior and try to understand why they make the different decisions. Gaining insight into these decisions is especially important in the context of autonomous driving, where understanding the model's reasoning can help identify potential biases or inappropriate decision-making cues. For example, in our case if a model learns to turn based on the presence of a chair rather than relevant features of the

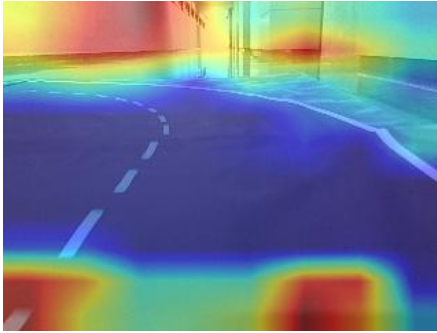
driving circuit, moving the chair could lead to a crash. This highlights the need to check that the model has learned to focus in relevant features of the track.

For this reason, we have decided to analyze the Grad-CAM visualizations of the trained models on a validation set.

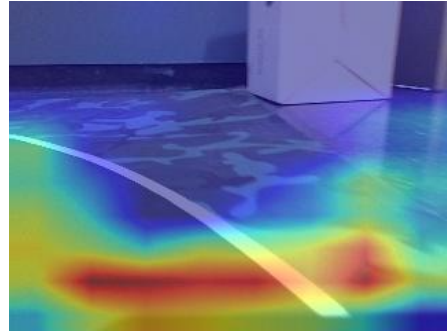
5.4.1 RGB Camera Model

In the model that uses only the RGB camera, we observe that the attention tends to focus both on regions very close to and far away from the vehicle when driving in a straight line. We hypothesize that the attention in the distant part of the image might be caused because the model is looking if tall objects are present, which are likely to appear in the upper region of the image. The focus near the bottom of the image may be related to the remote control car ability to stop quickly, obstacles detected in this region will require immediate braking (see Figure 14 (a)).

Additionally, when the vehicle is turning, the model mainly attends to the visible portion of the road ahead, often focusing on the white line that marks the road boundaries (see Figure 14 (b)).



(a) Straight road.



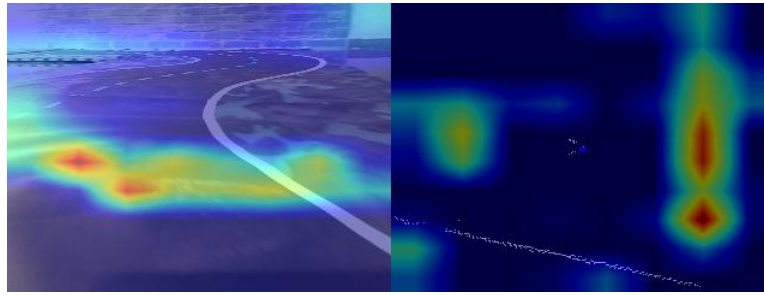
(b) Curve.

Figure 14: Grad-CAM visualizations for the RGB camera only model on (a) straight and (b) curved driving scenarios.

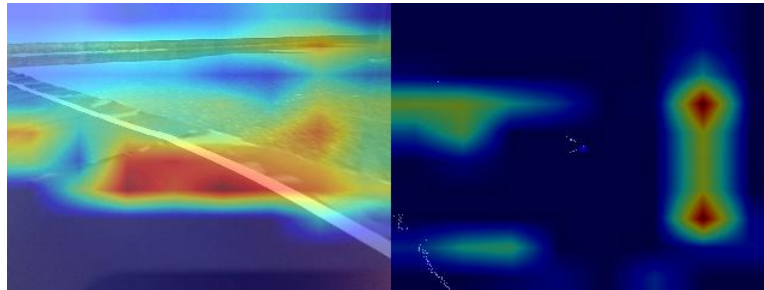
5.4.2 RGB Camera + LiDAR Model

In the model that incorporates both the RGB camera and LiDAR, we observe that attention is typically focused on the central region of the image (see Figures 15 (a) and (b)). This area contains relevant visual information for both straight-line driving, where potential obstacles can appear, and turning scenarios, where road boundaries or lane markings are often visible. In the LiDAR modality, the model generally attends to the region directly in front of the vehicle, which is appropriate for detecting obstacles.

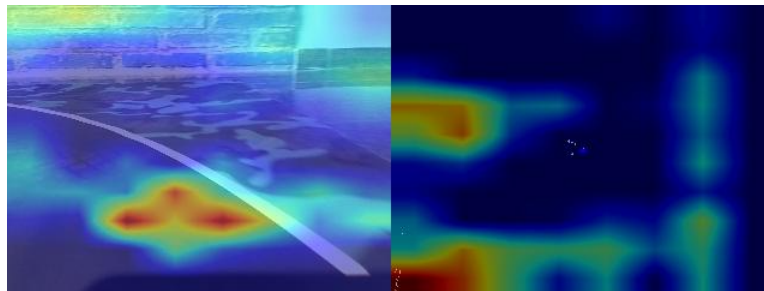
However, despite usually attending to informative regions, the model occasionally focuses on irrelevant areas, as shown in Figure 15 (c). In this case, the attention is directed toward the rear-right region relative to the car, which provides no useful information for the required left turn. These occasional errors in the focus areas may contribute to the model worse performance, with respect to the RGB camera model, in standard driving conditions that do not involve obstacle avoidance.



(a) Low-angle curve.



(b) Regular curve.



(c) Incorrect LiDAR attention.

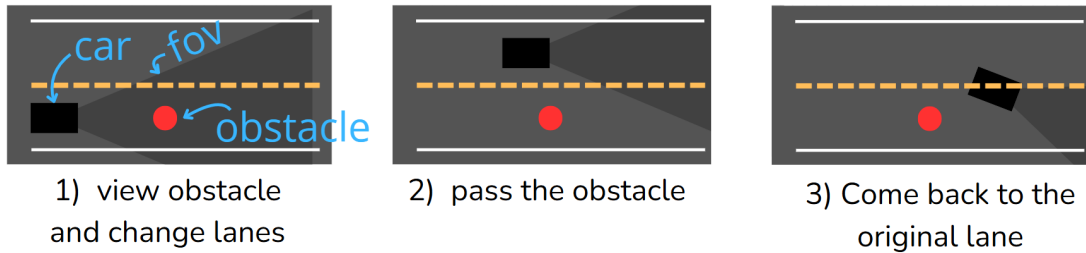
Figure 15: Grad-CAM visualizations for the RGB Camera + LiDAR model under different driving conditions: (a) low-angle curve, (b) regular curve and (c) incorrect attention focus in the lidar image.

6 Main problems

One of the main problems that we faced was making the car avoid obstacles. At the beginning we started by doing in the same rosbag avoid obstacles and stop obstacles mixed up. The ones to avoid where always small boxes while the ones to stop where big ones. However, we soon realized this approach had several problems. First, when to avoid and when to stop was a bit random and the model during inference didn't learn to avoid at all. This may be because when avoiding, there are actually 3 main phases:

1. When you see the obstacle and decide to change lanes.
2. When you continue in the other lane while you pass the object.

3. When you decide to come back to the correct lane.



The problem of avoiding obstacles with only one forward camera is that for the model, steps 2 and 3 look the same and basically you are just showing the model that it is okay to drive in the middle of the lane (as it has no other instruction that says otherwise, the camera doesn't see the obstacle anymore). The result of this is that the model got confused and learned to drive worse than if we didn't use that data.

There are 2 solutions that we thought to solve this issue. First, we could add some type of temporal information to the model, so that if it knows that in the previous step it changed lanes to avoid an obstacle then, it also learns that it should continue in this lane for a bit until it has passed the object and then come back. However, we decided to discard this idea as one of the teachers suggested this usually didn't give any good results.

The other idea was to use the LiDAR to give information to the model about when it has an object to pass or when it can come back to the other original lane. We tried this idea and it achieved actually good results. The trained model was able to avoid obstacles correctly and come back to the correct lane when it should based on the LiDAR information. As can be seen in 12 when having an object next to it, in the LiDAR appears a noticeable circle which helped the model to correctly evade it and not crash into it when coming back to lane.

7 What would we do different

7.1 What would we change if we started from scratch

If we were to start the project from scratch, we would certainly keep some of our previous strategies, but we would also change others. We believe that our approach of recording data across a wide range of circuits and lighting conditions was beneficial for the model, as diverse data is essential for effective generalization.

However, we could have started collecting data for low-light scenarios earlier in the project timeline. We initially focused on achieving strong performance in natural lighting scenario, both with and without reflections, as these conditions are more common. While this strategy allowed us to improve our model under well-lit environments, it limited the amount of data available for low-light conditions. Given the short timeframe of the project, this resulted in reduced performance in those scenarios. If we were to start again, we would collect data across all lighting conditions, including rare or extreme low-light cases, from the beginning. This would allow the model to extract useful information from a more varied dataset and generalize better.

An example of the effectiveness of collecting diverse data from the beginning was our decision

to use a variety of circuits during the data collection phase. This enabled the model to generalize well to unseen tracks, which is essential in real-world applications. Although we forgot to formally test this on the designated evaluation day, our model is capable of driving correctly on any new circuit constructed using the same base colors (black road with yellow and white lines), regardless of the track's shape. This can be clearly seen in video X, where our model is able to follow the white lines within the black segment not constructed from standard track pieces, even though this situation was not present in the training data.

Another potential improvement to our method would be to filter out the noise present in the images generated by the LiDAR sensor during inference. These images often contain significant noise, which may have negatively impacted the model's performance. However, during training, we do not consider this noise problematic, as it can act as a form of regularization and theoretically lead to the learning of more robust features. Additionally, future work could explore the use of Point Transformers or other point cloud processing architectures, which are better suited to exploit the inherent structure of this type of data and might improve the model performance.

7.2 What would we change if the vehicle had to drive around UAB

The main difference if the car were real and had to drive around the UAB campus, consequently real roads, is that it would not be feasible to do as many tests as we did with the small-scale vehicle. (In total, we trained X models for the small car.) In a real-world scenario, we would need some assurance that the model performs well before any on-road testing.

Much more data would be required before initiating real-world trials. For example, when we trained our first model with only one hour of data the results were poor. This was not a problem for the remote-controlled car, as we were in a controlled environment, but would be potentially dangerous in a real-world setting. Moreover, a proper validation and test set should be created to compute metrics that evaluate the model's performance prior to real-world deployment.

In addition, this data would allow us to identify potential failure cases without the need for physical testing. For example, the generated validation set could include scenarios where the model must execute emergency maneuvers, allowing us to assess its behavior in those situations.

Finally, when experimenting and looking for failure cases in the real world, the supervising driver should remain fully attentive and never lose the attention to what the car is doing. This is especially important since the real car would be traveling at 30 km/h or more, unlike the remote-controlled vehicle, which operated at significantly lower speeds implying a much higher level of risk.

8 Conclusions

In this project, an end-to-end pipeline for developing an autonomous driving model for a physical vehicle, going from manual data acquisition to training a CIL++ model has been developed. The core of the work was imitation learning, for which an extensive data was manually collected by having diverse drivers controlling the car through a variety of circuits, lighting conditions and obstacle scenarios to ensure model robustness. At the end, over 9 hours of driving data were collected to train the model to handle diverse situations going from "simple" lane-following to avoiding obstacles.

The final model demonstrated a strong capability to generalize the learned behaviors on new and unseen circuits successfully handling obstacles. This projects highlights the importance of a diverse and high-quality dataset in AI and Deep Learning models, and especially in imitation learning. Additionally, it proves the benefits of augmenting vision models with complementary sensor data such as a LiDAR to overcome inherent limitations of a system with only one camera building more capable and robust autonomous driving systems.