Autonomous RC Car platform for Research and Educational purposes

Reda Al batat
160602218
Diego Perez Liebana
MSc Artificial Intelligence

***Abstract*—Auto RC cars have been growing in popularity because of their great applications, however, currently there is no complete and extensive auto RC car platform. Here we start the development of an auto rc car, and show promising and exceptional results in terms of the configurable environment, accessible hardware, robust autonomous driving ability, network connection, object detection and recognition.**

I. Introduction

Self-driving cars has been growing in popularity Over the years have been growing in popularity consistently There has been great developments in self-driving cars recently, and has been growing in popularity, as they present many benefits such as road safety, traffic efficiency, cost of travel, and more as described in Zakharenko (2016); Howard and Dai (2014); Silberg et al. (2012); Levinson et al. (2011). Autonomous (auto) remote control (RC) cars have also been growing in popularity because of the wide range of applications it can be applied to and used for, which includes 1) Providing an additional testing tool to help the development full size self-driving cars in a form of a testing platform. 2) Helping researchers assess and experiment with different methods such as object detection and mapping more efficiently in a more dynamic and practical setting. 3) To be used as a practical and physical learning tool by individuals at any skill level.

1) There is no doubt that in a self driving car system, testing is one of the most important aspect in development. Currently there are only two main methods used for testing, using simulations and real-life tests. Chen et al. (2019, 2017). However, as described in Chen et al. (2019), simulations are not exceedingly effective as it makes it difficult to complete the migrations from the virtual simulations to the real physical environment, there exists a considerable gap. In the other hand, using real full-size cars for testing is expensive, time-consuming, and dangerous even in controlled environments. What is suggested is the possibility to use an auto RC car as an additional testing tool by providing a testing platform in physical miniature scale, to simulate and create exact real-life scenarios. To be used to evaluate and experiment with new algorithms and functionalities quickly and efficiently with low-cost on real-life environments. Which can be easily modified to create detailed specific real-life scenarios where there is high amount of variability, dynamic conditions, and interactions. An auto RC car platform, is a attractive and compelling combination of simulations and real-life testing.

2) As described in Ashton et al. (2019) an auto RC car platform can also be used to explore and assess different methods such as localization, object detection, environment mapping, and more. This has recently been demonstrated by Cho (2019b), where an auto RC car was used to test the proposed collision avoidance system to be used by full sized cars. However, they had to first go through creating and implementing an auto RC car to get to a stage where they can start focusing on the main proposed method. Even after making the auto RC car, it is mentioned how the work was limited by the auto RC car created, more specifically its inability to get a real-time video signal without increasing the CPU power. By providing a readily available auto RC car, such researchers do not need to spend time or large expenses up front on the project to create something that is very extensive, rather be able to fully focus on their proposed method. Furthermore the proposed research methods tested on the auto RC car, does not have to be for a self-driving cars, it will provide a great testbed for many different techniques and methods to be tested by any researcher.

3) An auto RC car platform can also be used as a learning tool by individuals at any skill level who are interested in the fields such as artificial intelligence, computer vision, robotics, or programming in general; using only one platform to learn at minimum the basics of these fields/topics. This can also be used in schools and colleges, as there is currently little exposure to technical fields such as the ones mentioned in schools and colleges. It is very beneficial to expose kids and teenagers to such topics who may find an interest in those areas at an earlier age, even at a very basic level, especially when it can be very interactive and engaging. There are some schools who use something similar, but are very limited by the current impractical, limited, and expensive prices of currently available auto RC car ”kits” and platforms.

There is currently no readily easily accessible, practical, and comprehensive development of an auto RC car platform to facilitate these applications efficiently and effectively to upstage this interest. Here we investigate and propose an Autonomous Remote Control (Auto RC) Car to contribute by **starting** the development of a comprehensive and expansive Auto RC Car platform that is accessible, inexpensive, and reliable with configurable and dynamic environment where the RC car can undertake multiple tasks.

II. Related work

There are a wide range of autonomous RC car projects, however, they are very limited in someway, they do not demonstrate any extensive and good performance functionalities, if they do well in one area, they are held back by another aspect. All of the current projects have common goals, avoid obstacles, and stay in lane. Most of them have very little interaction with the real world, are very limited to the environment they are put in and have a very fixed setting. Furthermore, they require lots of physical space to operate and most projects require very expensive hardware, which makes it very inaccessible for most. In this section, I will be reviewing and discussing all the latest autonomous RC car projects.

Since the nature of such projects involve multiple aspects which are done differently, I have split this section into four parts. A) Environment, the environment the RC Car is placed in. B) Hardware, the main hardware used to make up the car. C) Network connection, how the connection is handled between the RC car and where the main processing is done. D) Task(s), the different task(s) the auto RC car is made to perform.

A. Environment

Most current projects have very limited, fixed, and very simple environments, they contain only one road layout and only a few turns in one big road that takes a lot of physical space. The following projects Wood et al. (2012); Liniger et al. (2015); Shrivastava et al. (2019); Bechtel and Williams (2020); Sari and Cibooglu (2018) all use such environment, simple one-way road, with no intersections, just a few turns, and the road is relatively large, making it impractical to replicate or create compositions of road layouts that is useful to experiment with, they are also fixed (not configurable). Furthermore, some such as Chong (2017), just have one single gradual turn as the environment. In Fernandes et al. (2011); Ashton et al. (2019); Shrivastava et al. (2019) there has been an attempt to use smaller roads, however, limited because the RC car itself is large, hence requiring a much larger road. Having a large car can be beneficial because potentially, more sensors and hardware can be added easily. However, it is not worth the trade-off of physical space, as that is much more important as you can create much more detailed environments, and in most cases, all sensors are made in a variety of sizes to be used in small scale. Furthermore, it is also very important to have an environment that can be configurable so learnt models work in several environments and do not over-specialize to specific tracks, and not provide a variety.

B. Hardware components

Regarding the hardware components, it is noticed that multiple methods use very expensive components, mainly the microcomputer, where all computations are performed on. This is so the system can be stand-alone and provides convenience because of transferring all the sensors data to a more powerful processor like a computer introduces complications. Examples of this are Ashton et al. (2019); Shrivastava et al. (2019); Sari and Cibooglu (2018), they all use the NVIDIA Jetson, which is a powerful microcomputer. It can be used when a stand-alone system is needed, however, it is much more important to have all the hardware components be easily accessible to everyone by using minimal hardware and being relatively inexpensive. Additionally, for testing and research purposes, it is perfectly reasonable not to have a stand-alone system where all the computations are done on the RC car. Chong (2017); Bechtel and Williams (2020) had staying inexpensive in mind by using the widely used microcomputer such as the Raspberry Pi, which is very accessible and versatile. However, they were limited by the processing power of the Pi, because all the processing was performed on the Pi itself, which is not powerful enough, hence making their system not perform well. This will be discussed in more detail in section II-C.

Another important component is the camera, Bechtel and Williams (2020); Simmons et al. (2019) used the native camera of the Pi, which is very good as it is very accessible, where the most others use third-party USB cameras which makes it harder to replicate and reduces the accessibility and increases the cost for no significant benefits. Some use additionally to the microcomputer, a microcontroller, the Arduino, exclusively to control the motors, which is not good use of the substantial addition of the Arduino board.

It is very important for the RC Car to be accessible and easy to replicate and recreate, and we can see that most use very expensive or limited hardware with third-party components and do not utilise key functionalities of the microcomputer used, such as the wireless connectivity. For example, Ashton et al. (2019) report the total cost of the hardware to be \$2490, and Shrivastava et al. (2019) \$1082, which is extremely expensive to make it a practical or a good choice to replicate for most when compared to this project, where the total cost is £167.

C. Network connection

Regarding the network connection, Ashton et al. (2019); Shrivastava et al. (2019); Sari and Cibooglu (2018) have no connection established, because it is not needed, all computations are done on the RC car, which is made possible because of the expensive hardware used. However, this is also limiting, as when complex and more sophisticated practical methods are employed, it will require significantly more processing power compared to current tasks performed, which will make it unreasonable. Keeping all the computations on the car itself in this case is not ideal, even when using expensive hardware because of the limiting computing factor. A workaround to this is that some transfer the sensor data from the RC car to a more powerful unit, such as a computer, where all the computations and processing is done and then a signal is sent back to the RC car to tell the motors what to do. However, current methods have not established a consistent, fast, and reliable connection, most of them can only run a max of 30 FPS in very small resolutions. For example, Liniger et al. (2015); Chong (2017) can achieve a max of 25 FPS at 640x480 resolution, and Cho (2019a); Simmons et al. (2019) only 10

FPS. This is not practical, as the network connection itself becomes the bottleneck of the system, and additional processes that rely on the camera frames will be limited by it, having to wait for the frames. A consistent 30+ FPS is desirable to be effective, especially when there is lots of motion involved.

D. Task(s)

Most of the projects have the RC car perform only one task. For example, in Wood et al. (2012); Shrivastava et al. (2019) the only objective of the RC car is to drive on a one-lane road with some gradual turns. Liniger et al. (2015) has the same ability of Wood et al. (2012); Shrivastava et al. (2019), but can also avoid obstacles on the road, however, only focused on finding the optimal turn trajectories on a single track/road while avoiding other cars, which cannot be applied to any other road. Some do make the RC car perform multiple tasks, for example Ashton et al. (2019) manages to do localisation and mapping of the environment using LIDAR, which is great, as LIDAR is used in most full-size self-driving cars, however, they did not manage to make the RC car autonomous. Bechtel and Williams (2020) demonstrates ability to staying in lane with object detection. Chong (2017) did have some promising tasks for the car, such as the ability to detect if an obstacle is in the lane, it will move to another lane to overtake and then move back to the original lane using lane tracking. However, this was only reported using images, with no video demo, which does not show how well the overtaking or staying in lane works, they also mention themselves that their key aspect which was the lane detection did not work as well as they wanted.

As we can see, all current projects either do one task, or a few, but none perform multiple or a combination of tasks simultaneously. It is key to incorporate as many tasks for the RC car to do because an ideal auto RC car platform involves the car to perform multiple tasks constantly and be adaptable and reliable at all times, while still being able to handle any of the extra processing in real-time consistently.

E. Summary of related work

As discussed/shown, no current method is complete, if they do well in one aspect, they do not perform well in others. For example, simple environment, minimal tasks, uses expensive hardware, does not utilise the full power of the hardware or limited by the hardware, and the ones that use less powerful microprocessors and transfer sensor data to the computer, they end up having limited tasks, fixed environment or limited by the network connection. None show a complete and robust system. As we have seen, it is very challenging to develop a technically robust system whilst still having multiple well working functionalities and features because making an auto RC car is multidisciplinary.

Here an Auto RC Car system is presented with a configurable environment which include intersections and has a relatively small footprint. That uses inexpensive and easily accessible hardware, with the least amount of components as possible and use native hardware where applicable. A network connection that can consistently and reliably reach 30-60 FPS even at high resolutions. A robust and adaptable autonomous steering method that can handle intersections and multiple road configurations. Which can perform multiple tasks simultaneously such as traffic signs detection, traffic light detection, and obstacle avoidance in a configurable environment in real-time under heavy processing load without any hindrance in performance.

III. Methodology

In this section all the different aspects of the project are discussed and explained.

A. Hardware

It was very important to chose all hardware that was most accessible, easy to assemble, as well as being relatively inexpensive to make it more feasible for individuals and will require no or little hardware knowledge to quickly and efficiently get setup. Additionally, the size of the car also played a big role, having it as small as possible was essential to make it as practical as possible, meaning the environment itself could be smaller, hence taking up minimal physical space to operate. However, having all this in mind, all the components used needed to perform to a level to not be the bottleneck themselves, rather have the software implemented be the bottleneck to provide full freedom and flexibility on the software side, and not be limited by the hardware even in simple tasks. Table I shows a list of all the hardware components used alongside the prices of when this paper was written.

TABLE I
Hardware components list

Component	Purpose	Cost
Raspberry Pi 4 B - 8GB	Microcomputer	£73.50
STS-Pi Roving Robot	Chassis & Motors	£27.90
Explorer HAT Pro	H-Bridge & Analog input	£20.40
Raspberry Pi Camera v2.1	Camera	£24.21
Sharp IR GP2Y0A41SK0F	Distance sensor	£8.99
Poweradd EnergyCell 5000mAh	Power supply	£12.00
	Total cost	£167.00

As we can see the total cost is significantly lower than the projects mentioned in the previous section. Additionally the main components that were chosen are widely used and well supported by the RC car community. This is also open for easy modifications and additions. Here we go over the main reasons why the some of the key components were chosen.

1) Raspberry Pi: The Pi was chosen because it is currently one of the most used microcomputer, and here the latest version model 4 is used with the highest possible RAM of 8 GB. The Pi has a very big community, great support and documentation, and is very versatile. Most importantly it comes already with a built-in wireless adapter that supports 5GHz connections, which plays a big role in transferring data from and to the Pi in fast speeds, which will be discussed in more detail in section III-B.

2) Explorer HAT Pro: Allows both the motors and the infrared sensor (IS) to interface with the Pi seamlessly using only one single component, where normally two separate components are used. It acts like an H-Bridge(needed for the motors), and provides analog inputs(needed for the IS).

3) PiCamera: It is the native camera for the Pi, so that was an obvious choice, as it makes it more accessible and reliable, it is made to work with Pi for optimal performance by fully utilising the GPU, and not taking any processing power from the limited CPU.

4) Sharp IR: The distance sensor (DS) will be used to detect how far an obstacle is in front of the car. For the DS, the two main options that are widely used in the RC Car community are the IS or the ultrasonic sensor (US). The IS was chosen because it is much more reliable and stable than the US, as it uses light rather than sound to calculate the distance, which makes it much more reliable and consistent to obtain the distance and less likely to block the program run-time. Because for the US, the detection of the bounce (echo) was at times slower than the program run-time, blocking the program until the sound bounce was detected. To overcome this issue some introduce a fixed delay to compensate for the sound wave speed, however, adding a delay into the system is not desirable. The IR is however five times the cost of an US but in this case, the reliability and consistency of a DS is extremely important and the IR is still inexpensive compared to the other components.

B. Network connections

Using a low-cost microcomputer with limited processing power means you are not able to do a lot of computations, for example, as done in Liniger et al. (2015); Chong (2017), they are very limited by the Pi's internal hardware, which plays a big role on how good your results, it does not allow for elaborate systems to be implemented, making it a big hindrance to the overall project and impractical to do all the processing on the Pi.

To not be limited by the amount of computations that can be done, projects such as Simmons et al. (2019); Cho (2019a) transfer all the sensor data from the Pi(car) to the computer, to perform all the computations needed, and then send back a signal to the car telling the motors what to do. This provides significantly more processing power as now you are not limited by the Pi's hardware, namely the low-end CPU and GPU, which allows much more freedom on what can be done, and high consuming computational methods and processes can be implemented with no issues and limitations. Which plays a big role on how sophisticated your system or project can be.

However, the projects that do use the Pi and transfer the sensor data to the computer, struggle to get consistent 30 FPS, such as Simmons et al. (2019); Cho (2019a) report being able to only do 10 FPS. Which makes the network connection the bottleneck of the whole system, which in-turn means that they are not fully able to utilise the processing power of the computer. The main reason for this, is that they all use the TCP protocol to transfer the camera frames to the computer, which creates an overhead, since TCP ensures all frames are sent and received successfully, which blocks the whole system until that frame is tracked and re-sent, this delay accumulates each second, causing low overall FPS.

To overcome this issue, here the UDP protocol is used instead, it is a connection-less connection and does not have the overhead that TCP has. However, any lost packets will be lost and need to be handled manually, but this not an issue, because any packets that are not received successfully can simply be skipped completely and be ready to receive the next frame, losing one or a few frames is insignificant as the next frame will come straight after, allowing to maintain 30 FPS, and as we will see in section IV-A the ”skipping” of frames does not happen as often.

However, in addition to requiring to handle all packets (including unsuccessful ones) manually, using UDP introduces another problem, it has a packet size limit of 65,535 bytes, and sometimes frames sent to the computer exceeds this limit. To overcome this issue, before sending the frame to the computer, first the size of the frame in bytes is sent which is a 32-bit integer that is 4 bytes in size. Once the computer receives the size of the frame, a check is made if the frame exceeds the UDP limit or not. If it does, the frame is split in half, and sent as two separate packets, and since the computer knows exactly how many bytes to expect from the first signal, it can determine how many bytes each half is to be received , once both halves are received successfully, it is then reconstructed to the full frame. Unnecessary splitting is avoided if the frame size is under the UDP limit.

This connection setup turned out to work very well, it did not only allow for consistent 30 FPS, but also 60 FPS as we will see in section IV-A. For the distance sensor and motor control signals, TCP was used and it worked well, because in those cases only 4 bytes (float32, int32) was needed to be sent and received.

C. Environment

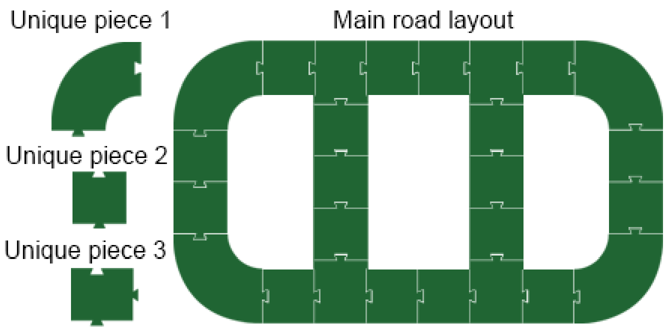

Fig. 2. Road pieces and full size road layout

It was very important for the environment to be small, configurable, easy to setup, which allows for expandability. Inspired by jigsaw puzzles, the road is made up of jigsaw puzzle pieces that fit together to make up different road layouts. Fig. 2 illustrates this, using only three unique pieces,

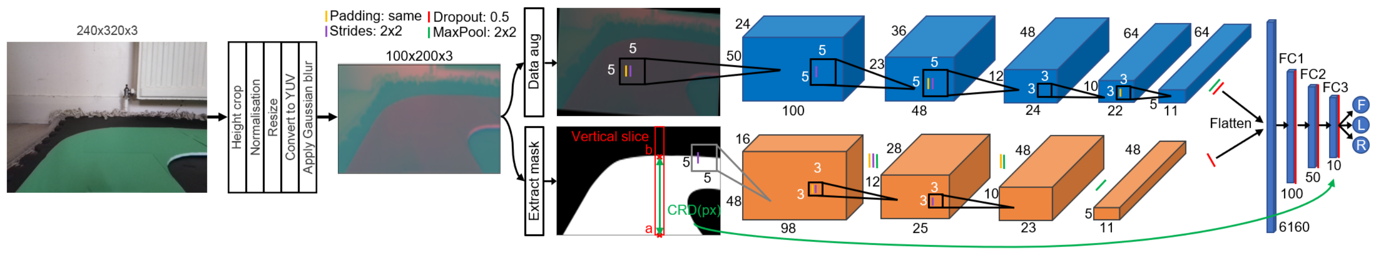

Fig. 1. Processing pipeline and CNN architectures for autonomous driving.

multiple road layouts can be created very easily. This turned out to be a very good solution and worked very well in practise.

D. Autonomous driving

Autonomous steering refers to how the car stays in the road while moving forward. This aspect is very important as it plays a big role on how well the auto RC Car is overall. There are many different methods used to make the car autonomous. Some use a proportional–integral–derivative controller (PIDC) Cho (2019b) with a lane detection algorithm. Based on the centre of the lane, the PIDC is used to determine the desired angle of steering. This is a very responsive way as it is all based on a manually fine-tuned formula. However, its performance heavily depends on how well the lane detection algorithm is. If different kind of lanes are introduced or there is an obstruction of a section of the lane, the algorithm might not be able to detect the lanes accurately, making it unreliable and not generalisable.

Some use purely mathematical based methods Liniger et al. (2015); Wood et al. (2012), or only computer vision (CV) based methods Chong (2017); Shrivastava et al. (2019). They are better but still very limited, if any change in the environment is made, it will fail as it is optimized and created only for a fixed environment and a specific task. In the other hand, there are some methods such as Simmons et al. (2019) that treat the problem as a classification problem and use a convolutional neural network (CNN) to predict the steering for each frame, which show promising results, the car is able to navigate around the road without using hand-crafted features or ”rules” to follow. However, they show signs of overfitting, not able to handle intersections and again is only able to tackle one road layout and environment.

Here a combination is used, a CNN and CV techniques to create a model that is capable of predicting the steering for each frame and generalise to different road layouts. The CNN architecture is inspired by full size autonomous cars, more specifically the the Nvidia end-to-end model, one plus side is that it was made in mind to work with minimal training data which is desirable for minimal data collection. Unlike current auto RC Car methods, rather than just having the frame as input, two additional inputs are introduced; the road mask and the central road distance (CRD) to help the model even further and give it more ”sensors” and information about the road to make a better decisions on what to do for each situation. The full architecture and how all the different inputs work together is illustrated in Fig. 1.

First the training sample goes through pre-processing.

1) Height crop: The image is cropped from the top 95 pixels down to get rid off most of the background, this is so the model will not rely on its surroundings to learn unnecessary features, rather focus on the road and its characteristics for decisions.

2) Normalisation: All pixel values are normalised to the [0, 1] range.

3) Resize: The image is resized to 100x200 (HxW), this is to allow for reduction in complexity of model and training time, whilst still maintaining all details// in the image.

4) Convert to YUV: The image is converted from RGB to the YUV colour space. YUV focuses more on the luminance which reduces the significance of the colour of the road. It allows to focus more on the shapes and characteristics of the road, which will be more useful for the model to learn and will make it more generalisable to different roads. Furthermore, YUV uses less information to represent the image than RGB without significant loss in quality as shown in Podpora et al. (2014).

5) Gaussian blur: Used to remove noise to reduce learning specific fine-grained details of the road. A kernel size of (3,3) was used.

The processed image then goes through the two CNNs. As we can see from Fig. 1, for the ”main” top CNN (1), random data augmentation is applied on the fly, which include a rotation range of 4 degrees and a brightness range of [0.6, 1.32]. For the second bottom CNN (2), the mask of the road is obtained using a lower and upper limits of all YUV channels on the pre-processed image, which is then fed into an architecture inspired by AlexNet Krizhevsky et al. (2012), but a much simplified version is used as the image is binary and smaller in size. At the end, both CNNs are flattened and concatenated together, and fed into the the fully connected (FC) layers. Finally, at FC3, the CRD is concatenated just before the final output layer. The CRD is obtained using the mask used for the second CNN. Looking at Fig. 1, a vertical slice from the middle of the frame is taken, and the CRD is from point a to point b, where point b y-coordinate is the first white pixel from the top in the vertical slice highlighted in a red box.

Regarding the training, the data was collected by driving the car manually using the keyboard around only the main road

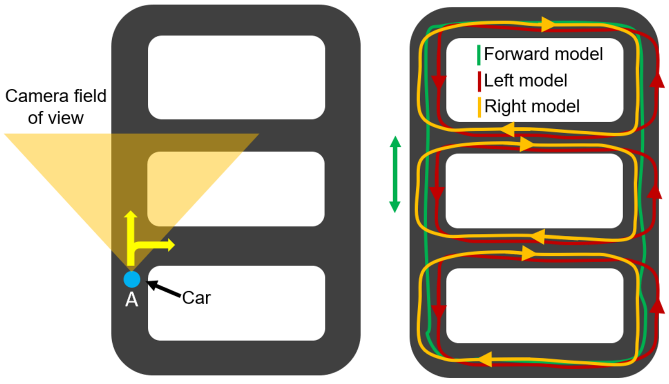

Fig. 3. Data collection driving paths

layout as shown in Fig. 2, where for each pressed key (forward, left, right) the frame was recorded. The data collection (DC) was only done for 5 minutes. However, it was quickly realised that having intersections in the road is problematic, and using one model was not feasible. Looking at Fig. 3, considering when the car is at point A, if in the DC the car is made to go forward then the corresponding label for that position would be forward. In the other hand if turned right, then that same frame would have a label of right. This creates a conflict as the same input has two different labels, which will not allow the model to learn if it should go forward or right.

To overcome this issue, the data collection was done in three stages and three separate trained models was used, one for each direction (forward, left, right). The paths taken for the data collection is shown in Fig. 3 on the right. This way, all ”states” of the road are obtained. The exact model architecture in Fig. 1 is then trained on three separate data sets, one for each direction,, producing three separate trained models. Then the chosen model to make predictions dictates what direction the car should take in the next intersection, which is communicated to the car using the directional signs on the road. This worked very well, and provided the car ability to deal with intersection very nicely with just the expense of multiple models to be trained separately rather than just one.

E. Directional signs detection and recognition

To recognise the signs, a sign classifier was trained to classify patches of traffic signs. The classifier is a CNN, trained on a German traffic data set Stallkamp et al. (2011). It consists of 43 classes, which most importantly include the directional signs forward, left, and right. The LeNet model LeCun et al. (1998) was used as a starting skeleton, where it needed an increase in the number of filters and addition of two convolutional layers to allow the model to learn the signs more effectively as it was suffering from under-fitting using the ”vanilla” LeNet model. It was very important to still keep the model simple to reduce the time taken to make predictions.

Minimal processing is done on the images, only converted to gray-scale and used histogram equalization to increase the contrast so the shapes of each sign is more pronounced on the solid background of the sign.

To overcome overfitting, several data augmentation techniques was used: width and height shifts of 0.1%, zoom, shear, and rotation range of 0.2%, 0.1%, and 10 degrees respectively.

The classifier was trained on the patches of the signs, so the signs needed to be detected in the frame. Sliding window could be used, however, it will be computationally expensive especially for different scales. We can use the fact that the directional signs are circular, where if we can detect circles on the frame, it is a potential sign. The OpenCV library conveniently has a function to do just that (HoughCircles). It takes in four key arguments, the minimum and maximum radius of the circle, 7 and 26 was used. The other two are method specific arguments, p1 and p2. After multiple experiments and fine-tuning the best values obtained for p1 and p2 were 386 and 35. Once a circle is detected, it is then cropped and fed into the classifier for classification. To make this method robust, consistent, and avoid false detections, the same sign needed to be detected and correctly classified in two consecutive frames to be valid, which worked very well in practise.

F. Traffic light (TL) detection and recognition

To detect the TLs, a Haar cascade classifier was trained using 167 positive samples and 688 negative samples in 32 iterations. Where the positive samples were 25x40 cropped patches of the TL and negative samples images were with no TL present. Equal number of red light and green light samples was obtained to ensure that the classifier can detect both states of the TL. Once the the TL was detected, to determine which light was on, the TL was split in half horizontally (yellow light was treated as red) and two methods were tried, 1) Checking which half had the highest average pixel intensity, 2) Checking which half contained the brightest pixel. Method 2 was chosen as it was better, and it worked surprisingly well to detect which light was on.

G. Obstacle avoidance

To avoid colliding into obstacles in front of the car, the IR was used to get a distance reading every 0.2 seconds. To reduce noisy data and inaccuracies of the sensor, 10 distance readings was used to obtain an average for the final distance.

IV. Results and Experiments

In this section, the main aspects of the projects will be tested and evaluated, and provide an insight and show results of appropriate experiments.

A. Network connection

To test the consistency and reliability of the main camera UDP network connection, a connection was established for 10 minutes and recorded the FPS, during the 10 minutes. The camera was pointed to a screen that was running the same video for each experiment to keep it consistent and to allow the image size (number of bytes) to change as the frames were being sent and received, rather than being stationary where the

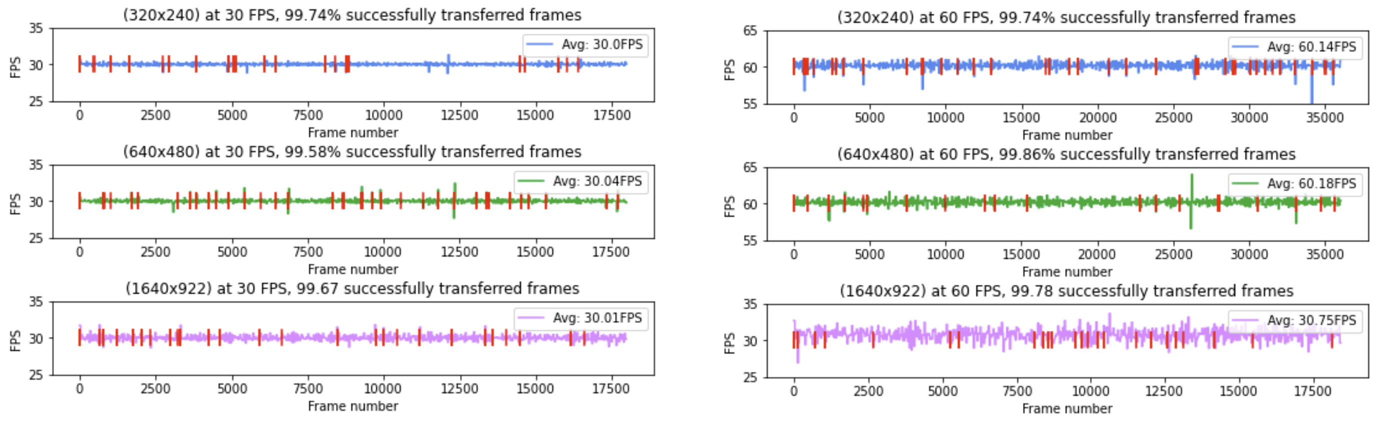

Fig. 4. Network connections for 30 FPS (left) and 60 FPS (right)

similar/static frame will be sent. This was done for 30 and 60 FPS for three different resolutions. The results are shown in Fig. 4.

For the 30 FPS connections (left), we can see a consistent 30 FPS was easily maintained in all three resolutions with no issues. The vertical bars show when a frame was lost or skipped as discussed previously. The lost frames are very infrequent and a single or a couple frames (thicker vertical line) lost is not an issue in the overall connection with 99% of the frames being successfully transferred. Mostly only a single frame is lost every 10-20 seconds, which will not cause any problems.

For when 60 FPS is used, we see some interesting results. From Fig. 4, we can see that for the lower resolutions (320x240 and 640x480) we are able to get a consistent 60 FPS with no issues at all, and see the similar random pattern of lost frames as the 30 FPS tests, which might be due to the nature of a WiFi connection, with lots of interference. However, When looking at the higher resolution (1640x922), we can see that 60 FPS is not achieved, but a consistent 30 FPS is achieved instead. This is strange, as the number of frames received is half than the others at 17500 (looking at the x-axis) while still having a 99% success rate. To further investigate this, the delay in the connection was obtained, and it was found that there was an average of 30 ms delay for the higher resolution compared to the 15 ms delay for the lower two resolution, this explains why a 30 FPS is achieved. Because of the large size of the frame, it takes longer to send all the bytes, but despite this, 30 FPS is still not a terrible result and is acceptable for that resolution.

B. Autonomous driving

Before arriving to the final network shown in Fig. 1 with multiple inputs, what was first tried was just the frame, which corresponds to the top architecture in Fig. 1. For model training, the following hyper-parameters (HPs) were used, validation split of 20%, batch size of 256, categorical cross-entropy loss, the Adam optimiser with a fixed learning rate (LR) of 0.0001. 4000 forward samples were removed from the total data set to make the ratios of of each class more balanced while still having more forward samples, this is to encourage the car to drive in the centre of the lane with minimal turns to avoid jittering. The model was trained for 1000 epochs.

To evaluate the RC cars autonomous ability, the car was first driven in a specific path that covers the whole main road layout, this was used as a benchmark, which was considered as faultless driving, this also emulates a test set. Then the car was made to autonomously follow the exact same path using the directional signs. The accuracy was obtained by calculating the number of wrong samples/turns made when following that path. The training accuracy obtained was 93.43%, with a test accuracy of 91.20%. This worked well for the most part, however, in certain situations and locations on the road, especially the curves, the car scrambled and lead it to going off course. Although this did not happen so often, it was not reliable or consistent.

To attempt to increase the accuracy of the model, what was tried was to provide the model with additional information about the road to help with the steering using feature extraction. The first feature that was experimented with was the road mask as shown at the bottom of Fig. 1. The model architecture for the road mask was obtained by first experimenting and training the model on its own, and only then used in combination. For the combined network, the same HPs were used, only differences were the number of epochs, 1500 was used instead, as the model required more time to converge, and a LR of 0.00001 for the last 500 epochs. The model with the frame combined with the road mask resulted in significantly better training accuracy of 98.71% with a test accuracy of 95.04%. There is slightly more overfitting present, but the model performed very well in practise, with minimal mistakes. The 5% error in accuracy on the test set is small enough for the car to be able to re-adjust and re-establish good position before going too far off course, which made the car able to handle all road layout configurations very well.

Another feature that was introduced to possibly increase the performance of the model even further was the CRD, which is shown in Fig. 1. After multiple experiments with different

HPs and configurations, the addition of the CRD made little or no noticeable change in the model performance. This might be because the CRD is dependant on how well the road mask is generated. The mask is not extracted correctly 100% of the time, and if there is even a slight mistake in the road mask, the CRD in-turn will have a further error. However, to overcome this and to possibly make the CRD have more impact on the performance of the model, it may be given more weight when used for the final prediction. Unfortunately this was not attempted due to time constraints.

C. Directional signs detection and recognition

The sign detector, which can also be said to be a circle detector, was tested using various frames that included multiple shapes, triangles, rectangles, hexagons, and circles at different sizes, to ensure it was only detecting ”perfect” round circles(the signs) and not just any ”circular looking” shape. An accuracy metric was obtained by calculating the correct detections in all the frames with the various shapes. The detection in each frame was only considered to be correct if all circles were detected with no miss-detections anywhere else in the frame. This provided a detection accuracy of 91.91%, which is not very reliable but it was good enough for this application. Because as the car moves along the road, it will have multiple chances to detect the circle as it gets closer because the size, lighting, and orientation changes. This coupled with the fact that a sign needs to be detected and recognised twice to be valid worked very well.

The classifier for the sign recognition was trained using a validation and test split of 10%, 25% respectively, LR of 0.0001, the Adam optimizer, categorical cross-entropy as the loss function, batch size of 128, After training for 200 epochs, a training accuracy of 98.52%. was achieved with a test accuracy of 97.71%, which is a great result with minimal overfitting. This worked great in practise, the signs were able to be detected even in very low level light conditions and at different orientations.

D. Traffic light (TL) detection and recognition

The TL detection and recognition was tested in a similar way to the sign detection. For the detection, multiple frames with and without a TL were taken, and the accuracy was measured the same way. The accuracy obtained was 86.78%, which is relatively low. The low detection accuracy was overcome by using the same ”rule” that was introduced in the sign detection, where the same TL needed to be detected twice to be valid. This alone increased the TL detection accuracy to 94.64%.

As well with the sign detection and recognition, this was tested in real-time and each detector and recognition worked very in practise, the signs and TLs were detected with acceptable consistency. However, ideally, rather than having the detection and recognition be separate processes that need to be fine-tuned on their own, it would be much more ideal to have an end-to-end method such as using the fast YOLO object detector Redmon and Farhadi (2018), where it can be trained to detect and recognise both the signs and TLs at the same time, using one single model.

E. IR sensor

To test the accuracy of the IR sensor, the obtained distances from the sensor was compared to real actual distances. Table II

TABLE II
IR SENSOR DISTANCES COMPARED TO REAL DISTANCES

Real dist (cm)	4.00	5.00	6.00	7.00	8.00	9.00	10.00
IR sensor (cm)	5.21	6.93	8.11	9.56	10.19	11.7	12.25

We can see that the IR sensor is not extremely accurate. In other applications this might not be tolerable as it is within 2 cm of the actual distance. However, in this case, the relative distance, as long it is consistent(which it is) will be sufficient, because only the rough distance is required and it is only needed to choose a reasonable relative distance to not collide with obstacles in front of the car.

F. Multi-threading

When all of these tasks and processes ran simultaneously, each process introduced a delay due to its computation time, which accumulated to produce a significant delay in the whole system. This prevents achieving a consistent FPS because the computer is not ready to receive the next frame in time to maintain a consistent FPS. To solve this issue, multithreading was used for each different process, such as the auto driving, sign and TL detectors, and mainly a thread was used for each of the three connections, the camera, sensor, and controller connections. This solved the issue, and the all processes could be ran at the same time with no issues with consistent FPS.

V. Conclusion

The project as it is right now, it is still not 100% clear where such auto RC car platform fits in as it is a relatively new concept with ambitious, innovative, and unique applications discussed. However, a great successful start and an advancement of a comprehensive autonomous RC car platform was introduced and evaluated that showed excellent/promising results, which provides a great foundation to build upon to lead into the applications of an auto RC car. New concepts were introduced into the auto RC car ”realm”, such as having a small-scale configurable road, a robust autonomous steering method that can handle intersections, with integrated object detection and recognition and obstacles avoidance. Contributed with significant achievements such as accessible, minimal, inexpensive hardware setup. A network connection that can reliably and consistently achieve 30-60 FPS, where anyone with a Pi and camera can use. An adaptable and generalisable autonomous driving method that provides an excellent base for expansion and development.

VI. Future work

Potential next steps for the auto RC car platform to bring it closer to the practical applications, are to have a more realistic environment, using sculptures and real objects to imitate the real-world as detailed as possible. To create an out of the box product that is accessible for anyone that includes everything needed, including all hardware components and software with learning tools provided if required.

References

Jason Louis Ashton, Myles Emmolo Spencer, and Sean R Hunt. Autonomous rc car platform. 2019.

Michael Bechtel and Andrew Williams. Deeptaxi: Teaching an autonomous car with social scaffolding. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 131–132, 2020.

Shitao Chen, Jinghao Shang, Songyi Zhang, and Nanning Zheng. Cognitive map-based model: Toward a developmental framework for self-driving cars. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

Shitao Chen, Yu Chen, Songyi Zhang, and Nanning Zheng. A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. *IEEE Transactions on Intelligent Vehicles*, 4(3):425–436, 2019.

Myeon-gyun Cho. A study on the obstacle recognition for autonomous driving rc car using lidar and thermal infrared camera. In *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 544–546. IEEE, 2019a.

Myeon-gyun Cho. A study on the obstacle recognition for autonomous driving rc car using lidar and thermal infrared camera. In *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 544–546. IEEE, 2019b.

Wen Yang Chong. *Autonomous RC Car Control Using Computer Vision*. PhD thesis, UTAR, 2017.

Chandra Fernandes, Kok Yew Ng, and Boon How Khoo. Development of a convenient wireless control of an autonomous vehicle using apple ios sdk. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 1025–1029. IEEE, 2011.

Daniel Howard and Danielle Dai. Public perceptions of self-driving cars: The case of berkeley, california. In *Transportation research board 93rd annual meeting*, volume 14, pages 1–16, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5): 628–647, 2015.

Michal Podpora, Grzegorz Korba´s, and Aleksandra Kawala-Janik. Yuv vs rgb – choosing a color space for human-machine interaction. *Annals of Computer Science and Information Systems*, Vol. 3, 09 2014. doi: 10.15439/2014F206.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Aysegul Sari and Mertcan Cibooglu. Traffic sign detection and recognition system for autonomous rc cars. In *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, pages 1–5. IEEE, 2018.

Kakul Shrivastava, Maruthi S Inukonda, and Sparsh Mittal. Hardware-software stack for an rc car for testing autonomous driving algorithms. 2019.

Gary Silberg, Richard Wallace, G Matuszak, J Plessers, C Brower, and Deepak Subramanian. Self-driving cars: The next revolution. *White paper, KPMG LLP & Center of Automotive Research*, page 36, 2012.

B. Simmons, P. Adwani, H. Pham, Y. Alhuthaifi, and A. Wolek. Training a remote-control car to autonomously lane-follow using end-to-end neural networks. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2019.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

Tony Wood, Peyman Mohajerin Esfahani, and John Lygeros. Hybrid modelling and reachability on autonomous rc-cars. *IFAC Proceedings Volumes*, 45(9):430–435, 2012.

Roman Zakharenko. Self-driving cars will change cities. *Regional Science and Urban Economics*, 61:26–37, 2016.