Car Plate Recognition Using Machine Learning

Mohamed Al-Mheiri
Department of Computer Engineering
University of Sharjah
Sharjah, United Arab Emirates
u17105029@sharjah.ac.ae

Omar Kais
Department of Computer Engineering
University of Sharjah
Sharjah, United Arab Emirates
u16105933@sharjah.ac.ae

Talal Bonny
Department of Computer Engineering
University of Sharjah
Sharjah, United Arab Emirates
tbonny@sharjah.ac.ae

2022 Advances in Science and Engineering Technology International Conferences (ASET) | 978-1-6654-1801-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASET53988.2022.9734830

***Abstract*—Security has consistently been a significant worry for mankind. Today we have video surveillance cameras in schools, hospitals and every other public place that help keep these spaces secure. This is also including places with vehicles such as parking spaces and garages, and it would be embedded with a security guard that would also help monitor and verify entrance of any incoming vehicle, as well as controlling the opening and closing of the gate.**

The project will aim to develop a smart car plate recognition device that can monitor and survey the area, as well as detect and analyze vehicle license plates. A sensor will detect the incoming vehicle, then a camera will take a screenshot of the front of the vehicle with the license plate. The license plate is scanned then checked whether it is registered to determine whether it is allowed or denied entry, and the device will troubleshoot by sending SMS to a fixed phone number regarding any issues.

We will use a supervised Machine Learning Optical Character Recognition model known as Tesseract AI. This pre-trained, multi-language AI will detect and extract the numbers and letters on the license plate. Before this process starts, we will clean the image of any noise by performing changes to the original image, such as switching to grayscale and brightening, in-order to increase the accuracy of the OCR and minimize error. These extracted numbers and letters will then be checked within the database one entry after the other until it detects a match.

This device will be a direct upgrade over the traditional system of simply including a CCTV camera and a guard as the device will operate independently to a point that no human input is required and will require no installation of on-site servers or setup of databases, thus saving manpower and reducing cost and complexity.

Keywords—Embedded system, Car plate recognition, Machine Learning, CCTV, OCR.

I. Introduction

With the rise of theft, fraud and other criminal activity, security becomes a legitimate concern [1]. One must tread carefully and constantly watch for any illegal activity that could potentially harm one's self or others, as well as cause damage to property. In many places, surveillance cameras play a pivotal role in keeping areas safe from any illegal activity, such that criminals tend to stay away from surveillance-dominated areas, and if any criminal activity happens then evidence can be recovered from the cameras which would be used to report to the appropriate authority.

To name an example, Scotland Yard reported a 95% usage of surveillance cameras as evidence for murder cases in 2009. As such, 62% of UK's residents have become in-favor of including CCTV surveillance in their local area and homes [2].

Video surveillance is one of the applications that can be developed using an object recognition system [3] [5]. Object recognition is a broad term that can include many sub-categories such as face recognition, human gait recognition, road signs recognition, etc. It is an algorithm that tries to recognize and identify an object in a scene image or in a sequence of images (videos) [6] [7].

In 2012, the University of North Carolina conducted a study to analyze how effective surveillance is at keeping criminals away (Fig. 1). They found, by interviewing both male and female burglars, that surveillance cameras are rated first as the most effective deterrent in comparison to other deterrents [4].

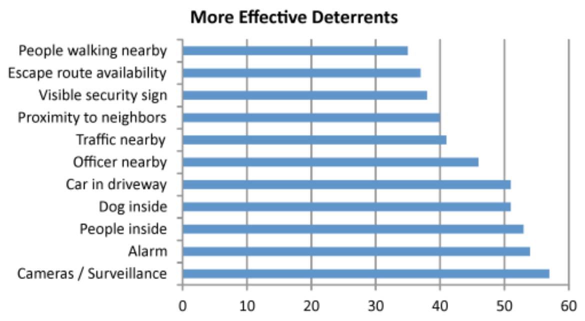

Fig. 1. Perception of deterrents according to burglars in North Carolina [8].

Security cameras are also used in plenty of vehicle-specific areas, such as parking lots, garages, or vehicle service buildings. Although cameras are effective at deterring criminals, they are not enough. Typically, a guard or a doorman is combined with the camera to survey the area. To add, the guard would need to be present in-order to control the gate, if any. The purpose of the project is to implement a smart surveillance camera system that not only survey the area, but also utilize Machine Learning to identify license plates on vehicles to allow or deny entry to a private or public garage or parking space, then open or close the gate accordingly. Processing the data can be done using CPU/GPU [9] [10], FPGA [5], Microcontroller or Raspberry Pi. The device will also use LED lights to inform the driver of the state of the device.

The system will mainly be used by garage and parking space owners who feel like they need to keep their properties secure. The car plate recognition system will take unlabeled images as inputs and detecting car plates. The recognition process follows two steps: detecting the license plates and recognizing the text within the bounding box of each car plate. Then, we will trigger an event based on the detected car (e.g. opening the garage door when it detects that your car has arrived home). The system will overcome the need of human input from a guard or a doorman and could operate independently.

II. Proposed System

This section explains in detail how the proposed system works. The proposed system consists of hardware and software parts.

A. *Hardware Parts*

The hardware parts consist of a Raspberry Pi computer, a camera and two ultrasonic sensors. All these parts work together to complete the device (Fig. 2).

- Raspberry Pi: This is a small, embedded device that we can program to do specific tasks and can also connect itself to other components to control them as well. It is incredibly popular due to its simplicity and accessibility, as well as the capabilities it harbors. This device is commonly used for a lot of different projects and purposes. It comes pre-loaded with a Linux-based OS and even has specific components constructed just to plug-in and communicate with easily. The version we use is the Raspberry Pi 4 with 4GB of RAM as we believe it is sufficient enough and will reduce expenses. We'll use this embedded computer as our main component and the heart of the car plate device.
- Camera: The camera will be used to record and survey the entry area for any activity, as well as take unlabeled images for when a vehicle arrives in proximity of the car plate device. The resolution will need to be at the least 1080p for both video and image, and it will need to be compatible and programmable with the Raspberry Pi.
- Ultrasonic sensor: The Ultrasonic sensor uses a series of ultrasonic waves sent at high frequencies that come out of the emitter end of the sensor. The waves will reflect back to the sensor if there is an obstacle and will be picked up by the receiver end. The time it takes for these waves to reach the receiver end of the sensor can then be used to calculate the distance. We will need our sensor to have a minimum range of 3 meters, and we will use it to detect the incoming vehicle as to inform the device to start the process of taking and processing the image of the vehicle's front along with its license plate.

B. *Software Parts*

There are also software parts that are used in the device to help work with the hardware in-order to achieve the requirements.

- Python: This is the primary programming language used to develop the project. It is a high-level, object-oriented programming language with many supported libraries, and it is a lot more accessible and easier than C++. It is also compatible with many mainstream OS' but works even smoother on Linux-based devices such as the Raspberry Pi. For this project, we will use the latest version; Python 3.9, and we will also use the OpenCV library with a custom algorithm to make use of its features. This library is an open-source computer vision and machine learning software library to be used in conjunction with the OCR.
- MongoDB: This is a flexible database to be used to store all information related to authorized and non-authorized vehicles, such as owners, plate numbers, type of vehicles and so on. MongoDB is a free-to-use, document database with drivers that support 10+ programming languages, and among them is the Python language. The database will be implemented along with the server to run along with the virtual machine in the cloud, and as the device takes an image and process it, it will access the cloud with the VM that contains the server code, which accesses the database to confirm the existence of the license plate number and verify authorization.
- Virtual Machine: A virtual machine is specialized software that can emulate a computer system. This virtual machine consists of its own Operating System and applications in the physical device. However, it doesn't directly interact with the physical device, leaving it encapsuled. The virtual machine will help host our DB and server code, and also enhance the security of our device as VM's are capable of easy backup and restoration in-case malicious software is detected and the VM is compromised. The VM itself will be hosted in a Cloud to further enhance the security and flexibility.
- OCR: The Optical Character Recognition engine will handle detecting the characters and numbers on the license plate during the image processing segment. The OCR requires Python compatibility and multi-language support. The OCR is a form of supervised Machine Learning, which means it will constantly improve its accuracy the more images it processes. We are going to also modify the images captured to further increase the accuracy of the OCR. Since license plates in the UAE have both Arabic and English characters, we are going to utilize the OCR's multi-language ability to detect the license plate type depending on the city.
- Cloud: A Cloud storage is a form of data storage where the data is in logical pools through the Internet, rather than being in a physically-connected component such as a hard disk. It is typically managed by a hosting company that provides these clouds. We use the cloud to be able to store our VM, server and DB on it. This allows us the flexibility of changing, updating or removing parts of our software universally through all supported car plate devices, rather than doing it for each individual device. It also increases the security and protects the data from being lost if a device suddenly fails or otherwise.

C. *Miscellaneous Parts*

Lastly, there are also miscellaneous parts which are necessary to the function of the device, but they are fairly basic that there is no need to get into detail. These parts include the LED lights that are attached to the device, and we are using two; one red, and one green. After authorization, and when access is denied, the red LED will flash to signal to the driver of the denial. Otherwise, when access is granted, then the green LED will flash to signal to the driver of such and will also open the gate.

The device will also need power to run. A conventional power outlet (similar to that used for desktops) can be used. A battery can also be used if it is compatible with the power input plug on the device.

D. *Block Diagram*

The block diagram helps as a representation of how the entire system is designed in-terms of its parts and the connections. This can serve as a simple look into the device's design.

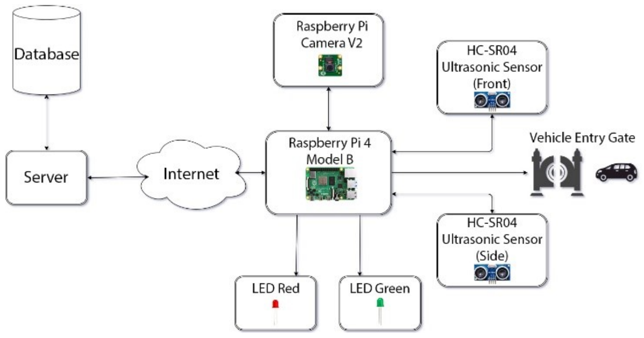

Fig. 2. Block diagram of the device.

III. Image Processing

Before the AI runs to extract the characters from the plate, the image would need to be modified to ensure AI accuracy is at its best. These modifications are done in a specific order, as each stage of modification complements the one after.

A. *Grayscale, Brightness and Blur*

We need to start off by detailing the characters on the plate for better detection. The first step is to convert the image into grayscale, which removes color and turns the image into shades of gray. This is important for later steps. Then, the brightness is increased by 50% so we can remove light shades of gray such as shadows on the plate and elsewhere, which will make detecting contours and characters more accurate. We then apply Gaussian Blur to smooth the image and remove any small shapes that could be picked up by the AI such as specks of dirt (Fig. 3).

B. *Blackhat and Close Morphological Operation*

Next, we apply Blackhat. The Blackhat operation will make black regions on top of white ones more prominent, which is very useful for UAE license plates due to white backgrounds. This process will help highlight the characters on the plate better. (Fig. 4)

The detailing of the characters is complete from this point on. Next, the location of the plate must be obtained.

For that, we start with applying the Close operation, which will expand the white regions then return them back to their original size. This simplifies the image, giving it a 'brush' look, which is useful for contour detection. (Fig. 5)

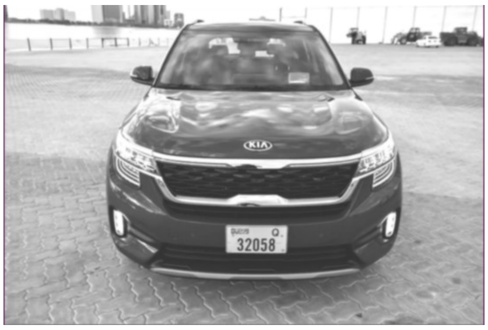

Fig. 3. Image after Grayscale, Brightness and Blur modifications.

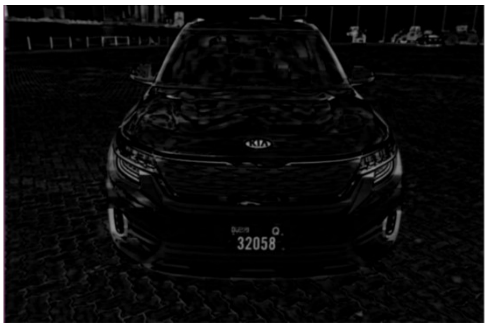

Fig. 4. Image after Blackhat operation.

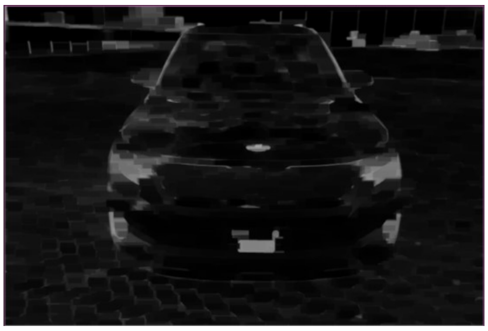

Fig. 5. Image after Close Morphological operation.

C. *Binarization and OTSU's algorithm*

Now we need to remove unimportant details such as reflections on the car and whatnot, and to do this we will binarize the image using a threshold. This will make areas of a certain brightness that is under the threshold black, and otherwise (which is above the threshold) white. This makes the image completely black and white. (Fig. 6)

The issue with Binarization is that each image has a different threshold that works for it, so we will use an algorithm known as the OTSU algorithm [12]. This algorithm is named after Nobuyuki Otsu, who created the OTSU method used in this algorithm. It will calculate the threshold for a given image and apply that threshold, and we should have an appropriate black and white image that retains the white rectangles and shapes where the characters of the license plate are located.

D. *Erode and Dilate Operations*

Once the image is black and white completely, we need to remove any small dots and lines on the image to further simplify it. We will apply the Erode operation, which will shrink all white regions. This will cause the small details to disappear, while major ones will simply reduce in size. (Fig. 7)

We will then try to return the major details to their original size using the Dilate operation. This is the opposite of Erode, in which it will expand the size of all white objects in the image. These operations further remove the noise in the background to simplify detection of the license plate's location. (Fig. 8)

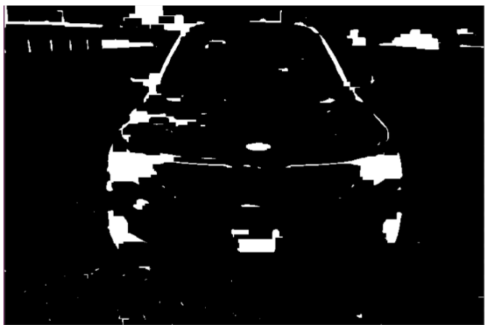
Fig. 6. Image after Binarization.

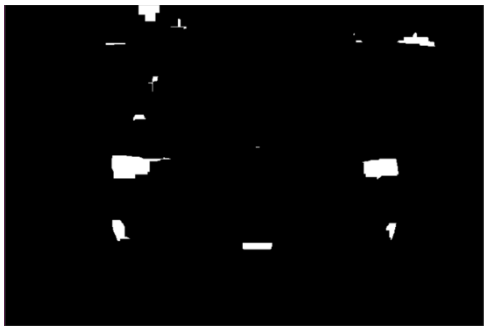
Fig. 7. Image after Erode.

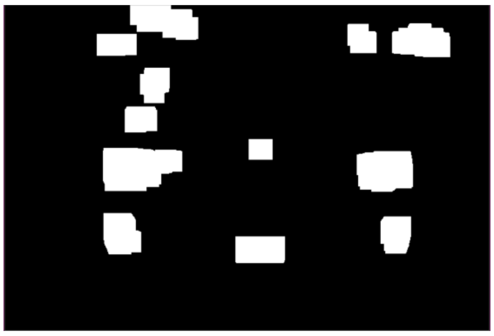
Fig. 8. Image after Dilate.

E. Contour Extraction

The image is now as simple as we can get it, and we can start with contour detection using OpenCV. We start by sorting the contours by area and keeping only the top ten to further remove noise. For each contour, there will be a bounding box for it, with their position and area using x and y coordinates, with the origin (0, 0) in the top-left corner of the image. (Fig. 9)

For each bounding box, we find its center. If the center for a bounding box is 30%-70% from the origin in the x-axis, and 50%-90% from the origin in the y-axis, then we allow that bounding box to be a candidate for character recognition. Otherwise, it is rejected.

Using this method, We can narrow it down to two or one bounding box, which we select as candidates for detection and highly expect for at least one to be the bounding box of the license plate. (Fig. 10)

Once the license plate has been detected, we need to extract the characters from it using the OCR. We run a Canny Edge operation [13] on the Blackhat image that we previously acquired (Fig. 11), then extract the contours from the Canny Edge image. This will highlight the characters correctly. (Fig. 12)

We will sort the contours by area, and remove the smallest 40% of them, which are usually noise. Then, we remove any contour that do not intersect with the license plate box candidates. For the remaining contours (which are usually within or intersecting with the candidate boxes), we find their bounding box and calculate the aspect ratio. The aspect ratio (width/height) needs to fall within 0.15-0.90, otherwise it is rejected. This is because the font used in UAE license plates are all the same, and the ratio of the characters' fall within the range specified.

We also remove any shapes within characters such as the circle in zero using an intersection percentage, where over 40% is removed.

For each remaining contour, we extract them one by one from the Blackhat image, then invert the colors to finally run them into the OCR engine. This is because Tesseract works best with black characters on white background.

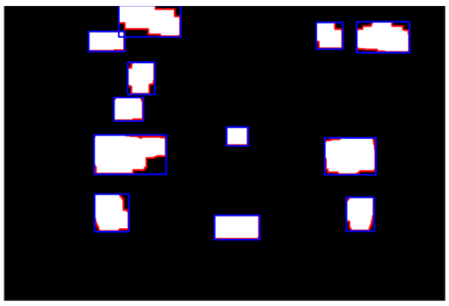
Fig. 9. Image after highlighting the bounding boxes.

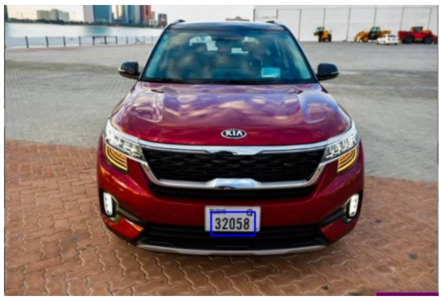

Fig. 10. Final bounding box selected.

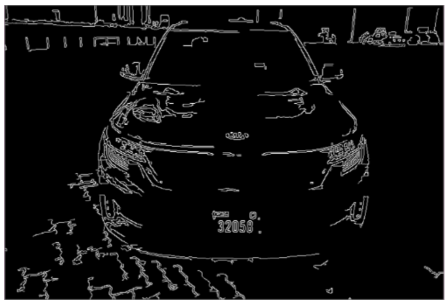
Fig. 11. Image after Canny Edge operation, which applies after Blackhat operation.

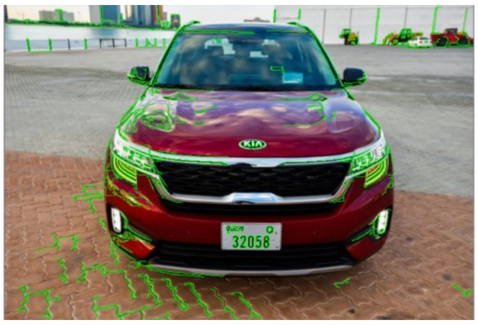

Fig. 12. Canny Edge operation and contours extracted.

IV. Alternative Solutions

In this section, we will demonstrate alternative solutions from both software and hardware perspectives. We have had many different options to work with, so we have selected the best 3 of each specific part of the device.

A. *Optical Character Recognition models*

OCR models come in open source, as well as retail. We are not going to focus on the models that cost us money, as we work on a limited budget. There are several Open-source models we can compare, and we picked the best three:- CuneiForm, GOCR and Tesseract.

CuneiForm is a free, open-source AI engine developed by Cognitive Technologies OpenOCR. It covers 23 languages and carries out text format scanning and identification. The detection accuracy is decent with this model as it uses a dictionary check to improve the recognition, but it does not support Arabic language out of the 23 supported languages. It also has little to no support for Linux, which is used with the Raspberry Pi embedded device. Due to that, we are forced to avoid this model.

GOCR is another free open-source AI engine developed by Jörg Schulenburg and is under the GNU Public License. GOCR covers a multitude of single-column sans-serif fonts, but it has no multi-language support. As such, it will also be avoided. Note that GOCR supports Linux.

Lastly, Tesseract is one of the most popular AI engines out there. It is a free, open-source engine like the previous two engines. This engine supports 116 languages as of version 4.1.1, and two of them are right-to-left languages which are Hebrew and Arabic. This model is regarded as one of the most accurate and has been in development for quite some time by Hewlett-Packard. It meets the criteria of supporting English and Arabic, as well as being open-source and fairly accurate. We believe this is the best choice for an OCR engine [11].

B. *Ultrasonic sensors*

For sensors, we consider the following; the HC-SR04, the US-100, and the GY-US42.

all sensors meet the range requirement as being above 3 meters, along with the power being efficient across the board. Where they differ is in their price and availability. The HC-SR04 can be found in many places in the UAE. Examples of where we have found it are Besomi Electronics and Popular Electronics in Naif city-center. It is also the cheapest of the three at 12 AED. The US-100 does not seem to be available at the mentioned stores, but is sold online at Edwin Robotics, which makes its availability slightly less. The rarest of them is the GY-US42, where it is hardly supplied in the UAE. We could only find foreign suppliers. As such, we believe the HC-SR04 is the best choice.

C. *GSM/GPRS communication module feature*

A GSM/GPRS communication module as a hardware component was considered. This module is capable of communicating with any device that has a SIM card installed, much like a modern smart phone. The objective of this module was to inform the driver's smart phone which is linked to the license plate number of any activity of the car with the device, such as detection and analysis.

Due to budget and time, it was difficult to implement this feature, as it required extra programming, and it also needed a sufficient SIM card running on it. Thus, this component was removed in the final version.

D. *Cameras*

Cameras are narrowed down to the Sony Spresense, The Raspberry Pi Camera V2, and the Raspberry Pi Night Vision Camera.

The video and image resolutions are all the same, but the V2 camera turns out to have a better overall pixel size at 8 MegaPixel. The two Raspberry Pi-compatible cameras are priced averagely the same at 110 AED, with the Spresense being more expensive at 130 AED. It is important to note that the Spresense does not guarantee compatibility with the Raspberry Pi, despite using the same connector. To add, the night vision camera, as indicated from its name, does have night vision capability in the form of two IR lights that illuminate the area from the camera's perspective. The V2 is clearly the best choice as it has the best pixel size and is the cheapest of them all.

V. Prototype

The first prototype had the Raspberry Pi 4 in a standard case with ventilation holes and a large fan that cools the device. The Ultrasonic sensor and the GSM/GPRS module are connected using the jumper wires to the USB converters, then these converters are connected to the Pi through its USB slots. The camera does not require such connection as it has its own port on the Raspberry Pi, so we connected it directly using its respective port. The GSM module was a consideration at this stage, hence why it is mentioned in this prototype version. The entire device is powered using an official Raspberry Pi 4 USB-C power supply.

The final version saw the removal of the GSM, a new custom case, and the addition of a side sensor. (Fig. 13)

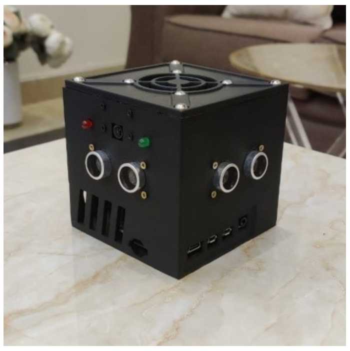

Fig. 13. Final version of the device.

VI. Results

We tested the sensors by making sure the implemented code is compiled and running correctly with the main code, then obstructing it with an object less than 2m away. The sensors are configured to calculate the amount of time it takes to receive the ultrasonic waves from the time it transmits them. Then, a formula is used to convert the time to a range in meters. We have configured our sensors to send a signal when they detect an object within a range of 2m or less. The signal is picked up by the client which it sends to the camera to begin taking the image.

The camera is simply configured so that it waits for a signal from the sensor once it detects an object within 2m. If it does, then the camera simply operates by taking the image, and encoding it to base64 for processing. The image resolution reaches 4K (3280×2464), which is useful for character extraction. We tested this device by first interacting with the sensor, then letting the camera output an image with a simple test code to make sure it is taking images correctly.

The main client simply operates so that it interacts with the three mentioned devices that have their own respective code that are imported to the client code. It awaits a signal from the front sensor, and once received, it sends it to the camera for the image capture and encoding. The client also interacts with the server, so that it sends the encoded image to it with a secret authentication key for security. Then it awaits a response from the server regarding the authorization.

We ran three tests under which we have one successful authorization with entrance acceptance, one successful authorization with entrance denial, and one unsuccessful authorization due to incorrect secret key.

The LED lights were also tested along these tests to see if they blink the correct pattern according to the state the device is in.

The limitations of testing were that testing was only possible in live, practical environments to maximize effectiveness of the device. It was also necessary that testing is done in lit conditions. Internet connection was required for the device to connect to the cloud, and it needed to be configured as well. If the device is to be exposed to the environment, the weather conditions need to be clear as to not damage the device. Any other obstacles also need to be clear of the front ultrasonic sensor to allow it to function correctly. Lastly, the device needs to be placed 1 or 2 meters above ground and placed on the left side of the gate at an angle of 15 to 20 degrees counter-clockwise.

VII. Future Work

For future work, we believe the results of this device will inspire to create more intelligent technological solutions to certain problems. With more focus on computer systems and applications in the current age, this device is a step to that direction. We also believe this design can even be improved on with the appropriate resources, such as facial recognition or motion sensors that detect gestures. Our device is also flexible in that the majority of the software is running in the cloud, thus updating the software would be very easy and immediately integrated into all connected devices.

VIII. Conclusion

With security requiring shifts and guards to work them, it becomes exhausting and difficult to perform the job especially in midnight shifts. We felt that by creating this device, we can simplify the job of any vehicle gate guard and putting the capabilities of computers to its test. With the device finalized, it will be capable of detecting the incoming vehicle with the sensor, survey and take the image of the vehicle front with its 4K camera, process and analyze the characters on the license plate using our sophisticated image processing steps and the Tesseract OCR engine, then checking for authorization by accessing our MongoDB database that is hosted in the cloud along with the server. From here, we could allow it to send a signal to a basic analogue gate to open and close it based on the authorization of the registered plate.

Acknowledgment

Thanks to Precise 3D hub for lending case design advice and 3D printing services to help finalize the case of the device.

We also extend thanks to Dr. Talal Bonny for helping with advice and direction of this project and research.

Lastly, we thank the University of Sharjah for helping with the project's budget and for the opportunity to develop this device.

References

[1] Nassan, W.A. et. al., A New Chaos-Based Cryptoystem for Voice Encryption, 2020 3rd International Conference on Signal Processing and Information Security, ICSPIS 2020, 9340132.

[2] Almas Industries Team. (2018). Does CCTV Prevent Crime? [Blog post]. https://almas-industries.com/blog/cctv-crime deterrent/#:~:text=Studies%20into%20effectiveness&text=The%20systematic%20review%20of%2041,and%20other%20anti-social%20behaviours.

[3] T. Bonny, T. Rabie et. al. "SHORT: Segmented Histogram Technique for Robust Real-Time Object Recognition", Journal of Multimedia Tools and Applications, 2019.

[4] Dofflemyer, B. (2017). Do Surveillance Cameras Actually Deter Criminals? https://www.arcdyn.com/articles/do-surveillance-cameras-actually-deter-criminals/

[5] T. Bonny, Tamer Rabie, et. al., "Multiple histogram-based face recognition with high speed FPGA implementation", Journal of Multimedia Tools and Applications, 2018.

[6] F. Almutairi, T. Bonny, New Image Encryption Algorithm Based on Switching-type Chaotic Oscillator, IEEE International Conference on Electrical and Computing Technologies and Applications, 2019

[7] Almutairi, F. et. al., Image Encryption Based on Chua Chaotic Oscillator, 2020 3rd International Conference on Signal Processing and Information Security, ICSPIS 2020, 9340157.

[8] Kuhns, J. (2012). Understanding Decisions to Burglarize from the Offender's Perspective. Charlotte, North Carolina, USA: University of North Carolina.

[9] M. Affan Zidan et. al. , High Performance Technique for Database Applications Using a Hybrid GPU/CPU Platform. Great Leak Simposium on VLSI, 2011.

[10] Bonny T. et. al. An Adaptive Hybrid Multiprocessor Technique for Bioinformatics Sequence Alignment.International Conference on Biomedical Engineering Conference, 2010.

[11] Taylor, K. (n.d.). List of Top 5 Open Source OCR Tools. Retrieved from https://www.hitechnectar.com/blogs/open-source-ocr-tools/

[12] Nobuyuki, O. (1979). A threshold selection method from grey-level histograms. IEEE Transactions on Systems, Man and Cybernetics.

[13] John, C. 1986. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence.