# Eye-Gaze Control of Mobile Vehicle for Paralyzed Patients

Eliseo Nunez, *Systems Engineer*, Minchul Kim, *Signals Engineer*, Nazaret V. Montano, *RF Engineer*, Emanuel David, *Software Engineer*, and Dr. Pramod Khargonekar, *Distinguished Professor/Advisor*

***Abstract*— In this paper, we propose an effective and affordable system that allows fully paralyzed patients to securely pilot a radio-frequency controlled car with nothing but their eye-gaze. The aim is to create a system accurate enough to help eliminate the dependence paralyzed patients have on others for their personal mobility. Our approach is rooted on the OpenCV framework for fast and reliable eye-gaze detection algorithms, an economical 2.4Ghz Antenna for unhindered communications, and a lightweight ATmega328p for motor controls. To date, our team has managed to build a prototype able to track eye-gaze and pilot a car with approximately 75% accuracy.**

***Index Terms*— OpenCV, eye-gaze, paralysis, video-processing**

## I. Introduction

IT is estimated that every year, there are approximately 17,700 new incidences of spinal cord injuries in the United States alone, 60% of which result in either complete or incomplete tetraplegia [4]. The total number of affected in the United States looms around 400,000 from year to year. While worldwide, there is speculated to be several million—though a firm estimate has not yet been produced.

Many of the injured will fortunately experience a recovery and regain some sense of mobility. But there remains a great deal of victims that are not as fortunate and are forced to depend on others for their mobility.

Our system aims to solve this issue at a universal scale, open to everyone who needs it, without imposing any more financial constraints on the victims than their medical costs already do. This is why our hardware and software components are affordable, accessible, and reliable.

We currently have a working prototype that can track a user's eye-gaze and generate controls for a remote-control car with accuracies up to 75%. This is a big step up from our previous low 50% accuracies in the beginning of the quarter. Aside from improving both precision and accuracy, we have also iterated through various control schemes, and have arrived at one that is comfortable, intuitive, and accessible for even the most severely incapacitated. All our user's need to control the car is a pair of eyes, the ability to blink, the ability to stare, and our simple, affordable equipment. Here is how it all works.

## II. Hardware

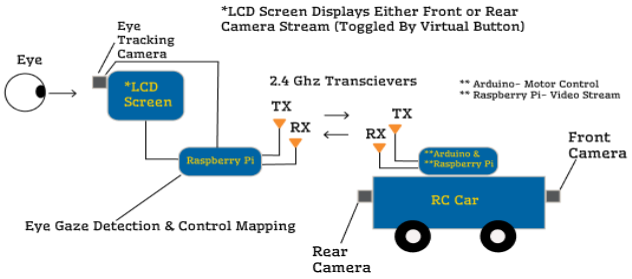



Fig. 1. General block diagram of the system structure.

### A. General Overview

The general block diagram displayed in figure one showcases how we imagine our system to look. At the heart of it lies a Raspberry Pi 3, wired to a small camera and a liquid crystal display (LCD). This camera and LCD combo will serve as our user interface, and the only point from which the user will control the car.

On the LCD we will display a live camera feed from one of two first-person-view (FPV) cameras mounted to the anterior and posterior of the remote-controlled car. This means that our user's will have an immediate, first-person view of the remote-controlled car's surroundings. To allow for both forward and reverse control, the users will have the ability to toggle between the front and rear-view cameras via a virtual button on the LCD. When they stare at a designated spot on the LCD for more than an allocated amount of time, the camera feed will switch.

The camera mounted on top of the LCD will be responsible for capturing the user's eye-gaze and feeding it into the raspberry pi for image-processing and eye-gaze detection.

Once our algorithms have worked their magic, the raspberry pi will send an encoded instruction to our 2.4Ghz NRF24L01 antenna module, marked as TX and RX in figure 1.

Through a secure channel, this radio-frequency module transmits the instruction to an identical receiving antenna hooked up to the ATmega328p (Arduino) directly on top of the RC car. The encoded signal is then translated to an instruction for our 6V DC motors, which will be driven by the affordable L298N motor driver module.

### B. Power Details

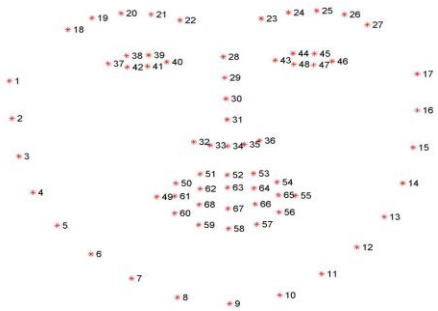
At this point, we have been powering our prototype system using four AA and two 9V alkaline batteries. We realize that this is a strange combination of voltages and the non-ideal chemistry to drive our motors. But the system does not drain as fast as we thought it would. In fact, we have only changed our batteries once in the past 8 weeks of on-off but consistent use. We estimate that in all they lasted a total of 20 hours of continuous use. Nevertheless, we plan on migrating our power electronics to the lithium-ion domain, in order to elongate our systems life and improve DC motor responses.

## III. Software

### A. Eye-Gaze and Blink Detection

To effectively track our user's eye-gaze we are using the OpenCV framework running on Python 2.7. The facial feature recognition is made possible by an open source trained dlib model that can detect 68 points on a user's face with outstanding accuracy. From these 68 points we are extracting only 8 and have engineered a way to exploit this minimal set to recognize blinks and gaze-direction.

Fig. 2. The 68 points the open source dlib model can detect



In order to detect blinks, we created a function to constantly monitor the distance between the midpoints of 38-39 and 42-41 on the left and 44-45 and 48-47 on the right. Whenever the distance between these points drops below a certain dynamic threshold, we register that as a blink. However, in order to make plausible use of this in our motor control, we had to make our system respond not to every blink but rather an extended blink time. We therefore tuned our system to recognize an extended blink lasting for more than .5s. This way we have an easy way to start and stop the motors.

In order to enable the users to steer the car, we also had to figure out a reliable way to detect a user's eye-gaze. The method we use to recognize this simple. First, we make sure that our image is converted to grayscale. This way we have an easier time processing the pixels. We then isolate the two eyes, by drawing a box around the outermost edges of their corresponding landmarks. These steps are very common across many eye-gaze detection systems [3].

After this is done, we apply our own little magic, in the form of an adaptive filter that converts the gray scale, isolated eye image into a binary, black and white image. It is important that the filter for this function be adaptive as changes in environmental light can cause undesirable thresholding with a constant filter, resulting in lack of pupil resolution. The goal of this step is to make the pupil as distinguishable from the sclera, the white part of the eye, as possible.

The following step is where the most exciting of the magic happens. We split the eye into two halves, a right half, and a left half. And we create a function to constantly monitor the ratio of black to white pixels in each halve of the eye. The side that is blackest, after taking an average of 5 samples, indicates the user's gaze-direction.

The final step is perhaps the most important step; and that is to normalize these ratios to the total pixel area of the eye. This makes the algorithm immune to changes in the user's distance from the camera. This is an issue that afflicts many other eye-gaze control systems and calls for the addition of other camera set ups to compensate for the error [1]. But our system resolves this rather efficiently without the addition of any more cameras via this normalization.

At this moment in time, we have not yet migrated our eye-gaze and blink detection algorithm to the Raspberry Pi from our original design. All our trials have been conducted on an HP laptop with the built in web-cam, however, we have indubitable confidence that these algorithms will also work on the Raspberry Pi, after some optimization for the platform.

### B. Control Mapping

After weighing on different designs, we decided that the simplest and most effective way to control a car with just a pair of eyes, and nothing more, is using the following scheme.

To start the car, the user will blink, or shut their eyes, for a total length of time greater than .5s. Once the car has started to move, the user can stare at the middle, the left, or the right of the screen for more than .2s to steer the car in that direction. To stop the car, the user can blink, or shut their eyes for more than .4s, and the motors will immediately stop.

We are still on the look for more efficient schemes, but this one seems to work the best so far.

### C. Inter-Device Communications

Our python script is currently communicating with the Arduino's Universal Asynchronous Receiver/Transmitter (UART) communication ports via the PySerial library. Note that the Arduino in this case is substituting the Raspberry Pi in our block diagram of Fig.1. Additionally, the Arduinos are communicating with the NRF24L01 module via the Serial Port Interface (SPI) protocol.

## IV. Conclusion & Remarks

There is still plenty of room for improvement in our system. However, at the current protype stage, it is achieving accuracies that are relatively stable for control. One of the main issues that we are encountering at this moment is one that we unfortunately did not foresee. We have noticed that,

during active use, our motors do not spin at the same identical speed, and hence the car to steers in an undesirable direction. We are diagnosing the issue but believe that the culprit may be our L298N motor driver. If this is so, we will be faced with making decision of either changing our motor driver module or adding feedback control in order to remedy the discrepancies in motor speed.

After this issue, is fixed, we will simultaneously work on brining the accuracy our control towards 100% and tackling the infamous recalibration issue that many eye-gaze control systems have [2]. This being that there is currently no one-sized system model that fits all potential users, since there is significant eye-variation across every individual. At this moment our designs have been fitted to one of our team-members eyes, but we can figure out a dynamic approach to select our threshold values to match every unique eye geometry.

## APPENDIX

*1) What technical standards were relevant to your projects, how did you pick between them, and was your resulting design compliant with these standards? Some simple ones include Bluetooth version, WiFi version, USB version, SD card type, etc., but many more specialized standards exist too. Non-compliance can easily occur if, for example, FAA, FCC, etc., regulations are ignored, off-spec or counterfeit parts are used, and so on. Please review the standards document available on the class web site and as discussed in class.*

In order to comply with the FCC standards and rules, we had to choose an antenna which would operate quick enough to satisfy our hefty data-rate transmission needs, while simultaneously making sure that it did not interfere with any other high-frequency signals or sensitive equipment in our surroundings. This is we chose our antenna to operate in the Industrial, Scientific, and Medical (ISM) Band of the spectrum. We are able to satisfy our fast transmission rate design requirements, with a very narrow private channel for our devices, while ensuring that we do not violate any of the FCC standards or rules.

*2) What constraints have you faced in designing and building your projects and how did you cope with them? Examples of possible constraints include accessibility issues, safety code issues, constructability, cost (always a big one), power constraints, ergonomic difficulties, constraints that affect the ability to extend the functionality and interoperability of the project, legal considerations, maintainability issues, manufacturability, marketability, policy and regulatory issues, scheduling issues, sustainability issues, usability issues, etc. What constraints were important in your project and how did you work around them or solves them?*

Our entire project revolves around trying to resolve a host of accessibility issues while meeting very strict constraints. We are designing a system that is to be operated with nothing else but a paralyzed user's eyes. And one of the biggest challenges we are faced with is to figure out how to encode as much useful information in simple-eye movements, without inconveniencing, tiring, or confusing our users.

The eyes of all individuals are constantly moving, jittering, between fixations, in motions we call saccades. And it is important that when controlling a mobile, we ignore these sudden movements to concentrate on the real intentions of our users' eye-gaze, lest we cause an accident. We have worked, not around, but through this problem by iterating through as many control schemes designs as we can conceive. And even though we have currently arrived at a potentially good design, we are still actively looking for any more improvements we can make.

Another constraint we are faced with, though not as serious as the first is that of power consumption. We must figure out an effective way to power our devices without incurring too much weight to our design. To solve this, we will look at the average power consumption, amperage, and voltage draw of our system and find a corresponding power source for our needs.

*3) In our current world of unrelenting hacking and hidden vulnerabilities, what hardware and software security issues risked being present in your work and how did you mitigate them? What hardware insecurities did you face? For example, the "spectre" and "meltdown" problems are hardware insecurities that have plagued Intel over the last several years. Many, many software insecurities also exist, seemingly turning up at an exponential rate. What did you to identify security issues, which were found to be a threat, and what did you do to help prevent exposure to these vulnerabilities, etc?*

Because our system deals with the remote control of two potentially dangerous motors, it is important that we protect against any unwanted interference, jamming, or undesired manipulation of our transmission signals. These are potential threats that, when brought up to a larger prototype scale, can endanger not only our user but any nearby individual as well.

We can effectively deal with the jamming, by simply killing the DC motors whenever our connection is broken and or distorted. And we already deal with interference through the handshake protocol, which only drives the motors after the sending instruction has been acknowledged and confirmed by the both receiver and transmitter. This adds latency to our system but in the end, it is always best to have that as a security measure.

We are not entirely sure how to deal with the case of a malignant transmitter hijacking our channel, but we are currently investigating this vulnerability and its potential solutions.

## ACKNOWLEDGMENT

We would like to thank all the team members that made this project possible through financial and intellectual backing. And we would also like to thank Dr. Pramod Khargonekar for his counsel in times of need!

## REFERENCES AND FOOTNOTES

### REFERENCES

[1] A. Kar and P. Corcoran, "A Review and Analysis of Eye-Gaze Estimation Systems, Algorithms and Performance Evaluation Methods in Consumer Platforms," in IEEE Access, vol. 5, pp. 16495-16519, 2017.

[2] Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish S. Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. “Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design. In Proc. of CHI

[3] H. Nakayama, N. Yabuki, H. Inoue, Y. Sumi and T. Tsukutani, "A control system for electrical appliances using eye-gaze input," 2012 International Symposium on Intelligent Signal Processing and Communications Systems, Taipei, 2012, pp. 410-413.

[4] ”National Spinal Cord Injury Statistical Center, Facts and Figures at a Glance”. Birmingham, AL: University of Alabama at Birmingham, 2019.