A REPORT ON

Quadcopter Control by fusion of EEG data and Eye-tracking using Extended Kalman filters

PREPARED BY

Kshitij Chhabra	2017A8PS0691G
Pranay Mathur	2017A8PS0487G
Irish Hareesh Mehta	2017AAPS0295G

IN PARTIAL FULFILMENTS OF COURSE REQUIREMENTS FOR

INSTR F266/ECE F366

May, 2020

Quadcopter Control by fusion of EEG and Eye-tracking data using Extended Kalman filters

I. Introduction

Numerous people around the globe every year are affected by spinal cord or stroke injuries and because of which it becomes difficult for them to even perform their daily chores. Spinal cord is a medium which relays signals and information between the brain and specific organ of the body, the absence or defect of which causes people motions to be constraint even when the signals from brain are generating just fine.

Brain-Computer Interface (BCI) is one such technology that gives the user a solution to interact with computer peripherals just by the power of their thought by classifying them to specific actions which the user intends to do. Although this technology has come a long way since the last few decades, certain problems such as inadvert eye blinks or muscle artifacts affect the efficiency of the system. Even though methods like averaging filters or method of least square estimation can be used to improve the working of the system, it has been difficult to develop a system with high efficiency.

Eye tracking on the other hand is another upcoming technology where we predict the location where the user looks. In our project we have used the concept of BCIs further enhanced with the capabilities of eye tracking. The reason behind using eye tracking was that a user moves his eyes unconsciously while thinking about the motion of the drone.

The main aim of our project is to develop a technology to help those affected by such neurodegenerative diseases. The controlling of drone is just a proof of concept of the capabilities of this technology as a quadcopter suits to be an ideal test bed for research purposes due to its six degree of freedom and its agile nature. With help of using more sensors, which could give us EMG values, project could even be further extended to control an entire exoskeleton which would be an ideal aid and help us build a more independent society. Although, the execution of which still lies in the domains of science-fiction.

II. Eye Tracking

It refers to the process of measuring where humans look, also known as their point of gaze. These measurements are carried out by an eye tracker that records the position of the eyes and the movements they make. The working principle is based on reflections of infrared rays. Near-infrared light is directed towards the center of the eyes (pupil), causing detectable reflections in both the pupil and the cornea (the outer-most optical element of the eye). These reflections – the vector between the cornea and the pupil – are tracked by an infrared camera. This optical tracking of corneal reflections is known as pupil center corneal reflection (PCCR).

An infrared light source (and thus detection method) is necessary as the accuracy of gaze direction measurement is dependent on a clear demarcation (and detection) of the pupil as well as the detection of corneal reflection. Normal light sources (with ordinary cameras) are not able to provide as much contrast, meaning that an appropriate amount of accuracy is much harder to achieve without infrared light. Light from the visible spectrum is likely to generate uncontrolled specular reflection. In contrast, infrared light allows for a precise differentiation between the pupil and the iris – while the light directly enters the pupil, it just "bounces off" the iris. Additionally, as infrared light is not visible to humans, it does not cause any distraction while the eyes are being tracked.

Eye tracking using consumer webcams and offline software has been studied before and unsurprisingly has been found to be less accurate than specialized equipment, negating its utility in professional studies. WebGazer aims to overcome the accuracy problems that webcams typically face, by adopting user interactions to continuously self-calibrate during regular web browsing. Webcam images during these user interactions can be collected and used as cues for what the user's eyes look like. Future observations of the eyes can be matched to similar past instances as WebGazer collects mappings of eye features to on-screen gaze locations, allowing inference of the eye-gaze locations even when not interacting.

This project uses WebGazer, a new approach to browser-based eye tracking for common webcams. WebGazer.js, a javascript library aims to overcome the accuracy problems that webcams typically face, by adopting user interactions to continuously self-calibrate during regular web browsing. Webcam images during these user interactions can be collected and used as cues for what the user's eyes look like

A. Impact of ambient lighting on eye tracking

Static eye tracking devices generally face a problem with the lighting conditions as angled light reflects from the lens of the eye and mixing of infrared rays from the sun leading to inaccuracies in the tracking. Generally researchers avoid brightly lit rooms (no overhead light) and ideally, use ambient light. It's particularly important to keep the lighting levels consistent when measuring levels of pupil dilation. When considering computer web-cameras for tracking the eye movement, the concept of Computer Vision (CV) comes into play. There are two gaze estimation methods in WebGazer, one which detects the pupil and uses its location to linearly

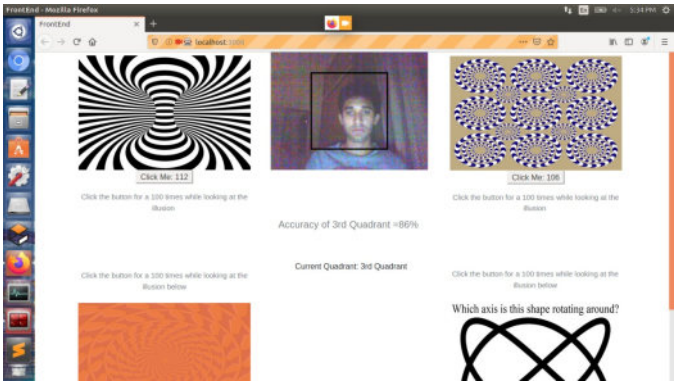

Fig. 1. A slight delay in recognizing the face in dim conditions

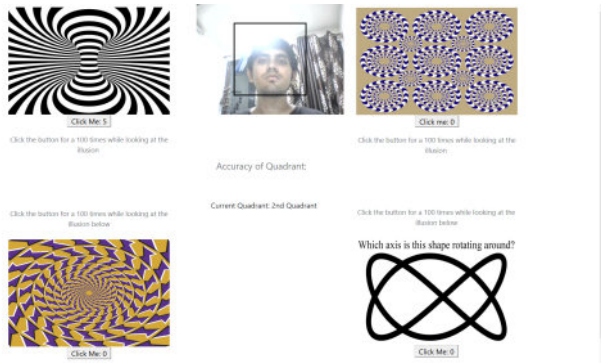

Fig. 2. The algorithm faces problem when subject is in bright lighting

estimate a gaze coordinate on the screen, and a second which treats the eye as a multi-dimensional feature vector and uses regularized linear regression combined with user interactions [19]. Our implementation using WebGazer.js utilizes clmtrackr [22], a face and eye detection algorithm. Clmtrackr performs a realistic fitting of the facial and eye contour. [19] To provide consistent input for WebGazer, we use the smallest rectangle that fits the eye contour. The algorithm is able to give accuracies of more than 90% even when lighting conditions are dim as shown in Fig. 1 as well as ambient as shown in Fig. 2. The only condition where accuracies are negatively affected is when a light source is kept right behind the user as depicted in Fig.2.

B. Eye Head Movement Correlation

To accurately match the pupil locations and eye features to screen coordinates, the software/device finds a mapping between the 2D and 120D (2 eye images of 6 X 10 pixels) [21] vectors respectively and the gaze coordinates on the device. This relationship is complex—it depends on the 3D position and rotation of the head with respect to the camera and screen, requiring careful calibration and expensive computation. WebGazer instead relies on continual self-calibration through user interactions that normally takes place in web navigation (clicks, mouse movements, scrolls).

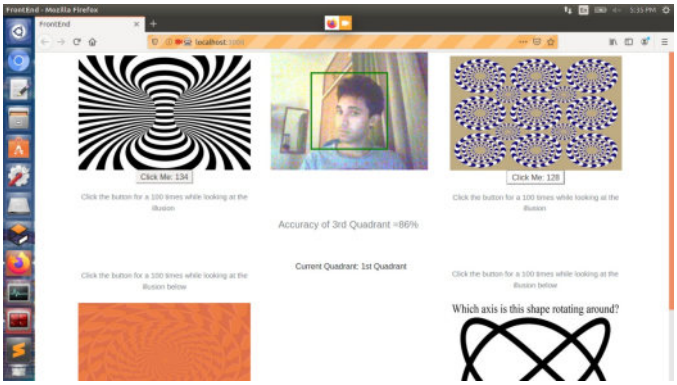

Fig. 3. Clear markings of nose, face outline and the eyes. The algorithm assumes the pupil to be the darkest in its surroundings

C. User eyeglasses with high index or tinted

Conventional gaze tracking systems are limited in cases where the user is wearing glasses because the glasses usually produce noise due to reflections caused by the gaze tracker's lights. This makes it difficult to locate the pupil and the specular reflections (SRs) from the cornea of the user's eye. These difficulties increase the likelihood of gaze detection errors because the gaze position is estimated based on the location of the pupil center and the positions of the corneal SRs [20]

D. Physical limitation in eye control due to squint

Strabismus, termed squint and heterotropia as well, is that the two eyes do not point to the same direction. General eye tracking for people with squint eyes is bound to give errors. For some strabismic people, their eyes can fixate precisely at some directions, whereas for some other directions the two eyes do not align well. This case happens a lot in particular to incomitant strabismus, the strabismic eye of which cannot move in some specific directions. [23]

III. Electroencephalography

Human brain consists of millions of neurons which are playing an important role for controlling behavior of human body with respect to internal/external motor/sensory stimuli. These neurons act as information carriers between human body and brain. These neurons transmit information through the polarization and the depolarization. These electric activities add up to form action potentials which can be recorded from the scalp by placing electrodes. These EEG waves can be classified according to their frequency into-:delta, theta, alpha, beta and gama waves. Our main focus in this project was two-fold, coming up with a metric for quantifying the accuracy and then improving accuracy.

1) EEG Headset: To build this novel application of BCI, the headset that we decided to use is Emotiv EPOC+, which is a 14 channel headset with two reference electrodes. The data internally sampled at 2048 samples per second (SPS), which is downsampled to 128/256 SPS depending on the user's choice. The electrodes that were used for acquiring data were

positioned at AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, and AF4 positions according to the 10-20 international convention of placing of electrodes [11]. The advantage of having these positions is that it covers most of the frontal and occipital regions of the brain, which are the prime spots of harvesting Beta waves of the brain. This high frequency and low amplitude brain waves lie mostly in the range of 13-30 Hz and are mostly responsible for carrying information about the motor actions from the human brain. The headset is equipped with a built-in digital fifth-order sinc filter and also offers usability of an internal IMU, Magnetometer, and an accelerometer.

A. Data extraction and Classification

Once the data was collected by the headset it was transferred to the PC via Bluetooth Low Energy. We used Emotiv BCI application to classify the thoughts of the user which does so by using an SVM. The Emotiv app clasified the actions into five states depending on users intend: neutral, forward, backward, left, and right. Data sets were trained for specific user profiles. Fig. 4 shows the trained data example. Further we used Node-RED toolbox to extract data out of the Emotiv BCI app to actual hardware and was further transferred to a Linux PC for the actual drone control. Fig 5 shows the Node-RED system that was used for data extraction.

IV. Extended Kalman Filter for Sensor Fusion

In theory, Kalman Filters are also known as Linear Quadratic Estimators, which deploys the use of an algorithm that uses a series of measurements, containing statistical noised to estimate the further states of a system more accurately than normal measurements. The key concepts in using Kalman Filters are that it provides different weightage, what is called Kalman Gain to statistical estimation and measurements which are otherwise obtained from onboard sensors, by estimating a joint probability distribution over the variables for each timeframe. This algorithm is a two-step recursive process and involves a prediction phase and an update step. In the prediction phase Kalman filter products the estimate of current states of the system along with the errors it may be having. Once this is done, during the update phase measurements are taken which are necessarily corrupted with uncertainties, a weighted average is then taken and more weight is given to estimates with higher uncertainty.

Kalman Filters has two limitations.

- They assume the noise to be Gaussian
- They work only on Linear Systems

As a solution to the second problem, the concept of Extended Kalman Filters was introduced which doesn't compulsory expect the system to be linear but demands that the functions to be differentiable.

$$x_k = f(x_{k-1}, u_k) + w_k \tag{1}$$

$$z_k = h(x_k) + v_k \tag{2}$$

Here w_k and v_k are the processes and observation noises which are both assumed to be zero-mean multivariate Gaussian noises with covariance Q_k and R_k respectively. u_k is the control vector.

Function f and h are used to calculate predicted state and measurement respectively. The major application of EKF comes in use when we have to use data from two or more sensors at the same time which have a similar application. For eg. Using GPS and LIDAR data at the same time for navigation.

Our project involves the use of Eye-tracking and BCI and the key challenge here is assigning individual weights to both of them which will be further worked on using EKF which in turn would rely on different accuracies of EEG Measurement and Eye tracker.

Comparison There exist muliple algorithms for sensor fusion apart from the Extended Kalman Filter like the Bayes filter, Dempster-Shafer method, Central limit theorem, Unscented Kalman Filter and convolutional neural network. We chose EKF for fusion due to a tradeoff between computational complexity, minimising response time and hardware constraints. Running methods like convolutional neural networks to produce results in real time with the current hardware was not possible. Methods like the central limit theorem are easy to implement and would have performed well with low response times and computational complexity. They however

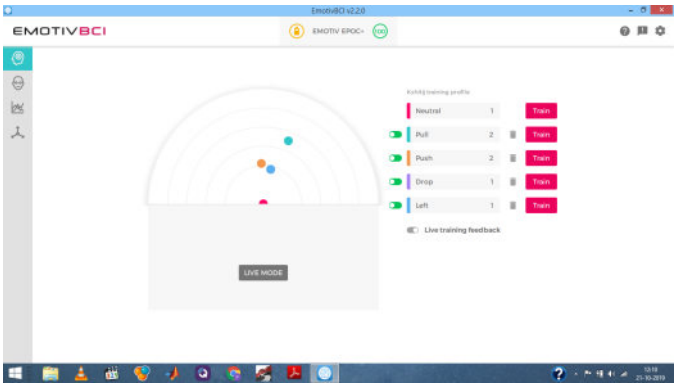

Fig. 4. Classification of Actions

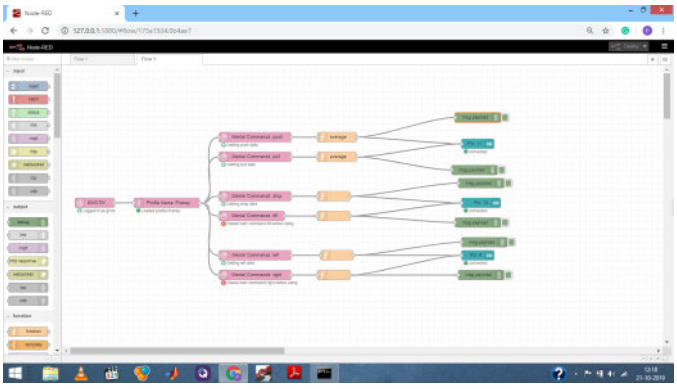

Fig. 5. Node-RED the system used for data extraction

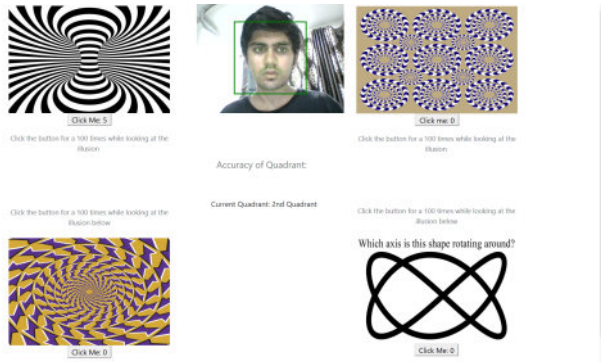

Fig. 6. The calibration window on HTML

would not have taken into account the large uncertainties that come up in our use case and result in lower accuracy. The closest replacement is the unscented Kalman filter that works with non-differentiable functions as well but was more computationally expensive for obtaining accurate results.

A. Fusion methodology

The major challenge in using webcam based tracking is the low quality of webcam and setting up an interface to calibrate the algorithm. Webgazer uses JavaScript to generate the Computer Vision algorithm "clmtrackr" which detects the face and the eyes within a bounding box. We used HTML and CSS to create an interactive calibration page as displayed in Fig. 4, which runs WebGazer.js in hindsight. This webpage is hosted on a local server and gives a real-time prediction of quadrant numbers. To pipeline the predicted data to ROS, we use Python and Selenium. The final output of the eye tracking module is the quadrant number where the gaze of the user is. In our project we use the EEG data to generate an initial estimate x_k of the user action and the covariance matrix P_{k-1}. We then use the eye-tracking data to calculate our innovation y_k and multiply it with our Kalman Gain K_{k-1} to get our updated state estimate. We also get our updated covariance estimate P_k and our Kalman Gain K_k.

$$x_{k|k-1} = f(x_{k-1|k-1}, u_k) \tag{3}$$

where u_k is the action. $F_k$and H_k are our jacobian matrices.

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{4}$$

We now calculate the innovation and innovation covariance using the eye-tracking data z_k

$$y_k = z_k - h(x_{k|k-1}) \tag{5}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{6}$$

We now calculate our Kalman gain K_k

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{7}$$

Using this we can now calculate our updated state estimate and covariance

$$x_k = x_{k|k-1} + K_k y_k \tag{8}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{9}$$

V. Hardware and Implementation

A. Quadcopter

The Quadcopter that we chose for our tests was a generic DJI F450, which had a X configuration and used Pixhawk 2.4.8 Flight Controller.to power our onboard computer the Intel latte panda along with the Intel Realsense camera we have chosen a 3300 mAh 3C 11.1 V Li-polymer battery because of its optimum discharge rate and lightweight. The propellers being used are the 10x45 size. We use a PID controller with error calculation carried according to intended position by the user at time k+1 denoted by θ^w_{k+1} and present position in the world frame θ^w_k. We display our algorithm in Algorithm 1.

Algorithm 1 Controller

Input: $\theta^w_k, x_l, x_f, x_b, x_r$
Initialize $error_{previous} \leftarrow 0$
Initialize $Integral \leftarrow 0$
if (x_{action} equals $Neutral$) **then**
    Update time $k_{Neutral}$
    if $k_{Neutral} >= 5$ **then**
        Initialize Return to Launch(RTL)
    else
        continue
    end if
end if
Send $velocity$ on the basis intensity of x_{action}
Calculate θ^w_{k+1} from velocity and time dk
$position_{final} \leftarrow \theta^w_{k+1}$
while (θ^w_k—$position_{final}$ is not 0) **do**
    $error \leftarrow \theta^w_k$—$position_{final}$
    $Integral \leftarrow Integral + error * dk$
    $derivative \leftarrow (error - error_{previous})/dk$
    $output \leftarrow Kp * error + Ki * integral + Kd * derivative$
    $error_{previous} \leftarrow error$
    delay(dk)
end while
return: $output$

B. Pixhawk 2.4.8 flight controller

The flight controller being used is the Pixhawk 2.4.8. Pixhawk is an independent open-hardware project providing readily-available, autopilot hardware designs to the academic and research communities. Their boards run PX4 on the NuttX OS. [3] The specifications of the Pixhawk boards processor are 32-bit STM32F427 Cortex M4 core with FPU coupled with 168 MHz/256 KB RAM/2 MB Flash memory and a Coprocessor, the 32-bit STM32F103. [3]

The sensors inbuilt into the board are :

- ST Micro L3GD20 3-axis 16-bit gyroscope
- ST Micro LSM303D 3-axis 14-bit accelerometer/magnetometer
- Invensense MPU 6000 3-axis accelerometer/gyroscope
- MEAS MS5611 barometer

The benefits of the Pixhawk system include:integrated multi-threading, a Unix/Linux-like programming environment, completely new autopilot functions such as sophisticated scripting of missions and flight behavior, and a custom PX4 driver layer ensuring tight timing across all processes [3].

C. Intel Realsense D435i Depth Camera

Intel RealSense Technology is a suite of depth and tracking technologies designed to give machines and devices depth perceptions capabilities that will enable them to "see" and understand the world. There are many uses for these computer vision capabilities including autonomous drones, robots, AR/VR, smart home devices amongst many others broad market products. RealSense technology is made of Vision Processors, Depth and Tracking Modules, and Depth Cameras, supported by an open-source, cross-platform SDK called librealsense that simplifies supporting cameras for third party software developers, system integrators, ODMs, and OEMs. The camera works by projecting a known IR pattern on obstacles in the room and then using an IR camera to look at how this known IR pattern now looks to calculate where obstacles are present. We have successfully been able to carry out SLAM using the realsense2-camera package. [5] [6]

1) Calibration and Setup: Before we start actual control of the quadcopter, both the user and the quadcopter itself undergo a rigorous calibration procedure. The barometric sensors and IMU sensors on the Pixhawk are checked for any offsets and excessive noise. The Intel Realsense D435i is also calibrated to calculate transformation matrices. We then proceed to calibrate to the EMOTIV EPOC+. The user is asked to maintain a neutral state of thought initially to set a baseline. The user is then asked to visualize each drone action, and a dataset is created on the basis of which further classification takes place. A total of five actions are classified. Initializing the system results in the drone rising to a pre-defined height set by the user. The action the user actually wants to achieve, ξ is classified, processed, and the final array after processing through the shared control algorithm is received.

D. Autonomy

The integration of our project with the Robot Operating System (ROS) enabled us to incorporate packages for drone autonomy which ensure safety at all times. For perception, the drone has been equipped with a depth camera that we used for obstacle avoidance and indoor localisation using the Real-Time Appearance-Based Mapping (RTAB-Map) implementation of SLAM [16] [17] [18]. To provide a 3-Dimensional reconstruction of the environment we use OctoMap-the Bayes filter-based, probabilistic mapping and reconstruction algorithm [15] which performs a recursive subdivision into octants and further storing them in an Octree data structure. We also carry out obstacle avoidance based on a monocular vision by semantic segmentation of the image. Since pure vision-based methods are often susceptible to light changes and drift, we incorporate depth information by using the point cloud to laserscan library for accurate perception of obstacles. We assume that the

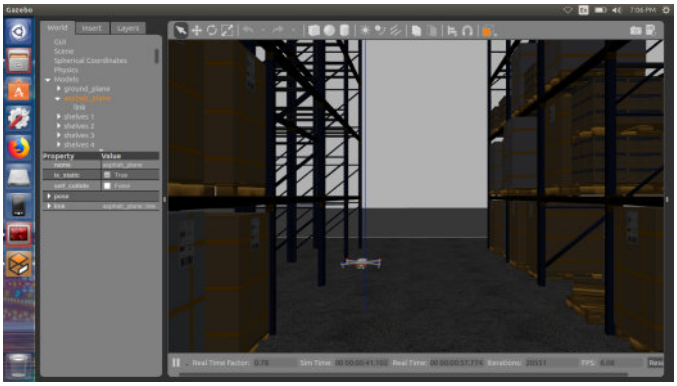

Fig. 7. Simulation of quadcopter in gazebo

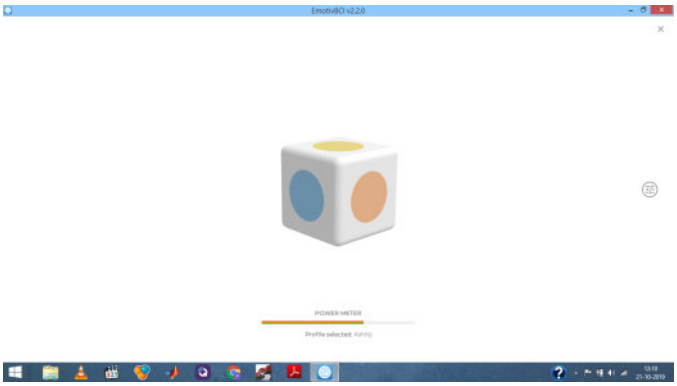

Fig. 8. Representation of User Action Intensity

position of our quadcopter at time k is θ_k^w in the world frame with the origin being the point of launch and $x_l, x_f, x_b and x_r$ being the classified actions. After sampling our environment using a depth camera we classify our space into $freespace$ and $occupiedspace$ Fig. 7 shows the simulation of drone in Gazebo Environment

Algorithm 2 Shared Control Algorithm

Input: $\theta_{k-1}^w, \theta_k^w, x_l, x_f, x_b, x_r$
while (user provides action) **do**
  Calculate updated position θ_{k+1}^w
  if ($\theta_{k+1}^w \in OccupiedSpace$) **then**
    $flag \leftarrow False$
    $\theta_{k+1}^w \leftarrow \theta_{k-1}^w$
  else
    $flag \leftarrow True$
  end if
end while
return: θ_{k+1}^w

E. Optimising battery life and flight path

While under human control the path followed by quadcopter is determined by the final actions classified based upon EEG and eye-tracking data however in the unlikely event of loss of communication the quadcopter will return to the point of

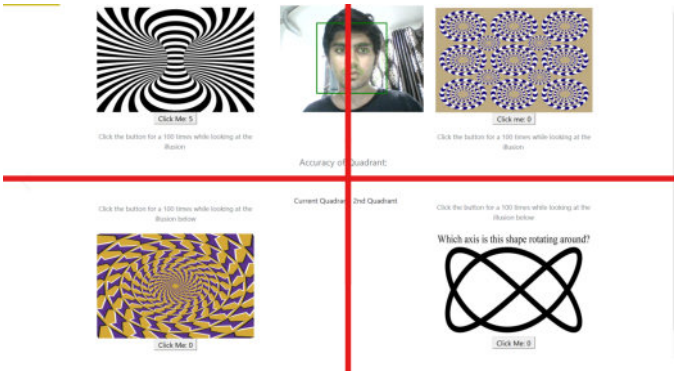

Fig. 9. Division of Webpage according to Cartesian 2D Quadrants

launch after rising to a height of 5 metres. A Simple grid based path-planning algorithm has been used for this. For optimising battery life we have opted for a velocity control method rather than a goal position oriented method so if they action is redundant it is not performed.As an example if the user action is classified as forward movement whereas the user intention is backward motion, it does not wait for the quadcopter to reach its new position for the new command to be sent thus preventing an erroneous redundant path.

VI. Results and Metrics

The metric we considered for measuring the error during eye tracking is the quadrant-wise accuracy, considering that WebGazer.js gives an average error of 175-210 pixels at normal calibration [19] and the quadcopter movement is only along 4 directions. To measure the performance of tracking in this use case, the users are instructed to click on the button while looking at an image above it in each quadrant. This reinforces the ridge regression and customizes it to the users gaze. In a parallel task, on each mouse click, WebGazer first computes the predicted gaze location. This coordinate is compared with the X and Y axis dividing the calibration page into 4 quadrants. The point is categorized into either of 1,2,3,4 depending on the quadrant number as classified in Fig. 9. After 100 clicks on a button, the algorithm computes the number of times the generated prediction was correct in predicting the quadrant number. The count of correct predictions gives the accuracy of prediction in the quadrant This metric allows the error to decrease to almost negligible quantities as our calibration process involves extreme improvement in the ridge regression. With this setup, considering the user is looking at a specific area in the first quadrant, there is approximately a 90% chance that the model correctly predicts the first quadrant. Table 1 below shows the calibration results of one person with spectacles and without spectacles showing that the results are not largely affected due to presence of noisy elements like glasses.

For measuring accuracy we have opted for constructing a covariance matrix that has the intended action one axis and the classified action on the other and the intersection of each displaying the covariance between the two. We calculate this

TABLE I
Average Accuracy

Trial	**A**verage accuracy of quadrants				Overall
	1	***2***	***3***	***4***	***Accuracy***
1	98	96	100	100	98.5
2	100	72	82	92	86.5
3	86	98	98	92	93.5
4	96	96	100	100	98
5	100	100	98	96	98.5

by calculating the $Error\ \eta_k = |\theta^w_k — \theta^w_{k_{ref}}|$ from the reference at every time instant k corresponding to the difference between an intended trajectory given to the user and actual trajectory carried out.

VII. Discussions and Specific Use Conditions

The goal of this project was to develop a robust system for quadcopter control using non-invasive BCI to be used as an assistive device for those with impaired mobility. The application of this project area are diverse and the algorithm can be customized to account for the requirements of a given problem statement. In defence applications it can primarily be used for controlling unmanned vehicles without the need for an actual controller to be operated apart from its secondary application where it can be used to test subject focus under varying degrees of stress and pressure. In the fields of marketing and advertising eye tracking technology may be used to check optimal advertisement placement, attention level of the users while viewing a particular advertisement and to check the benefit of advertising on different platforms.

References

[1] Rabah, M., Rohan, A., Talha, M. et al. Int. J. Control Autom. Syst. (2018): Autonomous Vision-based Target Detection and Safe Landing for UAV

[2] R. Mahony, V. Kumar and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," in IEEE Robotics Automation Magazine, vol. 19, no. 3, pp. 20-32, Sept. 2012

[3] http://pixhawk.org/

[4] https://px4.io/

[5] https://github.com/IntelRealSense/realsense-ros/wiki/SLAM-with-D435i

[6] https://www.intelrealsense.com/depth-camera-d435i/

[7] https://en.wikipedia.org/wiki/Lidar

[8] Riisgaard, Søren ; Blas,Rufus, Morten: SLAM for Dummies

[9] https://dspace.mit.edu/bitstream/handle/1721.1/119149/16-412j-spring-2005/contents/projects/1aslam-blas-repo.pdf

[10] Zunino: SLAM in realistic environments: http://www.nada.kth.se/utbildning/forsk.utb/avhandlingar/lic/020220.pdf

[11] EMOTIV EPOC+ 14 Channel Mobile Brainwear® — EMOTIV", EMOTIV, 2020. [Online]. Available: https://www.emotiv.com/product/emotiv-epoc-14-channel-mobile-eeg/tab-description. [Accessed: 15- Apr- 2020].

[12] https://www.ros.org/

[13] What is Eye Tracking and How Does it Work? - iMotions. https://imotions.com/blog/eye-tracking-work/ 13. Semmelmann, K., Weigelt, S. Online webcam-based eye tracking in cognitive science: A first look. Behav Res 50, 451–465 (2018).

[14] https://doi.org/10.3758/s13428-017-0913-7

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," Autonomous Robots, vol. 34, no. 3, pp. 189–206, Jul. 2013.

[16] Labbé, M, Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. J Field Robotics. 2019; 35: 416– 446. https://doi.org/10.1002/rob.21831

[17] M. Labbé and F. Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," in IEEE Transactions on Robotics, vol. 29, no. 3, pp. 734-745, June 2013.

[18] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, 2011, pp. 1271-1276.

[19] Papoutsaki, Alexandra Sangkloy, Patsorn Laskey, James Daskalova, Nediyana Huang, Jeff Hays, James. (2016). WebGazer: Scalable Webcam Eye Tracking Using User Interactions.

[20] Gwon SY, Cho CW, Lee HC, Lee WO, Park KR. Gaze tracking system for user wearing glasses. Sensors (Basel). 2014;14(2):2110-2134. Published 2014 Jan 27. doi:10.3390/s140202110

[21] P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao, "TurkerGaze: crowdsourcing saliency with webcam based eye tracking," http://arxiv.org/abs/1504.06755.

[22] Audun Mathias. clmtrackr: Javascript library for precise tracking of facial features via Constrained Local Models. https://github.com/auduno/clmtrackr, 2014. [Online; accessed 2015-07-08].

[23] Chen ZH, Fu H, Lo WL, Chi Z, Xu B. Eye-tracking-aided digital system for strabismus diagnosis. Healthc Technol Lett. 2018;5(1):1-6. Published 2018 Jan 26. doi:10.1049/htl.2016.0081