# *Mata*: An Android Eye-Tracking Based User Interface Control Application

Yaya Heryadi
GameLab, Computer Science Program
Bina Nusantara University
Jakarta, Indonesia
yayaheryadi@binus.edu

Michael James
Computer Science Program
Bina Nusantara University
Jakarta, Indonesia
michael.james@gmail.com

***Abstract* — the advent of smartphone technology has provided us with intelligent devices for communication as well as playing game. Unfortunately, applications that exploit available sensors in the smartphone are mostly designed for people with no physical handicap. This paper presents *Mata*, a game user interface using eye-tracking to operate and control games running on Android smartphone. This system is designed to enhance user experiences and help motoric impaired peoples in using smartphone for playing games. Development and testing of the *Mata* system has proven the concepts of eye-tracking and eye-gazing usage as unimodal input for game user interface.**

**Keywords—eye gaze tracking, game user interface**

## I. Introduction

In the past decade, eye gaze tracking has gained research interest from various research communities as it provides a means of user input for man-machine interaction. The popularity of eye gaze tracking among game researchers is due to several reasons. First, game user interface that uses eye gaze tracking can increase the game accessibility to people with physical impaired [1]. Second, the eye gazing patterns recorded by game user interface can be used for various purposes ranging from evaluating game design [2], recognizing emotion in learning process [3], recognizing the effect various control stimuli in social interaction context [4] to controlling engine workload [5].

Game user interface in general refers to the space where interactions between humans and computer games (games) take place. Game user interface is very importance in facilitating two way human-computer interactions: users can operate and control of the games effectively and games can feeding back information so that the users can make appropriate decision. With the advent of game technologies in the last decade, game has many potential applications in various domains such as education, entertainment, and behavioral change. Therefore, research in game user interface to develop robust, efficient, natural interaction, and user-friendly game user interface is also very challenging.

The pervasive intelligence devices such as smartphones in the last decade have motivated many researches on game user interface to develop more adaptable user interface for the use of various player categories regardless their physical abilities, ages, and genders. Thanks to the advent and proliferation of mobile phone technologies. For example, touch screens [6]. This user interface technology makes it possible for human to interact with a computer system with less user intervention and effort to communicate with a computer system yet it gives better user experience because the user feels more natural navigation control. The drawback of this technology in game environment is mainly the user should use their hand(s) to play games. This requirement hinders its usability for physical handicap game players.

Adopting eye gaze tracking in game user interface have gained research interest from various research communities due to its adaptability for physical handicap such as motoric impaired people who has limitation in using their hands but still can use their eyes [7]. The main challenge of UI using eye movement is performance of eye-detection and eye-tracking methods, which are part of eye-gazing recognition method, based on the image data captured by camera which is often affected by lighting, pose, camera angle, and occlusion.

Despite a vast number of user interface technologies have been reported, to the best of our knowledge, there are only a few studies on designing game user interface for Android. The objective of this research is to develop a prototype (proof of concept) of eye gaze tracking-based game user interface to control games running on Android and gain feedback on its responsiveness and accuracy from users.

The rest of the paper is organized as follows. Chapter 2 describes some related works. Chapter 3 explains the system design. Chapter 4 explains the experiment result followed by conclusion and future works in Chapter 5.

## II. Related Works

Eye gaze tracking based user interface have gained wide researcher attention since 80's resulted in numerous publications. The study by Hutchinson *et al.* [8], for example, might be among prominent research in human-computer interaction using eye-gaze input. Another prominent study has been reported by Wang [9] who utilizes multimodal input including eye-gaze, voice and manual response in the design of user interface. The proposed method is demonstrated in developing a device and application-independent model for 3D cube manipulation task. Since then, a vast number of researches in eye-tracking and eye-gazing to solve various problems in the context of various domains have been reported.

The study by Eid *et al.* [10] explores an eye-gaze-controlled method for navigating wheelchair system in unknown environments. Kurauchi *et al.* [11] propose EyeSwipe which is a dwell-free Text entry using gaze paths. The proposed method makes it possible for motor impairment peoples to interact with computer such as typing text. Sharma *et al.* [12] report an experimental finding in attempt to understand the behavior of the operator in a typical chemical plant control room using the information obtained from eye tracker. Experimental studies reveal that fixation patterns contain signatures about the operators learning and awareness at various situations. These research findings on human error in process plant operations are then used as basis to develop a more appropriate method to reduce incidents due to human error in operating a control room.

The increasing interests among academics and industries in developing eye-gaze based human-computer interaction has motivated Giannopoulos *et al.* [13] to develop viGaze, which is a framework to exploit the use of eye gaze as an input modality in various contexts. The framework provides features necessity to develop eye-gaze based human-computer interfaces which explore explicit and implicit interactions in complex virtual environments.

From industry side, a number of commercial products based on eye-tracking also has emerged in the market with various shapes and sizes among others are: Eye-tribe [14], Eyegaze Edge [15], DynaVox EyeMax [16], Eyewrite-Tobii [17],

Although there has been a vast number of eye-gaze based user interface method and technologies have been reported, the novelty of the proposed studies is that the proposed eye gaze tracking based game user interface is designed for controlling game while exploiting available smartphone sensors and Android SDK. In addition, the resulted user interface is intended to be freely available for researchers in the Human-Computer Interface area.

## III. System Design

Following Alavi [18], development of this system uses prototyping approach due to several reasons mainly: (1) it gives a tangible means of comprehending and evaluating the proposed system, (2) it provides a common baseline to identify potential problems and opportunities early in the development process, (3) it gives a practical way to cultivate user participation and commitment to the project, (4) it helps to ensure that the core of the developed system is right. The main development steps are as follows.

### A. *Preliminary Survey.*

This step aims to collect and analyze prominent methods and challenges on eye-detecting and tracking from facial image from available literatures.

### B. *User Requirements Analysis.*

The main objective of this step is to formulate several requirements such as: the most convenience eye-blink to imitate touch screen operations, system functional/ non-functional requirements. This step is implemented in collaboration with targeted users. The system requirements are further represented by means of: use case diagram, use case definition, use case description, functional/nonfunctional requirement tables.

### C. *System Architecture Development.*

This step aims to identify hierarchical structure of the system comprises: (1) application services, (2) view, and (3) logic layers (see Fig. 1). The main components of the logic layer are implemented using the following methods.

*1). Face detection method.* In this research, face detection from input image is implemented using a method proposed by Jones and Viola [19] especially with regards to: Haar features, integral image concept, and cascading classifier.

*a). Haar Feature Selection.* Following Jones and Viola [19], features used in this research involve the sums of image pixels within rectangular areas which are used to match regularities of human faces (see Fig. 2). There are three type features: (1) Two-rectangle feature computes the difference between the sum of the pixels within two rectangular regions; (2) Three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle; and (3) Four-rectangle feature computes the difference between diagonal pairs of rectangles.

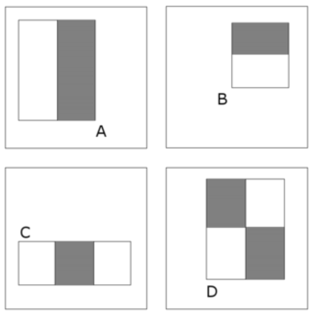

Fig. 2. Haar Feature for Object Detection (Source: [19])

*b). Creating an Integral Image.* Integral images is a term proposed by Jones and Viola [19] which refers to intermediate representation of an image used to speed up computing rectangular features (see Fig. 3). The authors described integral image as "*the sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4+1.*"

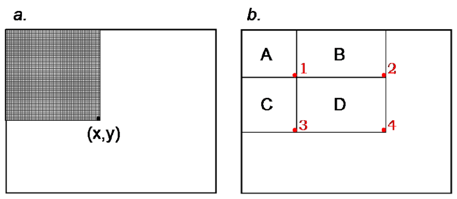

Fig. 3. Integral Image (source: [19]).

*c). AdaBoost Training.* Following Jones and Viola [19], this research uses AdaBoost to select the features and to train the classifier. AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm by combining a set of weak classification functions to form a stronger classifier. The strength of AdaBoost has been reported in a prominent study by [20] who conclude that training error of the strong classifier approaches zero exponentially in the number of rounds.

*d). Cascading Classifiers.* As it has been described in Jones and Viola [19], the strength of cascading classifier is its ability to construct smaller and efficient boosted classifiers which is capable to reject negative sub-windows while detecting almost all positive instances. The simpler classifiers; therefore, is designed to reject most of sub-windows before more complex classifiers are used to achieve low false positive rates. Algorithm for training cascading algorithm can be described as follows.

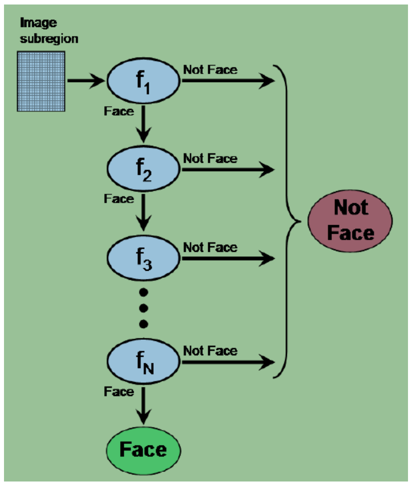

Fig. 4. Scheme of Cascading Classifier

*2). Gaze detection method.* Method for detecting gaze is implemented using the following algorithm (see Fig. 5).

```
/* Compute both eye's area, coordinates, and sizes */
Initialize threshold=5
A = Gaze Horizontal Coordinate
B = Left Eye Horizontal Coordinate
C = Left Eye Horizontal Calibrated Center Coordinate
D = Right Eye Horizontal Calibrated Center Coordinate
E = Gaze Vertical Coordinate
F = Left Eye Vertical Coordinate
G = Left Eye Vertical Calibrated Center Coordinate
H = Right Eye Vertical Coordinate
I = Right Eye Vertical Calibrated Center Coordinate

If both eyes are detected then
    If both eyes are not looking at the same direction horizontally left or right then
        A = B – C – D – E; else A = 0.5* (B – C – D – E);
    If both eyes are not looking at the same direction vertically up or down then
        E = F – G – H – I ; else E = 0.5*( F – G – H – I);

/* Compute eyes area horizontally and vertically */
WLEA = Width Left Eye Area;     HLEA = Height Left Eye Area
WREA = Width Right Eye Area;    HREA = Height Right Eye Area
Eyes area horizontal size = (WLEA+WREA)/threshold
Eyes area vertical size = (HLEA+HREA)/threshold
WSS =Width of Screen Size
EAHS = Eyes Area Horizontal Size
HSS =Height of Screen Size
EAVS =Eyes Area Vertical Size

/* Compute estimated gaze coordinate (horizontally and vertically */
A = (A * WSS/EAHS) + 2 * WSS
E = (A * HSS/EAVS) + 2 * HSS
```

Fig. 5. Pseudocode of Gaze Detection Algorithm

*3). Blink detection method.* Method for detecting eye blink is implemented using the following algorithm (See Fig 6.)

*4). Testing.* This step comprises of functional, module, integration, and system tests. In addition, user acceptance test which are implemented using qualitative survey involving respondents which are selected using purposive sampling. The user acceptance testing is focused on system responsiveness and accuracy.

## IV. Experiment Result

### A. Eye-blinking Identification.

In order to simplify the research, it is assumed that the user's eyes are visible (not occluded by any object such as eye glasses). Movement of the detected user's eyes is then used to control user interface elements which imitate window operation using screen touch or mouse clicking in a conventional user interface.

Three type of screen operations are implemented which are (1) eye blink for normal screen touch for pointing a window object; (2) eye blink for long screen touch for opening window menu; and (3) eye gazing to move cursor in a window. Brief survey involving 7 (seven) male and female students found that left eye blink (71%) is more convenient for normal screen touch and right eye blink (71%) is for long screen touch. These eye blink preferences correspond to the fact that most of respondents are right-handed peoples who get use to perform selection using left mouse button and displaying alternative selection using right mouse button. Eye gazing is assumed to be implemented by moving eyes across the screen.

### B. State Diagram.

In response to a series of events in user interface as external stimuli, behavior of classes can be modeled using state diagram (see Fig. 7). As can be seen from the diagram, the user interface system can be in one of the main states as follows: main menu, service started, control navigation pointer, screen-touch imitated, calibration page, and service stop. A labeled solid arrow in the diagram represents the path between different states of an object due to the triggered event and produces the resulted action.

### C. System Activity Diagram.

General logic of the program can be described using high level activity diagram (see Fig. 8). As can be seen from the diagram, when the system starts, the user is given three options: (1) start services, (2) calibrate tracker, and (3) stop service.

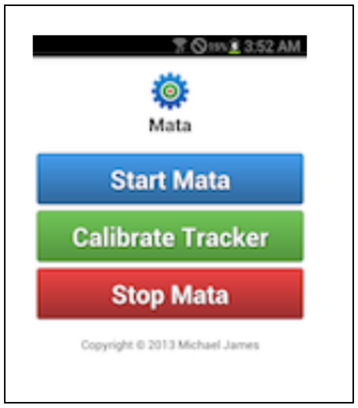

Fig. 9. Main menu of the *Mata* Eye-tracking System.

### D. *User Acceptance Testing.*

In order to evaluate the prototype, a simple qualitative survey using self-administered questionnaires toward user acceptance testing is implemented. The main objectives of quantitative testing involving users are to gain feedback on:

*1).* Responsiveness: how fast the system triggers a navigation pointer movement based on eye gaze tracking and imitates screen touch event based on blink detection.

*2).* Accuracy: how accurate is the eye gaze tracking to move the navigation pointer to the same correct coordinate as the eye gaze of the user. Low accuracy means that most of the time the pointer does not go to the same coordinate with the eye gaze of the user. High accuracy means that most of the time the pointer goes onto the same correct coordinate which the eye gaze points.

This information will be used for refine the game user interface further.

Respondents for this survey were 7 (seven) college students which were selected purposively. Each questionnaire uses 1-10 Likert scale. The main survey findings are: (1) the prototypes is high responsive (score 8) and (2) the eye-tracking is accurate (score 7.4). Among of comments from respondents are: *"… generally satisfied with the responsiveness"*, *"accuracy is still below expectation"*, and *"a well-lit area is not always easy to be fulfilled."* These feedbacks on responsiveness and accuracy show that the prototype of game user interface based on eye gaze tracking is promising to be improved one step further, e.g. combining with a computer game.

### E. *Research Limitation.*

Low number of samples has become a limitation in generalizing many indications found in the research finding. Although the empirical results validates some of the previous result studies reported by [19], performance of the system is still suffered in several cases: (1) the face is not located right in front of camera or frontal image, (2) the face is slightly rotated around both around the vertical and horizontal axis, and (3) the face image is captured with low light intensity.

## V. Conclusion and Future Work

Development and testing of the *Mata* system has proven the concepts of a user interface using eye-tracking as unimodal input for game user interface running on Android smartphone. Low accuracy found from experiment suggest the future research should be focused, among others, on improvement particularly in the area of eye detection and eye tracking methods so it can robustly handle face image taken in low light intensity, rotated images, and partial occlusion conditions.

## References

[1] Velloso, E., Fleming, A., Alexander, J. and Gellersen, H. "Gaze-supported gaming: MAGIC techniques for first person shooters." Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play. ACM, 2015.

[2] Tsai, M.J., Huang, L.J., Hou, H.T., Hsu, C.Y. and Chiou, G.L. "Visual behavior, flow and achievement in game-based learning." Computers & Education 98 (2016): 115-129.

[3] Manssuer, L. R., Pawling, R., Hayes, A. E., and Tipper, S. P. "The role of emotion in learning trustworthiness from eye-gaze: Evidence from facial electromyography," Cognitive neuroscience, 7(1-4), 82-102, 2016.

[4] Saitovitch, A., Popa, T., Lemaitre, H., Rechtman, E., Lamy, J. C., Grévent, D. and Boddaert, N. "Tuning Eye-Gaze Perception by Transitory STS Inhibition," Cerebral Cortex, 26(6), 2823-2831, 2016.

[5] Mannaru, P., Balasingam, B., Pattipati, K., Sibley, C., and Coyne, J. "Cognitive context detection in UAS operators using eye-gaze patterns on computer screens," In SPIE Defense+ Security (pp. 98510F-98510F). International Society for Optics and Photonics, 2016.

[6] Bellis, M. "Touch Screen," [Online:] Available: http://inventors.about.com/library/inventors/bltouch.htm

[7] Anacan, R., Alcayde, J. G., Antegra, R., and Luna, L. "Eye-GUIDE (Eye-Gaze User Interface Design) Messaging for Physically-Impaired People," arXiv preprint arXiv:1302.1649, 2013.

[8] Hutchinson, T. E., White, K. P., Martin, W. N., Reichert, K. C., and Frey, L. A. "Human-computer interaction using eye-gaze input," IEEE Transactions on systems, man, and cybernetics, 19(6), 1527-1534, 1989.

[9] Wang, J. "Integration of eye-gaze, voice and manual response in multimodal user interface," In Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on (Vol. 5, pp. 3938-3942). IEEE, 1995.

[10] Eid, M. A., Giakoumidis, N., and El Saddik, A. (2016). A Novel Eye-Gaze-Controlled Wheelchair System for Navigating Unknown Environments: Case Study With a Person With ALS. IEEE Access, 4, 558-573.

[11] Kurauchi, A., Feng, W., Joshi, A., Morimoto, C., and Betke, M. "EyeSwipe: Dwell-free Text Entry Using Gaze Paths," In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 1952-1956). ACM, 2016.

[12] Sharma, C., Bhavsar, P., Srinivasan, B., and Srinivasan, R. “Eye gaze movement studies of control room operators: A novel approach to improve process safety,” Computers & Chemical Engineering, 85, 43-57, 2016.

[13] Giannopoulos, I., Schöning, J., Krüger, A., and Raubal, M. “Attention as an input modality for Post-WIMP interfaces using the viGaze eye tracking framework,” Multimedia Tools and Applications, 75(6), 2913-2929, 2016.

[14] Dalmaijer E. “Is the low-cost EyeTribe eye tracker any good for research?,” [Online:] https://doi.org/10.7287/peerj.preprints.585v1, 2014.

[15] LC Tech. “Communicate with the world using the power of your eyes,” [Online:] http://www.eyegaze.com/eye-tracking-assistive-technology-device/, 2016

[16] Dynavox. “Say it with your eyes,” [Online:] http://www.dynavoxtech.com/products/eyemax/benefits/, 2016.

[17] Wobbrock, J. O., J. Rubinstein, M. Sawyer, and A. T. Duchowski. "Not typing but writing: Eye-based text entry using letter-like gestures," In Proceedings of The Conference on Communications by Gaze Interaction (COGAIN), pp. 61-64. 2007.

[18] Alavi, M. "An assessment of the prototyping approach to information systems development." Communications of the ACM 27, no. 6, pp. 556-563, 1984.

[19] Jones, M. J., and Viola, P. “Robust real-time object detection,” In Workshop on statistical and computational theories of vision (Vol. 266, p. 56), 2001.

[20] Freund, Y. and R. E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting,” In Computational Learning Theory: Eurocolt ’95, pages 23–37. Springer-Verlag, 1995

## APPENDICES

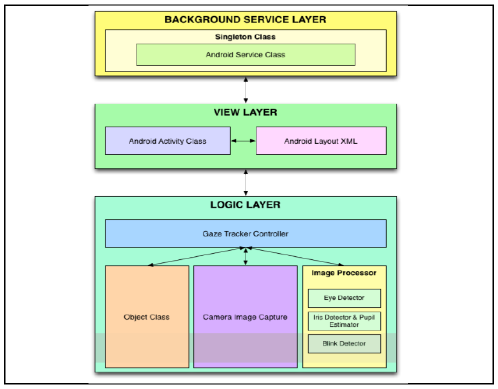

Fig. 1. System Architecture of *Mata*

```
Initialize max-counter = 3
If two eyes are detected then
    Last- condition ← two-eyes-detected;
    Left, Right, No-eye counters ← 0;
    Face object ← detected eyes (left or right);
    Compute center of the detected eyes.
Else if one-eye-detected then
    Retrieve the detected eye
    If H-coord < H-center-coord then
        If Last- condition NOT two-eyes-detected then
            Left, Right, No-eye counter ← 0; Return FALSE.
        Else
            Left counter++;
            Right and No-eye counter ← 0
            If (Left counter > max-counter) then
                Left counter ← 0
                Last-condition ← Left-eye-blink
                Return TRUE
            Else
                Return FALSE
    Else
        If Last-condition is NOT two-eyes-detected then
            Left, Right, No-eye counter ← 0
            Return FALSE
        Else
            Right-counter ++;
            Left, No-eye counter ← 0
            If (Right counter > max-counter) then
                Right counter ← 0
                Last-condition ← Right-eye-blink
                Return TRUE
            Else
                Return FALSE
    Else
        if No-eye is detected then
            If Last-condition is NOT two-eyes-detected then
                Left, Right, No-eye counter ← 0
                Return FALSE
            Else
                No-eye counter ++;
                Left, Right counter ← 0
                If (No-eye counter > max-counter) then
                    No-eye counter ← 0
                    Last-condition ← Both-eyes-blink
                    Return TRUE
                Else
                    Return FALSE
        Else
            Last-condition ← Two-eyes-detected
            Left, Right, No-eyes counter ← 0
```

Fig. 6. Pseudocode of Blink Detection Algorithm

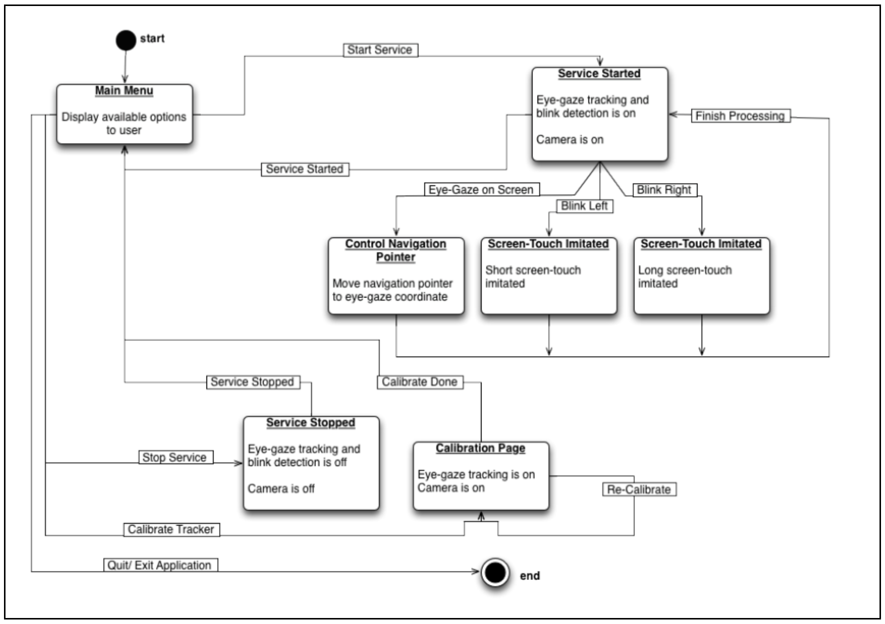

Fig. 8. State Diagram of *Mata*

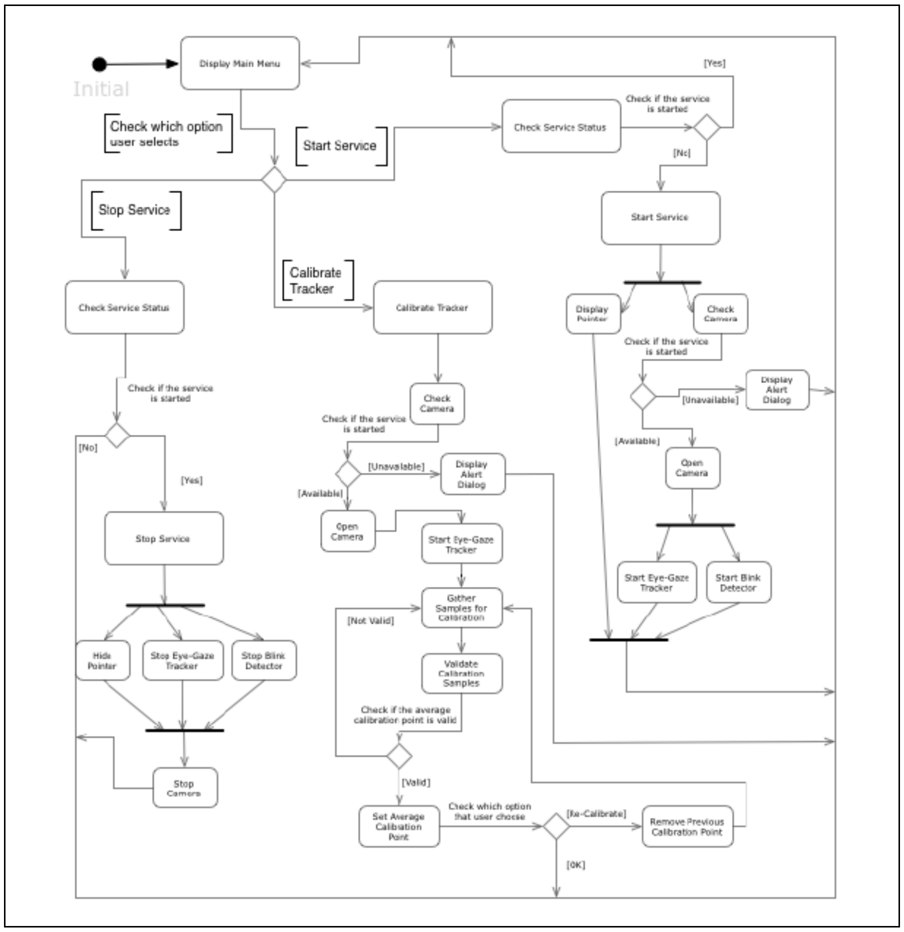

Fig. 9. System Activity Diagram of *Mata*