IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 25, 2021, accepted April 6, 2021, date of publication April 9, 2021, date of current version April 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072195

Developing a Novel Hands-Free Interaction Technique Based on Nose and Teeth Movements for Using Mobile Devices

MUHAMMAD NAZRUL ISLAM[1], MD SHADMAN AADEEB[1], RAFIUR RAHMAN KHAN[1], MD. MAHADI HASSAN MUNNA[1], MAHMUD SARWAR[1], SHAMIMA NASRIN[1], AND A. K. M. NAJMUL ISLAM[2]

[1]Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST), Dhaka 1216, Bangladesh
[2]LUT School of Engineering Science, LUT University, 53850 Lappeenranta, Finland

Corresponding author: A. K. M. Najmul Islam (najmul.islam@utu.fi)

ABSTRACT Human-mobile interaction is aimed at facilitating interaction with the smartphone devices. The conventional way to interact with mobile devices is through manual input where most of the applications are made assuming that the end user has full control over their hand movements. However, this assumption excludes people who are unable to use their hands or have suffered limb damage. In this paper, we proposed a nose and teeth based interaction system, which allows the users to control their mobile devices completely hands free. The proposed system uses the front facing camera of the smartphone to track the position of the nose for cursor control on the smartphone screen. The system detects teeth for performing the touch screen events such as tap, scroll, long press, and drag. *Viola-Jones* algorithm is used to detect the face and teeth based on the Haar features. After detecting the face, the nose position is calculated and tracked continuously using *Lucas Kanade's* method for optical flow estimation. All the touch screen events have been implemented in the system so that the user can execute all the operations of the smartphone. To evaluate the performance and the effect of (smartphone) device type on the execution time, the proposed system was installed in 3 smartphone devices and 7 trials for each device were performed by 3 different able-bodied elderly persons. The result shows a significant success rate for the detection of nose and teeth, and for the execution of the operations. The execution time of each operation slightly varies by 0.72s on average because of the configuration of the smartphones.

INDEX TERMS HCI, human-mobile interaction, gesture operations, disabled user, accessibility, mobile device, smartphone.

I. INTRODUCTION

Human Computer Interaction (HCI) is a field of study in computer science, which is dedicated to facilitate the interaction between computing devices (desktop computers, mobiles, etc.) and the users [1]. It has become a very important research field due to the fact that HCI makes systems more functional, safe and increases the usability and user experience (UX). Furthermore, in recent times, research interests on real time HCI have been growing [2].

Smartphones are widely adopted and used around the world. According to a recent statistic, the number of smartphone users in the world is 3.5 billion, which means 45.04% of the world's population own a smartphone and it is expected that by the year 2021 there will be 3.8 billion smartphone users [3]. But not much has been done to enable a user to use the smartphone completely hands free.

People with different conditions like cerebral palsy, neurological injury or stroke, disability of the limbs, loss of arms due to accidents or other hand related disabilities are unable to interact with desktop computers and smartphones in the usual way [1], [4]–[6]. Research in the field of HCI has introduced various alternative means of interaction with mostly desktop computers through different facial components like nose, eyes, teeth, etc. Among these, most of the articles focused on the nose and teeth for interaction with desktop computers [1], [2], [7] whereas, voice based interaction is used in some mouse operations [7]. Some other works focus on the use

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

of eye movements to interact with desktop computers [4], [5], [8] and eye blinking to interact with smartphones [6]. An Android-based application has also been developed to use smartphone using only the movement of the human-face [9].

The prior works thus confirm that there are few systems that explicitly focus on the interaction between the disabled users and their smartphones, even though the number of mobile users around the world are more than desktop users. To aid these people (disabled mobile users), HCI can play a very important role. Thus, an innovative hands-free system is required to be developed for the smooth interaction with smartphone for these users. Therefore, the objective of this research is to develop a nose-mobile-interaction (NMI) system for disabled people to facilitate effective and efficient interactions to seamlessly operate mobile devices or smartphones. To attain this research objective, an NMI system is developed to enable a person, who does not have the ability to use his or her hands freely, to use the smartphone by means of cursor pointer manipulation. A 'nose-teeth' interaction approach is adopted to develop the NMI system where a cursor pointer can be moved by the user's nose and the user will be able to execute different smartphone operations like tap, drag, swipe, scroll, etc. by only showing the teeth. As such, the term 'nose-teeth' is referred to the means of interaction with smartphone through the user's nose and teeth.

This article has been organized as follows. The literature review has been presented in LITERATURE REVIEW section. The development and design of the system has been provided in DEVELOPMENT OF THE SYSTEM section. DEMONSTRATION section presents the various use cases of our system. In DISCUSSION section, results of the experiment are discussed and concluding remarks are presented in CONCLUSION section.

II. LITERATURE REVIEW

A number of research has been conducted focusing on the alternate means of interaction between human and computers or mobile devices; some of which were concluded by highlighting the important issues and possible research scopes for further investigation.

A. INTERACTION WITH DESKTOP COMPUTER

A solution with assistive technologies is presented in [7]. Here, the *Viola-Jones* algorithm [10] is used for the detection of face, and nose tracking is done with *template matching* method [11]. For voice interaction, speech Application Program Interface (API) is used to detect and identify the voice commands. These voice commands are mapped to various mouse related functions. Similarly, in [1], *Viola-Jones* algorithm is adopted to control the mouse cursor by moving the user's nose, and the click function is activated on the detection of teeth. The system proposed by Lal & Chiddarwar [2] tracked the user's nose for cursor control, eye blink is detected for click operation and smile is detected for scrolling of the pages. In [2], *face landmark detection* algorithm [12] is used to localize facial features for controlling the computer. However, in low light conditions, the face landmark detection accuracy decreases, which leads to missing of few eye blinks, and the scrolling works only if a big smile is detected. Again, in [13] authors implemented a vision-based face tracking system that allows the user to control the cursor. The cursor movement is controlled with the nose, and clicking is simulated by opening the mouth. The location of the nose tip is estimated within a small region below the eyes. Systematic evaluation of the functionality of the clicking operation is yet to be done.

Eye movements and the opening and closing of the eyes are used for cursor movement [4], [8]. In [8], no mouse action is proposed, but eye blinking is used for executing a mouse action in [4]. In this case *SVM*, classification method [14] is used for classifying the eye movements. Using face recognition and extraction of user eyes' location as a method is proposed in [5]. The machine learning model is trained manually using *Adaboost* algorithm [15] based on the skin colour feature. Point feature is tracked over the video frame by *Lucas Kanade* algorithm [16] & rotation matrix and translation vector is used to calculate the location of pointer for creating click events [17]. A system is proposed in [18], where head movement is used to control the cursor and speech recognition is used to perform mouse action. In [19], a Convolutional Neural Network (CNN) model is proposed for face point detection. For facial expression recognition, they use an end-to-end learning model that simultaneously performs facial image synthesis and gesture-invariant facial expression recognition. For sight tracking, they mathematically describe the relationship between the possible center of eye and the direction of all image gradients. Similarly, in [20], a system is developed to allow the user to interact with an application using hand gestures. Also, computer access becomes a very challenging task for the people having limited head movement, since the users depend on single head-gesture to interact with a computer. Very recently to address this limitation, Esiyok *et al.* [21] proposed and evaluated two novel interaction techniques namely HeadGyro and HeadCam based on software switch approach. In this case, the HeadGyro software switch showed slightly higher performance than the HeadCam. Again, in [22], a system is designed to control the movements of the mouse cursor by capturing the head movement. Their study shows that using CNN, the accuracy for head classification is greater than eye blink classification and CNN models perform better than the multilayer perceptron and Histogram of oriented gradients-Support Vector Machine models.

B. INTERACTION WITH SMARTPHONE

There is little research done till now using different facial components to enable users to interact with their smartphone. For example, a system is proposed in [6] for the disabled users so that they can make phone calls completely hands-free. This system presents a real time method based on some video and image processing algorithms for eye blink detection. If the eyes are closed, it means the eyes blinked, and a call to a

TABLE 1. Summary of existing tools in terms of their features and objectives.

Reference	App or hardware device	Features of the tools	Remarks
[9]	EVA Facial Mouse Pro	In this app face movement is required for cursor movement. There is a UI menu for selecting a gesture.Click occurs automatically if the cursor is kept stationary for some time.	Time consuming and automatic clicking can result in accidental click at any times.
[34]	GlassOuse	It is a Bluetooth mouse that's worn like glasses. Based on user's head movements, it moves the cursor on screen. It has a switch which is required to put inside the mouth to perform click operation by biting on it.	The device is expensive.
[35]	QuhaZono 2	Head movement is used to move cursor on screen. There are multiple wearing options for head and limb attachment. Gestures are needed to activate common features such as cursor pausing, window scrolling etc.	The device is expensive.

specific phone number will be made, otherwise no phone call will be made [6]. In another system [23], hand gestures are used to give commands to the mobile device. Here *skin colour detection* algorithm based on HSV color space [24] is applied to detect the color of both the face and the hand in an image. In [25], a systematic literature review is conducted focusing on various foot based interaction techniques to highlight the potential applications of foot-based interfaces for mobile device.

C. EMBEDDED SYSTEM TO INTERACT WITH DESKTOP COMPUTER

Few other studies have focused on the development of embedded systems to interact with desktop computers. For example, an embedded system is implemented in [26] where, magnetometer and accelerometer sensors are used to measure head movements for controlling cursor movements and a flex sensor is attached with the user's cheek muscle, so that, when it bends, a click event occurs. In [27], a system based on the EPOC+ device to control the computer with facial expressions and motion sensors is developed for motor impaired users. Similarly, Abiri *et al.* [28] discussed a Brain Computer Interaction (BCI) based interaction system for controlling the cursor using EEG signals. Another interaction technique based on a person's respiration is presented in [29] where two methods are used to create an event in the system for interaction. In one of the two methods, a microphone is used to detect hard exhalation (person's breath) by the user. In the other method, a thin material is placed in front of a camera. On that material, a quadrilateral is drawn. Upon exhalation by the user, the material expands and the quadrilateral size increases. This change of quadrilateral size is detected by the camera and this results in exhalation detection.

D. INTERACTION WITH OTHER DEVICES

A few other recently published articles focused on different ways to interact with other devices such as controlling robots remotely, interacting with a virtual environment, controlling 3DTV, etc. For example, in [30], a system is proposed to control a robot from a distance using hand gestures in LMC (Leap Motion Controller) on the desktops. Similarly, in [31], BCI and AR technologies are combined together to interact with robots remotely. In [32], eye gaze tracking is used to pinpoint a location in a virtual environment and using forearm muscle contractions, an action can be performed on that location. The main limitation of this system is the accuracy of eye-gaze tracking in VR. Human-3DTV interaction system is presented in [33], which enables users to control the television by directly touching a virtual interface with simple free-hand gestures.

E. EXISTING TOOLS/APPLICATION FOR HANDS FREE INTERACTION WITH SMARTPHONE

A few tools and applications are also available for hands-free cursor control. A summary of the existing tools in terms of their features and objectives are provided in Table 1. The first one is an application called EVA facial Mouse Pro. It tracks the human face to move the cursor. The other two are hardware tools to perform the functions of a computer mouse. Both use head movements to move the cursor, both are light-weight but very expensive.

F. EVALUATION AND COMPARING THE SYSTEM PERFORMANCE

A few other studies were conducted that focused on comparing the performance between different types of interaction. In [36], a usability study is conducted to explore which approach is better between the eye movement approach and the nose tracking approach. The study showed that using eye movement the user can actually move the cursor horizontally, but cannot properly move it vertically as the pupil movement is limited. On the other hand, with a nose tracking approach, the user is able to move the cursor to the desired position easily. To perform mouse click with an eye blink, a few attempts are needed for the user to get the desired result.

The various Algorithms and techniques used in such kind of research is shown in Table 2, which shows that *Viola-Jones* algorithm is used most to detect facial features and *Lucas Kanade optical flow* algorithm, *template matching* methods are used for tracking facial features. Patel and Shukla com-

TABLE 2. Algorithm or method or technique used in previous research.

Algorithm / Method / Technique	Purpose	References
Viola-Jones algorithm	To detect face	[1], [4], [7], [8], [23]
Viola Jones algorithm using Haar Like feature	To detect teeth To detect smile To detect eye	[1] [2] [6]
Lucas Kanade optical flow algorithm	To track point feature	[5], [17], [18], [20]
Template matching method	To track nose. To track eye	[7] [4], [8]
SVM classification method.	To classify eye movement	[4], [8], [23]
Adaboost algorithm	To train classifier. To detect face	[2] [5]
Face landmark detection algorithm	To localize facial features.	[2]
Skin Colour algorithm.	To detect hand and face.	[23]

pared the optical flow algorithm [37] which showed that *Lucas Kanade* algorithm is robust in presence of noise and angular error is less than *Horn-Schunck* algorithm.

In summary, the existing research has provided a number of issues. Firstly, most of the studies focus on the interactions with desktop computer using facial gestures. Secondly, a limited number of works focus on smartphone interaction. Moreover, these studies do not properly investigate on how to perform all the basic operations of smartphone using facial gestures. Thirdly, various alternate means of interaction like nose, eye, teeth, etc. are proposed to enable users to interact with different kinds of devices. In this case, NMI performs better than eye computer interaction. Fourthly, a few embedded or hardware based systems are proposed but these are mostly for interacting with desktop computers and other devices like robot,3DTV, etc. Such solutions are based on additional hardware devices and sensors which are very costly. Fifthly, some research do not focus on the disabled people and consider their limitations. For example, some people may not have hands and hence, hand gesture based interactions are not applicable for them. However, analyzing the bottlenecks and hindrances of these above mentioned problems and to mitigate the research gap to some extent, we set the objective of this research to develop a nose-teeth based mobile interaction system. The proposed system does not require any additional hardware device for controlling an android smartphone.

III. DEVELOPMENT OF THE SYSTEM

In this section, the development process of NMI system is discussed in two phases namely (a) cursor movement by nose and (b) teeth detection and gesture operation execution. This is in accordance with the aim of this research. In both phases, image frames are continuously taken in real time using the system's (smartphone's) camera.

A. CURSOR MOVEMENT BY NOSE

This is the first phase of the development of the system. A basic flowchart explaining the process of cursor movement is shown in Fig. 1.

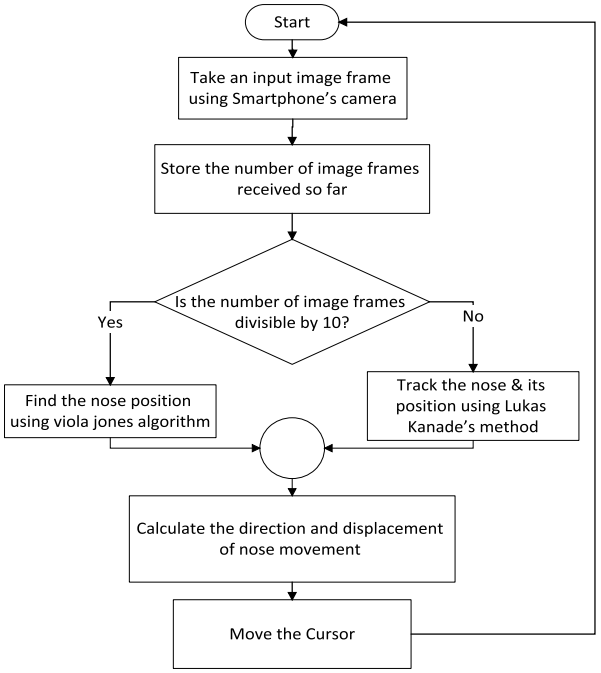


FIGURE 1. A basic flow chart to represent the process of cursor movement.

In order to move the cursor by nose, firstly an image frame is taken using the smartphone camera. The image is then converted into a grayscale image. The grayscale image is then stored as a matrix. The number of elements in the matrix is equal to the total number of pixels in the grayscale image. Each element of the matrix contains the intensity value of its corresponding pixel in the grayscale image. The intensity values range from 0 to 255 with 0 being pure black and 255 being pure white. Image frames are taken continuously from the smartphone camera. A counter is kept to keep track of the number of image frames processed so far.

A mechanism for nose detection and tracking was needed. For nose detection, we used the *Viola-Jones* algorithm [10] and for nose tracking, we used *Lukas Kanade's* method for optical flow estimation [38]. The *Viola-Jones* algorithm is a face detection algorithm. It outputs the co-ordinates of a rectangle surrounding the user's face in the image frame. The center of that rectangle is the position of the nose.

Once the position of the nose is found, it needs to be tracked. For this reason, *Lukas Kanade's* method (for optical flow estimation) is used. This method takes in 3 parameters which are the last image frame, the image frame that was taken in immediately before the last image frame and the last

TABLE 3. Description of variables.

Variable	Purpose
frameCount	To store the total number of frames processed including the present image frame.
presentMatrix	To represent Grayscale image of the present image frame which is the last image frame received by the Smartphone's camera stored as a matrix.
prevMatrix	To represent Grayscale image of the previous image frame which is the image frame just before the present image frame.
presentNosePoint	To store the coordinates of the nose detected in the *presentMatrix*.
prevNosePoint	To store the coordinates of the nose detected in the *prevMatrix*.
matrixCentrePoint	To store the center point of *presentMatrix*.
score	To control the timer.
timerRunning	To store Boolean values. True if the track of time is being kept otherwise false.
timePassed	To keep track of the amount of time which has passed while the user is showing teeth.
inBetweentime	Time between previous image frame and present image frame.
thresholdTime	It is the minimum value which is to be attained by the *timePassed* variable in order to set the *actionFlag*.
actionFlag	Flag is set to true to indicate that the system is ready to execute a tap or some other actions as per the cursor position.

known position of the nose. It outputs the present position of the nose based on these three parameters.

The last two positions of the nose in the last and second last image frames respectively are used to calculate how much the nose has moved. Based on the displacement and direction of the nose movement, the cursor is moved.

Lucas Kanade's method for optical flow estimation may sometimes lose track of the nose position. So after every 10 frames, we apply the *Viola-Jones* algorithm to re-detect the nose position. Then it is again tracked by the *Lukas Kanade's optical flow* algorithm.

The process of cursor movement using nose movement has been showed in Algorithm 1. The variables used in Algorithm 1 are briefly presented in Table 3.

As presented in Algorithm 1, the image frames are continuously received using the smartphone's camera. Each image frame is stored as a grayscale matrix representation by a variable named *presentMatrix*. The total number of input image frames taken in so far using camera is stored in a variable named *frameCount*. The value of the *frameCount* variable is initially set to 0. It is again set to 0 when it has reached the value of 100. Whenever the value of *frameCount* is divisible by 10, the *Viola-Jones* algorithm is executed to form a rectangle around the face of the user in the *presentMatrix* variable. The top left and bottom right co-ordinate of the face rectangle is stored in *faceRect* variable. The centre of this rectangle gives the co-ordinates of the nose. This co-ordinate of the nose is stored in the variable *presentNosePoint*. The center of the *presentMatrix* is stored in the variable *matrixCentrePoint*.

Algorithm 1 Cursor Movement by Nose Tracking

input : Image frame of face is taken using the smartphone's camera
output: Movement of the cursor as per the nose position

```
frameCount=−1
while true do
    Continuously take image frame of face from the selfie camera
    Store input image as greyscale matrix in presentMatrix
    frameCount=(frameCount+1)%101
    if frameCount%10==0 then
        faceRect= ViolaJonesFaceDetection(presentMatrix)
        presentNosePoint=centre of the faceRect
    else
        presentNosePoint =LukasKanadeOpticalFlow(presentMatrix, prevMatrix, prevNosePoint)
    end
    prevNosePoint=presentNosePoint
    Calculate difference between presentNosePoint and prevNosePoint
    Move cursor based on the difference
    teethMatrix=submatrix of presentMatrix for teeth detection
    TeethDetectionModule(teethMatrix)
    prevMatrix=presentMatrix
end
```

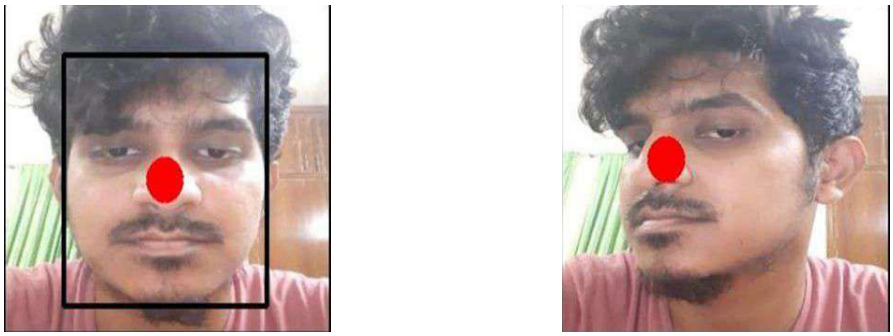
The nose is thus detected using *Viola-Jones* algorithm as showed in Fig. 2(a)

Before taking the next image frame, this nose co-ordinate is stored in *prevNosePoint* and the matrix in *presentMatrix* is stored in *prevMatrix*.

When the *frameCount* variable is not divisible by 10, the *Lukas Kanade's* method is executed for optical flow estimation. This method takes *prevMatrix, presentMatrix* and *prevNosePoint* as parameters. It tracks the co-ordinates of the nose and stores it in the *presentNosePoint* variable. Nose tracking using *Lukas Kanade's* method is showed in Fig. 2(b).

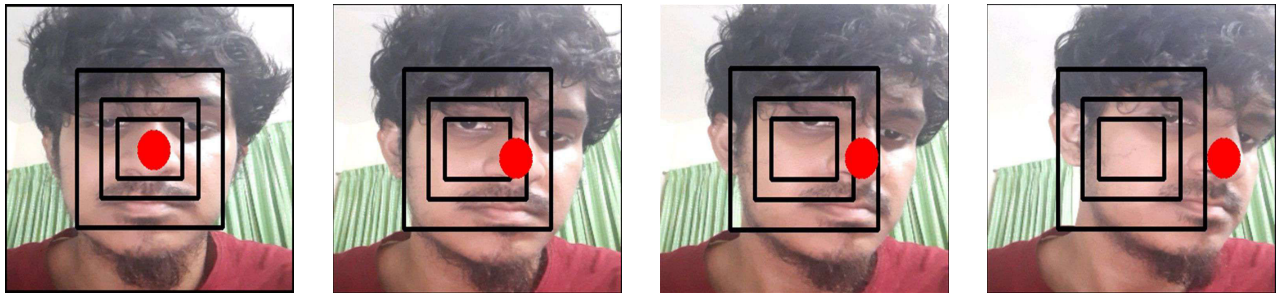
After the use of *Lukas Kanade's* method and before taking the next image frame, this nose co-ordinate is stored in *prevNosePoint* and the *presentMatrix* is stored in *prevMatrix*. The center of the *presentMatrix* is stored in the variable *matrixCentrePoint*.

For cursor movement and cursor speed control, 3 regions have been considered around the center *matrixCentrePoint* of the *presentMatrix* as shown in Fig. 3. In the first region (rectangle) at the center, if the *presentNosePoint* remains located as shown in Fig. 3(a) then the cursor will be stationary. If the *presentNosePoint* goes into the second region as shown in Fig. 3(b) the cursor will move at a very slow speed. The displacement of the cursor will be in the direction of



(a) (b)

FIGURE 2. (a) Face and nose detection using *Viola-Jones* algorithm, (b) Nose tracking using *Lucas-Kanade* optical flow algorithm.



(a) (b) (c) (d)

FIGURE 3. (a) Nose point is within the first rectangle and cursor will not move, (b) nose point is outside the first rectangle and cursor will move at a slow speed in the direction of the nose movement, (c) nose point is outside the second rectangle and cursor will move at a medium speed in the direction of the nose movement, (d) nose point is outside the third rectangle and cursor will move at a fast speed in the direction of the nose movement.

displacement of the *presentNosePoint* from the *matrixCentrePoint*. The cursor will move at medium and fast speed if the *presentNosePoint* is located in the third and fourth region respectively as shown in Fig. 3(c) and 3(d). In both of these cases, the displacement of the cursor will be in the direction of displacement of the *presentNosePoint* from the *matrixCentrePoint*. Using the *matrixCentrePoint*, a sub-matrix of *presentMatrix* named *teethMatrix*, below the nose is created and is passed to the *TeethDetectionModule*.The *TeethDetectionModule* has been discussed elaborately in the next subsection Teeth Detection and Gesture Operation Execution.

B. TEETH DETECTION AND GESTURE OPERATION EXECUTION

The working process of the teeth detection module has been shown in Algorithm 2. In Algorithm 1, *teethMatrix* is given as input to the *TeethDetectionModule*. For the detection of the teeth, Haar cascade classifier [39] of teeth is used. In every input image frame, the Haar cascade classifier of teeth is run by this module to detect the user's teeth.

A timer is used to keep track of how long the user is showing teeth. If the classifier detects the user's teeth, a variable *score* is set to 50. The range of the *score* variable is between 0 to 50. The *score* variable controls whether the timer is running or not. If the *score* variable has a positive value, then the timer is allowed to run. If the *score* variable has the value 0, then the timer is immediately stopped. Whenever in an input image frame, teeth are detected, the *score* variable is immediately set to 50. If no teeth is detected, the *score* variable is reduced by 1. If continuously no teeth is detected, the *score* variable is decreased until it reaches 0. After it reaches 0, the *score* variable is not further decreased.

When the *score* variable has a positive value, the timer keeps on increasing its value. After crossing a definite amount of time, specified by the variable *thresholdTime*, a Boolean variable *actionFlag* is set to true. When the user stops showing teeth, the *score* variable settles to zero and stops the timer. At that time, if the *actionFlag* variable is set to true, a gesture operation is executed as per the cursor position. Fig. 4 shows the successful detection of teeth of the user.

The Android operating system provides special types of touch screen events such as pinch, double tap, scrolls, long presses and flinch which are known as gestures [40]. These gestures have been implemented programmatically in the proposed system so that the user does not require to use hands. The following classes were used in programming the gestures:

(a) *Point* - The Point class stores the x and y co-ordinates of a location on the screen.

Algorithm 2 Teeth Detection and Operation Execution

input : Image frame of teeth taken as *teethMatrix*
output: Execution of an operation

```
Store Haar cascade classifier of teeth
Classifier uses teethMatrix to classify image frame for teeth detection
if teeth is detected then
    teethDetected=true
else
    teethDetected=false
end
if teethDetected==true then
    score=50
else
    score=score-1
    if score<0 then
        score=0
    end
end
if score>0 then
    if timerRunning==true then
        timePassed = timePassed + inBetweentime // add the time passed after the last image frame
    else
        timerRunning==true // timer is started to run
    end
    if timePassed ≥ thresholdTime then
        actionFlag=0
    end
else
    if actionFlag=true then
        Execute relevant gesture or action
        timerRunning=false // timer is stopped
        actionFlag=false
        timePassed=0
    end
end
```

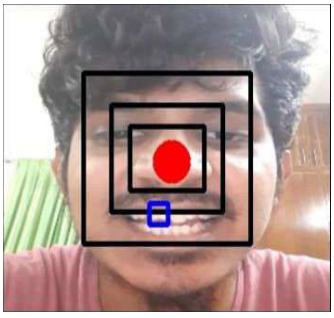


FIGURE 4. Teeth detection (shown with blue rectangle).

(b) *Path* - This class stores the full path or partial path of a gesture using one or more Point objects.

(c) *PathTime* - This class is used to control the start time and duration of a Path object.

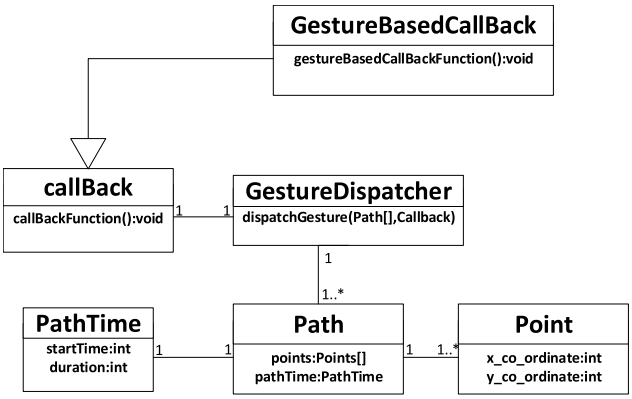


FIGURE 5. Class diagram for Android gesture operation.

(d) *Callback* - This class is used to instantiate a callback object. It will execute a callback function after a gesture is successfully executed. It can be extended to define different child classes based on different types of gestures.

(e) *GestureDispatcher* - An object of this class takes one or more Path objects and a Callback object as a parameter for the function dedicated to dispatch the gesture.

Figure 5 contains the class diagram of the gesture system.

Every gesture contains one or more *Path* object depicting the path of the gesture. For example, when we tap on the smartphone screen, at a particular point, there is only one path which is the point of tapping. The start point and end point of that path is the same i.e. the point of tapping. In case of a swipe, a normal user with functional hands uses a finger and drags it on the screen from one point to another. Let us suppose that the points are named point A and point B. Here the *Path* object depicts the path of swipe whose start point is point A and end point is point B. In case of a double tap, there are two *Path* objects, each of them depicts the point of double tapping and hence the gesture, double tap is a two path gesture.

The time of these gestures are controlled by *PathTime* object, which has two attributes related to the *Path* object. The attributes are *startTime* and *duration*. The *startTime* parameter specifies how long the program should wait to execute a gesture after it has been dispatched. The *duration* parameter specifies the amount of time during which the gesture is executed. For example, in Table 4, the *startTime* for the tap gesture is 0 second. It means that during the execution of the line of code, which gives the command to dispatch the tap gesture, the device will wait no longer and execute a tap gesture immediately. The *duration* for tap, in Table 4 is shown to be 150ms. It means that the tap gesture will be active at the point of tap for 150ms. This event is analogous to the case where a normal user uses his/her finger to tap on a point on the smartphone screen and keeps the finger on the screen for 150ms.

Table 4 contains the description of various parameters used for building a gesture. In order to execute any gesture other than tap, the user has to move the cursor to a region at the

TABLE 4. Programmatic implementation of different gesture operations.

Gesture operations	Number of path objects	Path object number	start Time (ms)	duration (ms)
Tap	1	Path Object 1	0	150
Double Tap	2	Path Object 1	0	10
		Path Object 2	100	150
Long Press	1	Path Object 1	0	650
Drag and Drop	2	Path Object 1	0	1000
		Path Object 2	2000	1000
Sharp Drag	1	Path Object 1	0	1000
Swipe	1	Path Object 1	0	1000
Scroll	1	Path Object 1	0	1000

corner of the smartphone screen. Particular corner region is set for particular gestures. After moving the cursor to a particular region for a particular gesture, the user has to show teeth to enable the cursor to execute the gesture. After the cursor becomes enabled, the user can use it to execute that gesture at the point(s) he/she desires. This section's discussion remains limited to the programmatic execution of gesture operations only; thus the practical execution of gesture operations by the user has been further discussed in the following section.

IV. DEMONSTRATION

Gesture operations are finger movements. It is performed to interact with the touchscreen of the smartphone. A smartphone contains the scope of several different types of gesture operations, for example tap, swipe, drag, double tap, long press, etc. Besides this, there are some other interactions like setting up the volume, seeing the notification, navigating to the home screen, navigating to the previous screen, viewing the recently used applications, etc. These are called global operations. Depending on the user's Android device, there might be physical or software buttons for these interactions. The proposed system implemented these operations to perform all types of interactions using nose and teeth. In this section, we demonstrate the use of the system, how different operations are performed and measure the execution time of those operations.

A. DEMONSTRATION OF DIFFERENT OPERATIONS

To perform a particular gesture operation, at first the cursor needs to be positioned by the users using their nose at a certain location (reserved for a specific operation) on the smartphone screen (see Fig. 6). When the cursor is positioned properly, the color and icon of the cursor will change according to the gesture operation. Then the operation can be executed by showing teeth. How the cursor is controlled using nose point movement is discussed in DEVELOPMENT OF THE SYSTEM section and showed in Fig. 3. The Table 5 shows where to place the cursor to change the cursor icon to activate the specific gesture operation and also show the icons for each operation. The icons are activated by showing teeth

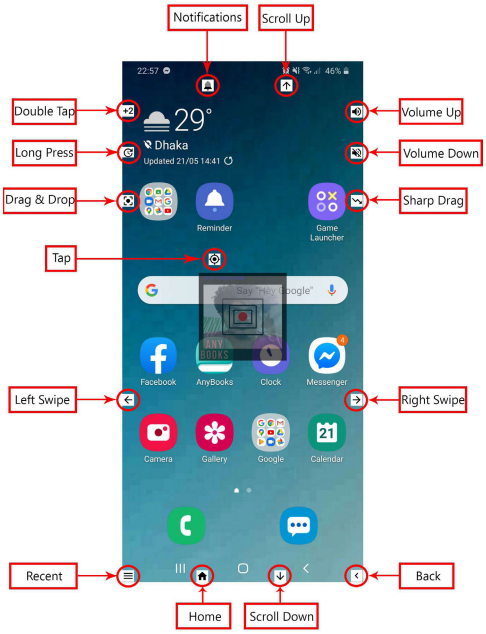
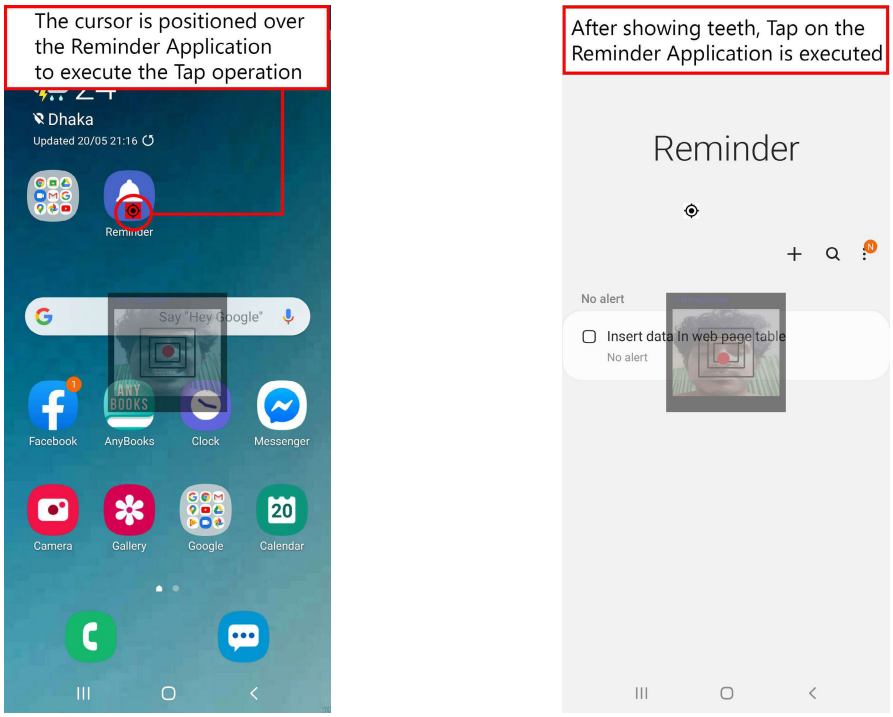


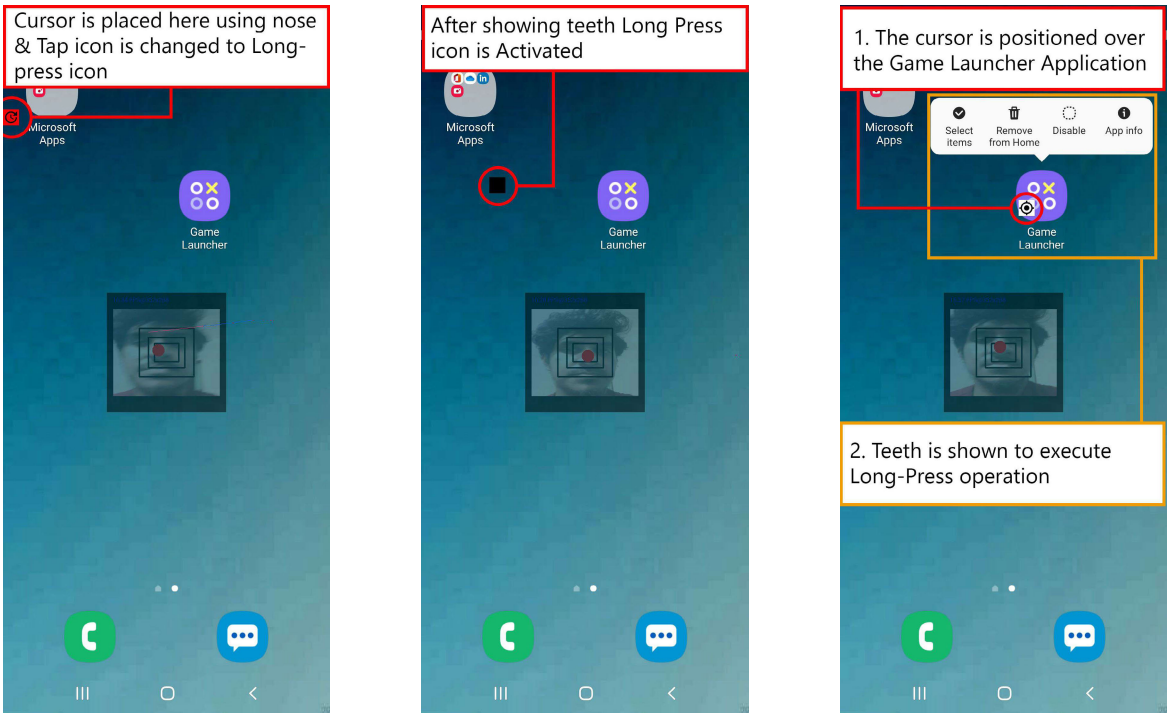
FIGURE 6. Cursor positions for all the gesture and global operations.

after placing the cursor in the designated place except the tap icon, since the normal cursor icon is used as the tap icon in the proposed system. After placing the cursor on the pre-specified position, the cursor changes to the specific icon for the specific operation. These positions for different operations are shown in Fig. 6. For example, to perform a tap operation, the cursor needs to be placed on the desired screen position and teeth should be shown until the cursor icon turns red. The face should be around 25cm away from the mobile when showing teeth. After the users stop showing teeth, the tap operation is performed. A tap operation for Reminder application is shown in Fig. 7. Cursor is placed on the application using nose in Fig. 7(a) and tap operation is executed in Fig. 7(b). Double tap and long press icons are required to be activated by showing teeth after placing the cursor in the designated (pre-specified) position as presented in Table 5. Then, the user needs to place the cursor on the desired screen position to execute the gesture operation by showing teeth. An example case of long press gesture operation is shown in Fig. 8. Fig. 8(a) shows if the cursor is placed near the top left position of the mobile screen, the tap icon is changed to long press icon and in Fig. 8(b) long press icon is activated by showing teeth. Fig. 8(c) shows that the cursor is placed on the Game Launcher application and long press operation is executed after showing teeth. After taking the cursor to the extreme left or right of the screen, a swipe to the left or a swipe to the right respectively is performed. Scrolling up or down is similar to the swipe operation. To perform drag and drop, after activating the drag and drop icon the cursor is placed on the desired content and teeth needs to be shown. Then by moving the cursor to the desired place which is the location where the desired content is to be dropped, the content is moved to that place by showing teeth.



(a) (b)

FIGURE 7. **(a) The cursor is placed to execute tap operation, (b) Tap operation is executed.**



(a) (b) (c)

FIGURE 8. **(a)The cursor is placed to see long press icon, (b) Long press icon is activated, (c) Long press operation is executed.**

Some particular icons for some particular global operations are activated by showing teeth after placing the cursor in the designated position which is shown in Table 6. By showing teeth user can increase or decrease the volume, navigate to the home screen, navigate back to the previous screen, see the notification window, see the recently used apps.

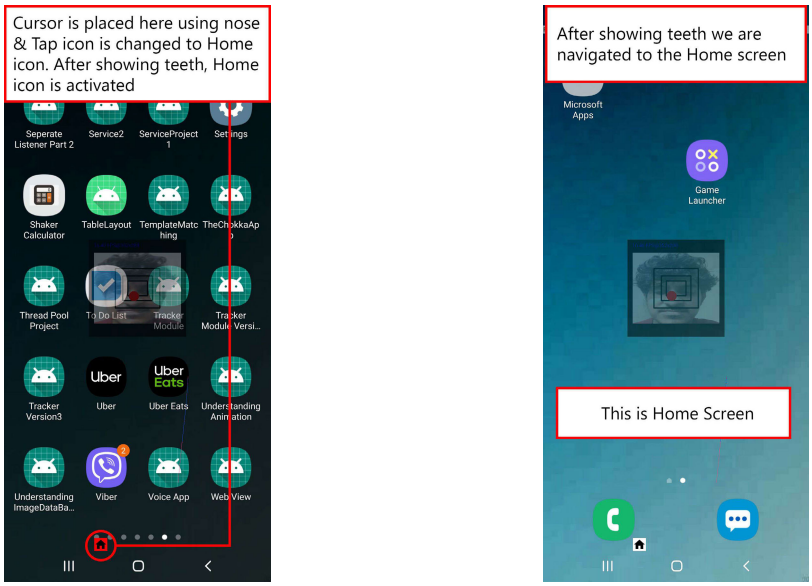


FIGURE 9. (a) Home icon is activated, (b) Navigate to the home screen.

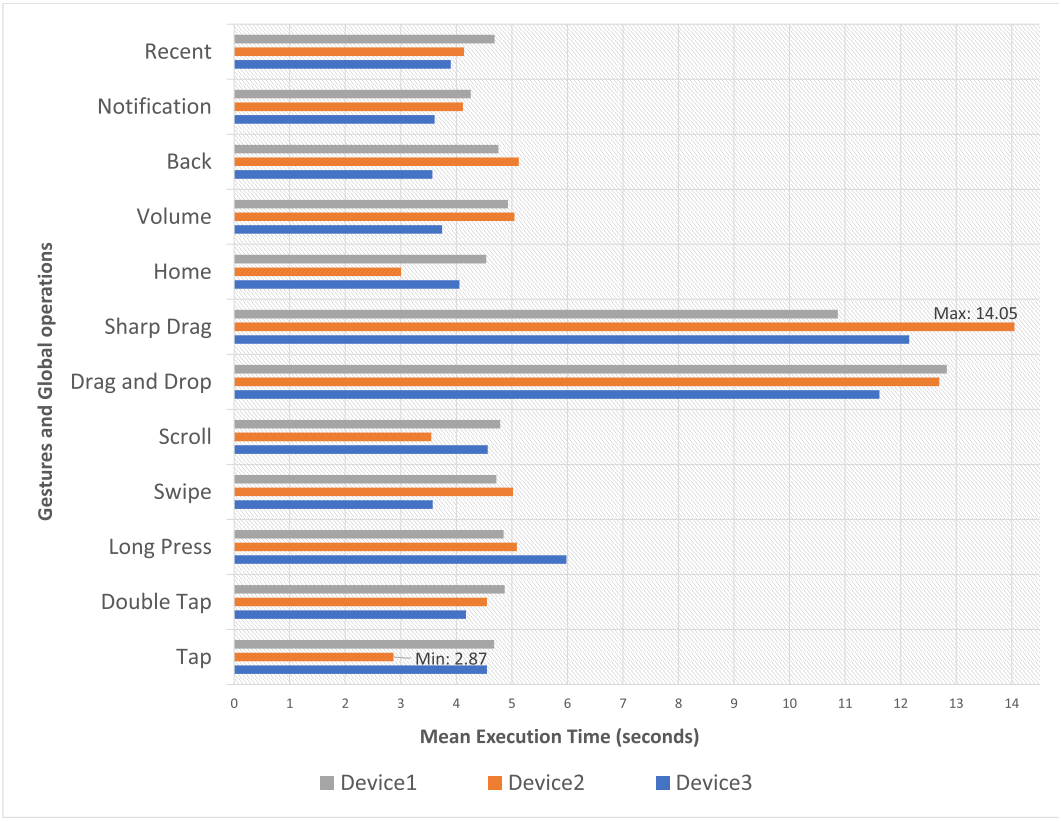


FIGURE 10. Mean execution time of each operation for the three smartphone devices.

For example, in the Fig. 9(a) cursor icon is placed to the designated position using nose as presented in Table 6 to activate home icon and after showing teeth we navigate to home-screen which is shown in Fig. 9(b). A number of screenshots and video demonstrations are available at our website [41].

TABLE 5. Position of the cursor on the screen to activate different gesture operations.

Gesture operations	NMI system icons	Cursor position on the smartphone screen to activate different gesture operation icon (see Fig. 6)
Tap		Placed anywhere on the screen without the side of the screen.
Double Tap	+2	Placed near the top left portion of the screen.
Long Press		Placed near the top left portion of the screen.
Swipe		Placed extreme left or right of the screen.
Scroll		Placed extreme up or down of the screen.
Drag and Drop		Placed near the top left portion of the screen.
Sharp Drag		Placed near the top right portion of the screen.

TABLE 6. Position of the cursor on the screen to activate different global operations.

Global operations	NMI system icons	Cursor position on the smartphone screen to activate different global operation icon (see Fig. 6)
Home		Placed at the down left of the screen.
Volume		Placed at the top right of the screen.
Back		Placed at the down right of the screen.
Notification		Placed at the top of the screen.
Recent		Placed at the bottom left corner of the screen.

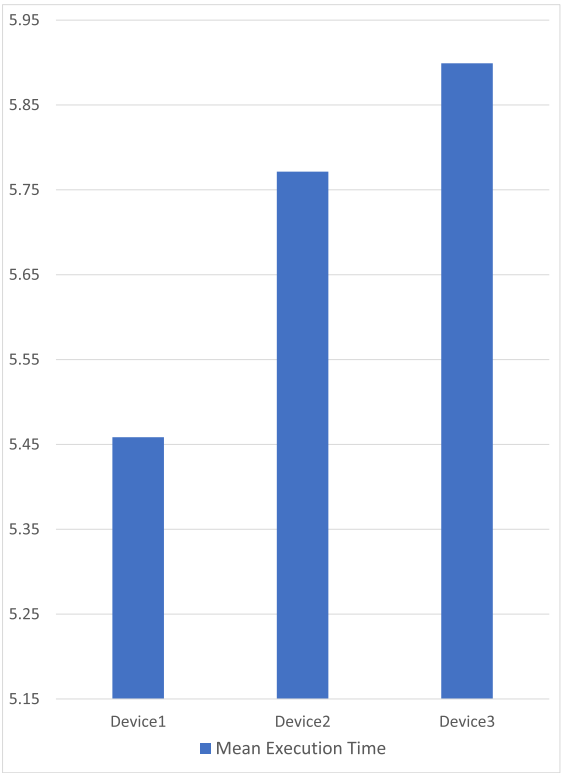


FIGURE 11. A comparison of the mean execution time of all operations on each of the three devices.

B. EVALUATION OF THE SYSTEM

1) INITIAL SETUP

The proposed system is installed in three smartphone devices. *Device1* is Samsung Galaxy A30. It runs on Android v9.0 (Pie) operating system with 6.4 inch display, 1080 × 2340 pixels resolution and 16 megapixel camera, Samsung 7 octa 7904 chipset and the phone is powered by octa-core (2 × 1.8 GHz Cortex-A73 & 6 × 1.6 GHz Cortex-A53) processor. *Device2* is Xiaomi Redmi 7. It runs on Android v9.0(Pie) operating system with 6.26 inch display, 720 × 1520 pixels resolution, 12 megapixel camera, Qualcomm SDM632 Snapdragon 632 chipset and the phone is powered by octa-core (4 × 1.8 GHz Kryo 250 Gold & 4 × 1.8 GHz Kryo 250 Silver) processor. *Device3* is TECNO CA7. It runs on Android v8.1.0 operating system with 6 inch display, 2 GHz octa-core processor, screen resolution 1440 × 720, 20 megapixel camera and MediaTekHelio P23 chipset. The proposed system works in the background as a service. A companion of the disabled user needs to help him turn on the background process. After turning on, the system will start running the background process and a cursor will appear. The color of the cursor is white when it is stationary. During cursor movement, the color of the cursor will become yellow.

2) PARTICIPANTS PROFILE AND STUDY PROCEDURE

A total of three participants were invited to conduct the test. All of them were male and had an average age of 60. They all were able bodied elderly person and did not participate in the development process. Although each of them was familiar with using a smartphone they did not have experience with interacting with the smartphone using gestures other than touch. To explore the effectiveness of device (smartphone) type on the execution time for each gesture and global operation, a total of 21 trials were conducted. 7 trials for each operation were performed in each of the 3 devices by each participant.

3) PERFORMANCE EVALUATION

The resultant data are synthesized and presented in Table 7. Success frequency of 7 trials of each device and mean execution time with standard deviation of each operation on each device are shown in Table 7. A one-way ANOVA analysis was conducted to compare the effect of device type on the execution time of different operations. The results of ANOVA

TABLE 7. Frequency of success, execution time and result of ANOVA analysis of each operation in three devices.

Type of Operation	Operations	Device1		Device2		Device3		Execution Time (m±SD)	ANOVA Analysis		
		Freq of Success(%) (n=7)	Execution Time (m±SD)	Freq of Success(%) (n=7)	Execution Time (m±SD)	Freq of Success(%) (n=7)	Execution Time (m±SD)		f-ratio value	p-value	result
Gesture	Tap	100%	4.68 ± 0.16	100%	2.87 ± 0.26	100%	4.55 ± 1.34	4.03 ± 1.01	11.26	0.00067	Significant
	Double tap	100%	4.87 ± 0.07	100%	4.55 ± 0.33	100%	4.174 ± 0.37	4.53 ± 0.35	10.44	0.00097	Significant
	Long press	100%	4.85 ± 0.07	100%	5.09 ± 1.58	100%	5.98 ± 0.52	5.31 ± 0.59	2.71	0.093566	Not significant
	Swipe	100%	4.72 ± 0.17	100%	5.02 ± 1.23	100%	3.57 ± 0.55	4.44 ± 0.76	6.65	0.00688	Significant
	Scroll	100%	4.79 ± 0.14	100%	3.55 ± 0.90	100%	4.56 ± 0.74	4.30 ± 0.66	6.23	0.00877	Significant
	Drag and drop	100%	12.83 ± 0.07	100%	12.69 ± 1.06	100%	11.62 ± 1.31	12.38 ± 0.66	3.02	0.074035	Not significant
	Sharp drag	100%	10.87 ± 0.08	100%	14.05 ± 0.58	100%	12.15 ± 0.66	12.36 ± 1.60	68.9	0.00001	Significant
Global	Home	100%	4.54 ± 0.03	100%	3.00 ± 0.33	100%	4.06 ± 1.06	3.87 ± 0.77	10.59	0.000912	Significant
	Volume	100%	4.93 ± 0.03	100%	5.05 ± 0.55	100%	3.75 ± 0.81	4.57 ±0.72	11.36	0.000645	Significant
	Back	100%	4.76 ± 0.21	100%	5.13 ± 2.28	100%	3.57 ± 0.97	4.49 ±0.81	2.25	0.133997	Not significant
	Show Notification	100%	4.26 ± 0.09	100%	4.12 ± 0.77	100%	3.61 ± 0.79	3.99 ± 0.34	2.01	0.162331	Not significant
	Recent	100%	4.69 ± 0.1	100%	4.14 ± 0.53	100%	3.90 ± 0.85	4.24 ±0.41	3.44	0.054226	Not significant

test is also presented in Table 7. The results indicated that all the operations are executable in different types of smartphone. The proposed system provides a significant detection accuracy of face and teeth from a distance of about 25 cm and for that we get success to execute each operation. Again, the execution time of each operation varies from device to device because of the overall configuration of the selected devices which is shown in Fig. 10.

For example, to perform tap operation, on average, *device1* required 4.68s, *device2* required 2.87s and *device3* required 4.55s (see Fig. 10). The averaged values of the execution time of the operations for each device are presented in Fig. 11. It shows that *device1* has the lowest mean execution time which is 5.46s and *device3* has the highest mean execution time which is 5.89s. Mean execution time with standard deviation of each operation is shown in Fig. 12. It shows that the execution time varies from 0.34s to maximum 1.60s for any of the selected operation. The results of ANOVA analysis (Table 7) showed that out of 12 operations, there is no significant effect of device type on the execution time for long press, drag and drop, navigate to back screen, notification and recent operations and for other 7 operations there is a significant effect of device type on the execution time. For example, the result shows for tap operation, there is a significant effect of device type on the execution time at the $p < .05$ level for the three conditions [$F(2, 18) = 11.26$, $p = 0.00067$] and for long press operation there is no significant effect of device type on the execution time at the $p < .05$ level for the three conditions [$F(2, 18) = 2.71$, $p = 0.093566$].

V. DISCUSSION

A nose-teeth based mobile interaction system employing *Viola-Jones* and *Lucas Kanade optical flow* algorithm has been proposed and implemented in this research to control almost all the gesture and global operations of a smartphone using nose and teeth. Most of the existing systems focus on the interactions with desktop computer using different gestures [1], [2], [4], [5], [7], [8], [13], [21], BCI [28], [30], or person's breath [29]. There are a very few studies focusing on smartphone interaction [6], [23]. In [6], eye blink detection based on the eyelids state is used for only controlling the phone call functionality. Again, in [20], [23] they do not focus on the disabled people and consider their limitations to use the devices with their hands. Various alternate means of interactions are proposed to interact with different kinds of devices for example facial interaction or interaction through the feet [25], but in this case nose computer interaction performs better than other eye interaction [36]. Therefore,

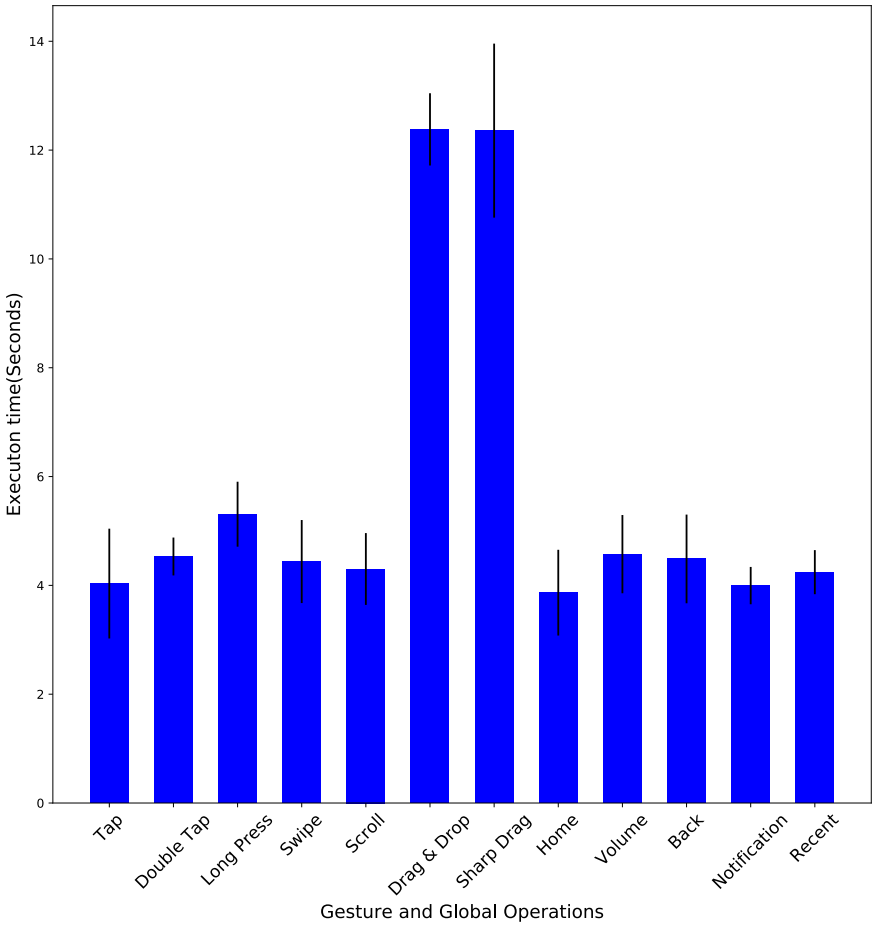


FIGURE 12. Mean execution time and standard deviation of execution time of each operation.

considering all this, we developed the nose-teeth mobile interaction system which will stretch a helping hand to the people who are incapable of using the android smartphones due to their disabilities. Some other hardware based systems are proposed for interacting with other devices like robot, 3DTV in VR environment and these are based on additional hardware devices and sensors [30]–[33], [42], which are costly and maybe difficult to afford for some people. Any new efficient solution based on an expensive device will not be able to effectively help the disabled people. The proposed NMI system requires no additional hardware devices and only uses the front camera of the smartphone. As a low-cost alternative, the proposed system can be replaced with the expensive systems for accessing smartphones.

In [1], the mean execution time for a click operation is 4.12s, whereas our proposed system has a mean execution time of 4.03s for the click operation. There is an application EVA [9] which is developed for smartphone interaction. To perform gesture actions using EVA is time consuming (slow) as they require the aid of a UI menu. Clicking function occurs automatically if the cursor is kept stationary for some time which can result in accidental clicks. Again, some important gestures are not implemented here, for example, drag and drop. While in our system, we have implemented all of the operations of the smartphone and the user can activate any gesture based operation by placing the cursor on the designated position of the screen, so there is no need of a UI menu. The user can execute all the operations by showing teeth, so there is no possibility of accidental click. In [27], the proposed EPOC+ based system was evaluated by analyzing the task completion times for both the current and their proposed system, while in this research, the proposed NMI system was installed in three smartphone devices to perform a trial experiment and a one-way ANOVA analysis was also conducted to compare the effect of device type on the execution time of different operations.

VI. CONCLUSION

In this research, various alternate means of interaction has been explored. It is seen that there are no prior studies that explicitly focus on hands-free interaction with the smartphone. In this work, we have developed a new hands-free interaction system based on nose and teeth for controlling the smartphone. Evaluation of the system shows that it is very simple and efficient to use, which provides an easy way for disabled people to use their smartphones. Thus, the con-

tributions of this paper are as follows: Firstly, the proposed NMI system is new, innovative and no other previous research has developed such system to execute all the gesture and global operations of a smartphone through alternative means like nose and teeth. Thus, the people who cannot move their hands or fingers properly or those who don't have any hands can use this system to interact with their smartphones. Secondly, the algorithms used in the proposed system are light weight and hence ensure that the system does not generate much overhead in computation (CPU usage). Finally the proposed system does not depend much on the device type and performs well in different smartphone devices with highest possible success rate (100%). The execution time for some operations may slightly vary due to the overall configuration of the smartphone.

The research has a few limitations as well. Firstly, due to the Coronavirus pandemic, we were unable to evaluate the system performance with disabled users. Secondly, the system has been developed for android based smartphones only. Finally, we did not consider the impact of camera on the phone battery lifetime which may cause continuous battery drain. In the future, the proposed system will be evaluated by comparing test data of normal and disabled users. We will also develop the proposed system so that it can run in other platforms. We will evaluate the impact of camera for continuous battery drain. In the future, we can use Convolutional Neural Networks to train classifiers more efficiently and detect nose and teeth more accurately. We believe that, this system will make it easier for the disabled people to handle Android based smartphones and enjoy its facilities.

REFERENCES

[1] S. S. Khan, M. S. H. Sunny, M. S. Hossain, E. Hossain, and M. Ahmad, "Nose tracking cursor control for the people with disabilities: An improved HCI," in *Proc. 3rd Int. Conf. Electr. Inf. Commun. Technol. (EICT)*, Dec. 2017, pp. 1–5.

[2] R. Lal and S. Chiddarwar, "Real time human computer interaction using facial gestures," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–6.

[3] S. O'Dea. (Feb. 28, 2020). *Smartphone Users Worldwide 2016-2021*. Accessed: Jun. 22, 2020. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[4] S. Dongre and S. Patil, "A face as a mouse for disabled person," *Int. J. Comput. Sci. Mobile Comput.*, vol. 4, no. 4, pp. 156–160, 2015.

[5] P. Gyawal, A. Alsadoon, P. W. C. Prasad, L. S. Hoe, and A. Elchouemi, "A novel robust camera mouse for disabled people (RCMDP)," in *Proc. 7th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2016, pp. 217–220.

[6] A. Prof. and M. Student, "Efficient eye blink detection method for disabled-helping domain," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 5, p. P2, 2014.

[7] S. K. Chathuranga, K. C. Samarawickrama, H. M. L. Chandima, K. G. T. D. Chathuranga, and A. M. H. S. Abeykoon, "Hands free interface for human computer interaction," in *Proc. 5th Int. Conf. Inf. Autom. Sustainability*, Dec. 2010, pp. 359–364.

[8] M. Mangaiyarkarasi and A. Geetha, "Cursor control system using facial expressions for human-computer interaction," *Int. J. Emerg. Technol. Comput. Sci. Electron.*, vol. 8, pp. 30–34, Oct. 2014.

[9] CREA. (2020) *EVA Facial Mouse-Apps on Google Play*. Accessed: Jun. 22, 2020. [Online]. Available: https://bit.ly/3hSlzje

[10] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[11] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Hoboken, NJ, USA: Wiley, 2009.

[12] J. Cech and T. Soukupova, "Real-time eye blink detection using facial landmarks," in *Proc. 21st Comput. Vis. Winter Workshop*, Feb. 2016, pp. 1–8.

[13] G. C. D. Silva, M. J. Lyons, S. Kawato, and N. Tetsutani, "Human factors evaluation of a vision-based facial gesture interface," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, vol. 5, Jun. 2003, p. 52.

[14] P. Michel and R. El Kaliouby, "Real time facial expression recognition in video using support vector machines," in *Proc. 5th Int. Conf. Multimodal Interface (ICMI)*, 2003, pp. 258–264.

[15] Y. Wu and X. Ai, "Face detection in color images using AdaBoost algorithm based on skin color information," in *Proc. 1st Int. Workshop Knowl. Discovery Data Mining (WKDD)*, Jan. 2008, pp. 339–342.

[16] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corp.*, vol. 5, nos. 1–10, p. 4, 2001.

[17] M. Nabati and A. Behrad, "Camera mouse implementation using 3D head pose estimation by monocular video camera and 2D to 3D point and line correspondences," in *Proc. 5th Int. Symp. Telecommun.*, Dec. 2010, pp. 825–830.

[18] F. Loewenich and F. Maire, "Hands-free mouse-pointer manipulation using motion-tracking and speech recognition," in *Proc. Conf. Comput.-Hum. Interact. Special Interest Group (CHISIG)*, 2007, pp. 295–302.

[19] Y. Shi, Z. Zhang, K. Huang, W. Ma, and S. Tu, "Human-computer interaction based on face feature localization," *J. Vis. Commun. Image Represent.*, vol. 70, Jul. 2020, Art. no. 102740.

[20] S. S. Rautaray and A. Agrawal, "A novel human computer interface based on hand gesture recognition using computer vision techniques," in *Proc. 1st Int. Conf. Intell. Interact. Technol. Multimedia (IITM)*, 2010, pp. 292–296.

[21] C. Esiyok, A. Askin, A. Tosun, and S. Albayrak, "Novel hands-free interaction techniques based on the software switch approach for computer access with head movements," *Universal Access Inf. Soc.*, vol. 1, pp. 1–15, Jul. 2020.

[22] R. H. Abiyev and M. Arslan, "Head mouse control system for people with disabilities," *Expert Syst.*, vol. 37, no. 1, Feb. 2020, Art. no. e12398.

[23] H. Elleuch, A. Wali, A. Samet, and A. M. Alimi, "A static hand gesture recognition system for real time mobile device monitoring," in *Proc. 15th Int. Conf. Intell. Syst. Design Appl. (ISDA)*, Dec. 2015, pp. 195–200.

[24] V. Oliveira and A. Conci, "Skin detection using HSV color space," in *Proc. Workshops Sibgrapi*, Jan. 2009, pp. 1–2.

[25] T. Kim, J. R. Blum, P. Alirezaee, A. G. Arnold, P. E. Fortin, and J. R. Cooperstock, *Usability of Foot-Based Interaction Techniques for Mobile Solutions*. Cham, Switzerland: Springer, 2019, pp. 309–329, doi: 10.1007/978-3-319-93491-4_16.

[26] K. Sancheti, S. K. K, S. A, and S. P, "Hands-free cursor control using intuitive head movements and cheek muscle twitches," in *Proc. IEEE Region 10 Conf.*, Oct. 2018, pp. 356–361.

[27] B. Šumak, M. Špindler, M. Debeljak, M. Heriäko, and M. Puänik, "An empirical evaluation of a hands-free computer interaction for users with motor disabilities," *J. Biomed. Informat.*, vol. 96, Aug. 2019, Art. no. 103249.

[28] R. Abiri, S. Borhani, J. Kilmarx, C. Esterwood, Y. Jiang, and X. Zhao, "A usability study of low-cost wireless brain-computer interface for cursor control using online linear model," *IEEE Trans. Human-Mach. Syst.*, vol. 50, no. 4, pp. 287–297, Aug. 2020.

[29] C. Esiyok, A. Askin, A. Tosun, and S. Albayrak, "Software switches: Novel hands-free interaction techniques for quadriplegics based on respiration–machine interaction," *Universal Access Inf. Soc.*, vol. 19, no. 2, pp. 347–359, Jun. 2020.

[30] S. Ahmed, V. Popov, A. Topalov, and N. Shakev, "Hand gesture based concept of Human–Mobile robot interaction with leap motion sensor," *IFAC-PapersOnLine*, vol. 52, no. 25, pp. 321–326, 2019.

[31] H. Si-Mohammed, J. Petit, C. Jeunet, F. Argelaguet, F. Spindler, A. Evain, N. Roussel, G. Casiez, and A. Lecuyer, "Towards BCI-based interfaces for augmented reality: Feasibility, design and evaluation," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 3, pp. 1608–1621, Mar. 2020.

[32] Y. S. Pai, T. Dingler, and K. Kunze, "Assessing hands-free interactions for vr using eye gaze and electromyography," *Virtual Reality*, vol. 23, no. 2, pp. 119–131, 2019.

[33] S. Zhang and S. Zhang, "A novel human-3DTV interaction system based on free hand gestures and a touch-based virtual interface," *IEEE Access*, vol. 7, pp. 165961–165973, 2019.

[34] G. A. Devices. *GlassOuse Assistive Device*. Accessed: Jun. 22, 2020. [Online]. Available: https://glassouse.com

[35] Quha. *Quha Zono*. Accessed: Jun. 22, 2020. [Online]. Available: https://www.quha.com/products-2/zono/
[36] R. L. Santos, A. Abrantes, and P. M. Jorge, "Eye gaze tracking system for adapted human-computer interface," *Academic J. Electron. Telecommun. Comput.*, vol. 3, no. 1, p. 1, 2017.
[37] E. Patel and D. Shukla, "Comparison of optical flow algorithms for speed determination of moving objects," *Int. J. Comput. Appl.*, vol. 63, no. 5, pp. 32–37, Feb. 2013.
[38] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, vol. 2, 1981, pp. 674–679.
[39] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Dec. 2001, p. 1.
[40] A. Developers. *Detect Common Gestures*. Accessed: Jun. 23, 2020. [Online]. Available: https://developer.android.com/training/gestures/detector
[41] M. N. Islam, M. S. Aadeeb, R. R. Khan, M. M. H. Munna, M. Sarwar, and S. N. Zarin. *A New Way to Interact With Your Smartphone*. Accessed: Jul. 1, 2020. [Online]. Available: https://nose-mobile-interaction.github.io/
[42] H. Wu, W. Luo, N. Pan, S. Nan, Y. Deng, S. Fu, and L. Yang, "Understanding freehand gestures: A study of freehand gestural interaction for immersive VR shopping applications," *Hum.-Centric Comput. Inf. Sci.*, vol. 9, no. 1, p. 43, Dec. 2019, doi: 10.1186/s13673-019-0204-7.

MUHAMMAD NAZRUL ISLAM received the B.Sc. degree in computer science and information technology from the Islamic University of Technology, Bangladesh, in 2002, the M.Sc. degree in computer engineering from the Politecnico di Milano, Italy, in 2007, and the Ph.D. degree in information systems from Åbo Akademi University, Finland, in 2014. Before joining MIST, he was working as a Visiting Teaching Fellow with Uppsala University, Sweden, and as a Post-Doctoral Research Fellow with Åbo Akademi University. From 2003 to 2012, he was also a Lecturer and an Assistant Professor with the Department of Computer Science and Engineering, Khulna University of Engineering and Technology (KUET), Bangladesh. He is currently an Associate Professor with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST), Dhaka, Bangladesh. He has authored more than 100 peer-reviewed publications in international journals and conferences. His research interests include but not limited to human–computer interaction (HCI), humanitarian technology, health informatics, military information systems, information systems usability, and computer semiotics. He is a member of the Institution of Engineers, Bangladesh (IEB).

MD SHADMAN AADEEB is currently pursuing the B.Sc. degree with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST). His research interests include artificial intelligence, image processing, and human–computer interaction (HCI).

RAFIUR RAHMAN KHAN is currently pursuing the B.Sc. degree with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST). His research interests include artificial intelligence, image processing, and human–computer interaction (HCI).

MD. MAHADI HASSAN MUNNA is currently pursuing the B.Sc. degree with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST). His research interests include artificial intelligence, image processing, and human–computer interaction (HCI).

MAHMUD SARWAR is currently pursuing the B.Sc. degree with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST). His research interests include artificial intelligence, image processing, and human–computer interaction (HCI).

SHAMIMA NASRIN is currently pursuing the B.Sc. degree with the Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST). Her research interests include network security and human–computer interaction (HCI).

A. K. M. NAJMUL ISLAM received the M.Sc. (Eng.) degree from the Tampere University of Technology, Finland, and the Ph.D. degree in information systems from the University of Turku, Finland. He is currently an Associate Professor with LUT University, Finland. He has 90+ publications. His research has been published in top outlets, such as IEEE Access, *European Journal of Information Systems*, *Information Systems Journal*, *Journal of Strategic Information Systems*, *Technological Forecasting and Social Change*, *Computers in Human Behavior*, *Internet Research*, *Computers & Education*, *Journal of Medical Internet Research*, *Information Technology & People*, *Telematics and Informatics*, *Journal of Retailing and Consumer Services*, *Communications of the AIS*, *Journal of Information Systems Education*, *AIS Transactions on Human–Computer Interaction*, and *Behaviour and Information Technology*, among others. His research interest includes human centered computing.