İSTANBUL UNIVERSITY CERRAHPAŞA

RESEARCH ARTICLE

Electrica, 2020; 20(2): 143-152

Touchless Authentication System Using Visual Fingertip Trajectories

Vefak Murat Akman[1], Revna Acar Vural[2], Kemal Talha Koç[2]

[1]Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey
[2]Department of Electronics and Communication, Yildiz Technical University, Istanbul, Turkey

Cite this article as: Akman VM, Acar Vural R, Koç KT. Touchless Authentication System Using Visual Fingertip Trajectories. *Electrica*, 2020; 20(2): 143-152.

ABSTRACT

Biometric traces that are left on the screen of a security device or mobile phone may cause a threat to the privacy of people when such personal data is likely to be stolen. The touchless authentication system is proposed to overcome this problem, which allows the usage of fingertip trajectories for creating or entering personal information such as passwords. First, fingertip regions are detected with the Haar Cascade method, and these regions are tracked using the Lucas–Kanade algorithm. The accurate detection of fingertips with invariance to pose, lighting, and brightness are essential for security. Experimental results show that the proposed system proved its efficiency in various backgrounds where the finger is oriented in various directions. This system is suitable for low cost and real-time processing, having low computational effort.
Keywords: Authentication, fingertip detection, fingertip tracking, machine learning, secure computation

Corresponding Author:
Revna Acar Vural

E-mail:
racar@yildiz.edu.tr

Received: 02.29.2020

Accepted: 06.10.2020

DOI: 10.5152/electrica.2020.20030

Introduction

The validity of authentication with high security is becoming increasingly important as technology develops. The biometric identities, such as fingerprints and iris patterns, are used in home security systems, mobile phones, and institutions where security is crucial. The usage of these identities requires an enormous amount of caution when personal data is likely to be stolen. Face information, fingerprints, and other biometric traces can be accessed and reused, threatening to people's privacy.

Entering a password by touching a screen may not be completely secure since a biometric trace is left on the surface. Therefore, a secure methodology based on visual fingertip trajectories is proposed that allows untouched usage of authentication devices. The user can either create a new password or enter a predefined password to open the lock system. The proposed methodology first detects fingertips via the Haar Cascade classifier, and the tracking algorithm is initiated. There are nine points on it to be tracked by the Lucas-Kanade algorithm. This work's contribution is the efficient detection and tracking of the fingertip movement without leaving any trace on the screen, thus providing a secure authentication with high accuracy.

Most of the state-of-the-art image processing studies, including this work, were based on Rapid Object Detection using a Boosted Cascade of Simple Features [1]. This three-step method started with proposing an integral imaging technique to calculate features of different dimensions. The integral image was calculated by performing a small number of operations, and once calculated, other Haar-like features could be recalculated at any size or in any position.

Following, the AdaBoost algorithm was used to select relevant features while training the classifier. The weak learning feature was limited to a single one. As a result, each stage's booting process, which selected the new weak classifier, could be viewed as a feature selection process [1]. In [2], a system that recognized hand gesture and finger movements for HCI projects were developed using the OpenCV library. After the hand boundaries were determined, the fingertip was detected by the convex hull method. Convex hull is an accurate

method; however, it is problematic in terms of performance and power consumption due to its mathematical process intensity in real-time situations. In [3], a system was developed whereby making specific hand movements, and a motorcycle driver could control the action camera on the helmets. Seven thousand fist photos were collected to train the classifier. It resolved the problem of perception when the hand was in proximity. In [4], CS-AdaBoost was proposed to be more effective than the traditional method of far face detection. The misclassified positive samples' weights were increased, and correctly classified negative samples weights were decreased. The method focused on the hardest positive. Therefore, detection rates of this method were high. Another crucial factor was the AdaBoost algorithm, which selected essential and useful features from the large dataset [5]. Thus, the classifier became more efficient and available for real-time operations. In [6], the approach was presented for image retrieval by using highly selective features. Effective queries were formulated by using just a small number of training samples and a small set of features through the specially selected features. In [7], feature extraction, Haar-like features, and AdaBoost learning algorithm were used together for hand detection. The effects of other skin-colored objects in the frame were not mentioned, which could reduce the system's performance. Different methods were developed for an object to be detected indirectly, as in [8], [9], [10], [11]. In [8], detected fingertip was used as a mouse device in different digital devices. Different horizontal lines were taken on hand, and the midpoint of these lines was connected to the vertical axis to determine the fingertip by looking at the slope of the line. Motion blur was used for tracking a finger in adverse ambient conditions [8]. In [9], convex hull, Curvature of Perimeter, and K-Curvature methods were proposed for hand and finger detection. Among these methods, Curvature of Perimeter was selected as the most efficient one. In [10], the hand gesture was segmented by using area growing algorithm and applied morphological operations. The center and radius of the hand were calculated to detect fingertips by the convex hull method. In [11-13] hand gesture detection and skin color detection were investigated. It was stated that color detection was an easy way to identify an object; however, the performance could be from the ambient, especially when there was no clear illumination. In [14], fingertip identification was performed in a complex environment. First, the hand region was detected from a complex background by using dense optical flow and constructing a skin filter. Then, the centroid was calculated, and the maximum local distance outside circles of the extended circle was found to detect fingertips. However, this algorithm had shortages. If two fingers were adjacent, the fingertips were detected, as one and the whole fist was detected instead of fingertips when a hand was positioned in perpendicular. It is also possible to detect objects with CNN, which provides excellent accuracy, but it interferes with real-time processing. In [15], the bi-level cascade structure of CNN was proposed to solve problems such as illumination variety, shape distortions, complex background, etc. The focus of HCI projects is utilizing biometric identities for device applications. In [16], the user was interacted with the computer by using the fingertip through video projection devices. The webcam was used instead of sensors. Even the cost of these devices was reduced, the performance of the system was decreased due to the usage of the Kalman filter and correlation. Another drawback could be noted as that the background was constant, which required certain conditions to operate. In [17], a remote control system that enabled counting open fingers and motion detection was presented. After the motion in the frame was detected, the related area was zoomed to extract hand gesture, and the number of open fingers was counted to control the television menu.

Another challenge in image processing is object tracking. In [18], a new technique used as a spatial intensity gradient of the images was presented. This method analyzed fewer potential matches between images and could track one or more pixels with an optical flow method that brought a great advantage for that study. In [19], a method proposed to estimate motion or displacement by tracking all motions in a frame instead of the tracking single pixel. In [20], the fingertip detection and tracking were performed by using different algorithms. First, the hand region and fingertips were found, respectively, the same as in [14]. Then, the fingertip points were used for motion estimation with bi-directional optical flow. The fingertip model with a perceptual hash sequence was presented to locate fingertips more efficiently. Instead of tracking all motions, it was possible to track a single target as in [21-25]. MeanShift algorithm is commonly used in object tracking applications because it does not need prior knowledge about the target to be tracked. This algorithm does not work well under some cases example fast-moving target object or obstacle in front of the target object. In [21], improved MeanShift and Kalman filter were combined to resolve these problems. This combined algorithm computed the target direction even there was an obstacle between object and camera. Nevertheless, there was no significant difference between the performance of MeanShift and the proposed algorithm according to the reported results. MeanShift uses only the color information of the target object and does not contain environmental information. Thus, when there exists a similarity between the ambiance color and object color, the tracking was likely to fail.

A method was proposed in [22], which used an adaptive block color histogram to fix occlusion and rotation problems of the MeanShift. In case the size of the bounding rectangle of the tracking object was changed, it was needed to readjust the block method and accordingly weigh them. These studies showed that the MeanShift algorithm is not adaptive to changes in target size and rotation. To solve the problem, Adaptive MeanShift (CamShift) algorithm was proposed. This algorithm was able to track changes in probability distributions within video sequence by adjusting its search window size considering the distribution [23]. Even so, the CamShift algorithm was prone to be affected by excessive illumination and obstacles. These problems were tried to be resolved by using improved

algorithms [24, 25]. In [24], the CamShift search windows were updated using the contour features of the target to decrease the lighting effects. The prediction of tracking object location was carried out using the Kalman filter to overcome obstacles. In the other study [25], the proposed algorithm used the Gaussian mixture model to determine the tracking area accurately. Same as in [24], this study used the Kalman filter to predict the occluded objects. However, both studies [24, 25] have issues about the real-time operation because of the need for high computational effort. In some devices, such as the wireless mouse, the data sent to the computer can be intercepted by modified algorithms. Crucial information, such as passwords, can be obtained by tracking the mouse cursor [26]. In [27], the secure fingertips mouse was proposed to prevent attacks like in [26]. This algorithm was used the mobile phone's back camera to detect fingertips, move a cursor on the screen, and perform click event by click gestures. Also, a randomized mouse acceleration algorithm was designed to prevent the third person from finding clicks on the keyboard by observing the finger movement.

In this work, a touchless authentication system is realized using fingertip trajectories. First, the dataset is trained with Haar-like features and AdaBoost to detect fingertips. Then, fingertip tracking is carried out with a sparse optical flow algorithm. The fingertip movements control the proposed authentication system without any biometrical traces left to the screen while entering or creating the password. Having low computational effort, it is suitable for low- cost real-time processing.

Following the introduction, Section 2 introduces the theory and application of fundamental algorithms for fingertip detection and tracking. The proposed methodology for the touchless authentication system and experimental results of the system are given in Section 3 and Section 4. Finally, the concluding remarks and future work are presented in Section 5.

Fundamental Algorithms for Fingertip Detection and Tracking

This section explains the fundamental algorithms used in the touchless authentication system for fingertip detection and tracking. However, some preprocessing methods are applied before the fingertip detection phase to increase accuracy. These methods are histogram equalization, erosion, and dilation, respectively. The histogram is a graphical representation of the intensity distribution of an image. It counts the number of pixels for each density value considered. Equalization initiates the mapping of the histogram distribution to a wider and more uniform distribution of intensity values. So, these values are spread over the whole range. After the implementation of histogram equalization, the image may still contain noise. Erosion and dilation methods are used to remove noise and correct the shape of the object that will be detected.

Fingertip Detection

The Haar Cascade algorithm classifies images based on the value of simple features (Figure 1) since they can encode temporal domain knowledge, which is difficult to learn using a limited quantity of training data. Moreover, feature-based operations are fast when compared to pixel-based operations [1]. Haar Cascade requires thousands of samples to train the AdaBoost classifier to determine whether the object exists in another image by extracting and marking the essential features. The algorithm uses weak classifiers to perform this process. Weak classifiers cannot define an object by own. However, they can classify objects together with other features. These features sort of filters, as shown in Figure 1. If one of them exists in the object, it is saved as a weak classifier. In the next step, the saved weak classifiers are searched in the new image. If these features are found in the new image, there is an object in a related image. This system forms the basis of the Haar Cascade classifier. In addition, gradual processing is used to optimize the Haar Cascade method. If a feature cannot be found in searching for an object in the first step, other features will not be searched in the related image.

The value of a feature is the sum of the pixels in the white rectangles and is subtracted from the sum of the pixels in the black rectangles for each feature. The regions have the same size and shape, where they are adjoined horizontally or vertically. Given that the size of the detector is 24x24, the set of rectangular features is over 180.000, which introduces an extensive computational effort to the hardware. These rectangle features are calculated by an integral image method. The integral image at (x, y) location, contains the sum of the pixels (1). The functions

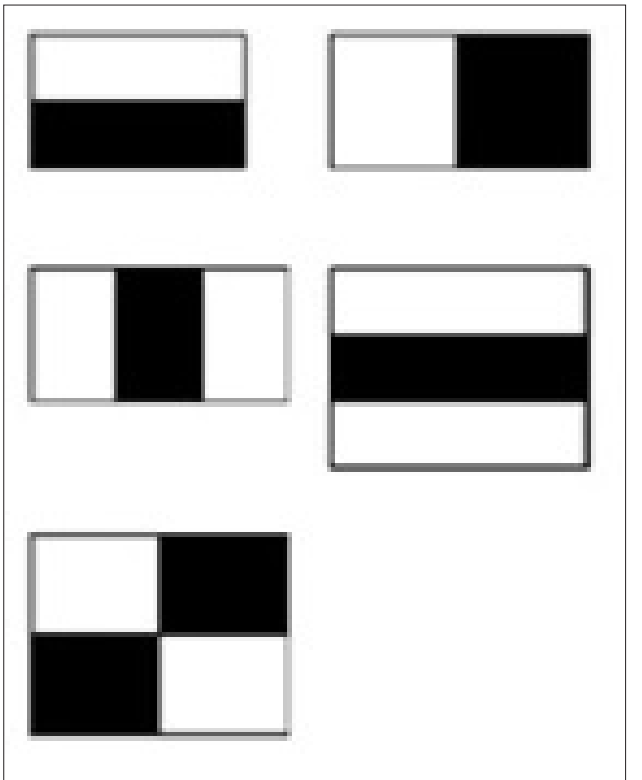


Figure 1. Sample rectangle features [1]

ii(x, y) and i(x, y) are used to define the integral image and original image, respectively, as given below [1].

$$ii(x,y) = ii(x,y) \sum_{x\prime<x,y\prime\leq y} i(x',y') \quad (1)$$

$$s(x,y) = s(x,y-1) + i(x,y) \quad (2)$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \quad (3)$$

An AdaBoost learning algorithm is used to select a small set of features (Figure 2) and train the classifier. The larger the dataset, the more efficiently AdaBoost will work. A more efficient classifier can be constructed with fewer features to be selected from this large dataset [5]. To support this goal, the weak learning algorithm is integrated to select a single feature that distinguishes positive and negative samples. For each feature, the weak learning algorithm determines the optimal threshold classification function to ensure that the minimum number of samples is misclassified [6]. To improve classification performance, individual classifiers are cascaded to construct a single and robust classifier as the final step of the training process. At the same time the computational effort is reduced remarkably. The basic aim is to construct smaller and more efficient classifiers while rejecting most of the negative sub-windows and detecting almost all positive samples. The general form of the detection process is a cascade. Positive results from the first classifier initiate the second one. The general form of the detection process is a cascade.

Positive results from the first classifier initiate the second one. Then, the positive ones from the second classifier trigger the third classifier. This triggering continues until the last classifier. Negative results at any stage lead to the rejection of sub-windows. The stages in a cascade are constructed by the training classifiers using AdaBoost, and the threshold is adjusted to minimize false negatives. The default threshold value is set for providing higher detection rates and higher false positive rates. The false positive rate is reduced, and each stage in cascade decreases the detection rate. A target is determined for the minimum decrease in false positives and the maximum detection rate. Each stage is trained by adding features until false positive rates and target detection are met.

The steps of Haar Cascade classifier training are explained as follows: First, the file paths of the images and fingertip coordinates are recorded in a text, and the "OpenCV Create Vectors" tool is used to create vectors of positive samples as the first step of training. The second step is the application of "OpenCV Train Cascade" tool where "numPos," "numNeg," "numStages," "minHitRate," and "maxFalseAlarmRate" parameter values are determined. "minHitRate" is the parameter which ensures that positive training data yields a proper detection output. For instance, a value of 0.7 would mean that 30% of positive object training data can be misclassified, which is a very poor detection rate. "maxFalseAlarmRate" is used to determine how many features are needed to be added. Each weak classifier should

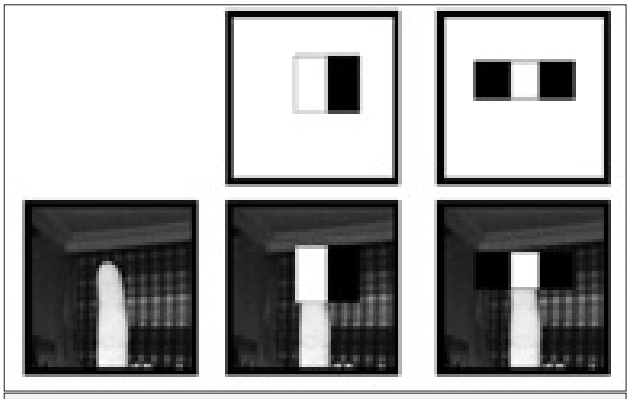


Figure 2. The first and the second features on fingertips selected by AdaBoost

have a good hit rate on positive samples. If this parameter value is high, the possibility of detecting non-fingertip objects will increase. On the other hand, if this value is low, it will not be able to detect the object even when there are slight differences. "numPos" is not the exact number of positive samples that are used in each stage. This value must be lower than the real number of positive ones. If the first stage of training uses all positive samples, there will be no samples left for the second stage. The formula to find proper "numPos" value is given (4).

$$numPos \leq \frac{PS - numNeg}{1 + (numStages) \times (1 - minHitRate)} \quad (4)$$

Fingertip Tracking

The MeanShift and CamShift algorithms are used for tracking by gathering the maximum value of the density function [21-25]. Both algorithms calculate the histogram in the selected area and determine the pixels with these values in the image. Tracking can be described as the movement of the center of the coverage area as long as the object moves. However, the center of this area does not match the centroid of maximum pixel density in varying perspectives when uses by the MeanShift algorithm. This algorithm is not adaptive to the size of objects, It remains constant, and the algorithm only considered statistical color information of object [21-22]. Continuously Adaptive MeanShift (CamShift) is an enhanced version of MeanShift and based on HSV color space. However, considering the fingertip tracking, these algorithms fail. Since an object is close to skin color in the background, example the human head or wall, the algorithm will naturally move toward a similar color region. The other issue with these algorithms is that they may be affected by environmental conditions such as illumination or obstacles [23-25]. The other method is an optical flow, which is the pattern of apparent motion of image between two sequential frames originating from the movement. It is the 2D vector field where each one is a displacement vector that shows the movement of points from the first frame to the last one. The movement of a ball in five sequential frames is shown in Figure 3. The

arrow shows the direction of the displacement vector. Optical flow operates if the pixel densities of an object do not change between sequential frames and pixels next to each other and have the same motion. Assume that the pixel in the first frame is I(x, y, z). The pixel moves up to distance (dx, dy) in time (dt) to the next frame. As long as these pixels are the same, and intensity does not change, the following can be derived.

$$I(x,y,z) = I(x+dx, y+dy, t+dt) \tag{5}$$

$$f_x u + f_y v + f_t = 0 \tag{6}$$

$$f_x = \frac{\delta f}{\delta x}; f_y = \frac{\delta f}{\delta y} \tag{7}$$

$$u = \frac{\delta x}{\delta t}; v = \frac{\delta y}{\delta t} \tag{8}$$

where f_x and f_y are image gradients, and f_t is the time gradient. To solve these equations, dense optical flow and sparse optical flow methods were proposed. Dense optical flow computes the optical flow for all pixels in every frame, which is based on the Gunnar Farneback's algorithm. Two channel arrays are gathered with optical flow vectors (u, v), and their magnitude and direction are found. Value and Hue, which are color planes from HSV, are related to the magnitude and direction of the image, respectively. Since this algorithm tracks all the motions in the frame, other motions, which are caused by finger or environment variables, may not allow following the fingertip properly [19]. The sparse optical flow method, which is based on Lucas-Kanade's algorithm, follows only a selected area or a pixel. Neighboring pixels have similar motion, as mentioned before. This algorithm takes a 3x3 patch around the point. (fx, fy, ft) as given in (9) can be found with nine points that have the same motion. The least square fit method is preferred for the solution of nine equations. Figure 4 shows the outputs of fingertip tracking algorithms.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \tag{9}$$

Sparse optical flow with pyramids is selected in this study because it allows for tracking selected pixels on the fingertip [18]. According to this method, as moving upwards inside the pyramid, the small movements disappear, and the large movements become smaller. The tracking loss issue is eliminated through the algorithm can track multiple points at the same time. In addition, the system's computational decreases when several points are tracked instead of all points in motion. This computational optimization improves the real-time operation as well.

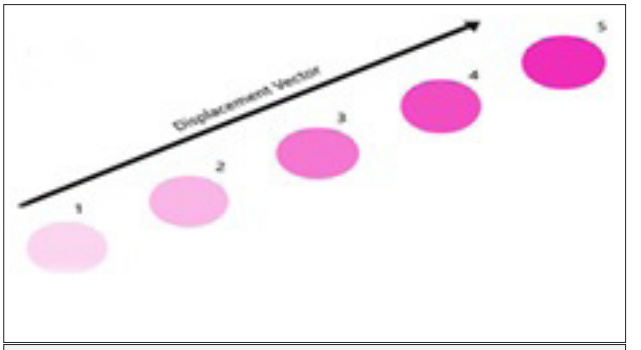


Figure 3. Displacement Vector of Ball Direction

Figure 4. Tracking Algorithms as CamShift, Dense Optical Flow and Sparse Optical Flow

Touchless Authentication Methodology

The flowchart of the proposed methodology is given in Figure 5. Initially, fingertip images are collected from several people within their consent, where positive and negative samples are separated. Positive samples are the fingertip images detected, whereas negative samples can be any image excluding fingertips, such as sky or room. 2591 positive samples and 1000 negative samples were collected from people with different skin colors. Fingertip images were taken within five different positions for each hand. The locations of fingertips in positive samples are marked using an algorithm that creates a square area in the acquiring image from the camera. When the fingertip enters this area, the algorithm saves images to file path and pixel locations to the text document. Following, the file paths of the images and fingertip coordinates are recorded in a text, and the "OpenCV Create Vectors" tool is used to create vectors of positive samples as the first step of training. The second step applying "OpenCV Train Cascade" tool where parameter values are evaluated as given in Table 1.

Once the classifier is trained with the optimal parameters of Table 1, whereas Classifier 3 is selected as explained in the experimental results section, a new frame from the user is acquired. In this study, a webcam with a resolution of 640x480 is used. However, when processing the image on hardware, the performance requirement will be maximum. Therefore, images with a resolution of 640x480, which are gathered from the webcam, are processed by transforming them to a resolution of 320x240 for each frame. This transformation decreases the computational effort to a large extent. The acquired image is preprocessed to improve computational performance. First, it is converted to a gray format. Following, histogram equalization is used to enhance the contrast quality of the images. Erosion and dilation methods are used to remove possible noise

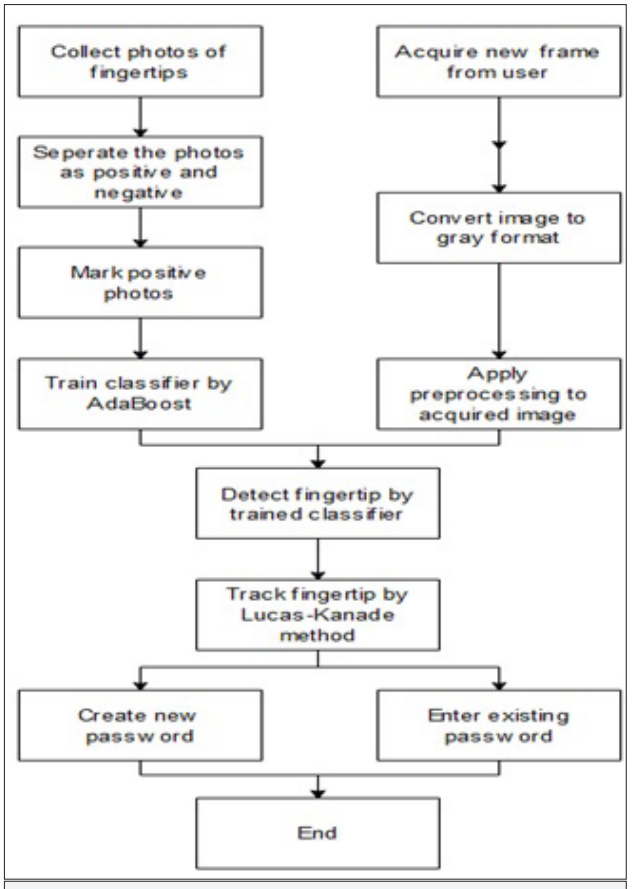


Figure 5. The Proposed Algorithm

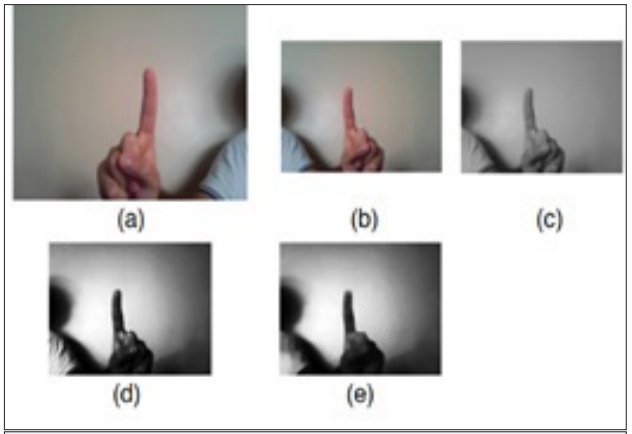


Figure 6. a-e. Preprocessing: (a) Acquired Image (b) Flip & Resize (c) Convert to Gray from RGB (d) Histogram Equalization (e) Erosion and Dilation

and correct the shape of the object that is to be detected. The steps of preprocessing, which are color conversion, resizing the image, histogram equalization, removing noises, and fixing shape distortions, are given in Figure 6.

After detecting the fingertip location by the Haar Cascade algorithm, coordinates are transferred to a sparse optical flow tracking algorithm. A zero matrix is created, and the first frame that is taken before the tracking is assigned to zero matrix so that consecutive images will not conflict. Determining nine points on the fingertip provides a more robust tracking than a single point.

The fingertip tracking is used for two purposes in this study; either creating or entering a password. If the user chooses to create a new password, the algorithm will continue to run and wait for the password entry. For creating a password, the user's fingertip should enter the area of the first digit of the password and then move into other areas, which are numbered from 1 to 9 to continue creating the password. The number of the area is transferred to the related string. The solution is found with a Boolean variable so that the same number is not transferred to the password string while the finger is still in the area. If the user has completed the password and remains within 50 frames in the last digit area of the password, the system will accept this as the completion of the password. Following, the user should save the password by moving into the "Save" area if this attempt is the creation of the password. When saving the password, the related string is made null again, and the password is transferred to the string that is used for registration. Once the password is created, if the user prefers to unlock the system, the fingertip should be moved to the area where the password starts. If the user enters the predefined password and then confirms it, the lock will be opened. The proposed algorithm figures out the difference between the password entry and the new password creation whether the relevant string variable is empty or not. The application of the proposed system is given in Figure 7.

Experimental Results

Haar Cascade classifier is trained with 2591 positive samples and 1000 negative samples, as explained in the previous section. By setting varying values for parameters "minHitRate," "maxFalseAlarmRate," and "numPos," four classifiers are constructed. The performances of these classifiers are tested on real-time finger detection by setting the number of frames as 25, 50, and 100, respectively. Detection rate (DR) and error rate (ER) are evaluated using Algorithm 1. Table 2 shows that the most effective results are obtained with Classifier 3. The rest of the experiments are conducted using this classifier.

Algorithm 1. Performance Evaluation of Classifier Versions

Require:

(a) DO, number of detected fingertips within a frame;
(b) DN, total number of detected fingertips;
(c) EN, total number of misdetected fingertips;
(d) FN, total frame number;

1: Camera is initialized
2: Number of frames is started to count

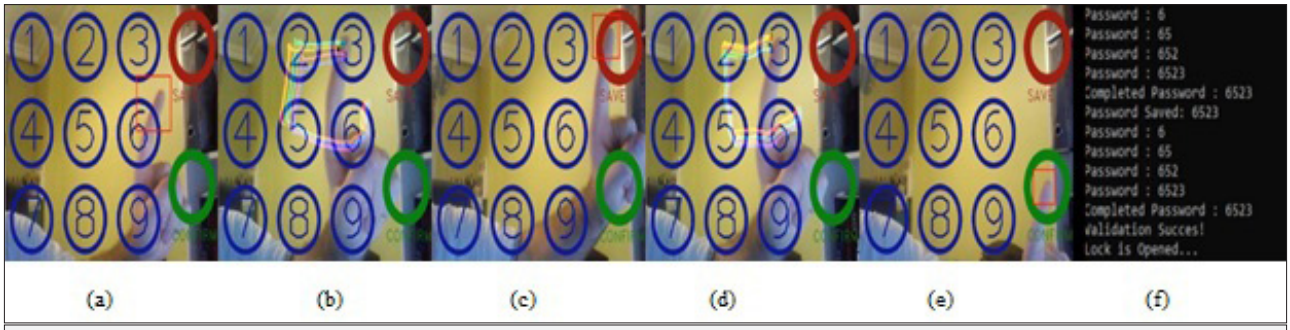


Figure 7. a-f. Touchless Authentication System: (a) Finger Detection (b) Entering New Password (c) Saving Password (d) Entering Existing Password (e) Confirming Password (f) System Outputs

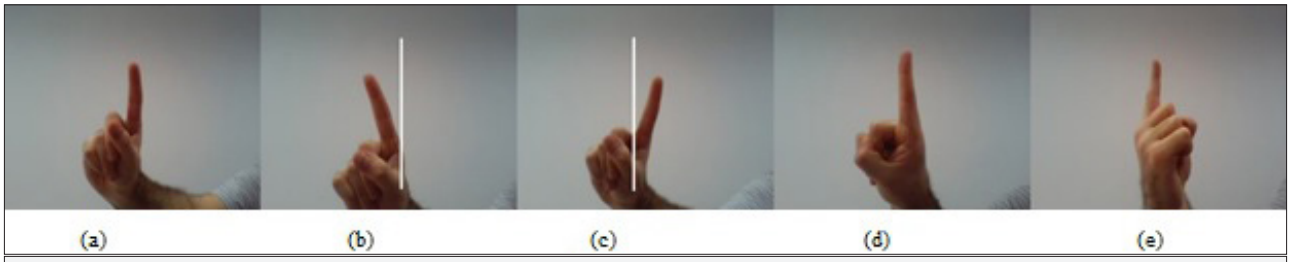


Figure 8. a-e. Orientation: (a) 0˚ (b) 30˚ left in y-axis (c) 30˚ right in y-axis (d) 90˚ left in z-axis (e) 90˚ right in z-axis

Table 1. Parameters and Execution Time of Classifiers

	Classifier 1	Classifier 2
minHitRate	0.995	0.995
maxFalseAlarmRate	0.25	0.5
numPos	1450	1450
numNeg	1000	1000
numStages	17	17
TPT	Approx. 5 days	32 min
	Classifier 3	**Classifier 4**
minHitRate	0.995	0.99
maxFalseAlarmRate	0.45	0.4
numPos	1400	1350
numNeg	1000	1000
numStages	17	17
TPT	37 min	62 min.

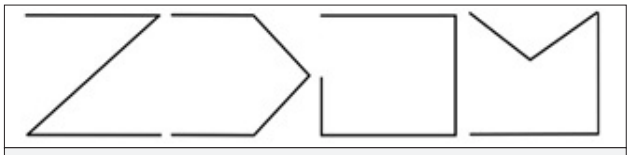


Figure 9. Password Patterns

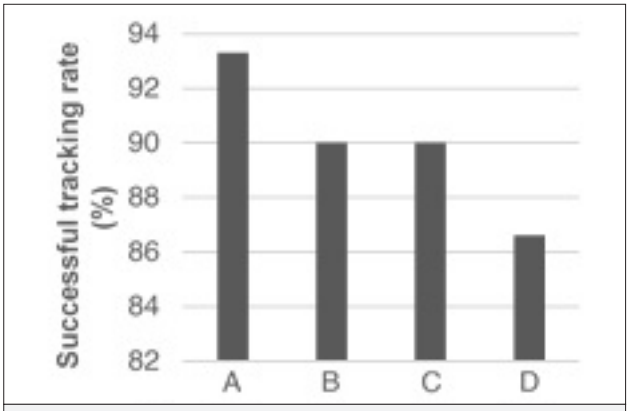


Figure 10. Test Results of Tracking of Password Patterns

3: Haar Cascade is run for fingertip detection within a frame
4: if DO == 1 then
5: DN++
6: if DO > 1 then
7: DN++, EN++
8: DR = (100*DN)/ FN
9: ER = (100*EN)/FN
10: end

The accurate detection of the fingertip with invariance to pose, lighting, and brightness are essential and security. In this study, fingertips of 15 people are positioned in different orientations independently from environment brightness and lighting conditions for evaluating the performance of a touchless authentication system. The images of rotated fingertips by 30° in the

Table 2. Haarcascade Classifiers' Test Performance on Fingertip Detection

	# of frames=25		# of frames=50		# of frames=100	
Classifier	**Detection rate**	**Error rate**	**Detection rate**	**Error rate**	**Detection rate**	**Error rate**
1	44%	0	40%	0	39%	3%
2	76%	24%	88%	36%	91%	41%
3	84%	0	92%	2%	95%	2%
4	60%	12%	68%	10%	70%	7%

Table 3. Fingertip Detection Performance in Various Orientations

	Detection Rate	
Orientation	**Left Hand**	**Right Hand**
0	100%	93,3%
30º left in y-axis	93,3%	93,3%
30º right in y-axis	86,6%	86,6%
90º left in z-axis	86,6%	86,6%
90º right in z-axis	80%	86,6%

Table 4. Touchless Authentication System Performance in Various Backgrounds

	Password Identification	
Background	**True**	**False**
Plain wall	92%	86%
Human Face	76%	82%
Complicated	44%	50%

y-axis and 90° in z-axis on the white background constituted the test dataset, which is given in Figure 8.

Haar Cascade classifier with the optimal parameters given in the previous subsection is applied to detect fingertips of 15 people with given orientations. Test results of Haar Cascade classifier detection rates and error rates are provided in Table 3. Figure 9 shows some of the preferred password patterns by the users. The test performance of sparse optical flow with the Lucas-Kanade algorithm of tracking those patterns is provided in Table 4. The most successful tracking rate is obtained with pattern (a), followed by (b) and (c). Pattern (d) is more complicated to track than the rest, therefore tracking rate is also less when compared with the rates of other patterns. Password identification with various backgrounds is provided in Table 4. Here, the user sets a password pattern to the system, and validation is performed by entering the right password pattern for 50 times and wrong patterns for 50 times. Tests are carried out using three backgrounds, a plain wall, a human face and a complicated background, respectively. The identification rate of true and false entries decreases as the complexity of the background increases. Finally, the practicality of the proposed system is tested with 15 people of different ages using a plain wall background. 12 people out of 15 successfully enter a new password and reenter it to unlock the system. 80% success in the usability of the system is achieved, which shows that the touchless authentication system is user-friendly and suitable for secure applications.

OS used to implement the proposed system is Windows 10, and the running software in C++. The processor type is Intel® Core™ i7-6700HQ with 8 cores and 2.6 GHz of clock speed. Total RAM is 16 GB DDR4L 1.2V 2133 MHz. CPU usage for the proposed system is 42%, and Visual Studio is 3.2%. RAM usage for the proposed system is 3%, and Visual Studio is 8.6%. Elapsed time for initialization of camera and system, algorithm detection part, and algorithm tracking part are 58 ms, 11 ms, and 12 ms, respectively. Camera's overall setting requires 9 FPS, and the limit of finger movement speed is 6 FPS.

Conclusion

Personal data security is crucial since biometric traces of a person are likely to be accessed and reused. Most of the authentication devices used in public or private require personal information such as passwords through a screen where the fingertip pattern is left. In this work, a secure methodology based on visual fingertip trajectories is proposed that allows untouched usage of authentication devices. This study's contribution is the efficient detection and tracking of the fingertip movement without leaving any biometric trace on the screen, thus providing secure and dependable authentication.

Haar Cascade algorithm is preferred for fingertip detection. The main reason for the usage of this algorithm is that the computational effort is less, and therefore, it is very suitable for low cost and real-time application. Deep learning may be considered as another solution for fingertip detection; however, it requires more computational source, and execution time is likely to be higher than that of the Haar Cascade algorithm. In addition, more expensive and complex hardware is required for the implementation of a deep learning method than that

of the Haar Cascade algorithm when this methodology will be implemented through hardware.

Several tests are carried out for evaluating the performance of fingertip detection. Fingertips of 15 people are positioned in different orientations independently from environment brightness and lightning. Users provide their password patterns, which are tracked with the sparse optical flow with the Lukas-Kanade algorithm. Password identification performance sing these patterns in the various background is provided. To demonstrate the practicality of the system, 15 people from various ages are asked to set a password and reenter it. Eventually, 12 people are successful in using the system on their own. This rate may be increased if age groups are comprised of young people.

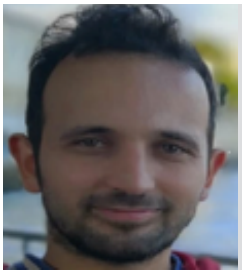
As further work, this methodology will be applied for usage in mobile devices. An algorithm will be developed for random letter/number placement on screen to increase the security and dependability of the proposed system.

References

1. P. Viola and M. Jones "Rapid Object Detection using a Boosted Cascade of Simple Features," IEEE Conf. Computer Vision and Pattern Recognition (CVPR), vol. 1, Dec. 2001.
2. R.M. Gurav, P.K. Kadbe, "Real time Finger Tracking and Contour Detection for Gesture Recognition using OpenCV," Int. Conf. Industrial Instrumentation and Control (ICIC), pp. 974-977, May 2015. [Crossref]
3. Y. Chunxuan, W. Jingtao, "Finger-Fist Detection in First-Person View based on Monocular Vision using Haar-Like Features" 33rd Chinese Control Conf., pp. 4920-4923, June 2014.
4. Y. Ma, X. Ding, "Robust Real-Time Face Detection Based on Cost-Sensitive AdaBoost Method," Int. Conf. Multimedia and Expo, ICME'03 Proc., vol. 2, pp. 465-472, 2003.
5. Y. Freund, R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," J. Computer and Systems Sciences, vol. 55, pp. 119-139, 1997. [Crossref]
6. K. Tieu, P. Viola, "Boosting Image Retrieval," IEEE Conf. Computer Vision and Pattern Recognition, vol. 1, pp. 228-235, 2000.
7. E. ul Haq, S.J.H. Pirzada, M.W. Bing, "New Hand Gesture Recognition Method for Mouse Operations," IEEE 54th Int. Midwest Symp. Circuits and Systems, pp. 1-4, Aug. 2011.
8. D. Popa, V. Gui, M. Otesteanu, "Real-Time Finger Tracking with Improved Performance in Video Sequences with Motion Blur," 38th Int. Conf. Telecommunications and Signal Processing (TSP), pp. 1-6, June 2015. [Crossref]
9. R. Agrawal, N. Gupta, "Real Time Hand Gesture Recognition for Computer Interaction," IEEE 6th Int. Conf. Advanced Computing, pp. 73-77, Feb. 2014.
10. R. Meena Prakash, T. Deepa, T. Gunasundari, N. Kasthuri, "Gesture Recognition and Finger Tip Detection for Human Computer Interaction," Int. Conf. Innovations in Information, Embedded and Communication Systems, pp. 1-4, March 2017. [Crossref]
11. G. Simion, V. Gui, M. Otesteanu, "Finger Detection Based on Hand Contour and Colour Information," 6th IEEEInt. Symp. Applied Computational Intelligence and Informatics (SACI), pp. 97-100, May 2011. [Crossref]
12. S.K. Kang, M.Y. Nam, P.K. Rhee, "Color Based Hand and Finger Detection Technology for User Interaction," Int. Conf. Convergence Hybrid Information Technology, pp. 229-236, 2008. [Crossref]
13. C. Joniez, E. Monari, C. Qui, "Towards Touchless Palm and Finger Detection for Fingerprint Extraction with Mobile Devices," Int. Conf. Biometrics Special Interest Group, pp. 1-8, Sep. 2015. [Crossref]
14. G. Wu, W. Kang, "Robust Fingertip Detection in a Complex Environment," IEEE Trans. Multimedia, vol. 18, no. 6, pp. 978-987, June 2016. [Crossref]
15. Y. Huang, X. Liu, L. Jin, X. Zhang, "DeepFinger: A Cascade Convolutional Neuron Network Approach to Finger Key Point Detection in Egocentric Vision with Mobile Camera," IEEE Int. Conf. Systems, Man, and Cybernetics, pp. 2944-2949, 2015. [Crossref]
16. Crowley James L., F. Berard, J. Coutaz, "Finger Tracking as an Input Device for Augmented Reality," Int. Workshop on Face and Gesture Recognition, June 1995.
17. D. Lee, Y. Park, "Vision-based Remote-Control System by Motion Detection and Open Finger Counting," IEEE Trans. Consumer Electronics, vol. 55, pp. 2308-2313, Nov. 2009. [Crossref]
18. B.D. Lucas, T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," Proc. the 7th Int. Joint Conf. Artificial Intelligence, pp. 674-679, Apr. 1981.
19. G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," Proc. the 13th Scandinavian Conf. Image Analysis, pp. 363-370, June 2003. [Crossref]
20. G. Wu, W. Kang, "Vision-Based Fingertip Tracking Utilizing Curvature Points Clustering and Hash Model Representation," IEEE Trans. Multimedia, vol. 19, pp. 1730-1741, Apr. 2017. [Crossref]
21. L. Yu, "Moving Target Tracking Based on Improved MeanShift And Kalman Filter Algorithm," 13th IEEE Conf. Industrial Electronics and Applications, pp. 2486-2490, May 2018.
22. K. Du, Y. Ju, Y. Jin, G. Li, S. Qian, Y. Li, "MeanShift Tracking Algorithm with Adaptive Block Color Histogram," 2nd Int Conf. Consumer Electronics, Communications and Networks, pp. 2692-2695, Apr. 2012.
23. G.R. Bradski, "Real Time Face and Object Tracking as a Component of a Perceptual User Interface," Proc. IEEE Workshop on Applications of Computer Vision, pp. 214-219, Oct. 1998.
24. C. Xiu, X. Su, X. Pan, "Improved Target Tracking Algorithm Based on Camshift," Chinese Control and Decision Conf., pp. 4449-4454, June 2018. [Crossref]
25. L. Li, Y. Luo, "Improved Video Moving Target Tracking Based on Camshift," Am. J Computational Math., vol. 6, no. 4, pp. 357-364, Jan. 2016. [Crossref]
26. X. Pan, Z. Ling, A. Pingley, W. Yu. N. Zhang, K. Ren, X. Fu, "Password Extraction via Reconstructed Wireless Mouse Trajectory," IEEE Trans. Dependable and Secure Computing, vol. 13, pp. 461-473, March 2015. [Crossref]

Vefak Murat Akman received a B.Sc. degree in Electronics and Communications Engineering from Yıldız Technical University, Turkey in 2018 He is currently working toward the M.Sc degree in Computer Engineering, Istanbul Technical University, Turkey. His research interests include computer vision, secure biometrics and machine learning.

Revna Acar Vural received the B.Sc. in 2002, M.Sc. in 2004 and PhD in 2011 on electronics and communication engineering, Yildiz Technical University, Istanbul, Turkey. She is an Assistant Professor in Department of Electronics and Communication Engineering in Yildiz Technical University in Istanbul, Turkey and part time lecturer in Department of Electrical and Electronics Engineering in Bilgi University in Istanbul, Turkey. Her research interests include circuit design optimization, machine learning and image processing based secure drive assistance.



Kemal Talha Koç received the B.Sc. and M.Sc in Electronics and Communications Engineering from Yildiz Technical University, Turkey, in 2015 and 2019, respectively. His research interests include machine learning, object-oriented programming, image processing and data science.