# PursuitPass: A Visual Pursuit-Based User Authentication System

Alex Torquato S. Carneiro*, Carlos Eduardo L. Elmadjian*, Candy Gonzales*,
Flavio L. Coutinho† and Carlos H. Morimoto*

* Institute of Mathematics and Statistics
University of São Paulo, São Paulo, Brazil
Email: {alexc, elmad, candytg, hitoshi}@ime.usp.br
†School of Arts, Sciences and Humanities
University of São Paulo, São Paulo, Brazil
Email: flcoutinho@usp.br

***Abstract*—As our lives get more deeply submerged in digital format, ubiquitous access to sensitive data requires more secure and efficient user authentication procedures. Methods that solely relied on password entry were lately enhanced with the use of biometrics. Yet, these techniques can still be tricked by, for example, recordings of the face, voice, and fingerprint cloning. In this paper we introduce PursuitPass, a compact, robust, and efficient visual pursuit-based authentication system. PursuitPass is a user calibration-free method that requires the user to enter a password by visually pursuing moving targets on a small screen, such as a public ATM or a personal mobile phone. Because eye movements are used as input, passwords are better protected against shoulder surfing. Also, since targets can potentially move in unpredictable ways, it naturally imposes a liveness feature that cannot be counterfeited by recordings of the eyes. We investigated four pattern-matching algorithms to match visual pursuit user data with the movement of the targets. Two experiments were conducted. The first experiment aimed to define the best performing matching algorithm and configuration for PursuitPass. The second experiment aimed to evaluate the performance of our prototype. PursuitPass achieved a 96.82% accuracy with an average time of 10.42 s on a series of 4-digit PIN entry trials.**

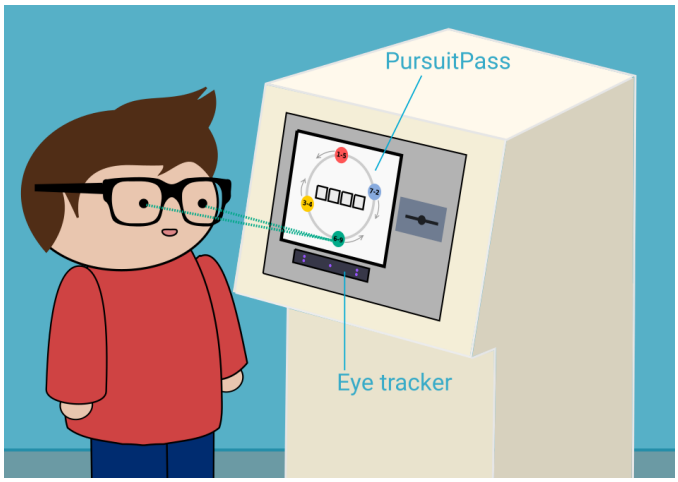


Fig. 1. PursuitPass concept and use case as a secure authentication mechanism based on visual pursuit.

## I. Introduction

To avoid attacks that can expose the user's credentials such as smudge [1] and shoulder surfing [2], authentication systems rely on input methods that are hard to be monitored by a third party and leave no traces of use. In this scenario, the use of eye movements as input can be advantageous since it is harder to observe the users' eyes in comparison to manual key presses, and eye movements do not leave any marks behind, preventing smudge and thermal attacks. Figure 1 shows a possible scenario of a user interacting with an eye movement-based interface.

Eye movements can be captured using a video-based eye tracker [3], a device that uses at least one camera and computer vision algorithms to detect and track the eyes. Basic eye movements can be classified into fixations (when the eye position is kept relatively stable), saccades (rapid eye movements between fixations), and smooth pursuits (when the eye is pursuing moving objects, such as flying birds and running dogs).

When it comes to eye-based selection techniques, dwell-time is typically used to select a target on a computer screen, i.e., a target is only selected after the user fixates at its location for a certain dwell-time (usually fixations between 400 ms to 1000 ms). This technique can be used to control virtual keys on an ATM machine using eye movements, for example. One of its major drawbacks is that it requires user calibration to map an eye feature, such as the pupil center, from the eye camera coordinate system to the computer (or ATM) screen. Smooth pursuit movements, however, do not require calibration and have been considered as an efficient and reliable alternative to select moving targets, though real world applications are still lacking, such as authentication by entering a PIN code.

The EyePassword, developed by Kumar et al. [2], is an example of an authentication system that uses dwell-time to control a virtual keyboard to enter passwords using the user's gaze, after calibration. The authors investigated three types of on-screen keyboards: QWERTY, alphabetic and ATM numeric keypads. They evaluated the performance of the dwell-time

method and compared it with a multimodal technique that used a trigger key (spacebar) to select a key instead of dwell-time. Their results showed that the dwell-based method had lower error rates (about 3%) than the multimodal method (about 4%). The average speed to enter 8-character length passwords was 9.2 seconds using the dwell-based method with a 450 ms dwell on the QWERTY keyboard.

To avoid shoulder surfing attacks, visual feedback must be avoided. Kumar et al. [2] used audio feedback to indicate that a key was selected, which reveals the length of the password. To enhance security and facilitate the task of entering the password by gaze, Weaver et al. introduced EyeDent [4], a technique that processes a constant stream of data from the eye tracker and identifies the main clusters that correspond to the set of keys gazed by the user. Using a virtual numeric keypad, users of EyeDent were able to enter a 4-digit PIN (personal identification number) in about 2.7 s with 83% accuracy.

Eye gestures defined by a sequence of fixations and saccades can be recognized directly from the eye tracker data stream without user calibration. One such system was proposed by Rajanna et al. [5]. Their system displays 36 geometric shapes on the computer screen and the user must follow the contours of 3 shapes. In their experiments using a pattern matching technique to compute a score between the shapes and eye movements, they obtained 96% of authentication accuracy with disturbed calibration.

De Luca et al. [6] suggested two gesture-based methods: EyePIN and EyePassShapes. EyePIN uses one gesture per symbol, similar to graffiti alphabets used in old palm computers. Besides the problem of learning each gesture, performing several of them to complete a password can be slow. To reduce memory load and facilitate learnability, they proposed EyePassShapes, where users must replicate geometric shapes described by a sequence of vertices within a regular grid, similar to shape passwords used in smart phones.

Though eye gestures can be detected directly from the eye tracker stream without user calibration, gestures can be recorded and used by an attacker. A calibration-free alternative that can be robust to recorded videos of eye movements rely on smooth pursuits. A key aspect of smooth pursuits is that a moving stimulus is required to trigger them. They also do not require any memorization as in gesture-based techniques, and because the motion of visual targets can be randomized, pursuit-based methods tend to be more robust against video recordings of eye movements. Examples of authentication techniques that exploited pursuits were presented by Cymek et al. [7] and Liu et al. [8].

Cymek et al. [7] proposed a method with 16 moving targets that basically corresponded to a numeric keypad commonly seen in ATM machines. Each key moved along predefined paths composed of horizontal and vertical strips. In their experiments, this technique resulted in an average speed of 25 s to enter a 4-digit PIN, with 97.57% accuracy. Because keys moved in distinct directions, their method requires a larger screen than available in most ATMs, and because each key always performs the same pattern, it is vulnerable to recording attacks.

Liu et al. [8] suggested a visual pursuit technique for mobile phones. Their proposed interface was composed of 4 visual targets that could move along a line, either up, down, left, or right. In their experiments, this technique resulted in an average speed of 9.6 s to enter a 5 symbol PIN, with 91.6% accuracy. Due to the simplicity of the trajectories, attackers could also easily identify the eye movements.

Smooth pursuit-based systems for PIN authentication do not require users to memorize gestures or fixate targets to select virtual keys. They just need the user to pursuit a target associated with an intended symbol with his/her eyes. Because of that, interaction occurs in a much more spontaneous and intuitive way, something that motivated us to design and evaluate *PursuitPass*, our proposed authentication system for ATM machines based on smooth pursuit.

PursuitPass is a proof of concept that demonstrates the advantages of smooth pursuits and matching methods as two fundamental parts in the problem of private and secure user authentication. The application's main purpose is to validate someone's credentials without exposing the user to *shoulder surfing* and without requiring prior user calibration, which is the most common downside of other related gaze-based methods.

Next sections are organized as follows: we describe the design of PursuitPass in the Section II, detail the theoretical background employed by us in Section III, describe how PursuitPass was evaluated in Section IV, present our results in Section V, discuss about our results and compare to other similar systems in Section VI, and, finally, make our conclusions and future intentions in Section VII.

## II. PursuitPass Design

PursuitPass was designed as an authentication method suited for both public and personal small displays, such as ATM machines and mobile phones. Because it is based on visual pursuits, it is calibration-free and robust to shoulder surfing, smudge, thermal, and acoustic attacks. We have included random features in the behavior of the visual targets that requires user liveness, i.e., the technique is robust against recorded videos of eye movement patterns.

The interface comprises four targets moving in circular orbits around a PIN input field. Each target has its own color and is associated with a label corresponding to two digits, as it is frequently seen in ATMs. Initially, the targets are rendered evenly, 90 degrees apart, around the circular track in a stationary way, as shown in Figure 2. After one second, the visual targets start rotating along the circular track. All targets move with the same angular speed, but with two of them rotating in clockwise direction while the other two rotate counterclockwise. Targets moving in the same direction also have a phase gap of 180 degrees between them.

Due to the circular orbits, PursuitPass is appropriate for small screen devices such as ATMs and mobile phones. We opted for four pair of digits instead of five in our prototype in order to maximize the phase difference between targets

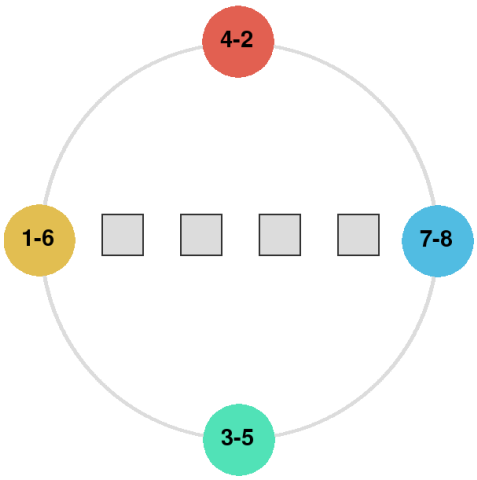


Fig. 2. Initial state of the PursuitPass interface: all targets are distributed evenly across a circular track with randomly assigned pair of digits.

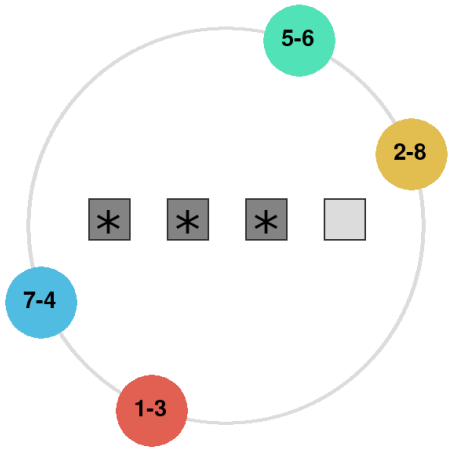


Fig. 3. When a selection is triggered, the leftmost empty input box is filled with an asterisk and changes its color. Targets rotating in the same direction (i.e., red-green, blue-yellow) always maintain a 180-degree gap between them.

rotating in the same direction. Figure 4 presents the working architecture of PursuitPass with the information flow from eye movement data and the four targets, yielding the similarity values $\{m_1, m_2, m_3, m_4\}$ for the chosen pattern matching method.

Digit selection is performed by simply following the associated target's path with one's eyes. Once a selection intent is recognized by the system, the leftmost empty square is filled with an asterisk and changes its color to dark gray in order to indicate that a PIN number was entered, as it can be seen in Figure 3. At the same time, all targets stop rotating, return to their original positions and their labels are randomized, remaining stationary again for another second before they start moving once more. When the fourth selection is triggered, the system performs the authentication procedure, informing the user whether the entered PIN number was valid or not.

One remaining challenge is how to match the eye movement data stream with the movement of the visual targets and

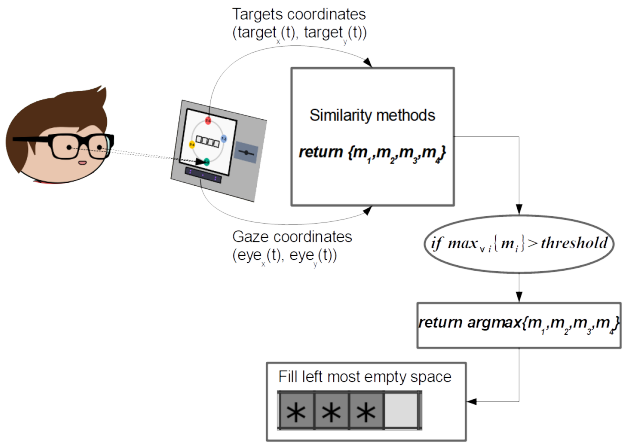


Fig. 4. PursuitPass architecture.

detect when the user is performing a pursuit. The next section describes four algorithms that have been used to detect pursuits along circular orbits.

## III. Algorithms for Detecting Circular Pursuits

In 2013, Vidal et al. [9] introduced *Pursuits*, a technique in which elements of an interface (displayed on large screens) can be selected by simply following the desired element with the eyes. In *Pursuits*, some moving objects (targets) are displayed on the screen, and the path described by each object is compared with the gaze path to determine which object is being followed by the user.

To measure the similarity between a target path and the gaze path, the authors resorted to Pearson's correlation coefficient. For a sequence of 2D target coordinates ($target$) and a sequence of 2D gaze coordinates ($eye$) in a given time window, the correlation is defined as:

$$\rho = \frac{\sum_i (eye_i - \mu_{eye})\,(target_i - \mu_{target})}{\sigma_{eye}\sigma_{target}}, \qquad (1)$$

where the sum is calculated with the samples that correspond to a time window, i.e., $t_i \leq i \leq t_f$ for the time interval defined by $[t_i, t_f]$, whereas $\mu_{eye}$, $\mu_{target}$, $\sigma_{eye}$, and $\sigma_{target}$ are, respectively, the mean values and standard deviations of $eye$ and $target$. The correlation coefficient is calculated for both axes $(eye_x, eye_y)$ and $(target_x, target_y)$ within a time window in the range [0.1s, 2s].

In *Pursuits*, the authors assumed generic paths for moving targets, but more recent works proposed the use of circular paths, such as [10]–[13]. In the most recent study on target selection based on similarity between eye movements and target paths [13], the authors evaluated 4 distinct methods to compute the similarity between the eye and target movements: Basic Correlation, Rotated Correlation, 2D Correlation, and Profile Matching. Each one of them is described in greater detail in the following subsections.

### A. Basic Correlation

The Basic Correlation method, based on the proposal of Vidal et al. [9], consists in calculating the Pearson's correlation coefficient to both axes of movement. Being $\rho_x$ and $\rho_y$ the correlation coefficients for, respectively, the $x$ and $y$ axis, a target is considered selected if both $\rho_x$ and $\rho_y$ are greater than a specified threshold. Esteves et al. [10] simplified this process by comparing just the smallest correlation value against the threshold. Hence, this method returns $\min\{\rho_x, \rho_y\} > threshold$ as the result for pursuit detection.

### B. Rotated Correlation

Depending on the trajectory shape described by a target, computation of Basic Correlation can be problematic when the 2D coordinates in a given time window presents no variance in one of the axes (i.e., the standard deviation for such axis will be zero). To cope with this limitation, Carter et al. [11] introduced the Rotated Correlation method. In this method, both the target and eye coordinates are rotated in order to maximize the variation in both axes, thus avoiding scenarios where no variance exists for one of them. To apply the rotation to both the target and eye coordinates, a rotation matrix ($RotMat$) is defined as:

$$RotMat = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \times \begin{bmatrix} \vec{v}_{1x} & \vec{v}_{2x} \\ \vec{v}_{1y} & \vec{v}_{2y} \end{bmatrix}, \tag{2}$$

where $\vec{v}_1$ and $\vec{v}_2$ are the eigenvectors computed from the target coordinates. The rotated values obtained by multiplication of the eye and target coordinates by $RotMat$, respectively $(eye_u, eye_v)$ and $(target_u, target_v)$, are then used to calculate the Pearson's correlation coefficients $\rho_u$ and $\rho_v$. This method returns $\min\{\rho_u, \rho_v\} > threshold$ as the result for pursuit detection.

### C. 2D Correlation

The basic and rotated correlation methods compute the correlation for each axis independently. The 2D correlation method, introduced by Velloso et al. [13], differs from these previous methods by considering both axes simultaneously in the computation of the correlation value. The 2D Correlation method first normalizes the i-st sample of the target and eye coordinates, using $\sigma_{eye} = \max\{\sigma_{eye_x}, \sigma_{eye_y}\}$ and $\sigma_{target} = \max\{\sigma_{target_x}, \sigma_{target_y}\}$, as follows:

$$\begin{aligned} \overline{eye}_{x,i} &= \frac{eye_{x,i} - \mu_{eye_x}}{\sigma_{eye}} \\ \overline{eye}_{y,i} &= \frac{eye_{y,i} - \mu_{eye_y}}{\sigma_{eye}} \\ \overline{target}_{x,i} &= \frac{target_{x,i} - \mu_{target_x}}{\sigma_{target}} \\ \overline{target}_{y,i} &= \frac{target_{y,i} - \mu_{target_y}}{\sigma_{target}} \end{aligned} \tag{3}$$

where the values of $\mu$ and $\sigma$ are calculated with the time window. The correlation coefficient $c$ is then computed as:

$$c = \frac{\sum_i \sqrt{(\overline{eye}_{x,i} - \overline{target}_{x,i})^2 + (\overline{eye}_{y,i} - \overline{target}_{y,i})^2}}{\sum \sqrt{\overline{eye}_{x,i}^2 + \overline{eye}_{y,i}^2}} \tag{4}$$

Finally, the 2D Correlation method returns $c < threshold$ as the result for pursuit detection.

### D. Profile Matching

The Profile Matching method, also introduced by Velloso et al. [13], describes a sequence of coordinates (either from a target or eye) in a time window as an orthogonal profile that is computed by projecting the coordinates over a *baseline*. The *baseline* is defined as the vector that connects $P_0$ (first coordinate in the sequence) to $P_m$ (farthest coordinate from $P_0$). The projected points are also scaled, to normalize the baseline length.

To compute the similarity between a target and the eye trajectory, orthogonal profiles are computed for each of them. Let $(target_u, target_v)$ and $(eye_u, eye_v)$ be, respectively, the target and eye coordinates expressed in the orthogonal profile representation. A coefficient $c$, which indicates how similar they are, is then computed as follows:

$$c = \sum_i (eye_{u,i} - target_{u,i})^2 + (eye_{v,i} - target_{v,i})^2 \tag{5}$$

Due to the projection and normalization process, the comparison of two profiles will be both scale and rotation (phase) invariant. While scale invariance is desirable (since target and eye coordinates are expressed in distinct coordinate systems if an uncalibrated eye tracker is used), rotation invariance might be problematic when targets follow circular paths (two or more targets following a circular path with the same angular speed, but shifted from one another will always exhibit the same profile. Thus, assuming a person is following one of them, all targets will always be highly correlated to the eye, making distinction of which one is being observed impossible).

To overcome this limitation, Velloso et al. [13] proposed combining the *baseline* vectors associated with both the target and eye to recover phase information. Given $\vec{b}_{target}$ and $\vec{b}_{eye}$ the *baselines* for the target and eye coordinates, respectively, an adjusted similarity coefficient can be computed by:

$$c_{adj} = \frac{\vec{b}_{eye} \cdot \vec{b}_{target}}{1 + ln(1 + c)} \tag{6}$$

and after computing $c_{adj}$, the Profile Matching method returns $c_{adj} > threshold$ as the result for the pursuit detection.

Though these four methods have been evaluated in [13], with the 2D correlation presenting the best performance in a scenario distinct of ours, those methods have not been evaluated in a practical application. To choose which algorithm to use in PursuitPass and evaluate its performance, we have conducted user experiments that are described in the next section.

Fig. 5. Nature photography appreciated by users to represent a non-pursuit task.

## IV. Evaluation

We conducted one pilot test and one experiment to evaluate PursuitPass. The pilot test was designed to investigate the performance of the pursuit detection algorithms described in Section III, and to decide which algorithm and parameters to use in PursuitPass based on their speed and accuracy. In the experiment, the performance of the PursuitPass authentication system was evaluated based on the findings obtained from the pilot test.

### A. Pilot test

The pilot test was designed to investigate the performance of pursuit detection algorithms. Four students and researchers from the local institution (1 female), all able-bodied, with normal or corrected-to-normal vision, participated in the experiment. Ages varied between 33 and 38 years old (mean: 35). All of them reported previous experiences with eye tracking devices.

The pilot test consisted of 2 trials. In each trial participants were required to pursuit one moving target among four (so that data related to target pursuit could be obtained) for a total time of 10 seconds. After that, they were instructed to appreciate a nature photography (to obtain samples that don't correspond to pursuit action), presented to them (see Figure 5), also for 10 seconds.

Participants were positioned between 60 to 70 cm away from the computer monitor. In this setup, the diameter of the circular track spanned a total of 12-14 degrees of the user's visual angle. After collecting the data from the 4 participants, we analyzed the pursuit detection efficiency by tracing the ROC curves for each method. Figure 6 displays the curves obtained for a time window of 1.5s, considering the pursued target, the three avoided targets, and the observation of the nature photography.

True / false positive rates and accuracy for time windows of 0.5s and 1.0s were also computed, but we decided to use the 1.5s time window, that corresponds to the ROC curves presented in Figure 6, since it should reduce involuntary selection during the visual search for the desired target, and it also presented the highest accuracy with no false positives in the pilot test.

### B. Experiment

The experiment was conducted with 11 participants (6 females), all able-bodied, with normal or corrected-to-normal vision. They were either undergraduate or graduate students. Ages varied between 21 and 46 years old (mean: 30.5). Two of them also reported some short previous experiences with eye tracking devices, but they did not consider themselves active users or experts in any way.

The experiment consisted of 20 trials in which users were required to enter a 4-digit PIN by following the appropriate targets with their eyes. Each target was labelled with a pair of digits, ranging from 1 to 8, with no repetition. The required PIN was always the same in all sessions — "1-2-3-4" — but the two digits associated with a target were randomly assigned following a uniform distribution at the beginning of a trial and whenever a selection was triggered, allowing for different pursuit scenarios (e.g., following the same target more than once).

Participants were placed between 60 to 70 cm from the computer screen, as in the pilot test. They were also allowed to practice for three trials before the actual data collection procedure started so that they could get familiar to the interface and the task. Additionally, we encouraged participants to enter the expected and a wrong PIN number to demonstrate the corresponding application feedback for each case, as well as assert that the software was working as intended.

### C. Data collection

Eye-tracking data was provided by a Tobii 4C 90 Hz remote eye tracker. The data was streamed through a UDP socket to the PursuitPass app, which was written in Python, using the Pygame rendering library. Though the developed code is cross-platform, we restricted our tests to the Microsoft Windows 10 operating system due to eye tracker compatibility. The same setup was used in both the pilot test and the experiment. A desktop computer with an Intel i7-7700 CPU and a NVidia GTX 1070 GPU was used. The application interface had a $600\times600$ pixel window size, and was displayed in a 144 Hz full HD 22" monitor.

Although the pursuit detection methods are calibration-free (as is PursuitPass) and absolute gaze coordinates are not required to determine which target a person is pursuing (the pupil center coordinates in the eye image could be used instead of the on-screen gaze coordinates), the eye tracker employed in the experiment only provided the absolute on-screen gaze coordinates. As a consequence, a user calibration process was necessary in order to estimate parameters that mapped detected eye features into gaze points on screen. This was evidently a requirement from the proprietary tracking software, not from the pursuit detection technique.

To deal with this limitation while keeping the calibration-free characteristic of PursuitPass, a calibration procedure was

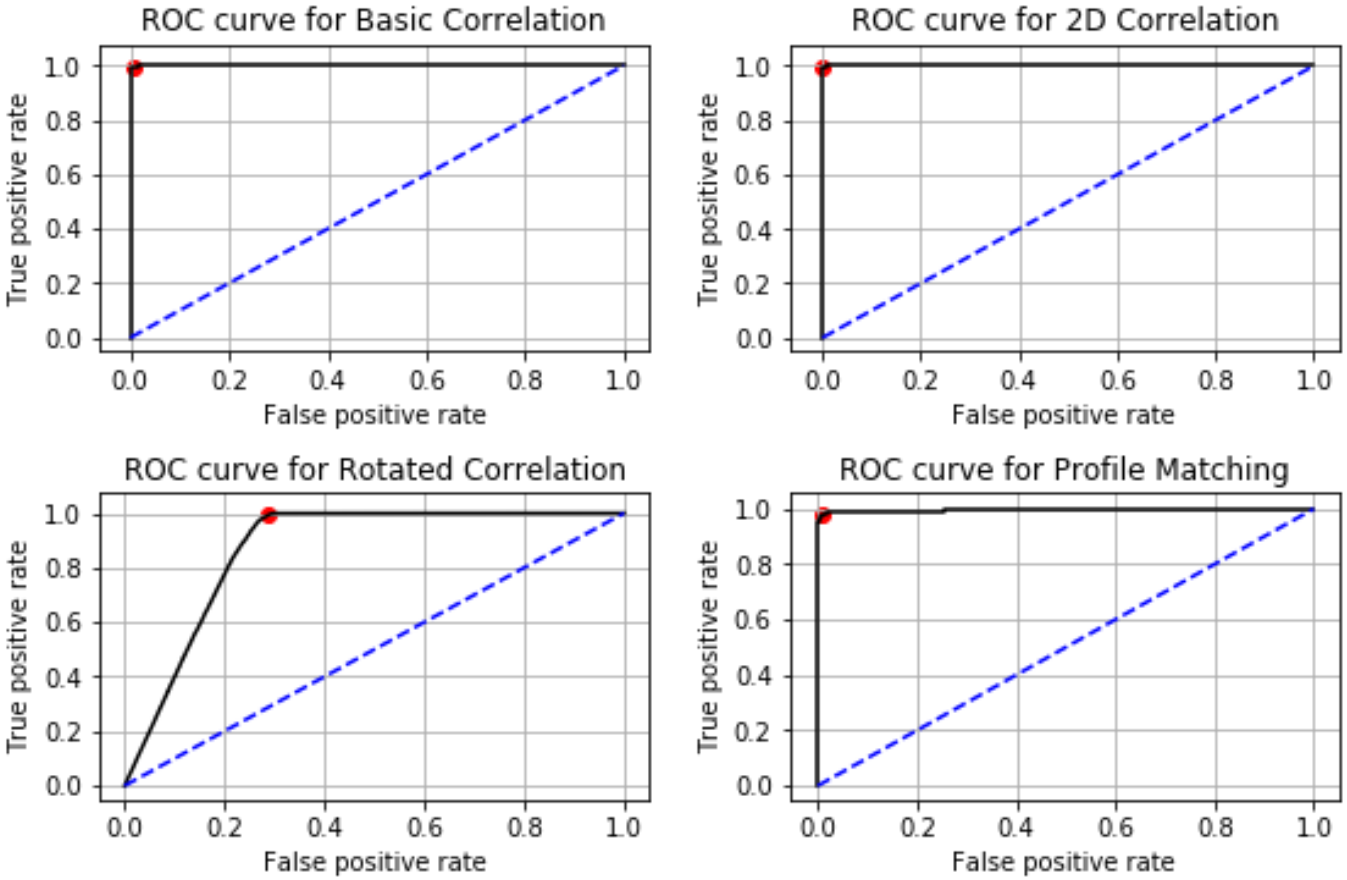


Fig. 6. ROC curves built to evaluate the methods employed by PursuitPass.

carried out just once with a *non-participant* in the experiment. This unique calibration profile, optimized for a specific person, was then used for all participants that took part in the experiment. Therefore, no prior user calibration was required from any participant. This evidently implies that accurate gaze prediction could not be guaranteed, but we opted for this approach to show that our application does not depend on any knowledge of absolute gaze coordinates, but rather its relative movement, rendering it in practice as "calibration-free".

We employed the best performing method observed in the pilot test to evaluate participant performance in the experiment in terms of error rate and average completion time over all methods detailed in Section III and with a time window of 1.5 seconds. Therefore, we gathered data about user digit input, PIN entry success rate, and elapsed time per trial. After the experiment, we also inquired participants concerning their experience with PursuitPass and whether they were aware of eventual input mistakes. This latter assessment was important to discriminate between user and system-related errors.

## V. Results

From the pilot test, we collected $4 \times 900$ samples of pursuit-pair points composed by $((eye_x, eye_y), (target_x, target_y))$, $3 \times 4 \times 900$ samples of non-pursuit-pair points $((eye_x, eye_y), (target_x, target_y))$ that corresponds to the avoided targets, and $4 \times 900$ samples of non-pursuit-pair points $((eye_x, eye_y), (target_x, target_y))$ that corresponds to the appreciation of the photography with an invisible virtual target used only to produce a similarity value with the methods detailed in Section III.

Each method was evaluated using the collected data to build ROC curves considering time windows of 0.5s, 1.0s and 1.5s (although just the curves considering a 1.5s time window are presented in Figure 6). In Figure 6, red dots on each curve are also shown, corresponding to the best threshold with respective false positive and true positive rates. These values are summarized in Table I. The best threshold for each method was defined as the most distant point in the ROC curve from the identity line.

| Method | False positive | True positive | Accuracy |
|---|---|---|---|
| Basic Correlation | 0.47% | 99.58% | 99.53% |
| 2D Correlation | 0.0% | 99.70% | 99.94% |
| Rotated Correlation | 28.89% | 99.48% | 76.78% |
| Profile Matching | 0.95% | 97.85% | 98.81% |

TABLE I
PERFORMANCE MEASUREMENTS FOR EACH METHOD USING A 1.5S TIME WINDOW.

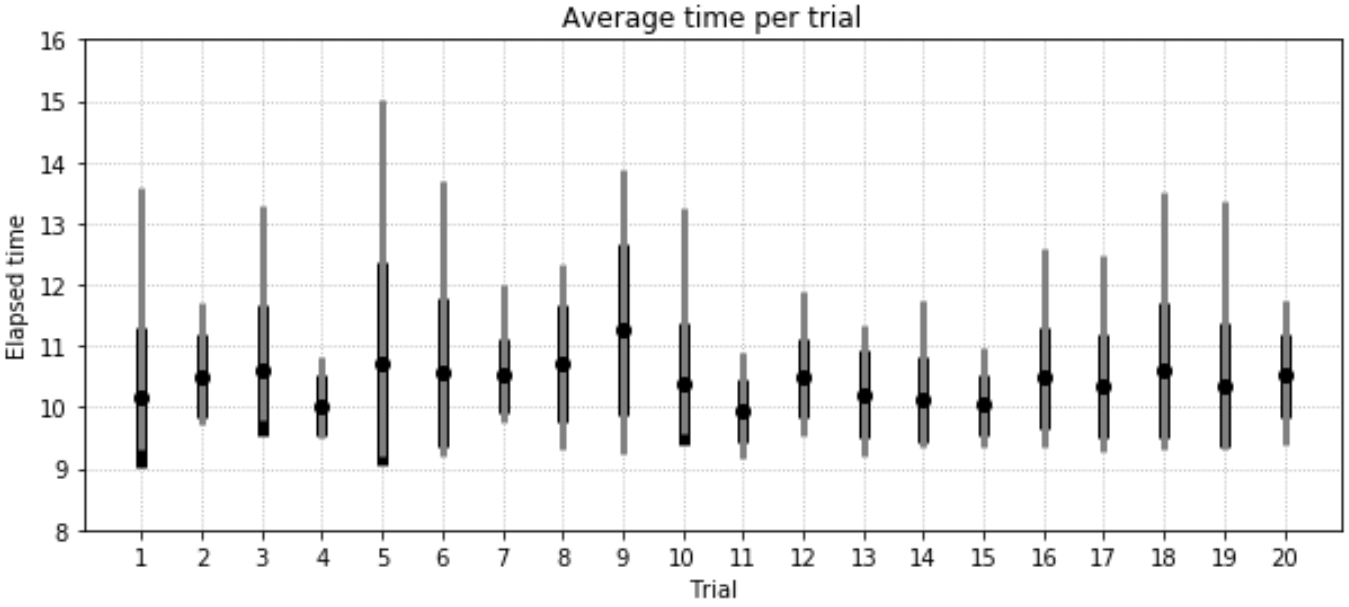


Fig. 7. Distribution of elapsed times for each trial to complete the PIN entry.

The false / true positive rates and accuracies for the 2D Correlation method that we obtained considering time windows of 0.5s and 1.0s, were 1%, 96.3%, 98.37%, 0.3%, 98.4%, and 99.24%, respectively. The false / true positive rates and accuracies for the Profile Matching method that we obtained with time windows of 0.5s and 1.0s, were 0.1%, 98.4%, 99.62%, 0.05%, 98.54%, and 99.65%, respectively. We decided to keep the larger time window to reduce the probability of involuntary selection occurrences during the search of the desired target by a naive user, although 2D Correlation had not produced false positives within the 1.5 time window.

The 2D Correlation method was selected for the experiment since it showed the best accuracy as well as robustness to false positives, which in turn meant that this method should produce the least amount of involuntary selections among the four tested methods.

For the experiment, 220 password entry attempts were collected, of which 7 have failed, rendering an accuracy of 96.82% for password entry. The average time to complete the password entry task was 10.42 seconds (mean SD = 0.886 seconds). Considering individual digits, a total of 880 digit entries we collected, of which 9 were wrong, meaning that PursuitPass achieved an accuracy of 98.98% per digit. Figure 7 presents the average elapsed time, as well as respective standard deviation and limits, for all trials. From Figure 7, it is also possible to observe that all PIN-entry trials took a time between 9.17 and 15.01 seconds to be performed.

## VI. Discussion

PursuitPass demonstrated the best trade-off so far between authentication completion time and accuracy in comparison to the state of the art, as shown in Table II. When compared to the fastest work in the literature [8], PursuitPass was only 0.8s slower on average, but much more accurate, with a gain of 5.68% over it. Compared to the approach with highest accuracy [7], our error rate was only 0.29% larger, but more than 13s faster in contrast. Observe that these results were achieved using 1.5s time windows. We expect that we could easily reduce the windows to 1s, saving therefore about 2s from the total time to enter a 4-digit PIN, without sacrificing much of the accuracy, since the 2D correlation method achieved a 98.4% accuracy using a 1s window in the pilot test, just 1% worse than using the 1.5s window that was chosen for the experiment. Alternatively, a 0.5s window would also be a viable choice if we chose the Profile Matching method since it achieved a 98.4% accuracy for this window size in the first experiment. Future investigation will reveal if other combinations of algorithms and window sizes would create even better performing authentication systems.

The interviews after the second experiment also provided us with some grounds to believe that our technique could be even more robust than already reported. Among participants who made an entry mistake, only one claimed to have entered the correct PIN despite being told otherwise by the application. All the others acknowledged to have selected wrong targets due to some distraction, and they would willingly have fixed those entries should they had been given the opportunity to do so. This means that PursuitPass could have achieved 99.55% of accuracy — possibly at the expense of a slightly longer average completion time — if an “undo” option had been implemented.

It should also be noted that because PursuitPass requires only 12-14 degrees of visual angle, it means that the application could be embedded in mobile or wearable devices

to be used at comfortable distances as an authentication mechanism, something that might not be possible with other related methods.

| Method | Pin | Time | Error rate | Calibration free |
|---|---|---|---|---|
| PursuitPass | 4 | 10.4s | 2.7% | yes |
| Look & Shoot [14] | 4 | 12s | 23.8% | no |
| Dwell-time [14] | 4 | 13s | 20.6% | no |
| Eye Gestures [14] | 4 | 54s | 9.5% | yes |
| Smooth Pursuit [7] | 4 | 25s | 2.4% | yes |
| Smooth Pursuit [8] | 5 | 9.6s | 8.4% | yes |

TABLE II
COMPARISON BETWEEN PURSUITPASS AND OTHER SIMILAR SYSTEMS.

## VII. CONCLUSION

In this paper we have introduced PursuitPass, a visual pursuit-based user authentication system designed to be suited for both public and personal small displays, such as ATM machines and mobile phones. Because it is based on visual pursuits, our method is calibration-free and robust to shoulder surfing, smudge, thermal, and acoustic attacks. We have included random features in the visual target behavior that ensures user liveness and improves the security of the technique against eye movement video recordings.

Another contribution of this paper is the evaluation of 4 different algorithms for pattern matching of the eye tracker data with the circular target trajectories. We have conducted an experiment to determine the best performing matching algorithm and configuration parameters for PursuitPass. We concluded that the 2D correlation method using 1.5 s windows provided the best accuracy, close to 99%. Using this method, a second experiment was conducted to evaluate the performance of PursuitPass. The results show that PursuitPass can achieve a 96.82% accuracy when entering a 4-digit PIN, with an average completion time of 10.4 s. These results improve the state of the art, since no other system presents, simultaneously, both high accuracy and fast authentication as PursuitPass does.

In a future work we will evaluate PursuitPass with more targets to enable more complex passwords, test the performance using shorter time windows, such as 0.5s or 1.0s, and investigate how accuracy might drop with higher detection speeds.

## ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors also would like to thank Prof. Eduardo Velloso for provinding support for the development of the algorithms evaluated in this paper, and the financial support from the São Paulo Research Foundation (FAPESP), grants 2016/10148-3 and 2015/26802-1, and the Google LARA program.

## REFERENCES

[1] A. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. Smith, "Smudge attacks on smartphone touch screens," in *Proc. of USENIX2010*. USENIX Association, 2010, p. 17.

[2] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd, "Reducing shoulder-surfing by using gaze-based password entry," in *Proceedings of the 3rd Symposium on Usable Privacy and Security*, ser. SOUPS '07. New York, NY, USA: ACM, 2007, pp. 13–19. [Online]. Available: http://doi.acm.org/10.1145/1280680.1280683

[3] C. Morimoto and M. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, 2005.

[4] J. Weaver, K. Mock, and B. Hoanca, "Gaze-based password authentication through automatic clustering of gaze points," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2011, pp. 2749–2754.

[5] V. Rajanna, S. Polsley, P. Taele, and T. Hammond, "A gaze gesture-based user authentication system to counter shoulder-surfing attacks," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '17. New York, NY, USA: ACM, 2017, pp. 1978–1986. [Online]. Available: http://doi.acm.org/10.1145/3027063.3053070

[6] A. De Luca, M. Denzel, and H. Hussmann, "Look into my eyes!: Can you guess my password?" in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ser. SOUPS '09. New York, NY, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1572532.1572542

[7] D. Cymek, A. Venjakob, S. Ruff, O. Lutz, S. Hofmann, and M. Roetting, "Entering pin codes by smooth pursuit eye movements," *Journal of Eye Movement Research*, vol. 7, pp. 1–11, 08 2014.

[8] D. Liu, B. Dong, X. Gao, and H. Wang, "Exploiting eye tracking for smartphone authentication," in *Applied Cryptography and Network Security*, T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, Eds. Cham: Springer International Publishing, 2015, pp. 457–477.

[9] M. Vidal, A. Bulling, and H. Gellersen, "Pursuits: Spontaneous interaction with displays based on smooth pursuit eye movement and moving targets," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13. New York, NY, USA: ACM, 2013, pp. 439–448. [Online]. Available: http://doi.acm.org/10.1145/2493432.2493477

[10] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen, "Orbits: Gaze interaction for smart watches using smooth pursuit eye movements," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 2015, pp. 457–466.

[11] M. Carter, E. Velloso, J. Downs, A. Sellen, K. O'Hara, and F. Vetere, "Pathsync: Multi-user gestural interaction with touchless rhythmic path mimicry," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 3415–3427. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858284

[12] E. Velloso, M. Carter, J. Newn, A. Esteves, C. Clarke, and H. Gellersen, "Motion correlation: Selecting objects by matching their movement," *ACM Trans. Comput.-Hum. Interact.*, vol. 24, no. 3, pp. 22:1–22:35, Apr. 2017. [Online]. Available: http://doi.acm.org.ezp.lib.unimelb.edu.au/10.1145/3064937

[13] E. Velloso, F. L. Coutinho, A. Kurauchi, and C. H. Morimoto, "Circular orbits detection for gaze interaction using 2d correlation and profile matching algorithms," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, 2018, p. 25.

[14] A. De Luca, R. Weiss, and H. Drewes, "Evaluation of eye-gaze interaction methods for security enhanced pin-entry," in *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, ser. OZCHI '07. New York, NY, USA: ACM, 2007, pp. 199–202. [Online]. Available: http://doi.acm.org/10.1145/1324892.1324932