Do it Yourself: Building a Low-Cost Iris Recognition System at Home Using Off-The-Shelf Components

Conference Paper · September 2017

CITATIONS
0

READS
4,107

5 authors, including:

Žiga Emeršič
University of Ljubljana
37 PUBLICATIONS **386** CITATIONS
SEE PROFILE

Blaž Meden
University of Ljubljana
19 PUBLICATIONS **111** CITATIONS
SEE PROFILE

Vitomir Štruc
University of Ljubljana
149 PUBLICATIONS **1,885** CITATIONS
SEE PROFILE

Peter Peer
University of Ljubljana
127 PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

Do it Yourself: Building a Low-Cost Iris Recognition System at Home Using Off-The-Shelf Components

Primož Lavrič[1], Žiga Emeršič[1], Blaž Meden[1], Vitomir Štruc[2], Peter Peer[1]

[1]*Faculty of Computer and Information Science*
[2]*Faculty of Electrical Engineering*
University of Ljubljana, Večna pot 113, Slovenia
E-mail: pl9506@student.uni-lj.si, {ziga.emersic, blaz.meden, peter.peer}@fri.uni-lj.si, vitomir.struc@fe.uni-lj.si

Abstract

Among the different biometric traits that can be used for person recognition, the human iris is generally considered to be among the most accurate. However, despite a plethora of desirable characteristics, iris recognition is not widely as widely used as competing biometric modalities likely due to the high cost of existing commercial iris-recognition systems. In this paper we contribute towards the availability of low-cost iris recognition systems and present a prototype system built using off-the-shelf components. We describe the prototype device, the pipeline used for iris recognition, evaluate the performance of our solution on a small in-house dataset and discuss directions for future work. The current version of our prototype includes complete hardware and software implementations and has a combined bill-of-materials of 110 €.

1 Introduction

The human iris is often described as one of the most suitable traits for biometric authentication due to several desirable characteristics. The iris has a complex texture that is unique for each individual and more importantly is also highly discriminative. As an internal organ it is well protected against injuries and is reasonably stable over time unlike other biometrics traits, which often experience significant age-induced changes, e.g., faces.

Despite the many advantages of the iris as a biometric trait, it still isn't used as widely in practice as fingerprints or facial images. While this can largely be attributed to intellectual-property rights pertaining to iris-recognition technology, we believe a significant factor here is also the cost of existing iris recognition systems. Commercial systems, such as the *BK-2121* produced by Iritech (Fairfax, VA, USA), the *I Scan 2* by Cross Match Technologies (Palm Beach Gardens, FL, USA) or the *3M CIS 202* by 3M (St. Paul, MN, USA) are commonly priced between 500$ and 700$, while low-end solutions such as the *IriShield MO-2120* by Iritech (Fairfax, VA, USA) may still weigh in at 190$ [1, 2, 3]. Despite the relatively wide range of prices, commercial iris-recognition systems are still less affordable than competing low-end systems based, for example, on fingerprints.

In this paper we describe our efforts on building a low-cost iris recognition system at home using off-the-shelf components. Our prototype, shown in Fig. 1, comprises complete hardware and software solutions and has a total bill-of-materials of only 110 €– making it competitive to low-cost iris recognition systems readily available on the market. The hardware part of our prototype comprises a self-devised base, a Raspberry PI 3, an eyepiece and a camera similar to fully-fledged iris recognition systems, such as [1, 2]. The software part of the prototype uses the segmentation, feature extraction and matching algorithms originally proposed by Daugman in [4]. While competing software solutions based on correlation filters [5, 6], Haar wavelets [7] or local binary patterns [8, 9] could also have been selected for this part, Daugman's approach was chosen, as it is well documented and one of the most widely used.

Figure 1: Illustration of our iris-recognition prototype. The system is build using off-the-shelf components and has a combined bill-of-materials of around 110 €.

Our experimental results obtained with the prototype system suggest that the prototype performs adequately for smaller groups of users and could be used as the basis for a low-cost consumer product. In the current form, it could also be deployed in conjunction with other authentication mechanisms, for example to access areas with lesser security requirements.

In summary, we make the following contributions in this paper:

- We describe a complete iris-recognition prototype (hardware and software) build using off-the-shelf components that can easily be replicated,
- We present a proof-of-concept evaluation of the developed prototype, and
- We discuss limitations of the current version of the system and provide directions for further research.

2 Prototype description

We now describe the developed prototype.

2.1 Hardware Setup

The hardware setup of our portable iris-recognition prototype shown in Fig. 1 consists of three main parts: the base, a height adjustable eyepiece (ocular) and a camera.

The base contains a single-board computer Raspberry PI 3 which controls the image acquisition procedure, performs the processing and conducts the authentication. It is powered by a $10Ah$ battery located next to the Raspberry PI. When deployed, the battery can also be replaced by an external source of electricity. The base container is made of plastic and is screwed onto a wooden board for additional stability. Additional holes for wires connecting the camera to the Raspberry PI for easier power supply access and HDMI output were also drilled into the base (a display is required for supervised user enrollment).

The height adjustable eyepiece allows the user to correctly adjust the distance between the ocular and the camera (mounted in a fixed position) in order to capture a clear and focused iris image. The main part of the eyepiece is made of a plastic height-adjustable furniture leg. Bent and welded together steel rods provide stability and sturdiness to the eyepiece and a soft ear pad allows users to comfortably position their eye on for image capture. When operational, the eyepiece is wrapped with a dark cloth to reduce the amount of light coming from external sources. It is important that the illumination of the iris can be controlled as precisely as possible to get the best contrast of the iris pattern and to always ensure uniformly illuminated images for the recognition pipeline.

The camera consist of two parts: an 8 mega-pixel focal Raspberry PI camera without the IR filter and a printed circuit board (PCB) with 8 infrared and 4 white LED lights attached to it. The PCB enables modifying the intensity of the infrared lights so the light intensity can be adjusted to provide enough illumination while not being harmful to the eye. We also tried to use white LEDs to briefly illuminate the eye and reduce the pupil dilation but the effects were minimal and the lights only distracted the user. During image acquisition the camera itself is positioned approximately 6–7 centimeters from the eye.

The Raspberry PI camera comes with the lens set to an infinity focus, which means that objects closer than 30 centimeters appear blurry. For this reason the lens distance from the sensor was adjusted and ensured a focus range between 5 and 8 centimeters.

This first implementation of the iris recognition device is relatively bulky and impractical to deploy. However, this is reasonable for a first-generation prototype. The size of the next prototype could be drastically reduced by 3D printing the enclosure and the eyepiece, making it better suitable for deployment. The device currently also captures only images of a single eye and could become more practical and also perform better if a second eyepiece was be added, allowing the device to simultaneously capture and perform recognition with both the left and the right iris.

The total bill-of-materials for the final prototype was approximately 110 € of which we used 50 € for the Raspberry PI 3, 30 € for the camera, 20 € for the infrared led module and the remaining 10 € for the construction materials. Clearly, with economies of scale this initial cost could easily be further reduced.

2.2 The Software Solution

While developing the software part of our prototype, a crucial prerequisite was computational efficiency because the targeted system (Raspberry PI) is a computationally weak device. With this in mind we developed an efficient recognition pipeline that is able to quickly process the input eye images and authorize or reject the users.

The developed iris recognition pipeline used in the prototype is based on Daugman's iris recognition pipeline model [10] with few minor modifications. The pipeline can operate in two modes: authentication and supervised enrollment mode. Both modes consist of six main steps: image acquisition, prepossessing, segmentation, normalization, encoding and matching or enrollment.

Acquisition: For the image acquisition, the user needs to lean over the device, place his/her eye against the eyepiece.and then start the capturing procedure either using the remote desktop (can be used for authentication and enrollment) or with the press of a button (can only be used for authentication). When the user initiates the capturing procedure, infrared LED lights are turned on and shortly after that the device starts acquiring images. When the user is correctly positioned and is not moving or blinking the capturing procedure takes approximately 3 seconds. However, the image capturing can take up to 15 seconds before an image of sufficient quality is captured. Each image is taken at the resolution of 8 mega-pixels and takes approximately 1 second to capture and 0.5 seconds to process. Currently the image is captured by an external process (command line tool Raspistill) written to file which is then read by the iris recognition pipeline.

Prepossessing: The prepossessing consists of *i)* initial transformations that crop and scale the input image to ensure consistent images for the recognition pipeline and *ii)* a procedure for specular reflection removal which aids the segmentation process.

With the first transformation step we want to crop out the window with the eye in the center (i.e., the region-of-interest - ROI) and scale it to a fixed size. In order to center the eye we need to locate the pupil. We do this by thresholding the image creating a mask of pixels whose intensity is below a threshold. We then locate all contours in the thresholded image and consider the center of the pupil contour as the center of the eye Here, we also calculate the minimal enclosing circle of the contour which we later use to limit the area considered for specular reflection removal. The masking and detection of the reflections is performed on a smaller image ($1/8$ of the original size) in order to increase the computational efficiency. The acquired circle is then scaled back to the original size. With the acquired location of the pupil

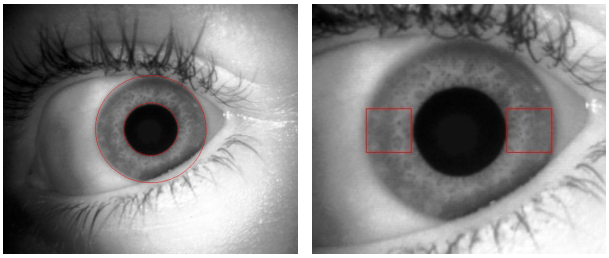

Figure 2: Examples of intermediate results in the iris recognition pipeline: detection of the iris boundaries with Daugman's integro-differential operator (left), square regions used for blurriness estimation (right).

center we then cut out an eye-centered window of size 1920×1680 pixels from the original image.

In the second step we remove the specular reflections from the transformed image. We do that by replacing all pixels in the pupil area (this is the only place where the specular reflections are present) whose intensity is above a predefined specular threshold with the average intensity of all pupil pixels below the specular threshold.

The centered and masked image is then down-sampled to the final size of 640×560 pixels. Before the image is passed to the segmentation part of the pipeline we perform contrast equalization. The contrast equalization makes the segmentation more reliable and also improves the quality of the extracted iris code. The preprocessing procedures used in our prototype are based on [11].

Segmentation: For the iris segmentation we use Daugman's integro-differential operator to locate the circular shaped iris [11]. The integro-differential operator is defined as

$$max_{(r,x_0,y_0)}|G_\sigma(r) * \frac{\delta}{\delta r}\oint_{r,x_0,y0}\frac{I(x,y)}{2\pi r}ds|, \qquad (1)$$

where $I(x,y)$ is the eye image, r is the radius to search for, $G_\sigma(R)$ is a Gaussian smoothing function and s is the contour of the circle given by r, x_0, y_0. The operator searches for a circle where there's a maximum change of pixel intensity (maximal gradient) by testing different circle parameters (r, x, y). The operator itself is applied iteratively, reducing smoothing in each iteration to obtain a precise location. The iterative approach also yields better performance as we only precisely search the area near the iris/pupil boundary. This algorithm is very reliable on the preprocessed images where the contrast has been equalized and the reflections were masked, with the masking improving results the most. There is still the possibility of imperfect segmentation but this could be addressed by using for example class (subject) specific dictionaries [12]. The segmentation could also be further improved by implementing eyelid and eyelash masking. The result of segmentation performed on a preprocessed image is shown on the left side of Fig. 2.

Quality evaluation: After segmentation we evaluate the quality of the image. Quality evaluation is performed in two steps:

- In the first step we check if the segmentation was successful. This is done by checking if the iris boundary circles are within expected bounds. The difference between the detected centers of the iris to the pupil and the iris to the sclera-boundary circles must not exceed a predefined threshold. In addition the difference of the radii of the two detected boundary circles must be within a predefined range. If the difference is lower than a certain threshold the iris is either too contracted or the segmentation failed. The latter also applies if the difference above a specific threshold.
- In the second step we evaluate the blurriness of the iris. This is done by calculating the variance of the Laplacian [13] in the two square regions on each side of the iris (as shown in Fig. 2 right). Because the regions are located in the vertical center of the eye they are least likely to be occluded by the eyelid or the eyelashes. The average variance of both regions gives a good estimate of the blurriness of the iris (higher the variance sharper the iris is). By thresholding the variance we can assure that only sharp images are accepted.

Figure 3: The normalized iris image (top) and the final iris code (bottom).

Normalization and iris encoding: To compute the iris code we must first transform the iris to the polar coordinate system and map it to a frame whose size is determined by the sample rate of Θ and r. To address the difference in resolution we use linear interpolation. The normalization minimizes the effect of iris dilation and makes subsequent processing and iris matching easier and faster.

After we normalize the iris image we generate the iris code using Log-Gabor filtering [14]. We apply the filter to the frequency-domain representation of the normalized iris images and then quantize the phase to obtain the iris code. The used Log-Gabor filter is defined as: $G(f) = exp(\frac{-(log(w*f_0)^2}{2(log(\sigma))^2})$ where w represents the wavelength (we use 18 units), f_0 represents the center frequency vector (we used $[0, 0.5]$ with a step of $\frac{N_\Theta}{2}$ where N_Θ represents the number of θ samples in the normalized image and σ represents the parameter that affects the bandwidth of the filter (we used 0.5).

The entire process of generating the iris code can be described as follows. First we transform each row of the normalized iris to the frequency domain using FFT (Fast Fourier Transform). We convolve the constructed Log-Gabor filter with each row and then apply the inverse FFT to transform the filtered images back into the spatial domain. With this process we obtain the phase which we then quantize in four different levels (one for each quadrant of the complex plane) represented by two bits. We end up with a binary matrix of size $N_r \times (N_\Theta * 2)$ that is used as the actual iris code. The normalized iris and final iris code are shown in Fig. 3.

Matching and enrollment: For the image enrollment, the user needs to manually evaluate the quality of the iris image and segmentation. The image of the segmented iris and the quality parameters mentioned in the

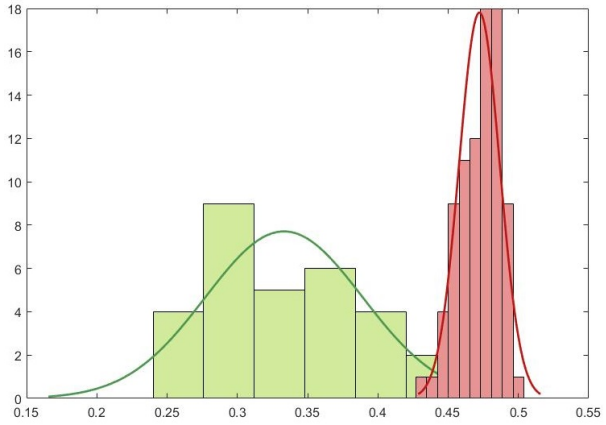

Figure 4: Genuine (green) and impostor (red) Hamming distance distributions of our experiments.

quality evaluation subsection are provided to the user who can then decide if the image is of adequate quality for enrollment. In a potential future implementation this could be completely automatized by improving the quality evaluation of the image.

In the case of authentication the acquired iris code needs to be compared with the entries stored in the prototype's database. The match score is defined as the Hamming distance between two (compared) iris codes. To achieve rotation invariance one of the iris codes is shifted and the best match is considered for the final score. Our prototype shifts up to 16 times in each direction.

3 Experiments and Results

We tested our prototype on a group of 10 subjects. For each subject we recorded 3 images of the right eye, each captured in a different session using the procedure described in the previous section (using the first image with adequate quality). We tested the pipeline's performance and the device's ability to capture images of adequate quality for iris recognition by calculating the matching score (Hamming distance) for genuine (matching) and impostor (non-matching) subjects. For the genuine scores we calculated matching scores for three combinations of image pairs for each subject and for the impostor scores we calculated scores for all pairs of non-matching images taken in the same session (session 1, 2 and 3). The matching-score distributions are shown in Fig. 4. The distributions were generated based on 30 genuine scores and 84 impostor scores.

From the graph we can see that the impostor and genuine distributions are reasonably separated but there is still some overlap. This can mainly be attributed to the lack of eyelid and eyelash segmentation in our pipeline as all of the genuine samples with a Hamming distance higher than 0.37 exhibited moderate eyelid overlap. For the current implementation the reasonable Hamming distance acceptance criterion could range from 0.40 to 0.42.

With an Equal Error Rate (EER) below 2% the results we obtained are satisfactory considering the fact that this is our first iris-recognition prototype. To improve the results further, a better scoring method could be employed that would focus more on important areas of the iris [15]. Further improvements could also be made to the hardware part of our prototype and to the recognition pipeline which would further improve the recognition performance.

4 Conclusion

We have presented an initial iris-recognition prototype built using off-the-shelf components. The results of our tests suggest that the prototype is capable of performing iris recognition with reasonable performance for small to moderately sized groups.With a total bill-of-materials of around 110 €, our prototype provides an affordable alternative to more expensive commercial iris recognition systems. As part of our future work, we plan to develop a second, 3D-printed prototype that is smaller, more practical, and would employ two cameras that would allow to perform authentication using both irises. Further improvements need to be made to the pipeline, such as improved segmentation, quality enhancement and a better scoring method.

References

[1] The Iritech homepage, (Accessed on 07/15/2017). [Online]. Available: http://www.iritech.com/

[2] The Crossmatch technologies homepage, (Accessed on 07/14/2017). [Online]. Available: https://www.crossmatch.com/

[3] The 3M homepage, (Accessed on 07/14/2017). [Online]. Available: http://www.3m.com/

[4] J. Daugman, "How iris recognition works," *IEEE TCSVT*, vol. 14, no. 1, pp. 21–30, 2004.

[5] N. Boddeti and B. V. Kumar, "Extended depth of field iris recognition with correlation filters," in *BTAS*. IEEE, 2008, pp. 1–8.

[6] B. V. Kumar, J. Thornton, M. Savvides, V. N. Boddeti, and J. M. Smereka, "Application of correlation filters for iris recognition," in *Handbook of iris recognition*. Springer, 2016, pp. 211–228.

[7] A. Kumar and A. Passi, "Comparison and combination of iris matchers for reliable personal authentication," *PR*, vol. 43, no. 3, pp. 1016–1026, 2010.

[8] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, "Local binary patterns and its application to facial image analysis: a survey," *IEEE TCMC - Part C*, vol. 41, no. 6, pp. 765–781, 2011.

[9] T. Tan, Z. He, and Z. Sun, "Efficient and robust segmentation of noisy iris images for non-cooperative iris recognition," *IVC*, vol. 28, no. 2, pp. 223–230, 2010.

[10] J. Daugman, "Iris encoding and recognition using gabor wavelets," *Encyclopedia of Biometrics*, pp. 973–983, 2015.

[11] ——, "Information theory and the iriscode," *IEEE TIFS*, vol. 11, no. 2, pp. 400–409, 2016.

[12] I. Naseem, A. Aleem, R. Togneri, and M. Bennamoun, "Iris recognition using class-specific dictionaries," *Computers & Electrical Engineering*, 2016.

[13] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," in *ICPR*, vol. 3. IEEE, 2000, pp. 314–317.

[14] P. Yao, J. Li, X. Ye, Z. Zhuang, and B. Li, "Iris recognition algorithm using modified log-gabor filters," in *ICPR*, vol. 4. IEEE, 2006, pp. 461–464.

[15] H. Rai and A. Yadav, "Iris recognition using combined support vector machine and hamming distance approach," *ESWA*, vol. 41, no. 2, pp. 588–593, 2014.