Face recognition with Raspberry Pi for IoT Environments

Conference Paper · September 2017

CITATIONS
2

READS
6,133

5 authors, including:

Rok Novosel
University of Ljubljana
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Blaž Meden
University of Ljubljana
17 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Žiga Emeršič
University of Ljubljana
31 PUBLICATIONS **182** CITATIONS
SEE PROFILE

Vitomir Štruc
University of Ljubljana
138 PUBLICATIONS **1,267** CITATIONS
SEE PROFILE

Face recognition with Raspberry Pi for IoT Environments

Rok Novosel[1], Blaž Meden[1], Žiga Emeršič[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: rn0450@student.uni-lj.si, {blaz.meden, ziga.emersic, peter.peer}@fri.uni-lj.si, vitomir.struc@fe.uni-lj.si

Abstract

IoT has seen steady growth over recent years – smart home appliances, smart personal gear, personal assistants and many more. The same is true for the field of biometrics where the need for automatic and secure recognition schemes have spurred the development of fingerprint- and face-recognition mechanisms found today in most smart phones and similar hand-held devices. Devices used in the Internet of Things (IoT) are often low-powered with limited computational resources. This means that biometric recognition pipelines aimed at IoT need to be streamlined and as efficient as possible. Towards this end, we describe in this paper how image-based biometrics can be leveraged in an IoT environment using a Raspberry Pi. We present a proof-of-concept web-based information system, secured by a face-recognition procedure, that gives authorized users access to potentially sensitive information.

1 Introduction

With the advances made recently in the field of biometrics and automatic identity recognition, the need for lightweight and streamlined person-recognition schemes increased as well. These developments are reflected in the use of personal recognition schemes in many mobile devices that enable users to unlock their devices using, e.g., fingerprints or faces. Another prominent use of biometrics are personal assistants and smart-home driven applications, where adapting to a specific person is an integral part of the user-experience. However, with the latter the security requirements are not as strict as with login application for mobile devices that can potentially hold sensitive personal information. The recognition process in home-use needs to be quick and simple – the precision is not always the most important, as miss-identifying the user has for the most part no serious consequences.

In this paper, we present a low-cost and lightweight solution for face recognition suitable for deployment with a personal smart-home assistant. The goal of our work is to build a simple interface which is capable of recognizing users automatically without inconvenient authentication protocols based, for example, on fingerprint scanning or passwords. The idea here is to use an automatic recognition procedure that uses facial images to recognize the user and then grant (or deny) him/her access to a certain service, application or network. To demonstrate the usefulness of the interface, we integrate it into a proof-of-concept web application developed based on the following requirements:

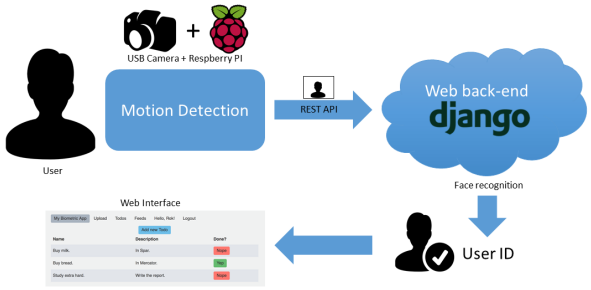


Figure 1: Schematic representation of the developed system: a USB camera attached to a Respberry Pi captures an image of the user and sends it to the web-server back-end for recognition. If the user is successfully recognized, he/she is granted access to a personal configurable web-service.

- *speech interaction*: the application should be (at least partially) voice-enabled for maximum user convenience and must allow for (as much as possible) touch-free operation,
- *efficiency and efficacy*: the application needs to be fast, secure (no local storing of sensitive biometric data), low-cost and simple (intuitive) to use,
- *biometrics-enabled*: the application should show user-specific content after a successful recognition attempt.

Based on the outlined requirements we implement our proof-of-concept system, as illustrated in Fig. 1. The system uses a Raspberry Pi and relies on a supporting web-server for face recognition and back-end processing. The communication with the user is based on text-to-speech synthesizers. A separate web interface enables the user to personalize the experience and set up services of interest: weather reports (status and forecast), personal to-do lists, RSS feeds. The text-to-speech synthesizer then reads the data to users. We believe this application a viable solution for a smart-home personal assistant interface and could represent the basis for potential commercial products.

The rest of the paper is structured as follows. In Section 2 we briefly review existing work related to our pa-

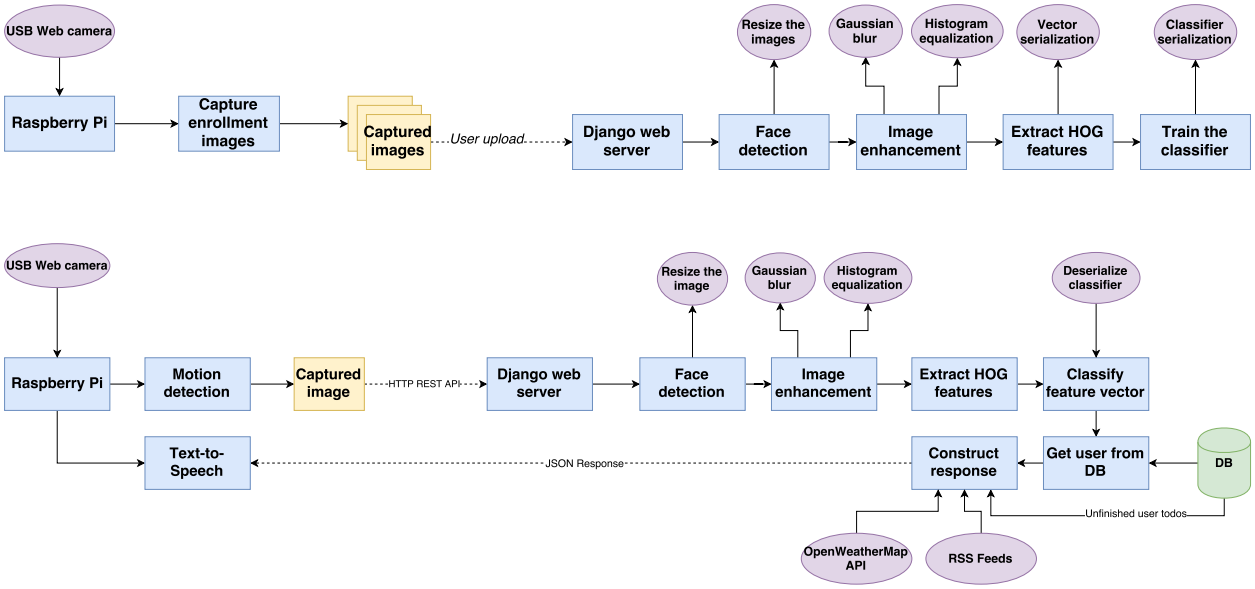


Figure 2: High-level overview of the developed system: (top) block diagram showing the individual components involved in the enrollment stage, (bottom) block diagram showing the individual components involved in the recognition stage

per. In Section 3 we describe the developed application and evaluate it in Section 4. We conclude the paper in Section 5.

2 Related Work

A significant body of existing work focuses on the use of biometrics for user authentication in IoT (Internet of Things) environments.

The closest to the work presented in this paper is the research in [1], where the authors implement a similar system to ours using a Raspberry Pi. The authors use a weighted Local Binary Pattern (LBP) algorithm to extract feature histograms and compare them using histogram intersection. The system is build in the C++ programming language with the OpenCV (`http://opencv.org`) library and runs entirely on the Raspberry Pi – but without a back-end part.

In [2] the authors develop a multi-biometric system again using a Raspberry Pi. The authors use the Azure cloud for cost reduction and greater computational performance. They also introduce several encryption techniques like RSA and enhanced AES-256 for greater security.

Last but not least, in [3] the authors implement a fingerprint recognition system entirely on the Raspberry Pi using PHP and PostgreSQL database.

3 System description

In this section we describe the developed system. We discuss implementation issues, the deployed face recognition procedure as well as other technical details.

3.1 Overview

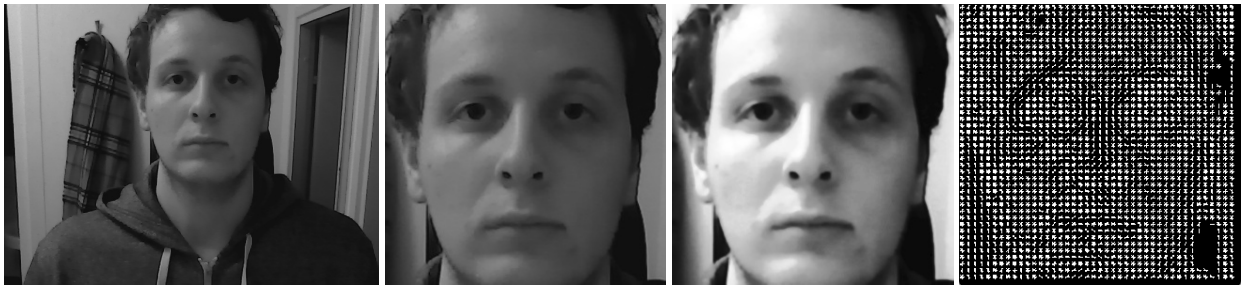
Our system comprises two separate parts: a *front-end* and a *back-end*. The front-end consists of a Raspberry Pi running OpenCV with a USB web camera and the back-end consists of server running the Django web framework (`https://www.djangoproject.com`) and again exploiting the OpenCV library. The front- and back-end communicate through a REST [4] API exposed by the back-end server. The front-end takes care of motion detection and image capturing while the back-end serves the web application where users register and enroll into the system. Figure 1 depicts a high-level overview of the system.

3.2 The Face Recognition Procedure

The main component of our system is a face recognition procedure that identifies the user based on the captured facial image. In general, biometric identification consists of two distinct phases: enrollment and recognition [5]. During the enrollment phase we acquire biometric data, process it and then store it in the system's database along with the users identity. During the recognition phase we again acquire the biometric data of the user and compare it against the stored data to determine the users identity. In the following sections we describe the implementation of both phases in our system.

Enrollment: An overview of the enrollment procedure is shown at the top part of Fig. 2. Users acquire enrollment images with a provided script, which by default, captures 10 images from the USB web camera on the Raspberry Pi. Users then upload the images to the web application running on a back-end server where they are further processed.

For the enrollment procedure, face detection is first performed on the captured images to exclude the image background from the processing and focus solely on the the region of interest. For the detection step OpenCV's implementation of the Viola-Jones algorithm [6] is used, but this could be implemented in the future with contem-



(a) Input image (b) Detected face (c) Enhanced image (d) HOG features

Figure 3: Visualization of the individual steps of our processing pipeline: (a) the input image that form the basis for the recognition procedure, (b) the region-of-interest (ROI) after face detection, (c) the enhanced image after Gaussian blurring and histogram equalization, and (d) vizualization of the computed HOG features.

porary techniques based, for example, on Convolutional Neural Networks (CNNs) [7]. Once the face region is detected, it is resized to predefined width and height, i.e., 200×200 pixels.

Next, the resized image is enhanced by applying Gaussian blurring and histogram equalization. This step is needed to reduce noise and mitigate potential effect of the external lighting. Each image is represented as a 1D feature vector using Histogram of Oriented Gradients (HOG) [8] (implemented in the Scikit-image [9] library). The feature vectors of all enrollment images are collected in a 2D matrix and stored in the system's database.

For each newly enrolled user the Support Vector Machine classifier (SVM) [10] used for identity inference in our system needs to be re-trained. Towards this end the feature vectors off all enrolled users are loaded and a multi-class SVM is trained in a one-against-all training regime.

Recognition: An overview of the recognition procedure is shown at the bottom part of Fig. 2.

During the recognition phase, motion detection is employed to detect whether the user would like to start using the system. When a sufficient level of motion is detected, the frame that triggers the recognition procedure is captured. Each frame is preprocessed by converting it to gray-scale and applying Gaussian blur to remove high frequency noise so we can focus on the structural objects of the image.

For the motion detection, we start by initializing the average frame as the first frame of the captured video stream. As the stream continues we compute the weighted average of all previous frames including the current one. With this procedure, we dynamically adjust to the background and account for the changes in lightning. We refer to this average as our *background model*. Next, we subtract the current frame from our background model leaving us with the the *frame delta*. We then apply thresholding to the calculated delta and find regions that are substantially different from the background model. If the regions are large enough, we assume that we have found motion in the current frame. We send this frame to the back-end for recognition. An example of an input frame is shown in Figure 3 (a).

On the back-end we convert the input frame to a feature vector using the same steps as in the enrollment phase. We classify the feature vector with our pretrained SVM model and get the users primary key (ID, label) as the result. With the primary key we query the database for additional information specific to the recognized user that can then be displayed or examined.

On the server we first segment the image by using face detection and then again resize the facial area to a predefined size. A sample detection result is shown in Figure 3 (b). In the enhancement step we apply Gaussian blurring in order to reduce the noise and histogram equalization to minimize illumination variations. A sample image after these two operations is shown in Figure 3 (c). The final step before classification is to represent the enhanced image with HOG features. A visualization of the computed HOG features is shown in Figure 3 (d).

3.3 Feedback to the user

Once the image is classified, the response for the front-end is constructed. While any personalized information could be returned at this point, we query in this proof-of-concept system the OpenWeatherMap API (`https://openweathermap.org/`) for weather data, users' RSS feeds and users' list of unfinished "TODOs". We then build the response and return it in JSON format to the front-end. An example of the response is shown in Listing 1. The response is read on the front-end by text-to-speech software running on the Respberry Pi.

4 Evaluation & Results

In this section we evaluate the performance of our face recognition procedure to estimate the level of security our system offers to it's users.

We evaluate our classifier on the ORL face database [11] that contains 10 different images of 40 distinct subjects. We train our SVM classifier on 7 images from each subject and test on the remaining 3 images. This setup gives us 280 images in the training set and 120 images in the testing set.

We experiment with multiple HOG parameters such as number of orientations the HOG features are quantized to or pixels per cell. Cells per block are fixed to 1. We

Listing 1: JSON returned by the back-end server

```
{
    "person": "Rok",
    "feeds": "Here are the latest news from
        your feeds: . The latest titles on BBC
        Sport are: #1: Pep Guardiola:
        Manchester City boss clarifies
        retirement comments. #2: Saints captain
         Fonte hands in transfer request. #3:
        Leicester City sign midfielder Ndidi
        for 31.5m.. The latest titles on Hacker
         News are: #1: I've removed all ad
        network code from my blog. #2: Yubikey
        with USB-C. #3: 105-Year-Old Cyclist
        Rides 14 Miles in an Hour En Route to a
         World Record.",
    "weather": "The temperature outside is -1.1
        1 degrees Celsius. I would describe the
         weather as broken clouds.",
    "todos": "You have the following things to
        do: #1: Buy milk.. #2: Study extra hard
        .."
}
```

Table 1: Results of the classifier evaluation show that the best configuration for our setup is 34 orientations, 7 pixels per cell.

No. of orientations	Pixels per cell	Recog. accuracy [%]
39	6	92.50
43	7	93.33
38	7	94.16
34	7	95.00

measure the accuracy of our classifier by calculating the percentage of correctly classified faces. The results of the best performing parameter combinations are shown in Table 1. The combinations are selected empirically and only the top performing parameters are shown here.

The results show that with the 95% of correctly classified subjects the best configuration for our setup is 34 orientations, 7 pixels per cell with the cells per block fixed to 1. This configuration outperforms the second best configuration with 38 orientations and 94.16% accuracy. However, the goal of the tests was not to achieve state-of-the-art or to outperform it, but to get reasonable results that show the feasiblitiy of our system. The 95% recognition rate on the ORL dataset is, we believe, a good starting point for future work that could make our system suitable for home use in a real-life scenario similar to the one presented in this work.

5 Conclusion

We implemented a biometric system using a Raspberry Pi and a web server. Images are captured based on motion detection and sent to the server where the are subjected to a face recognition procedure Our experiments suggest that the developed recognition procedure is able to achieve a 95% recognition accuracy on the ORL dataset. This shows the feasibility of the system as a low-cost component for home automation.

However, there is still a lot of room for improvement. The system could be improves by introducing an additional modalities like fingerprints or irises. This would add some additional complexity to the system, but would also contribute to a higher level of security. The system could also be extended with voice controls and GPIOs.

References

[1] O. Nikisins, R. Fuksis, A. Kadikis, and M. Greitans, "Face recognition system on raspberry pi," *Proc. of ICIPCE*, vol. 15, 2015.

[2] D. Shah and V. haradi, "Iot based biometrics implementation on raspberry pi," *Procedia Computer Science*, vol. 79, pp. 328 – 336, 2016.

[3] L. Arunkumar and A. A. Raja, "Biometrics authentication using raspberry pi."

[4] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[5] A. Jain, A. A. Ross, and K. Nandakumar, *Introduction to biometrics*. Springer Science & Business Media, 2011.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[7] I. Masi, S. Rawls, G. Medioni, and P. Natarajan, "Pose-aware face recognition in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4838–4846.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[9] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: http://dx.doi.org/10.7717/peerj.453

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[11] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. IEEE, 1994, pp. 138–142.