Real-time Face Recognition

Utilization of face recognition algorithms in real-time environment

Technical Report - FIT - VG20102015006 - 2015 – 03

Ing. Štěpán Mráček

Faculty of Information Technology, Brno University of Technology

July 20, 2015

Abstract

This report describes practical aspects of implementing and evaluating real-time face recognition biometric system. We propose a modular face recognition pipeline consisting of individual modules that deal with specific tasks, such are face detection, face landmarks detection, face pose orientation, feature extraction and template comparison. We describe each of those components and provide overview of available solutions and how they can be incorporated to the recognition pipeline and further tailored to our needs.

Contents

1 Introduction

Face recognition involves several consecutive tasks. The first goal is to select a region containing face in a given input image. The input of this first task is a frame from the ongoing video stream and the output is a set $R = \{r_1, r_2, \ldots, r_3\}$ of regions r_i in the image that contain a face. This task is described in Section 2.

If the set R is non-empty, the frame with annotated face regions is further processed with pre-processing part of the recognition pipeline. Here, the face pose (head rotation) is normalized and improper lighting conditions may be compensated. This topic is discussed in Section 3.

The normalized face image is further processed in such way that the most distinguishing features are extracted and the image is represented as a tuple of numerical values (*feature vector*). The process of gaining feature vector from the face image is described in Section 4. Feature vectors can be easily compared using an arbitrary metrics and thus we can say whether two vectors belong to the same person or not.

However, we are not relying on just one feature extraction algorithm. Some form of fusion that can incorporate comparison results from several algorithms is thus needed. Section 5 is devoted to this topic. The flow diagram of the process described above is in Figure 1.

Last to sections describe the implementation of the face recognition algorithm utilizing modern multi-core CPUs and possible extension to incorporate 3D sensors.

2 Face detection

Face detection is the first task in the real-time face recognition problem. We need a fast and reliable method that can detect whether the input frame contains a face. And if so, *Region of Interest* (ROI) has to be selected. There are several options how to detect faces. Perhaps the most used is the cascade classifier [25] (See Figure 2). The sequence of classifiers is trained in such way that the initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation.

In this report, we compare two cascade classifiers utilizing Haar cascades [25] and *Local Binary Patterns* (LBP) classifier [14]. Both Haar and LBP classifiers are implemented in OpenCV[1] library. Furthermore, we tested detector utilizing *Histogram of Oriented Gradients* (HOG) with subsequent *Support Vector Machines* (SVM) classifier [8, 11] that is implemented in dlib computer vision library[2].

We compared number of false detections (false positives, #FP) and number of false non-detections (false negatives, #FN). These were evaluated manually

[1] http://opencv.org/
[2] http://dlib.net/

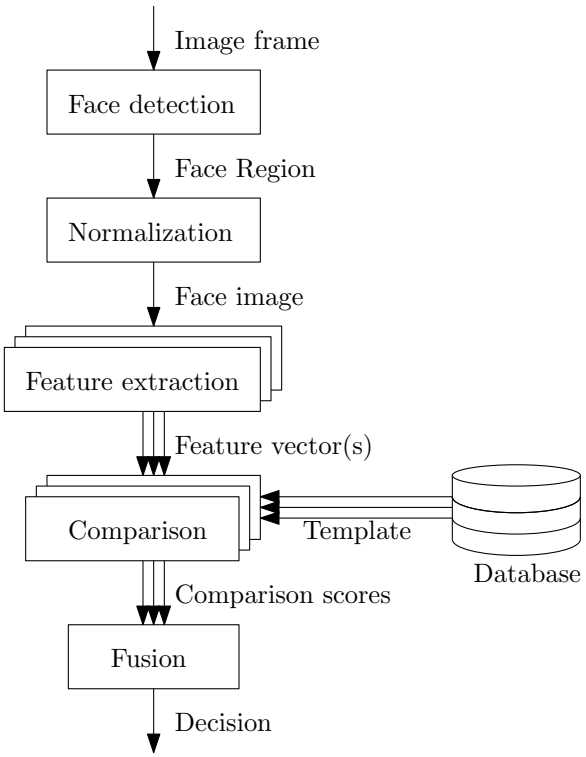

Figure 1: Flow diagram of the proposed face recognition algorithm

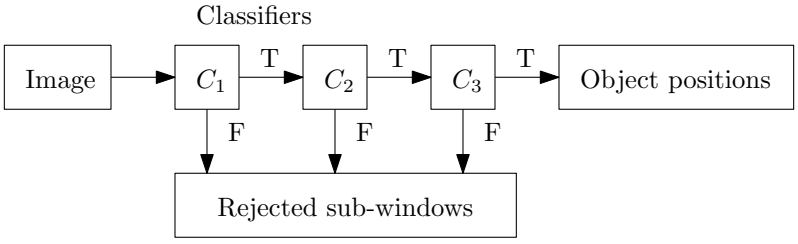

Figure 2: Cascade classifier

on set of 100 randomly selected face images from FRGC database [18] and set of 100 non-face images.

We measured speed of video frames processing on video with resolution 1280×720 pixels (column FPS_1 in Table 1). This video was also sub-sampled to resolutions 640×360 (FPS_2) and 320×180 (FPS_3). Time of processing was measured on notebook with Intel Core i7-4702MQ CPU and 8GB RAM.

Detector	Implementation	#FP	#FN	FPS_1	FPS_2	FPS_3
Haar-cascade	OpenCV	4	0	3.31	14.35	61.24
Haar-tree	OpenCV	1	3	5.24	19.25	54.60
LBP-cascade	OpenCV	3	4	13.70	50.11	163.78
HOG-SVM	Dlib	0	2	6.80	26.23	102.37

Table 1: Comparison of face detection algorithms

3 Face pose normalization

The acquired and selected face image should be normalized to some predefined pose. We can use the region marked by the face detector as the initial estimation of subsequent landmark detector. It will locate important face features such are eyes, nose and mouth. These landmarks are further use in order to rotate and scale face image, such the pose variation is minimized.

There are a lot of landmark detectors available. Most of them rely on some generative probability model of mutual landmark positions. Such landmark detectors iteratively deform the model in order to fit to a given input image. We tried several well known and established landmark detectors, see Table 2.

Detector	Implementation	Reference
Active Shape Models (ASM)	VOSM	Cootes 1999 [6]
Active Appearance Models (AAM)	VOSM	Cootes 2004 [7]
Ensemble of Regression Trees	Dlib	Kazemi 2014 [10]
Structured Output SVM	cLandmark	Uricar 2015

Table 2: Landmark detectors overview

The next step is the utliziation of detected landmarks. We use a simple heuristic approach – the image is rotated and scaled in such way, that the distance between eyes is 100 pixels and both eyes lay on the same y coordinate.

4 Feature extraction

Biometric recognition pipeline usually consists of data acquisition, preprocessing, feature extraction, and comparison that yields to the final decision whether the user is accepted or not [24].

In this section, the main emphasis will be put on the preprocessing and feature extraction parts. The data acquisition is given by the nature of face recognition, since we have to deal with consecutive video frames. The last part of the pipeline – comparison – will be more discussed in section 5.

A general preprocessing and feature extraction pipeline is in Figure 3. It has to be noted that all particular boxes that represents stages of the pipeline are optional. Image Filter can be omitted and features can be extracted directly. Or the dimensionality reduction may be applied directly on the input image. In the same manner, if the number of features provided by the image descriptor is adequate, dimensionality reduction is not needed.

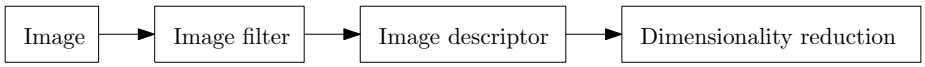

Figure 3: Generalized preprocessing and feature extraction pipeline

The list of image filters that are used in the area of face recognition is in Table 3. Literature references are provided for each filter. The resulting image with applied filter may be further processed by the image descriptor module or it can be directly send to dimensionality reduction block.

Table 3: List of image filters used in face recognition tasks

Name of filter	References
Gabor filter	Yang 2004 [26], Zheng 2007 [27]
Gauss-Laguerre filter	Ahmadi 2007 [1], Mráček 2014 [16]
Difference of Gaussians (DoG)	Tan 2010 [21]

The list of various image descriptors used in face recognition is in Table 4. We are again providing literature references for each image descriptor. There is a common practice to use image descriptors with histograms. The face image is divided into N $(i \times j)$ non-overlapping sub-images. The histogram of LBP values or orientations is calculated within each cell. Finaly, the individual histograms are concatenated and the feature vector is thus extracted.

Table 4: List of image descriptors used in face recognition tasks

Name of descriptor	References
Local Binary Patterns (LBP)	Ahonen 2004 [2], Tang 2013 [22]
Weber Local Descriptor (WLD)	Chen 2009 [5], Li 2013 [13]
Histogram of Oriented Gradients (HOG)	Deniz 2011 [9]

The final and also optional part of the feature extraction process is the dimensionality reduction. Feature vector obtained from the previous blocks can be projected using some dimensionality reduction technique in order to reduce redundancy, increase speed of comparison, reduce amount of data needed to store biometric templates or improve recognition performance.

Table 5: List of dimensionality reduction techniques

Projection	References
PCA	Turk 1991 [23]
LDA	Belhumeur 1997 [4]
ICA	Bartlett 2002 [3]

5 Score-level fusion

The feature extraction process described in previous section may provide various implementations. We can select different image filters, parameters of those filters, image descriptors and many dimensionality reduction techniques. We can also use various comparison metrics for extracted feature vectors in order to compare two input samples.

We can evaluate all possible combinations mentioned above and select the one that outperform others. The second option is to employ a score-level fusion [19, 20, 17]. A diagram of score level fusion approach is in Figure 4.

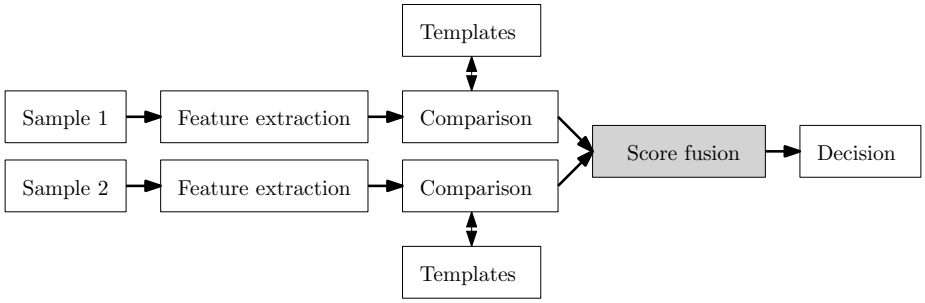

Figure 4: Diagram of score-level fusion

However, the number of possible combinations is very high, therefore some optimization selection method is needed in order to improve speed as well as remove redundancy. We employ hill-climbing selection and the optimization criterion is fusion *Equal Error Rate* (EER) [12]. Some examples of possible individual recognition pipelines (units) is in Figure 5.

The individual units were trained on the first 300 images from the Spring2004 portion of FRGC database. Next 300 samples were used for the training of unit selection and the remaining 1509 scans were used for evaluation. The selection of units in in Table 6. The following abbreviations are used in the descriptions of the pipelines:

- orientedGradients – oriented gradients
- histogramBins-x-y-z – divide the image to $x \times y$ grid and calculate histogram of values divided into z bins
- zpca – PCA projection with z-score normalization of projection components (also called whitened PCA)

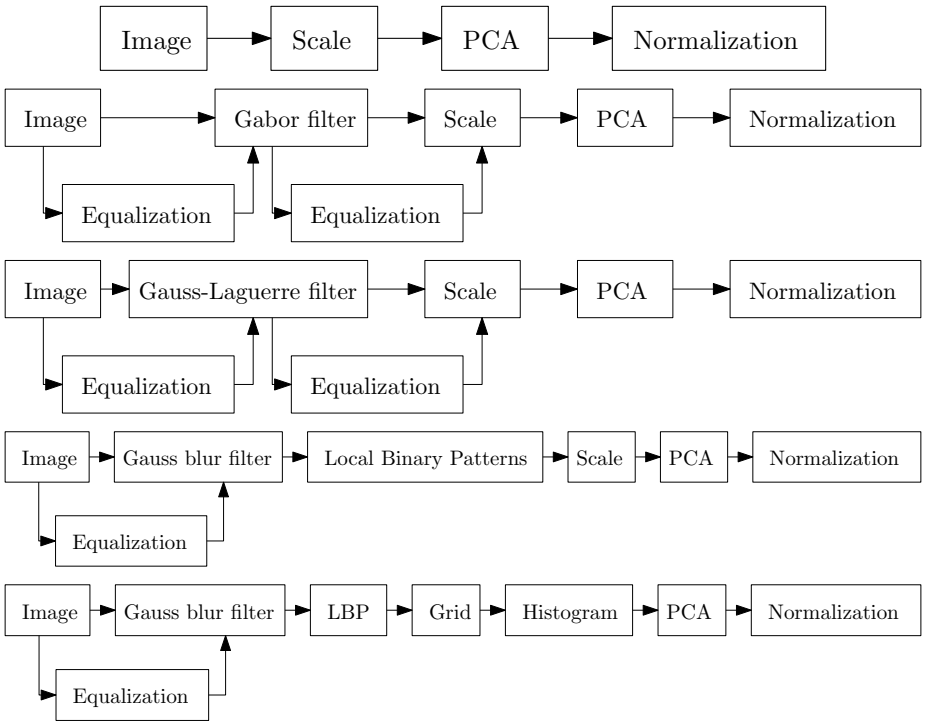

Figure 5: Visualisation of various feature extraction techniques

- dog-x-y – difference of Gaussians with kernel sizes x and y
- gaborAbs-x-y – absolute response on Gabor filter with scale x and orientation y
- scale-x – Scale the image with factor x
- equalize – equalization of histogram of pixel intensity values
- contrast – contrast enhancement [21]
- gaussLaguerreAbs-x-y – absolute response on Gauss-Laguerre filter with parameters x and y

Figure 6 shows evaluation on FRGC database. *Detection Error Tradeoff* (DET) curve that plots *False Match Rate* (FMR) against *False Non-match Rate* (FNMR) is shown [20].

6 Parallel implementation

If we want to have a real-time face recognition system, it should utilize modern multi-core CPUs. There are several points within the pipeline where parallel computation may be involved. Moreover, we can take advantage of processors

Table 6: Selected recognition units

	Feature extraction pipeline	Metric
1	orientedGradients;histogramBins-10-9-8;zpca	correlation
2	dog-5-3;contrast;gaborAbs-3-3;scale-0.5;zpca	correlation
3	orientedGradients;histogramBins-10-5-16;zpca	correlation
4	equalize;gaborAbs-6-1;scale-0.5;zpca	correlation
5	dog-5-3;contrast;gaborAbs-7-6;scale-0.5;zpca	correlation
6	gaborAbs-2-4;scale-0.5;zpca	correlation
7	dog-5-3;contrast;gaborAbs-7-3;scale-0.5;zpca	correlation
8	dog-5-3;contrast;scale-0.5;zpca	correlation
9	orientedGradients;histogramBins-10-5-16	correlation
10	equalize;gaborAbs-7-3;scale-0.5;zpca	correlation
11	dog-5-3;contrast;gaborAbs-5-1;scale-0.5;zpca	correlation
12	orientedGradients;histogramBins-5-9-4;zpca	correlation
13	gaborAbs-4-6;scale-0.5;zpca	correlation
14	dog-5-3;contrast;gaussLaguerreAbs-16-3-0;scale-0.5;zpca	correlation
15	gaborAbs-2-2;scale-0.5;zpca	correlation

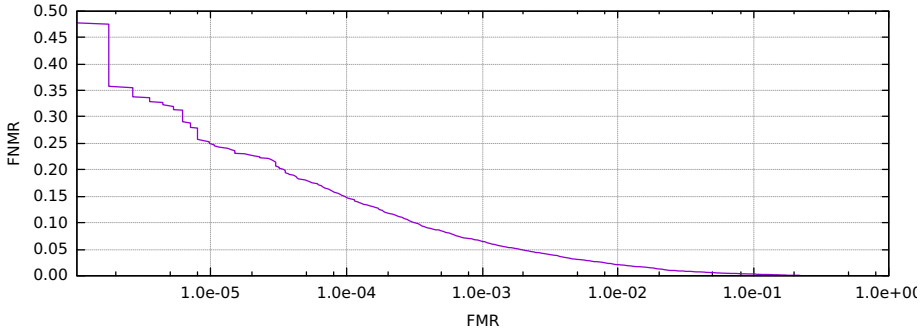

Figure 6: Evaluation on the FRGC databse

Table 7: Utilization of multi-core CPUs and GPUs for face detection

Library	Class	Multi-core CPU	OpenCL	CUDA
OpenCV	CascadeClassifier	Yes	Yes	Yes
Dlib	object_detector	No	No	No

on graphic cards (GPU). Table 7 shows the utilization of multi-core CPUs and GPUs for face detection for OpenCV and Dlib computer vision libraries.

The next part of the recognition pipeline where the parallelism can be introduced is the feature extraction. Each feature extraction unit from Table 6 is completely independent. Since the feature extraction takes approximately several milliseconds, it is not a good idea to create new processes or threads every

time. Much better solution is the thread pool. Threads are created when the biometric system is initialized and the feature extraction unit just query them when needed.

Table 8 shows the time consumption of individual components of the recognition pipeline. It was measured on notebook with Intel Core i7-4702MQ CPU and 8GB RAM. 1000 consecutive frames with resolution 640×480 were captured with build-in video camera. Frames were scaled to size 320×240 for the face detection component. In each frame, after the feature vector was extracted, a comparison with previously extracted 300 templates was made. It should be noted that the feature extraction portion is the most time consuming part. Here we gain the advantage of the parallel thread-pool-based computation. If there is no thread pool, the average time of feature extraction is 70 milliseconds. With thread-pool we gain 16 frames per seconds recognition rate with thread pool and only 10 frames per second without it.

Table 8: Single frame time consumption of individual components of the recognition pipeline

Component	Average duration (ms)
Face detection	14
Landmark detection	3
Face pose normalization	0
Feature extraction	29
Comparison and identification	17

The illustration of the demo application is in Figure 7. Users can enroll to the system. At the same time runs the identification loop.

7 Real-time 3D Face Recognition

3D face recognition is the natural extension of the classical 2D approach. It solves several disadvantages of 2D approach such are head rotation, poor lighting conditions and liveness detection. However, the usage of 3D sensors is limited – we can't use 3D sensors in direct sunlight. Table 9 brings an overview of tested low-cost depth sensors.

Table 9: Overview of low-cost depth sensors

Vendor	Sensor	Involved technology
Intel	RealSense	structured light
SoftKinetic	DepthSense DS325	time of flight
Occipital	Structure.io	structured light
Microsoft	Kinect 360	structured light

Since we are not relying on just a texture, we can incorporate 3D shape and curvature into the feature extraction. The processing is similar - 3D shape is

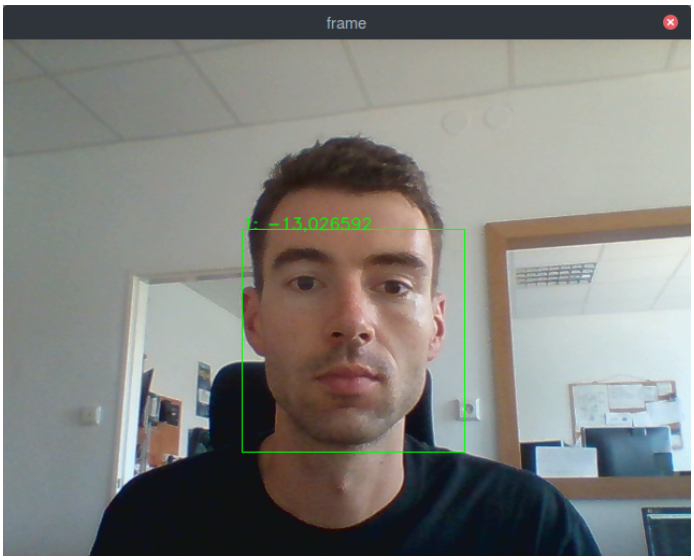

Figure 7: Illustration of positive identification in the demo application.

transformed to the image representation and further processed as it was a plain face image. Figure 8 shows various image representations of the 3D face model from the FRGC database.

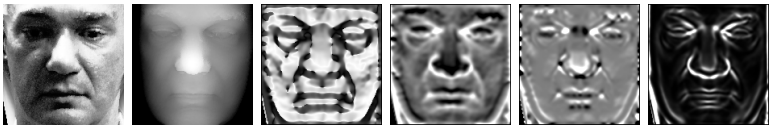

Figure 8: Various representations of the 3D face model. From left to right: texture, depth, shape index, mean curvature, Gaussian curvature, and eigen-curvature.

We tested Intel RealSense, SoftKinetic DepthSense DS325, and Microsoft Kinect 360. Figure 9 shows the evaluation on the database consisting of 400 scans and 42 subjects.

8 Conclusion

This reports described the real time face recognition system. The entire source code can be found at `https://github.com/stepanmracek/face`. More detailed description of the involved algorithms may be also found in our previous publications: [16, 15].

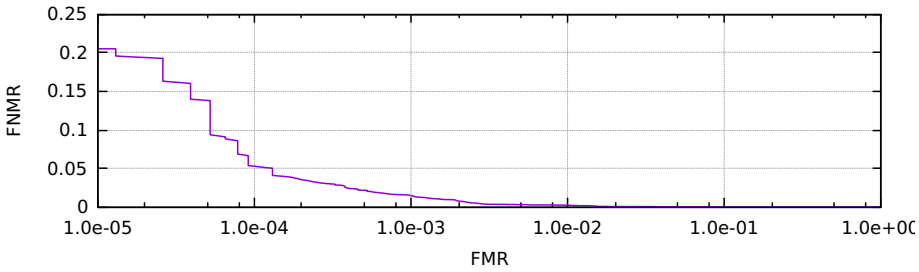

Figure 9: Evaluation of 3D face recognition system employing Intel RealSense camera.

References

[1] H. Ahmadi and A. Pousaberi. An Efficient Iris Coding Based on Gauss-Laguerre Wavelets. In *Advances in Biometrics*, pages 917–926. 2007.

[2] T. Ahonen, A. Hadid, and M. Pietikäinen. Face Recognition With Local Binary Patterns. In *8th European Conference on Computer Vision*, pages 469–481, 2004.

[3] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski. Face Recognition by Independent Component Analysis. *IEEE transactions on neural networks*, 13(6):1450–1464, 2002.

[4] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[5] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikäinen, Senior Member, Xilin Chen, and Wen Gao. WLD: A Robust Local Image Descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1705–1720, 2010.

[6] T. F. Cootes and C. J. Taylor. A Mixture Model for Representing Shape Variation. *Image and Vision Computing*, 17(8):567–573, 1999.

[7] T. F. Cootes and C. J. Taylor. Statistical Models of Appearance for Computer Vision. Technical report, 2004.

[8] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[9] O. Déniz, G. Bueno, J. Salido, and F. De La Torre. Face Recognition Using Histograms of Oriented Gradients. *Pattern Recognition Letters*, 32(12):1598–1603, 2011.

[10] V. Kazemi and S. Josephine. One Millisecond Face Alignment with an Ensemble of Regression Trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1867–1874, 2014.

[11] D. E. King. Dlib-ml : A Machine Learning Toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[12] R Kohavi. Wrappers for Feature Subset Selection. *Artificial intelligence*, 97(1):273–324, 1997.

[13] S. Li, D. Gong, and Y. Yuan. Face Recognition Using Weber Local Descriptors. *Neurocomputing*, 122:272–283, 2013.

[14] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Li. Learning Multi-scale Block Local Binary Patterns for Face Recognition. In *International Conference on Biometrics (ICB)*, pages 828–837, 2007.

[15] Š. Mráček, J. Váňa, R. Dvořák, M. Drahanský, and I. Provazník. 3D Face Recognition on Low-Cost Depth Sensors. In *Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG 2014)*, 2014.

[16] Š. Mráček, J. Váňa, K. Lankašová, M. Drahanský, and M. Doležel. 3D Face Recognition Based on the Hierarchical Score-Level Fusion Classifiers. In *Biometric and Surveillance Technology for Human and Activity Identification XI*, 2014.

[17] K. Nandakumar, Y. Chen, S. C. Dass, and A. K. Jain. Likelihood Ratio-Based Biometric Score Fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):342–347, February 2008.

[18] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the Face Recognition Grand Challenge. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 947–954. IEEE, 2005.

[19] L. Puente, M. J. Poza, B. Ruíz, and D. Carrero. Biometrical Fusion – Input Statistical Distribution. In G. Chetty and J. Yang, editors, *Advanced Biometric Technologies*, pages 87–110. InTech, 2011.

[20] A. Ross. An Introduction to Multibiometrics. In A. K. Jain, P. Flynn, and A. A. Ross, editors, *Handbook of Biometrics*, pages 271–292. Springer US, 2008.

[21] X. Tan and B. Triggs. Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions. *IEEE Transactions on Image Processing*, 19(6):1635–1650, 2010.

[22] H. Tang, B. Yin, Y. Sun, and Y. Hu. 3D Face Recognition Using Local Binary Patterns. *Signal Processing*, 93(8):2190–2198, 2013.

[23] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 591(1):586–591, 1991.

[24] C. Vielhauer, J. Dittmann, and S. Katzenbeisser. Security and Privacy in Biometrics. In P. Campisi, editor, *Security and Privacy in Biometrics*, pages 25–43. Springer London, London, 2013.

[25] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[26] P. Yang, S. Shan, W. Gao, S. Z. Li, and D. Zhang. Face Recognition Using Ada-Boosted Gabor Features. In *Proceeding of 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 356–361, 2004.

[27] Z. Zheng, F. Yang, W. Tan, J. Jia, and J. Yang. Gabor Feature-Based Face Recognition Using Supervised Locality Preserving Projection. *Signal Processing*, 87(10):2473–2483, October 2007.