www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

PAN POLSKA AKADEMIA NAUK

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

M&MS

LIVENESS MEASUREMENTS USING OPTICAL FLOW FOR BIOMETRIC PERSON AUTHENTICATION

Maciej Smiatacz

Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Department of Intelligent Interactive Systems, Narutowicza 11/12, 80-233 Gdansk, Poland (✉slowhand@eti.pg.gda.pl)

Abstract

Biometric identification systems, i.e. the systems that are able to recognize humans by analyzing their physiological or behavioral characteristics, have gained a lot of interest in recent years. They can be used to raise the security level in certain institutions or can be treated as a convenient replacement for PINs and passwords for regular users. Automatic face recognition is one of the most popular biometric technologies, widely used even by many low-end consumer devices such as netbooks. However, even the most accurate face identification algorithm would be useless if it could be cheated by presenting a photograph of a person instead of the real face. Therefore, the proper liveness measurement is extremely important. In this paper we present a method that differentiates between video sequences showing real persons and their photographs. First we calculate the optical flow of the face region using the Farnebäck algorithm. Then we convert the motion information into images and perform the initial data selection. Finally, we apply the Support Vector Machine to distinguish between real faces and photographs. The experimental results confirm that the proposed approach could be successfully applied in practice.

Keywords: liveness detection, biometrics, optical flow, face recognition.

1. Introduction

The main goal of every biometric system is to uniquely determine a person's identity based upon certain physiological features (such as the texture of iris) or particular behavioural characteristics (for example the way that someone moves while walking). In the early days biometrics was considered a part of the pattern recognition domain: persons were treated as classes, just as the alphanumerical symbols in OCR systems. Soon it appeared that in real life the task is not just to assign the input data to one of the templates representing known individuals. In practice we usually want to know if the biometric features are *similar enough* to the reference description of the given person. In other words, the user claims some identity and the system checks whether it is true or false. This is the case of biometric *verification* (or *authentication*) which is performed, for example, when someone tries to log in to the particular account of the operating system or inserts the credit card into the bank teller machine. This task is often much more difficult than the typical classification and it involves finding the trade-off between false acceptance rate (the security level) and the false rejection rate (the ease of use). Sometimes a biometric system works within a slightly different scenario, in which the goal is to perform the *identification* of a person: we may want, for example, to detect the presence of some known terrorist in the crowd at the airport. Again, this is not a regular classification problem. The detection rate must be kept as close to 100% as possible but a low level of false alarms is equally important.

During the last decade biometrics has become a field of very intensive research, mostly because of its potential practical advantages. A properly working biometric system would

Article history: received on Jan. 25, 2012; accepted on Apr. 23, 2012; available online on May 18, 2012; DOI: 10.2478/v10178-012-0022-y.

increase the level of security, because stealing a door key, a password or a PIN code would not be enough anymore for an intruder to break in. On the other hand, it would also make life much easier. We would not have to carry numerous smart cards and identifiers, remember dozens of passwords to web accounts; the machines would recognize us by who we are, not by what we have or what we remember.

Advertisements of biometric systems usually highlight the fact that the biometric characteristics cannot be lost or forgotten – it is just a part of us. Unfortunately, the truth is that at the same time it can be stolen easily. Of course, probability of identity theft depends on the type of biometric technology. It is possible to produce a special glove that would cheat the fingerprint scanner but it would take a lot of time and effort. The situation is different when it comes to face recognition: what if someone finds our photograph on the Internet, prints it out and puts in front of the camera? This way researchers managed to crack facial recognition technology in Lenovo, Toshiba and Asus laptops in 2009 [1]. This shows that even the most accurate and sophisticated face recognition system is useless if it cannot measure the *liveness* of the subject being identified. Therefore, a liveness detector that is able to distinguish between a real person and a photograph appears to be one of the most important modules of every biometric system relying on face image analysis.

During our work on SART-2, the biometric security system for mobile workstations [2] which uses face recognition technology, we tested some known algorithms for liveness measuring and we proposed our own method, based on optical flow. In the following sections we present the competing approaches and describe our solution in detail. Finally we report the results of experiments carried out on 100 video sequences.

2. Related work

Although liveness detection is a relatively new topic, many attempts to solve the problem have already been described in the literature. In this section we briefly review some of the most important approaches.

One of the simplest methods of liveness measurement was proposed in [3]. It is based on the observation that the binarized (black and white) images of the eye region, extracted from a video sequence, reveal high variation when the clip shows a real person, while in the case of a photograph the eye shape remains almost constant across the frames. The eyes are first detected using AdaBoost algorithm [4], then a normalization with Self Quotient Image [5] is performed. The latter is very important because otherwise the changes of illumination could severely disturb the binarization process. The analyzed image fragments have the form of 10×20 rectangles positioned at the eye centres. A sequence of 5 such rectangles is extracted for the left and right eye. The rectangles are converted to vectors containing binary values and then the Hamming distance d_H between every two vectors is calculated. If the mean value of d_H is higher than a certain threshold, liveness is detected. Although authors of [3] reported very good results, our experience shows that this method could work only on high quality images. Otherwise, the fluctuations of image intensity usually introduced by webcams lead to the detection of eye shape changes even when still photographs are used. Moreover, the ideal detection of eye centres is very problematic in practice. Finally, the method will fail (i.e. liveness will be erroneously detected) if the photograph is bent or rotated out of plane (along the vertical or left-right axis).

Another group of methods focuses on the eye-blink detection. An advanced algorithm of this kind was described in [6]. Authors introduced a real-valued feature $U(I)$ called *eye closity* which was determined for each eye image I by the ensemble of weak binary classifiers. The positive value of closity indicates that the algorithm classified the input as the closed eye, and the negative value as the open eye; additionally, the bigger the value of closity, the higher the

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

degree to which the eye is closed. A total of 1016 labelled images (24×24 pixels) of closed eyes (positive samples) and 1200 images of open eyes (negative samples) were used in the training stage. The core concept of the method, however, is that the neighboring images in the blinking sequence are dependent, since the blink is a procedure of the eye going from open to closed, and back to open. The use of this temporal information increases the robustness of liveness measurements. The conventional technique for sequence analysis is the Hidden Markov Model (HMM). In this case the values of closity $U(I)$ could be treated as observations linked to the underlying eye states (α – opening, β – closing, γ – ambiguous). HMM assumes that each state depends only on its immediate predecessor, and that each observation variable depends only on the current state. These two assumptions, however, are too restrictive, especially if we have to deal with the noise related to, for example, highlights in the eye region (often caused by artificial lighting) or reflections introduced by glasses. Therefore authors decided to apply the more flexible model called Conditional Random Field (CRF), which used the observation window of size W, describing the relationships between the current state and $(2W+1)$ temporal observations around the current one. This way the long-range dependencies were included in the model. The authors reported very high liveness detection rates (91-98% depending on the presence of glasses) but the method has serious disadvantages: the training is difficult (a large set of images must be collected), the implementation is complicated, and, first of all, the intruder could simply spoof the system by cutting two holes in the photograph, through which the eyes of a real person would be visible.

In case of face images, liveness detection can also be performed through the analysis of reactions to the illumination changes. Intuition tells us that when we apply side lighting, the image of a real face, which is a three dimensional object, will be significantly different than the image of a thin, flat surface of a photograph. The nose of a real face casts a characteristic shadow, for example, which is not present when a sheet of paper is illuminated from a side. Moreover, the surface of a real face is irregularly textured, while the typical glossy photograph produces unnatural reflections. Those observations were exploited by authors of [7]. They used binary images to determine the size and position of shadows by simply counting the black pixels on both sides of the face region. The presence of shadows indicates the proper reaction of the object to the side lighting. Unfortunately, the method is hardly useful in practice, because it requires a special hardware setup, including three appropriately positioned light sources. In order to confirm the liveness, the user has to pass by the first lamp, situated on the left side, and the second one placed further along the path on the right side. The third lamp is fitted to the camera and supplies the front light. Undoubtedly, forcing the user to walk in front of the camera during the verification procedure is unacceptable in other than laboratory conditions.

Since the human face is a 3D structure, its movements certainly generate different images than movements of a photograph, especially in the case of out-of-plane rotations. This notion was used in the method presented in [8]. The authors used the optical flow to represent the motion of the objects; this makes their solution similar to ours. The concept of optical flow will be described in more detail in the following section. Generally saying, optical flow converts each frame of the video sequence into a set of vectors that indicate the direction and velocity of motion observed between consecutive frames; those vectors are attached to the blocks of the image and their distribution contains important information about the type of motion and the object itself. Authors of [8] noticed that in the ideal case the horizontal and vertical components of the motion vectors change linearly across the image when the moving object is two dimensional. Thus, they split the image into halves horizontally and vertically and then, by counting and averaging the components of motion vectors inside those regions, they calculate the coefficients of linear equations describing the motion depicted by the optical flow. This way the model of ideal motion of the 2D object is created. Finally, the

difference *D* between the linear model and the actual motion field is calculated with a simple formula. The higher the value of *D*, the larger the probability that the face is real, not fake. A threshold value must be adjusted in order to find the trade-off between usability and the security level. It seems, however, that this algorithm could be easily spoofed by simply bending the photograph.

3. Liveness detection method

Similarly to the authors of [8] we decided to perform liveness measurements by detecting the differences between motion patterns of three dimensional objects (human faces) and two-dimensional ones (photographs). Consequently, the optical flow was chosen as the most effective motion description. We do not, however, assume any motion model for the real and the fake data. Instead, we use the Support Vector Machine that learns the differences between those motion patterns from examples at the training stage, and uses the collected knowledge during liveness measurements (at the testing stage). Fig. 1 shows the overall architecture of our method. In the following paragraphs we will discuss its building blocks in more detail.

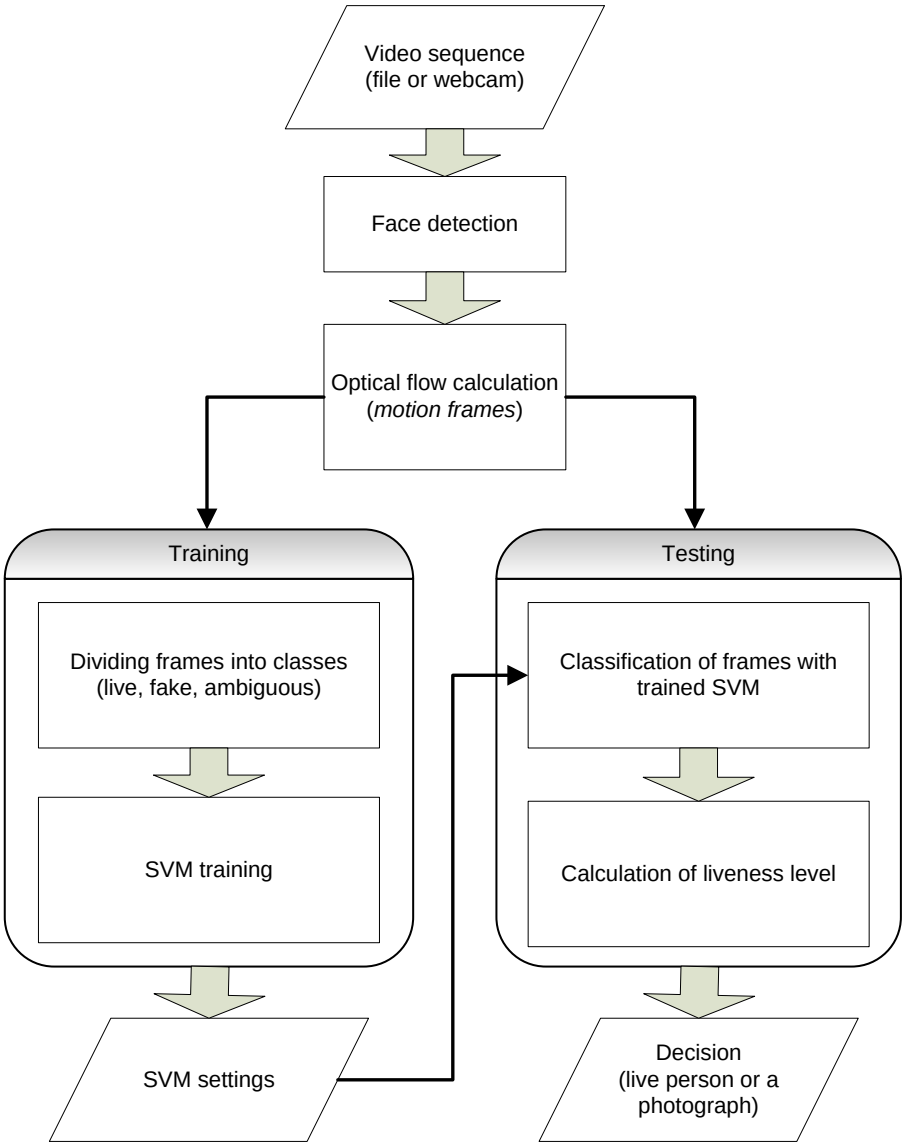

Fig. 1. The simplified block diagram of a liveness detection method.

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

3.1. Data acquisition

We assumed that in order to verify the liveness the user would have to rotate the head to the left, then to the right and once again to the left. This movement, theoretically, should provide enough information for proper motion analysis. At the first stage the face image must be detected and cropped from the background; this process is repeated for each frame of the video sequence. Our system uses the most popular face detector based on the hierarchical classifier introduced by Viola and Jones [4, 9]. In the originally provided square regions of interest about 5% of the area is discarded at left and right side to reduce the influence of the background on the results of motion analysis. The cropped images are then scaled down four times to decrease the noise and speed up further processing.

3.2. Optical flow calculation

Having extracted the sequence of frames containing face images we compute the optical flow. The idea of optical flow was introduced to the field of computer science in the early 1980's [10] and since then it is continuously being developed and improved. Optical flow represents motion in a form of a vector field that allows to transform one image from the video sequence into the next one, by moving the blocks of the first image in the direction indicated by the components of the vector field. Additionally, the length of each vector characterizes the speed observed at the given block. In other words, optical flow is a set of translations describing the motion. In many books and papers the results obtained with this technique are often illustrated by very idealistic motion fields. Although it is possible to achieve such results in certain cases (Fig. 2a), our experiments show that usually, even for artificial objects like a textured ellipse on a uniform black background, the motion field gets quite noisy (Fig. 2b). Therefore, in practice the algorithm must be prepared to handle the unstable data.

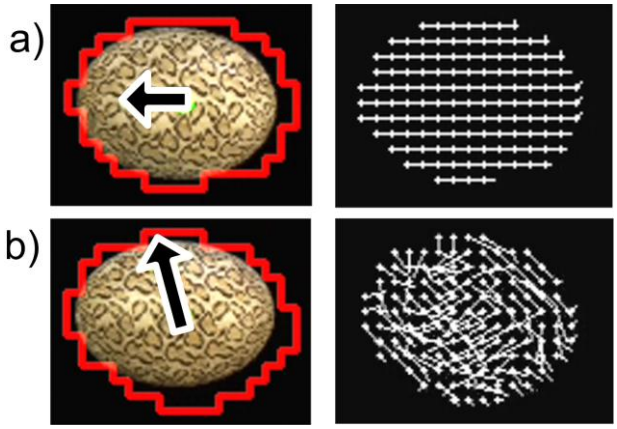

Fig. 2. a) Ideal motion field obtained with optical flow, b) typical, distorted motion field.

There are many approaches to the calculation of optical flow but the goal is always the same: for each pixel we want to calculate the velocity vector **v**

$$\mathbf{v} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (v_x, v_y), \tag{1}$$

that contains the information about the speed and the direction in which the pixel is moving.

Originally, optical flow algorithms were using the gradient-based estimation. Unfortunately, results that they provide are satisfactory only for certain kinds of motion.

Thus, our liveness detection procedure calculates optical flow with the tensor-based algorithm, proposed by Farnebäck in [11]. A detailed description of the method is beyond the scope of this paper but we will briefly introduce the most important concepts.

First, the video sequence is treated as a stack of consecutive frames. In a spatio-temporal image volume like this, the "vertical" axis corresponds to the moments of time. A moving point depicted in the image sequence will create a line inside this three-dimensional space. The orientation of the line can be expressed with the so-called orientation tensor **T**. For *N*-dimensional signals **T** takes the form of an $N \times N$ real symmetric matrix. For a simple signal whose direction is determined by a unit vector **n** we can calculate **T** as

$$\mathbf{T} = a\,\mathbf{n}\mathbf{n}^{\mathrm{T}}, \tag{2}$$

where a is some constant that may encode, for example, the importance of the signal. For our 3D problem, the tensor is a 3×3 positive semidefinite matrix and the expression $\mathbf{v}^{\mathrm{T}}\mathbf{T}\mathbf{v}$ can be treated as a measure describing the local variation of the signal along the direction **v**. The main idea of the method is to estimate **T** from the image data and then to find the velocity vector $\mathbf{v} = [v_x, v_y, 1]^{\mathrm{T}}$ that minimizes the quadratic form $\mathbf{v}^{\mathrm{T}}\mathbf{T}\mathbf{v}$.

In order to approximate **T** we first assume that the signal can be locally (within some neighbourhood) modelled by the second degree polynomial

$$f(\mathbf{x}) \approx \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c\,. \tag{3}$$

The parameters **A**, **b**, and c are computed from image data by the *Normalized Convolution* [12], an algorithm that assigns the signal a certainty component expressing the level of confidence in the reliability of each measure (in this case the weighting function is usually Gaussian). The procedure is equivalent to calculating a weighted least squares approximation of the signal but can be efficiently implemented with the help of convolution filters. Finally **T** is expressed as [11]:

$$\mathbf{T} = \mathbf{A}\mathbf{A}^{\mathrm{T}} + \gamma\,\mathbf{b}\mathbf{b}^{\mathrm{T}}, \tag{4}$$

where the value of parameter γ should decrease when the size of the neighbourhood becomes larger (as the quadratic part should become more significant than the linear component). The faster version of the algorithm does not estimate the velocity from the tensors point by point but assumes that the velocity field over a region can be parameterized according to the affine motion model and uses all the tensors in the region to compute the parameters.

Finally a motion vector is determined for each small neighbourhood (Fig. 3). Initially we planned to exploit the data describing the *directions* of the movements but this information appeared to be too noisy and unstable in practice. Thus we decided to use only the information about the *velocity*, i.e. we considered only the length of each motion vector. Sometimes during the acquisition of the video sequence, for example at the beginning and the end of the process, the subject is not moving at all or the movements are small and random. To prevent the algorithm from processing inadequate data we only record frames in which for more than 30% of pixels the length of the motion vector is greater than 1. Each frame is converted to an 8-bit greyscale image in which the intensity of a pixel corresponds to the speed detected at the given point. These images are normalized in order to utilize the whole intensity range of 0..255. Finally a histogram equalization for each frame is performed and images are saved as regular bitmaps. Fig. 4 shows some examples.

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

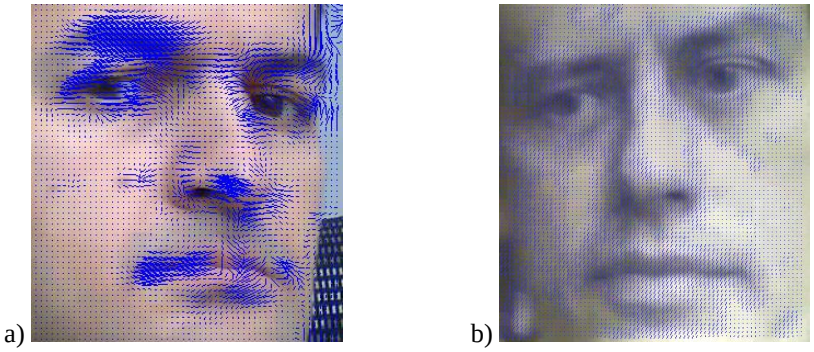

Fig. 3. The examples of optical flow extracted by Farnebäck algorithm: a) live frame, b) fake frame.

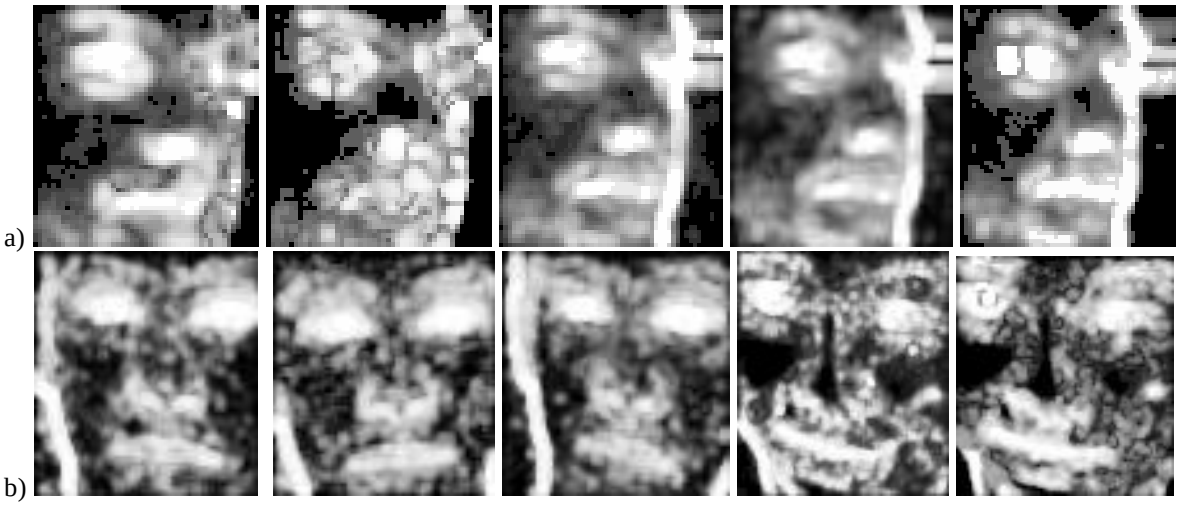

Fig. 4. The examples of "motion frames": a) live sequence, b) fake sequence.

3.3. Initial data selection

The actual decision about the liveness of the subject is provided by the Support Vector Machine. First, however, the SVM classifier must be trained with the sequences of bitmaps representing motion of real faces (genuine data) and the movements of photographs (fake data). In addition to those two classes we introduced the third kind of data: ambiguous frames, characterizing the intersection of the two original training sets. In order to collect them we compare each frame from the "live" training set, represented by the pixel intensity matrix $\mathbf{L}_i$, with each frame $\mathbf{F}_j$ from the "fake" dataset. If the Frobenius norm of the difference matrix $\mathbf{D}_{ij} = \mathbf{L}_i - \mathbf{F}_i$ falls below a threshold (by default set to 1), both frames are moved to the "ambiguous" class.

3.4. Classification with Support Vector Machine

The calculation of optical flow provides information characterizing the motion of the object. Our ultimate goal, however, is to decide whether the information describes a live person or a photograph. Suppose that we are able to encode the motion information representing one video frame within some N-dimensional observation vector $\mathbf{x}$. In order to make a decision concerning $\mathbf{x}$ we need a classifier, i.e. an algorithm that would assign a class label to $\mathbf{x}$ (e.g. +1 meaning a live person, –1 meaning a photograph). Usually, a classifier needs some training; this means that first we have to collect M samples representing both

classes (live persons and photographs) and assign the labels manually. Using this training set the classifier is able to tune its internal parameters.

Our liveness detection method is based on the state-of-the-art classification algorithm called the Support Vector Machine [13]. The overall idea of the SVM is simple (Fig. 5). Each observation from the training set is treated as a point in *N*-dimensional space and the classifier tries to find the hyperplane (or line in the two-dimensional case) separating samples belonging to the opposite classes. More precisely, it looks for the line that guarantees the widest possible margin separating the samples. To this end it identifies the samples lying close to the boundary, the so-called *support vectors*.

More formally, the training data can be represented as a set of pairs $\{\mathbf{x}_i, y_i\}$, where $i = 1, ..., M$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in \mathbf{R}^N$. Let us consider some hyperplane in $\mathbf{R}^N$ that can be described by the equation $\mathbf{w}\cdot\mathbf{x} + b = 0$, where $\mathbf{w}$ is normal to the hyperplane , $|b| / \|\mathbf{w}\|$ is the perpendicular distance from the hyperplane to the origin, and $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$. The task of finding the maximum-margin hyperplane can be formulated as a Lagrangian problem:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \,. \tag{5}$$

As we can see, there is a Lagrange multiplier α_i for every training point. However, when we find the solution by maximizing L_D with respect to α_i subject to constraints

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \,, \tag{6}$$

α_i is greater than 0 only for some of them: those points are the support vectors. They allow us to find the $\mathbf{w}$ parameter of the hyperplane but the more important thing is that the classifier uses them to assign a label y to the unknown object $\mathbf{x}$ according to the following formula

$$y = \operatorname{sgn}\left(\sum_{j=1}^{K} \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x} + b\right), \tag{7}$$

where $K << M$ is the number of support vectors.

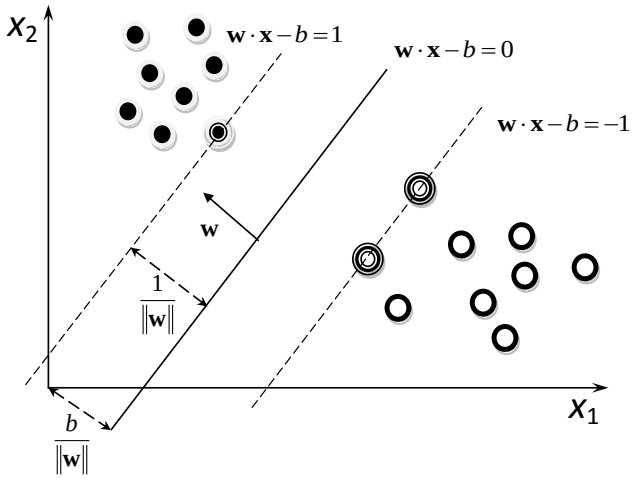

Fig. 5. The maximum-margin hyperplane separating samples from two classes; support vectors are marked with double-line circles.

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

So far we were assuming that the vectors $\mathbf{x}_i$ representing opposite classes were linearly separable. In practice, however, this condition is rarely satisfied. The common way to overcome the problem is to make use of the so called *kernel trick* [13]. The idea is to replace the original *N*-dimensional space of measurements *S* with a new space *V* that has much higher dimensionality. Linear classification in the modified space corresponds to the nonlinear decision boundaries in the original space. We do not even have to perform explicit mapping from *S* to *V*. As formula 5 indicates, we just need the definition of a dot product in space *V*. Thus, the only modification to the linear SVM algorithm is that in equations (5) and (7) the product $\mathbf{x}_i \cdot \mathbf{x}_j$ is replaced with a more general function $K(\mathbf{x}_i, \mathbf{x}_j)$, called *kernel function*. The basic version of the SVM can be viewed as the algorithm with the linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$. In most of the cases, however, two other options are used: the polynomial kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\gamma\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) + b\right)^d \tag{8}$$

or the Gaussian radial basis function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right). \tag{9}$$

As we can see, the original SVM is a binary classifier. Because in our algorithm the three categories of motion frames are defined (live, fake, ambiguous) we have to apply the multiclass version of the SVM that prepares the final decision by solving the set of pairwise classification problems [14]. We use the nonlinear SVM with the polynomial kernel (8) in which $\gamma = 1$, $b = 1$ and $d = 4$. Obviously, other classification methods could be used at this stage, for example the SDF approach that we have improved recently [15].

3.5. Liveness level calculation

During the liveness test the trained classifier assigns each bitmap from the sequence to one of the classes. As a result each sequence is described by four values:

- N_T, the total number of frames in the sequence,
- N_L, the number of "live" frames,
- N_F, the number of "fake" frames,
- N_A, the number of "ambiguous" frames.

Different definitions of liveness can be created from the above parameters. In our solution we adopted the following formula

$$L = \frac{N_L}{N_T - N_A} \cdot 100\% = \frac{N_L}{N_L + N_F} \cdot 100\%. \tag{10}$$

When the value of L exceeds 50%, the sequence is considered as representing a person; otherwise it is classified as a fake.

6. Experiments

Our method has been implemented using C++ and C# and tested during a series of experiments that involved 5 people and a total number of 100 video sequences: for each subject 10 sequences showed a live person and the next 10 showed a rotating photograph. The SVM was trained individually for each person. However, for all persons the training sequence representing the fake data (the negative set) was the same. It was prepared by recording the

movements of the photograph showing the so-called ideal male face, available from Regensburg University [16]. On the other hand, for each individual, one of the "live" sequences was used as a positive set during training. This setup simulated real-life conditions, in which the user would be requested to record only one live sequence during the registration. The model of the "fake" movements should be built into the system, as it would be too impractical for the user to simulate the attack by printing photos and rotating them properly in front of the camera. To make the task more difficult, in 4 out of 5 cases the photographs used to prepare the "fake" testing sequences of a person were extracted from the "live" testing videos of the same subject. It was important, because otherwise the differences between the live and fake sequences could be caused by the variations of appearance rather than changes of the motion patterns. The testing videos were recorded under uncontrolled conditions (different backgrounds, illumination types and light directions). The results of experiments are summarized in Figs 6 and 7.

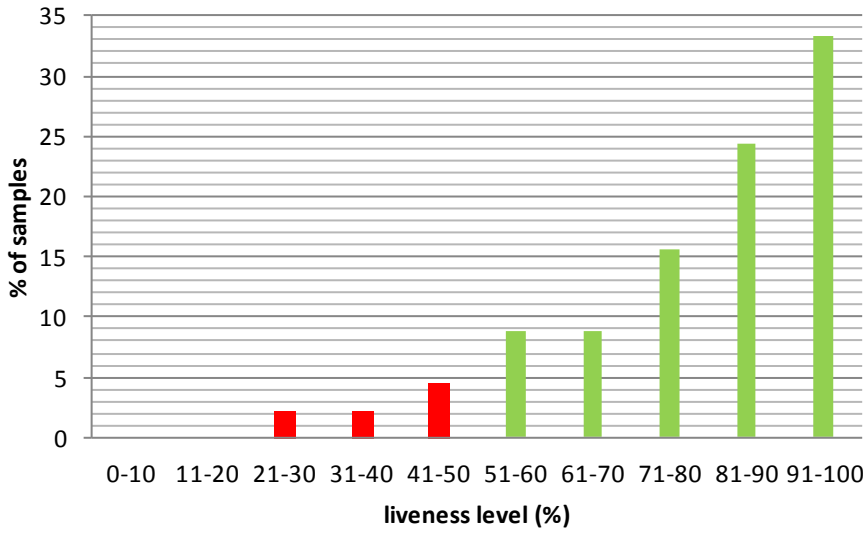

Fig. 6. The test results for live sequences.

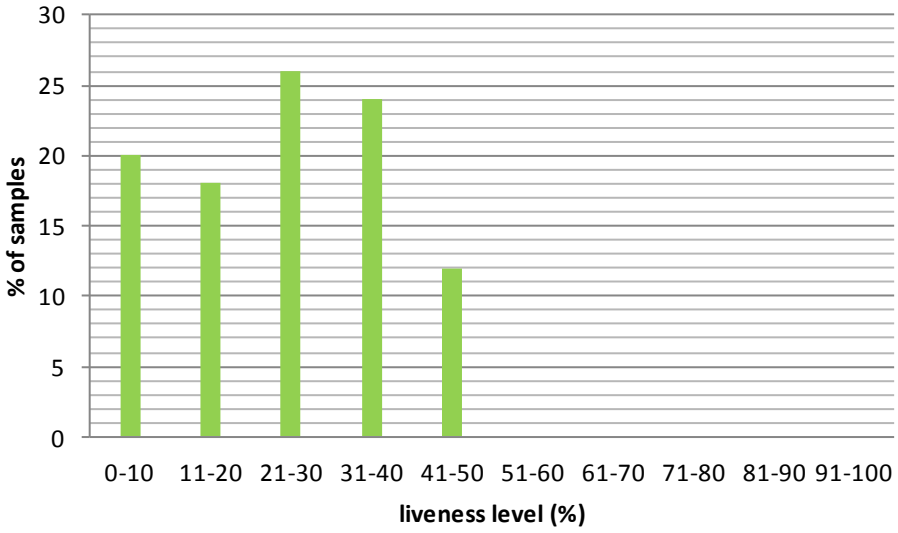

Fig. 7. The test results for fake sequences.

Metrol. Meas. Syst., Vol. XIX (2012), No. 2, pp. 257-268.

The results can be considered satisfactory, as we reached the false rejection rate of 8.9% and none of the fake sequences passed the liveness test (FAR = 0%). Some of the intrusion trials, however, (about 12%) were dangerously close to success. This indicates that although determining the liveness by performing motion analysis makes sense, improvements are necessary to make our method more robust. First of all, the *temporal* information must be included in the decision process. Currently we only classify the individual "motion frames" determined with the optical flow. The next version of the algorithm is planned to analyze the whole sequence of motion changes to find out which "motion frame" follows which. The Hidden Markov Models or Conditional Random Fields could be applied to learn the temporal templates of live and fake data.

We have also made some additional observations:

- The accuracy of the face detector is crucial. During experiments, when a person rotated the head too fast, many frames were dropped. This was the reason for the small number of data that represented some of the sequences. In the future version we will try to perform face tracking with the help of our implementation of Active Appearance Models [17].
- In poor illumination conditions the useful information gets close to the noise level and the measurements become unstable. Most of the situations when the algorithm almost allowed false acceptance were related to the analysis of videos taken in dimmed light.
- Although it is quite difficult to cheat the system by rotating the conventional flat photograph, the results are much more worrying (liveness close to 50%) when the intruder crops the face precisely and uses it as a kind of a simple 3D mask attached to the head.

4. Conclusions

In this paper we proposed a novel method for measuring the liveness of face images. Our experiments showed that the combination of optical flow estimation and SVM classifier can be used to achieve the task of liveness verification. Although it proves that the overall idea is correct, the robustness of our algorithm should be improved before introduction to real-life biometric systems. Therefore, the next version of the method will include the temporal analysis of motion patterns extracted from video sequences.

Acknowledgements

This work was supported by the Polish National Science Centre (research project No. N N516 367936).

References

[1] Researchers spoof, bypass face-recognition authentication systems, *Homeland Security NewsWire* (2009). *http://www.homelandsecuritynewswire.com/researchers-spoof-bypass-face-recognition-authentication-systems.*

[2] SART-2, Biometric security system for mobile workstations (2012). *http://sart2.eti.pg.gda.pl/en/.*

[3] Jee, H., Jung, S., Yoo, J. (2006). Liveness Detection for Embedded Face Recognition System. *International Journal of Biomedical Sciences*, 235-238.

[4] Viola, P., Jones, M.J., (2004). Robust Real-Time Face Detection. *Int. J. Comp. Vision*, 57(2), 137-154.

[5] Wang, H., Li, S.Z., Wang, Y. (2004). Face Recognition under Varying Lighting Conditions Using Self Quotient Image. *Proc. 6th IEEE Int. Conf. on Automatic Face and Gesture Recognition (FGR'04)*.

[6] Pan, G., Sun, L., Wu, Z. (2007). Eyeblink-based Anti-Spoofing in Face Recognition from a Generic Webcamera. *Proc. IEEE 11th Int. Conf. on Computer Vision (ICCV 2007)*.

[7] Huang, C.-H., Wang, J.-F. (2008). SVM-based One-Against-Many Algorithm for Liveness Face Authentication. *Proc. IEEE 11th Int. Conf. on Systems, Man and Cybernetics (SMC 2008)*, 744-748.

[8] Bao, W., Li, H., Li, N., Jiang, W. (2009). A Liveness Detection Method for Face Recognition Based on Optical Flow Field. *Proc. Int. Conf. on Image Analysis and Signal Processing (IASP 2009)*, 233-236.

[9] Dembski, J., Smiatacz, M. (2010). Modular machine learning system for training object detection algorithms on a supercomputer. *Advances in Systems Science*, Academic Publishing House EXIT, 353-361.

[10] Horn, B., Schunk, B. (1981). Determining Optical Flow. *Artificial Intelligence*, 17, 185-204.

[11] Farnebäck, G. (2000). Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models. *Proc. Int. Conf. on Pattern Recognition (ICPR 2000).*

[12] Farnebäck, G. (1999). Spatial Domain Methods for Orientation and Velocity Estimation. *Lic. Thesis LiU-Tek-Lic-1999:13*. Dept. EE, Linköping University.

[13] Burges, Ch.J.C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2), 121-167.

[14] Cui, J., Wang, Y. (2011). Analog circuit fault classification using improved one-against-one Support Vector Machines. *Metrol. Meas. Sys.*, 18(4), 569-582.

[15] Smiatacz, M., Malina, W. (2011). SDF classifier revisited. *Expert Systems*. DOI: 10.1111/j.1468-0394.2011.00589.x

[16] Beauty check, Average faces (2012). *http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II /Psychologie/Psy_II/beautycheck/english/durchschnittsgesichter/durchschnittsgesichter.htm.*

[17] Smiatacz, M., Sikora D. (2010). AAM Toolkit: a system for visual object appearance modelling. *Advances in Multimedia and Network Information System Technologies*, Advances in Intelligent and Soft Computing, 80, 121-129.