Using Particle Swarm Optimization for Scaling and Rotation invariant Face Detection

Ermioni Marami and Anastasios Tefas, *Member, IEEE*

Abstract— **Common face detection algorithms exhaustively search in all possible locations in the image for precisely located, frontal faces. In this paper, a novel face detection algorithm based on Particle Swarm Optimization (PSO) method for searching in the image is proposed. The algorithm uses a linear Support Vector Machine (SVM) as fast and accurate classifier and searches for a face in four dimensions: plane, orientation of the face, size of the face. Using PSO, the exhaustive search in all possible combinations of the 4D coordinates can be avoided, saving time and decreasing the computational complexity. Moreover, linear SVMs are proved to be a powerful and fast classifier for demanding applications. Experimental results under real recording conditions in the BioID and VALID database are very promising and indicate the potential use of the proposed approach to real applications.**

I. Introduction

The goal of face detection is to find and localize faces in images or videos [1] and return the location and extent of each face. It is by far the most active specialization in object detection, since it is an essential step in most face-analysis applications, such as facial expression analysis for human computer interfaces, face recognition for access control and surveillance, as well as multimedia retrieval.

Face detection from a single image is a challenging task due to the variability of the object of interest itself and the environment: scale, location, orientation (up-right, rotated), pose (frontal, profile), background, lighting and camera characteristics. The following problems are associated with face detection and need to be considered:

- *Size*: A face detector should be able to detect faces in different sizes. This is usually achieved by either scaling the input image or the object model. Nevertheless, the size of the object usually influences the reliability of the detection since small faces are more difficult to detect than large faces.
- *Position*: A face detector should be able to detect faces at different positions within the image. This is usually achieved by sliding a window over the image and applying the detection step at each image position. The choice of the step size directly influences the detection speed and precision.
- *Orientation*: Faces can appear in different orientations within the image plane depending on the angle of the camera and the face. For 2D data, such as images, the common rotation is the inplane rotation (roll). The inplane rotation is a rotation along the axis which is perpendicular to the image plane and leads to a frontal nonupright face. This type of rotation can be easily handled by rotating the image and applying the frontal face detector at different angles.
- *Illumination*: Varying illumination can be a big problem for face detection since it changes the color and the appearance of the face depending on the color, the direction and the intensity of the light.
- *Number*: Most of the face detection approaches are able to detect multiple faces within a single image, while there are others which aim to detect one face within a single image.
- *Presence or absence of structural components* [2]: Facial features such as beards, mustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape, color, and size.

In [2] existing face detection techniques to detect faces from a single intensity or color image are reviewed and classified into four major categories. These categories are:

- *Knowledge-based methods*: These rule-based methods encode human knowledge of what constitutes a typical face. Usually, the rules capture the relationships between facial features.
- *Feature invariant approaches*: These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use these to locate faces. These features could be facial features (such as eyebrows, eyes, nose, mouth, hairline), texture or skin color.
- *Template matching methods*: Several standard patterns of a face are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection.
- *Appearance-based methods*: In contrast to template matching, the models (or templates) are learned from a set of training images which should capture the representative variability of facial appearance. These learned models are then used for detection. Subcategories of this category are: Eigenfaces, Distribution-Based Methods, Neural Networks, Support Vector Machines, Sparce Network of Winnows, Naive Bayes Classifier, Hidden Markov Model, Information-Theoretical Approach and Inductive Learning.

On the contrary to the most of the state-of-the-art face recognition and facial expression recognition methods that

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, Box 451, 54124, Thessaloniki, Greece, (phone: +30 2310 991932; email: emarami@csd.auth.gr, tefas@aiia.csd.auth.gr).

consider that the face has been correctly and precisely located in the image and that it is in frontal view, the faces we are searching for do not have a particular location, orientation or size. In the present paper we search for a face in four dimensions (4D): plane, orientation of the face, size of the face. While searching for a face on plane, we rotate and resize the image for a specific range of values for the rotation and the scaling factor. This way, we can detect faces which are frontal, inplane rotated, small or large. The most competitive face detection algorithms are searching exhaustively in the test image for localizing the face. To avoid the exhaustive search of all possible locations in the image, we propose a face detector algorithm based on swarm intelligence and more specifically the particle swarm optimization (PSO) method. Each particle is equipped with a very fast and accurate classifier and cooperates with the other particles to give an intelligent swarm that is able to detect faces. The presented optimization method proved that the exhaustive search in all possible combinations of the 4D coordinates can be avoided, saving time and decreasing the computational complexity. Indeed, approximately only 0.4% of the possible image positions had to be examined.

In order to check whether each image sub-window under investigation is a face or not, we used a very fast and efficient classifier, a linear support vector machine (SVM) that reduces the detection to an inner vector product.

The paper is structured as follows: Section II summarizes SVMs' theory. Section III outlines the main idea of the PSO method. Section IV describes the structure of the proposed face detection system. Section V includes the experiments executed and the corresponding results that verify the efficiency of the proposed algorithm. Finally, Section VI draws the conclusion of this work.

II. Support Vector Machines (SVMs)

In this section we briefly review the basis of the theory of SVMs in classification problems. SVMs perform pattern classification for two-class problems by determining the *separating hyperplane* with maximum distance (margin) to the closest points of the training classes. These points are called support vectors.

Suppose we are given a set $\mathcal{S}$ of labeled training points

$$(y_1, \mathbf{x}_1), ..., (y_l, \mathbf{x}_l). \tag{1}$$

Each training point $\mathbf{x}_i \in R^N$ belongs to either of the two classes and is given a label $y_i \in \{-1, 1\}$ for $i = 1, ..., l$ [3]. For the linearly separable case, suppose that all the training data can be separated by a hyperplane that is represented by the perpendicular vector $\mathbf{w}$ and the bias b such that:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0 \qquad \forall i \tag{2}$$

Those training points for which the equality in Equation (2) holds, are the support vectors and their removal would change the solution found.

For the Lagrangian formulation of the problem, we introduce positive Lagrange multipliers a_i, $i = 1, ..., l$, one for each of the inequality constraints (2). The rule is that for constraints of the form $c_i \geq 0$, the constraint equations are multiplied by *positive* Lagrange multipliers and subtracted from the objective function, to form the Lagrangian. For equality constraints, the Lagrange multipliers are unconstrained. This gives Lagrangian:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i (\mathbf{w}^T \mathbf{x}_i + b) + \sum_{i=1}^{l} a_i \tag{3}$$

We must now *minimize* L_P with respect to $\mathbf{w}$ and b, and simultaneously require that the derivatives of L_P with respect to all the a_i vanish, all subject to the constraints $a_i \geq 0$. We can, also, solve the following "dual" problem: *maximize* L_P, subject to the constraints that the gradient of L_P with respect to $\mathbf{w}$ and b vanish, and subject also to the constraints that $a_i \geq 0$. This particular dual formulation of the problem is called the Wolfe dual [4].

Requiring that the gradient of L_P with respect to $\mathbf{w}$ and b vanish give the conditions:

$$\mathbf{w} = \sum_i a_i y_i \mathbf{x}_i \tag{4}$$

$$\sum_i a_i y_i = 0. \tag{5}$$

For non-separable data, we can relax the constraints (2) using positive slack variables $\xi_i, i = 1, \ldots, l$ [5]. The constraints become:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \geq 0, \qquad \xi_i \geq 0, \qquad \forall i \tag{6}$$

A nonlinear separating hyperplane (Nonlinear SVM) can be found if we first map the data to a higher dimension feature space $\mathcal{H}$, using a nonlinear map function Φ:

$$\Phi : \mathbf{R}^d \mapsto \mathcal{H} \tag{7}$$

Then of course the training algorithm would only depend on the data through dot products in $\mathcal{H}$, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Now, we can use a "kernel function" K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. This way, we only need to use K in the training algorithm, and would never need to explicitly even know what Φ is. Some examples of kernels used in SVMs and investigated for pattern recognition problems are the polynomial and the Gaussian rbf kernel:

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^p \tag{8}$$

$$(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2 / 2\sigma^2} \tag{9}$$

It is obvious that when we use linear SVMs, the testing procedure requires only a multiplication of the input vector with the one given in (4) and addition of the bias term. However, if we use nonlinear kernels the computational cost in the testing procedure depends on the number of support vectors and in most cases (in our case also) is more than 10^3 bigger.

III. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique originally proposed by James Kennedy and Russell C. Eberhart in 1995 [6]. PSO is a search algorithm based on the simulation of the behavior of birds within a flock.

In order to establish a common terminology, in the following we provide some definitions of several technical terms commonly used [7]:

- **Swarm:** Population of particles.
- **Particle:** Member (individual) of the swarm. Each particle represents a potential solution to the problem being solved. The position of a particle is determined by the solution it currently represents.
- **pbest** (*personal best*)**:** Personal best position of a given particle, so far. That is, the position of the particle that has provided the greatest success (i.e. the maximum value given by the classification method used).
- **lbest** (*local best*)**:** Position of the best particle member of the neighborhood of a given particle.
- **gbest** (*global best*)**:** Position of the best particle of the entire swarm.
- **Leader:** Particle that is used to guide another particle towards better regions of the search space.
- **Velocity (vector):** This vector drives the optimization process, that is, it determines the direction in which a particle needs to "fly" (move), in order to improve its current position.
- **Inertia weight:** Denoted by W, the inertia weight is employed to control the impact of the previous history of velocities on the current velocity of a given particle.
- **Learning factor:** Represents the attraction that a particle has toward either its own success or that of its neighbors. Two are the learning factors used: C_1 and C_2. C_1 is the *cognitive* learning factor and represents the attraction that a particle has toward its own success. C_2 is the *social* learning factor and represents the attraction that a particle has toward the success of its neighbors. Both, C_1 and C_2, are usually defined as constants.
- **Neighborhood topology:** Determines the set of particles that contribute to the calculation of the *lbest* value of a given particle.

The position of each particle is changed according to its own experience (*pbest*) and that of its neighbors (*lbest* and *gbest*). Let $\mathbf{z}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\mathbf{u}_i(t)$ to the current position, i.e.:

$$\mathbf{z}_i(t) = \mathbf{z}_i(t-1) + \mathbf{u}_i(t) \qquad (10)$$

The velocity vector reflects the socially exchanged information and, in general, is defined in the following way:

$$\begin{aligned}\mathbf{u}_i(t) = W\mathbf{u}_i(t-1) &+ r_1 C_1(\mathbf{z}_{pbest_i} - \mathbf{z}_i(t)) \\ &+ r_2 C_2(\mathbf{z}_{leader} - \mathbf{z}_i(t)) \qquad (11)\end{aligned}$$

where $r_1, r_2 \in [0,1]$ are random values. Particles can be connected to each other in any kind of neighborhood topology represented as a graph. Our face detector uses the **fully connected graph**. The fully connected topology connects all members of the swarm to one another. Each particle uses its history of experiences in terms of its own best solution so far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*) as can be seen in Figure 1. In this case, *leader* $=$ *gbest* in Equation 11.

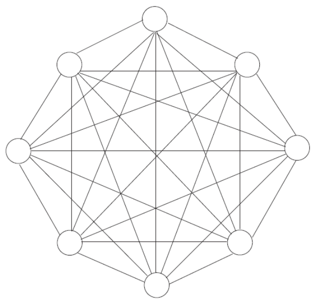

Fig. 1. The fully connected graph represents the *fully connected* neighbohood topology (each circle represents a particle). All members of the swarm are connected to one another.

The neighborhood topology is likely to affect the rate of convergence as it determines how much time it takes to the particles to find out about the location of good (better) regions of the search space. Since, in the *fully connected* topology all particles are connected to each other, all particles receive the information of the best solution at the same time and, thus, the swarm tends to converge more rapidly than when using other topologies. However, the *fully connected* topology is also more susceptible to suffer premature convergence (i.e. to converge to local optima) [8].

Algorithm 1 *PSO*

```
Initialize swarm (positions and velocities)
Locate leader
$g = 0$
while $g < gmax$ do
  for each particle do
    Update Position (Flight)
    Evaluation
    Update pbest
  end for
  Update leader
  $g$++
end while
```

Algorithm 1 shows, in pseudocode form, the way the general PSO algorithm works. First, the swarm is initialized, both positions and velocities. The corresponding *pbest* of each particle is initialized and the leader is located (the *gbest* solution is selected as the leader). Then, for a maximum number of iterations, each particle flies through the search space updating its position (using Equations 10 and 11) and its *pbest* and, finally, the leader is updated too.

IV. Combining PSO and SVMs for Face Detection

In this section the final face detection system is described. This discussion includes details on training the classifier used, preprocessing images and the structure of the system.

A. Training SVM

We trained and compared SVMs using linear and polynomial kernels. The comparison between them shows that, in our case, the computational complexity for nonlinear SVMs is 1000 times more intensive than linear SVMs. The linear SVMs give a slightly lower success rate ($\approx 1\%$) but they are much faster. Nonlinear SVMs are more intensive because we have to compute inner products for all the support vectors which in our case are above 1000. The training data for the linear SVM we used consisted of 2901 grayscale face images and 28121 grayscale non-face images of size 19×19. The face images presented a great variability of structural components, such as beards and glasses. These images, are taken from the CBCL Face Database, which is available at http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html. After fine-tuning the training parameters using cross-validation we trained the SVMs to the whole training and test set of the CBCL database and the success training rate for this classifier was found to be 98.74%. This trained classifier has been used for all the experiments performed in the test databases as explained in Section V. No image from the test database has been used for training.

B. Image Preprocessing

In order to eliminate or minimize the effect of different lighting conditions, images used for training SVMs were histogram equalized and normalized so that all pixel values are between 0 and 1. Normalization and histogram equalization are therefore necessary during detection as well. Histogram equalization enhances the contrast of images by transforming the values in an intensity image, so that the histogram of the output image approximately matches a uniform histogram. Each image sub-window under investigation is processed using the above methods. Histogram equalization is a computationally intensive process and thus avoiding the exhaustive search with the use of PSO is of outmost importance for real time detection.

C. Structure of the Face Detector

Algorithm 2 describes the face detection process using a linear SVM as classifier and the PSO method to decrease computation time. The general idea as described previously is that each particle has its own intelligence using the SVM classifier in order to evaluate the current position on whether it contains a face or not. The particles communicate and inform one another for the most possible face position at each iteration.

First, we initialize the parameters *inertia*, *correction factor*, *maxVelocity* and *minVelocity*, used to update velocity, for each dimension x, y, z (rotation factor) and w (scaling factor). After loading the trained linear SVM and reading the input image (the one in which we are looking for a face), we perform any necessary transformations. We initialize particles to random positions (x, y, z, w) using predetermined limits and random values.

Algorithm 2 *Face Detector using PSO and SVM*

```
Initialize parameters for each dimension: inertia, correction factor, maxVelocity, minVelocity
Load trained linear SVM and Read input image
Initialize particles to random positions $(x, y, z, w)$
$iteration = 0$, $repeat = 0$
while $repeat < R$ do
  for each particle do
    Update position
    Resize image using scaling factor $w$
    Rotate image using rotation factor $z$
    Compute $(x', y')$ for transformed image
    Process this square of image with histogram equalization
    Evaluate SVM's result for this position
    Update pbest
  end for
  Update gbest's index
  Update velocities for each particle
  $iteration++$
  if $gbest's$ position has not changed and $iteration > K$ then
    $repeat++$
  end if
end while
$gbest's$ position = left upper corner of detected face
```

Secondly, the position of each particle is updated using Equation 10. The initial image is resized by scaling factor w and rotated by rotation factor z. The point (x, y) is transformed meeting the above transformation of the image. This is the left upper corner of a 19×19 sub-window which is histogram equalized and used to compute the output of the linear SVM. The bigger the latter output is, the more possible this sub-window corresponds to a face. *Pbest* is updated using its history of experiences in terms of its own best solution (SVM output) so far.

In succession, *gbest* is computed from *pbest* values. The velocity of each particle is updated using Equation 11.

After K iterations we inspect whether the particles converge or not. If for $R + 1$ successive iterations *gbest* is at the same position, we assume that this location probably contains a face. We check whether the value of SVM at this point is above a predetermined threshold. If the value of *gbest* is larger than this threshold, we terminate the detection procedure. If *gbest*'s value is below the threshold we initialize the particles again at random positions and we repeat the above procedure.

D. Step size for each dimension

The choice of the step size for each dimension directly influences the detection speed and precision. In order to estimate algorithm's robustness for each dimension we did some tests. We came up with the following results:

- x: minimum velocity $= -5$ pixels and maximum velocity $= 5$ pixels
- y: minimum velocity $= -5$ pixels and maximum velocity $= 5$ pixels
- z: minimum velocity $= -10°$ and maximum velocity $= 10°$
- w: minimum velocity $= 10\%$ under the scaling factor and maximum velocity $= 10\%$ above the scaling factor

V. Experimental Results

A. Frame Detection Accuracy (FDA)

Frame Detection Accuracy (FDA) is an evaluation metric used to measure any object detection algorithm's performance. This measure calculates the spatial overlap between the ground-truth and the algorithm's output. The detection accuracy is estimated as the ratio of the spatial intersection between the ground-truth and the detected object and the spatial union of them [9]. If the face detection algorithm aims to detect multiple faces, the sum of all the overlaps is normalized over the average number of ground-truth and detected objects. If N_G is the number of ground-truth objects and N_D the number of detected objects, FDA is defined as:

$$FDA = \frac{Overlap\_Ratio}{\left[\frac{N_G+N_D}{2}\right]} \tag{12}$$

where

$$Overlap\_Ratio = \sum_{i=1}^{N_{mapped}} \frac{|G_i \cap D_i|}{|G_i \cup D_i|}, \tag{13}$$

N_{mapped} is the number of mapped object pairs in the image, G_i is the i-th ground-truth object image region and D_i is the i-th detected object image region.

B. Speed Performance of the Face Detector

The speed performance of the presented face detector is directly related to various parameters, such as the swarm size, the image's initial size and the repeat cycles. The average number of positions examined using PSO was a very small percentage of all the possible combinations of the 4D coordinates ranging from 0.2 to 0.6%. The latter demonstrates the significant reduction in the number of possible solutions to which we have to apply the classifier each time. That is, using PSO for searching we are able to reduce the time needed for a detection by a factor of 200 and greater for any given face detection algorithm. Furthermore, using linear SVM instead of nonlinear gives another 10^3 boost in the detection speed.

C. BioID Face Database results

In this section, we present the experimental results for the BioID Face Database. We applied the proposed algorithm to the above database consisted of 1521 gray level images with an initial resolution of 384×286 pixels. Each image record shows the frontal view of a face of one out of 23 different test persons in real indoor environments. Ground-truth for the exact location of the eyes in each image is given by the authors of the database and has been used in order to define a ground-truth bounding box around the face. The bounding-box has been defined to coincide with the images used for training. Emphasis has been placed on *real world* conditions and, therefore, the testset features a large variety of illumination, background and face size. The BioID Database is available at http://www.bioid.com/support/downloads/software/bioid-face-database.html.

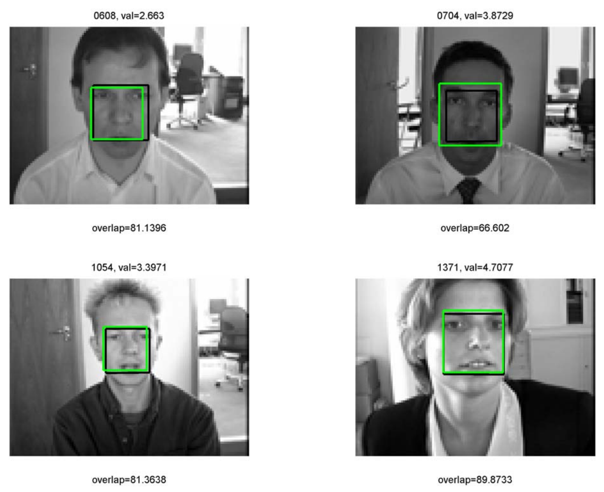

Fig. 2. Output of the proposed face detector on a number of frontal test images from the BioID Database.

Figure 2 shows the output of the proposed face detector on some frontal images from the BioID Face Database along with the overlap and the detector's output value. Black rectangles represent the ground-truths of every image, while green (lighter gray) windows represent the proposed algorithm's output.

TABLE I

Detection rates for the presented algorithm (SVM-PSO) in comparison with Viola-Jones' algorithm (OpenCV) for the BioID Face Database.

Detector	0.19, 0°	0.25, 0°	0.30, 0°	0.3, 20°
SVM-PSO	93.95%	93.62%	94.08%	88.36%
OpenCV	0%	83.50%	94.48%	37.67%

Table I lists the detection rate for the presented algorithm in comparison with Viola-Jones' state-of-the-art algorithm [10] using as threshold for overlap the value 25.00. In order to have a fair comparison we optimized the output of the competitive algorithm to match as much as possible with the ground-truth. For initial scaling factor 0.19 and rotation factor $0°$, images are too small for the Viola-Jones algorithm to be detected while our algorithm gives a very good detection rate. So, we apply the algorithms to larger images (scaling factor 0.25 and 0.3), while the rotation factor remains the same, and the detection rates for our algorithm remain very good. The Viola-Jones algorithm gives for scaling factor 0.25 a relatively good detection rate but lower to our algorithm's, whilst for scaling factor 0.3 it gives a detection rate similar to our's.

In order to investigate the performance of our algorithm for rotated images, we apply both algorithms on 20° rotated images from the available BioID Face Database. We have used rotated versions of the BioID images using prespecified rotation degrees in order to simulate a rotated version of a face that can be found in a challenging real image. So, for initial scaling factor 0.3 and rotation factor 20° the detection rate for our algorithm is satisfying good, while Viola-Jones' algorithm gives a very low detection rate.

Figure 3 shows the output of our face detector on some rotated images from the BioID Face Database along with the overlap and the detector's output value. As above, black rectangles represent the ground-truths of every image, while green (lighter gray) windows represent the proposed algorithm's output.

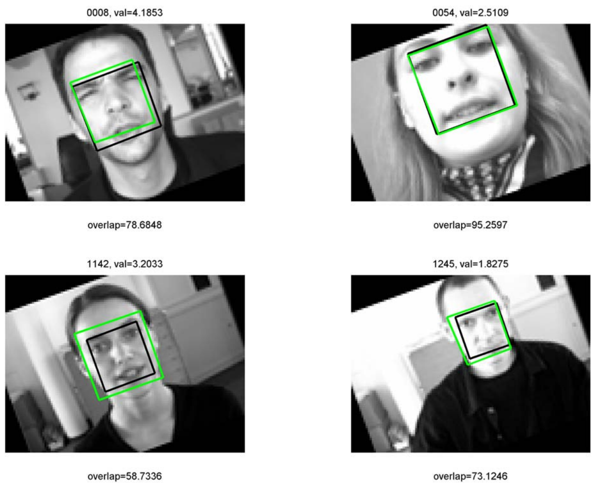

Fig. 3. Output of our face detector on a number of 20° rotated images from the BioID Database.

We can, also, mention that the classifier used in Viola-Jones' algorithm is trained using much more training images than our classifier and it is well known that the number of training samples has a direct effect on the classification performance. Unfortunately, we have not had access to a larger dataset in order to train better the proposed classifier. This shows that if we had used a larger training dataset, we would have probably had much better detection rates for our algorithm.

D. VALID Database results

We, also, applied both algorithms to the VALID Database, available at http://ee.ucd.ie/validdb/datasets.html. This database consists of still images of one face with 576×720 initial size, divided in three sets of 530 images. The initial scaling factor we used is 0.21, while the overlap threshold was the same as above.

Table II lists the detection rates of the proposed algorithm in comparison with Viola-Jones' algorithm detection rates for the VALID Database. It is obvious that SVM-PSO algorithm

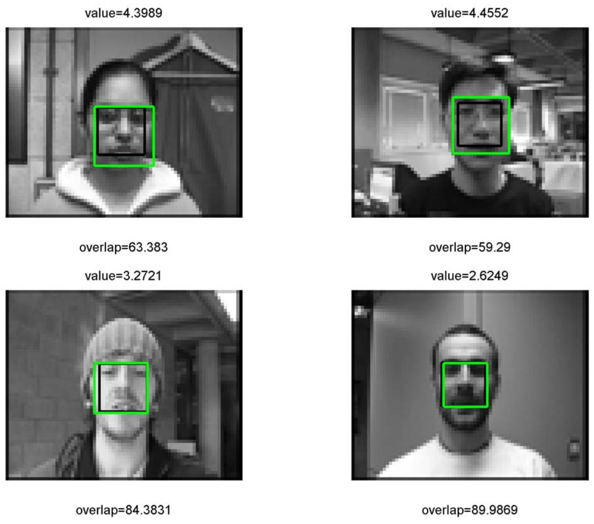

Fig. 4. Output of our detector on a number of test images from the VALID Database.

TABLE II
DETECTION RATES FOR THE PRESENTED ALGORITHM (SVM-PSO) IN COMPARISON WITH VIOLA-JONES' ALGORITHM (OPENCV) FOR THE THREE DATASETS OF THE VALID DATABASE.

Detector	dataset1-0	datset1-10	datset1-50
SVM-PSO	87.55%	88.49%	87.74%
OpenCV	46.98%	43.96%	46.42%

is far superior to OpenCV, because the detection rates for our algorithm are satisfying good for each dataset, whilst Viola-Jones algorithm's detection rates are very low.

Figure 4 shows some outputs for the proposed algorithm on images of that database. Black rectangles represent the ground-truths of every image, while green (lighter gray) windows represent the proposed algorithm's output.

VI. CONCLUSIONS

In this paper we have dealt with the problem of face detection in still images for frontal, rotated, large or small faces. We presented a fast and accurate face detection system that features a face detection, scaling and rotation-aware algorithm. To avoid exhaustive search in all possible combinations of coordinates in 4D space, we used the PSO method, while to save time and decrease the computational complexity we used as classifier a linear SVM. Experimental results demonstrated the algorithm's good performance in a dataset with frontal and rotated images recorded under *real world* conditions and proved its efficiency, especially in cases of rotated faces or faces of various sizes (very small or larger). A very important feature is the algorithm's ability to return an estimation of the face's rotation and scaling factor in addition to its location. A lot of attention has been drawn to this issue because it can be very useful in a variety of applications, especially when face detection is used as a preliminary step for future analysis in applications such as facial expression analysis for human computer interfaces, face recognition and multimedia retrieval. The proposed method can be combined with any face detector, e.g., the one used in OpenCV, to reduce their execution time.

Acknowledgement

This work has been funded by the Collaborative European Project MOBISERV FP7-248434 (http://www.mobiserv.eu), An Integrated Intelligent Home Environment for the Provision of Health, Nutrition and Mobility Services to the Elderly.

References

[1] L. Goldmann, U.J. Mönich and T. Sikora, "Components and Their Topology for Robust Face Detection in the Presence of Partial Occlusions," *IEEE Transactions on Information Forensics and Security*, Sept. 2007, vol. 2, no. 3, pp. 559–569.

[2] M. Yang, D.J. Kriegman and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2002, vol. 24, no. 1, pp. 34–58.

[3] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, Boston, Jun. 1998, vol. 2, no. 2, pp. 121–167.

[4] R. Fletcher, "Practical Methods of Optimization,", John Wiley and Sons, Inc., 2nd edition, 1987.

[5] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, 1995, vol. 20, pp. 273–297.

[6] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, IEEE Service Center, Piscataway, New Jersey, 1995, pp. 1942–1948.

[7] M. Reyes-Sierra and C.A. Coello Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," *International Journal of Computational Intelligence Research*, 2006, vol. 2, no. 3, pp. 287–308.

[8] A.P. Engelbrecht, "Fundamentals of Computational Swarm Intelligence," John Wiley & Sons, 2005.

[9] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova and J. Zhang, "Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Feb. 2009, vol. 31, no. 2, pp. 319–336.

[10] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, 2004, vol. 57, no. 2, pp. 137–154.