БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ • BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, **68**
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, **68**

София • 2017 • Sofia

Face detection and recognition for intelligent service robots

Denis Chikurtev

Institute of Information and Communication Technologies – BAS

Email: denis@iinf.bas.bg

Abstract: Intelligent service robots are made to replace daily human activities. If we want robots to live in our homes, they must have vision system that detects and recognizes *our faces. In that way they will know who is in the home, who is their owner and others. In this scientific research we represent the development of a vision system that detects and recognizes human faces. For the implementation of the system we use OpenCV libraries and for the face detection and recognition we use Haar Cascades. Our system first detects faces in the area, after that it try to recognize them. The results show that our vision system successfully detects and recognizes human faces.*

Keywords: *computer vision, OpenCV, service robot, face detection, face recognition*

1. Introduction

Intelligent service robot's tasks are very complicated. They will have to take care for elderly and disabled, to perform in unhealthy environment or to be robots for surgery [1], [2], [3], [4]. To be intelligent – service robots have to recognize and analyze the environment around. The data from the environment is very important for the decisions that robot will take. To work with people in homes, hospitals and

offices robots have to recognize these people [5]. This task provide security for the people and improves the social abilities of the robots [6]. In case of rubbery the robot can call to police and to detects the faces of the criminals. For social contacts the robots will know who is near them and will call him by name.

Face recognition in general and the recognition of moving people in natural scenes in particular, require a set of visual tasks to be performed robustly. These include Acquisition: the detection and tracking of face-like image patches in a dynamic scene, Normalisation: the segmentation, alignment and normalisation of the face images, and Recognition: the representation and modelling of face images as identities, and the association of novel face images with known models [5].

How do we analyze an image and how does the brain encode it? It was shown by David Hubel [6] and Torsten Wiesel [7], that our brain has specialized nerve cells responding to specific local features of a scene, such as lines, edges, angles or movement. Since we don't see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns. Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing some kind of classification on them.

Face recognition based on the geometric features of a face is probably the most intuitive approach to face recognition. One of the first automated face recognition systems was described in [8]: marker points (position of eyes, ears, nose, ...) were used to build a feature vector (distance between the points, angle between them, ...). The recognition was performed by calculating the euclidean distance between feature vectors of a probe and reference image. Such a method is robust against changes in illumination by its nature, but has a huge drawback: the accurate registration of the marker points is complicated, even with state of the art algorithms. Some of the latest work on geometric face recognition was carried out in [9]. A 22-dimensional feature vector was used and experiments on large datasets have shown, that geometrical features alone my not carry enough information for face recognition.

2. Methods

Our service robot is based on a wheel mobile platform. On the mobile platform are placed all modules of the robot (fig. 1). The robot has main controller, computer, Arduino controller, DC motors for the mobile base, ultrasound and infrared sensors, Kinect sensor and robotic arm manipulator. For the vision system we use Kinect sensor, which is connected to the computer. The location of the Kinect is on top of all other modules. It is the eye of the robot.

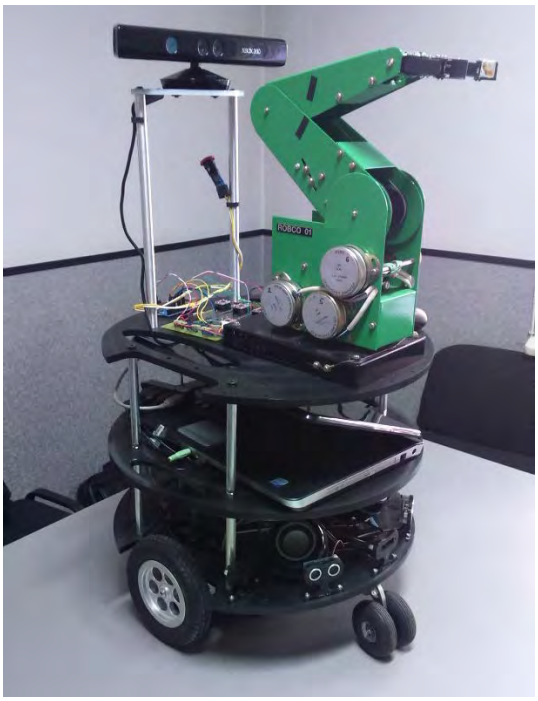

Figure 1. Intelligent service robot.

To control the robot, we use ROS. ROS includes a lot of libraries for robot and sensor control. For our vision system we use ROS with OpenCV. They can work together thanks to the CvBridge, which converts between ROS Image messages and OpenCV images. That allow us to control all of the robot software by ROS. We use RGB camera of the Kinect and computing received signals.

The object detector described below has been initially proposed by Paul Viola [10] and improved by Rainer Lienhart [11].

First, a classifier (namely a cascade of boosted classifiers working with haar-like features) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an

unknown size in the image the scan procedure should be done several times at different scales.

The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (stages) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:

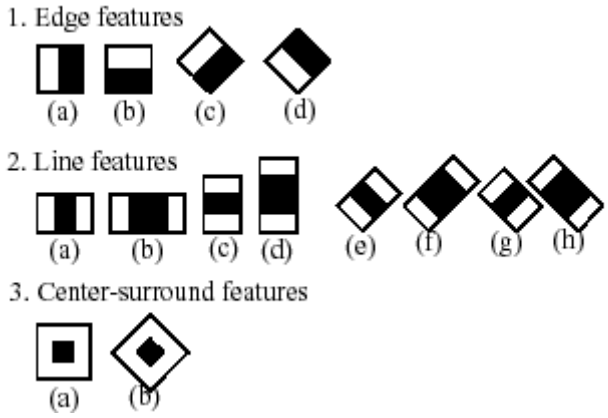

Figure 2. Algorithm for particular classifier.

The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over a rectangular region are calculated rapidly using integral images.

3. Face detection and recognition

3.1. Face detection

Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in

opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect (x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:

   cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

   roi_gray = gray[y:y+h, x:x+w]

   roi_color = img[y:y+h, x:x+w]

   eyes = eye_cascade.detectMultiScale(roi_gray)

   for (ex,ey,ew,eh) in eyes:

      cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

3.2. Face recognition

OpenCV 2.4 now comes with the very new **FaceRecognizer** class for face recognition, which provides a unified access to all face recognition algorithms in OpenCV. Every **FaceRecognizer** is an **Algorithm**, so you can easily get/set all model internals (if allowed by the implementation). **Algorithm** is a relatively new OpenCV concept, which is available since the 2.4 release. I suggest you take a look at its description.

Algorithm provides the following features for all derived classes:

- So called "virtual constructor". That is, each Algorithm derivative is registered at program start and you can get the list of registered algorithms and create instance of a particular algorithm by its name (see **Algorithm::create()**). If you plan to add your own algorithms, it is good practice to add a unique prefix to your algorithms to distinguish them from other algorithms.

- Setting/Retrieving algorithm parameters by name. If you used video capturing functionality from OpenCV highgui module, you are probably familar
  with **cvSetCaptureProperty()**, **cvGetCaptureProperty()**, **VideoCapture::set()** and **VideoCapture::get()**. **Algorithm** provides similar method where instead of integer id's you specify the parameter names as text strings. See **Algorithm::set()** and **Algorithm::get()** for details.

- Reading and writing parameters from/to XML or YAML files. Every Algorithm derivative can store all its parameters and then read them back. There is no need to re-implement it each time.

Moreover every **FaceRecognizer** supports the:

- **Training** of a **FaceRecognizer** with **FaceRecognizer::train()** on a given set of images (your face database!).

- **Prediction** of a given sample image, that means a face. The image is given as a **Mat**.

- **Loading/Saving** the model state from/to a given XML or YAML.

- **Setting/Getting labels info**, that is storaged as a string. String labels info is useful for keeping names of the recognized people.

The currently available algorithms are:

- Eigenfaces (see **createEigenFaceRecognizer()**)

- Fisherfaces (see **createFisherFaceRecognizer()**)

- Local Binary Patterns Histograms (see **createLBPHFaceRecognizer()**)

A more formal description of the Local Binary Patterns Histograms (LBP) operator can be given as:

$$\mathrm{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

This description enables you to capture very fine grained details in images. In fact, the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [12]. The idea is to align an abritrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:

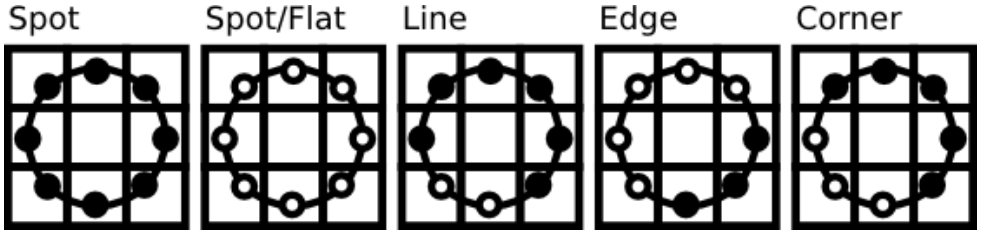

Figure 3. Local Binary Patterns Histograms (LBP) operating

For a given Point (x_c, y_c) the position of the neighbor $(x_p, y_p), p \in P$ can be calculated by:

$$x_p = x_c + R\cos(\frac{2\pi p}{P})$$

$$y_p = y_c - R\sin(\frac{2\pi p}{P})$$

Where R is the radius of the circle and P is the number of sample points.

The operator is an extension to the original LBP codes, so it's sometimes called *Extended LBP* (also referred to as *Circular LBP*). If a points coordinate on the circle doesn't correspond to image coordinates, the point gets interpolated. Computer science has a bunch of clever interpolation schemes, the OpenCV implementation does a bilinear interpolation:

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

By definition the LBP operator is robust against monotonic gray scale transformations. We can easily verify this by looking at the LBP image of an artificially modified image (so you see what an LBP image looks like!):

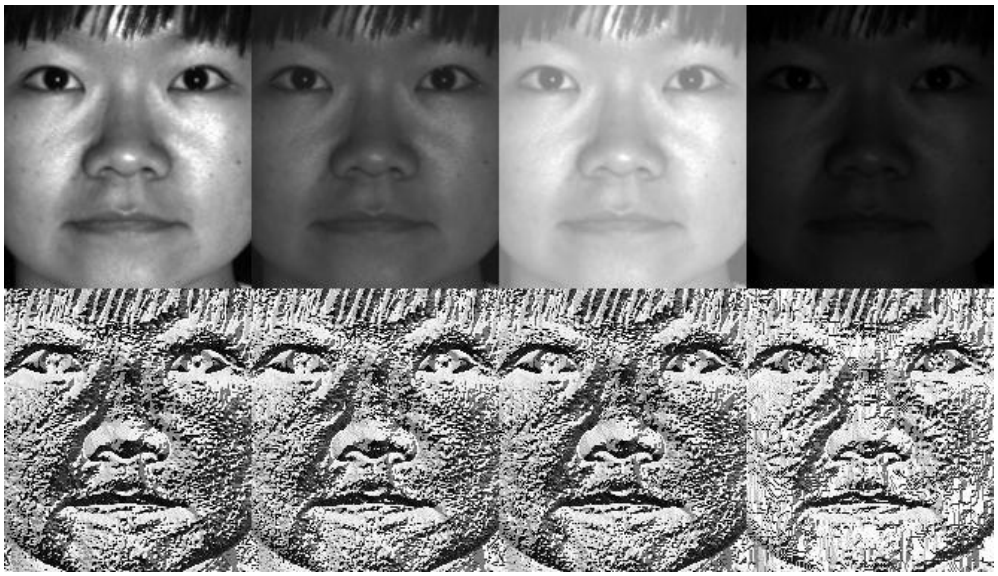

Figure 4. LBP image after computation.

The following lines create an LBPH model for face recognition and train it with the images and labels read from the given CSV file. The LBPHFaceRecognizer uses Extended Local Binary Patterns (it's probably configurable with other operators at a later point), and has the following default values: radius = 1, neighbors = 8, grid_x = 8, grid_y = 8. So if we want a LBPH FaceRecognizer using a radius of 2 and 16 neighbors, call the factory method with: cv::createLBPHFaceRecognizer(2, 16). And if we want a threshold (e.g. 123.0) call it with its default values: cv::createLBPHFaceRecognizer(1,8,8,8,123.0).

```
Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();

model->train(images, labels);
```

The following line predicts the label of a given test image:

```
int predictedLabel = model->predict(testSample);
```

To get the confidence of a prediction call the model with:

```
int predictedLabel = -1;

double confidence = 0.0;

model->predict(testSample, predictedLabel, confidence);
```

```
string result_message = format("Predicted class = %d / Actual class = %d.", predictedLabel, testLabel);

cout << result_message << endl;
```

4. Experiments and results

For the first experiment we use example image with two people and the task for the vision system is to detect the faces and face detection from video stream. By default, in the opencv libraries the detects faces are marked with blue square and the eyes are marked with geen squares.

For the second experiment we use one picture for face model, given by viodeo of usb camera. After that vision system have to recognize that face from picture. Recognized face have to be marked with red square and its eyes have to be marked with black squares.

The result from the first experiment with the picture is shown in figure 5. Our system successfully detects the twue faces. The result from the video stream is shown in figure 6.

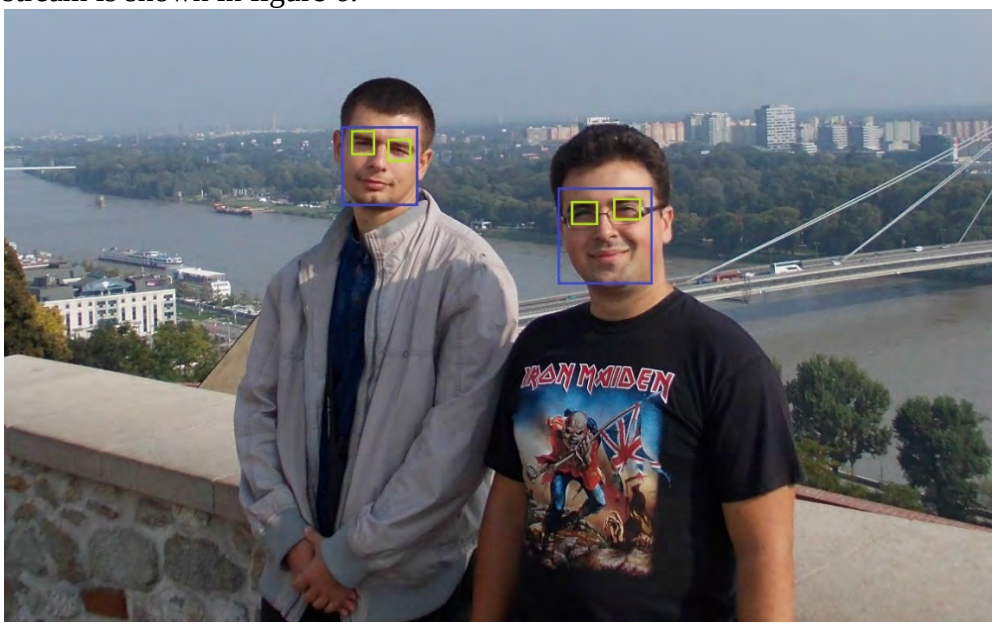

Figure 5. Face detection of image.

The result from the second experiment is shown in figures 6 and 7. Our vision system detects the face in fig. 6 and makes a model of that face. Than the system detects faces in figure 7 and compare the model recognizes the face on the second picture. The recognized face is marked with red square and it’s eyes are in black squares.

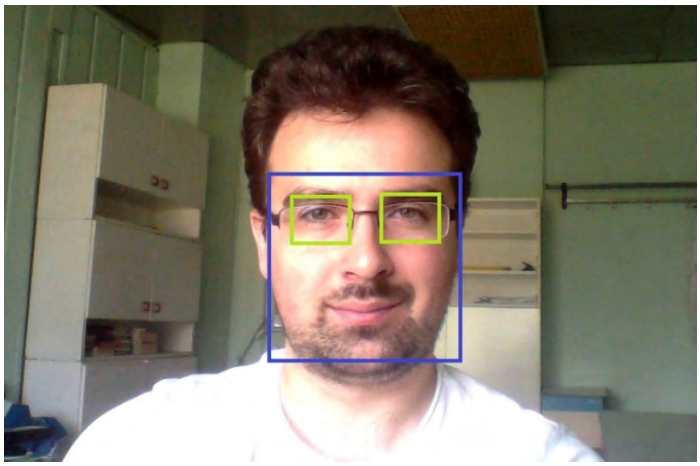

Figure 6. Making a model of a face.

Figure 7. Detecting a face by a given model.

5. Conclusion

In the near future service robots will become assistants in our homes and office. Our vision system for face detection and recognition have to be part of the robots. That systems provides good opportunities for development and functionalities of the robots. Having that system robots will detect all the people around and will recognize the people they belong to. This is important for the people, because when the robots call them by name, people will accept and allow them in their daily live more easily.

Acknowledgments

The research work reported in the paper is partly supported by the projects funded by the Young Scientists Grants, reg. No 102/2016.

References

1. Chivarov, N.; Shivarov, S.; Yovchev, K.; Chikurtev, D.; Shivarov, N. - Intelligent Modular Service Mobile Robot ROBCO 12 for Elderly and Disabled Persons Care; Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on, Smolenice, Slovakia, 3-5 Sept. 2014, Print ISBN: 978-1-4799-6797-1. p. 343-348.

2. Nayden Chivarov, **Denis Chikurtev**, Kaloyan Yovchev, Stefan Shivarov - Cost-Oriented Mobile Robot Assistant for Disabled Care; TECIS 2015, 16th IFAC Conference on Technology, Culture and International Stability, Sozopol, Bulgaria, 24-27 September 2015

3. Ulrich Reiser, Christian Connette, Jan Fischer, Jens Kubacki, Alexander Bubeck, Florian Weisshardt, Theo Jacobs, Christopher Parlitz, Martin H¨agele, Alexander Verl, “Care-O-bot 3 - Creating a product vision for service robot applications by integrating design and technology”, The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009 St. Louis, USA

4. Zheng Li and Yi Ruan, “Autonomous Inspection Robot for Power Transmission Lines Maintenance While Operating on the Overhead Ground Wires”, International Journal of Advanced Robotic Systems, Vol. 7, No. 4 (2010), ISSN 1729-8806, pp. 111-116

5.http://web.archive.org/web/20040829092603/http://www.dcs.qmul.ac.uk:80/research/vision/publications/papers/bmvc97/node1.html

6. https://en.wikipedia.org/wiki/David_H._Hubel

7. https://en.wikipedia.org/wiki/Torsten_Wiesel

8. Kanade, T. Picture processing system by computer complex and recognition of human faces. PhD thesis, Kyoto University, November 1973

9. Brunelli, R., Poggio, T. Face Recognition through Geometrical Features. European Conference on Computer Vision (ECCV) 1992, S. 792–800.

10. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1. IEEE, 2001, pp. 1-511.

11. Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

12. Ahonen, T., Hadid, A., and Pietikainen, M. Face Recognition with Local Binary Patterns. Computer Vision - ECCV 2004 (2004), 469–481.

Обнаружение и распознавание лиц для интеллектуальных обслуживающих роботов

Денис Чикуртев

Институт информационных и комуникационнных технологией - БАН

Email: denis@iinf.bas.bg

Аннотация*: Интеллектуальные сервисные роботы созданы для замены повседневной деятельности человека. Если мы хотим, чтобы роботы жили в наших домах, у них должна быть система зрения, которая обнаруживает и распознает наши лица. Таким образом, они узнают, кто в доме, кто их владелец и другие. В этом научном исследовании мы представляем развитие системы зрения, которая обнаруживает и распознает человеческие лица. Для реализации системы мы используем библиотеки OpenCV, а для обнаружения и распознавания лиц мы используем Haar Cascades. Наша система сначала обнаруживает лица в области, после чего стремится распознает их. Результаты показывают, что наша система зрения успешно обнаруживает и распознает человеческие лица.*

Ключевые слова: *Компьютерное зрение, OpenCV, сервисный робот, обнаруживание лиц, распознавание лиц*