Web application for facial wrinkle recognition

Elena Yu. Tarasova[1], Iryna S. Mintii[1,2]

[1]Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine
[2]Institute of Information Technologies and Learning Tools of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine

Abstract

Facial recognition technology is named one of the main trends of recent years. It's wide range of applications, such as access control, biometrics, video surveillance and many other interactive human-machine systems. Facial landmarks can be described as key characteristics of the human face. Commonly found landmarks are, for example, eyes, nose or mouth corners. Analyzing these key points is useful for a variety of computer vision use cases, including biometrics, face tracking, or emotion detection. Different methods produce different facial landmarks. Some methods use only basic facial landmarks, while others bring out more detail. We use 68 facial markup, which is a common format for many datasets. Cloud computing creates all the necessary conditions for the successful implementation of even the most complex tasks. We created a web application using the Django framework, Python language, OpenCv and Dlib libraries to recognize faces in the image. The purpose of our work is to create a software system for face recognition in the photo and identify wrinkles on the face. The algorithm for determining the presence and location of various types of wrinkles and determining their geometric determination on the face is programmed.

Keywords

pattern recognition, wrinkles, facial landmarks, the Viola-Jones algorithm, the Haar-Like features, OpenCV, DLib, image filters

1. Introduction

It's no secret that today cloud computing has become available to everyone. They combine supercomputer computing power with the accessibility and ease of managing software complexes, and the wide possibilities of modern programming languages and stable software complexes create all the necessary conditions for the successful implementation of even the most complex tasks.

For 50 last years the technology of computer vision evolved from the decision of simple tasks of recognizing symbols to creation of the augmented reality [1]. The most significant examples of using computer vision systems are: 3D-cameras and 3D-sensors; remote control devices; facilities of treatment and recognition of visual pat-terns and symbol information; deep learning on terminal devices and in the cloud; SLAM (Simultaneous Localization and Mapping)

CS&SE@SW 2021: 4th Workshop for Young Scientists in Computer Science & Software Engineering, December 18, 2021, Kryvyi Rih, Ukraine

e.ju.tarasova@gmail.com (E. Yu. Tarasova); irina.mintiy@kdpu.edu.ua (I. S. Mintii)
https://kdpu.edu.ua/personal/oyutarasova.html (E. Yu. Tarasova); https://kdpu.edu.ua/personal/ismintii.html (I. S. Mintii)
0000-0002-6001-5672 (E. Yu. Tarasova); 0000-0003-3586-4311 (I. S. Mintii)

in cars, robots, drones; augmented /virtual reality and perception calculation; the systems for monitoring and security; systems of medical diagnostics and biometrical control.

Face Detection Technology is one of the main trends of recent years. The most ambitious practical application of face recognition with the help of a computer is the creation of anthropomorphic robots that can read the emotional state of the interlocutor.

Algorithms for recognizing a human face are different in complexity. The simplest systems that can establish a person's personality on a photo are looking at the image of the nodal points that form the basic features of the face, and then measure the distances between them. The main parameters are the distance between the eyes, the width of the nose, the depth of the eye, the shape of the cheek-bones, the length of the jaw line. Key indicators in digital terms are a unique code or "face print" that represents a person in the database. For example, to identify a suspect's person, his photo is uploaded to the system and compared to the database.

In modern recognition systems, complex biometric algorithms that analyze unique skin texture, iris, and even vein patterns are used. The process of surface texture analysis works by its very nature similar to facial recognition. The image of a person (namely, her face) is divided into smaller blocks, and algorithms distinguish the fine lines, wrinkles, pores, birthmarks and even the structure of the skin on the texture and iris of the eye – all this appears in the form of a mathematical model. Thanks to this system it is easy to determine the differences even between identical twins.

So, identifying a person in a photo using a computer means solving two different tasks: the first one – to find a face in a photo, the second – to find in the image those features that distinguish a particular person from other people from the database. Face detection algorithms contain at the input of the image, at the output - the coordinates of the found rectangles of the person in the coordinate system of the image. This approach is used in the analysis of photo and video content.

Attempts to teach a computer to find faces in photos began in the early 1970's. But the most important step forward was the creation in 2001 by Viola and Jones [2] of the cascade boost method, which was called the Viola-Jones object detection. Historically, algorithms that work only with the intensity of the image (for example, the value of RGB in each pixel), have a high computational complexity. Papageorgiou et al. [3] considered a set of features based on the Haar wavelets. Viola and Jones [2, 4] adapted this idea and developed the hallmarks of Haar.

Biometric features of a person are unique, therefore identification of an individual is an interesting and difficult task. Face Detection is one of the methods of identifying a person. But over time, the process of aging and changing facial features requires the updating of existing databases, which is a cumbersome procedure. To solve this problem, it is necessary to apply a method that identifies a person, regardless of the effects on the face of aging processes.

In our time, it is necessary to establish personal identification systems, in particular, in educational institutions, in order to protect the lives and health of people, in order to prevent the loss or damage to property in the form of crime agents and intruders. It is known that the cost of personal identification systems from leading soft-ware developers is significant. In addition, technical support is also quite expensive for budget organizations.

The purpose of our work is to create a software system for face recognition in the photo and identify wrinkles on the face.

2. Related work

Facial wrinkles and emotions are often closely related. Researchers claim that people can express 7 basic emotions: anger, joy, surprise, fear, sadness, disgust, and neutrality. But there are also mixed emotions, which are composed of different combinations of the main emotions. Emotions are short-term manifestations of feelings. A person's face has a great ability to show different emotional shades. During the onset of these feelings, the muscles on the face move and noticeable changes in the image occur, for example, wrinkles, a change in the position of the shapes of the eyebrows, eyes, and other parts of the face.

There are three types of facial signals: static (skin tone), long lasting (wrinkles), and instant (raising the eyebrows). The researchers conditionally divided the face into three parts, which have the ability to move independently of other areas of the face (eyebrows; eyes; lower part of the face), and with a combination of the expression of these areas, a mask of each emotion is compiled. Automatic identification of images using a computer is one of the most important directions in the development of artificial intelligence technologies, which allows you to give a key to understanding the peculiarities of the work of human intelligence. Research into methods of automatic emotion recognition allows the computer to assess a person's mood. Facial expression analysis is a challenging area of research in the field of pattern recognition, mainly due to the difficulty in obtaining accurate facial features and facial expressions.

Emotion recognition methods are divided into four large classes: holistic, local recognition, dynamic, geometric. Holistic recognition methods analyze emotions in the face as a whole to find differences between images: Independent Component Analysis (ICA), Principal Component Analysis (PCA), Fisher's Linear Discriminants, Hidden Markov Models, cluster analysis. Local recognition methods are based on the analysis of individual parts of the face: PCA, Neural Networks, Facial Actions Code System (FACS). Dynamic methods take as a basis the changes in the location of parts of the face, while changing various expressions: Point Distribution Model (PDM), Active Shape Model (ASM). Geometric methods take as a basis the shape and position of various parts of a person's face to extract characteristic vectors that convey geometric features of the face. However, the above methods do not always allow us to recognize the user's emotions with a sufficient level of accuracy. Therefore, in this work, the task was set to develop a new combined method for recognizing emotions and detecting wrinkles.

Kazemi and Sullivan [5], Dollár et al. [6], Cao et al. [7] show that the task of determining the face detection on the image can be solved by means of a cascade of regression functions. In the paper of Kazemi and Sullivan [5], which is based on a large number of studies, the methodology for using the regression trees is described to determine the location of key points on the face.

Wrinkles on the face are important signs of aging of the human skin. Wrinkles are three-dimensional imperfections of the skin and are manifested in the form of thin inhomogeneities or cracks in the skin texture. But wrinkles on the face can be masked by the conditions of illumination / reception on 2D images due to the specific nature of the surface texture of the skin and its properties to reflect light. Existing approaches to the study of skin aging on the basis of images are based on the analysis of wrinkles as a texture, rather than as curvilinear cracks.

Jana and Basu [8] propose a methodology for evaluating the real age of a person by analyzing the areas of facial wrinkles. Depending on the location of the wrinkles, an algorithm for c-

average clustering was applied to each facial image. Then, the estimated age was determined based on the analysis of values and average age calculated for each cluster.

Batool and Chellappa [9], Mawale and Chaugule [10] proposed an effective algorithm based on Gabor filters and the morphology of the image to improve the results of locating wrinkles on the face. The authors carried out numerous experiments on sets of different images and the results suggest that the proposed algorithm provides visually better results than other existing systems.

Analyzing the results of many studies, one can conclude that it is convenient to determine wrinkles using filters. The small wrinkles in the corners of the mouth are calculated by the degree of scattering of the brightness of the pixels: the Gauss smoothing filter is applied, and then the received image is compared to the output. If wrinkles were present, they would disappear during smoothing and the difference between the images would be larger. If there was no wrinkle, smooth skin will remain smooth while smoothing. For well-known wrinkles that are quite pronounced (for example, wrinkles on the forehead or on the wings of the nose), the Gabor filter is better suited. This filter is a two-dimensional sinusoid, which is smoothed by a Gaussian window, and very "reacts" to the inherent great wrinkles of sharp changes in brightness in a certain direction.

3. The choice of programming tools

3.1. Select a programming language Python

The reasons to choose Python are: cross-platform; language supports the main programming paradigms used in solving problems; has a large set of useful libraries [11]. In addition, as a complement to Python, there are several popular and feature rich packages that extend the functional language for the full control of clouds. The cloud system itself looks like this: the programmer runs the program code via Django [12] (as in our application) or another framework that is stored along with the web application itself on the server of the cloud service provider, such as Google AppEngine. The control of Django and AppEngine can be done using an accessible interface, but the application itself is configured using Python.

3.2. OpenCV library

The implementation of computer vision can be done using the OpenCV library (Open Source Computer Vision Library) [13, 14]. The library provides tools for processing and analyzing the contents of images, including the identification of objects in photographs (for example, persons and figures of people, text, etc.), object tracking, image transformation, application of machine learning methods, and the discovery of common elements on various images. In this paper, the following functions of the described library were used: the Viola-Jones approach to face detection [2, 4] and functions for working with images, their filtration and transformation.

Recently, the Viola-Jones method, which has long been the main method for detecting objects in an image, has given way to new and improved algorithms. Nevertheless, the relevance of this method remains at the present time.

Of course, the Haar-based cascade classifier (Viola-Jones method) is inferior in performance to the LBP cascade classifier. It is less accurate than a detector based on HOG functions, and even more so a detector based on convolutional neural networks. And yet it has a certain niche when an accuracy higher than that of the LBP cascade is required, but the speed of operation of more accurate detectors is not high enough. The decisive factor for our choice of method was the fact that there are a large number of already trained cascades for the Haar cascade classifier, including in the OpenCV library. Therefore, the speed of this algorithm is very important. This prompted us to choose this particular algorithm as the basis for detecting a face in an image.

The Viola-Jones algorithm (figure 1) can be divided into four main stages:

1. Translate the image into an integral representation.
2. Use the Haar attributes as descriptors.
3. Use the AdaBoost as the classifier.
4. Use the cascade from the classifiers.

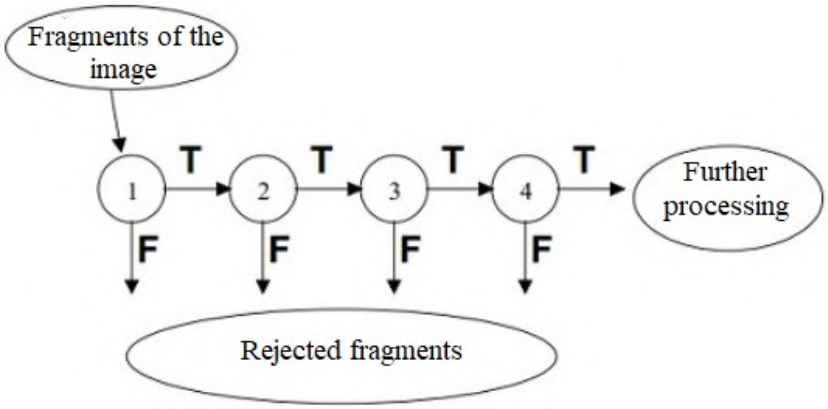

Figure 1: The scheme of the Viola-Jones algorithm.

Let's consider each of these stages in more detail.

In order to reduce the number of operations for computing the Haar-like features, the original image is translated into its integral representation. This view allows you to quickly calculate the total brightness of the rectangle in the image regardless of the size of this rectangle. Integral representation images is a matrix whose size is equal to the original image, and the value of its elements is calculated in the following way:

$$L(x,y) = \sum_{i=0,j=0}^{i\leq x,j\leq y} I(i,j), \tag{1}$$

where $I(i,j)$ is the brightness of the pixel of the original image. Each element of the matrix $L(x,y)$ is the sum of pixels in a rectangle $(0,0)$ $(0,x)$ (x,y) $(0,y)$.

The calculation of such a matrix takes linear time, which is proportional to the number of pixels in a given image. Therefore, the integral representation is calculated completely in one pass.

Let us demonstrate visually the calculation of the area of a rectangle in an integral image using the following example (figure 2).

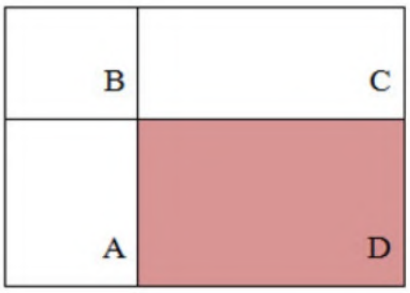

Figure 2: Finding the area of the rectangle $ABCD$ in the integral representation of the image.

Suppose that in the integral image you need to find the area of the rectangle with vertices $ABCD$. This calculation can be reduced to just three operations:

$$S\left(ABCD\right) = L\left(D\right) - L\left(A\right) - L\left(C\right) + L\left(B\right) \tag{2}$$

In the classical Viola-Jones method, the Haar-like features are used, which are presented in figure 3.

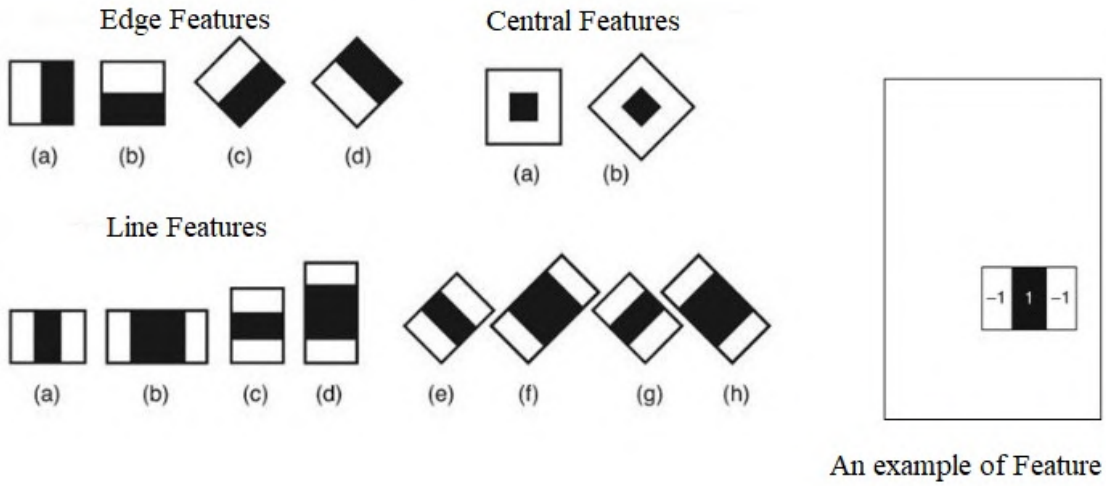

Figure 3: The Haar-like features.

These features are calculated in the following way:

$$F = X - Y \tag{3}$$

where X is the sum of the brightness values of the image pixels covered by the white area, and Y is the black area. Actually, for this purpose the integral representation of the image,

considered above, is used.

The number of different feature variations, calculated for a 24x24 window, is approximately 160,000+, and very many of them are useless for detecting an object. The AdaBoost algorithm is used to select useful features.

Boosting is a complex of methods that contributes to improving the accuracy of analytical models. A model is called "strong" if it admits a small number of classification errors. The "weak" model, on the contrary, makes a large number of errors and does not allow for the reliable separation of classes. Therefore, boosting is a procedure that, as a result of the consistent construction of the composition of machine learning algorithms, seeks to eliminate the shortcomings of all previous algorithms in each new algorithm

The most advanced algorithm for boosting is AdaBoost (Adaptive Boosting), proposed in 1999 by Freund and Schapire [15]. AdaBoost adaptive algorithm because every next the classifier is based on the objects that were incorrectly classified by the past. This algorithm is sensitive to noise in data and outliers, and also less susceptible to retraining

A cascade of "strong" classifiers is a kind of decision tree in which each node is built in such a way as to recognize almost all objects of interest in the image and reject areas in which these objects do not exist. In addition, the nodes of this decision tree are arranged in such a way that the closer the node is to the root of the tree, the smaller the number of descriptors it contains, therefore, it requires less time to make a decision.

This cascade model of classifiers is very well suited for processing images that contain relatively small detected objects. In this case, the classifier constructed in this way can quickly decide that this area does not contain an object and proceed to the next one.

So, summing up the above, the Viola-Jones algorithm involves the sequential execution of four steps:

1. Remove the color and convert the image to the matrix of brightness.
2. Impose one of the square black and white masks on the resulting matrix (the Haar-like features). Inspect the entire image with the mask by changing the position and size.
3. Add the numerical values of the brightness of those matrix cells that fall under the white part of the mask and subtract the values that fall under the black part. If at least in one case the difference between white and black areas is above a certain threshold, take this area of the image for further work, and if not – forget about it, – there is no face here.
4. Repeat from step 2 with a new mask – but only in the area of the image that passed the first exam.

3.3. The method based on localization of the facial landmarks

DLib contains methods of machine learning and various additional tools. Also, this library contains a ready-to-learn model, in this work it is used to find key points on the person's face. Our work uses a face detector from the Dlib library [16]. The detector is used to estimate the location of 68 (x, y) coordinates that are displayed on the face. Indices 68 coordinates are indicated in figure 4. These points are part of the data set on which the face detector predictor was taught.

Finding the key points of the face in the image consists of two stages:

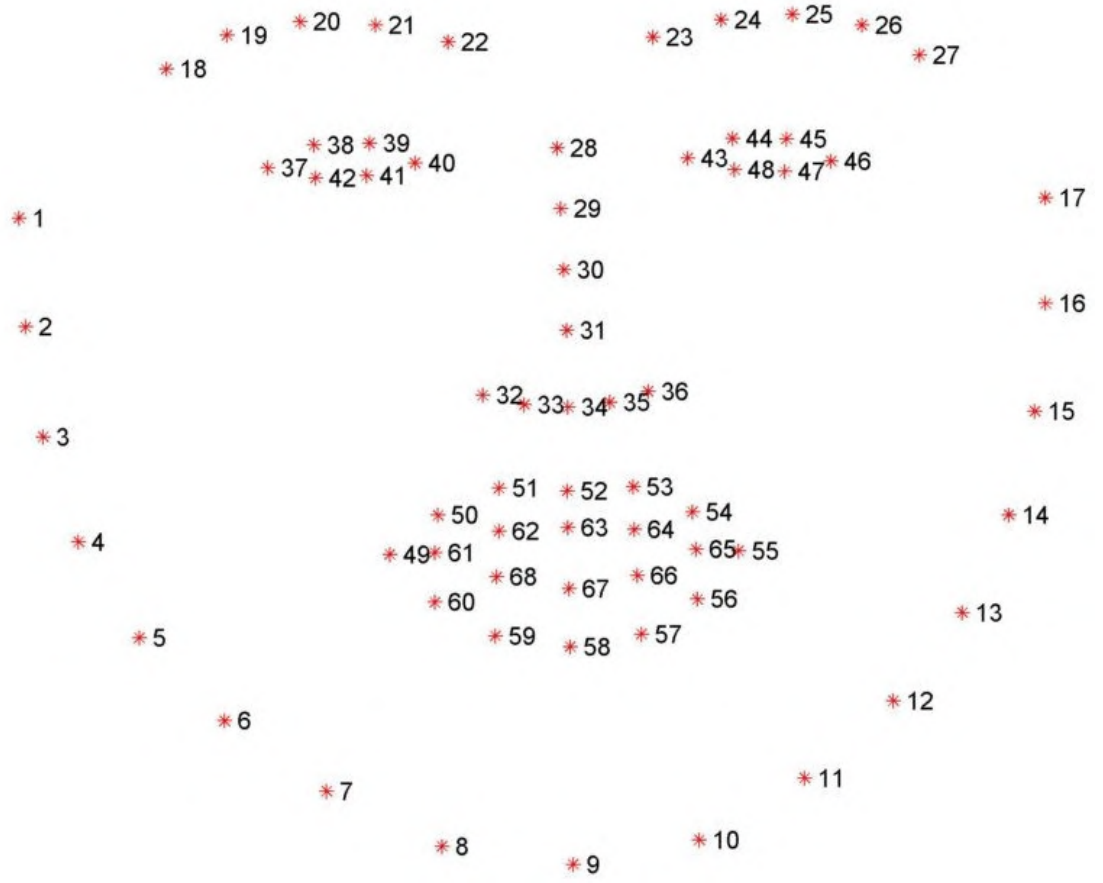

Figure 4: The facial landmarks.

1. First you need to localize the face in the image. This can be done using different methods, it is enough to use an approach that creates a limiting frame around the face.
2. Apply the shape predictor, in particular, the key point detector to get (x, y) coordinates of the faces.

3.4. Algorithm for recognizing wrinkles on the face, depending on their type

Anatomical guidelines were used to implement the wrinkle detection algorithm on the face image to evaluate different types of wrinkles (figure 5).

The texture information method is based on wrinkles, shadows, and other texture elements on the face. Various methods and ways of image filtering were used to calculate texture features on facial areas. To detect fine wrinkles (for example,the forehead area), we used the calculation of a measure of the root mean square dispersion of pixel values over a given area. Two-dimensional Gabor filters were used to detect large wrinkles (for example, nasolabial folds). The two-dimensional Gabor transforms are a harmonic function smoothed by a Gaussian window.

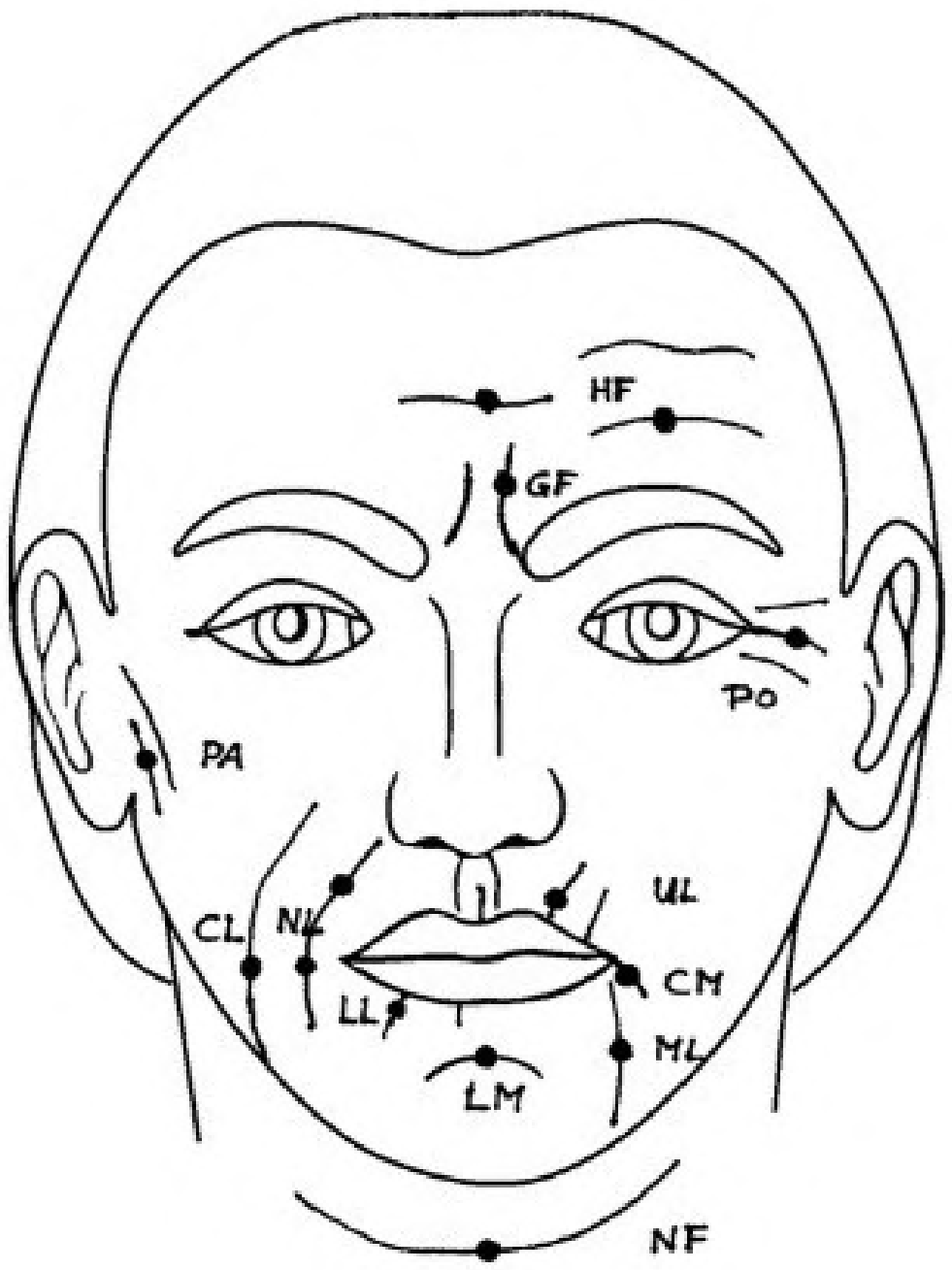

Figure 5: Anatomic reference points for assessment and measurement of wrinkle depth. HF, horizontal forehead lines; GF, glabellar frown lines; PO, periorbital lines; PA, preauricular lines; CL, cheek lines; NL, nasolabial folds; UL, upper radial lip lines; LL, lower radial lip lines; CM, corner of the mouth lines; ML, marionette lines; LM, labiomental crease; NF, horizontal neck folds [17]

Filter and spatial frequencies are calculated depending on the orientation of the wrinkle and its size. According to the control points, the faces are divided into parts where wrinkles are to be found. The filtered area applies to the area under consideration, depending on the location of the area on the face. At the next stage, the contours are determined and their geometric parameters are analyzed. Wrinkle is considered valid (detected) if the contour parameters meet the requirements for each type of wrinkles. For symmetrical wrinkles, an additional double-sided check is used. The results are shown in figure 6.

To train the model, we use the standard Dlib dataset – iBug 300W, which consists of over 6,000 images of different dimensions. The official Dlib dataset consists of a combination of four main datasets: afw, helen, ibug, and lfpw. It is possible to select custom images from social networks and news. The total dataset for training the model was 20,000 images.

Django accepts incoming messages for a request for face recognition and its characteristics in the image. Django sends an image for recognition to one of 20 streams (adjustable in the settings). If there are no free engine threads, then the incoming request is put on hold until the free thread is released or the client is disconnected by timeout. The larger the image, the longer it takes to process, so a normalize function is used to "normalize" the image to the desired size. Models work in shades of gray. The Dlib library processes the file using a trained model. After processing, an array with key points is returned. The wrinkle recognition module geometrically detects the location of the wrinkles and stores an array of coordinates for the lines on the recognized face. The FaceRecognizing class is the entry point to recognition. As soon as it is created, the class initiates a pool of separate recognition threads. A stream is captured for recognition by calling the getFacetRecognizer or getProfileRecognizer methods, which will return the recognition object. Methods can take two parameters: selfRelesase = True, callBack = None. If the selfRelesase = True, then upon completion of recognition the object will return itself to the free thread pool; otherwise, the parent script must do this by calling the releaseFacetRecognizer or releaseProfileRecognizer method. Through the callBack parameter, you can pass a callback function that will be called at the end of recognition.

Link to the archive of materials for this application, located in the repository on GitHub [18]. Fragment of the module code for filling an array of points of different areas of the face (for example, the bridge of the nose) to find wrinkles:

```
#PARRIER WRINKLES
#left nose bridge wrinkle
delta = int(
round((dict(res['left_eye'])[43][0] - dict(res['nose'])[28][0]) / 4, 0))
x1 = dict(res['right_eye'])[40][0]
x2 = dict(res['nose'])[28][0] - delta
# y1 - top point of the eyebrow minus the height of the eye
 y1 = dict(res['right_eyebrow'])[20][1] -
      (dict(res['right_eye'])[42][1] - dict(res['right_eye'])[38][1])
y2 = dict(res['right_eye'])[40][1]
roi = self.getRectFromGrayImage(x1, x2, y1, y2)
res['nosebridgeWrinkleLeft'] = self.__getNosebridgeWrinkle__('left',
roi, dict(res['nose'])[28][0], dict(res['nose'])[28][1], x1, y1,
```

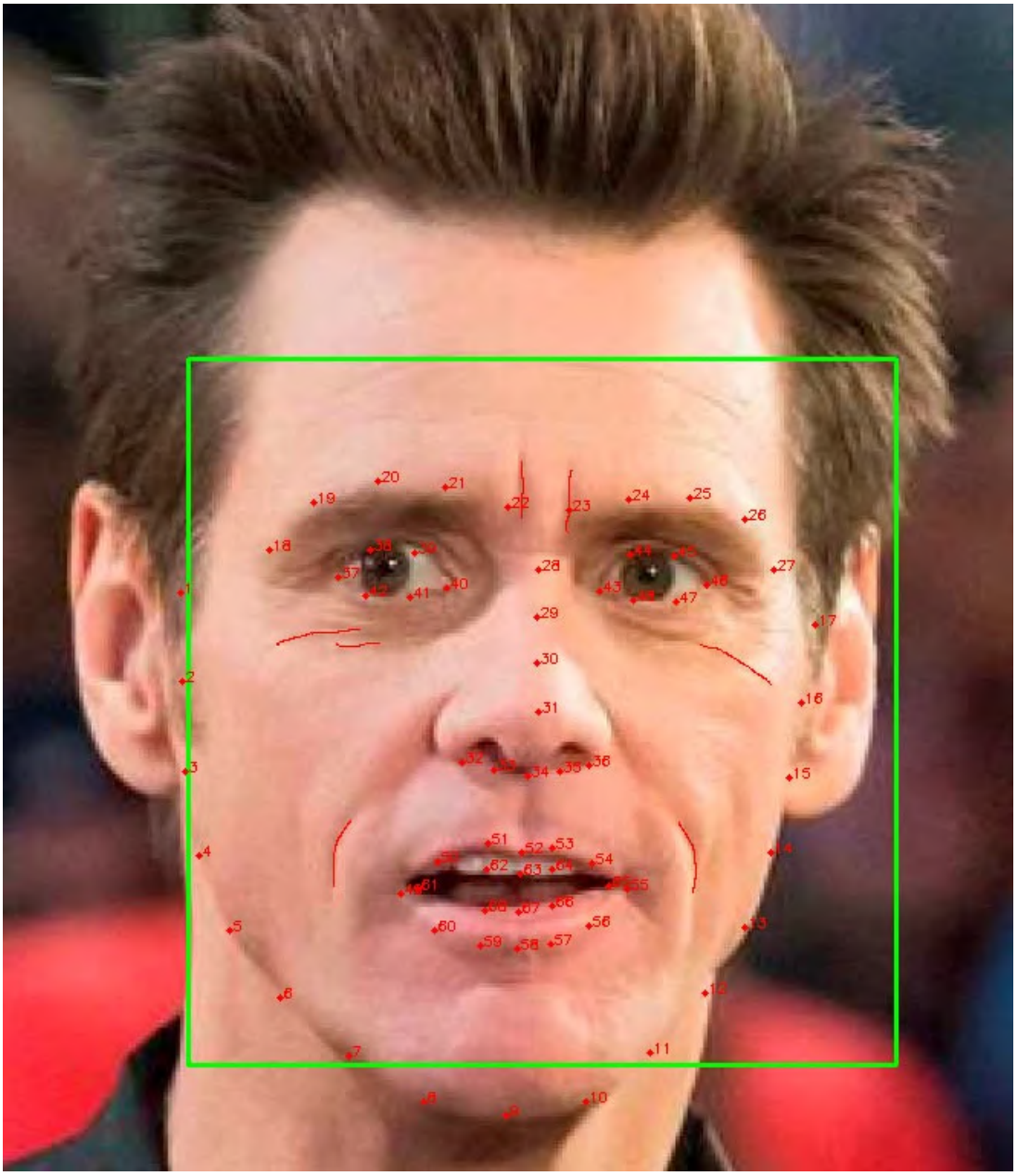

Figure 6: The result of the algorithm for finding wrinkles on the lines of CL-cheeks and CL-nasolabial folds.

```
dict(res['nose'])[28][0], dict(res['nose'])[28][1],
cannyParam1=const.config.getint('Wrinkle.NosebridgeWrinkle',
 'cannyparam1', fallback=const.NOSEBRIDGE_WRINKLE_CANNYPARAM_1),
cannyParam2=const.config.getint('Wrinkle.NosebridgeWrinkle',
'cannyparam2', fallback=const.NOSEBRIDGE_WRINKLE_CANNYPARAM_2))
```

4. Conclusions

Consequently, in this paper, we have shown that different types of wrinkles on the face image are conveniently recognized by using special filters. For example, to detect small wrinkles using the Gaussian filter. For strong wrinkles it is better to use a Gabor filter. In our work an algorithm for determining the presence and location of various types of wrinkles and determining their geometric location on the face is implemented. We created a web application using the Django framework, Python language, OpenCv and Dlib libraries to recognize faces in the image. Using the program you can get geometric parameters of wrinkles. This development can be used to develop a face-to-face identification system for research in the field of cosmetology as the basis for creating a system for recognizing emotions.

References

[1] S. M. Amelina, R. O. Tarasenko, S. O. Semerikov, Y. M. Kazhan, Teaching foreign language professional communication using augmented reality elements, in: S. Semerikov, V. Osadchyi, O. Kuzminska (Eds.), Proceedings of the Symposium on Advances in Educational Technology, AET 2020, University of Educational Management, SciTePress, Kyiv, 2022.

[2] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, 2001, pp. I–I. doi:10.1109/CVPR.2001.990517.

[3] C. Papageorgiou, M. Oren, T. Poggio, A general framework for object detection, in: Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), 1998, pp. 555–562. doi:10.1109/ICCV.1998.710772.

[4] P. Viola, M. J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2004) 137–154. doi:10.1023/B:VISI.0000013087.49260.fb.

[5] V. Kazemi, J. Sullivan, One millisecond face alignment with an ensemble of regression trees, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1867–1874. doi:10.1109/CVPR.2014.241.

[6] P. Dollár, P. Welinder, P. Perona, Cascaded pose regression, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 1078–1085. doi:10.1109/CVPR.2010.5540094.

[7] X. Cao, Y. Wei, F. Wen, J. Sun, Face alignment by explicit shape regression, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2887–2894. doi:10.1109/CVPR.2012.6248015.

[8] R. Jana, A. Basu, Automatic age estimation from face image, in: 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2017, pp. 87–90. doi:10.1109/ICIMIA.2017.7975577.

[9] N. Batool, R. Chellappa, Fast detection of facial wrinkles based on Gabor features using

image morphology and geometric constraints, Pattern Recognition 48 (2015) 642–658. doi:10.1016/j.patcog.2014.08.003.
[10] A. J. Mawale, A. Chaugule, Detecting facial wrinkles based on Gabor filter using geometric constraints, International Journal of Computer Science and Information Technologies 7 (2016) 2021–2025. URL: https://ijcsit.com/docs/Volume%207/vol7issue4/ijcsit2016070476.pdf.
[11] S. V. Shokaliuk, Y. Y. Bohunenko, I. V. Lovianova, M. P. Shyshkina, Technologies of distance learning for programming basics on the principles of integrated development of key competences, CEUR Workshop Proceedings 2643 (2020) 548–562.
[12] Django Software Foundation, individual contributors, Django documentation, 2022. URL: https://docs.djangoproject.com/en/4.0/.
[13] OpenCV team, Home - opencv, 2022. URL: http://opencv.org.
[14] opencv dev team, Face Recognition with OpenCV, 2014. URL: https://docs.opencv.org/3.0-last-rst/modules/contrib/doc/facerec/facerec_tutorial.html.
[15] Y. Freund, R. E. Schapire, A short introduction to boosting, Journal of Japanese Society for Artificial Intelligence 12 (1999) 771–780. URL: https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf.
[16] Dlib C++ Library, 2021. URL: http://dlib.net.
[17] G. Lemperle, R. E. Holmes, S. R. Cohen, S. M. Lemperle, A classification of facial wrinkles, Plastic and reconstructive surgery 108 (2001) 1735–1750. doi:10.1097/00006534-200111000-00049.
[18] Facereader, 2021. URL: https://github.com/charlynka/Facereader/.