IRSTI 28.23.15

U. Sadyk[1], D. Tursyn[2], Sh. Sarsenbay[3], A. Aitimov[4]
[1,2,3,4]Suleyman Demirel University, Kaskelen, Kazakhstan

COMPARATIVE ANALYSIS OF FACE RECOGNITION ALGORITHMS IN THE PROBLEM OF VISUAL IDENTIFICATION

Abstract. The article is devoted to two approaches to face recognition implemented in the visual identification software package. The corresponding algorithms are described, flowcharts and fragments of their software implementation are given, and the results of their applicability are analyzed on the basis of the conducted research.

Keywords: face recognition, EigenFace algorithm, Gabor filter, visual identification.

***

Аннотация. Статья посвящена двум подходам к распознаванию лиц, реализованным в программном комплексе визуальной идентификации. Описаны соответствующие алгоритмы, приведены блок-схемы и фрагменты их программной реализации, а также проанализированы результаты их применимости на основе проведенного исследования.

Ключевые слова: распознавание лиц, алгоритм EigenFace, фильтр Габора, визуальная идентификация.

***

Аңдатпа. Мақала визуалды сәйкестендіру бағдарламалық жасақтамасында жүзеге асырылған тұлғаны танудың екі тәсіліне арналған. Тиісті алгоритмдер сипатталады, блок-схемалар мен оларды бағдарламалық қамтамасыз етудің фрагменттері келтіріліп, жүргізілген зерттеулер негізінде оларды қолдану нәтижелері талданады.

Аңдатпа: тұлғаны тану, EigenFace алгоритмі, Габор фильтрі, визуалды идентификация.

I. Introduction

The identification system based on voice and visual data is designed to recognize individuals in crowded places, in control zones, and at strategic facilities. The functionality of the system allows you to increase the reliability of identification due to its multi-level structure based on the use of various biometric characteristics. An important part of this structure is the face recognition block, which implements two independent recognition algorithms that differ in accuracy, speed, and resource intensity.

Differences in the characteristics of the algorithms allow us to evaluate the feasibility of using them in a particular situation, while simultaneous use will increase the reliability of identification. Next, we will consider the algorithms themselves, some aspects of their implementation and integration into the system. The code snippets are written in C#.

II. Overview of Algorithms

Image analysis algorithms

The main idea of the EigenFace algorithm is to find the "average face", i.e. a generalized and averaged version of all user photos in the database. Using the resulting "average face", a "difference face" is found for each user's photo, i.e. the difference between it and the "average face". The resulting "difference face" represents those facial features that are least common in the rest of the images in the database. When an image arrives at the subsystem input, a "difference face" is calculated for it and compared with each "difference face" in the database using the Euclidean distance:

$$p(x, y) = \|x - y\| = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2},$$

where *x, y* are vectors of dimension *n*.

The sequence of image processing using the EigenFace algorithm:

Step 1: Getting an array of photos, which is done using Haar attributes and the OpenCV library. The image is stored in the database as a matrix of *I [m, n]* color values of its pixels.

$$I = \_img.HaarDetectObjects(\_Cascade,\ \_Storage).$$

Step 2: Getting the matrix of all images. Each photo is converted to a vector line by line:

$$ImageVector\,[\,i * m + j\,] = Incoming\,[\,i,\ j\,],$$

where Incoming is the original image in matrix form, and ImageVector is in vector form. The resulting image vectors are then written to the Common matrix by columns.

Step 3: Calculating the average face for the image database:

$$A_{vg}[i] = \frac{1}{N} \sum_{j=0}^{n-1} Common\,[i, j],$$

where Avg is the average face vector, N is the total number of images in the database.

```
// A is initially filled with 0
for (int i = 0; i < m; i++)
{
        for (int j = 0; j < n; j++)
        {
        A[i,j] += I[i, j];
        }
}
for (int i = 0; i < iLength; i++)
{
A[i,j] /= N;
}
```

Step 4: calculating the matrix of difference images according to the expression Dif [i, j] = Common [i, j] – Avg [i]

```
for (int i = 0; i < m; i++)
{
            for (int j = 0; j < n; j++)
            {
                      Dif[i,j] = I[i, j] — A[i,j];
            }
}
```

Step 5: Calculate the covariance matrix Cov and its eigenvectors v:

$$Cov = Dif^T * \; Dif\,.$$

Step 6: calculate the eigenvectors u of the matrix of difference images Dif and the vector of their weights *w*:

$$u = Dif * v\,,$$

$$w = u * Dif\,.$$

The vector w of the weights of the eigenvectors of each image is stored in the database. Further, recognition is carried out on their basis.

Each input image is converted using a similar algorithm. Recognition consists in calculating the Euclidean distance for each pair (wBase, wInc) and selecting the minimum one among them. Here wBase is the weight vector of the eigenvectors of the i-th image from the database, and wInc is the input image.

$$RecogIndex = \; (p(\,wBase_i\,,\, wInci)),\; i \in [1; N].$$

Recognition algorithm based on Gabor filters

The Gabor filter (Fig. 1) is a linear electronic filter, the pulse transient characteristic of which is defined as a harmonic function multiplied by the Gaussian:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right) cos\left(2\pi\frac{x'}{\lambda} + \psi\right),$$

where x' = x cos θ + y sin θ; y' = –x sinθ + y cosθ.

In this equation, λ is the wavelength of the cosine multiplier, θ determines the orientation of the normal of the parallel bands of the Gabor function in degrees, ψ is the phase shift in degrees, and γ is the compression coefficient that characterizes the ellipticity of the Gabor function.

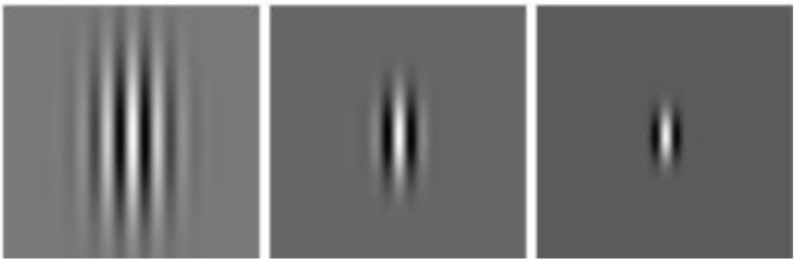

Figure 1. Examples of Gabor filters

The Gabor filter is an effective means of forming local features of a digital image. Thus, by conducting a two-dimensional convolution of the image with the Gabor kernel of the reconciliation at the point x, y, we obtain a local characteristic of this point. Since the Gabor filter is resistant to scaling and rotation operations, it can be used for user identification tasks. To achieve this goal, you need to follow several steps. Step 1: prepare the filter. Before using it, you need to get its core. This paper uses an 8×8 pixel filter core:

$$Filter\,[i + 4,\, j + 4] = g(x, y; \lambda, \theta, \psi, \sigma, \gamma),\ i \in [-4; 4],\ j \in [-4; 4].$$

Step 2: the search for reference areas in the image is performed using the Viola-Jones algorithm [1]. The left eye, right eye, nose, and mouth were selected as reference areas.

faces = img.HaarDetectObjects(_cascadeMouth, _storage);

faces = img.HaarDetectObjects(_cascadeRightEye, _storage);

faces = img.HaarDetectObjects(_cascadeLeftEye, _storage);

faces = img.HaarDetectObjects(_cascadeNose, _storage);

Step 3: apply the filter at the reference points in the image and get the feature vector, which is formed from the values of the Gabor filters at the reference points [2]. The value of the Gabor filter is calculated using the convolution operation at the point [x, y]:

$$Img\,[x, y] = \sum_{i=x-n/2}^{x+n/2} \sum_{j=y-n/2}^{y+n/2} Img[i, j] * Filter\left[i - \left(x - \frac{n}{2}\right), j - \left(y - \frac{n}{2}\right)\right],$$

where Img is the source image, and n is the width of the filter core.

The corners and centers of the reference areas obtained in step 2 were selected as reference points. The feature vector is calculated for each image in the database and stored with it. Then, based on these vectors, the recognition process is performed.

For the input image, a feature vector is also calculated, for which the Euclidean distance to each vector in the database is found in turn. A suitable image is the one that has the feature vector with the smallest distance to the input one.

Comparison of algorithm characteristics

The applicability of facial recognition algorithms in various situations can be evaluated based on the results of the conducted research. Figures 3 and 4 show the results of the algorithm accuracy study. Here, accuracy is understood as the ratio of the number of correct facial recognition results to the total number of attempts. The algorithms were tested for different distances between the camera and the user. The EigenFace algorithm is more accurate, and as the distance to the camera decreases, its accuracy increases.

The speed of recognition of a single frame depends on the number of photos in the database (Fig. 5). The results show that the algorithm using Gabor filters works faster. This is due to the fact that the dimension of the analyzed space in it is significantly smaller (dim 20) than in EigenFace (dim 10000).

It is important to note that due to the large number of calculations for each comparison, the EigenFace algorithm is very sensitive to the database volume, which is not the case with the algorithm using Gabor filters.

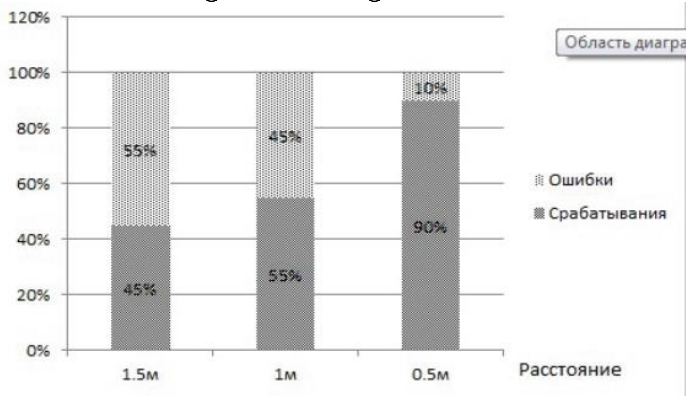

Figure 3. Accuracy of the EigenFace algorithm

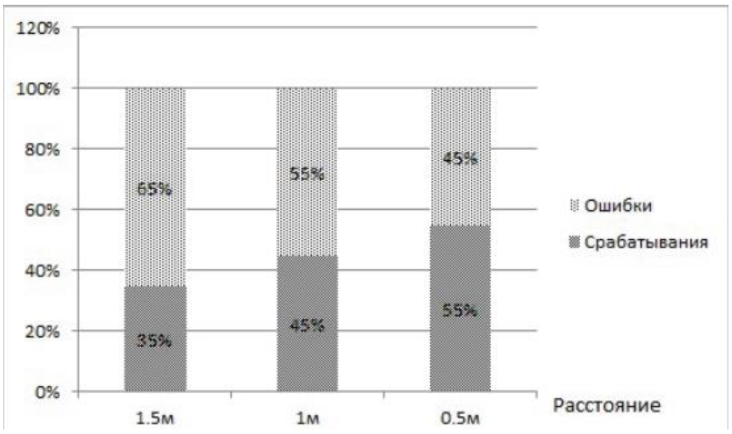

Fig. 4. Accuracy of the algorithm using Gabor filters

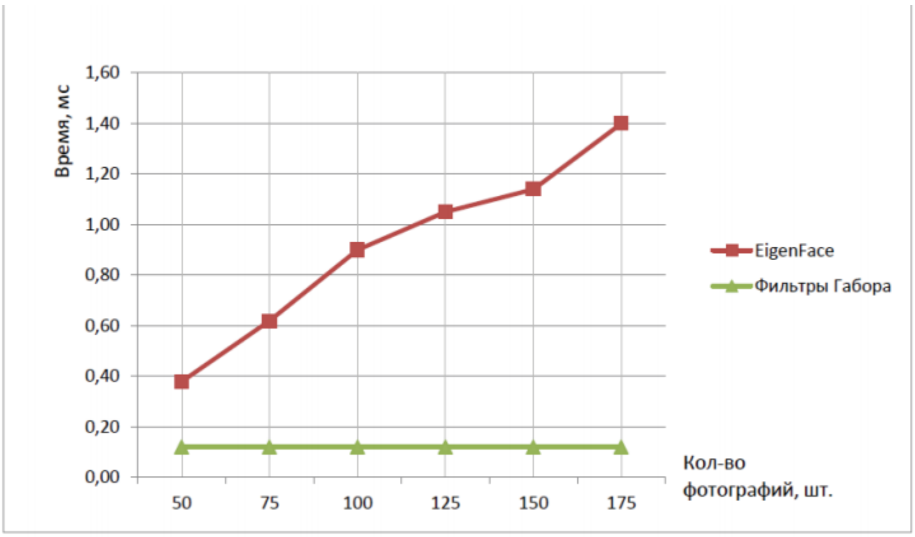

Fig. 5. Dependence of the algorithms' running time on the number of images in the database

The values of the indicators FAR (the proportion of false positives) and FRR (the proportion of false failures) for the developed system when using different algorithms for recognizing the user from the image are shown in Table 1.

III. Results

As a result made a comparison of FAR and FRR values, as show in Table 1

Table 1

	FAR	FRR	Triggers
EigenFace	13%	6%	81%
Gabor Filters	26,5%	0%	73,5%

IV. Conclusion

For the experiment, a system was used in which all results that did not exceed a certain threshold value were considered suitable. This approach allows you to get a selection of results instead of one, which increases the probability of a match. In addition, varying this value makes it possible to tighten or weaken the recognition criteria. Therefore, the speed of the algorithm using Gabor filters allows us to recommend it for face recognition in large streams for primary selection. However, insufficient accuracy requires secondary processing of the selected images by a more precise algorithm, such as EigenFace. In turn, the EigenFace algorithm can successfully function in special control systems with a small flow of users, since it requires significant resources.

References

1 Viola, P., Jones, M.J. Robust real-time face detection. International Journal of Computer Vision, vol. 57, no. 2, 2004, pp. 137–154.

2 Gonzalez, R., Woods, R. Digital image processing. Moscow, Technosphere, 2005, 1072 p.