I.P.SHERONOV[1], S.N.PETROV[1]

OPENCV-BASED APPLICATION FOR FACE DETECTION AND RECOGNITION

[1]Educational institution «The Belarusian State University of Informatics and Radioelectronics», Minsk, Republic of Belarus

Face recognition is the automatic localization of a human face in an image or video and, if necessary, identification of a person's identity based on available databases. Interest in these systems is very high due to the wide range of tasks that they solve. In particular, face recognition is widely used in security systems and data protection systems, as a way to verify personnel who have access to sensitive enterprise data. Today, machine learning methods, in particular Deep learning methods, have been widely used in the field of pattern recognition (especially face recognition).

The General principle of operation can be divided into four steps:

– Face detection, when a face is detected in the frames of a photo or video camera.

– Face analysis, which consists of scanning the nodal points on the detected face.

– Converting an image into data, or converting into unique numeric data called a faceprint.

– Search for data matches, during which the face print is compared with a database of photos with identifiers that can be compared.

This identification method can be implemented using the Python programming language with the OpenCV library (Open Source Computer Vision Library). OpenCV is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos and many other purposes.

To build our OpenCV face recognition pipeline, we'll be applying deep learning in two key steps: to apply face detection, which detects the presence and location of a face in an image, but does not identify it; to extract the 128-d feature vectors (so called embeddings) that quantify each face in an image. For working with data arrays, the Python programming language is used, including the scikit-learn library for extracting features from a data set

To implement such a project you need to work out 3 stages:

– Face detection and data collection.

– Training of Recognizer.

– Face recognition.

First of all, we need to capture a face to compare it with a new face captured in the future. The most common method of face detection uses the Haar cascade classifier [1].

Initially, the algorithm requires a lot of positive images (faces), and negative images (without faces) to train the classifier. Then you need to extract functions from the classifier. The OpenCV library comes with a trainer and detector that can recognize not only faces, but also any other objects similar to the specified parameters [2].

A data set is created that stores groups of photos in gray with the part that was used for face detection. A set of 30 samples for each identifier can be considered optimal.

On the training stage takes all user data from our dataset and the OpenCV – Recognizer library instructor. This is done directly using a specific OpenCV function. To train a face recognition model with deep learning, each input batch of data includes three images: the anchor (current face); the positive image (image of person with the same identity with anchor); the negative image (does not have the same identity

with anchor). The neural network computes the 128-d embeddings for each face and then tweaks the weights of the network.

On the recognition stage, we can see the face on the camera, and if the person was photographed and trained earlier, our Recognizer will do a «processing» that returns its ID and index, which shows how confident the Recognizer is in this feature.

REFERENCES

1 Face Detection using Haar Cascades [Electronic resource] / link https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html / Date of access: 08.09.2020

2 Cascade Classifier Training [Electronic resource] / link https://docs.opencv.org/ 3.3.0/dc/d88/tutorial_traincascade.html / Date of access: 08.09.2020