Kliuiev I. O., Master's Student,
Vakaliuk T. A., Dr. Sc., prof.,
Zhytomyr Polytechnic State University

METHODS OF FACE COMPARISON IN DIFFERENT PHOTOS USING COMPUTER VISION

The methods of face recognition in photos for subsequent comparison represent an important and relevant problem with numerous applications in various fields, including biometric authentication. Typically, face recognition is accomplished using computer vision tools and convolutional neural networks.

We will employ this technology in a dating and friend-finding application to determine if the photos in a profile depict the same person who underwent verification. To compare faces in different photos, three steps are necessary:

Extract facial feature vectors from the reference photo.

Extract facial feature vectors from the comparison photo.

Calculate the percentage match of these feature vectors (Euclidean distance between the feature vectors).

Convolutional neural networks are utilized to extract facial parameters. In our case, we will use the ResNet network from Microsoft without the classification module.

To implement this task, we will use the dlib library, which includes a pre-trained neural network that produces feature vectors in such a way that feature vectors from photos of the same person are close to each other, while vectors from photos of different individuals are far apart. In the dlib library, these feature vectors are called descriptors. Essentially, the dlib library is written in C; however, it also has API packages for other programming languages. In our case, we will use the Python package.

The algorithm for face recognition and comparison using the dlib library starts with loading pre-trained models, such as a model for predicting facial landmarks (68 key points) [1] and a model for face recognition [2]. Next, a function is defined that takes the path to a photo as input, reads the image, detects faces in the photo, optionally displays their coordinates, obtains facial key points using the shape prediction model, and computes a face descriptor using the face recognition model.

Then, the created function is called twice to extract face descriptors from two images: the reference image ("reference.jpg") and the one to be compared ("photo_to_compare.jpg"). Subsequently, a function for calculating the Euclidean distance is used to determine the numerical difference between the face descriptors.

```
import dlib
from skimage import io
from scipy.spatial import distance
# Use models:
sp=dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
facerec= dlib.face_recognition_model_v1(
'dlib_face_recognition_resnet_model_v1.dat')
detector =dlib.get_frontal_face_detector()
# To avoid code repetition, we put the descriptor in the function
def getFaceDescriptorFromPhoto(photo):
img=io.imread(photo)
    dets = detector(img,1)
# Optionally, display all the faces found in the photo on the screen
for k, d in enumerate(dets):
print("Face found {}: Left: {} Top: {} Right: {} Bottom: {}".format(
          k,d.left(),d.top(),d.right(),d.bottom()))
# Getting the shape of the face
      shape =sp(img, d)
# Getting a face descriptor
return facerec.compute_face_descriptor(img, shape)
# Getting a face descriptor from the reference photo
face_descriptor1 =getFaceDescriptorFromPhoto('reference.jpg')
# Get the face descriptor from the compared photo
face_descriptor2 =getFaceDescriptorFromPhoto('photo_to_compare.jpg')
# Calculate the Euclidean distance between descriptors
distance =distance.euclidean(face_descriptor1, face_descriptor2)
if distance <0.6:
print('The photo shows the SAME person')
else:
print('The photo shows DIFFERENT people')
```

Fig. 1. Basic face recognition and comparison algorithm

Afterwards, the calculated distance is compared with a threshold value of 0.6. If the distance is less than this value, a message is printed stating that the photos show the same person. Otherwise, a message is printed indicating that the photos show different people.This algorithm provides a straightforward mechanism for comparing faces based on their descriptors, and the threshold value allows controlling the algorithm's sensitivity to differences between faces.The code itself is shown in Fig. 1.

References

1. GitHub – italojs/facial landmarks recognition, Available: https://github.com/italojs/facial-landmarks-recognition.

GitHub – ageitgey/face recognition models: Trained models for the face recognition python library Available: https://github.com/ageitgey/face_recognition_models/tree/master.