DOI: https://doi.org/10.15276/aait.06.2023.20
UDC 004.032.26:004.946

Efficient face detection and replacement in the creation of simple fake videos

Oleksii I. Sheremet [1)]
ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus Author ID: 57170410800
Oleksandr V. Sadovoi [2)]
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
Denys V. Harshanov [3)]
ORCID: https://orcid.org/0009-0008-6257-468X; denysharshanov3@gmail.com
Oleh S. Kovalchuk[1)]
ORCID: https://orcid.org/0009-0009-5521-6451; 3289560@gmail.com
Kateryna S. Sheremet [1)]
ORCID: https://orcid.org/0000-0003-3783-5274; artks@ukr.net. Scopus Author ID: 57207768511
Yuliia V. Sokhina [4)]
ORCID: https://orcid.org/0000-0002-4329-5182; jvsokhina@gmail.com. Scopus Author ID: 57205445522
[1)] Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, 84313, Ukraine
[2)] Dnipro University of Technology, 19, Dmytra Yavornytskogo Ave. Dnipro, 49005, Ukraine
[3)] Kharkiv National University of Radioelectronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine
[4)] Dniprovsky State Technical University, 2, Dniprobudivska Str. Kamyanske, 51918, Ukraine

ABSTRACT

Face detection and facial recognition technologies are among the most intensively studied topics within the field of computer vision, owing to their vast application potential across a multitude of industries. These technologies have demonstrated practical applicability in varied contexts such as identifying suspicious individuals in crowded urban spaces, real-time recognition of smartphone owners, creating compelling deepfakes for entertainment applications, and specialized applications that modify the movements of facial features such as the lips or eyes. With the current state-of-the-art advancements in hardware and software technology, today's technological infrastructure provides more resources than are necessary for video streaming. As a result, simple face recognition systems can be implemented without the need for high-cost server instances that require specified pre-trained models. This abundance of resources is changing the landscape of face recognition, and the discussion within this paper will revolve around these emerging paradigms. The primary focus of this article is an in-depth analysis of the key concepts of face detection in streaming video data using prominent pre-trained models. The models under discussion include HRNet, RetinaFace, Dlib, MediaPipe, and KeyPoint R-CNN. Each of these models has its strengths and weaknesses, and the article discusses these attributes in the context of real-world case studies. This discussion provides valuable insights into the practical applications of these models and the trade-offs involved in their utilization. Moreover, this paper presents a comprehensive overview of image transformation techniques. It introduces an abstract method for affine image transformation, an important technique in image processing that changes the geometric properties of an image without affecting its pixel intensity. Additionally, the article discusses image transformation operations executed through the OpenCV library, one of the leading libraries in the field of computer vision, providing a highly flexible and efficient toolset for image manipulation. The culmination of this research is presented as a practical standalone system for image replacement in video. This system leverages the RetinaFace model for inference and employs OpenCV for affine transformations, demonstrating the concepts and technologies discussed in the paper. The work outlined in this article thereby advances the field of face detection and recognition, presenting an innovative approach that makes full use of contemporary hardware and software advances.

Keywords: Deepfake; affine transformation; face detection; video processing; alpha channel; binary masks

For citation*: Sheremet O. I., Sadovoi O. V., Harshanov D. V., Kovalchuk O. S., Sheremet K. S., Sokhina Yu. V. "Efficient face detection and replacement in the creation of simple fake videos".** ***Applied Aspects of Information Technology. **2023; Vol. 6 No.3: 286–303.**
DOI: https://doi.org/10.15276/aait.06.2023.20

INTRODUCTION

Video manipulation, a rapidly evolving field that intertwines technology and visual artistry, has witnessed a surge of interest and applicability over the past several years. There is an escalating demand for precise, high-performing techniques capable of face substitution within videos, a specialized procedure that hinges upon several core steps: initial face detection, precise face alignment, and finally, the actual face replacement.This complicated procedure of face substitution in videos holds a crucial role in digital manipulation, with a significant influence spreading across numerous sectors. The breadth and depth of the impact of face replacement algorithms within video technology are considerable and far-reaching, sparked by an array of potential applications.

In the bustling world of entertainment, the significance of face replacement algorithms is indisputable.

These techniques are employed for face substitution in videos to fabricate incredibly lifelike deepfakes or to carry out face replacements within movies and television series. The technology holds an incredible potential for the augmentation of the entertainment industry's realism, crafting more convincing special effects, making it possible for filmmakers to achieve feats that were once thought to be unattainable or extremely costly. Furthermore, in circumstances involving sensitive or private video content, this technology plays an instrumental role in protecting an individual's anonymity.

Aside from the entertainment industry, another critical application of face replacement algorithms is in the field of security and surveillance. Leveraging these algorithms, faces in surveillance footage can be skillfully obscured, hence offering a robust tool for protecting the privacy of individuals. This becomes particularly valuable in environments where privacy concerns are heightened - for instance, in public spaces where multiple individuals are captured on camera, or in highly sensitive settings that necessitate strict confidentiality.

Such a broad array of applications underscores the importance of face replacement algorithms in videos, highlighting their wide-ranging utility in a world that's increasingly sensitive to issues of privacy and more reliant on visual technology. Whether it's the world of entertainment or the more sensitive spheres of security and surveillance, these techniques offer possibilities that can shape the future of video manipulation.

LITERATURE REVIEW

There are several methods for replacing faces in videos using Python/C++, including HRNet, RetinaFace, Dlib, MediaPipe, KeyPoint R-CNN and OpenCV. Each of these methods has advantages and disadvantages, and choosing the best method for a particular project will depend on specific requirements and constraints.

HRNet [1] is a high-resolution network for object detection and face alignment. It uses deep learning algorithms to accurately detect and align faces in video. To replace faces in video with HRNet in Python, an implementation is available in popular deep learning libraries such as PyTorch or TensorFlow [2, 3].

RetinaFace [4] is a deep learning-based face detection and alignment algorithm developed by a research group at Megvii Technology. It uses a single neural network to detect and align faces, making it fast and accurate [5]. To replace faces in videos with RetinaFace in Python, you can use the RetinaFace implementation available in popular deep learning libraries such as PyTorch or TensorFlow [6].

Dlib [7] is a widely used library for computer vision and image processing. It provides a number of tools for face detection and recognition, including facial landmark detection and face alignment. You can use the face detection and face alignment features available in the Dlib library to replace faces in videos using Dlib in Python [8, 9].

MediaPipe (reimplementation of BlazeFace) [10, 11] is a library developed by Google that is optimized for real-time video processing. It provides high accuracy face detection and uses machine learning to improve accuracy. To replace faces in videos using MediaPipe in Python, you can use the face detection and alignment features available in the MediaPipe library [12]. It allows detecting 468 3D points of Face Mesh.

KeyPoint R-CNN [13] is a deep learning-based object detection algorithm that can be used for face detection and alignment. It is particularly well suited for face detection in complex and dynamic scenes. To replace faces in video with KeyPoint R-CNN in Python, you can use the KeyPoint R-CNN implementation available in popular deep learning libraries such as PyTorch or TensorFlow [14].

OpenCV is a widely used library for computer vision and image processing [15, 16]. It provides a number of tools out the box for face detection and recognition, including facial landmark detection and face alignment. To replace faces in video with OpenCV, it can be used the face detection and face alignment features available in the OpenCV library. In general, this library also utilizes ffmpeg for video processing as the result the performance of processing streams will be at high level [17].

There are several methods for replacing faces in video, including HRNet, RetinaFace, Dlib, MediaPipe, KeyPoint R-CNN, and OpenCV. Choosing the best method for a particular project depends on specific requirements and constraints, such as accuracy, speed, and computational requirements [18]. With the right approach and tools, you can achieve accurate and efficient face substitution in videos using Python or C++ languages.

PURPOSE AND OBJECTIVES OF THE RESEARCH

The purpose of this research is to enhance the efficiency of face detection and replacement in the creation of simple fake videos within video streaming contexts, by enhancing the use of advanced neural network technologies and pre-trained models such as RetinaFace, Mediapipe, and OpenCV, thereby

enabling faster processing, and broader adaptability in diverse real-world scenarios.

To achieve this aim, the following tasks are formulated:

– exploring the foundational principles of face detection in streaming video data using various pre-trained models (HRNet, RetinaFace, Dlib, MediaPipe, and KeyPoint R-CNN) within the context of simple fake video generation;

– analyzing the strengths and limitations of each pre-trained model when applied to the creation of simple fake videos;

– utilizing the OpenCV library to conduct affine image transformations and other essential image transformation operations integral to fake video creation;

– evaluating the speed and computational demands of these face detection methods with an emphasis on their efficiency in fake video generation;

– demonstrating the practical application of the RetinaFace model in replacing images in video via OpenCV affine transformation, specifically targeted at simple fake video generation.

FINDING THE FACIAL OVAL IN A CUSTOM PHOTO

Finding points on the face is an important step in the process of face substitution in videos. Facial landmarks, such as the eyes, nose, and mouth, are used to align the face and to determine the positions of the facial features. The positions of these landmarks are used to determine the transformations needed to replace the face in the video.

There are several methods available for finding points on the face in Python, including deep learning-based object detection algorithms and traditional computer vision algorithms. Deep learning-based object detection algorithms, such as RetinaNet [19] and YOLO [20, 21], use deep learning algorithms to accurately detect facial landmarks in real-time. Traditional computer vision algorithms, such as Dlib and OpenCV, use a combination of feature detection and pattern recognition algorithms to detect facial landmarks.

In addition to finding facial landmarks, it is also possible to find additional points on the face, such as the contours of the face and the position of the eyes, nose, and mouth. These additional points can be used to improve the accuracy of the face substitution and to create a more realistic result.

Various options for character detection in the video were considered: Keypoint R-CNN in Fig. 1, HRNet, Dlib, OpenPose [22]. These methods have not shown good results, detection of faces where they do not exist, or does not see the face because it is too small in relation to the image frame.

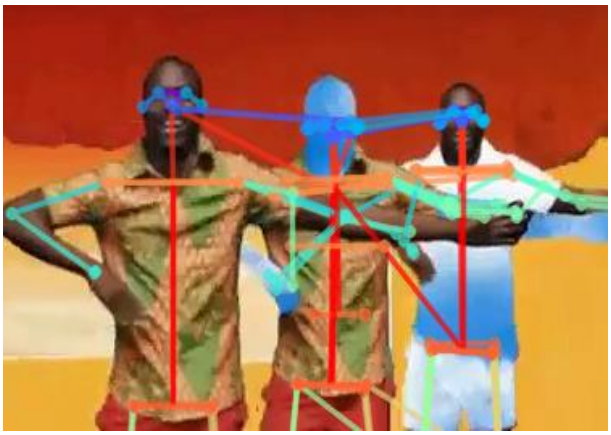


Fig. 1. **Keypoint R-CNN methods for point detection**
Source: **compiled by the authors**

The options Dlib, OpenCV in Fig. 2, could not detect faces on most frames, they work mainly on high resolution and high-quality photos.

As well as MediaPipe also did not give good results, on frames with a small face image as shown, no image detection. According to documentation of MediaPipe and Dlib the distance for the image should be at 1-1.5 m from camera observer. A high-quality image is required and the size of the face in the photo needs to be large enough in order to have the definition of points on the face increased.

These methods often work falsely in low-resolution photos (showing facial points where there are not any).

Fig. 2. **Dlib and OpenCV methods for point detection**
Source: **compiled *by the authors***

The RetinaFace model has been applied to accurately and quickly find reference points in photos. RetinaFace is designed to detect and align faces in real-time video, making it well suited for applications requiring fast and accurate face detection. Despite the poor photo options, this model finds all the points to further transform the image, in Fig.3.

OpenCV was used to decode the video. Each frame went through RetinaFace face point detector. The data from the frame could be retrieved and saved into vector $v = (fr, fa_i, dat)$, where fr is

current frame in video processing; fa_i is number of detected face label; *dat* is data points that RetinaFace model return.

Fig. 3. **RetinaFace methods for point detection**
Source: **compiled by the authors**

The meta-data vector representation the database for processing and replacing the object saved spreadsheet in Fig. 4.

However, the model detect almost all features from frames, there could be cases when the data should be corrected. The user can correct the data if necessary to make changes. For example, remove certain frames; change the point value, etc.

By deleting, its mean extracting frames where faces are not recognized. Each photo has its own code. This is a good way to merge the data after cleaning or correcting it.

MARKUP AND CHARACTER IDENTIFICATION

If there is more than one person in the image, then the point search algorithm searches from left to right and from top to bottom. Thus, if the characters are swapped in the photo, the algorithm will mark incorrectly.

Facial recognition algorithms use deep learning to identify unique features in faces and match them against a database of known faces [23]. This can be done using libraries such as Dlib and OpenCV, which provide pre-trained models and functions for facial recognition.

Another method for comparing photos of individuals' faces is using image similarity algorithms. Image similarity algorithms compare the visual content of two images and determine their similarity based on the differences between the images. This can be done using metrics such as Euclidean distance, cosine similarity, or Hamming distance.

Another approach for comparing photos of individuals' faces is using clustering algorithms. Clustering algorithms group similar individuals based on their similarity. This can be done using libraries such as Scikit-learn, which provides functions for clustering algorithms. Fig. 5 shows an example where the algorithm labels faces and it is not quite correct.

frame	face_nomber	facial_area	left_eye	right_eye	mouth_left	mouth_right	nose	code_photo
frame_0.png	face_1	[1349, 358, 1399	[1376.7784, 382	[1357.6359, 38	[1369.9354, 403.:	[1355.5392, 402.4	[1360.5295, 388.71646	#_0
frame_1.png	face_1	[1351, 370, 1400	[1380.3406, 394	[1360.9247, 39	[1373.7233, 415.:	[1358.1245, 412.7	[1364.7458, 401.602]	#_1
frame_2.png	face_1	[1352, 376, 1402	[1382.5325, 402	[1360.8551, 39	[1376.5387, 423.:	[1359.3939, 420.5	[1366.8827, 409.685]	#_2
frame_3.png	face_1	[1348, 374, 1398	[1381.9594, 400	[1360.4391, 39	[1375.781, 418.8	[1358.0575, 415.4	[1367.401, 405.7651]	#_3
frame_4.png	face_1	[1343, 365, 1391	[1372.9491, 390	[1352.2968, 39	[1371.6973, 412.:	[1355.0021, 413.4	[1360.5497, 401.1653]	#_4
frame_5.png	face_1	[1336, 358, 1383	[1365.8556, 383	[1345.1362, 38	[1364.6987, 406.(	[1348.6929, 408.0	[1353.6797, 394.65854	#_5
frame_6.png	face_1	[1324, 362, 1372	[1354.0409, 386	[1332.7917, 39	[1355.9215, 408.:	[1338.8104, 411.2	[1343.0986, 399.69543	#_6
frame_7.png	face_1	[1311, 371, 1360	[1339.4481, 393	[1319.6736, 40	[1345.135, 414.0	[1327.9226, 419.6	[1330.0078, 407.69238	#_7
frame_8.png	face_1	[1306, 375, 1355	[1333.1232, 395	[1315.285, 407	[1340.3971, 419.'	[1326.3408, 425.9	[1325.7551, 414.98877	#_8
frame_9.png	face_1	[1308, 369, 1356	[1334.9486, 393	[1317.3276, 40	[1342.8171, 413.!	[1329.7911, 420.4	[1328.7118, 409.455]	#_9
frame_10.png	face_1	[1316, 353, 1363	[1343.6904, 376	[1325.8595, 38	[1348.949, 396.8(	[1334.3573, 402.0	[1335.9065, 389.87946	#_10
frame_11.png	face_1	[1324, 337, 1372	[1354.7292, 361	[1333.5286, 36	[1354.6272, 382.:	[1336.851, 383.73	[1343.1047, 372.41208	#_11
frame_12.png	face_1	[1336, 324, 1384	[1367.1338, 347	[1345.0254, 34	[1364.0244, 368.:	[1346.2555, 367.3	[1353.4369, 356.75204	#_12
frame_13.png	face_1	[1334.30,372.50,	[1346.80,341.10	[1380.70,338.5	[1352.00,359.40]	[1371.50,356.70]	[1359.70,346.80]	#_13
frame_14.png	face_1	[1343.30,375.50,	[1353.60,339.60	[1388.20,342.3	[1355.10,359.40]	[1372.30,361.20]	[1368.00,346.00]	#_14

Fig. 4. **The structure of the data obtained after processing the stream of photos obtained from the video**
Source: **compiled by the authors**

The algorithm starts by reading the two portraits and converting them to grayscale. Grayscale conversion is used to reduce the complexity of the images and make it easier to compare them. The next step is to detect the faces in the portraits using a face detection library, such as OpenCV Haar cascades [24] or Deep Learning-based methods like RetinaFace.

Fig. 5. **Extracted faces from the video stream frame by frame**
Source: **compiled by the authors**

Once the faces have been detected, the algorithm can extract features from the faces, such as facial landmarks or deep features. These features can then be used to compare the portraits. For example, the Euclidean distance between the features can be used to calculate a similarity score. Alternatively, the features can be used to train a classifier to determine whether the two portraits belong to the same person or not.

It is important to note that the accuracy of the comparison depends on the quality of the face detection and feature extraction steps. In addition, the choice of similarity metric will also affect the accuracy of the comparison.

The foundation of computer vision algorithms in the OpenCV library is based on the Viola-Jones object detection system. The Viola-Jones method is also characterized by the fact that the process of training classifiers is quite slow, but the face detection process is fast and produces accurate results quickly. When using this method, the probability of a false positive detection is very low, and this accuracy works at angles of up to 30 degrees of tilt of the face. However, for arbitrary angles of more than 30 degrees, which can be an issue in the implementation of some specializations, it is worth considering other methods.

Let's take a closer look at the features of the Viola-Jones method [25]. First of all, it is worth considering the principle of the scanning window, which is the basis for identifying objects in the image. It is assumed that there is an image with the objects to be detected (human faces).

The image is represented as a two-dimensional matrix of pixels, with a size of $width \times height$, where each value corresponds to the color of the pixel: if the image is black and white, the value is in the range of 0 to 255: if the image is colored, the value is between 0 and 255^3, representing the BGR components of the color values of the pixels.

As a result of the algorithm's work, it determines the facial features, and the se arch is carried out in the active region of the image using rectangular features that describe the facial features.

$$rect_i = \{x, y, w, h, a\}, \quad (1)$$

where x and y are the coordinates of the center of the i-th rectangle; w – is the width; h is the height; and a is the angle of inclination of the rectangle to the vertical axis of the image.

Among the aforementioned principles of the Viola-Jones method is the integral representation of the image. In fact, this principle is also applied in other popular methods. This is because the integral representation of the image allows for the calculation of the total brightness of any rectangle, regardless of its size, in the same short time.

The scanning window approach is based on scanning the image with a rectangular area search, for each position of which a classifier is applied. Such a feature detection system is fully automated, does not require human intervention, and therefore provides fast results. This approach is the basic one for further work with face recognition, facial expression detection, personality identification, etc.

Each element of the matrix of the integral representation of the image is the sum of the intensity of all the points that are above and to the left of the current element, and is calculated by the following formula:

$$L(x, y) = \sum_{i=0, j=0}^{i \le x, j \le y} I(i, j). \quad (2)$$

where $I(i, j)$ is the brightness of each pixel of the original image.

Accordingly, the calculation of the integral matrix is also possible by the formula:

$$L(x,y) = I(x,y) - L(x-1,y-1) + L(x,y-1) + L(x-1,y). \quad (3)$$

The next important principle is the use of Haar's features. A feature is a mapping f: $X=D_f$, where D_f is the set of available values of the feature. Depending on this set, features are divided into the following types: a binary feature $D_f = \{0, 1\}$; a nominal feature in which D_f is a finite set; an ordinal characteristic in which D_f is an ordered finite set; a quantitative feature in which D_f is a set of real numbers.

The use of boosting is also important in the Viola-Jones method. Boosting is a set of methods for improving the efficiency of analytical models. A distinction is made between “strong” and “weak” models. A "strong" model is highly efficient and makes few errors in calculations and analysis, and allows you to distribute classes of objects more accurately. At the same time, the “weak” model makes a large number of errors. The process of synthesizing a composition of machine learning algorithms to compensate for the shortcomings of previous algorithms is also called boosting or amplification. This is what makes it possible to increase the efficiency of “weak models”.

Boosting is based on the construction of a chain of classifiers – a cascade driven by the training of each subsequent algorithm on the mistakes of the previous one. At the same time, boosting is a greedy compositional algorithm, each stage of which is accompanied by the selection of the optimal option in order to achieve the best final result. Boosting is a rather effective solution, given the pace of modern development of machine learning technologies, because if properly configured, the resulting boosting composition can contain any large number of algorithms that compensate for each other's shortcomings.

Image transformation

An affine transformation [26] is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation).

The mathematical representation of an affine transformation is usually given as

$$Y = AX + b,$$

where Y is the output vector; A is a matrix that represents the linear transformation component of the affine transformation; X is the input vector; b is a vector that represents the translation component of the affine transformation.

In a spatial transformation each point (x, y) of image is mapped to a point (u, v) in a new coordinate system.

$$u = f_1(x,y), \quad v = f_2(x,y). \quad (4)$$

Mapping from (x, y) to (u, v) coordinates. A digital image array has an implicit grid that is mapped to discrete points in the new domain. These points may not fall on grid points in the new domain.

In general, an affine transformation is a composition of rotations, translations, scaling, and shears.

$$u = c_{11}x + c_{12}y + c_{13}, \quad v = c_{21}x + c_{22}y + c_{23}. \quad (5)$$

where c_{13} and c_{23} – translation transformation; c_{11} and c_{22} – scaling transformation, and the combination of rotations and shears.

A rotation is produced by θ is produced by

$$u = x\cos\theta + y\sin\theta, \quad v = -x\sin\theta + y\cos\theta. \quad (6)$$

Complex affine transforms can be constructed by a sequence of basic affine transforms.

Transform combinations are most easily described in terms of matrix operations. To use matrix operations, homogeneous coordinates are entered. These enable all affine operations to be expressed as a matrix multiplication. Otherwise, translation is an exception.

The affine equations are expressed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (7)$$

The transformation matrices can be used as building blocks.

Translation by (x_0, y_0)

$$T = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (8)$$

Scale by s_1 and s_2

$$T = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (9)$$

Rotate by θ

$$T = \begin{bmatrix} cos\,\theta & sin\,\theta & 0 \\ -sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (10)$$

A is a matrix representing the transformation, and b is a vector representing the translation. *A* is a 2x2 matrix for 2D images and a 3x3 matrix for 3D images.

Scaling involves changing the size of the image by multiplying the pixel coordinates with scaling factors along the *x* and *y* axes. Rotation involves rotating the image by a specified angle around the origin. Shearing involves skewing the image along one of the axes.

Translation involves shifting the image by a specified amount in the x and y directions. The vector b represents the translation parameters, which can be added to the transformed pixel coordinates to obtain the final coordinates.

Affine transformation can be applied to various image processing applications, including image registration, image alignment, image warping, and image rescaling. In image registration, affine transformation is used to align two images to each other by minimizing the distance between corresponding points. In image warping, affine transformation can be used to distort an image to match a given shape or template.

Image rescaling involves resizing an image by applying an affine transformation with a scaling factor. The scaling factor can be different for the *x* and *y* axes to maintain the aspect ratio of the image. Affine transformation can also be used for data augmentation in machine learning applications, where the image is randomly transformed to improve the training data quality.

DeepFace algorithm

The DeepFace verification algorithm is used to solve the problem of identifying a character in a previously obtained portrait [27].

DeepFace uses a deep neural network architecture known as a convolutional neural network (CNN) to analyze facial features. CNNs are designed to recognize patterns in data by processing it through multiple layers of non-linear transformations. In the case of facial recognition, the input is an image of a face, and the output is a set of features that describe the face in a way that is invariant to lighting, pose, and other factors.

The architecture of DeepFace consists of nine layers, including five convolution layers and three fully connected layers. The first layers of the network extract simple features such as edges and corners, while the later layers extract more complex features such as eyes, nose, and mouth. The output of the network is a 4096-dimensional vector, which represents the unique facial features of the input image.

To train DeepFace, Facebook's researchers used a dataset of over 4 million images from over 4,000 individuals. The dataset was carefully curated to include a diverse range of ethnicities, ages, and genders, as well as varying lighting and pose conditions. The images were aligned and normalized to ensure that the faces were in the same position and scale, which is important for accurate recognition.

During training, the neural network is optimized to minimize the difference between the features extracted from two images of the same person, and maximize the difference between the features of two images of different people. This is known as the triplet loss function, and it ensures that the features learned by the network are discriminative and invariant to variations in lighting, pose, and other factors. DeepFace has many applications, including face recognition for security and law enforcement, social media tagging, and personalized marketing. However, there are also concerns about privacy and the potential misuse of facial recognition technology. Facebook has stated that it will only use DeepFace for research purposes, and has implemented privacy safeguards to prevent misuse.

In conclusion, DeepFace is a state-of-the-art facial recognition software developed by Facebook's AI Research division. It uses deep learning algorithms and a massive dataset to achieve high accuracy in face recognition. While it has many potential applications, there are also concerns about privacy and misuse, which need to be carefully addressed.

DeepFace is a deep learning-based face recognition library that can accurately detect and align faces in images. This makes it an ideal choice for use in a portrait comparison algorithm.

The algorithm begins by using DeepFace to detect and align faces in two portraits. Once the faces are detected and aligned, the algorithm can extract deep features from them using a deep learning model.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

These deep features can then be used to compare the portraits. For example, the Euclidean distance between features can be used to calculate similarity estimates [28]. In addition, the characteristics can be used to train a classifier to determine whether two portraits belong to the same person or not.

It is important to note that the accuracy of the comparison depends on the quality of the deep learning model used to extract the features. In addition, the choice of similarity metric also affects the comparison accuracy.

VGG-Face algorithm

VGG-Face is a deep convolutional neural network for face recognition developed by the Visual Geometry Group (VGG) at the University of Oxford. The network is based on the architecture of the VGG network, which is a popular deep learning architecture for image recognition tasks.

The VGG-Face network [29] has 16 layers, including 13 convolutional layers and 3 fully connected layers. The network was trained on a large dataset of faces, including over 2.6 million images of more than 2,600 individuals, using a supervised learning approach.

One of the key features of the VGG-Face network is that it uses a simple and uniform architecture with small 3x3 filters in all convolutional layers. This design choice enables the network to learn rich feature representations that are invariant to variations in pose, expression, and lighting conditions.

Another important aspect of the VGG-Face network is that it uses a large number of parameters, which allows it to capture fine-grained details in face images. However, this also makes the network computationally expensive and requires a significant amount of memory to store the model.

The VGG-Face network has achieved state-of-the-art performance on several benchmark face recognition datasets, including the Labeled Faces in the Wild (LFW) dataset and the YouTube Faces dataset.

The network has also been used in various applications, such as facial authentication, face detection, and emotion recognition.

Google FaceNet algorithm

Google FaceNet is a deep neural network for face recognition developed by researchers at Google [30]. The network uses a triplet loss function to learn a mapping of face images into a high-dimensional feature space, where distances between feature vectors correspond to similarities between faces.

The FaceNet network is based on the Inception architecture, which is a popular deep learning architecture for image recognition tasks. The network has 22 layers, including 9 inception modules and a final fully connected layer with 128 units that output a 128-dimensional feature vector for each face image.

The triplet loss function used by FaceNet is designed to encourage the network to learn embeddings of face images that are close together if they belong to the same person and far apart if they belong to different people. The loss function is computed for triplets of face images, where one image is an anchor, one is a positive example of the same person, and one is a negative example of a different person. The goal is to minimize the distance between the anchor and the positive example, while maximizing the distance between the anchor and the negative example.

The FaceNet network has achieved state-of-the-art performance on several benchmark face recognition datasets, including the Labeled Faces in the Wild (LFW) dataset and the MegaFace Challenge. The network has also been used in various applications, such as facial authentication, face detection, and emotion recognition.

Markup and character identification

After the anchor points are defined as in Fig. 4 and the portraits are marked as in Fig. 5. It is necessary to prepare the image of the user photo to replace the marked photo. MediaPipe is used for this purpose.

Mediapipe is an open-source library for media processing that provides a wide range of functionality for tasks such as computer vision, audio processing, and machine learning. This library is designed to simplify the process of building complex pipelines for processing media, making it easier for developers to focus on the core functionality of their applications.

One of the key strengths of Mediapipe is its modular design, which allows developers to build pipelines using a combination of existing components and custom components. The library includes a large number of pre-built components for tasks such as object detection, image processing, and audio processing, making it possible to build complex pipelines with ease.

MediaPipe Face Mesh is a solution that evaluates 468 3D facial landmarks in real time, even on mobile devices. It uses machine learning (ML) to determine the three-dimensional surface of a face, requiring only one camera input without a dedicated depth sensor.

According to Mediapipe Face Mesh map. It's possible to detect any shape of the face in the photo. After processing Face Mesh, we apply the affine transformation and apply transparent background for extracted image. In the output we get a photo with the face oval and dots to snap to the video frame, in Fig. 6.

Fig. 6. **Extracted faces from the video stream frame by frame**
Source: **compiled by the authors**

Animation of the user's face

The algorithm for creating a lip animation involves two main steps: lip detection and lip animation.

Lip detection can be performed using a face detection library such as OpenCV's Haar cascades or Deep Learning-based methods like MediaPipe. Once the face has been detected, the algorithm can extract the lip region using facial landmarks or a deep learning model. For example, we decided for lips detection use the Mediapipe Face Mesh capabilities, as the result the lips Region of Interest (ROI) seems to be really accurate for detected face.

Lip animation can be achieved by warping the lip region using a transformation matrix [31]. The transformation matrix can be calculated based on the position of the lip landmarks or the deep features extracted from the lip region.

The warped lip region can then be blended with the original face to create the final animation. This process can be repeated for each frame of the video to create a continuous lip animation.

It is important to note that the accuracy and quality of the lip animation will depend on the quality of the lip detection and the transformation matrix calculation. The idea of the animation is to move the mouth and chin area by a certain amount at a certain speed or detect the. The whole process involves superimposing two areas of color background and the cut part of the chin along the cut of the mouth in Fig. 7.

Fig. 7. **Overlaying matrices using OpenCV for animation of the mouth**
Source: **compiled by the authors**

IMAGE OVERLAYING BASED ON BINARY MASKS

When we are talking about blending two images, we should consider next important images attribute: alpha channel and dimensionality. By default, when desired software (OpenCV in our case) process images to matrix M with $h \times w$ dimensionality which responds to height and width of image, the image stores in BGR/RGB subpixel rendering format. This format is good when the image blending does not consider the transparency.

But in our case, one of the images or both have alpha layer, this attribute should be processed in an appropriate way and saved into the final blended image. First of all, to resolve this issue we should know what alpha channel does mean and how it is connected with transparency.

Alpha channel represents the degree of transparency of each pixel in the image channels and by default it scales in $\alpha \in [0,1]$: lower bound means all pixels of channels turns to 0 (no coverage), upper bound means all pixels of channels will be masked (full coverage), with other fractions inside bounds color will be covered with defined quadruple:

$$(x_1/\alpha, x_2/\alpha, x_3/\alpha), \qquad (11)$$

where $x_i, i = \overline{1,3}$ are image channels; $0 < \alpha < 1$ is alpha channel fraction.

Simply, the alpha channel could be interpreted as additional mask layer for image color channels, this binary mask keeps the information about specific pixel $\{x_1, x_2, x_3\}$ opaque α. So when image pixel in RGB rendering format $\{0,0,0\}$ (refers to black color) have alpha channel fraction $\alpha = 1$ – means all black colors will be transparent. This case is shown in Fig. 8.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

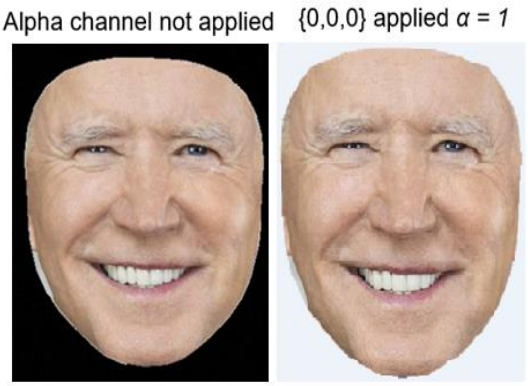


Fig. 8. **Alpha channel applied to black color**
Source: **compiled by the authors**

In Fig. 8 shown a good example when the image has homogeneous structure: pixels outside face represented in black color, but the image may have heterogeneous structure.

To transform the image from heterogeneous to homogeneous which means the Region of Interest (now and hereafter ROI) will have same matrix structure – outside the "object" pixels will have black or white color. The solution of this problem could be found with image thresholding.

Thresholding is the tool for segmentation images that could return a binary mask which stores only some parts or objects of images according to the specified method and requirements.

There are a lot of thresholding methods: manual, automatic or adaptive, Otsu, Mode, p-tile, Histogram Concavity Analysis and etc. [32]. It is most expedient to use a simple method of adaptive thresholding.

Let's define *src* matrix with dimensionality $h \times w$ which stores the X grayscale pixels format then the function of automatic thresholding could be expressed with next equation:

$$dst(x,y) = \begin{cases} max(src(x,y)) \ if \ src(x,y) > T(x,y), \\ 0 \ otherwise. \end{cases} \quad (12)$$

where $T(x,y)$ is a threshold calculated for each pixel (x,y).

So the meaning of defined equation could be expressed as every pixel (x,y) from *src* matrix which is bigger than $T(x,y)$ will be saved without transformation into $dst(x,y)$ otherwise the value is equal to 0 [32]. Applying thesholding for image will return us a binary mask for it in Fig. 9.

Fig. 9. **Adaptive thresholding for image**
Source: **compiled by the authors**

The returned result is used for image masking to create new "homogeneous" image with blank outside of the ROI. Since the matrix *dst* have same dimensionality as *src* the masking will be expressed as (13).

$$masked(I) = src \wedge src, mask(I) = dst \quad (13)$$

Result of masking is shown in Fig. 10.

Fig. 10. **Masked image with adaptive threshold mask**
Source: **compiled by the authors**

Images could contain own alpha channel in order to correctly merge 2 images with alpha channels next equation is used:

$$\begin{aligned} c_1 &= \alpha f_1 + (1-\alpha) b_1, \\ c_2 &= \alpha f_2 + (1-\alpha) b_2, \\ c_3 &= \alpha f_3 + (1-\alpha) b_3, \end{aligned} \quad (14)$$

where c_i is final alpha channel for each channels; f_i is foreground image that is needed to blend with background b_i; $i = \overline{1,3}$ and α is the opacity value of the foreground pixel f_i.

In order to get the result, it is necessary to combine the alpha channel to blend the image along the axis $x = 4$, the scheme to this operation show in Fig. 11.

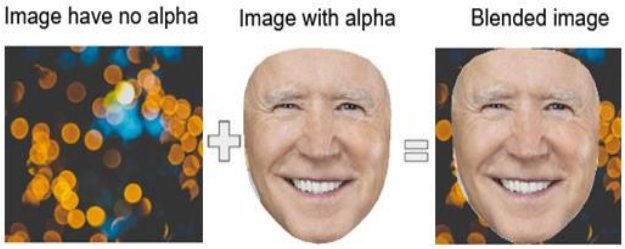


Fig. 11. **Image blend at specific point (x, y) with respected alpha channels**
Source: **compiled by the authors**

During the research, an algorithm was defined that combines 2 BGR images with an alpha channel:

1. Load images with OpenCV in BGRA format.
2. Split alpha channel(s) into vector α.
3. Convert BGRA image to BGR.
4. Multiply the BGR image with the inverted alpha channel of the BGRA image.
5. Add the result of step 3 with BGRA image to obtain final image.

IMAGE MASKING BASED ON COLOR CONDITIONS

Image masking is a powerful technique, applying different conditions, providing isolated shape area or color could effect on really hyper realistic results.

This technique could be applied not only for the image to blend but also for every single video frame without applying any changes of input data, working with preloaded frame masks only.

But providing areas to mask manually would reflect to a really long process, video streams for nowadays have a really outstanding resolution and frames per seconds (FPS) parameters. More and more software packages provide options to play and save videos in 30-60-120 FPS. As an example, 60 seconds duration video with 60 FPS parameter will decode 3600 frames to operate.

To simplify the video processing and keep attraction on the highest level with affordable processing rate we could use HSV rendering format.

HSV stands for Hue, Saturation, and Value. Hue refers to the dominant wavelength of a color, and is often represented as an angle on a color wheel.

Saturation refers to the purity or intensity of the color, with fully saturated colors being vivid and intense, and desaturated colors appearing more gray or washed out. Value refers to the brightness of the color, with higher values being brighter and lower values being darker.

This format was developed with RBG transformation to cylinder coordinates. Image frames in HSV format could have same color range in specific ROIs that is calculated according to shape of the object. In research case this reference shape responds to dimensions of the face on frames, but in fact the object is not important for masking.

The main idea in masking with HSV – select the correct color range from low up to high for the ROI mask, convert it to binary mask and then concatenate this mask with another mask retrieved from thresholding operations.

For example, let's operate with HSV image from Fig. 12 as for ROI select next part of image which have "light green and light red color" or in HSV range $low=(75,94,69)$, $high=(24,88,78)$ and apply binarization for selected range (this operation already built in OpenCV).

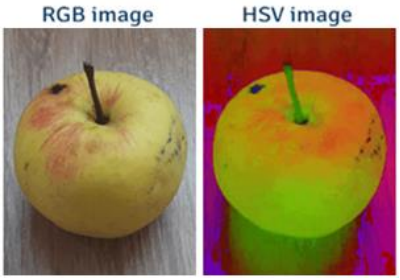


Fig. 12. **RGB to HSV conversion sample**
Source: **compiled by the authors**

According to HSV masking pipeline: concatenate mask (Fig.13) to affined mask using equation modified equation (15).

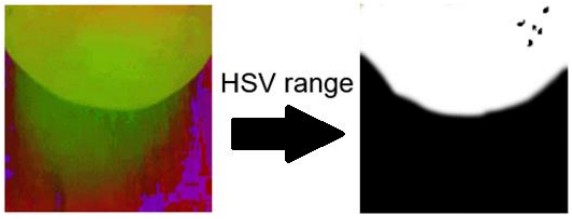


Fig. 13. **HSV range operation masking**
Source: **compiled by the authors**

$$masked(I)=src_1 \wedge src_2, mask(I)=src_2, \quad (15)$$

where src_1 – affined mask; src_2 – HSV mask.

The result of the binary masks concatenation visualized in Fig. 14.

Fig. 14. **Concatenation of binary masks**
Source: **compiled by the authors**

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

As the result, final mask could be applied to transformed image with *X* rendering format (Fig. 15).

Fig. 15. **Result after applying HSV masking to specified ROI of image**
Source: **compiled by the authors**

BOUNDARY CONDITIONS FOR IMAGE ALLINMENT

Dimensionality is the second point that is needed to be considered while blending images. The dimensions of the inserted image should be less that second image. Image dimensions could be dynamic in replace process and because of insert point (*x*, *y*) it is varying from really small dimensions up to real affined image. This effect happens when the insert point (*x*, *y*) at specific image have invalid coordinates.

The term invalid point means next: the coordinates are less than image start point or bigger than the image height and width value along all axis:

$$0 < x \cup x \geq width \cup y > 0 \cup y \geq height\,. \quad (16)$$

So, in this case the coordinates of specific point are invalid, as the result image cannot be processed and inserted to this point without any transformation, but it does not mean that It is impossible.

To resolve this issue let's define four main cases:

1. $x < 0 \wedge y < 0$
2. $x < 0 \wedge y \geq height$
3. $x \geq width \wedge y < 0$
4. $x \geq width \wedge y \geq height$

These cases for reference point will not allow insert image without transformation since image cannot be inserted outside the bounds of the second image. So the main idea of image transformation in case of invalid coordinates is to slice only visible part of images.

For these we should define next statements: ins – matrix of destination image and dst – matrix of inserted image, the reference point $p = (x, y)$ – have invalid coordinates according to F-case, which map conditions above; visible inserted image ROI will be represented as matrix upd and appropriate ROI for destination matrix – cor.

For each cases corrected transformation for point $(x, y) \to (x', y')$ and images ROI (*cor* and *upd* matrices are the same for all cases, only the sizes change, therefore they are shown only for case 1):

1. $F = x < 0 \wedge y < 0,$

$$x' = width_{ins} + x,$$
$$y' = height_{ins} + y,$$
$$cor = \begin{bmatrix} a_{11} & a_{12} & a_{1j} \\ a_{21} & a_{22} & a_{2j} \\ a_{i1} & a_{i2} & a_{ij} \end{bmatrix}, \quad (17)$$
$$i = \overline{0, y'},\ j = \overline{0, x'},$$
$$upd = \begin{bmatrix} b_{11} & b_{12} & b_{1j} \\ b_{21} & b_{22} & b_{2j} \\ b_{i1} & b_{i2} & b_{ij} \end{bmatrix},$$
$$i = \overline{height_{ins} - y', y'},\ j = \overline{width_{ins} - x', x'}.$$

2. $F = x < 0 \wedge y \geq height,$

$$x' = width_{ins} + x,$$
$$y' = height_{ins} - max(0, y + height_{ins} - height_{dst}),$$
$$cor = [...], \quad (18)$$
$$i = \overline{y, y'},\ j = \overline{0, x'},$$
$$upd = [...],$$
$$i = \overline{0, height_{ins}},\ j = \overline{width_{ins} - x', x'},$$

3. $F = x \geq width \wedge y < 0,$

$$x' = width_{ins} - max(0, x + width_{ins} - width_{dst}),$$
$$y' = height_{ins} + y,$$
$$cor = [...], \quad (19)$$
$$i = \overline{0, y'},\ j = \overline{x, x'},$$
$$upd = [...],$$
$$i = \overline{|height_{dst} - height_{ins}|, y'},\ j = \overline{0, x'},$$

4. $F = x \geq 0 \wedge y \geq 0,$

$$x' = width_{ins} - max(0, x + width_{ins} - width_{dst}),$$
$$y' = height_{ins} - max(0, x + height_{ins} - height_{dst}),$$
$$cor = [...], \quad (20)$$

$$i = \overline{y, y'},\ j = \overline{x, x'},$$
$$upd = [...],$$
$$i = \overline{0, y'},\ j = \overline{0, x'},$$

The visualization for cases is shown in Fig. 16.

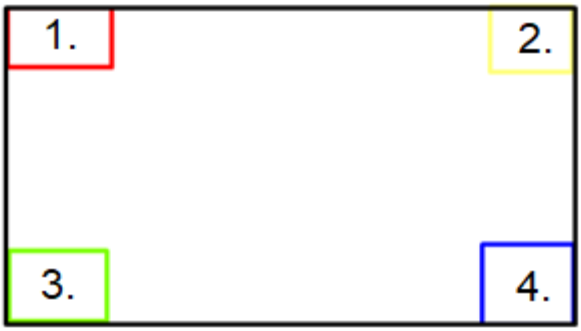


Fig. 16. **F-cases for invalid reference point visible part of inserted image compared to destination image**
Source: **compiled by the authors**

OPENCV POSSIBILITIES TO UTILIZE CUDA TECHNOLOGY

Infocommunication systems are becoming increasingly complex, and their performance requirements are growing rapidly. To meet these requirements, new technologies are being developed that can process large amounts of data quickly and efficiently. One such technology is GPGPU, or General-Purpose Computing on Graphics Processing Units.

GPGPU is a technique that uses the parallel processing power of graphics processing units (GPUs) for general-purpose computing tasks. GPUs were originally designed for rendering graphics in video games and other applications, but they have since evolved to become powerful computing devices in their own right. By using GPGPU, infocommunication systems can take advantage of the massive parallel processing power of GPUs to perform complex calculations and other tasks more quickly and efficiently than with traditional CPU-based systems.

In OpenCV applications all tasks execute with CPU cores by default but there is an option to increase performance of image or video processing with GPGPU technologies like OpenCL or CUDA [33]. However, the biggest boost to performance could be achieved only with Nvidia CUDA technologies but since it is Nvidia product it is available from the box only on Nvidia graphical cards.

As for OpenCV API – the library support CUDA casting within a few additional calls. The performance curve for applying a Gaussian filter to video frames with specific samples was examined to determine the relationship between performance and number of samples (Fig. 17).

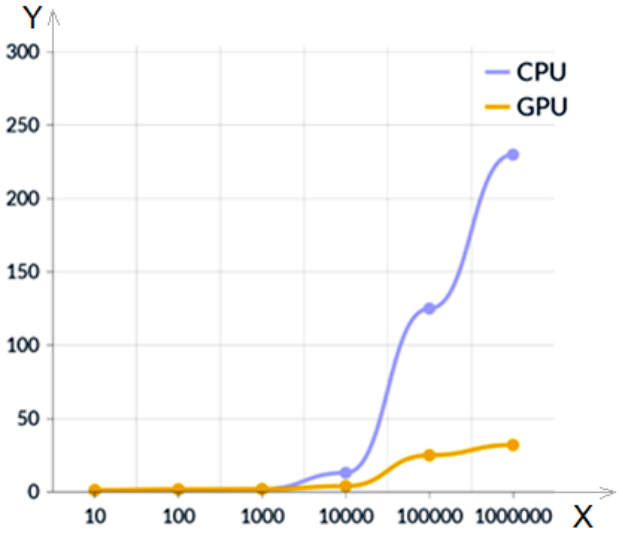


Fig. 17. **Time dependency in performance on CPU and GPU with CUDA technology**
Source: **compiled by the authors**

As seen in Fig. 17, GPU cores show a good performance enhancement when the large amount of data is needed to process. When the amount batch of digital images is less than 10 000 samples, there is no need to call CUDA operations because it will cause the lag in initialization functions. For most not complex operations, CPU could be also used. The complete algorithmic pipeline studied in the article is presented in Fig. 18.

CONCLUSIONS

In conclusion, face replacement in video streaming presents unique challenges due to the real-time demands and the complexity of accurately detecting and replacing faces under diverse conditions. This study aimed to enhance the efficiency and accuracy of face detection and replacement within video streaming contexts by utilizing the capabilities of advanced neural network technologies and pre-trained models, with a particular focus on RetinaFace, Medipipe, and OpenCV.

Findings indicate that each of these tools possesses its unique strengths, with varying trade-offs between accuracy and processing speed. RetinaFace, a deep learning-based face detection and alignment algorithm, emerges as a powerful tool. Its ability to swiftly detect faces of different scales and orientations (within 10-20 milliseconds per frame) underscores its potential in real-world applications that demand quick face detection and replacement.

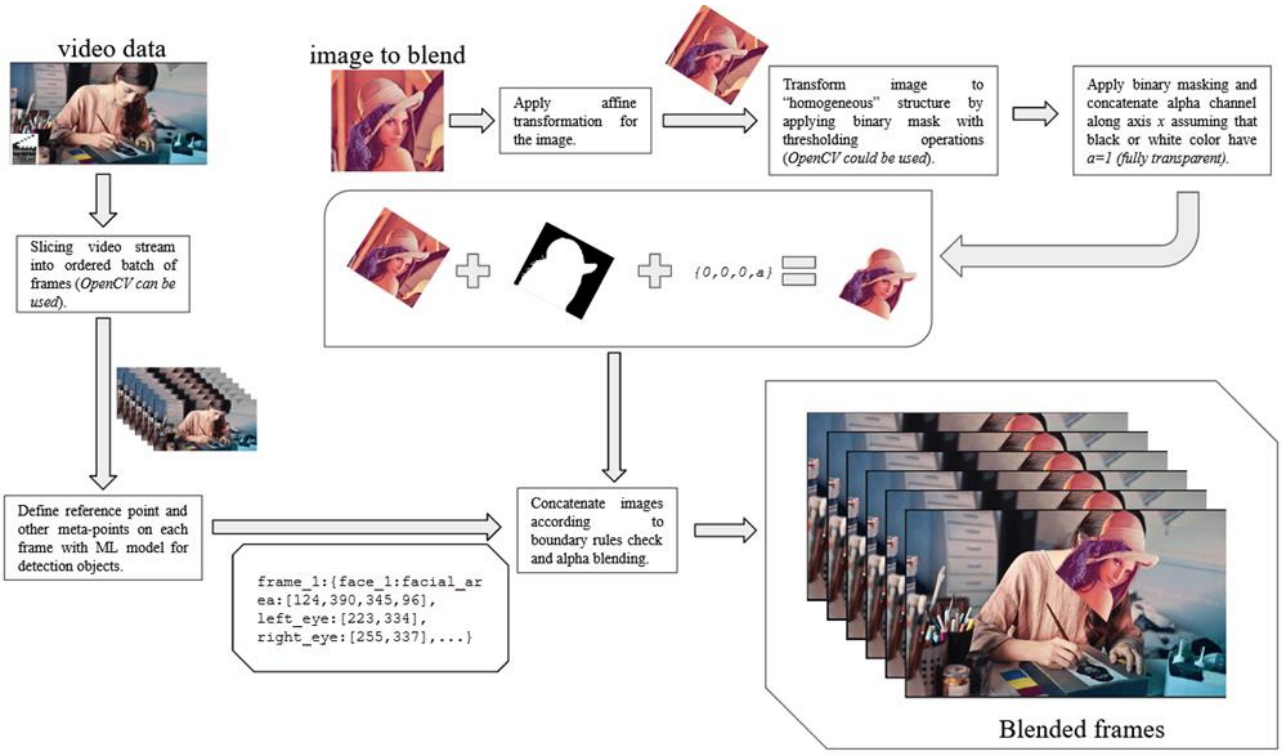


Fig. 18. **Pipeline scheme for inserting images at specific reference point**
Source: **compiled by the authors**

Medipipe, another deep learning-based framework, offers high performance and flexibility. Its user-friendly interface facilitates the integration of various computer vision models, including those for face and landmark detection, in approximately 20-30 milliseconds per frame. This versatility makes Medipipe an adaptable tool for face replacement tasks, particularly in diverse and complex video streams.

OpenCV, a comprehensive computer vision library, provides an array of tools for face detection and replacement. Despite a slower processing time (50-100 milliseconds per frame), its robustness and use of pre-trained models can achieve higher accuracy, a critical quality for scenarios where precision is of greater importance than speed.

It is critical to acknowledge that the choice of face detection and replacement method is heavily influenced by the specific requirements and constraints of a given application. RetinaFace and Medipipe may prove more suitable for scenarios demanding high-speed processing, while OpenCV might be the preferred choice in contexts that prioritize accuracy.

This study significantly contributes to the understanding of the scope and applicability of these three methods, providing a solid foundation for future research. With advancements in deep learning and computer vision technologies, face replacement in video streaming has become more accessible, efficient, and adaptable to various scenarios. The RetinaFace, Medipipe, and OpenCV methods emerge as leading approaches, offering an effective balance of performance, accuracy, and flexibility.

The potential to create simple fake videos quickly and with higher accuracy has far-reaching implications across various sectors, including entertainment, education, research, and even forensics. Further exploration and refinement of these techniques are needed, potentially incorporating elements like emotion detection or aging effects to increase the realism and applicability of the replaced faces. As this fast-evolving field continues to advance, ensuring the responsible use of such powerful technologies is crucial, with ethical guidelines and safeguards in place to prevent misuse.

Looking ahead, there will undoubtedly be an evolution of these methods and potentially the development of new techniques that are even more efficient and accurate. Such developments will need to account for increasingly sophisticated requirements, from handling variable lighting conditions to managing the intricacies of face orientation or subtle facial expressions.

REFERENCES

1. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y. & Xiao, B. “Deep high-resolution representation learning for visual recognition”. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2021; 43 (10): 3349–3364. DOI: https://doi.org/10.1109/TPAMI.2020.2983686.

2. Li, R., Huang, H. & Zheng, Y. “Human pose estimation based on lite HRNet with coordinate attention". *7th International Conference on Intelligent Computing and Signal Processing (ICSP*). Xi'an: China. 2022. p. 1166–1170. DOI: https://doi.org/10.1109/ICSP54964.2022.9778346.

3. Cheng, Z. & Fu, D. “Remote sensing image segmentation method based on HRNET”. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Waikoloa: HI, USA. 2020. p. 6750–6753, https://www.scopus.com/authid/detail.uri?authorId=57207759702. DOI: https://doi.org/10.1109/IGARSS39084.2020.9324289.

4. Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I. & Zafeiriou, S. “Retinaface: Single-stage dense face localisation in the wild”. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle: USA. 2020. p. 5202–5211. DOI: https://doi.org/10.1109/CVPR42600.2020.00525.

5. Sitepu, S. E., Jati, G., Alhamidi, M. R., Caesarendra, W. & Jatmiko, W. “FaceNet with RetinaFace to identify Masked Face”. *6th International Workshop on Big Data and Information Security (IWBIS*). Depok: Indonesia. 2021. p. 81–86, https://www.scopus.com/authid/detail.uri?authorId=55490339500. DOI: https://doi.org/10.1109/IWBIS53353.2021.9631848.

6. Xue, B., Hu, J. & Zhang, P. “Intelligent detection and recognition system for mask wearing based on improved RetinaFace algorithm”. *2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI).* Taiyuan: China. 2020. p. 474–479, https://www.scopus.com/authid/detail.uri?authorId=55499454700.
DOI: https://doi.org/10.1109/ MLBDBI51377.2020.00100.

7. Noor Reza, M. A., Zaki Hamidi, E. A., Ismail, N., Effendi, M. R., Mulyana, E. & Shalannanda, W. “Design a landmark facial-based drowsiness detection using dlib and opencv for four-wheeled vehicle drivers”. *15th International Conference on Telecommunication Systems, Services, and Applications (TSSA).* Bali: Indonesia. 2021. p. 1–5, https://www.scopus.com/authid/detail.uri?authorId=57200569226. DOI: https://doi.org/10.1109/TSSA52866.2021.9768278.

8. Zhang, D., Li, J. & Shan, Z. “Implementation of dlib deep learning face recognition technology”. *International Conference on Robots & Intelligent System (ICRIS).* Sanya: China. 2020. p. 88–91, https://www.scopus.com/authid/detail.uri?authorId=55085374700.
DOI: https://doi.org/10.1109/ICRIS52159.2020.00030.

9. Mahdi, W. A., Mahdi, S. Q. & Al-Naji, A. “Generating masked facial datasets using dlib-machine learning library”. *4th International Conference on Advanced Science and Engineering (ICOASE)*. Zakho: Iraq. 2022. p. 66–70, https://www.scopus.com/authid/detail.uri?authorId=57189363725. DOI: https://doi.org/10.1109/ICOASE56293.2022.10075601.

10. Latreche, A, Kelaiaia, R., Chemori, A., Kerboua, A. “Reliability and validity analysis of MediaPipe-based measurement system for some human rehabilitation motions”. *Measurement.* 2023; 214: 112826, https://www.scopus.com/authid/detail.uri?authorId=58026746000.
DOI: https://doi.org/10.1016/j.measurement.2023.112826.

11. Bayar, N., Güzel, K. & Kumlu, D. “A novel blazeface based pre-processing for MobileFaceNet in face verification”. *45th International Conference on Telecommunications and Signal Processing (TSP)*. Prague: Czech Republic. 2022. p. 179–182, https://www.scopus.com/authid/detail.uri?authorId=57244661400. DOI: https://doi.org/10.1109/TSP55681.2022.9851255.

12. Singhal, R., Modi, H., Srihari, S., Gandhi, A., Prakash, C. O. & Eswaran, S. “Body posture correction and hand gesture detection using federated learning and mediapipe”. *2nd International Conference for Innovation in Technology (INOCON).* Bangalore: India. 2023. p. 1–6, https://www.scopus.com/authid/detail.uri?authorId=57460519900.
DOI: https://doi.org/10.1109/INOCON57975.2023.10101124.

13. Jiang, L., Chen, J., Todo, H., Tang, Z., Liu, S. & Li, Y. “Application of a Fast RCNN Based on Upper and Lower Layers in Face Recognition”. *Computational Intelligence and Neuroscience*. 2021; 2021: 9945934, https://www.scopus.com/authid/detail.uri?authorId=57209284865.
DOI: https://doi.org/10.1155/2021/9945934.

14.He, K., Gkioxari, G., Dollár P. & Girshick, R. "Mask R-CNN". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020; 42 (2): 386–397, https://www.scopus.com/authid/detail.uri?authorId=15622876800. DOI: https://doi.org/10.1109/TPAMI.2018.2844175.

15. Tirupal, T., Kumar, M. N., Basha, P. M., Babu J. M. & Rathan, O. "OPENCV based smart attendance system using facial recognition". *4th International Conference for Emerging Technology (INCET)*. Belgaum: India. 2023. p. 1–6. DOI: https://doi.org/10.1109/INCET57972.2023.10170456.

16. Duan, C. & Luo, S. "Design of pedestrian detection system based on OpenCV". *4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*. Hamburg: Germany. 2022. p. 256–259, https://www.scopus.com/authid/detail.uri?authorId=57555516400.
DOI: https://doi.org/10.1109/AIAM57466.2022.00055.

17. Hou, W., Xia, D. & Jung, H. "Video road vehicle detection and tracking based on OpenCV". *International Conference on Information Science and Education (ICISE-IE)*. Sanya: China. 2020. p. 315–318, https://www.scopus.com/authid/detail.uri?authorId=12778468200.
DOI: https://doi.org/10.1109/ICISE51755.2020.00076.

18. Mishra, S. & Pradhan, R. K. "Analyzing the impact of feature correlation on classification accuracy of machine learning model". *International Conference on Artificial Intelligence and Smart Communication (AISC)*. Greater Noida: India. 2023. p. 949–953, https://www.scopus.com/authid/detail.uri?authorId=58195199400. DOI: https://doi.org/10.1109/AISC56616.2023.10085293.

19. Wei, Q., Hu, X., Wang, X. & Wang, H. "Improved RetinaNet target detection model". *2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence (AHPCAI)*. Guangzhou: China. 2022. p. 470–476. DOI: https://doi.org/10.1109/AHPCAI57455.2022.10087635.

20. Luo, S. & Zheng, W. "You-only-look-once-v5 based table detection for academic papers". *International Conference on Digital Society and Intelligent Systems (DSInS)*. Chengdu: China. 2021. p. 53–56, https://www.scopus.com/authid/detail.uri?authorId=57194978524.
DOI: https://doi.org/10.1109/DSInS54396.2021.9670600.

21. Pulipalupula, M., Patlola, S., Nayaki, M., Yadlapati, M., Das, J. & Sanjeeva Reddy, B. R. "Object detection using you only look once (YOLO), algorithm in convolution neural network (CNN)". *IEEE 8th International Conference for Convergence in Technology (I2CT)*. Lonavla: India. 2023. p. 1–4. DOI: https://doi.org/10.1109/I2CT57861.2023.10126213.

22. Lina, W. & Ding, J. "Behavior detection method of OpenPose combined with Yolo network". *International Conference on Communications, Information System and Computer Engineering (CISCE)*. Kuala Lumpur: Malaysia. 2020. p. 326–330. DOI: https://doi.org/10.1109/CISCE50729.2020.00072.

23. Mira, F. "Deep learning technique for recognition of deep fake videos". *IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. London: United Kingdom. 2023. p. 1–4, https://www.scopus.com/authid/detail.uri?authorId=57192162466.
DOI: https://doi.org/10.1109/GlobConET56651.2023.10150143.

24. Anggadhita, M. P. & Widiastiwi, Y. "Breaches detection in zebra cross traffic light using haar cascade classifier". *International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. Jakarta: Indonesia. 2020. p. 272–277. https://www.scopus.com/authid/detail.uri?authorId=57218626891. DOI: https://doi.org/10.1109/ICIMCIS51567.2020.9354275.

25. Viola, P. & Jones, M. "Rapid object detection using a boosted cascade of simple features". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Kauai: USA. 2001. p. I–I. DOI: https://doi.org/10.1109/CVPR.2001.990517.

26. Cao, C., Preda, M. & Zaharia, T. "Affine transformation-based color compression for dynamic 3D point clouds". *IEEE International Conference on Image Processing (ICIP)*. Bordeaux: France. 2022. p. 1556–1560, https://www.scopus.com/authid/detail.uri?authorId=7005864787.
DOI: https://doi.org/10.1109/ICIP46576.2022.9897788.

27. Gunasekar, M., Panneerselvam, A., Sneharathna, V., Suganneshan, M. & Logeswaran, K. "Improved Facial Emotion Recognition using Yolo and DeepFace for Music suggestion". *2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*. Coimbatore, India, 2022, p. 1124-1127, https://www.scopus.com/authid/detail.uri?authorId=57209690820. DOI: https://doi.org/10.1109/ICESC54411.2022.9885456.

28. Singh, M. K., Singh, N. & Singh, A. K. "Speaker's voice characteristics and similarity measurement using euclidean distances". *International Conference on Signal Processing and Communication (ICSC)*. Noida: India. 2019. p. 317–322. DOI: https://doi.org/ 10.1109/ICSC45622.2019.8938366.

29. Gyawali, D., Pokharel, P., Chauhan, A. & Shakya, S. C. “Age range estimation using MTCNN and VGG-Face model”. *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* Kharagpur: India. 2020. p. 1–6. DOI: https://doi.org/10.1109/ICCCNT49239.2020.9225443.

30. Bouallegue, G. & Djemal, R. “EEG person identification using facenet, LSTM-RNN and SVM”. *17th International Multi-Conference on Systems, Signals & Devices (SSD).* Monastir: Tunisia. 2020. p. 22–28, https://www.scopus.com/authid/detail.uri?authorId=55959113300. DOI: https://doi.org/10.1109/SSD49366.2020.9364129.

31. Agarwal, S., Das, D., Bhowmick, B. “Realistic Lip Animation from Speech for Unseen Subjects using Few-shot Cross-modal Learning”. *2020 28th European Signal Processing Conference (EUSIPCO)*. Amsterdam, Netherlands. 2021. p. 690-694, https://www.scopus.com/authid/detail.uri?authorId=57212306170. DOI: https://doi.org/10.23919/Eusipco47968.2020.9287778.

32. Choi, K. -H. & Ha, J. -E. “An adaptive threshold for the canny edge with actor-critic algorithm”. *IEEE Access*. 2023; 11: 67058–67069, https://www.scopus.com/authid/detail.uri?authorId=58481363800. DOI: https://doi.org/ 10.1109/ACCESS.2023.3291593.

33. Asaduzzaman, A., Trent, A., Osborne, S., Aldershof, C. & Sibai, F. N. “Impact of CUDA and OpenCL on Parallel and Distributed Computing”. *8th International Conference on Electrical and Electronics Engineering (ICEEE).* Antalya: Turkey. 2021. p. 238–242, https://www.scopus.com/authid/detail.uri?authorId=35316746700. DOI: https://doi.org/10.1109/ICEEE52452.2021.9415927.

Conflicts of Interest: the authors declare no conflict of interest

Received 15.06.2023
Received after revision 28.08.2023
Accepted 14.09.2023

DOI: https://doi.org/10.15276/aait.06.2023.20
УДК 004.032.26:004.946

Ефективне детектування і заміна облич при створенні простого фейкового відео

Шеремет Олексій Іванович[1)]
ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus Author ID: 57170410800
Садовой Олександр Валентинович[2)]
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
Гаршанов Денис Володимирович[3)]
ORCID: https://orcid.org/0009-0008-6257-468X; denysharshanov3@gmail.com
Ковальчук Олег Степанович[1)]
ORCID: https://orcid.org/0009-0009-5521-6451; 3289560@gmail.com
Шеремет Катерина Сергіївна[1)]
ORCID: https://orcid.org/0000-0003-3783-5274; artks@ukr.net. Scopus Author ID: 57207768511
Сохіна Юлія Віталіївна[4)]
ORCID: https://orcid.org/0000-0002-4329-5182; jvsokhina@gmail.com. Scopus Author ID: 57205445522
[1)] Донбаська державна машинобудівна академія, бул. Машинобудівників, 39. Краматорськ, 84313, Україна
[2)] Національний ТУ «Дніпровська політехніка», пр. Дмитра Яворницького, 19. Дніпро, 49005, Україна
[3)] Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна
[4)] Дніпровський державний технічний університет, вул. Дніпробудівська, 2. Кам’янське, 51918, Україна

АНОТАЦІЯ

Технології виявлення та розпізнавання обличь є одними з найбільш інтенсивно досліджуваних тем у галузі комп’ютерного зору завдяки їх величезному потенціалу застосування в багатьох галузях. Ці технології продемонстрували практичне застосування в різних контекстах, таких як виявлення підозрілих осіб у багатолюдних міських просторах, розпізнавання власників смартфонів у реальному часі, створення переконливих діпфейків для розважальних додатків і спеціалізованих програм, які змінюють рухи рис обличчя, наприклад губ або очей. Завдяки сучасним досягненням апаратного та програмного забезпечення, сучасна технологічна інфраструктура надає більше ресурсів, ніж необхідно для потокового відео. У результаті прості системи розпізнавання обличь можуть бути реалізовані без використання дорогих серверів, які вимагають певних попередньо навчених моделей. Така велика кількість ресурсів змінює ландшафт розпізнавання обличь, і дискусія в даній статті обе-

ртається навколо цих нових парадигм. Основна увага в цій статті – поглиблений аналіз ключових концепцій детектування обличчя в потокових відеоданих за допомогою відомих попередньо навчених моделей. Обговорювані моделі включають HRNet, RetinaFace, Dlib, MediaPipe і KeyPoint R-CNN. Кожна з цих моделей має свої сильні та слабкі сторони, і дана стаття розглядає ці атрибути в контексті практичних прикладів із реального світу. Такий розгляд дає цінну інформацію про практичне застосування цих моделей і компроміси, пов'язані з їх використанням. Крім того, стаття представляє вичерпний огляд методів трансформації зображення. Представлено абстрактний метод афінного перетворення зображення, важливу техніку обробки зображень, яка змінює геометричні властивості зображення, не впливаючи на інтенсивність його пікселів. Крім того, у статті розглядаються операції перетворення зображень, які виконуються за допомогою бібліотеки OpenCV, однієї з провідних бібліотек у галузі комп'ютерного зору, що забезпечує дуже гнучкий і ефективний набір інструментів для маніпулювання зображеннями. Кульмінацією цього дослідження є практична автономна система для заміни зображення у відео. Ця система використовує модель RetinaFace для здійснення висновків і використовує OpenCV для афінних перетворень, демонструючи концепції та технології, які обговорюються в статті. Таким чином, проведена робот просуває сферу виявлення та розпізнавання обличь, представляючи інноваційний підхід, який повною мірою використовує сучасні апаратні та програмні досягнення.

Ключові слова: діпфейк; афінна трансформація; виявлення обличчя; відео обробка; альфа-канал; бінарні маски

ABOUT THE AUTHORS

Oleksii I. Sheremet – Doctor of Engineering Sciences, Professor, Head of the Department of Electromechanical Systems of Automation and Electric Drive. Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine
ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus ID: 57170410800
***Research field*:** Machine learning and artificial intelligence in general technical problems and electromechanics; predicative analytics based on artificial intelligence technology

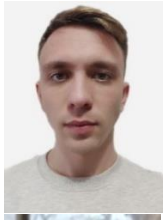
Шеремет Олексій Іванович – доктор технічних наук, професор, завідувач кафедри Електромеханічних систем автоматизації Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна

Oleksandr V. Sadovoi – Doctor of Engineering Sciences, Professor of the Department of Electric Drive, Dnipro University of Technology, 19, Dmytra Yavornytskogo Ave. Dnipro, Ukraine
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
***Research field*:** Optimal control of electromechanical systems

Садовой Олександр Валентинович – доктор технічних наук, професор кафедри Електроприводу національного ТУ «Дніпровська політехніка», пр. Дмитра Яворницького, 19. Дніпро, Україна

Denys V. Harshanov – Student, Department of Computer Engineering and Control, Kharkiv National University of Radioelectronics, 14, Nauky Ave. Kharkiv, Ukraine
ORCID: https://orcid.org/0009-0008-6257-468X; denysharshanov3@gmail.com
***Research field*:** Image processing in infocommunication systems; bilinear filtering with binary masking methods

Гаршанов Денис Володимирович – студент факультету Комп'ютерної інженерії та управління Харківського національного університету радіоелектроніки, пр. Науки, 14. Харків, Україна



Oleh S. Kovalchuk – Student, Department of Automation of Mechanical Engineering and Information Technology, Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine
ORCID: https://orcid.org/0009-0009-5521-6451, 3289560@gmail.com
***Research field*:** Computer vision; natural language processing

Ковальчук Олег Степанович – студент факультету Автоматизації машинобудування та інформаційних технологій Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна

Kateryna S. Sheremet – Department Engineering of the Department of Intelligent Decision Support Systems. Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine
ORCID: https://orcid.org/0000-0003-3783-5274, artks@ukr.net. Scopus Author ID: 57207768511
***Research field*:** Machine learning; decision support systems

Шеремет Катерина Сергіївна - інженер кафедри Інтелектуальних систем прийняття рішень Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна

Yuliia V. Sokhina – PhD in Engineering Sciences, Associate Professor of the Department of Electrical Engineering and Electromechanics, Dniprovsky State Technical University, 2, Dniprobudivska, Str. Kamyanske, Ukraine
ORCID: https://orcid.org/0000-0002-4329-5182; jvsokhina@gmail.com. Scopus Author ID: 57205445522
***Research field*:** Optimal control of electromechanical systems

Сохіна Юлія Віталіївна – кандидат технічних наук, доцент кафедри Електротехніки та електромеханіки Дніпровського державного технічного університету, вул. Дніпробудівська, 2. Кам'янське, Україна