ISSN 1512-1232

**UDC 519.685**

# ABOUT COMPUTER VISION WITH MACHINE LEARNING

Giga Kokaia

Sokhumi State University
Faculty of Natural Sciences, Mathematics, Technologies and Pharmacy

***Abstract***

*In the article discusses how machine learning models can be integrated and used in the modern programs. In particular an example of determining the location of a person face from a picture or video and further processing to make face blur. OpenCV library is used, because it is open-source software and can be used for free.*

***Keywords***: *Image recognition, Computer Vision, Machine Learning, OpenCV, HaarCascade.*

## Introduction

In our application, we use the OpenCV library to identify the location of human faces in a video stream. The method is called “computer vision”. Computer vision is the process through which we can understand images and videos, and how they are stored, and how to manipulate and retrieve data from them. Computer vision is the basis of Artificial Intelligence and is primarily used for this purpose. Computer vision plays an important role in technology such as self-driving cars and robotics, and is also commonly used in photo editing programs (such as the auto-correct function on a mobile phone’s camera).

As their website describes it, OpenCV is a library of open-source computer vision and machine learning software. It is a BSD-licensed product with more than 2500 optimized algorithms, including both classic and cutting-edge machine learning and computer vision algorithms. Built with the goal of providing common infrastructure so that computer vision applications can speed the uptake of machine learning in business products, OpenCV facilitates businesses’ using and modifying code. These algorithms can be used to identify objects, track moving objects or camera movements, follow eye movements, detect and recognize faces or scenery, remove red eye from flash photographs, stitch multiple images together to make an entire scene in high resolution, search for similar images within a database, extract 3D models of objects, produce 3D point clouds from stereo cameras, and establish the markers for augmented reality overlays, among many other common applications [1].

OpenCV plays a key function in real-time operations, which makes it critical for today’s systems. By using OpenCV, we can process images and videos to identify objects, faces, or even a person’s handwriting. OpenCV is focused on providing tools which help in solving computer vision problems. Sometimes, the high-level functionalities provided by OpenCV are enough to solve computer vision’s complex problems. Most of the time, the basic components of OpenCV are enough to create a complete solution for almost any computer vision problem.

OpenCV uses machine learning algorithms to find faces in images. Because images are really 3D projections into 2D dimensions, it is not possible to truly recover the original data from a picture. There is not a simple test that will tell if the program has detected a face or not, so OpenCV breaks the operation into a lot of smaller tasks, called classifiers, which are easy to solve.

For faces, we need to check more than 6000 classifiers, and the face should match in all of them (with some approximation). To detect a face, the algorithm starts in the top left corner of a picture and moves downwards across small blocks of data, looking at each block and constantly asking if there is a face in that area. Since there are 6,000 or more tests in every block, this can result in millions of calculations, which take a long time.

To avoid this issue, the OpenCV algorithm uses cascades. The OpenCV cascade breaks the problem of detecting faces down into multiple stages. For each block, OpenCV runs a quick test, and if that test is passed, it completes a slightly more detailed test. Then it repeats this process, again and again if necessary. The algorithm may have 40-60 of these stages or cascades, and it will only detect a face if every one of the stages passes the test [11].

Because most of the blocks within the picture will quickly return a negative during the first steps, the algorithm will move on and not waste time testing all 6000 classifiers on that block. This makes the OpenCV algorithm so fast that it can be used to detect faces on streaming video in real time [2].

OpenCV comes with built-in cascades that can detect things including faces, eyes, hands, and legs. It uses the Haar Cascade algorithm [3].

## Discussion

In order to put this theory into practice, we must first create the cascade. Next, we must initialize it using the face cascade [4], [5].

```
face_cascade = cv2.CascadeClassifier( os.path.dirname(cv2.__file__) +
"/data/haarcascade_frontalface_default.xml" )[6]
```

The code above loads the face cascade data into the computer’s memory, making it ready to use. The cascade is just an XML file that contains the data to detect faces. Next, the image should be converted to grayscale:

```
gray = cv2.cvtColor( image, cv2.COLOR_BGR2GRAY )
```

This function “ *detectMultiScale”* is a general function for detecting objects. We call it on the face cascade (above), so it will detect faces. This function has a few arguments to configure precisely [8], [10].

```
faces = face_cascade.detectMultiScale(
   gray,
   scaleFactor = 1.1,
   minNeighbors = 5,
   minSize = (40, 40),
   flags = cv2.CASCADE_SCALE_IMAGE
) [9]
```

The detectMultiScale function returns a list of blocks which could possibly be faces. To visualise the detected faces, draw a rectangle around the faces using the following code.

```
For ( x, y, w, h ) in faces:
   cv2.rectangle( img, ( x, y ), ( x+w, y+h ), ( 0, 255, 0 ), 2 )
```

This function returns 4 data points: the rectangle’s height and width (h, w), and the x and y locations of the rectangle.

The resulting image with detected faces:

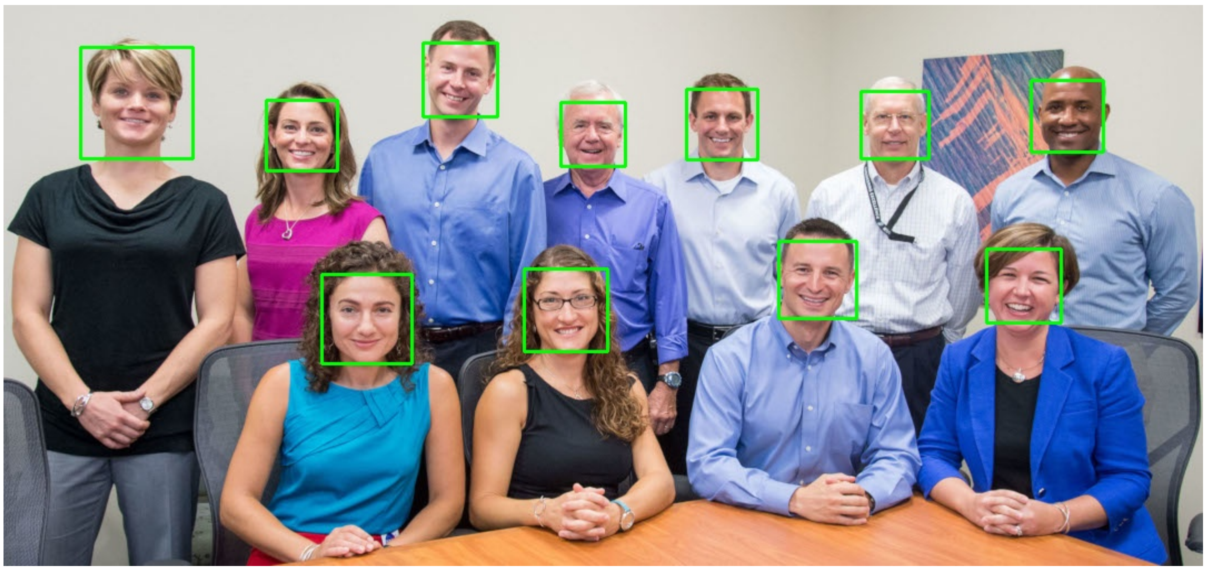

*Image 1: detected faces* [7]

We can also make the faces blur; OpenCV has multiple internal functions: *blur, medianBlur* and *GaussianBlur*, but they will modify the whole image. To get around this limitation, we can copy the part of the image where a face was detected:

*face = img[ y:y + h,  x:x + w ]*

And then make it blur:

*blur = cv2.GaussianBlur( face, (55, 55), 40 )*

Then, copy this part of the image back to the original:

*img[ y:y + h, x:x + w ] = blur*

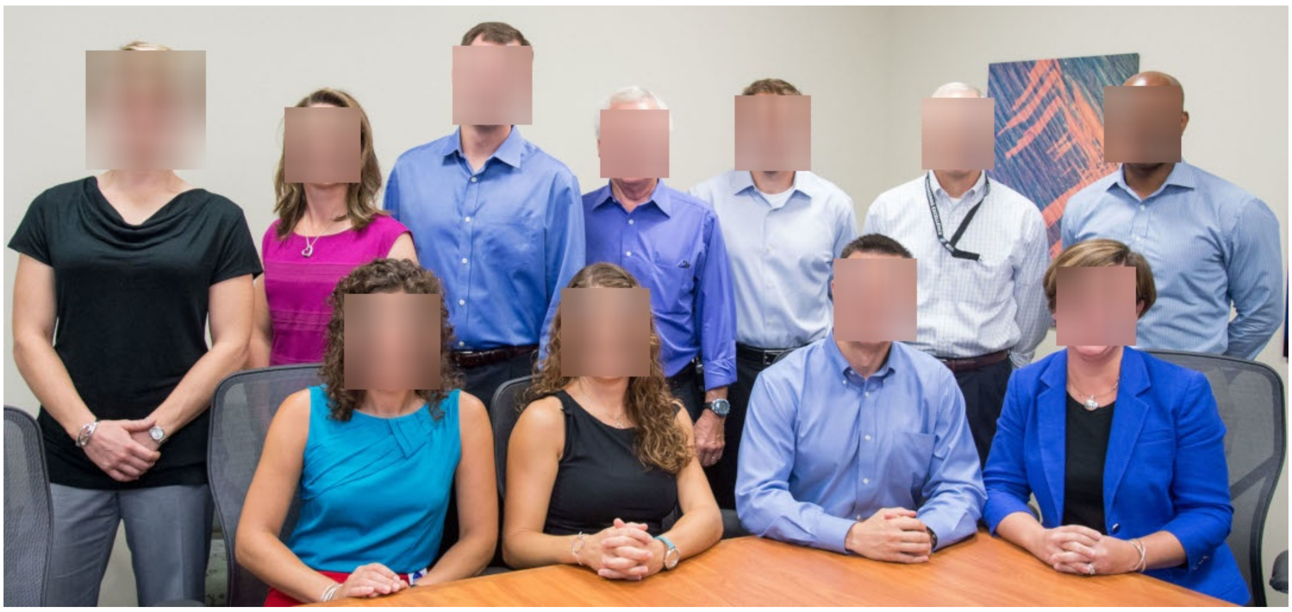

*Image 2: blurred face* [7]

Because videos are just a series of images, to detect faces on video, the same code can be used for each image. OpenCV provides a function “VideoCapture”:

*cv2.VideoCapture(“path of video file”)*

Then read each frame with function “read”:

*ret, frame = video_capture.read()*

After that, these frames can be processed as described earlier.

## Conclusion

OpenCV is a very useful and comprehensive library to solve computer vision problems. It is very fast and can be used on a standard laptop or smartphone. It is free to use and does not require a lot of resources to work. The accuracy of the result depends on the trained models. The default model may be less accurate then necessary, but the model can be trained with machine learning to make it better. The major advantages of using machine learning solutions are that the programs are simple and easy to write, and to increase accuracy we just need to retrain the model and update the model on users’ devices.  There is no need to rewrite the program to update it.

## REFERENCES

[1] Website OpenCV  https://opencv.org/;
[2] Website realpython.com;
[3] Website Haar Cascade https://docs.opencv.org/4.5.5/db/d28/tutorial_cascade_classifier.html;
[4] David Beazley, Brian K. Jones, "Python Cookbook, 3rd Edition", Publisher: O'Reilly Media, Inc, ISBN: 9781449340377, 2013;
[5] Andreas C. Müller, Sarah G, "Introduction to Machine Learning with Python". Publisher: O'Reilly Media, Inc. ISBN: 9781449369415, 2016;
[6] Sebastian R, "Python Machine Learning" ISBN 9781783555130, 2015;
[7] Sowa M, LOCATION: Bldg. 12, Room 253 - JSC Human Resources Conference (Photo), NASA, 2013. upload.wikimedia.org/wikipedia/commons/d/da/2013_NASA_class.jpg;
[8] Tom Hope, Yehezkel S. Resheff, Itay Lieder, "Learning TensorFlow", Published by O’Reilly Media, Inc, ISBN: 9781491978511, 201;
[9] Adrian Kaehler, Gary Bradski, "Learning OpenCV 3", Published by O’Reilly Media, Inc, ISBN: 9781491937990, 2017;
[10] Josh Patterson, Adam Gibson, "Deep Learning", Published by O’Reilly Media, Inc., ISBN: 9781491914250, 2017;
[11] Peter Morgan, "Machine Learning Is Changing the Rules", Published by O’Reilly Media, Inc., ISBN: 9781492035350, 2018.

Article received: 2022-02-03

ISSN 1512-1232