Fighting COVID-19 with the OpenCV Library

Yasin Sahin
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
sahin.yasin@student.htlwrn.ac.at

Julian Kerer
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
kerer.julian@student.htlwrn.ac.at

Fabian Hitzenberger
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
hitzenberger.fabian@student.htlwrn.ac.at

Sven Oberwalder
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
oberwalder.sven@student.htlwrn.ac.at

Raphael Ackerl
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
ackerl.raphael@student.htlwrn.ac.at

Karl Dopplinger
Department of Computer Science
HTBLuVA Wiener Neustadt
Wiener Neustadt, Austria
dopplinger.karl@student.htlwrn.ac.at

***Abstract*— The Corona virus still impacts our daily lives. In this document, some useful functions of the OpenCV (Open Source Computer Vision) library are explained. After that, two practical project ideas are listed. The projects are meant to help the humanity fight the currently ongoing pandemic. The one project being a fever detector and the other one evaluating rapid antigen tests.**

Keywords*— *OpenCV; library; functionalities; Covid-19; applications

I. Introduction

Mankind is dealing with another pandemic after a long time [7] and we humans are giving our best to prevent COVID-19 spread even more. Medicine and pharmacy industries have accomplished a big breakthrough by manufacturing vaccinations. Health organizations work on detecting infected people and treating them in the best way. Now it is time to use the technology more intensely. We believe that computer vision could help health authorities in repetitive tasks.

Our motivation comes from this year's Botball® game, which has also a reference to the Corona virus. The robots are tasked with isolating, sequencing and then synthesizing mRNA [1]. In this document, we will focus on one library - OpenCV. It is an open-source computer vision and machine learning software library.

We will start explaining what OpenCV can be used for (II) and then write about how we will use it for given tasks (III). Because OpenCV is a vast library, we will be just explaining functionalities we will use for our ideas. Our goal is to prove humanity we, roboticists, can also have a part in this -and effectively. When there are lots of positive cases, health authorities have problem evaluating too many Covid-tests. There are also people, who ignore their symptoms. We listed two practical project ideas to handle both problems.

II. Functionalities

A. Color Recognition

OpenCV is used for color recognition. The inRange() function allows to filter out pixels within a specified color range. This function requires an array, an upper and a lower boundary which includes RGB (=Red, Green, Blue)-values or HSV values if desired. If the three values of each color channel (or the hue, saturation, and value) of a pixel are in between the corresponding boundary values, it is considered in between the boundaries. It then creates an output-array, which consists of a set of numbers. These numbers can only be 0 or 255, a 0 meaning that the color-value is not in between the given boundaries and 255 the opposite [2]. This array allows us to create a mask and with this mask we can black out parts of the picture outside given boundaries. In the following picture it can be seen the boundary-values only accepting green. Color recognition will later be used in fever detection and antigen test evaluation under section III.

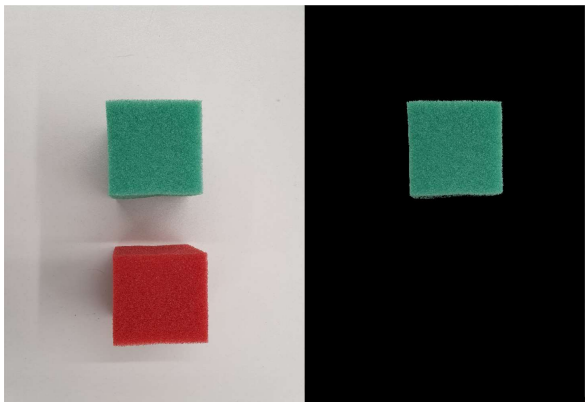

Figure 1: *on the left is the input image and on the right the output image, which is the result of the created mask. The mask uses following boundaries: (HSV) Lower: [71, 73, 0], Upper: [84, 255, 255]*

1) Problems

The example above uses hard coded values as value boundaries, whereas in reality this method should not be used. Assume that images or videos are captured in different lighting conditions. These are situations where we realize lighting plays a huge role on the output image. Colors might look very different under varying illuminations and when that happens the hard-coded boundary-values will fail. To prevent this, we have to adjust boundary- or the pixel values of the input image.

2) Solution

a) Color Correction Card

One way of solving the problem above is to use a color correction card [3]. We need to capture the card as well as the object we want to capture. Then you need to detect the color block region within the card and then apply histogram matching. The ArUco markers help detect the color block region.

Figure 2: *A color correction card (left) [9]*

b) Histogram Matching

This method requires a reference image. In this method we take the input image and adjust its pixel intensities so that the histogram of the input image matches the one of a separate reference picture. As mentioned, histogram matching can be applied to a picture with a color correction card. But it also works with two similar images even though there is no color correction card in them.

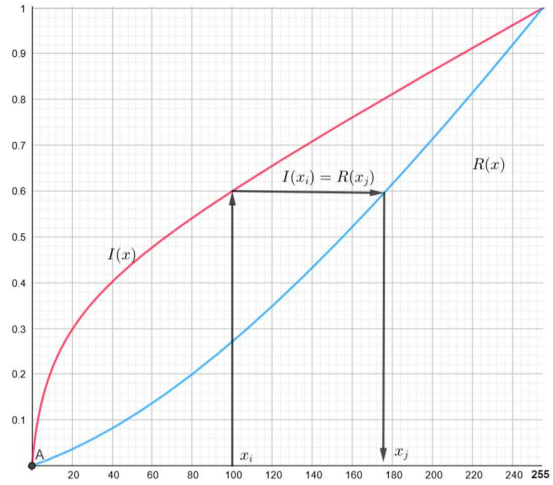

Figure 3: *Histogram of two images; visual description on how pixel intensities are transformed*

Histogram matching is applied to create aesthetic results or a basic color correction. This allows us to process images quicker and easier without the use of complex algorithms of machine learning.

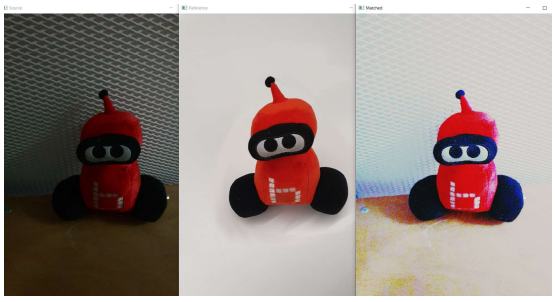

Figure 4: *The result of histogram matching; left is the input image, in the middle the reference image and on the right the output image*

B. *ArUco Markers*

ArUco markers are quadratic tags with a 2D pattern: a black border and a binary matrix within the borders. ArUco markers are an integral part of many computer vision systems, including but not limited to:

- 3D positioning
- Object size estimation
- Distance measurement
- And many more…

An ArUco dictionary specifies the type of ArUco tags we want to detect/generate. The ArUco tags' patterns are predefined in these dictionaries and if the pattern of a detected marker matches one in the dictionary, the ID of it will be returned. ArUco marker detection will be used in evaluating rapid antigen tests under section III.

C. *Face Detection*

Face detection is a process, which consists of analyzing the input image or video and determining the location, size, position, and the orientation of a face. One method which is used in OpenCV is the statistics-based approach. Haar classifier face detection is used to create a search window that slides through an image and check whether a certain region of an image looks like a face or not [5]. Haar-like features and a large set of very weak classifiers use a single feature, which tells if in a certain image a face is detected or not. It is possible to train classifiers with a plentiful of negative and positive examples, but OpenCV provides a pre-trained dataset of classifiers. Face detection will be used in fever detection under section III because fever detection will only be started if a face has been detected.

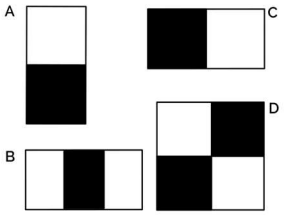

Figure 5: *Haar-like features*

Haar-like features' value is the difference between the sum of pixels located in the white area(s) and the sum of pixels located in the black area(s) [4]. If this difference is closer to 1 than 0 the feature could be detected. A feature can be an edge, a line, or any structure in the image where there is a sudden change of intensities. For example, the feature C in figure 5 would detect an edge with darker pixels on its left and lighter pixels on its right.

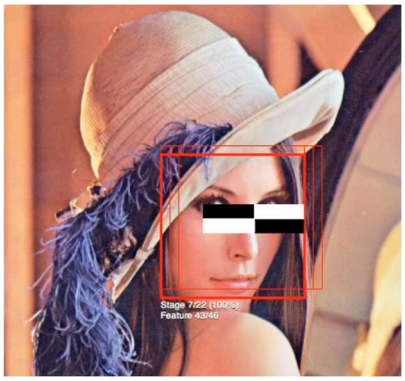

Figure 6: *Scanning an image for a face visualized; you can see a Haar-like feature at the eyes [10]*

1) Problems

The illumination and other factors may decrease the accuracy of the predictions. To obtain correct predictions the image needs to be preprocessed.

2) Solution

The first step to the solution is to create a blob (=binary large object). Blobs can store binary data and are used to store images, audio, or other multimedia files. Preprocessing tasks normally include:

- Mean subtraction
- Scale

For the mean subtraction task, we need to compute the average pixel intensity (=mean). All means for each color channels are described with three variables:

$$\mu_R, \mu_G, \text{ and } \mu_B \tag{1}$$

The means from (1) are then subtracted from the color channels of the input image.

$$\begin{aligned} R &= R - \mu_R \\ G &= G - \mu_G \\ B &= B - \mu_B \end{aligned} \tag{2}$$

We can also divide by a scaling factor, which adds in a normalization:

$$\begin{aligned} R &= (R - \mu_R) / \sigma \\ G &= (G - \mu_G) / \sigma \\ B &= (B - \mu_B) / \sigma \end{aligned} \tag{3}$$

If we need to mean subtract across a set of images, the value of the scaling factor σ should be the standard deviation across the set [6]. Otherwise, OpenCV does not necessarily require a calculated scale factor. You can set it to 1.

The blobFromImage() function creates blobs. The function has following parameters:

- image: The input image
- scalefactor: Scale factor (σ), optional
- size: spatial size the neural network expects
- mean: a tuple of three mean values
- swapRB: if your mean tuple is in the (R, G, B) order, then it should be set to True. The reason for this is, that the function expects it to be in the (B, G, R) order

III. Application

In this section, we will list possible Covid-related implementations of the functionalities mentioned above.

A. IR Fever Detector

1) Concept/Design

After the lockdown in November 2021 in Austria, we needed a vaccination, or a proof of recovery to be able to go almost anywhere (e.g., to restaurants, museums, amusement parks, events, etc.). At the school or workplace, we had to make a Covid-test [8]. But places which did not require them often checked for symptoms, especially your temperature, even though it was not mandatory. Bots are designed to do repetitive tasks, and because checking for fever is a repetitive task, we propose to automate this task using methods of computer vision. Reference [12] shows a project very similar to this. isual fever detection was also used during the swine flu pandemic caused by the H1N1 virus at the Narita International Airport (Japan).

2) Implementation

This bot is placed at the entrance of the building. It will adjust its camera until it centers a face and then it will analyze its input stream. If it detects a large area with temperatures between 37.5°C and 41°C (= 99.5 F and 105.8 F) it will alert an employee for further examination of that person. Furthermore, the temperature check will not be started if a face could not be captured. It is unnecessary to analyze the input from the IR-camera if there is no one in front of it.

3) Used functionalities

Color recognition (section II A) is used to scan the thermos-picture and the face detection functionality (section II C) to adjust the camera so that a face is centered. A histogram matching is not needed for the reason that the illumination in a thermos-picture will not change that much.

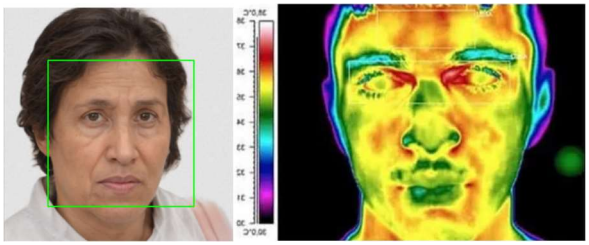

Figure 7: *Left: A face being detected; Right: temperature analysis [11]*

B. Antigen Tester

1) Concept/Design

Rapid antigen tests are a quick way to detect a Corona-infection. In Austria antigen tests are done for that reason and also for its availability. A robot could be used to automate the process of evaluating a Covid-test. It would also be quicker and a basic algorithm with OpenCV and a camera is all it needs. Reference [14] shows a project on interpreting rapid diagnostic test results designed just for smartphones.

2) Implementation

Rapid antigen tests use (depending on the manufacturer) barcodes as their IDs. But we will use ArUco markers because we explained them. After we can tell the result of the test using a color detection algorithm, its result will be updated in a database with the matching ID. The ID can be determined by scanning the ArUco marker. To avoid detecting red areas outside of the result area, we will use the position and size of the ArUco tag. We know that the position of the ArUco marker and the red line(s) should be approximately the same. The width of the red lines should be less or equal to the width of the ArUco marker.

3) Used functionalities

Color recognition (section II A) is used to detect the red lines on the antigen tests. Because the pictures can have different illumination it is also necessary to apply histogram matching (section II A 2 b). We also have to detect ArUco tags (section II B)

Figure 8: *Antigen-test example: left: negative result, middle: positive result, right: invalid*

IV. Conclusion

In this document, we have mentioned and described useful functionalities of the OpenCV library and explained how they work. These functionalities were: color recognition, detecting and generating ArUco markers, and face detection. We also found solutions for problems which occur. All these functionalities were then used in III in Corona related applications. Since we restricted our main tools just to the OpenCV library, we listed two practical project ideas. But we are very convinced that we roboticists can help in the pandemic in much more ways.

Acknowledgment

The authors of this paper would like to thank Dr. Michael Stifter for his constant support throughout this paper and the seniors of robo4you organization for proofreading and giving feedback. Also special thanks to HTBLuVA Wiener Neustadt, which provided us with a great robotics lab. The co-work with the Slovak team also helped us improve this paper.

References

[1] KIPR, *2022 Botball game review,* KIPR, 2022, [Online] Available: https://www.kipr.org/wp-content/uploads/2022%20Botball/2022%20Botball%20Game%20Review%20v1.4.pdf [Accessed Mar. 4, 2022.]

[2] EDUCBA, *OpenCV inRange*, EDUCBA, [Online] Available: https://www.educba.com/opencv-inrange/ [Accessed Mar. 9, 2022.]

[3] A. Rosebrock, *OpenCV and Python color detection,* Aug. 4, 2014 [Online] Available: https://pyimagesearch.com/2014/08/04/opencv-python-color-detection/ [Accessed Mar. 11, 2022.]

[4] OpenCV, *Cascade Classifier*, OpenCV, [Online] Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[5] P. Y. Kumbhar, M. Attaullah, S. Dhere., S. Hipparagi, "Real time face detection and tracking using OpenCV", vol. 4, issue 4, International Journal for Research in Emerging Science and Technology, 2017

[6] A. Rosebrock, *Deep learning: How OpenCV's blobFromImage works,* Nov. 6, 2017 [Online] Available: https://pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/ [Accessed Mar. 11, 2022]

[7] Centers for Disease Control and Prevention, *History of 1918 Flu Pandemic*, Mar. 21, 2018 [Online] Available: https://www.cdc.gov/flu/pandemic-resources/1918-commemoration/1918-pandemic-history.htm [Accessed Apr. 3, 2022.]

[8] "Wegweiser durch 2G-Österreich: Die wichtigsten Fragen und Antworten"[Guide through 2G Austria: The most important questions and answers], *Der Standard*, Nov. 6, 2021, [Online] Available: https://www.derstandard.at/story/2000130955558/wegweiser-durch-2g-oesterreich-die-wichtigsten-fragen-und-antworten [Accessed Apr. 3, 2022]

[9] Pantone, "Pantone color match card". [digital image], Pantone, https://www.pantone.com/eu/de/pantone-color-match-card [Accessed Apr. 3, 2022]

[10] A. Harvey, "OpenCV Face Detection: Visualized", June 22, 2010. [video file] Vimeo. Available at: https://vimeo.com/12774628 [Accessed Apr. 3, 2022].

[11] Flir, "Use of infrared to detect elevated body temperatures" p. 1, [digital image] Available at: http://support.flir.com/appstories/AppStories/Medical/Swine_Flu_EN.pdf [Accessed Apr. 3, 2022]

[12] A. Somboonkaew et al., "Mobile-platform for automatic fever screening system based on infrared forehead temperature," *2017 Opto-Electronics and Communications Conference (OECC) and Photonics Global Conference (PGC)*, 2017, pp. 1-4, doi: 10.1109/OECC.2017.8114910.

[13] H. Nishiura and K. Kamiya, "Fever screening during the influenza (H1N1-2009) pandemic at Narita International Airport, Japan", *BMC Infectious Diseases*, vol. 11, no. 1, 2011. doi: 10.1186/1471-2334-11-111.

[14] C. Park et al., "Supporting Smartphone-Based Image Capture of Rapid Diagnostic Tests in Low-Resource Settings", *Proceedings of the 2020 International Conference on Information and Communication Technologies and Development*, 2020. doi: 10.1145/3392561.3394630 [Accessed 18 April 2022].