# Image color calibration by using differential evolution and a convolution kernel

Cristina Velázquez
*Facultad de ingeniería*
*Universidad Panamericana*
Aguascalientes, México
0224433@up.edu.mx

Fernanda Zavala
*Facultad de ingeniería*
*Universidad Panamericana*
Aguascalientes, México
0227729@up.edu.mx

***Abstract*—Images are the main source of information in multiple areas. Because of this, problems such as image processing have taken priority, leading to the creation of multiple methods that seek to transform images into reliable data. Image enhancement is one of the main issues in image processing. It is the task of applying certain transformations to an input image to obtain a visually more pleasant, detailed, or less noisy output image. It helps to obtain information as it facilitates both image segmentation and the recognition or interpretation of data contained in the image. There are many proposed methods to enhance images. This paper proposes one that seeks to calibrate the image's colors. This is helpful when working with images that present bad illumination conditions, thus creating output images in which objects are not easily recognizable and can be misinterpreted. The proposal uses a digital color calibration chart that contains 24 colors that are considered standard values. Then whenever an image is taken, it must contain a printed version of the same calibration chart alongside any object that desires to be color calibrated. Afterward, with the use of cv2 functions and differential evolution algorithm, a convolution kernel is found. This kernel serves as a transformation matrix. When this is applied to an input image, it reduces the differences between the goal colors (the ones present on the digital color calibration chart), and the colors present on the calibration chart contained in the input image. This method may be a good starting point in the creation of different convolution kernels that can emphasize certain colors in different images, thus creating a powerful tool for object and color detection.**

***Index Terms*—differential evolution, kernel, convolution, calibration**

## I. Introduction

Images are changing the way humans perceive and interpret the world around them. They have slowly become one of the primary sources of information in multiple areas, such as medicine, agriculture, navigation, geography, and many more. Usually, images must be processed before they can be useful. One of the main issues in image processing is image enhancement. It helps to obtain information from the images, as it facilitates both image segmentation and the recognition or interpretation of information each image contains. [5] Image enhancement is the task of applying certain transformations to an input image to obtain a visually more pleasant, detailed, or less noisy output image. [3] This paper presents an image enhancement method that allows the calibration of the colors an image contains. This is helpful when working with images that were taken under bad illumination conditions. Multiple people have proposed similar image enhancement methods, each one of them with different purposes, yet all of them lead to the creation of images better prepared to deliver important data.

According to [1] image enhancement is treated as an optimization problem. Some of the most important methods used to solve this are classical and hybrid metaheuristics. The procedures applied in classical metaheuristics are Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, and Simulated Annealing. A proposed solution mentioned in this paper is a hybrid image contrast enhancement approach. The process is as follows: First, they make color quantization using the K-Means algorithm, then create a segmentation of the objects using multi-level thresholding, they follow with indexing of the image, continue with creating an alpha value prediction of different objects using a genetic algorithm, followed by developing object enhancement using Artificial Neural Network using the alpha values obtained previously, and finalize by performing light intensity normalization using histogram equalization method. [1] [20]

Continuing with the use of algorithms, in [5] simulated annealing, DE, PSO, and harmony search are used to obtain the best optimization for gray-level and color image contrast enhancement. When working with colored images, they convert RGB color space to YIQ (luminance, hue, saturation) color space, then a metaheuristic algorithm is applied to the saturation component and finish by turning back to RGB.

Accuracy is of the utmost importance when performing a diagnosis. This is why many of the previously created methods have been applied to biomedical images. A method for contrast optimization for biomedical image enhancement based on a metaheuristic (elitist genetic algorithm) and a novel convolutional kernel is presented in [2]. In this case, the image under consideration has 256 gray levels. The method may be used

on color enhancement too, yet it may not always work as it can lead to loss of information and the distortion of the image. Having a similar approach, on [3]a real-coded genetic algorithm (GA) is applied. It contains significant modifications such as PCA mutation and it evolves the parameters of a local enhancement method (enhancement kernel) that better adapts to the local features in the image.
Lately, underwater images have been considered when creating these processes. Because of the absorption and scattering of light in an underwater environment, images taken underwater tend to have poor contrast and resolution. This then leads to having a color that is more dominant than the other ones. There have been many proposed solutions, one of them being to apply a differential evolution algorithm to improve the contrast of underwater images on each RGB component. The contrast limits are specified by using the algorithm. [4]

All the previous processes are important and have been applied to develop images better qualified to solve different problems. Yet, none of those methods use a calibration chart, something that is necessary in out proposed method.
In reference [6] all of the input images contain a color calibration chart (24 colors). This chart is used to map the average RGB values of certain color patches (present in the calibration chart) to standard RGB values determined in a digital calibration chart. A color calibration matrix is derived based on the mapping and then applied to each pixel of the input image. It's important to recognize that on each image, the color patches selected to create the mapping depend on the light conditions the input image has. They do not calibrate the 24 colors on all the images, something that makes the approach different from the one proposed in this paper.
Our proposed method is very similar, yet it contains some major differences. All our input images contain a color calibration chart, yet the selection of this part of the image is done automatically with the help of some cv2 functions. Then, when only the calibration chart is selected, the RGB components of each color present in the chart are saved. The goal is to transform the values of the RGB components present on the input image to those RGB component values present on a digital color calibration chart that is defined as a standard. To achieve this, we seek a convolution kernel (3x3) using differential evolution. When the kernel is defined, it is later on applied to the entire input image and thus creates a resulting image whose colors are more similar to those defined as the standard.

## II. Methodology

This proposal is divided in two sections:

- Image processing to extract the color calibration chart.
- Use of differential evolution to calibrate the image's colors.

### *A. Image processing to extract the color calibration chart*

First, every input image needs to contain an object alongside a color calibration chart. Both the input image and a digital copy of the color calibration chart previously used are read and saved. The digital version of the color calibration chart is very important, as it is established as the standard which the input image's colors must reach. Then, the contours present on the input image are found. To achieve this, the following steps must be followed:

- The input image must be turned into grayscale using cv2.cvtColor().
- A threshold has to be applied to the image with cv2.treshold().
- Contours are the boundaries of a shape with the same intensity, they are useful in object detection. All the image's contours are found using cv2.findContours(). This function stores the (x,y) coordinates of the boundary of a shape. It is what will help us detect the calibration chart within the input image.

When the contours present on the image are found, it is necessary to detect the biggest one, as that should be the calibration chart. Because of this, one of the main limitations of the method is found: the objects whose colors are meant to be calibrated must be of a different shape than the one the calibration chart has. This means that they mustn´t be rectangular. To find the biggest contour, it is needed to find the contour which has the biggest area, to achieve that cv2.contourArea is applied.
Once the calibration chart's contour is detected,the function cv2.approxPolyDP() is applied to determine the shape of the polygons present in the image.
Once the corners of the calibration chart are identified, they are stored in a list and then sorted, this is only to be able to identify the order of the coordinates. To be capable of recognizing the chart, it's necessary to create a rectangular shape within the input image with the coordinates of the corners obtained with the previous methods. To create the right shape, there must be an order between the coordinates used.
One thing that is important to consider is that the input image is not always going to be taken with the same conditions. This means that, sometimes the calibration chart may be at a different angle, smaller or bigger. Since the main comparison of this method is realized between the digital and input version of the calibration chart, it is necessary to make sure that both images are always of the same size and are viewed on the same perspective. To achieve this, the functions cv2.getPerspectiveTransform() and cv2.warpPerspective are applied. The first one calculates a transformation matrix that can transform a combination of the rotation, scale, translation and projection of one image into another. The second one applies this transformation matrix to the image.
Finally, it's needed to ensure that the size of the input's calibration chart is the same as the digital version of the chart. This is achieved by using cv2.resize().
To demonstrate that the placement of the chart is identified on the input image, cv2.drawContours() is used to draw the shape of the contour found with the previous functions and cv2.circle() is used to draw circles on each of the corners. This is just to evidentiate that the calibration chart is properly

identified within the input image. [14] [15] [16] [17]

### B. Use of differential evolution to calibrate the image's colors

Convolution is to use a kernel to extract some features from an image. A kernel is none other than a matrix of weights, that is moved across an image and multiplied with the input such that the output is enhanced in a certain way. The values within the kernel determine what features are extracted from the image it is applied to. [18]
The purpose of this section is to find a kernel that allows the colors of the input image to be as similar as possible to the colors that the digital version of the calibration chart contains. In order to find this kernel, differential evolution is applied.
Differential evolution is an algorithm proposed by Storn and Price in 1995. Here, individuals are seen as vectors. There are three main operators in differential evolution algorithm which are mutation, crossover and selection. The novelty in this algorithm is the mutation operator, which uses three individuals to mutate another one, and the mutation depends on the distance. The individual's here converge to the local minimums, and eventually, all converge to the global minimum. It is a robust algorithm for solving continuous multidimensional optimization problems. [4] [22]
For differential evolution to obtain the convolution kernel it is necessary to obtain the RGB components of the colors present in both color calibration charts that are to be compared. To do this both images are read and divided into the 24 sections of color they have. Instead of only taking one pixel of each color patch, a whole section of it is stored in an array. This way there are two arrays, each containing samples of all the colors present on each color calibration chart to be compared. Then, boundaries and an objective function are needed to start using DE. The boundaries are a min and max value of [-2,2] within a vector of 9 touples. It looks like this ([-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2],[-2,2]). The objective function reshapes the boundary vector into a 3x3 matrix (this is the kernel), then creates an image by applying the received convolution kernel to the image previously created that only contains the samples of colors present on the calibration charts. Finally, the difference between the image with the standard colors and the one to which the kernel was applied to represents the fitness of each individual. The goal is to look for the kernel that obtains the best fitness.

## III. Experiments and results

### A. Metrics used to measure the results

The metrics used to determine the method's success are: SSIM, histogram comparison and macro f1.
SSIM is used to measure the similarity between two images. This metric extracts three key features from an image:

- Luminance: It's measured by averaging over all the pixel values.
- Contrast: Measured by taking the standard deviation of all the pixel values.
- Structure: Obtained by dividing the input signal with its standard deviation so that the result has unit standard deviation.

The values this metric calculates are between 0 and 1. A value of 1 shows that the two images are very similar, or even the same one. And a value of 0 represents that the images are really different. [21]
You can find the similarity between two images using the cv2.compareHist() function. This compares the histogram of two images according to the colors they have. It only compares the images based on the colors they have, the shapes or objects within the images are not relevant. This is useful since the proposed method needs to compare the color difference between the two images The steps to follow are:

- Convert the images to HSV using the function cv2.cvtColor().
- Find the images histograms using cv2.calcHist()
- Normalize the histograms for comparison using cv2.normalize().
- Use the normalized histograms to cv2.compareHist().

If the similarity is 1, the images are identical. The closer to 0 it is, the greater the difference between the images. [19]
F1 is like an average between precision and recall, the values range between 0 and 1, where 1 is that all predictions are correct. When having a multi-classification problem macro-f1 score is used. [20]

### B. Experiments made and results

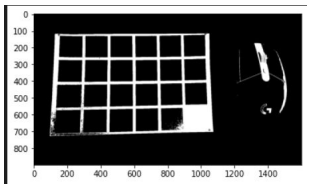

Fig. 1. Applying threshold to obtain the image's contours.

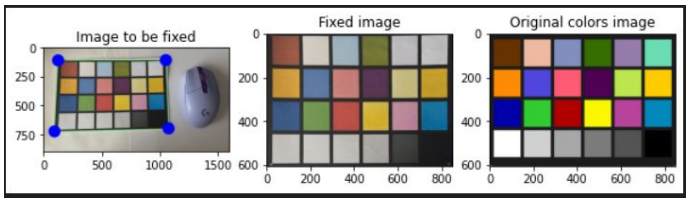


Fig. 2. Extracting the color calibration chart from the input image and place it against the standard calibration chart.

Multiple experiments were performed. In general all of them contained a population size of 200 and sought a convolution kernel (3x3) with bounds [-2,2] for each value. Those bounds were selected, as the use of greater values such as [-5,5] led to color distortion and generally provided not positive results. On the paper are shown and compared most of the experiments performed. Each of them with different images, and a different amount of iterations.In order

to optimize the problem, the images used in the differential evolution algorithm only contain the colors present on both color calibration charts. Once the convolution kernel is found, it is applied to the entire input image.

**1000 iterations:**

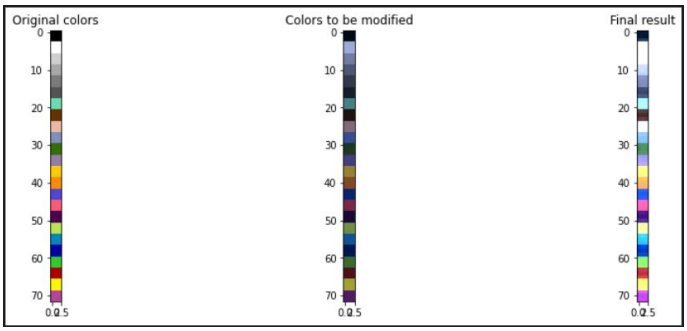


Fig. 3. Modification of image's colors using differential evolution.

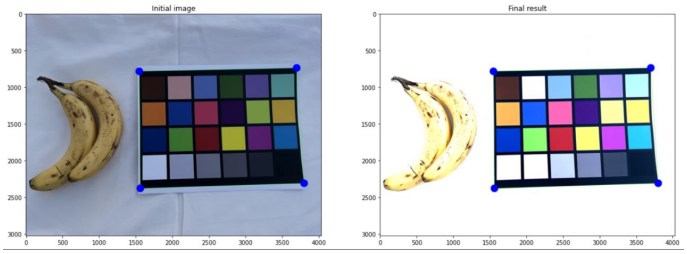


Fig. 4. Initial input with convolution kernel applied.

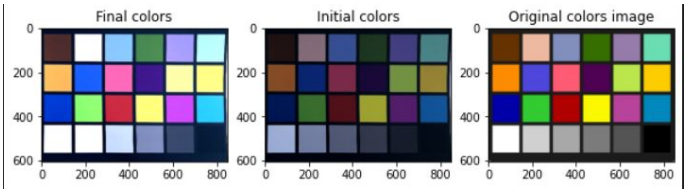


Fig. 5. Color charts with convolution kernel applied.

The previously mentioned metrics are used to compare both color calibration charts: the digital one, which is considered the standard, and the one found on the input image with the convolution kernel applied. First, the SSIM (Structural Similarity Index) throws a value of 0.68219997. Since this is closer to 1 than to 0, the images are somewhat similar. The similitudes between them are bigger than their differences. When comparing the images' histogram colors, the result is 0.75079785. Since this is a value closer to 1 than 0, there are many similarities between the distributions of color present in both images. When obtaining the macro-f1 score, the value is 0.00310505. This is in no way a good result, but it is important to consider that in this comparison, not only the colors are considered. This uses both precision and recall, meaning that the position in which each color is present on both images also plays an important part. To really check if there was any improvement on the final image due to the method's application, it is necessary to obtain the same metrics, only using the input color calibration chart without the convolution kernel's application. On SSIM, the value is 0.63296517. The histogram comparison results in 0.00686958 and the macro-f1 throws 0.00140447. The differences present in the SSIM and macro-f1 values are really small, yet they represent an improvement created by the applied method. Yet, when both histogram results are compared there is a significant difference. This demonstrates that the applied method does change the image's colors in a positive way, making them more similar to those present on the digital version of the calibration chart.

**2000 iterations**

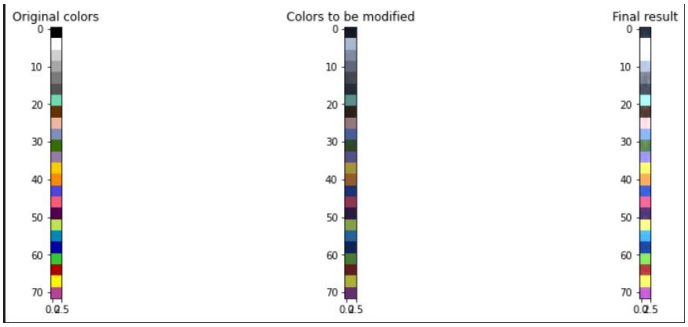


Fig. 6. Modification of image's colors using differential evolution.

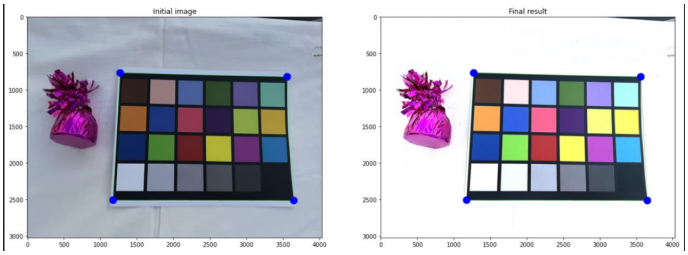


Fig. 7. Initial input with convolution kernel applied.

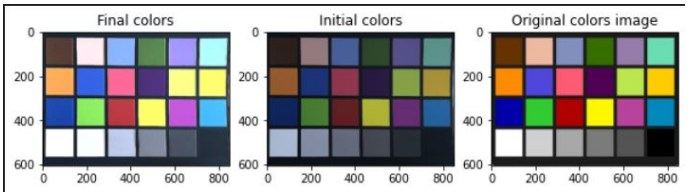


Fig. 8. Color charts with convolution kernel applied.

Applying the same process as before, the first step is to check the metrics values when the convolution kernel is applied to the input calibration chart. The SSIM throws a value if 0.52955922, allowing the understanding that both the image's differences and similarities are almost equal. Then the histogram comparison returns a value of 0.74140943, demonstrating that the distribution of colors present on both

images are fairly similar. Finally, the macro-f1 score calculates the value of 0.00424797. To gain a complete comparison, the metrics values when the convolution kernel is not applied are as follows: SSIM is 0.75037968, histogram comparison 0.00420556 and the macro-f1 is 0.00294669. The SSIM is actually higher when the kernel hasn't been applied, showing that at least, considering this metric, the method did not accomplish its goal. Yet, both the histogram comparison and macro-f1 score have improvements. The main thing to notice is the great difference between the values obtained with the histogram comparison. After the application of the kernel, this value increases considerably, showing that, once again, with a different object, the method keeps helping to increase the similarity between the input and standard values present on the calibration chart.

It is not possible to demonstrate the entirety of the experiments made. Yet the final results tend to have the similar behaviour. Based on the metrics, there is almost always an improvement on the values obtained before and after the kernel is applied to the image. Without considering the metrics, by only seeing the results obtained it is possible to understand that what this method has mainly achieved is the brightening of the image's colors. This leads to the enhancement of darker colors. For example, pink, orange, blue and red are some of the colors that end up being fairly similar to the desired tone after this method is applied on the image. Yet, lighter colors, such as gray and pastel tones become so much brighter than they were originally, making them almost white in most cases.

An important factor on these experiments, is the quality of the calibration chart used on the input image. Its colors must be as similar as possible to the ones present on the digital version of the image. By ensuring this, the only thing the method must focus on is working with bad illumination conditions, otherwise it will sometimes be difficult to detect certain colors as they already seem almost white, even in perfect light conditions.

## IV. Conclusion

The proposed method uses a color calibration chart and differential evolution to find a convolution kernel that when applied to an input image transforms it to make its colors as similar as possible to those considered as a standard. This can be helpful in image enhancement as it can fix illumination problems on different images. The process starts by identifying the color calibration chart present in an input image with the use of multiple cv2 functions. Then, with the use of differential evolution a convolution kernel is calculated by finding the transformation that reduces the color differences between the input's calibration chart and a digital calibration chart that is considered as the goal to reach.

The final results demonstrate that what the methods achieves is to generally brighten the colors present on the input image. This can be useful when working with images that contain darker colors, as it enhances them and can fix illumination problems. On the other hand, if the goal is to enhance lighter tones, this method is not useful, as it tends to turn them into white, thus eliminating that information from the image.

The amount of iterations used within the differential evolution algorithm vary between 1000, 2000 and 5000. All of them reach the same result of image brightening.

Some of the method's disadvantages are:

- The impossibility of using rectangular objects on the input image alongside the color calibration chart. This can cause problems when detecting the calibration chart, thus making impossible to complete the method's process.
- The importance of using on the input image a calibration chart with colors very similar to those considered a standard. Otherwise, the method won't only worry about fixing illumination issues, as it may be incapable of detecting certain colors and confuse them for white, resulting in and a badly calibrated image.
- The impossibility of calibrating bright colors.

In a future those problems may be solved by changing determinate parts of the process. This method may be a great staring point in the creation of different convolution kernels that can emphasize certain objects and colors within images. Later, that can be applied in the creation of a powerful tool that can help many areas such as medicine, artificial intelligence and map creation.


## Acknowledgment

This is to thank our Optimization and Metaheuristics II teacher Claudia Nallely Sánchez Gómez and our college Universidad Panamericana.


## References


[1] S. Chakraborty et al., "Bio-medical image enhancement using hybrid metaheuristic coupled soft computing tools," 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017, pp. 231-236, doi: 10.1109/UEMCON.2017.8249036.

[2] S. Chakraborty, A. Raman, S. Sen, K. Mali, S. Chatterjee and H. Hachimi, "Contrast Optimization using Elitist Metaheuristic Optimization and Gradient Approximation for Biomedical Image Enhancement," 2019 Amity International Conference on Artificial Intelligence (AICAI), 2019, pp. 712-717, doi: 10.1109/AICAI.2019.8701367.

[3] C. Munteanu and A. Rosa, "Gray-scale image enhancement as an automatic process driven by evolution," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 34, no. 2, pp. 1292-1298, April 2004, doi: 10.1109/TSMCB.2003.818533.

[4] G. E. Güraksin, U. Köse and Ö. Deperlıoğlu, "Underwater image enhancement based on contrast adjustment via differential evolution algorithm," 2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), 2016, pp. 1-5, doi: 10.1109/INISTA.2016.7571849.

[5] L. M. Rasdi Rere, M. Ivan Fanany and A. Murni, "Application of metaheuristic algorithms for optimal smartphone-photo enhancement," 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE), 2014, pp. 542-546, doi: 10.1109/GCCE.2014.7031307.

[6] S. Sunoj, C. Igathinathane, N. Saliendra, J. Hendrickson, D. Archer, Color calibration of digital images for agriculture and other applications, ISPRS Journal of Photogrammetry and Remote Sensing, Volume 146, 2018, Pages 221-234, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2018.09.015.

[7] Ashish Kumar Bhandari, Bharath Subramani, Magudeeswaran Veluchamy, Multi-exposure optimized contrast and brightness balance color image enhancement, Digital Signal Processing, Volume 123, 2022, 103406, ISSN 1051-2004, https://doi.org/10.1016/j.dsp.2022.103406.

[8] I.J. Image, Graphics and Signal Processing, 2015, 7, 69-76. Published Online June 2015 in MECS (http://www.mecs-press.org/)DOI: 10.5815/ijigsp.2015.07.08

[9] de la Fraga, L.G., Schütze, O. Direct Calibration by Fitting of Cuboids to a Single Image Using Differential Evolution. Int J Comput Vis 81, 119–127 (2009). https://doi.org/10.1007/s11263-008-0183-z

[10] Alsmadi, M.K. Content-Based Image Retrieval Using Color, Shape and Texture Descriptors and Features. Arab J Sci Eng 45, 3317–3330 (2020). https://doi.org/10.1007/s13369-020-04384-y

[11] T. W. K. Poon and M. R. Friesen, "Algorithms for Size and Color Detection of Smartphone Images of Chronic Wounds for Healthcare Applications," in IEEE Access, vol. 3, pp. 1799-1808, 2015, doi: 10.1109/ACCESS.2015.2487859.

[12] SAFAEI, AMIR and FAZLI, SAEID (2018) "A novel solution in the simultaneous deep optimization of RGB-D camera calibration parameters using metaheuristic algorithms," Turkish Journal of Electrical Engineering and Computer Sciences: Vol. 26: No. 2, Article 10. https://doi.org/10.3906/elk-1706-250

[13] Li Deng, Gen Lu, Yuying Shao, Minrui Fei, Huosheng Hu, "A novel camera calibration technique based on differential evolution particle swarm optimization algorithm", Neurocomputing, Volume 174, Part A, 2016, Pages 456-465, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2015.03.119.

[14] GeeksforGeeks. (2019, April 29). Find and Draw Contours using OpenCV — Python. https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/

[15] OpenCV: Contour Features. (2022, November 28). https://docs.opencv.org/4.x/dd/d49/tutorial-py-contour-features.html

[16] (2022, June 8). OpenCV ApproxPolyDP. EDUCBA. https://www.educba.com/opencv-approxpolydp/

[17] Atul, K. ., Atul, K. . (2020, November 6). cv2.getPerspectiveTransform(). TheAILearner. https://theailearner.com/tag/cv2-getperspectivetransform/

[18] Ganesh, P. (2021, December 12). Types of Convolution Kernels: Simplified - Towards Data Science. Medium. https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37

[19] Ali, A. (2022, January 29). OpenCV Compare Images. Delft Stack. https://www.delftstack.com/howto/python/opencv-compare-images/

[20] Shmueli, B. (2022, July 21). Multi-Class Metrics Made Simple, Part II: the F1-score. Medium. https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

[21] Datta, P. (2021, December 15). All about Structural Similarity Index (SSIM): Theory + Code in PyTorch. Medium. https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e

[22] Sánchez, C. "Differential Evolution (DE)".