IMAGE DIFFERENCES ANALYSIS USING FINDCONTOURS

Kevin Juan Rizky Hutomo
Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata kevinhutomo@yandex.com

ABSTRACT

One of the common problems in find the difference games is trying to quickly and accurately detect the difference between two pictures. Especially if the image has a similar appearance that is difficult to detect with the naked eye. For example, in a game, many of them have difficulty distinguishing based on color, number of objects, pattern complexity, or the like. This project concentrates on how the system can look for differences in the two images by the color of each pixel. Images taken via the neok12 and hellokids websites will then be checked for differences in each pixel and use the findContours algorithm to extract the contours. The results obtained from this study are 72% success in detecting differences and having an error rate of 28% in 50 trials using 100 images.

Keywords: cv2.findContours, Find Differences, OpenCV

INTRODUCTION

One of the common problems in finding the difference games is trying to detect the differences between two pictures quickly and accurately. This is especially so if the image has a similar appearance that is hard to detect with the naked eye. For example, in a game, many of them have difficulty distinguishing based on color, number of objects, pattern complexity, or the like. Of the many games available, some don't provide clues to help find the differences and also have to find all the differences to be able to go to the next level, or have a time limit. By using OpenCV, we can easily complete all of these challenges, even though they sound not challenging, they still sound interesting to complete.

OpenCV is a library that can be used to process and analyze images and videos. By using OpenCV, we can quickly search for differences between two images. OpenCV has the advantage of fast calculations because it only depends on the number of pixels in an image. OpenCV can also be used to process images to detect differences between two images based on color, number of objects, pattern complexity, or the like.

Based on the results of this study, OpenCV is a solution that can be used to find differences in two images. In the first stage, OpenCV will read the two input images and read each pixel individually before comparing their level of similarity and saving them in black and white. The program will look for contours in the image and create bounding boxes in any areas where there are differences by combining the bounding boxes if they touch each other. The program also calculates the percentage of similarity between two images. Therefore, I want to help people who

have trouble spotting the differences between two pictures quickly and accurately.

RESEARCH METHODOLOGY

This chapter will explain the entire research methodology used. The research methodology is divided into three major parts: looking for differences based on color, searching for contours, and combining bounding boxes, as shown in Figure 1 and explained in detail in the next sub-chapter.

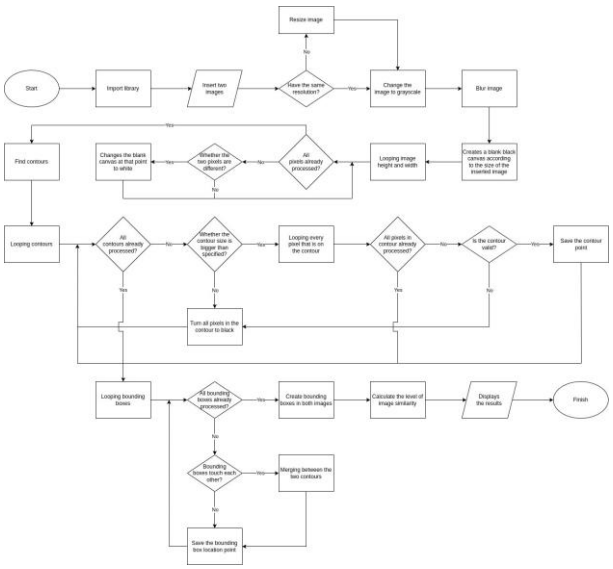


Figure 1. Research Methodology

Looking for Differences Based on Color

Iterates based on the height and width of the image to check if the two pixels at that point differ from the specified color difference tolerance limit. Then if the two pixels are different then the pixels at the point on the canvas that has been made will be changed to white and the difference will be calculated.

Searching for Contours

Finds a contour on the black canvas that has been filled with differences and repeats it based on the number of contours found. On each contour it will be checked whether the contour size is larger than specified. If it is larger, it will repeat based on the number of pixels contained in the contour. Each pixel in the contour is checked to check if it has an area larger than specified using the 5x5 kernel. If the contour is valid then every point on the contour will be stored which will later be checked whether they touch each other or not and if it is considered invalid then it changes all the pixels on the contour to black.

Combining Bounding Boxes

Based on the list of bounding boxes that have been found. Each saved contour will be checked if the contours touch each other. And if they touch each other, it will carry out the process of

merging one bounding box with another bounding box and saving the bounding boxes that have been combined. Storage also applies to bounding boxes that do not touch each other.

RESULT AND DISCUSSION

Result

The following are the results of detecting differences from the two images:

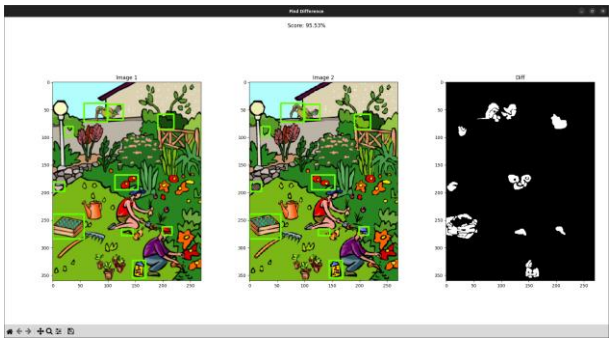


Figure 2. Images of different sizes

Figure 2 The results of detecting two images can run smoothly with each image having a different size, namely 330 x 440 px and 270 x 360 px. It can be seen that the final result of the two images has a size of 270 x360 px because before processing the images, they are rescaled first. But the two images only have 9 out of 10 differences that can be detected with a similarity level of 95.53%.

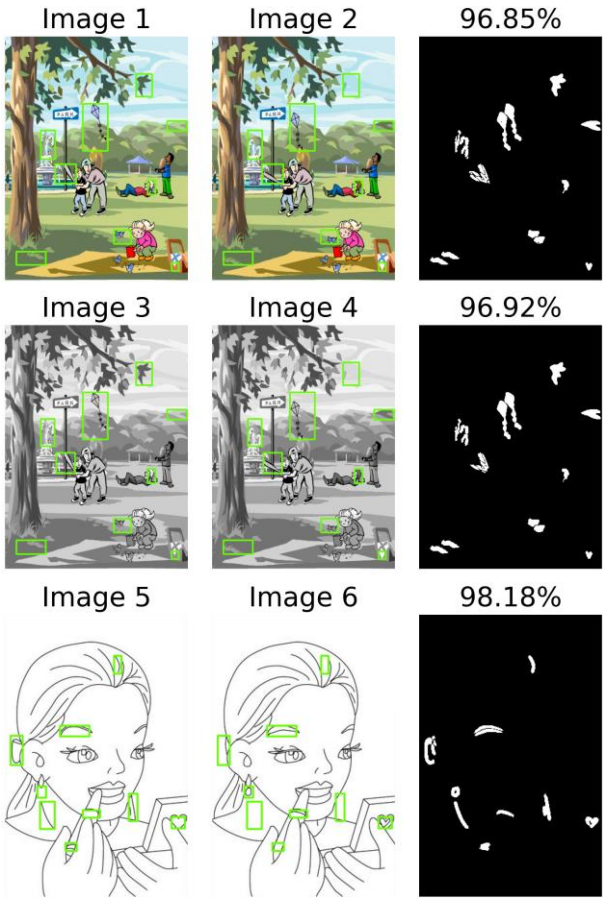


Figure 3. Image in RGB, grayscale and black and white

Figure 3 The result of running a program in which there are color, grayscale and black and white images. The results of the detection of image 1 and image 2 can be carried out in color mode, with a similarity level of 96.85% and 9 out of 10 differences from the two images can be found. The results of the detection of image 3 and image 4 can be carried out in grayscale mode, with a similarity level of 96.92% and 9 out of 10 differences from the two images can be found. The results

of the detection of the two images can also be carried out in black and white mode as in image 5 and image 6, with a similarity level of
98.18% and 9 out of 11 differences from the two images can be found.

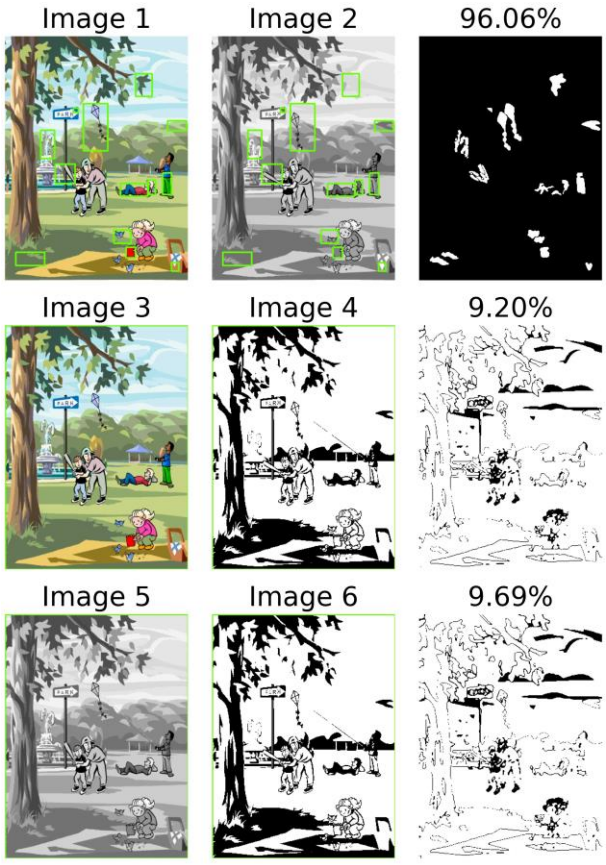


Figure 4. Images with different colors

Figure 4 Not only can it be used in color, grayscale or black and white mode but the results of detecting two images can also be done by combining color, grayscale or black and white mode. First, try using image 1 using color mode and image 2 using grayscale mode with similarity level of 96.06% but has error detection so finds 13 out of 10 differences between the two images of which only 9 are correct. Second, try using image 3 using color mode and image 4 using black and white mode. The program assumes that the images are very different with a similarity level of

9.20% and where a bounding box is attached to the entire surface of the image to indicate that all images are different. Third, try using image 5 using grayscale mode and image 6 using black and white mode. Same as the previous experiment where the program assumes that the images are very different with a similarity level of 9.69% and where a bounding box is attached to the entire surface of the image to indicate that all images are different.

which has many colors, thereby significantly reducing the level of similarity.

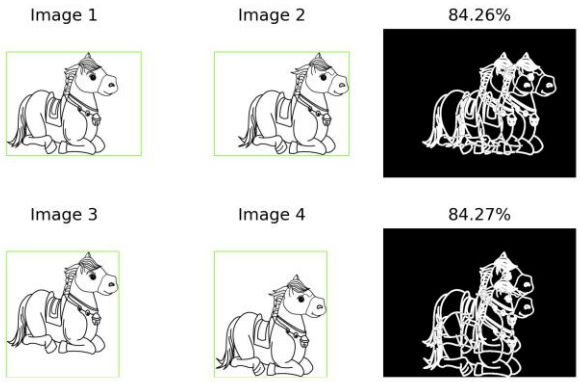
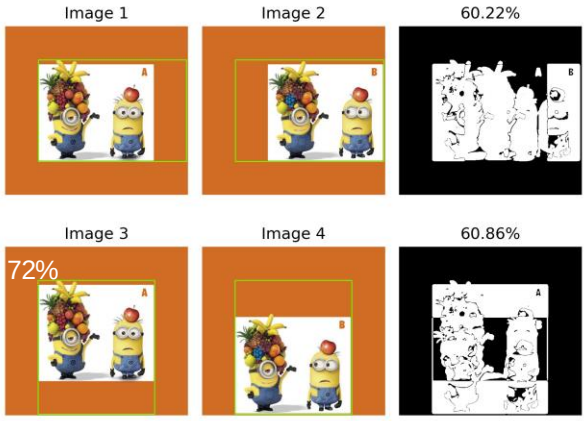
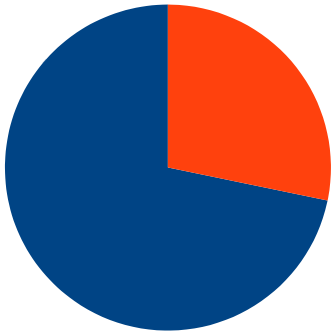


Figure 6. Two misaligned black and white images

Figure 6 Two misaligned black and white images can not be found the difference with certainty. Couldn't find the difference because each pixel doesn't match another pixel. However, the two images still have a high similarity score of 84% because the images only move a few pixels up or to the side and there is no significant change in the pattern of the two black and white images**Figure 5.** Two misaligned color images

Figure 5 Two color images that are not parallel can not be found the difference with certainty. Couldn't find the difference because each pixel doesn't match another pixel. The two images have a relatively low similarity score of 60%. Because the image moves a few pixels up or sideways, it makes significant pattern and color changes in the two color





Success rate

Error rateimages. In contrast to the previous experiment where the two black and white images still have a high degree of similarity because the two images only have 2 colors, namely black and white, it is different from the color image

Figure 7. Experiment Results

Figure 7 is the result of a study using 100 images with a success rate of 72% and an error rate of 28%. Each experiment has a different number of differences, eachdifference is taken from the amount determined by the source. The correct number is calculated manually based on whether the area is in the bounding box and the wrong number is calculated from the number of incorrect and not found bounding boxes. When the images are not parallel, the success rate of accuracy is smaller than the parallel images.

Discussion

The analysis of the results obtained, it shows that the program can find differences in the two images. This program can run in color, grayscale, black and white mode and also at the same or different resolutions. This program has a success rate of 72% and an error rate of 28% from 50 trials using 100 images. This research will be limited if the two images are not parallel because the program checks every pixel, which has a high error rate.

CONCLUSION

Based on the results of the research that has been done, it can be concluded that the program can run well in finding differences in the two images. Based on the problems that have been described, the first is how to calculate the degree of similarity of the two images, namely by multiplying the height and width of the image, then subtracting the number of pixels that are considered different, then multiplying by 100, then dividing by the result of the height and width of the image. The second problem discusses whether it can be used to compare images with different resolutions. The program can run even though the resolution is different because before looking for images the difference is that the two images are checked first whether they have the same or different sizes. And if they are different then the image size will be resized. The third problem discusses whether it can be used to compare images in different color spaces, or only works with grayscale images. The program can run even in color mode, gray scale or black and white, because the program does not determine whether the image must be in color, gray scale or black and white. The program will also convert the image to grayscale and then look for differences through each pixel. The program can also run when the two images are in a different color mode, but cannot be compared to the black and white mode. The final problem statement addresses how accurately the program can find differences. Based on the results that have been tried, the success rate is 72% and the error rate is 28%. Each experiment has a different number of differences, each difference is taken from the amount determined by the source. The correct number is calculated manually based on whether the area is in the bounding box and the wrong number is calculated

from the number of incorrect and not found bounding boxes. When the images are not parallel, the success rate of accuracy is smaller than the parallel images.

After doing this research, this project still has a drawback when the images are not aligned so that the search for differences cannot run smoothly. Suggestions for further research are that the program can run smoothly when the two images are not aligned.

REFERENCES

[1] A. P. Ismail, F. A. A. Aziz, N. M. Kasim, and K. Daud, "Hand gesture recognition on python and opencv," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1045, no. 1, p. 012043, Feb. 2021, doi: 10.1088/1757-899X/1045/1/012043. https://iopscience.iop.org/article/10.1088/1757-899X/1045/1/012043

[2] M. K. Hossen, S. M. Bari, Partho Protim Barman, R. Roy, and Pranajit Kumar Das, "Application of Python-OpenCV to detect contour of shapes and colour of a real image," May 2022, doi: 10.5281/ ZENODO.6576264. https://zenodo.org/record/6576264

[3] W. Supriyatin, "Perbandingan Metode Sobel, Prewitt, Robert dan Canny pada Deteksi Tepi Objek Bergerak," *Ilk. J. Ilm.*, vol. 12, no. 2, pp. 112–120, Aug. 2020, doi: 10.33096/ ilkom.v12i2.541.112-120. http://jurnal.fikom.umi.ac.id/index.php/I LKOM/article/view/541

[4] F. Alexander and I. Imelda, "Analisis Model Pengukuran Tinggi Permukaan Air Dengan Metode Canny Edge Detec- tion dan Image Contouring Sebagai In- dikator Peringatan Dini Bencana Banjir," *FaktorExacta*, vol. 14, no. 3, p. 117, Oct. 2021, doi: 10.30998/faktorexac- ta.v14i3.9567. https://journal.lppmunindra.ac.id/index.p hp/Faktor_Exacta/article/view/9567

[5] M. Z. Andrekha and Y. Huda, "Deteksi Warna Manggis Menggunakan Pengola- han Citra dengan Opencv Python," *Voteteknika*, vol. 9, no. 4, p. 27, Dec. 2021, doi: 10.24036/votetekni- ka.v9i4.114251. http://ejournal.unp.ac.id/index.php/votek nika/article/view/114251

[6] A. Asmaidi, D. S. Putra, M. M. Risky, and F. U. R, "Implementation of Sobel Method Based Edge Detection for Flower Image Segmentation," *SinkrOn*, vol. 3, no. 2, p. 161, Mar. 2019, doi: 10.33395/ sinkron.v3i2.10050. https://jurnal.polgan.ac.id/index.php/sinkron/article/view/10050

[7] K. Diantoro and B. Adriasyah, "SISTEM IDENTIFIKASI JENIS BURUNG DEN- GAN IMAGE CLASSIFICATION MENGGUNAKAN OPENCV," vol. 20, no. 1, 2019. https://journals.upi- yai.ac.id/index.php/TEKINFO/article/view/1159/945

[8] S. Jatmika and D. Purnamasari, "RAN- CANG BANGUN ALAT PENDETEKSI KEMATANGAN BUAH APEL DEN- GAN MENGGUNAKAN METODE IM- AGE PROCESSING BERDASARKAN KOMPOSISI WARNA," vol. 8, no. 1, 2014. https://jurnal.stmikasia.ac.id/index.php/jit ika/article/view/128/99

[9] L. Han, Y. Tian, and Q. Qi, "Research on edge detection algorithm based on im- proved sobel operator," *MATEC Web Conf.*, vol. 309, p. 03031, 2020, doi: 10.1051/matecconf/202030903031. https://www.matec-conferences.org/10.10

51/matecconf/202030903031
[10] R. F. Casamaximo and N. M. L. Romeiro, “Algorithm for extracting points from images: irregular contours,” 2021. https://www.researchgate.net/publication/354930765_Algorithm_for_extracting_p oints_from_images_irregular_contours