Prof. Dr. Moacir Antonelli Ponti

Final report: Remaining puzzle pieces counter

Area: Image segmentation

Igor Martinelli 9006336

12 de Julho de 2020

Contents

1 Proposal **2**

2 Input images **2**

3 Steps to reach the objective **3**

4 Restrictions **3**

5 Repository **3**

6 Metodology **3**

6.1 Segmentation 4

6.2 Split 5

7 Aditional Step **5**

7.1 Piece description 5

8 Conclusions and future steps **6**

9 Appendix **7**

1 Proposal

The task of build a puzzle is very nice and could be hard sometimes, depending on the size of the puzzle. In cases that the puzzle has more than five hundred, a thousand of pieces or even more, it's quiet interesting if we had a tool that could count the remaining pieces with the objective to observe your evolution. Thinking in this, thi project proposes the development of an algorithm with the objective of detecting how many pieces are remaining of a puzzle, based on an image of the remaining pieces, on a surface.

2 Input images

The input images of this project were collected by taking photos of a puzzle. The type of the file is .png and they were taken in a blank surface. Below we can see two examples of the photos.

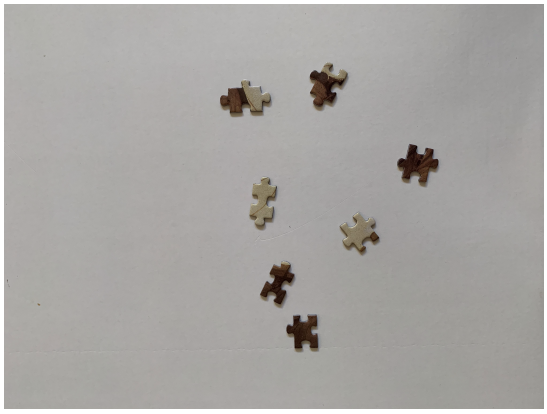

Figure 1: Example 1 of the input image.

Figure 2: Example 2 of the input mage.

3 Steps to reach the objective

For this project, we will define some steps to develop the objective of this tool and, when obtaining them, improvements will be added. The steps are:

- Check if the image is good enough to perform the identification process, that is, check if the blank surface is sufficient for this (segmentation).
- If needed, an API will be used to improve the image background.
- Count how many pieces are in the figure using a greedy solution.
- Separate each piece of the puzzle using masks and try to find the one that best fits in a set of assembled pieces (feature extraction and color descriptors).

4 Restrictions

To follow restrictions needs to be considered to reach the objective well.

- The pieces can't be superposed.
- Good ilumination is needed to the images to preserve it's characteristics.
- The captured images needs to have an 90° angle with the surface to avoid problems with positioning.
- The puzzle pieces must stay in a blank surface or in a surface that preserves good enough it's colors.

5 Repository

As a repository for this project, we'll use GitHub, and the project can be founded in: `https://github.com/martinelligor/DIP/tree/master/Final%20Project`.

6 Metodology

The photos that compound the database were taken in the camera of an iPhone XR, using a fixed distance and trying to balance the natural brightness, trying to avoid the sheen.

6.1 Segmentation

An edge detection-based algorithm named Canny[2] was implemented with the purpose to detect every piece in the picture to count how many pieces are in the image[1]. The tests using Canny edge detector were not so good as we can se in the image 3, that represents the figure 1, showed above, after aplication of the algorithm.

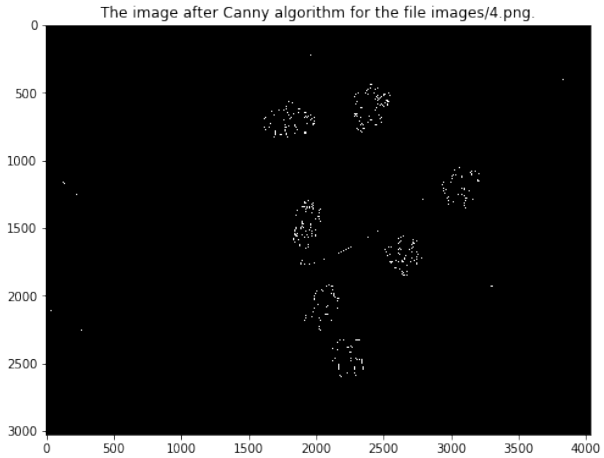

Figure 3: Canny edge detection.

Therefore, it was decided to use a threshold-based method to perform the image segmentation, but the results are not satisfied too. In this way, one more restriction were added: white background to the images. For this, we used an API [4] to process the images and applied again the threshold method, obtaining a sucessful result 4.

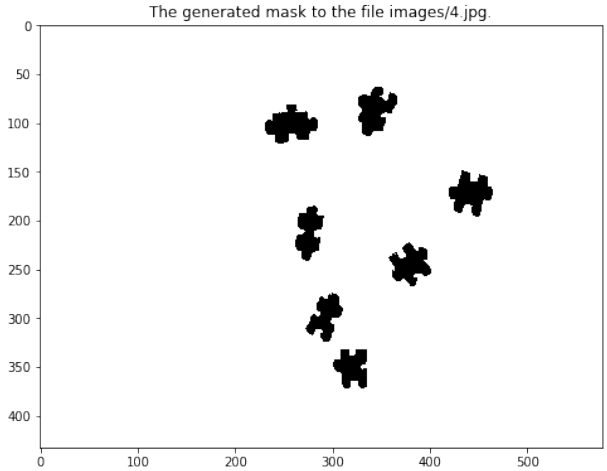

Figure 4: Obtained mask after threshold segmentation after using the API.

6.2 Split

After recognizing the pieces, we need to split the images in order to obtain the number of pieces in the image. For this, was implemented a greedy algorithm named flood-fill [3].

Then, the algorithm was applied into the mask obtained from the image and this step resulted in the image 5.

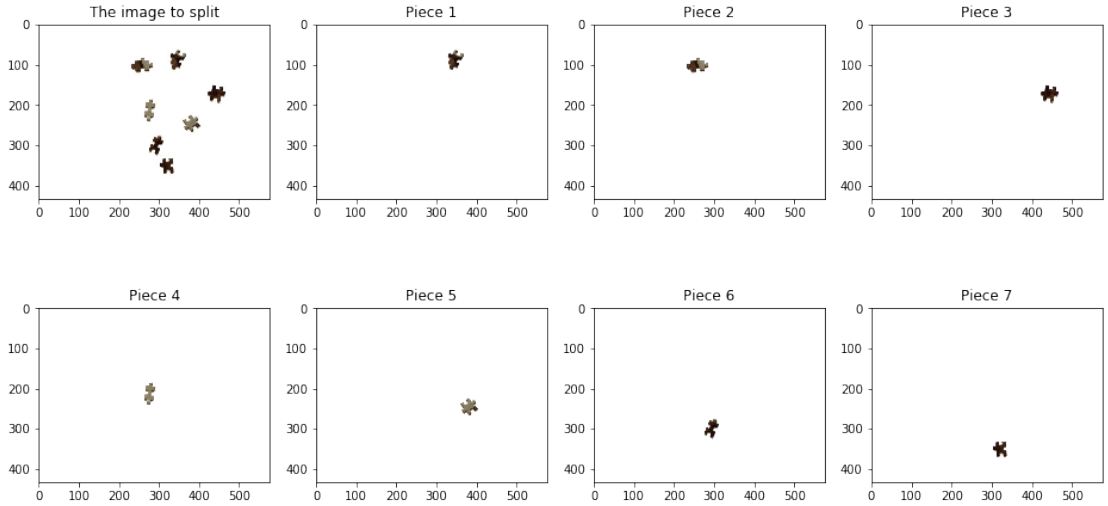

Figure 5: An example of split step.

7 Aditional Step

After recognizing the pieces, and aditional step was proposed: suggest the best piece to be fitted some region of the puzzle. Considering the time, it was possible just to find the best piece that has the best similarity with a given piece.

7.1 Piece description

To realize the step proposed above, was implemented three descriptors to find the pieces characteristics: color, texture and gradient. For this task, the scikit-image library was used to implement the Haralick descriptors.

After extract the descriptors, an distance function was computed in order to fit the best piece in comparison to another and some variations of color, texture and gradient weights were tested on the algorithm to get a better result.

The possible implementation of this step, given the time to complete the development, was to find the piece located in test case 1, in the other test cases,

that has the same piece in them. The ideia was to use the mechanisms used to complete this task to find an optimal piece to fill a part of the puzzle. The results of this step are showed below in the images 6 and 7.

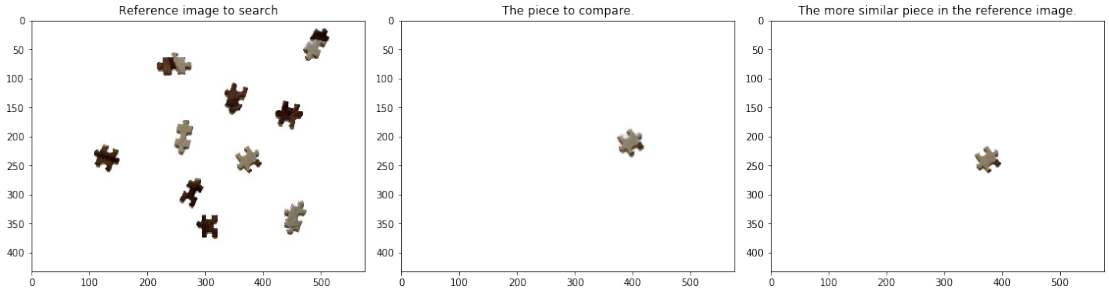

Figure 6: Result of the piece sugestion for the test case 5.

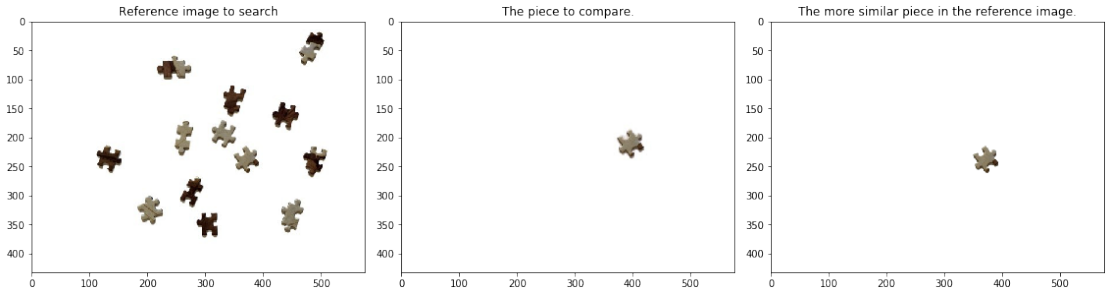

Figure 7: Result of the piece sugestion for the test case 7.

8 Conclusions and future steps

Future steps of this work can be done with the objective of achieve better results to count pieces without the necessity to use an API to preprocess the background or maybe develop an neural network that can do this job. After that, the technique to detect where a piece can be plugged need relative efforts to improve the idea proposed in 7. All the algorithms and the knowledge to make this tasks possible was obtained during the course and from the discipline book [5].

9 Appendix

This section is only a bonus to show the puzzle that was used to obtain the images. During the discipline the puzzle was completed and that was the moment when the idea to use this to make the final project come in the mind 8.

Figure 8: The puzzle used for the development.

References

[1] Canny edge detector on medium. `https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123`.

[2] Canny edge detector on wikipedia. `https://en.wikipedia.org/wiki/Canny_edge_detector`.

[3] Flood fill. `https://en.wikipedia.org/wiki/Flood_fill`.

[4] Remove bg api. `https://www.remove.bg`.

[5] R. Gonzalez and R. Woods. *Processamento Digital De Imagens*. ADDISON WESLEY BRA.