State University of Campinas
Institute of Computer Science
MO443 - Professor: Hélio Pedrini

Project 3

May 27, 2021

Pedro Henrique Vaz Valois
p265676@dac.unicamp.br

***Abstract*—Digital images are abundant in the current age. Be it as means of modern art or source of data, they have a high degree of importance in modern society. Thus, the need for processing, transforming and understanding such images has risen and a multitude of techniques were developed for these tasks. In this project, we analyze full color figures of blobs in a white background using image segmentation techniques.**

I. Background

A monochromatic image is defined as a $M \times N$ matrix

$$I = (I_{mn}), 0 \leq m \leq M-1, 0 \leq n \leq N-1$$

where M is the image pixel width, N is the image pixel height and I_{mn} indicates the intensity in the $[0, 255]$ range for the pixel in position (m, n).

In this sense, an image can be clustered in different semantic groups, in which each blob is a contiguous group of pixels represented with a label $0 \geq l \geq L$, where L is the number of blobs and $l = 0$ represents the background. Therefore, we can find a labeling function $f(l) = \{m, n\}$ such that, for each l, the function f returns all pixels (m, n) that belong to the label l.

A. labeling (or clustering)

In this project, we use an open source implementation of Block Based with Decision Trees for labeling blobs in a white background image [1]. This algorithm uses a decision tree formed by 2 x 2 blocks of a binary input image to label different regions within it.

B. blob analysis

The items below summarize some analysis and measurements that can be done on a blob.

1) area: The area of a blob is simply the total number of pixels with it, considering the edge too. Therefore, we can measure the area of a label l as

$$A(l) = \sum_{m}^{M-1} \sum_{n}^{N-1} f(l)_{mn}$$

2) edge detection: A simple mechanism to find edges within an image is by using Robert's gradients [2, p. 2], that are two first order gradient operators. These operators can be represented as the order 2 matrices in equations 1 and 2. These matrices represent the partial derivatives of the gradient and the edge of an image can be retrieved from then following equation 3.

$$G_x = \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1}$$

$$G_y = \begin{pmatrix} 0 & +1 \\ -1 & 0 \end{pmatrix} \tag{2}$$

$$E = |I * G_x| + |I * G_y| \tag{3}$$

3) perimeter: The perimeter of a blob labeled l is here defined as the sum of all pixels in the blob's edge. We can get the blob's edge from the intersection between the edge and the labeling $e(l) = E \cap f(l)$. This gives us the perimeter

$$p(l) = \sum e(l)_{mn}$$

.

It is important to notice that there are other ways to define the perimeter and the usage of Robert's gradients generates neighborhood-4 edges, making our perimeters larger than if other approaches were taken, such as binary erosion for edge detection [3].

4) centroid: The centroid determines the position where the center of a blob is located. It can be represented by (c_x, c_y) and is simply the average of the positions of the pixels in the blob (or on its edge) [4, p. 3].

$$c_x(l) = \frac{1}{F(l)} \sum m(f(l)_{mn}) \tag{4}$$

$$c_y(l) = \frac{1}{F(l)} \sum n(f(l)_{mn}) \tag{5}$$

where $F(l)$ is the number of pixels in the blob $f(l)$.

5) eccentricity: Eccentricity measures the aspect ratio of a shape, defined as the ratio between the major axis length and the minor axis length [4, p. 4].

The principal axes method finds the two segments that cross each other orthogonally in the centroid (c_x, c_y). For that, we must diagonalize the covariance matrix $C = \begin{pmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{pmatrix}$ where

$$c_{xx} = \frac{1}{F}\sum_m (x_m - c_x)^2$$

$$c_{yy} = \frac{1}{F}\sum_n (y_n - c_y)^2$$

$$c_{xy} = c_{yx} = \frac{1}{F}\sum_k (x_k - c_x)(y_k - c_y)$$

Therefore, the eigenvalues of C will be

$$\lambda_1 = \frac{1}{2}\left(c_{xx} + c_{yy} + \sqrt{(c_{xx} + c_{yy})^2 - 4(c_{xx}c_{yy} - c_{xy}^2)}\right)$$

$$\lambda_2 = \frac{1}{2}\left(c_{xx} + c_{yy} - \sqrt{(c_{xx} + c_{yy})^2 - 4(c_{xx}c_{yy} - c_{xy}^2)}\right)$$

Finally, the eccentricity can be measured as $E = \frac{\lambda_2}{\lambda_1}$.

6) solidity: Solidity measures how much a shape is convex or concave. It is defined as

$$S = \frac{A}{H}$$

, where A is the total blob area and H is the area of its convex hull, i.e., the area of the smallest convex region including the blob. In this project we use an open source implementation of a convex hull finder to measure the blob solidity.

II. Implementation

All code for this project has been developed and tested with `Python 3.7.9`, `numpy 1.20.2`, `scipy 1.6.3`, `matplotlib 3.4.1` and `scikit-image 0.18.1`

The software that implements the methods discussed in the last section is contained in the `script.py` file. It depends on `numpy` to perform the vectorized operations, `scipy` for convolution and labeling, `matplotlib` for reading and saving image files and `scikit-image` for convex hull extraction. Moreover, it uses python standard library's `os` module for checking if the output file already exists and the `argparse` module for parsing command line arguments.

Images and kernels are represented in the code as 2D `numpy` arrays in order to gain performance and also to simplify most math operations.

The `argparse` module provides a help text for using the script. By typing `python script.py --help`, a helper message is shown in the console.

The code was designed to deal RGB files, and no tests were performed with monochromatic versions of the images.

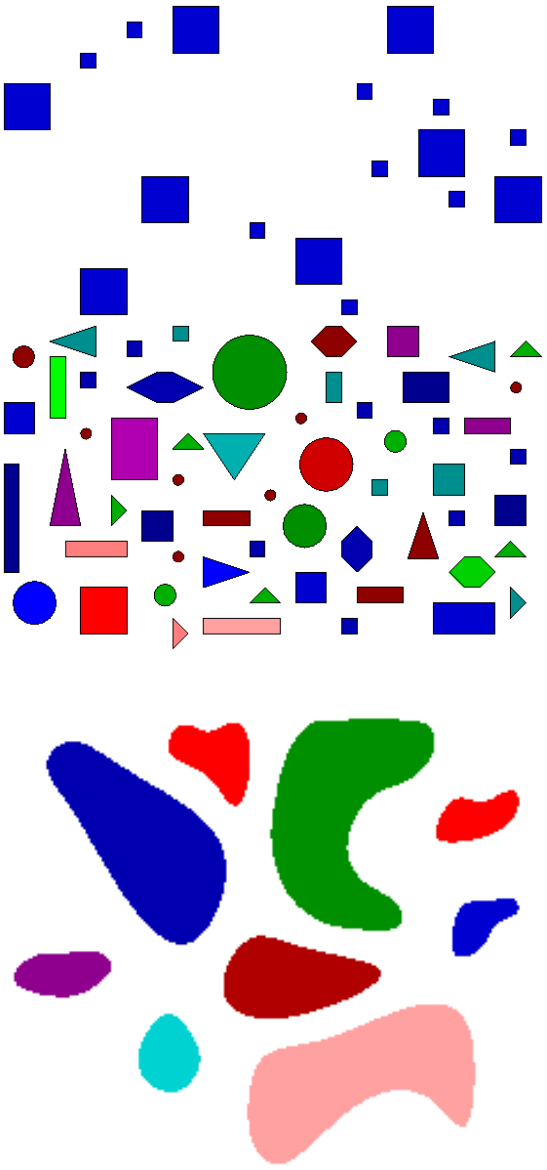

Figure 1: Original images tested

III. Results and Discussion

All tests were conducted on 3 PNG 512×512 images of geometric shapes in a white background. The original images are shown in figure 1.

A. Binary image

As a first step, we make all images binary, pushing all non-white colors to black. The results are shown in figure 2. This step is important, as most algorithms applied in this project require the images to be binary, such as convex hull extraction and labeling.

B. Edge detection

Second, we apply a 2D convolution with Robert's gradients to find neighborhood-4 edges. It is visible that the

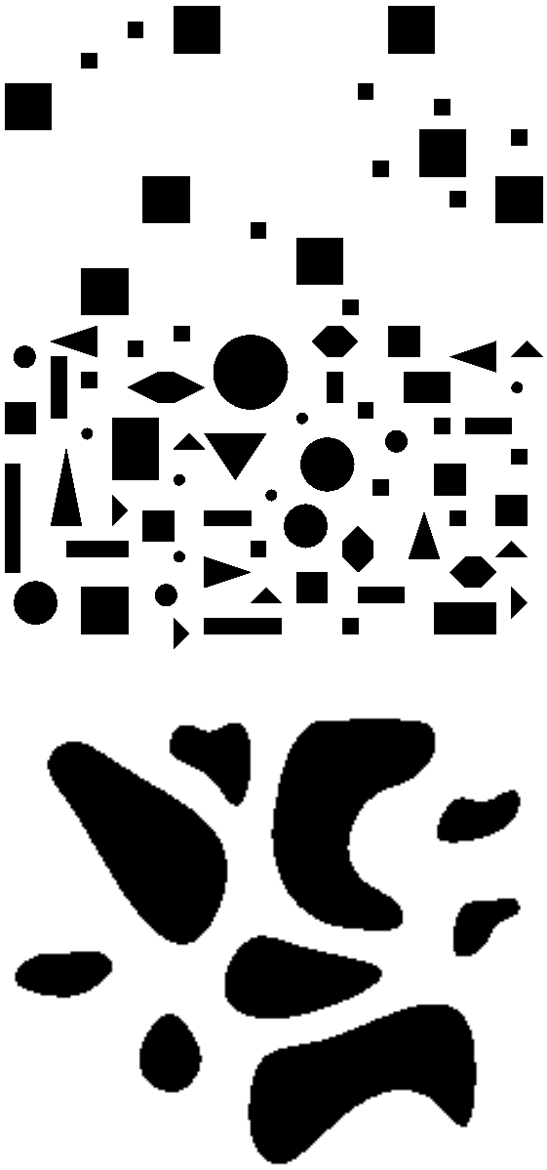

Figure 2: Images converted to binary color values

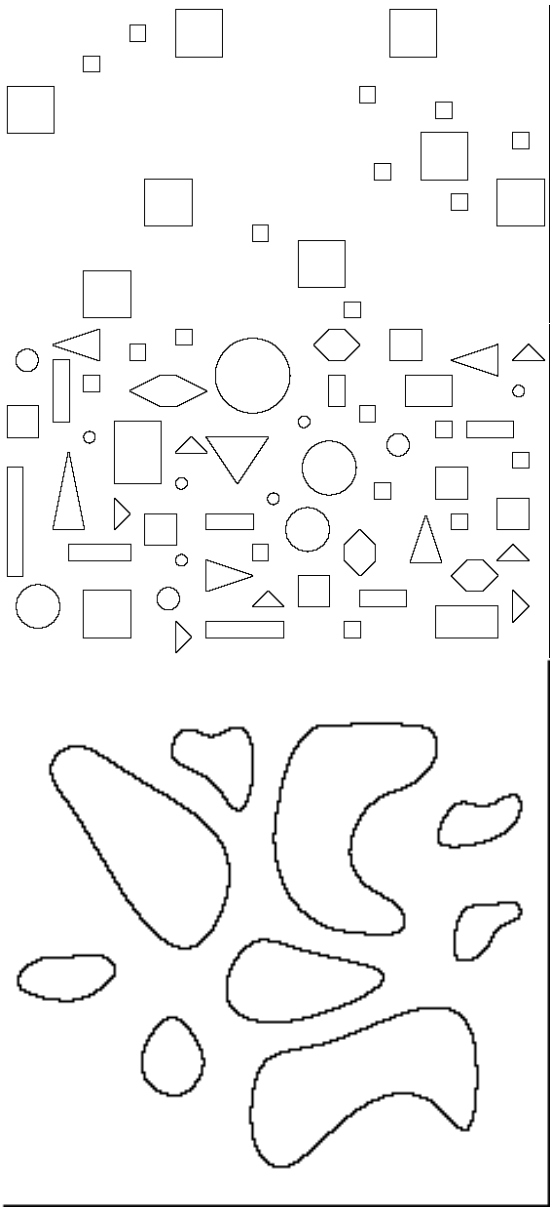

Figure 3: Edges extracted from each blobs using Robert's gradients

neighborhood-4 edges are not well suited when diagonal segments are present, such as in images 2 and 3 at figure 3, adding more pixels than needed on the edges. In spite of that, the overall result is satisfactory given its simple nature.

Moreover, the output depicts the image border as an edge, what may or may not be desired. In any case, the image border will not be considered on the next steps.

C. Labeling

Next, we apply the labeling algorithm available with `scipy.ndimage.label`. This uses a fast `cython` implementation of Block Based with Decision Trees that clusters the black blobs from the binary image produced a few steps back. In our implementation, this step returns a labeled version of the image, where a red number representing its label is placed on the centroid of each shape as shown in Figure 4.

D. Shapes properties

We implement a special option `--data` that prints the number of regions found and produce a table of shapes properties: area, perimeter, centroid, eccentricity and solidity. An example for the third image is shown in table I, that has 9 distinct regions.

E. Histograms of Areas

We implement a special option `--hist` that plots the histogram of the areas for each region found separating it in 3 distinct bins: small, if the area has less than 1500 pixel; large,

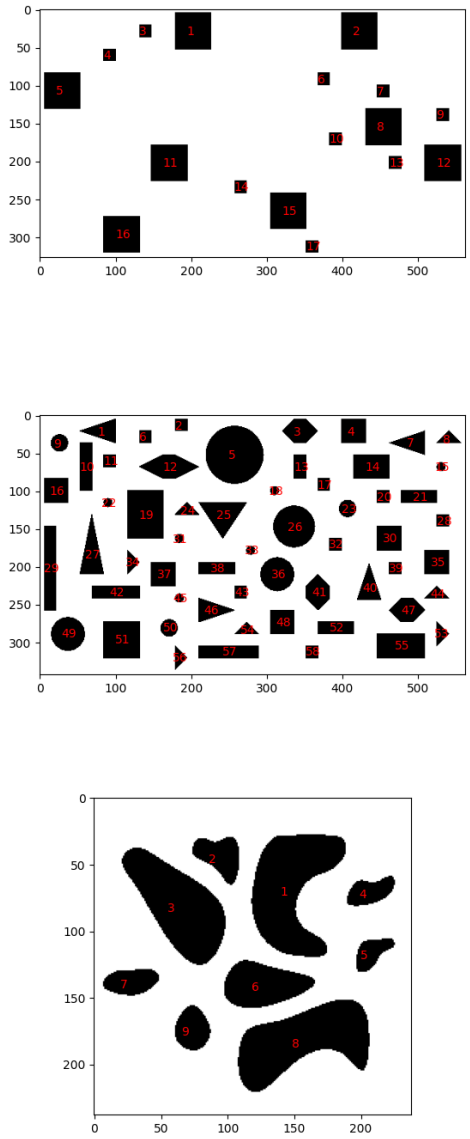

Figure 4: Labeling of blobs with `scipy.ndimage.label`

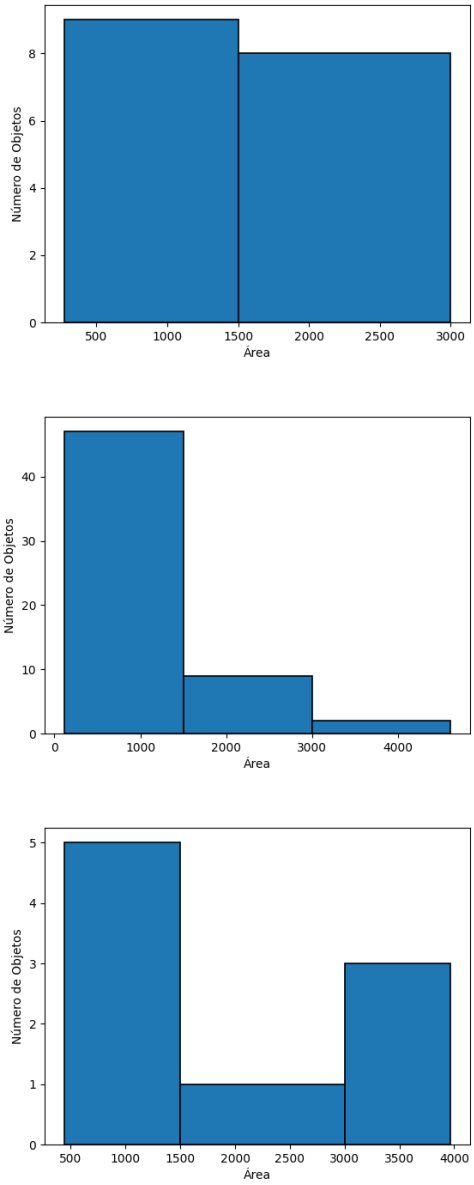

Figure 5: Histogram of the distribution of the blobs areas

Table I: Properties of each shape in the third image tested (9 blobs). The regions number match the numbers from the labeling at Figure 4

region	area	perimeter	centroid	eccentricity	solidity
1	3969	3783	[67, 147]	0.32	0.75
2	791	718	[43, 93]	0.43	0.90
3	3584	3419	[80, 62]	0.19	0.98
4	540	480	[70, 206]	0.19	0.90
5	438	385	[115, 207]	0.26	0.91
6	1684	1581	[139, 125]	0.24	0.97
7	642	581	[137, 27]	0.20	0.96
8	3934	3751	[182, 156]	0.16	0.77
9	675	615	[172, 73]	0.60	0.97

if the area has more than 3000 pixels, and medium if it lies in between. This command also prints the numbers in the scree. The plots can be seen in Figure 5 and an example of how to execute it is displayed on listing 1.

```
>>> python script.py objetos3.png out.png -w --hist
number of small regions: 5
number of medium regions: 1
number of large regions: 3
```

Listing 1: Output from –hist option showing the number of shapes grouped by area

IV. Conclusion

In this project, labeling and edge detection algorithms were applied as means of performing image segmentation,

clustering and shape analysis. These tools have a wide range of applications with some even being robust against noise.

REFERENCES

[1] AImageLab, "Connected components labeling." [Online]. Available: "https://aimagelab.ing.unimore.it/imagelab/researchActivity.asp?idActivity=15"

[2] M. A. Ansari, D. Kurchaniya, and M. Dixit, "A comprehensive analysis of image edge detection techniques," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 12, no. 11, pp. 1–12, 2017.

[3] K. Benkrid, D. Crookes, and A. Benkrid, "Design and fpga implementation of a perimeter estimator," in *Proceedings of the Irish Machine Vision and Image Processing Conference*, 2000, pp. 51–57.

[4] Y. Mingqiang, K. Kidiyo, and R. Joseph, "A survey of shape feature extraction techniques," *Pattern recognition*, vol. 15, no. 7, pp. 43–90, 2008.