Algorithm for extracting points from images: irregular contours

Rafael Furlanetto Casamaximo[1], Neyva Maria Lopes Romeiro[2], Pedro Zaffalon da Silva[1], Iury Pereira de Souza[1]
Júllia Thainna Alves da Silva[1], Paulo Laerte Natti[2], Eliandro Rodrigues Cirilo[2]

[1]*Dept. of Computer Science, State University of Londrina*
Rodovia Celso Garcia Cid - PR 445 Km 380, 86.057-970, Paraná/Londrina, Brasil
rafael.furlanetto@uel.br, pedro.zaffalon@uel.br, iury.pereira.souza@uel.br, jullia.thainna@uel.br
[2]*Dept. of Math, State University of Londrina*
Rodovia Celso Garcia Cid - PR 445 Km 380, 86.057-970, Paraná/Londrina, Brasil
nromeiro@uel.br, plnatti@uel.br, ercirilo@uel.br

Abstract. This work proposes the development of a software to implement a contour extraction algorithm for two-dimensional geometries. The algorithm uses different image processing and data analysis techniques to build an object mask. From the object mask, a finite set of points that describe the image contour is extracted. Next, the contour is separated by parts with different enumerations delimited by rectangles. All data processed and extracted from the image can be downloaded by the user in file form. Then, the information from the contour data is used to generate two-dimensional meshes for finite difference calculations and numerical simulations. Our automated algorithm generates meshes for irregularly contoured geometries. For this, the mesh elements in rectangular coordinates are not equally spaced, so that the vertices of the border coincide with the contour of the geometry obtained by the algorithm. For the development of the algorithm, techniques are used that select a pixel range of an object to generate the mask. The software uses the Python programming language, together with the OpenCV library that performs image and data analysis.

Keywords: Image processing, mesh, finite differences

1 Introduction

Digital image processing is a set of techniques focused on the analysis of multidimensional data acquired through different types of sensors, whose objective is to manipulate image data and its essential information. Such techniques enable the integration of various types of data, for example, as in the analysis of biomedical images through ultrasound, nuclear radiation or computed tomography techniques Theodosios [1]. In this context, many techniques of digital image processing use simple data organization structures, such as matrices and vectors to represent geometries, that is, the domain to be studied in a computational environment to represent the object of study.

Mathematically any geometry can be described in the Cartesian system. However, for the mathematical analysis it is often necessary to provide data information on the irregular contour of the mesh, so that for a realistic representation of the problem a great level of refinement is necessary. Tools in the literature use image processing and other techniques to partially solve this problem, as can be seen in Weeks et al. [2], which uses the HSL (Hue, Saturation, Lithness) color space to identify the contour of an image, preserving only the important details. The HSL technique suffers from the amount of noise that is identified as contour. In Sharma et al. [3], the Moore Neighborhood algorithm is used to identify the contour of an image. However, the proposed solution does not allow the creation of any kind of mask to isolate certain aspects of the image as objects, colors, contrasts, .. In addition, the HSL color space and Moore Neighborhood algorithm tools do not allow the points that form the object's contour to be exported for external use. On the other hand, the software WebPlotDigitizer 4.3 Rohatgi [4] allows the user to insert an image and generate all the contour points manually. It also allows the user to enter only points that are convenient, but these manual procedures involve high time costs. Thus, the development of a software capable of performing the procedure for extracting points from the contour of an image, enabling the

parameterization and export of the geometry's contour points, it is of great importance for methods that use this data.

In this context, the work proposes the development of a software, in Python programming language Rossum and Guido [5], to extract a finite set of points from an irregular region that form the contour of objects contained in images. Such domain points use HSV color space for parameterization and mask generation, and the Moore Neighborhood algorithm Weisstein and W [6] for contour extraction. From this information, the finite difference method can be applied to the solution of partial differential equations.

2 Development

For the software development we used the Python programming language Rossum and Guido [5], together with the OpenCV library Bradski et al. [7], which allows manipulations for the image, data and treatment processing functions. The Click library Rossum and Guido [5] was used for the management of the command line tools, while the Tkinter framework Rossum and Guido [5] was used for the development of the graphical user interface (GUI).

The developed software has three different tabs at the top: Mask, Contour and Sections. The user has access to the software's tools and functionalities through the tabs. The graphical interface can be seen in Figure 1.

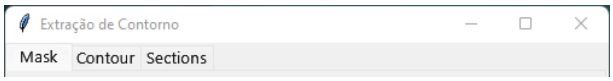

Figure 1. Window and Tabs of the Software

2.1 Mask

The Mask tab aims to allow the user to insert an image, choose the necessary parameters to improve the fit, and extract a mask from the image. Masking is a conversion of the pixels that make up the original image into two colors, black and white, through user-defined conditions that vary for each image, according to the HSV parameters.

Initially, when the user opens the Mask tab, he gets a window with no image, Figure 2a. When selecting the image, the user can insert it through the Insert button. Thus, the program receives the image in one of the allowed formats as input. See Figure 2b. Next, the image is transformed into an array of pixels. Each element of the matrix contains information about the color of the pixel and the different red, green and blue channels. For the conversion of a specific format, such as JPEG or PNG or others, to a pixel matrix, the OpenCV library Bradski et al. [7] is used, which allows this and other manipulations on the data files.

a) b)

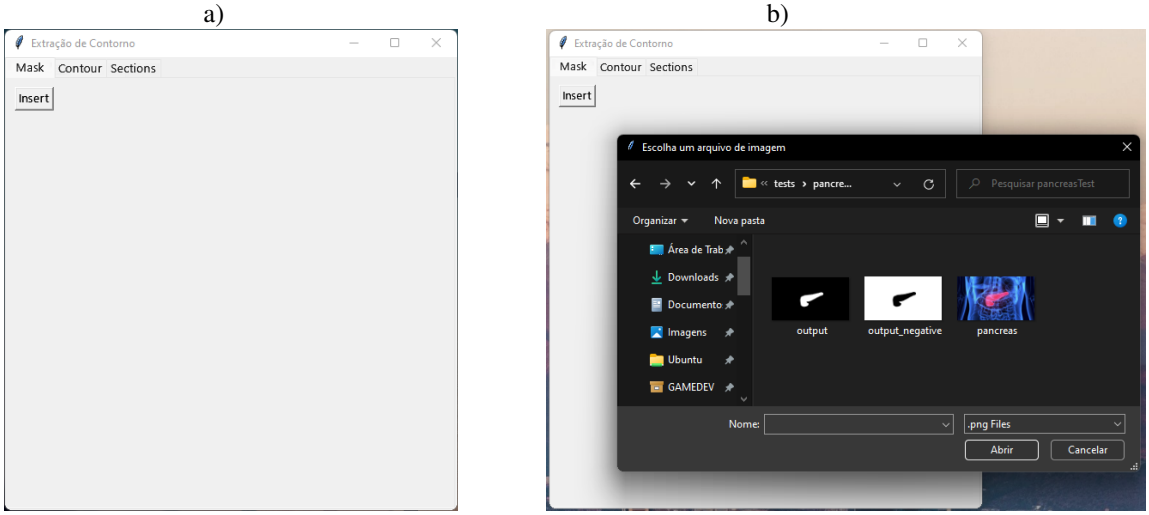

Figure 2. Software mask tab: a) with no image inserted, b) with image insertion.

When the image is inserted in the Mask tab, new visual elements and data appear and are processed. The program uses the matrix that describes the image to create a new matrix, whose values for each pixel are separated in HSV format. In this way, the same image is generated, but in a different format, still available for the user

to view. Along with the image appear another seven sliders, in which the user can determine the minimum and maximum range of hue, saturation and value.

The two sliders that control hue range between 0 and 180. The four sliders that control saturation and value range between 0 and 255. An additional slider has been inserted to control opacity and view the mask and the original image at the same time. The opacity slider ranges between 0 and 100, representing the percentage of the mask's opacity.

To demonstrate the parameterization and creation of a mask, a figure with a pancreas Fernandes [8] is used, highlighted in Figure 3. Details of the process of parameterization and construction of the pancreas mask can be seen in Figure 4.

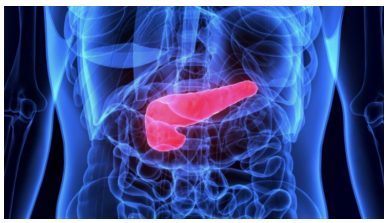

Figure 3. 3D representation of the human body with the pancreas highlighted.

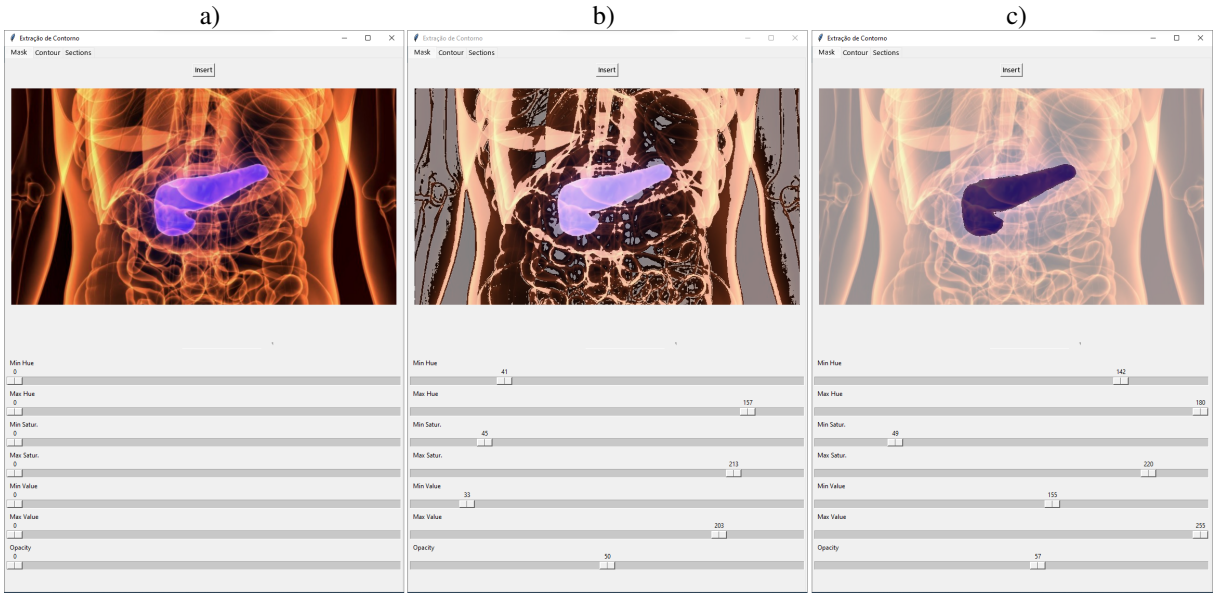

Figure 4. Parameterization process of the HSV values for the construction of the pancreas mask.

In Figure 4a there is the image immediately after being inserted, only with a color change to facilitate visualization. Note that the graphical interface components are available. In Figure 4b it is verified that the parameters are altered, generating a mask of the figure, represented by the dark parts. The mask does not yet represent the desired object, the pancreas, requiring parameter adjustments.In Figure 4c it can be seen that the parameters are more adjusted to the object (pancreas).

For a better understanding of the process used for the parameterization and construction of the mask, from the HSV matrix and the minimum and maximum parameters acquired in the "Mask" tab, the mask generation algorithm is presented, Algorithm 1. The hue, saturation, and value variables are represented by H, S and V, respectively. The quantities H_{ij}, S_{ij} and V_{ij} represent the HSV components of the pixel p_{ij}, with i describing the position of the row and j of the column in the HSV matrix. The condition for a pixel to belong to the mask is given by:

- Hue: $H_{ij} \in]H_{min}, H_{max}]$;
- Saturation: $S_{ij} \in]S_{min}, S_{max}]$;
- Value: $V_{ij} \in]V_{min}, V_{max}]$.

Algorithm 1: Mask Generation

Input : Matrix with each element having three components: HSV; HSV minimum and maximum ranges.
Output : Matrix that represents the desired mask, generated from the initial matrix with variation of parameters.

begin
  Create a new matrix with the same dimensions as the original HSV matrix, called matrixMascara;
  Create variables i and j and assign 0 to them;
  while *i < HSV matrix width* **do**
    while *j < HSV matrix height* **do**
      The pixel of the HSV matrix of the current iteration is represented by p_{ij};
      H_{ij} is the tone component of p_{ij};
      S_{ij} is the saturation component of p_{ij};
      V_{ij} is the value component of p_{ij};
      if $H_{min} < H_{ij} \leq H_{max}$ *e* $S_{min} < S_{ij} \leq S_{max}$ *e* $V_{min} < V_{ij} \leq V_{max}$ **then**
        Sets the pixel p_{ij} to black, at the same position in the Mascaramatrix;
      else
        Sets the pixel p_{ij} to white, at the same position in theMascaramatrix;
      end
      j = j + 1;
    end
    i = i + 1;
  end
end

2.2 Contour

The Contour tab is responsible for generating the object's contour based on the mask constructed on the Mask tab. Initially, when opening the Contour tab, there is the Generate Contour button, as can be seen in Figure 5a. When you click on the Generate Contour button, new visual elements and data appear on the tab, such as the desired and adjusted object, the black part of the figure, and the white part representing the object 5b.

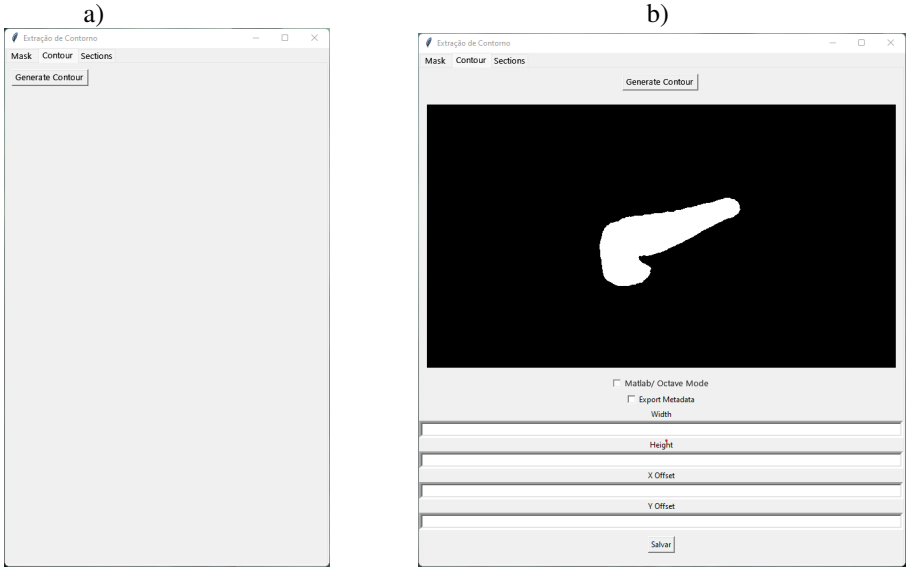

Figure 5. Contour generation and data export options

In the interface is inserted the image of the mask generated in the last tab, Figure 5b. This image now doesn't have any kind of transparency or opacity control and is color reversed. The white area represents what is contained in the mask and the black area what is not in the mask.

Other configuration fields were generated, namely:

- **Width**: Control of the width of the generated pixel set;
- **Height**: Control of the height of the generated pixel set;
- **X Offset**: Displacement of the X axis from the origin point;

- **Y Offset**: Displacement of the Y axis from the origin point;
- **Octave Mode**: Export points in desired format to Octave software Eaton et al. [9], with the origin in the lower left position;
- **Export Metadata**: Exports extra data related to the contour, such as the time of the test performed, the change in width or height, the displacement of the axes, the total area of the figure after transformations, the image directory, among other information.

The mask contour can be extracted using the Moore Neighborhood algorithm Weisstein and W [6]. This algorithm builds a binary matrix, from black and white binary colors, using the color of the dots (pixels) to differentiate between contour and image filling. The structure of the algorithm can be seen in Algorithm 2.

Algorithm 2: Moore Neighborhood

Input : Matrix **T**, containing a connected grid P of white pixels;
Output : list **B** $(b1, b2, ..., b_k)$ of pixels that form the outline;

```
begin
    Set M(a) to be the Moore Neighborhood of pixel a;
    Set p as the current center pixel;
    Set c as the current pixel under observation in M(p), that is, the pixel being checked;
    Set b as the backtrack of c (neighbor of the p pixel that was previously tested);
    Set B to empty;
    Scan the cells of the matrix T, bottom to top and left to right, until a white pixel s, from p, is found;
    Insert s into list B;
    Set the current center pixel p to s (p = s);
    Set b to the pixel where s was inserted during the image scan;
    Set c as the next clockwise pixel from b in M(p);
    while c is different from s (initial) do
        if c is a white pixel then
            Insert c in list B;
            Set b to p;
            Set p to c (move the current pixel c to the pixel in p);
            Set c as the next clockwise pixel from b in M(p);
        else
            Set b to c (updates backtrack);

            Set c as the next clockwise pixel from b in M(p);
    end
end
```

2.3 Sections

For the initial generation of the mesh, from the developed software, all figures present the maximum level of refinement. While this high level of refinement is adequate for greater precision, it impacts code execution speed and performance, as well as the application of the finite difference method for solving partial differential equations in the resulting mesh.

With that in mind, the Sections tab allows the user to choose the level of refinement of the mesh, whether the elements are evenly spaced or not, and the interval between each exported point. Thus, it allows you to adapt each image to your needs, balancing performance and precision. It is also possible to perform a quality analysis of the generated elements. This tab has not yet been fully implemented and is under development.

3 Results

To view the results, three tests are performed, denoted by BasicTest3, FeetTest and PancreasTest, which have different parameters and sizes. All tests have descriptions in the images.

In Table 1 there is information about the areas of the test images, the intervals of the HSV values, the number of nodes in the contour and the internal region of the images. This information is useful in finite difference calculations. Results are presented using maximum refinement.

Table 1. Parameters and information of the three tests performed

		Range of values			Number of points	
Tests	Area (pixels)	Hue (H)	Saturation (S)	Value (V)	Contour	Interior
BasicTest3	285005	-	-	-	9057	289534
FeetTest	83970	$]4, 180]$	$]0, 224]$	$]0, 255]$	1629	84785
PancreasTest	13514	$]142, 180]$	$]142, 180]$	$]155, 255]$	568	13799

3.1 Test with binary color figure - BasicTest3 image

The BasicTest3 test has a binary color scheme, as seen in Figure 6a, so there is no need for a treatment to create the masks through the HSV values. The test aims to verify the operation of the Moore Neighborhood algorithm, developed for contour extraction. The test image was chosen due to its high resolution and the large number of points to form the contour, in order to assess the impact on the algorithm's execution time. It is observed the data plot in Figure 6b and a delimited and enlarged area of these points in Figure 6c.

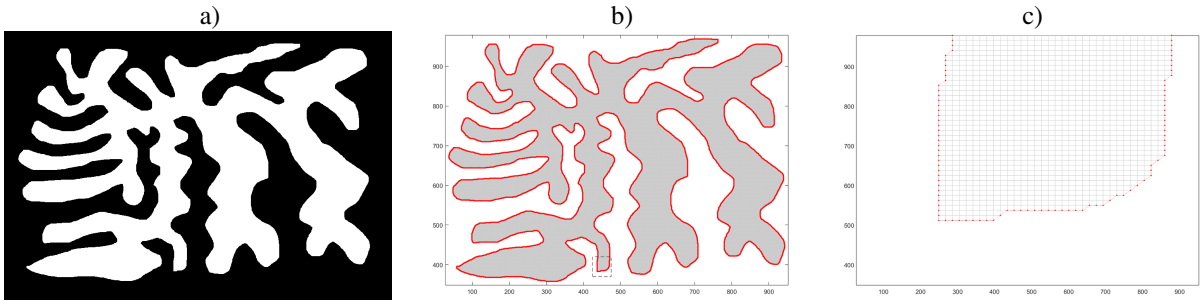

Figure 6. Binary color figure: a) original image (mask), b) Plot of data, c) Zoom of points.

Note, Figure 6b, that the software satisfactorily described the image, as well as its details, Figure 6c. It is verified that all the contour points are located on the mesh nodes.

3.2 Test with no binary color figure - FeetTest and PancreasTest images

Next, the functioning of the 2 Algorithm, developed for contour extraction, is verified. Tests are presented in which the images go through the mask creation procedure, through the parameterization of the minimum and maximum, using the HSV values.

The FeetTest test, visualized in Figure 7a, has an image with little hue variation. The PancreasTest test, visualized in Figure 7d, has a greater range of blue and red hues, in addition to a variation in saturation and value, due to the gradients present at the contour. Such images were obtained from Fernandes [8] and Fernandes [8], respectively. Figures 7b and 7e represent the masks generated from the tests. The figures 7c and 7f) represent the data plot by Octave.

Again, it is observed in Figures 7c) and 7f) that the software satisfactorily described the images, as well as their details. It is verified that all the contour points are on the mesh nodes.

4 Conclusions

In this work we developed a software for extracting points from the contour of an object, which is contained in a figure. Such data are essential in numerical simulations.

It can be seen through the results obtained that the developed software, using the HSV mask and the Moore Neighborhood algorithm, successfully extracts the necessary and essential information from the test cases. Note that the tests have different aspects in hue, contrast, value, texture and noise. Finally, it is important to emphasize that several features of the software are still under development, in particular, the quality tests for the generated contours and meshes.

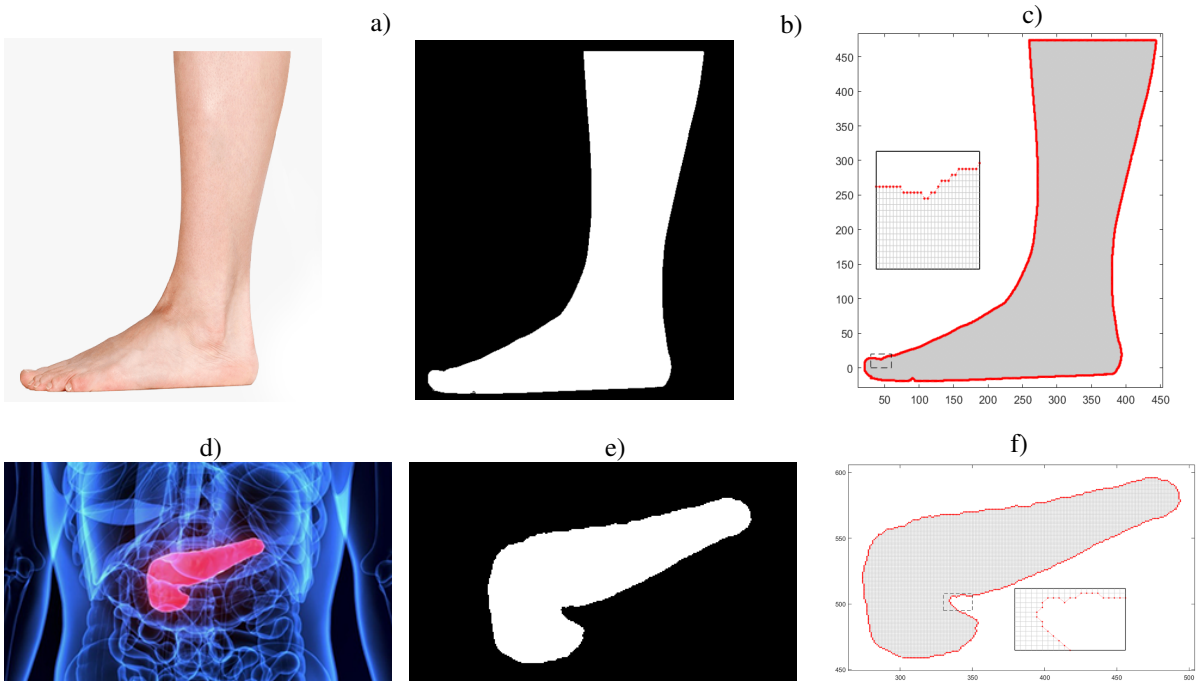

Figure 7. Colored figures: a) e d) original images; b) e e) generated masks from FeetTest and PancreasTest, respectively; c) e f) Plot of data from FeetTest and PancreasTest, respectively, with zoom of points.

Acknowledgements.

The work of Silva, P. Z was partially supported by CNPq under the process 152547/2019-3, and the project of I. P. de Souza was partially supported by CNPq under the process 11600.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work.

References

[1] P. Theodosios. *Algorithms for graphics and image processing*. Springer-Verlag, Rockville, 2012.

[2] A. R. Weeks, C. E. Felix, and H. R. Myler. Edge detection of color images using the hsl color space. *Symposium on Electronic Imaging: Science & Technology*, vol. 2424, n. 1, pp. 291–301, 1995.

[3] P. Sharma, M. Diwakar, and N. Lal. Edge detection using moore neighborhood. *International Journal of Computer Applications*, vol. 61, n. 3, pp. 26–30, 2013.

[4] A. Rohatgi. Webplotdigitizer: Version 4.4, 2020.

[5] V. Rossum and Guido. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[6] Weisstein and E. W. Moore neighborhood. `https://mathworld.wolfram.com/MooreNeighborhood.html`

[7] Bradski, Gary, and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[8] E. Fernandes. Conheça as doenças do pâncreas: *Câncer, pancreatite aguda e cistos na glândula podem colocar a saúde em risco*. `https://apps.automeris.io/wpd/`

[9] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. *GNU Octave version 5.1.0 manual: a high-level interactive language for numerical computations*, 2019.