COMP204 Project 1: Object Counting

Alp Gökçek
Faculty of Engineering, Computer Engineering Department
MEF University
Istanbul, 34396 TR
gokcekal@mef.edu.tr

Abstract SIMD (Single Instruction Multiple Data) is a parallel algorithm that has the same operations on multiple data points performed on the class of parallel computers with multiple processing elements. In parallel shrinking algorithms, the SIMD concept is used which results in each pixel on the image modified simultaneously. In this project, parallel shrinking algorithms were used to count the number of connected components on an image by using well-known Levialdi's Parallel Shrinking Algorithm and Gokmen and Hall's TSF Parallel Shrinking Algorithm. This project covers the conditions that have to be satisfied and compares these algorithms from their number of iterations on several images.

***Index Terms*—Object counting, parallel shrinking algorithms, Levialdi algorithm, TSF algorithm**

I. Introduction & Problem Definition

A parallel algorithm is an algorithm that can be executed simultaneously on many different processing devices and then combine together to get the correct result. With the benefits of parallel programming, it is much easier to solve complex or huge problems which cannot be handled by a single computer. Parallel algorithms have got many concepts but, in this project, we concentrated on the SIMD (Single Instruction Multiple Data) concepts. In SIMD, same operations on multiple data points performed on the class of

1

parallel computers with multiple processing elements. In this project, our main goal is to count the number of connected components on an image using 2 parallel shrinking algorithms. Therefore, while implementing the algorithms to solve our problem, each pixel on the image modified simultaneously.

II. Solution and Design

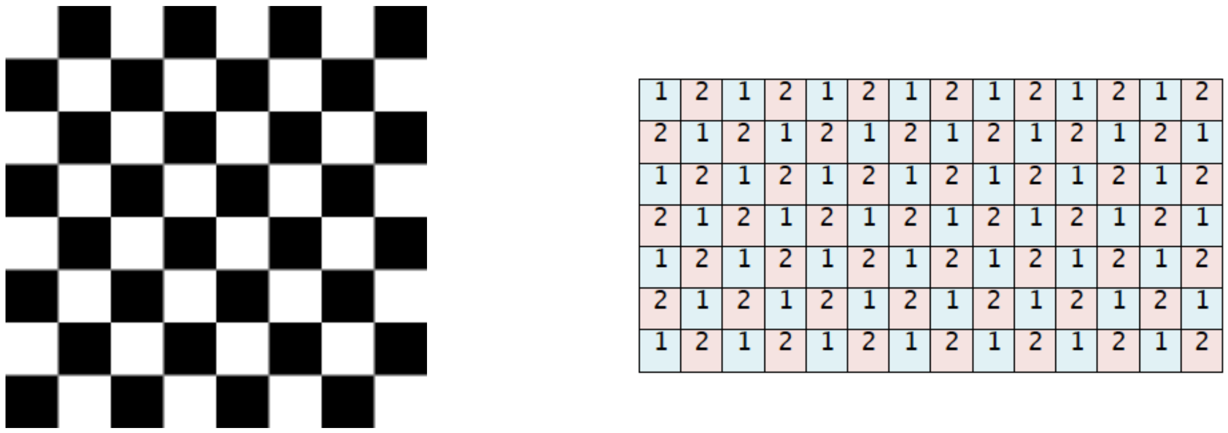
To solve our problem, we implemented 2 well-known algorithms which are Levialdi Parallel Shrinking Algorithm and the other is called TSF Parallel Shrinking Algorithm by Gokmen and Hall as the methods to solve the defined problem which is finding the number of connected components on a given image. First of all, there are two requirements that image should satisfy for both of these algorithms which are input image must have a whole black border and has to be converted into binary form. After satisfying these requirements, there are several different conditions to check while iterating over the images. In both of these algorithms, algorithms are checking isolated points, augmentation and deletion conditions and termination condition which is the same for both of these algorithms. The termination condition checks if anything has changed between last 2 iterations of the algorithm and if nothing has changed, terminates and prints the NCC(number of connected components) and number of iterations.

A. Levialdi Algorithm

I first implemented the Levialdi algorithm which is the much simpler one compared to the TSF algorithm. In Levialdi algorithm, the algorithm first checks if there is an isolated point by checking the 8-neighborhood of that central-pixel and if the value of that pixel is equal to 1 and its 8-neighborhood's each element is equal to 0. If current state satisfies these conditions, the algorithm increments the NCC variable by 1 and deletes the central pixel. After checking isolated point, the algorithm then checks deletion condition which is deleting a 1-valued central-pixel if the left, cross-left and below pixels values are equal to 0. Finally, the algorithm checks if the central pixel is equal to 0 also left and below pixels are equal to 1. If these conditions are satisfied, the algorithm then changes a 0-valued pixel to 1. After iterating over the whole image, the algorithm increments the "iteration" variable by 1 if there is any change on the image.

B. TSF Parallel Shrinking Algorithm

In TSF Algorithm, the image is divided into 2 separate subfields like a checkerboard as shown in Figure 1, and both of the subfields are getting iterated separately. On each subfield, the algorithm checks the same conditions as Levialdi algorithm (isolated point detection, augmentation, and deletion), but requirements of these conditions are different.



1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	1	2	1	2	1	2	1	2	1	2	1	2	1
1	2	1	2	1	2	1	2	1	2	1	2	1	2

Fig. 1. Checkerboard pattern vs subfields of TSF Algorithm.

Before implementing the TSF algorithm, I first implemented 2 functions which are required in the implementation of the TSF algorithm. The first function is “calc_cp” which is the number of distinct 8-connected component of one’s in central-pixel’s 8-neighborhood. The other function is called “lengthofzeros” which checks if there are 3-length or more connected zeros in central-pixel’s 8-neighborhood. After implementing these helpful functions, I started implementing the TSF algorithm. In augmentation condition of TSF algorithm, the algorithm starts checking if the central pixel is 0-valued. After passing this condition algorithm checks if “calc_cp” function’s output is valued 1 and also p8 and p2 or p8 and p6 are 1-valued where p2,p6, and p8’s position can be found in Figure 2. In isolated point detection, the algorithm first checks if central-pixel is 1-valued then if it is 1-valued, then checks if “B(p)” is 0-valued which B(p) is the number of ones in p’s 8-neighborhood. If this condition is satisfied algorithm then increments the NCC variable by 1 and deletes the central-pixel. In the last condition, which is the deletion condition, algorithm checks if the central pixel is 1-valued then if it is 1-valued, algorithm checks if “calc_cp” function’s output is valued 1 and then checks if B(p) is 1-valued then p1 and p7 is 0-valued. After satisfying these requirements, if the “lengthofzeros” function returns true, then the algorithm deletes the central-pixel. After iterating over 2-subfields, the algorithm increments the variable called iterations by 1 if there are any changes on the image.

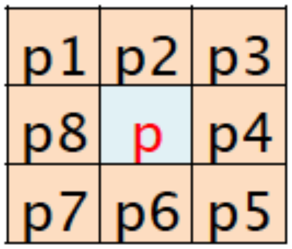


Fig. 2. 8-neigborhood of central pixel p

C. Design of Graphical User Interface

After implementing the solution methods, I implemented the Graphical User Interface which is also known as GUI. I have designed a 3x2 GUI which has functions like opening or creating an image, showing the original image, showing the iterations of TSF and Levialdi algorithms and saving the number of iterations and number of connected components of these algorithms. In Figure 3, the start-up screen of my program is shown. The user should import or create an image via the buttons in the top left corner. With clicking the "Open" button, a file dialog window pops up and the user picks and imports an image from their computer.

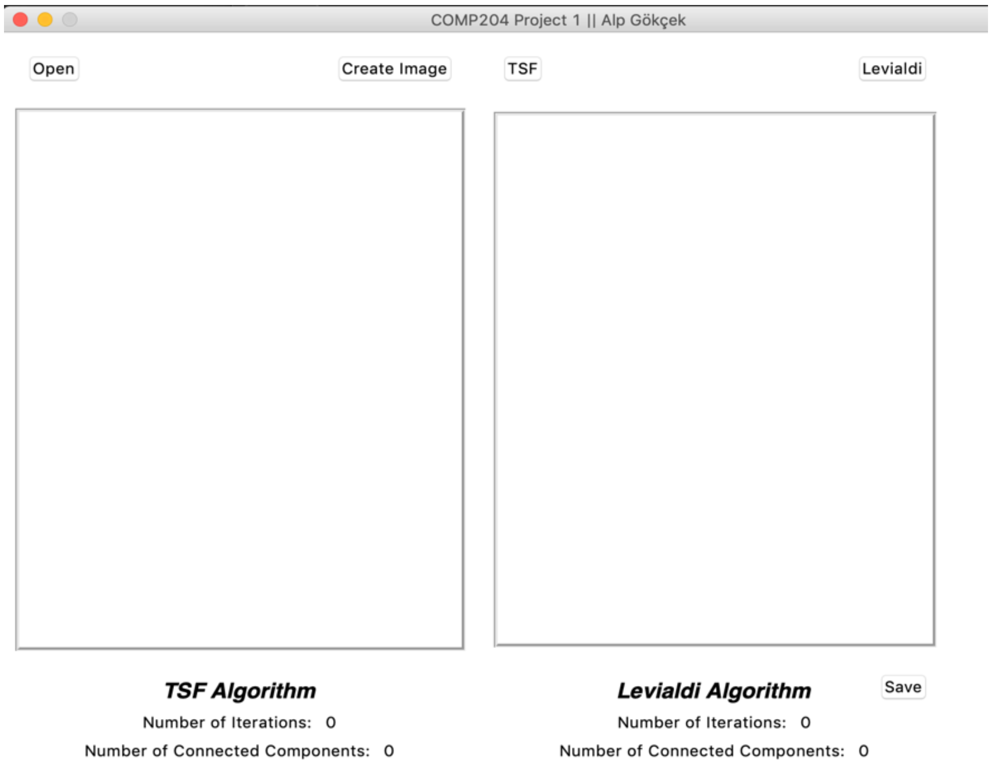


Fig. 3. Start-up screen of the program

With clicking the "Create Image" button, another frame pops up as shown in Figure 4, which has 5 different tools to create an object, erase an object and draw an object and there are also utility buttons to discard the canvas, save the canvas, choose a color for objects, create a random image etc...

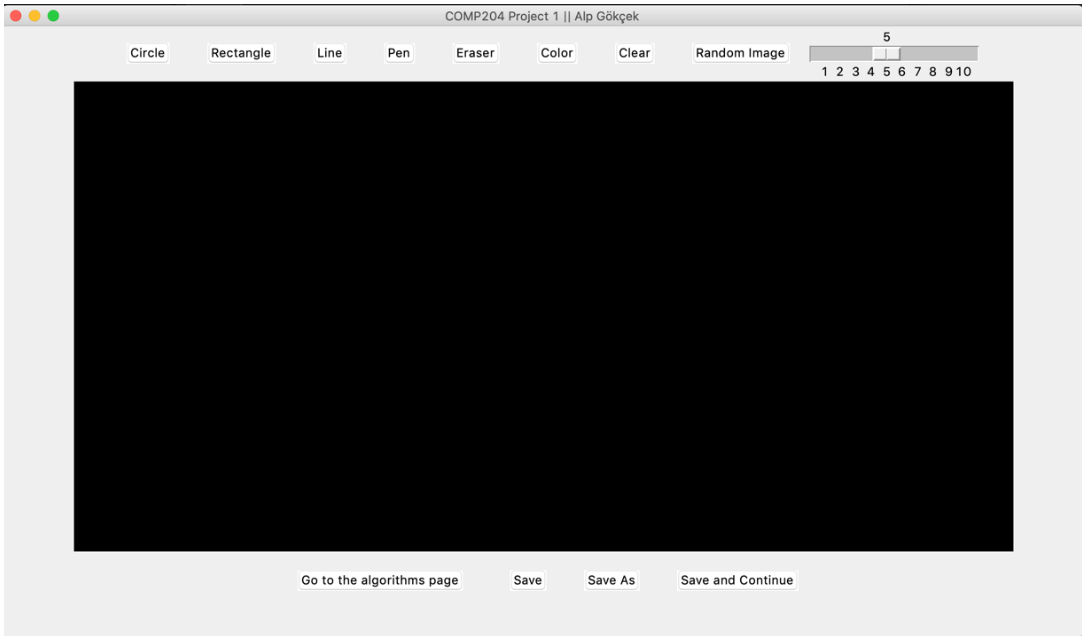
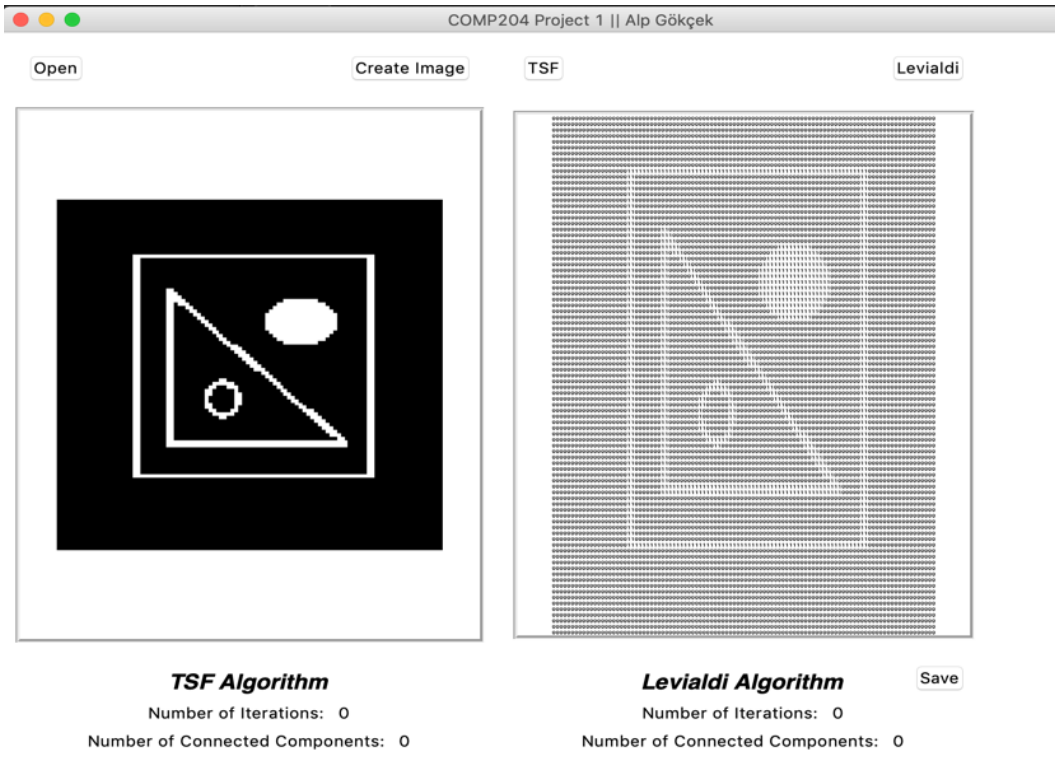


Fig. 4. Create image module

By clicking “Save and Continue” button, the user returns to the start-up screen with the image that created. After importing the image, binary form of that image will be shown on the screen as shown in Figure 5. The user should click on an algorithm button to see the live iterations of that algorithm and finally after the completion of iterations with the selected algorithm, the user could want to see iterations of the other algorithm on that image or save the output of current state of the program.



TSF Algorithm
Number of Iterations: 0
Number of Connected Components: 0

Levialdi Algorithm
Number of Iterations: 0
Number of Connected Components: 0

Fig. 5. After importing an image

III. Results

A. Number of Iterations on Increasing Number of Square Objects

First of all, I tried to check the relationship between number of iterations and increasing number of square objects on an image. Images that are compared is shown in Figure 6. In Table 1, number of squares and iterations on these squares are provided. In Graph 1, the relationship between number of iterations and increasing number of square objects is provided.

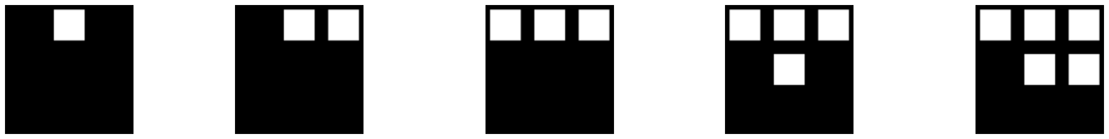
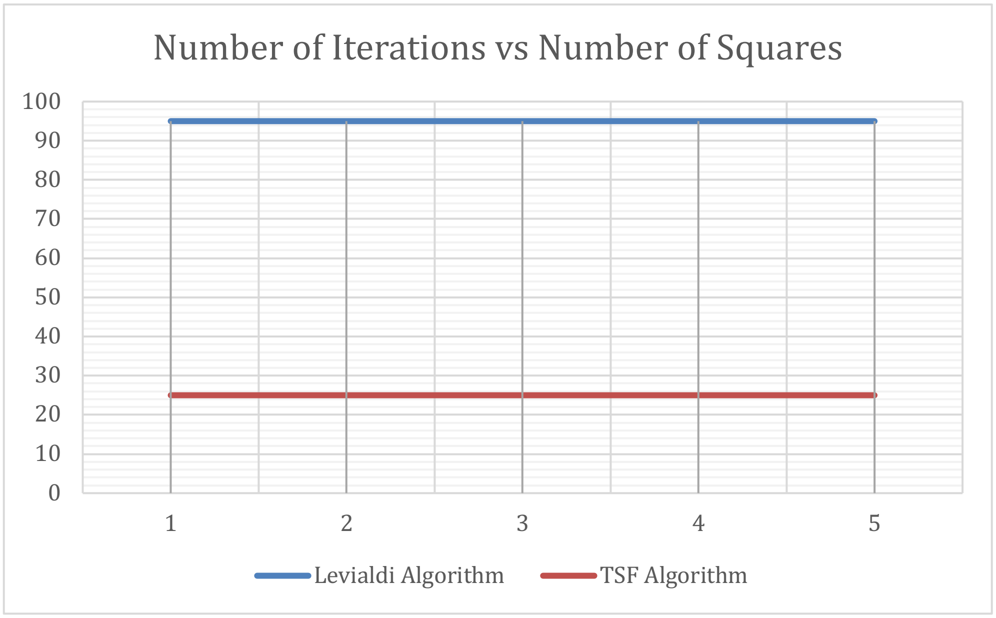


Fig. 6. Increasing number of same sized squares

Table 1. Number of iterations on images that provided in Fig.6

	Number of Iterations	
Number of Squares	**Levialdi Algorithm**	**TSF Algorithm**
1	95	25
2	95	25
3	95	25
4	95	25
5	95	25



Graph 1. Number of iterations vs Number of Squares

B. Number of Iterations on Provided Images with Project

Second of all, I tried to check the relationship between number of iterations on the images that provided with the project. Images that are compared is shown in Figure 7. In Table 2, images and their properties, number of iterations for both algorithms are provided. And lastly, in Graph 2, the performances of algorithms compared.

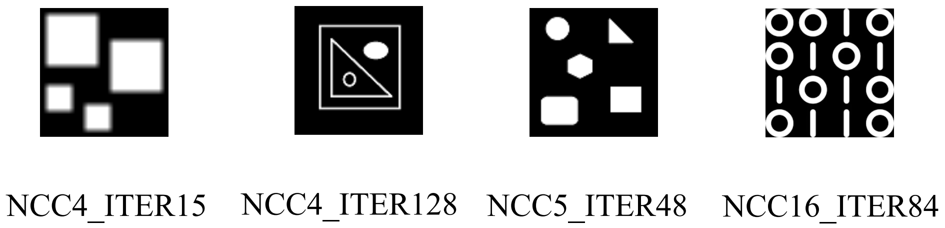
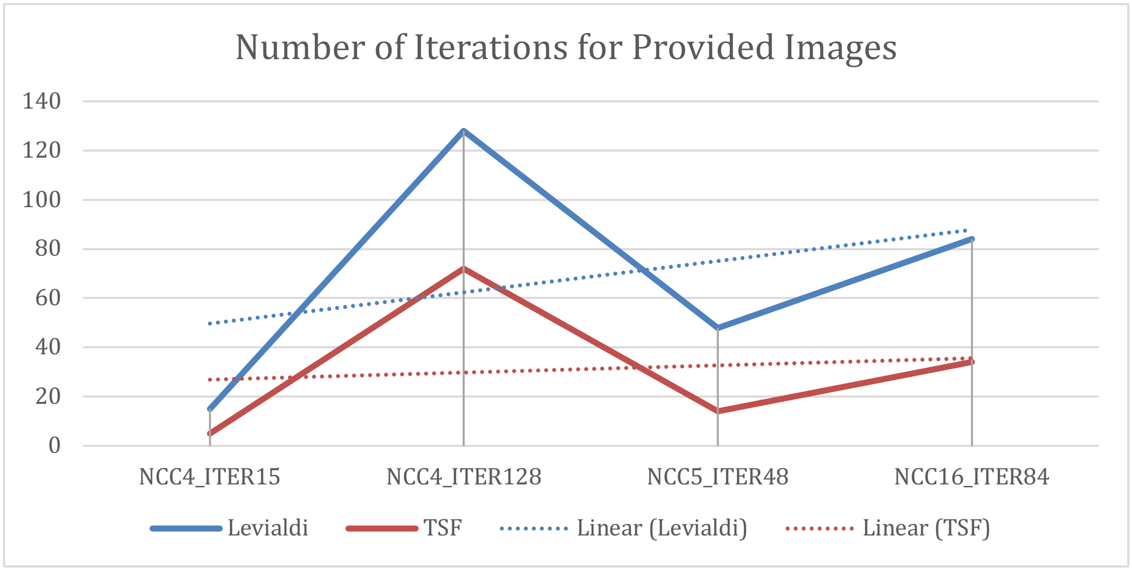


Fig. 7. Images that are provided with the project and their file names

Table 2. Number of iterations on images that provided in Fig.7

			Number of Iterations	
Image Names	**Image Dimensions**	**NCC**	**Levialdi**	**TSF**
NCC4_ITER15	22x22	4	15	5
NCC4_ITER128	102x102	4	128	72
NCC5_ITER48	102x102	5	48	14
NCC16_ITER84	227x227	16	84	34



Graph 2. Number of iterations on images that are provided in Figure 7

C. *Number of Iterations for Complex Images*

Last of all, I tried to check the relationship between number of iterations on the images that are more complex than the images provided with the project. Images that are compared is shown in Figure 8. In Table 3, images and their properties, number of iterations for both algorithms are provided. And lastly, in Graph 3, the performances of algorithms compared.

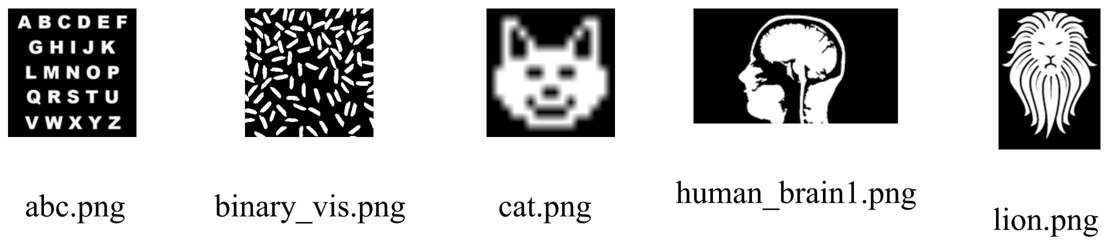
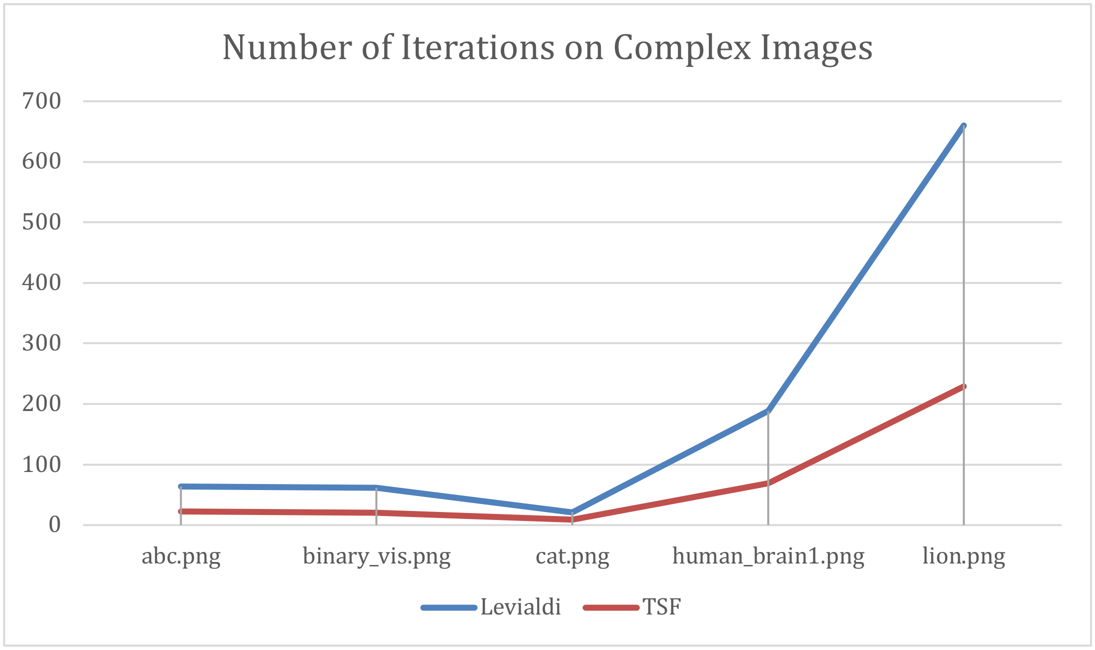


Fig. 8. Images that are compared and their file names

Table 3. Number of iterations on images that provided in Fig.8

			Number of Iterations	
Image Names	**Image Dimensions**	**NCC**	**Levialdi**	**TSF**
abc.png	302x302	26	64	23
binary_vis.png	258x258	99	62	20
cat.png	18x15	1	21	9
human_brain1.png	202x115	1	189	69
lion.png	364x502	1	660	229



Graph 3. Number of iterations on images that are provided in Figure 8

IV. Conclusion

In this project, we have implemented 2 different parallel algorithms for object counting which are by the name of Levialdi algorithm and TSF algorithm and learned their usage and properties that should satisfy to solve our problem. In results section I compared 3 different groups of images. As a result of part A, the relationship between number of iterations and increasing number of square objects on an image is stable. In other words increasing the equal sized squares won't change the number of iterations. In part B, I compared the images that are provided by their number of iterations and as a result, increment in dimensions of the image will increase the number of iterations and also TSF algorithm is solving the problem in less iterations compared to Levialdi algorithm. At part C, I compared the images that are more complex compared to images that are provided with the project and as a result, I reached the same results as part B. As the contribution of this project, I also learned how to design a GUI (Graphical User Interface), how to process an image and improve my coding skills with Python. Also, I find out how to start a programming project and planning the main phases of the software process.

V. References

[1] "Parallel Algorithm Introduction", *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/parallel_algorithm/parallel_algorithm_introduction.htm. [Accessed: 05- Mar- 2019].

[2]M. Gokmen and R. Hall, "Parallel Shrinking Algorithms Using 2-Subfields Approaches", Academic Press, Pennsylvania, 1989.

[3]"Paralel Programlama Nedir?", *Hermes İletişim*. [Online]. Available: https://www.hermesiletisim.net/dev/paralel-programlama-nedir-2#.XH1bdVMzbUp. [Accessed: 05- Mar- 2019].