05.03.2019

MEF UNIVERSTY

COMP 204 - Programming Studio

Computer Engineering

Project 1 – OBJECT COUNTING

Name =Ramazan Surname=Yetişmiş Numbe[redacted]

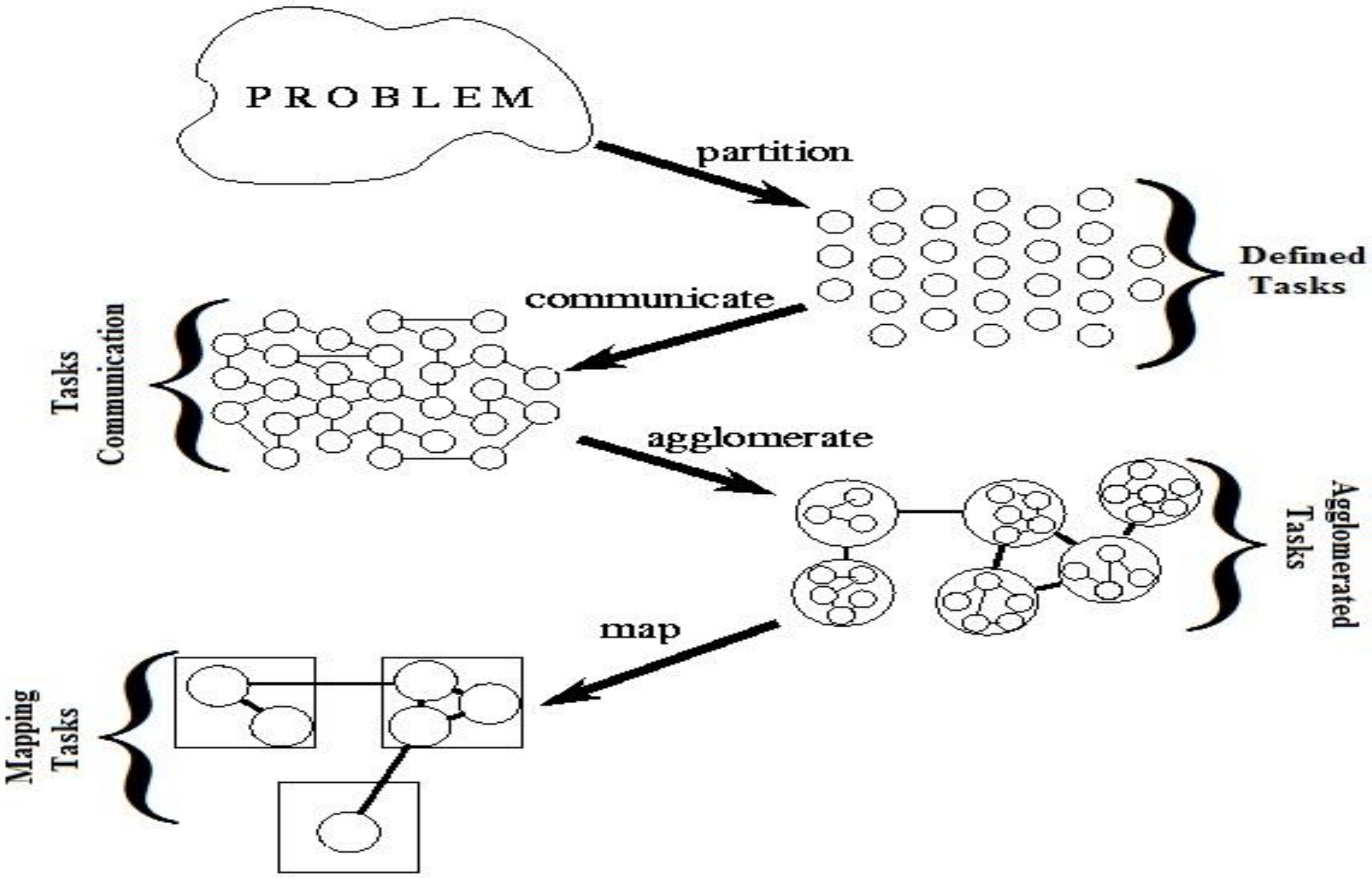

Figure 1

Abstruct:

This Project wants us to count the objects by using the parallel shrinking algorithms such as Levialdi and TSF algorithms.First we learned what is object and how can we reduce their pixels to the minimum so we can count them.This projects main goal is to compare the two algorithms due to their number of iterations at a specified image.I tried different images with these two algorithms and the results were eye catching Overall in this Project we used two algorithms and compared them by the output of the program to make neat comparasion .After the comparasion there can be understood that TSF() algorithm is faster than Levialdi() algorithm.

Problem Definition:

Object counting is used for many areas such as medical and industrial applications.Overcoming this problem is difficult but not imposiable:In this Project the essential problem is time which algortihm finds the correct NCC with leats iterations.In this applications we used parallel shrinking algoritms.Firstly we have answer the question ‘What is parallel Algorithm’.Parallel algorithm can execute several instructions simultaneously on different processing devices and then combine all the individual outputs to produce the final result.The algorithms have to satisfy some conditions such as ,they have to divide and execute seperately and after that they have to combine the parts and carry on until it terminates.If these are not stisfied we call thaht Serial Algortihm.

Solution methods:

<u>Levialdi Algorithm</u>

This algortihm firstly implemented to count the connected patterns in a binary image then it adopted to usage of the labeling procces. Basicly we can say that

The main goal is to shrink each components into an isolated pixel cornered at the top_right rectangle.The algorithm uses the 2x2(figure2) matrices to execute parallel algortihm.

- İf the figure 2 conditions are satisfied we change black to White

1	0
	1

- If the figure 3 conditiıons are satisfied replace White to Black

0	1
0	0

Figure 3

- If the index(figure4) is isolated point then change White to black(increment NCC)

0	0	0
0	1	0
0	0	0

Figure 4

- If the one of the upper 3 operations are not staisfied do nothing

These 4 are the shrinking operations.Levialdi's algorithm uses them to shrink the image until zero pixel that means no change has left and the algoritm reminates itself.The issues that have to becarefull about Levialdi:

- A correct shrinking will be obtanined if and only if black pixels that do not disconnect a component will be changed to White pixels,and White pixels are not allowed to became black pixels when this merges 2 or more components
- Each component will become unique isolated pixel
- The maximum number of steps fort he MxN bounding rectangle is M+N-1

TSF Algortihm

This algortihm uses subfields method,this method helps us to shrink from all directions at each iteration.The significant part of this method m-neigbors of a deleted pixel could not change in value at the same iteration, connectivity preservation was isured since onlym-simple 1' were deleted.We can see how this subfields looks like a chess bord(figure5)

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

Figure 5

Deletion Condition:

If the current index is 1 then we haveto follow the below conditions:

p1	p2	p3
p8	p	p4
p7	p6	p5

Figure 6

- The index is isolated index increment NCC.
- The index has one 8-path connected component
- The index has one 1 around then P1 and P7 have to be zero in figure6
- The index contains a 4-path three or more 0's

If the following conditions are satisfied the deletion condition will be occurs in the image.

Augmented Condition:

If the current index is 0 then we haveto follow the below conditions:

- The index has one 8-path connected component
- The P8,P2 is one or P8,P6is one(figure6)

If the following conditions are satisfied the ougmented condition will be occurs in the image.

We have to becarefull to execute this code because the implementations is not as simple as it seems.The above conditions have be included in two subfields seperately.

Code Analysis:

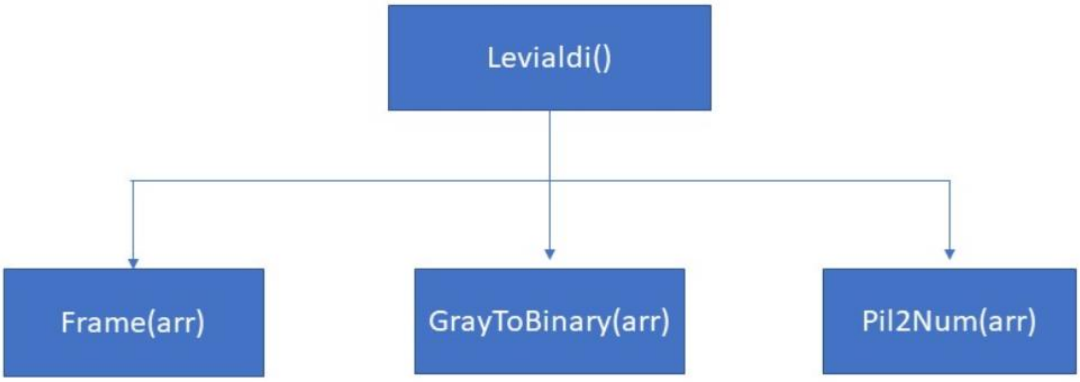

Figure 7-UML Diagram for Levialdi Algorithm

Frame(arr):

```
    nrows=len(arr[0])
    ncols=len(arr[1])
    temp = np.zeros(shape=(nrows+2, ncols+2))
    for i in range(0, nrows):
        for j in range(0, ncols):
            temp[i + 1][j + 1] = arr[i][j]
    return temp
```

Figure 8-Frame function

This function takes an as a parameter and add zeros to the four sides and returns framed array

Gray2Bin(arr):

```python
nrows = len(newarr[0])
ncols = len(newarr[1])
array=np.zeros(shape=(nrows, ncols))
for i in range(nrows):
    for j in range(ncols):
        if newarr[i][j]==True:
            array[i][j] = 1
        else:
            array[i][j] = 0
return array
```

Figure 9-Gray2Bin Fucntion

This function takes boolean array,in this array TRUE means one and False means zero,by the help of this information the given array can be simply turned to binary array.

Levialdi():

```python
global NCC, flag#flag for change,number of connected components
nrows = len(arr[0]);ncols = len(arr[1]); counter = 0;temp=arr.__copy__()
#this loops helps me to scan the array from upper right part to lower left
for i in range(1,nrows - 1):
    for j in range(ncols - 2,0,-1):
        if arr[i][j]==0:#augmented condition
            counter = counter + 1
            if (counter == (nrows - 2)*(ncols - 2)):
                flag = 1
            if (arr[i+1][j]==1 and arr[i][j-1]==1):
                temp[i][j] = 1
        else:#deletion conditions
            if arr[i][j+1]==0 and arr[i][j-1]==0 and arr[i+1][j]==0 and arr[i-1][j]==0 and arr[i+1][j+1]==0 and arr[i+1][j-1]==0 and arr[i-1][j+1]==0 and arr[i-1][j-1]==0:
                NCC = NCC + 1
                temp[i][j] = 0
            elif arr[i+1][j]==0 and arr[i][j-1]==0 and arr[i+1][j-1]==0:
                temp[i][j]=0
return temp
```

Figure 10-Levialdi/ single iteraiton

In this part of the algorithm one iteration occurs.I divided the levialdi function into 2 parts.This function scans the array from upper_right bottom.The first if condition determn,nes whether the index is 0 or 1 means Deletion or Augmented condition.The counter in the code means the number zeros.If the counter equals to the image size[(ROW-2)x(COL-2)] the termination condition occurs otherwise the code continues to execute.

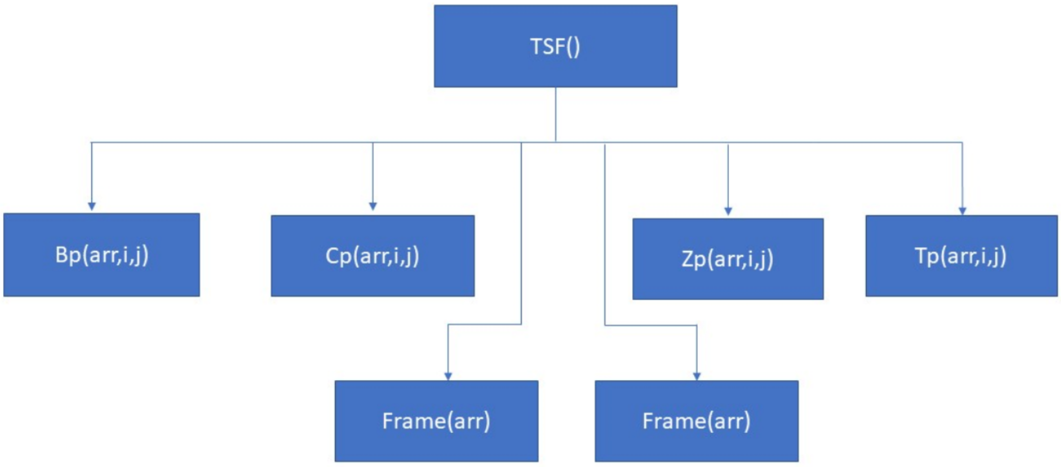

Figure 11-UML diagram of TSF() function

Bp(arr,i,j):

```
if arr[i][j+1]==1:
    counter=counter+1
if arr[i][j-1]==1:
    counter = counter + 1
if arr[i+1][j]  == 1:
    counter = counter + 1
if arr[i+1][j +1] == 1:
    counter = counter + 1
if arr[i+1][j -1] == 1:
    counter = counter + 1
if arr[i-1][j]  == 1:
    counter = counter + 1
if arr[i-1][j +1] == 1:
    counter = counter + 1
if arr[i-1][j -1] == 1:
    counter = counter + 1
return counter
```

Figure 12-Bp() function

In this function it takes the index location and array then checks the 8 neighbors whether any 1 is occurs around 8-connceted neighbors.And returns the number of ones.

Tp(arr,i,j):

```
counter=0
# defines 8 lenght array and add the neigbors to the array
temp=np.array([0,0,0,0,0,0,0,0])
temp[0]=arr[i-1][j-1];temp[1]=arr[i-1][j];temp[2]=arr[i-1][j+1];temp[3]=arr[i][j+1]
temp[4]=arr[i + 1][j + 1];temp[5]=arr[i+1][j];temp[6]=arr[i+1][j-1];temp[7]=arr[i][j-1]
#searches if there is any 0 1 transititon
for i in range(0,8):
    if temp[i]==0 and temp[(i+1)%8]==1:
        counter=counter+1
return counter
```

Figure 13-Tp() function

Function takes 3 parameters arrray and exact location.Then creates neigbor array to check the 0 to 1 transition.The algortihm checks the two sequantial indexs to determine 0 -1 transition.

Cp(arr,i,j):

```
counter=0
ones=0
# defines 8 lenght array and add the neigbors to the array
temp = np.array([0, 0, 0, 0, 0, 0, 0, 0])
temp[0] = arr[i - 1][j - 1];temp[1] = arr[i - 1][j];temp[2] = arr[i - 1][j + 1];temp[3] = arr[i][j + 1]
temp[4] = arr[i + 1][j + 1];temp[5] = arr[i + 1][j];temp[6] = arr[i + 1][j - 1];temp[7] = arr[i][j - 1]
#if there is Conncected components it replaces the correct zeros with 1
for i in range(0,7,2):
    if temp[i]==1 and temp[(i+2)%8]==1:
        temp[i+1]=1
for i in range(0, 8):...
if ones==8:#if all of them are 1 that means the CP() is 1
    return 1
else:
    return counter
```

Figure 14-Cp() Function

This function again takes the same parameters.Then define neighbor array.Then It iterates the array and looks the 3 sequantial indexes,if the first and third indexes are one then that means it is a connected component so it replaceses the second index with one too.Then it counts the 0-1 transition.If all of the components are 1 the it returns 1.

Zp(arr,i,j):

```
counter = 0
temp = np.array([0, 0, 0, 0, 0, 0, 0, 0])
#defines 8 lenght array and add the neigbors to the array
temp[0] = arr[i - 1][j - 1];temp[1] = arr[i - 1][j];temp[2] = arr[i - 1][j + 1];temp[3] = arr[i][j + 1]
temp[4] = arr[i + 1][j + 1];temp[5] = arr[i + 1][j];temp[6] = arr[i + 1][j - 1]; temp[7] = arr[i][j - 1]
for i in range(0,8):#searches if there is 3 or more zeros
    if temp[i]==0 and temp[(i+1)%8]==0 and temp[(i+2)%8]==0:
        counter=counter+1
return counter
```

Figure 15-Zp() Function

The function determines whether in neighbor array the given index has 1 or more 4 connected zeros .Amoung 3 sequantial index if they all are zero then increment counter.

TSF():

```
j = 2
if i % 2 == 1:
    j = 1
while j < ncols-1:
    cp=Cp(arr,i,j)#calls the Cp()
    if arr[i][j]==0:#if index is 0 then
        counter+=1#looks if there is any change
        if (cp==1) and ((arr[i][j-1]==1 and arr[i-1][j]==1) or(arr[i][j-1]==1 and arr[i+1][j]==1)):#this is the deletion condition
            arr[i][j]=1
    else:  # this part we look for the aougmented condition because index is 1
        bp = Bp(arr, i, j)  # calls for bp()
        zp = Zp(arr, i, j)  # calls the Zp()
        if bp == 0:  # means isolated point
            NCC_TSF += 1
            arr[i][j] = 0
        if bp == 1:  # this is  nested if statement
            if cp == 1 and zp > 0 and (arr[i - 1][j - 1] == 0 and arr[i + 1][j - 1] == 0):  # if bp is 1 then these 3 have to be TRUE
                arr[i][j] = 0
        else:  # if it is different tham 1 then these 2 have to be TRUE
            if cp == 1 and zp > 0:
                arr[i][j] = 0
    j = j + 2#increment J with 2
```

Figure 16-TSF() Function_First subfield

This algorithm first checks the current index's value to determine which condition the index is going to enter.If the index is one than that means the Augmented coundition is going to accurs.I already talked about this conditions now I am going to mention about dividing the subfields.In my algorithm if the row value is odd the the column will begin with the first one if not it will be second column.So finaly all of these conditions(figure16) have to be included in subfiled2.

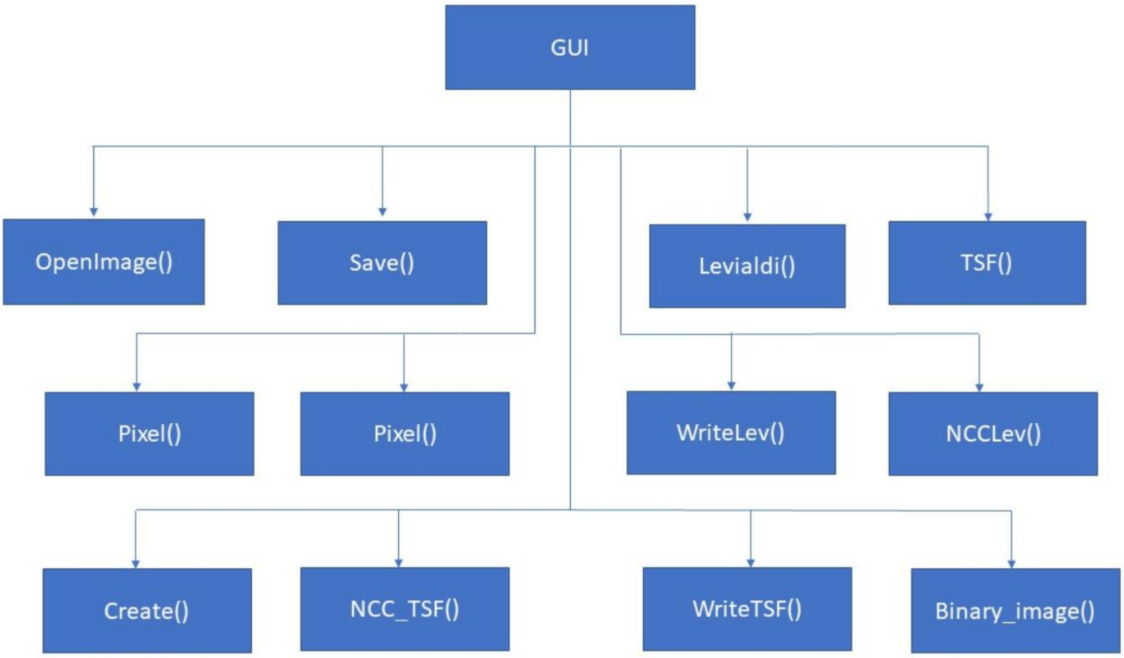

Figure 17-GUI UML diagram

Results:

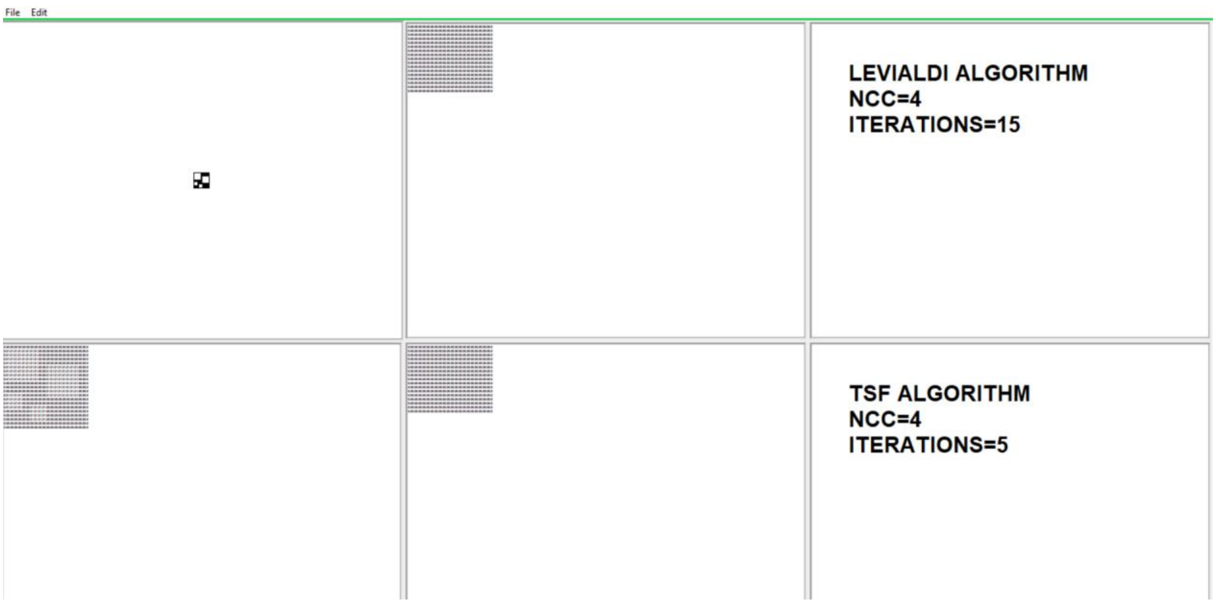

Figure 18-First image size of 20x20

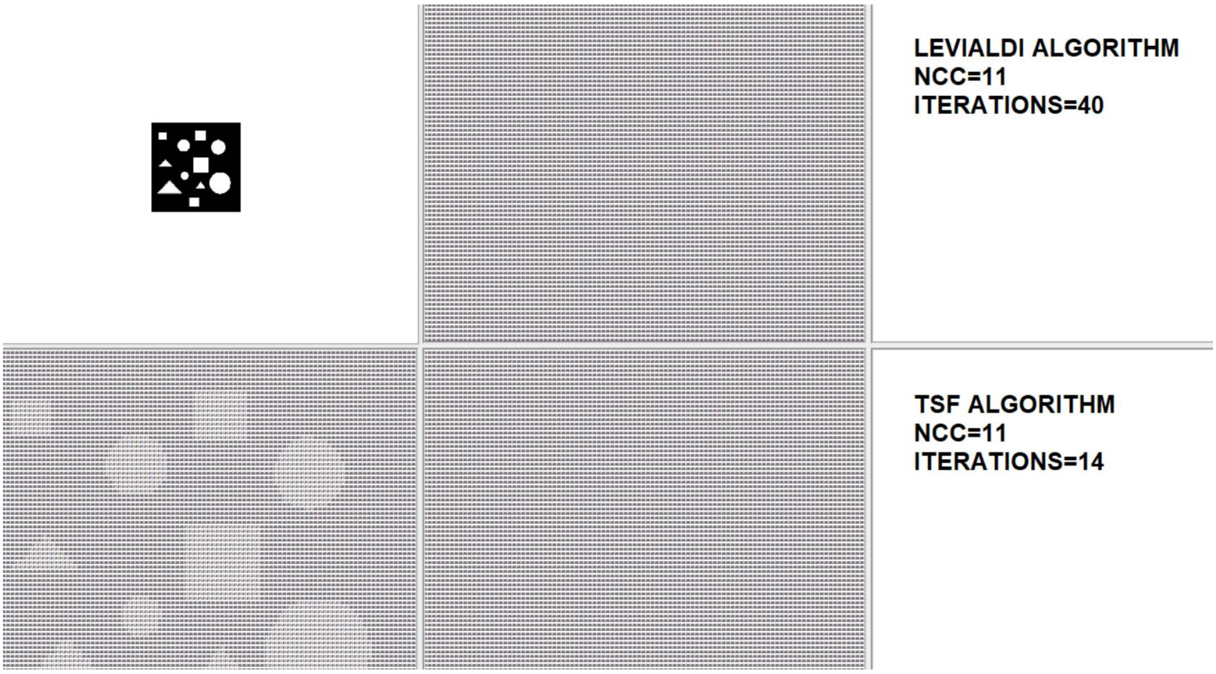

Fıgure 19-Second Image size of 100x100

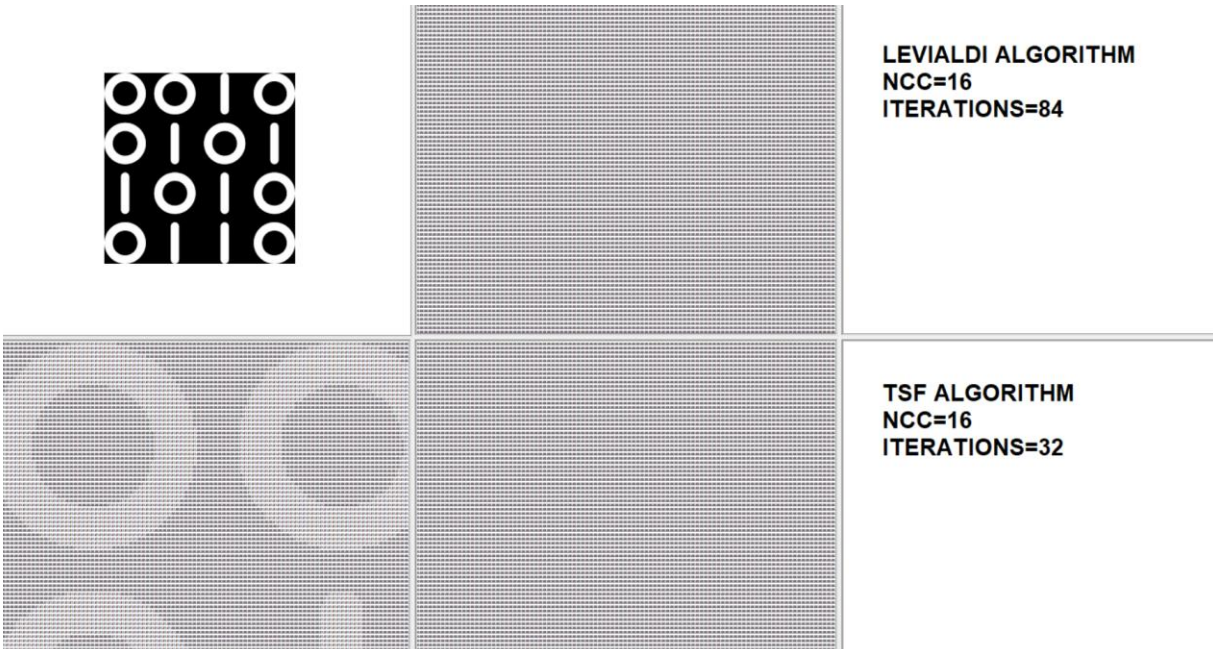

Fıgure 20 Third İmage size of 225x225

Contribution of this project:

In this Project I learned how to write neat algorithm because If I do not write my codes proper I saw that they did not work.Also learned how to Schedule my time this Project was my first long term Project in order to cope eith this Project I learned to manage my time.During this Project I deal with many difficulties such as lack of Pyhton language knowledge but I delt with them succesfully.To sum up what I learned obut this Project ,now I know what the image is and how I can perform operations with an image.

Conclusions:

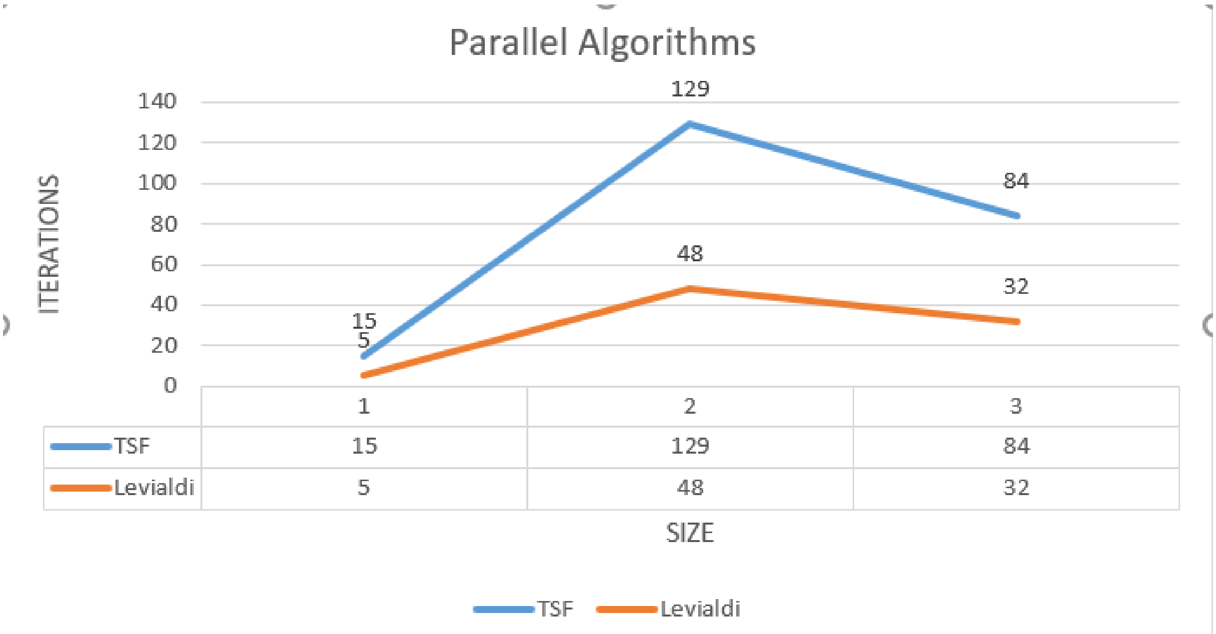

Figure 21-Graph of the iterations in figures 18-20

As a conclusion that can be simply said that the TSF algorithm is Faster than Levialdi.In parallel algorithms the most important trade of is timing.So in TSF number of iteration is less than Levialdi that means TSF is efficiant algortihm amoung these two algorithms.

Referances:

1. Shi, H., & Ritter, G.X. (1995). A new parallel binary image shrinking algorithm. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society, 4 2*, 224-6 .
2. Rosenfield, A. (2019). *Topological Algorithms for Digital Image Processing*. [online] Google Books. Available at: https://books.google.com.tr/books?id=4Yz3gLkISnwC&pg=PA80&lpg=PA80&dq=TSF+Shrinking+algorithm&source=bl&ots=twlxIuyVRJ&sig=ACfU3U2jfzoWfHARbsi1PgnXjBiXDyQb1A&hl=tr&sa=X&ved=2ahUKEwjP156YruvgAhXF_KQKHcNRBqAQ6AEwAHoECAUQAQ#v=onepage&q&f=false [Accessed 5 Mar. 2019].
3. Blelloch, G. (2019). [online] Cs.cmu.edu. Available at: http://www.cs.cmu.edu/~blelloch/papers/BM04.pdf [Accessed 5 Mar. 2019].