# IP Homework Set #3 - Binary Images

## Due

Homework Due: **14/012/15**
Submit the homework using the Submit system.

Explanations on how to use the Submit system can be found in help
On any problem, please contact the BODEK

In the following function the images are given as matrices of doubles. In those images 1 represents the object and 0 (black) represents the background.

**1.** Write a matlab function which finds the connected components in a binary image and returns a matrix with the same size in which each connected componnent is tagged with a different label. The tags should be sequential.

Format of Matlab function :

**function [newImg] = tagConnectedComponents(img)**

**2.** Write a matlab function which skeletonize objects in a given binary image and returns the new binary image.

Format of Matlab function :

**function [newImg] = skeletonizeImage(img)**

Notes:
Grade will depend on correct performance and on clean programming and documentation. Do not forget to put Names and Student I.D. in the Documentation.

הערה לפונקציה מזהה עצמים
הפונקציה tagConnectedComponents אינה טובה- היא לא עובדת עבור-
התמונות שמופיעות באתר הקורס!

הרעיון נכון אך יש משהו שגוי!
יש לשים לב לכך!!!

יום טוב,
I.G.F

```
function [newImg] = skeletonizeImage(img)
%function which skeletonize objects in a given binary
% image and returns the new binary image
[r,c]=size(img);
newImg=img;
count=1;
%if we didnt change anything in the matrix-stop the while loop
while count>0
    count=0;
    for i=1:r
        for j=1:c
            %represent the 4 neighbor of pixcel+the main
pixcel
            cv]=0,0,0,0,0;[
            cv(1)=img(i,j);
            %take his 4 neighbor
            if i>1
                cv(2)=img(i-1,j);
            end
            if j>1
                cv(3)=img(i,j-1);
            end
            if i<r
                cv(4)=img(i+1,j);
            end
            if j<c
                cv(5)=img(i,j+1);
            end
            %check if all the neighbors are equal to the main
pixcel
            summ=sum(cv==cv(1));
            if summ==5 && cv(1)~=0
                newImg(i,j)=img(i,j)+1;
                %count how much change we done
                count=count+1;
            end
        end
    end
    img=newImg;
end
newImg=zeros(r,c);
for i=1:r
    for j=1:c
       cv]=0,0,0,0;[
       %take his 4 neighbor
        if i>1
            cv(1)=img(i-1,j);
        end
        if j>1
            cv(2)=img(i,j-1);
        end
        if i<r
            cv(3)=img(i+1,j);
        end
        if j<c
            cv(4)=img(i,j+1);
        end
        %take the max between his neighbors
        [idx idx]=max(cv);
        %check if the main pixcel his the biggest from others
        if img(i,j)>=cv(idx) && img(i,j)~=0;
            %if yes mark him in new img
            newImg(i,j)=1;
        end
    end
end
end
```

```
function [newImg] = tagConnectedComponents(img)
%It finds the connected components in a binary image and
returns a matrix with
% the same size in which each connected componnent is tagged
with a different label

%get img with indexs on the img and get matrix of conectors
[newImg,A]=setTag(img);
%get vector of conection
cv=convertTag(A);
%put the new conection on img weith index
newImg=putTag(newImg,cv);
end

function[newImg,A]=setTag(img)
%matrix of conectors
A=zeros)1,1;(
index=0;
[r,c]=size(img);
newImg=zeros(r,c);
%save the 2 neighbors : up and left
cv=zeros)1,2;(
for i=1:r
    for j=1:c
        cv(1,1)=0;
        cv(1,2)=0;
        if img(i,j)==1
            %get up neighbor
            if i>1
                cv(1,1)=newImg(i-1,j);
            end
           %get left  neighbor
            if j>1
                cv(1,2)=newImg(i,j-1);
            end
            summ=sum(cv);
            switch summ
                %if the both neighbor are 0
            case }0{
                index=index+1;
                newImg(i,j)=index;
                A(index,index)=1;
                %if up neighbor!=0 and  left  neighbor==0
            case {cv)1,1(}
                newImg(i,j)=cv)1,1;(
                %if left  neighbor!=0 and up neighbor==0
            case {cv)1,2(}
                newImg(i,j)=cv)1,2;(
                %if  both left  neighbor and up neighbor!=0
                otherwise
                %take the left neighbor index
                newImg(i,j)=cv)1,2;(
                %put in the matric of conectors 2 new
connections
                ind1=cv)1,1;(
                ind2=cv)1,2;(
                A(ind1,ind2)=1;
                A(ind2,ind1)=1;
            end
        end
    end
end
end

function[cv]=convertTag(A)
r=size(A);
newA=A*A;
newA(newA>0)=1;
%find the most "right" matrix of conectors
while newA~=newA*newA
    newA=newA*newA;
    newA(newA>0)=1;
end
A=newA;
index=1;
cv=zeros(1,r);
%take the first line of matrix of conectors
cv(1,1:r)=cv(1,1:r)+A(1,1:r);
summ=sum(cv==0);
%check if we didnt get only matrix of zeros
checkZero=(A==0);
SumcheckZero=sum(sum(sum(checkZero)));
[a1,a2]=size(A);
%do it when we have zeros in cv and we didnt get matrix of
zeros
while summ>0 && SumcheckZero -a1*a2~=0
    index=index+1;
    %find the first zero in cv
     [idx idx]=min(cv);
     %dont show warnnig on idx
     warning('off','all')
     %add line of matrix of coenctors in place first zero of
cv*index
    cv(1,1:r)=cv(1,1:r)+index*A(idx,1:r);
    %check if we still have zeros
    summ=sum(cv==0);
end
end

function[newImg]=putTag(newImg,cv)
[r,c]=size(newImg);
for i=1:r
    for j=1:c
        %put the index on the pixcel
        if  newImg(i,j)~=0
        newImg(i,j)=cv(1, newImg(i,j));
        end
    end
end
end
```