*Amirali Rahimi* *810100146*

# بخش اول)

تصویر p1.m:

```
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'Choose an Image');
image = imread([path, file]);
% figure
% imshow(image);

image = imresize(image,[300 500]);
% figure
% imshow(image);

grayimage=mygrayfun(image);
% figure
% imshow(grayimage);

threshold= 255*graythresh(grayimage);
binaryimage=~mybinaryfun(grayimage,threshold);
% figure
% imshow(binaryimage);

cleanimage=myremovecom(binaryimage,400);
% figure;
```

ابتدا با استفاده از دستور uigetfile تصویر مورد نظر را در image ذخیره می کنیم. در هر مرحله می توانیم برای مشاهده ی نتیجه هر قسمت، دو خطی که در زیر آن کامنت شده را آنکامنت کرده و ران می کنیم. برای مثال تصویر نتیجه حاصل از uigetfile:

سپس  نتیجه حاصل از resize کردن عکس:


نتیجه حاصل استفاده از تابع  mygrayfun که عکس را به یک عکس خاکستری تبدیل می کند: (جلوتر به توضیح پیاده سازی هر یک از توابعی که تعریف کردیم می پردازیم.)

نتیجه استفاده از mybinaryfun که عکس را به یک عکسی که هر بیت آن یا 1 است یا 0 تبدیل می کند(threshold ای که به آن پاس دادیم، با استفاده از تابع آماده ی متلب که از راه otsu آن را محاسبه می کند، بدست آوردیم.):


نتیجه ی استفاده از myremovecom به ازای n=400 (که مقدار آن تجربی به دست آمده) که باعث حذف آبجکت های کوچک می شود که آن را در cleanimage ذخیره می کنیم و نمایش می دهیم:


تصویر ادامه ی p1.m:

mybinaryfun.m | p2.m | managegrayphotos.m | findplate.m | p1.m | p3.m | p2fun.m

```
19 -    cleanimage=myremovecom(binaryimage,400);
20      % figure;
21      % imshow(cleanimage);
22
23 -    background=myremovecom(binaryimage,2300);
24      % figure;
25      % imshow(background);
26
27 -    finalimage=logical(cleanimage-background);
28      % figure;
29      % imshow(finalimage);
30
31 -    [label,num]=mysegmentation(finalimage);
32 -    figure;
33 -    imshow(finalimage);
34 -    for i = 1:num
35 -        [row,col]=find(label==i);
36 -        thissegment=finalimage(min(row):max(row),min(col):max(col));
37 -        rectangle('Position', [min(col),min(row),max(col)-min(col),max(row)-min(row)], '
```

نتیجه ی استفاده از myremovecom به ازای n=2300 (که مقدار آن تجربی به دست آمده) که باعث می شود background تصویر مشخص شود چرا که تمامی آبجکت هایی که بیش از اندازه بزرگ نیستند را حذف می کند را در background ذخیره می کنیم و نمایش می دهیم:

حال با کم کردن background از cleanimage، به یک تصویری که تقریبا ایده آل است و نه بک گراند و نه لکه های اضافی دارد می رسیم و آن را در finalimage ذخیره می کنیم:

در ادامه، دستور mysegmenation را استفاده می کنیم. برای اینکه خروجی آن را بتوانیم به صورت تصویری مشاهده کنیم، با استفاده از یک for که تصویر آن در بالا آمده، دور هر آبجکت شناسایی شده، یک مستطیل سبز می کشیم. که خروجی آن به شکل زیر است:


تصویر ادامه p1:

```
+1 | p2.m | managegrayphotos.m | findplate.m | p1.m | p3.m | p2fun.m | myremovecom.m
39
40      %loading the map set:
41 -    folder='p1\MapSet';
42 -    filelist = dir(folder);
43 -    mapset=cell(2,size(filelist,1)-2);
44 -    for i=3:size(filelist)
45 -        filename=fullfile(folder,filelist(i).name);
46 -        mapset{1,i-2}=imread(filename);
47 -        mapset{2,i-2}=filelist(i).name(1);
48 -    end
```

حال همانطور که پیداست mapset را لود می کنیم.

تصویر ادامه p1:

```
+1 | p2.m | managegrayphotos.m | findplate.m | p1.m | p3.m | p2fun.m | myremovecom.m | +
49 -   corrthreshold = 750;
50 -   result=[];
51 -  for i=1:num
52 -      [row,col]=find(label==i);
53 -      thissegment=finalimage(min(row):max(row),min(col):max(col));
54 -      thissegment=imresize(thissegment,[42 24]);
55 -      corr=zeros(1,size(mapset,2));
56 -      for j=1:size(mapset,2)
57 -          newmat=~bitxor(thissegment,mapset{1,j});
58 -          corr(j)=sum(newmat,'all');
59 -      end
60 -      [corrval,index]=max(corr);
61 -      if(corrval>corrthreshold)
62 -          result=[result mapset{2,index}];
63 -      end
64 -  end
65 -   fprintf('%s',result);
66 -   file = fopen('number_Plate.txt', 'wt');
67 -   fprintf(file,'%s\n',result);
```

حال در حلقه ای که در بالا مشاهده می کنید، شروع به correlation گرفتن بین آبجکت های پیدا شده و اعضای maplist می کنیم. (مقدار corrthreshold، به صورت تجربی و با تست کردن انواع تست کیس ها بدست آمد.) تنها یک نکته در تصویر بالا نهفته است، bitxor~ است که دلیل استفاده از آن این است که در ماتریس دو بعدی، هر جا هر دو صفر بودند، یک امتیاز مثبت و هر جا هر دو یک بودند، یک امتیاز مثبت باید دریافت کنیم.( می توانستیم به جای آن از corr2 استفاده کنیم.) نتیجه ی غایی در result ذخیره می شود. آن را پرینت کرده و در فایل number_plate.txt هم ذخیره می کنیم. خروجی برای مثالی که آن را بررسی کردیم، به شکل زیر است:

```
>> p1
DL5CH8855>>
```

که دقیقا همین متن در فایل مورد نظر نیز ذخیره شده است. به ازای تمام عکس های موجود در فولدر p1، این تابع تست شده و خروجی صحیح گرفته است.

حال به توضیح توابعی که باید تعریف می کردیم، می پردازیم:

تصویر تابع mygrayfun:

```
function grayimage=mygrayfun(image)
    grayimage=zeros(size(image,1),size(image,2));
    for i=1:size(image,1)
        for j=1:size(image,2)
            grayimage(i,j)=0.299*image(i,j,1)+0.578*image(i,j,2)+0.114*image(i,j,3);
        end
    end
    grayimage=uint8(grayimage);
end
```

نحوه ی کار کردن این تابع، واضح است و نیازی به توضیح ندارد.

تصویر تابع mybinaryfun:

```
function binaryimage=mybinaryfun(grayimage,threshold)
    binaryimage=zeros(size(grayimage,1),size(grayimage,2));
    for i=1:size(grayimage,1)
        for j=1:size(grayimage,2)
            if(grayimage(i,j)<threshold)
                binaryimage(i,j)=0;
            else
                binaryimage(i,j)=1;
            end
        end
    end
    binaryimage=logical(binaryimage);
end
```

نحوه ی کار کردن این تابع، واضح است و تنها نکته این است که پیکسل هایی که مشکی نزدیک تر است را مشکی می کنیم. یعنی 0. و در هنگام استفاده از این تابع، از ~ آن استفاده می کنیم چرا که می خواهیم اعداد ما سفید شوند و با mapset همخوانی داشته باشند.

تصویر تابع myremovecom:

```
function cleanimage=myremovecom(binaryimage,n)
    [row,col]=find(binaryimage==1);
    nodes=[row';col'];
    objectSet={};
    explored=false(size(binaryimage));
    adjRow=[1,1,1,0,0,-1,-1,-1];
    adjCol=[-1,0,1,-1,1,-1,0,1];
    for i=1:size(nodes,2)
        object=[];
        myQueue=[];
        if(explored(nodes(1,i),nodes(2,i))==false)
            explored(nodes(1,i),nodes(2,i))=true;
            object=nodes(:,i);
            myQueue=nodes(:,i);
        else
            continue;
        end

        while size(myQueue,2)
        while size(myQueue,2)
            X = myQueue(1,1);
            Y = myQueue(2,1);
            myQueue(:,1)=[];


            for j=1:size(adjRow,2)
                if((X+adjRow(j)>0) && X+adjRow(j)<=size(binaryimage,1) &&...
                   (Y+adjCol(j)>0) && Y+adjCol(j)<=size(binaryimage,2) &&...
                   explored(X+adjRow(j),Y+adjCol(j))==false &&...
                   binaryimage(X+adjRow(j),Y+adjCol(j))==1)
                      explored(X+adjRow(j),Y+adjCol(j))=true;
                      myQueue=[myQueue [X+adjRow(j);Y+adjCol(j)]];
                      object=[object [X+adjRow(j);Y+adjCol(j)]];
                end
            end
        end
        objectSet = [objectSet object];
    end
```

```
    end
    %remove under n connected components:
    toErase=false(size(binaryimage));
    for i=1:size(objectSet, 2)
        if(size(objectSet{i},2)<n)
            index=sub2ind(size(binaryimage),objectSet{i}(1,:),objectSet{i}(2,:));
            toErase(index)=1;
        end
    end
    cleanimage=logical(binaryimage-toErase);
end
```

ایده حل این قسمت، استفاده از الگوریتم BFS است، به این صورت که ابتدا تمام پیکسل هایی که 1 هستند را پیدا می کنیم. هر کدام از این ها، یک node از گرافی که داریم مسئله را به آن مدل می کنیم می باشند. حال از یکی از این node ها شروع کرده و ادامه ی الگوریتم را که بدین شرح است، دنبال می کنیم. در هر node که حضور داریم، تنها node های دیگری که می توانیم به آنها یال داشته باشیم، 8 عضو مجاور آن می باشند. هر کدام از آنها را بررسی می کنیم و در صورتی که قبلا آنها را ملاقات نکرده بودیم، به queue اضافه می کنیم. ادامه ی این الگوریتم کاملا مطابق BFS است. در انتها، آبجکت های مختلفی که شناسایی کردیم، در objectSet ذخیره شده اند. در for آخر، سایز هرکدام را بررسی می کنیم و آنهایی که کمتر از n باشند را از عکس حذف می کنیم.

تصوير تابع mysegmentation:

```
function [label,num]=mysegmentation(image)
    [row,col]=find(image==1);
    nodes=[row';col'];
    objectSet={};
    explored=false(size(image));
    adjRow=[1,1,1,0,0,-1,-1,-1];
    adjCol=[-1,0,1,-1,1,-1,0,1];
    for i=1:size(nodes,2)
        object=[];
        myQueue=[];
        if(explored(nodes(1,i),nodes(2,i))==false)
            explored(nodes(1,i),nodes(2,i))=true;
            object=nodes(:,i);
            myQueue=nodes(:,i);
        else
            continue;
        end

        while size(myQueue,2)
            X = myQueue(1,1);
            Y = myQueue(2,1);
            myQueue(:,1)=[];


            for j=1:size(adjRow,2)
                if((X+adjRow(j)>0) && X+adjRow(j)<=size(image,1) &&...
                   (Y+adjCol(j)>0) && Y+adjCol(j)<=size(image,2) &&...
                   explored(X+adjRow(j),Y+adjCol(j))==false &&...
                   image(X+adjRow(j),Y+adjCol(j))==1)
                      explored(X+adjRow(j),Y+adjCol(j))=true;
                      myQueue=[myQueue [X+adjRow(j);Y+adjCol(j)]];
                      object=[object [X+adjRow(j);Y+adjCol(j)]];
                end
            end
        end
        objectSet = [objectSet object];
    end
```

```
    %label objects:
    label=zeros(size(image));
    num=0;
    for i=1:size(objectSet, 2)
        num=num+1;
        thisObject=cell2mat(objectSet(i));
        index=sub2ind(size(image),thisObject(1,:),thisObject(2,:));
        label(index)=i;
    end
end
```

این قسمت دقیقا مشابه قسمت قبل می باشد با این تفاوت که در انتها، یک label و num خروجی می دهیم که دقیقا مشابه خروجی bwlabel می باشد.

## بخش دوم)

در این بخش، تنها کاری که نیاز است انجام دهیم، این است که یک دیتاست از پلاک های فارسی جمع آوری کنیم. دقت شود که از آنجایی که آبجکت های کوچک حذف می شوند، نقطه ی حروف در پلاک هم حذف می کنیم. از این رو، دیتاستی که تهیه کردیم، فاقد نقطه ی حروف می باشد.p2.m از تابع p2fun استفاده کرده که این تابع دقیقا مشابه p1.m است و تنها تفاوت آن، مقدار corrthreshold و مقادیر n مورد نیاز برای حذف background و حذف آبجکت های کوچک می باشد. برای مثال، برای تصویر زیر، خروجی زیر را خواهیم داشت:


```
>> p2
63ghe76655>>
```

که می بینیم مقادیر درستی، نمایش داده شده اند. به ازای تمام عکس های موجود در فولدر p2، این تابع تست شده و خروجی صحیح گرفته است.

## بخش سوم)

ایده ای که برای این قسمت پیاده سازی کردیم، این است که bluestrip ای که در سمت چپ پلاک وجود دارد را با تصویر موجود، cross correlation می گیریم. در ابتدا یک پیاده سازی اولیه از این مسئله داشتم و آن هم این بود که یک نوار آبی با سایز ثابت را در عکس correlation می گرفتم. این راه حل برای تعدادی از تست کیس ها جواب مناسب ارائه میداد ولی برای سایر تست کیس ها جواب درستی نمیداد. برای مثال در عکس زیر، جای پلاک را به اشتباه پیدا کرده است:


این مشکل، از این نشأت می گیرد که ما با یک نوار آبی با سایزی ثابت در عکس شروع به corroloation گرفتن می کنیم. این باعث می شود که اگر scale عکس ما کمی جا به جا شود، دیگر جوابی که به ما داده می شود، جواب درستی نخواهد بود. برای رفع این مشکل، آن تصویری از نوار آبی لود کرده بودیم را توسط تابع زیر، 40 سایز مختلف از آن تولید می کنیم و تابع findplate را روی هر کدام از آنها صدا میزنیم.

```
39    function bluestrips = generatebluestrips(bluestrip)
40 -      bluestrips=cell(1,40);
41 -      for i=1:40
42 -          bluestrips{i}=imresize(bluestrip,[NaN size(bluestrip,2)-i+1]);
43 -      end
44 -  end
```

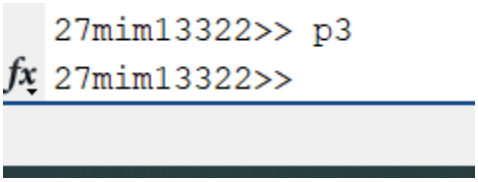
با انجام این اصلاح گفته شده، به نتیجه زیر می رسیم:

نوار آبی ای که یافت کردیم:


پلاکی که یافت کردیم:

و نتیجه ی نهایی:


که این عکس را به تابع p2fun پاس می دهیم و مابقی کار مانند قبل است. در نهایت خروجی ای که از برنامه میگیریم مطابق زیر است که مقدار صحیحی است:




همچنین p3.m را بر روی تمامی عکس های موجود در فولدر p3 تست کردیم و جواب صحیحی از آنها گرفتیم.

تنها نکته باقی مانده این است که بعد از اینکه نوار آبی را پیدا کردیم، پیدا کردن پلاک به این صورت است که نقطه ی چپ بالای مستطیل را کمی بالاتر و چپ تر از مقداری که پیدا کردیم قرار می دهیم که دلیل آن این است که اگر کمی خطا داشتیم و مقداری از سطر های بالای پلاک را نمیخواندیم، با اینکار این مشکل حل میشود. و اینکه نسبت طول پلاک به طول نوار آبی تقریبا برابر 14 است بنابراین طول مستطیلی که از عکس جدا می کنیم را 14 برابر طول نوار آبی در نظر می گیریم. اینکه ای نسبت و اینکه چه مقدار نقطه ی گوشه ی مستطیل را به بالا و چپ انتقال بدهیم به صورت تجربی به دست آمده است.