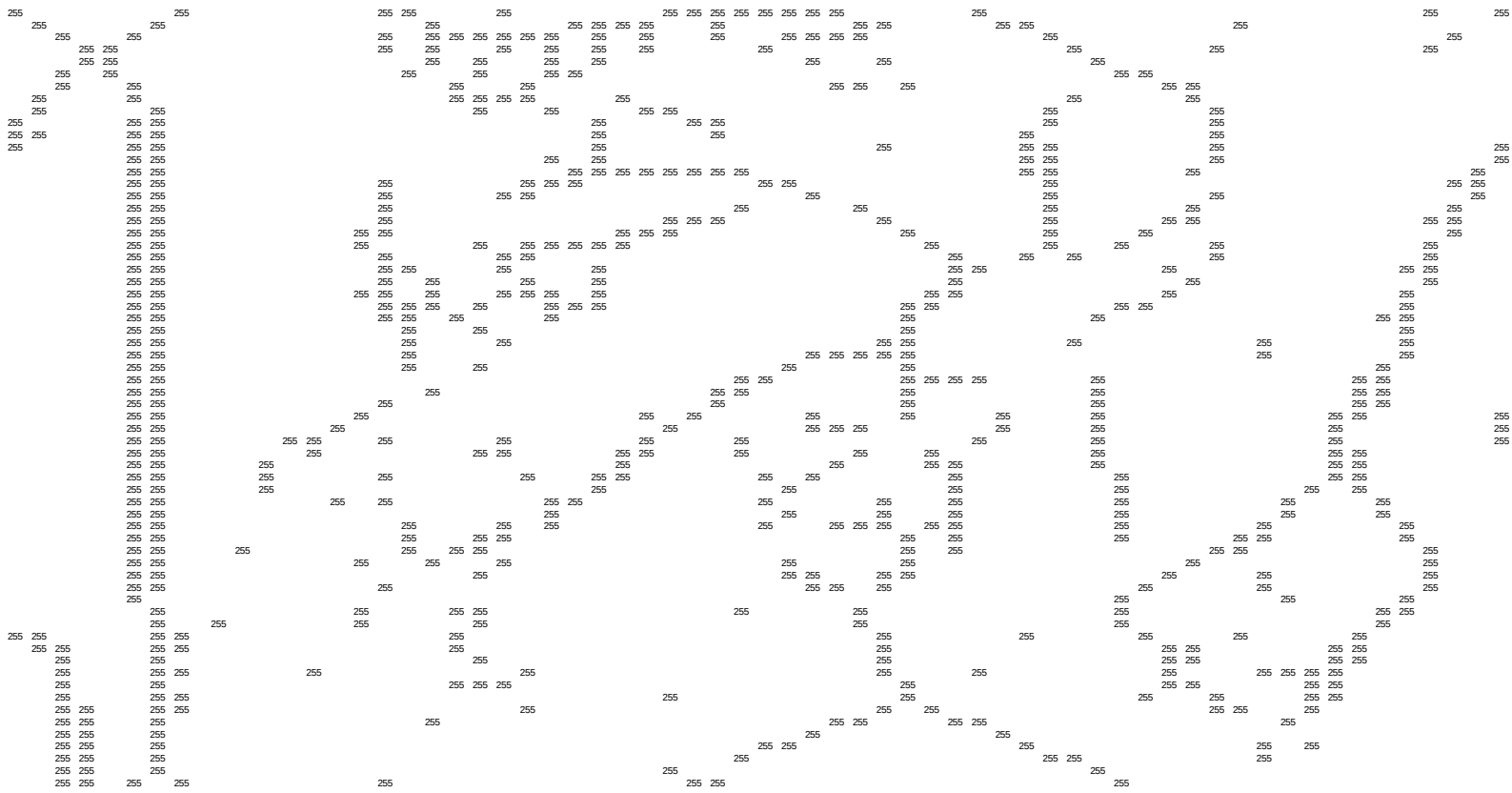
Computer Vision hw7 report

r07921010 余奇安

1.Thinning operator
Algorithm:
step1: mark-interior/border-pixel operator
step2: pair relationship operator
step3: marked-pixel connected shrink operator
Repeat step 1, 2, 3 until the output never change.

The input image is gray scale Lena.jpg downside from 512*512 to 64*64 and binary at 128.
Result:



Code:
Follow the slide to implement h, I, f, function for interior/border operator

```
def h(c,d):
    if c==d:
        return 255
    elif c!=d:
        return 0
def I(a,i):
    if a=="i":
        return 1
    else:
        return 0
def f(c):
    if c==0:
        return "b"
    elif c!=0:
        return "i"
```

Interior/border operator work as IB function

```
def IB(img):#thinning
    #row,col=img.shape
    row=len(img)
    col=len(img)
    new=[[" "for i in range(row+2)] for i in range(col+2)]
    res=[[" "for i in range(row+2)] for i in range(col+2)]
    fres=[[" "for i in range(row)] for i in range(col)]
    for i in range(row):
        for j in range(col):
            if img[i][j]==255:
                new[i+1][j+1]=img[i][j]
    for i in range(1,row+1):
        for j in range(1,col+1):
            if new[i][j]==255:
                a0=new[i][j]
                a1=h(a0,new[i][j+1])
                a2=h(a1,new[i-1][j])
                a3=h(a2,new[i][j-1])
                a4=h(a3,new[i+1][j])

                res[i][j]=f(a4)


    for i in range(row):
        for j in range(col):
            fres[i][j]=res[i+1][j+1]
```

After define border and interior point, we should do the pair relation operator

```
def PairRelation(i,j,img):
    if I(img[i][j+1],"i")+I(img[i-1][j],"i")+I(img[i][j-1],"i")+I(img[i+1][j],"i")<1 or img[i][j]!="b":
        #print(I(img[i][j+1],"i")+I(img[i-1][j],"i")+I(img[i][j-1],"i")+I(img[i+1][j],"i"))
        return "q"
    elif I(img[i][j+1],"i")+I(img[i-1][j],"i")+I(img[i][j-1],"i")+I(img[i+1][j],"i")>=1 and img[i][j]=="b":
        return "p"
def PR(img):
    row=len(img)
    col=len(img)
    #print(row,col)
    new=[[" "for i in range(row+2)] for i in range(col+2)]
    res=[[" "for i in range(row+2)] for i in range(col+2)]
    fres=[[" "for i in range(row)] for i in range(col)]
    for i in range(row):
        for j in range(col):
            if img[i][j]=="b" or img[i][j]=="i":
                new[i+1][j+1]=img[i][j]

    for i in range(1,row+1):
        for j in range(1,col+1):
            if new[i][j]=="b" or new[i][j]=="i":
                #print(i,j)
                res[i][j]=PairRelation(i,j,new)
    for i in range(row):
        for j in range(col):
            fres[i][j]=res[i+1][j+1]
    return fres
```

the pixel is removable if the pixel is marked as “p” and the value of f function return is exactly 1

```
def removeable(i,j,new):
    def h(b,c,d,e):
        if b==c and ((d != b) or (e != b)):
            return "q"
        elif b==c and ((d==b) and (e==b)):
            return "r"
        elif b != c:
            return "s"
    def f(a1,a2,a3,a4):
        if a1==a2 and a1==a3 and a1==a4 and a1=="r":
            return 5
        else:
            l=[a1,a2,a3,a4]
            #print(l)
            count=l.count("q")
            return count
    l=[]
    l.append(h(new[i][j],new[i][j+1],new[i-1][j+1],new[i-1][j]))
    l.append(h(new[i][j],new[i-1][j],new[i-1][j-1],new[i][j-1]))
    l.append(h(new[i][j],new[i][j-1],new[i+1][j-1],new[i+1][j]))
    l.append(h(new[i][j],new[i+1][j],new[i+1][j+1],new[i][j+1]))
    res=f(l[0],l[1],l[2],l[3])
    if res==1:
        return True
```

The thinning operator

```
def thinning(img):
    row=len(img)
    col=len(img)
    #print(row,col,"fuck")
    new=[[" "for i in range(row+2)] for i in range(col+2)]
    res=[[" "for i in range(row+2)] for i in range(col+2)]
    fres=[[" "for i in range(row)] for i in range(col)]

    ib=IB(img)
    pr=PR(ib)
    for i in range(row):
        for j in range(col):
            if img[i][j]==255:
                new[i+1][j+1]=img[i][j]
                res[i+1][j+1]=img[i][j]
    for i in range(1,row+1):
        for j in range(1,col+1):
            if removeable(i,j,new) and pr[i-1][j-1]=="p":
                #print(new[i][j],removeable(i,j,new),pr[i-1][j-1])
                res[i][j]=0
    for i in range(row):
        for j in range(col):
            fres[i][j]=res[i+1][j+1]
    return fres
```

Use main function to repeat step 1,2,3 until the output never change

```
def main():
    bi=bin(img)
    ds=downside(bi)#255
    yk=IB(ds)#interior border
    pr=PR(yk)#qp

    check=True
    while(check):
        th=thinning(ds)
        if th==ds:
            check=False
        else:
            ds=th
    new=np.zeros((64,64),dtype=np.int)
    for i in range(64):
        for j in range(64):
            if th[i][j]==255:
                new[i][j]=255

#   np.asarray(th)
    cv2.imwrite("thinning.jpg",new)
    df1=pd.DataFrame(th)
    df1.to_csv("hw7.csv")
```

Discussion:
we can find that there is a line with two pixel width. I first regard it as not reasonable, but after I check the algorithm in slides, this is the correct result. It is because when the line is shrunk to two parallel pixel, they are both border which are marked q(no interior point), so the pixel will not be canceled.