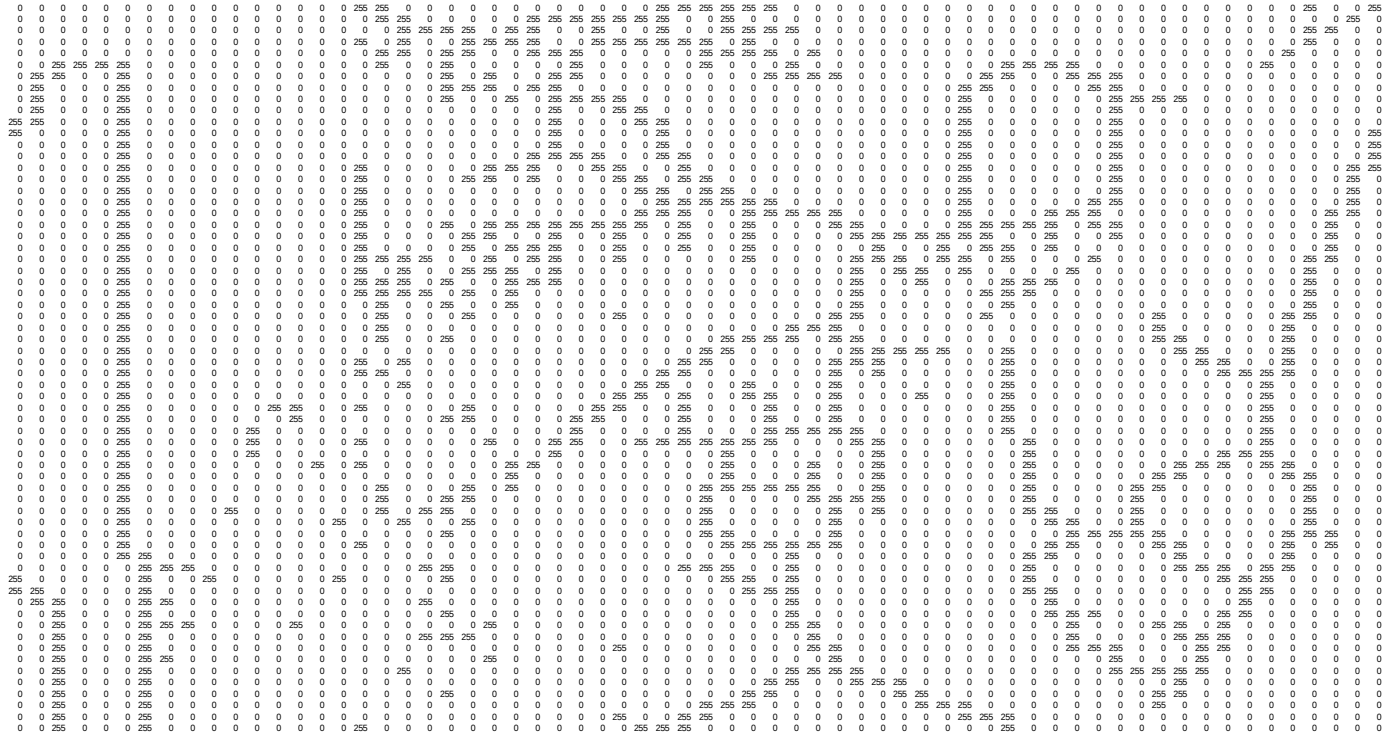
Computer Vision hw7 report

r07921010 余奇安

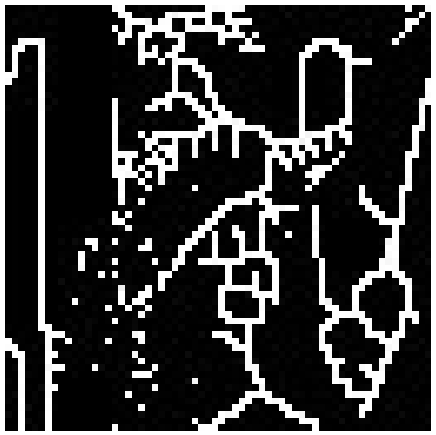
1.Thinning operator
Algorithm:
step1: do the yokoi operation
step2: pair relationship operator
step3: marked-pixel connected shrink operator
Repeat step 1, 2, 3 until the output never change.

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

Result:



Thinning process:



Code:
Downside, binary, yokoi

```
def downside(img):
    row=img.shape[0]
    col=img.shape[1]
    res=np.zeros((row/8,col/8), dtype=np.int)
    for i in range(0,row,8):
        for j in range(0,col,8):
            res[i/8][j/8]=img[i][j]
    return res
def bin(img):
    row,col=img.shape
    res=np.zeros((row,col), dtype=np.int)
    for i in range(row):
        for j in range(col):
            if img[i][j]>127:
                res[i][j]=255
            else:
                res[i][j]=0
    return res
```

```
def yokoi(img):
    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
    row=len(img[0])
    col=len(img[1])
    new=[[" "for i in range(66)] for i in range(66)]
    res=[[" "for i in range(66)] for i in range(66)]
    #new=np.zeros((row+2,col+2),dtype=np.int)
    #res=np.zeros((row+2,col+2),dtype=np.int)
    rsize=np.zeros((row,col),dtype=np.int)
    fres=[[" "for i in range(64)] for i in range(64)]
    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:
                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[i][j]=f(l[0],l[1],l[2],l[3])
    for i in range(row):
        for j in range(col):
            fres[i][j]=res[i+1][j+1]

    return fres
```

PairRelation

```
def PairRelation(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]!=" ":
                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]==1 and(new[i][j+1]==1 or new[i-1][j]==1 or new[i][j-1]==1 or new[i+1][j]==1):
                res[i][j]="p"
            elif new[i][j]!=" ":
                res[i][j]="q"

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

Removeable

```
def removeable(i,j,new):
    def h(b,c,d,e):
        if b==c and ((d != b) or (e != b)):
            return 1
        else:
            return 0
    def f(a1,a2,a3,a4):
        res=0
        if a1==1:
            res+=1
        if a2==1:
            res+=1
        if a3==1:
            res+=1
        if a4==1:
            res+=1
        return res


    a1=h(new[i][j],new[i][j+1],new[i-1][j+1],new[i-1][j])
    a2=h(new[i][j],new[i-1][j],new[i-1][j-1],new[i][j-1])
    a3=h(new[i][j],new[i][j-1],new[i+1][j-1],new[i+1][j])
    a4=h(new[i][j],new[i+1][j],new[i+1][j+1],new[i][j+1])
    res=f(a1,a2,a3,a4)
    if res==1:
        return True
```

Thinning operator

```
def thinning(img):
    row=len(img)
    col=len(img)
    #print(row,col,"fuck")
    new=[[0  for i in range(row+2)] for i in range(col+2)]
    res=[[0 for i in range(row+2)] for i in range(col+2)]
    fres=[[0 for i in range(row)] for i in range(col)]

    ib=yokoi(img)
    pr=PairRelation(ib)
    #df=pd.DataFrame(ib)
    #df.to_csv("yk3'.csv")
    #df=pd.DataFrame(pr)
    #df.to_csv("pr3'.csv")
    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 pr[i-1][j-1]=="p" and removeable(i,j,new):
                new[i][j]=0
    for i in range(row):
        for j in range(col):
            if new[i+1][j+1]==255:
                fres[i][j]=new[i+1][j+1]
    return fres
```

Revised Algorithm:
Check every yokoi label 1(edge) whether there is any other 1(edge) in its neighbor.
If there is label it as p(possible to shrink), else label it as q. It will fix the problem of two straight lines without interior points which can not be canceled in last vision algorithm.

For every p label in image, check whether it is shrinkable pixel which will not result in separate original image to two part from left to right up to down.

Repeat the steps until the image will not change.