Computer Vision
HW9

R10921104 王怡堯

December 13, 2021

1 Primary Procedures

1.1 Robert's Operator

```
def robert(lena,theshold):
    res=[[255]*len(lena[0]) for i in range(len(lena))]
    lena=expan(lena)
    for i in range(len(lena)-2):
        for j in range(len(lena[0])-2):
            grad=((lena[i+1][j+1]-lena[i+2][j+2])**2+(lena[i+2][j+1]-lena[i+1][j+2])**2)**0.5
            if grad>=theshold:
                res[i][j]=0
    return res
```

1.2 Prewitt's Edge Detector

```
def prewitt(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=lena[i+2][j]+lena[i+2][j+1]+lena[i+2][j+2]-lena[i][j]-lena[i][j+1]-lena[i][j+2]
            p2=lena[i][j+2]+lena[i+1][j+2]+lena[i+2][j+2]-lena[i][j]-lena[i+1][j]-lena[i+2][j]
            grad=(p1**2+p2**2)**0.5
            if grad<theshold:
                res[i][j]=255
    return res
```

1.3 Sobel's Edge Detector

```
def sobel(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=lena[i+2][j]+2*lena[i+2][j+1]+lena[i+2][j+2]-lena[i][j]-2*lena[i][j+1]-lena[i][j+2]
            p2=lena[i][j+2]+2*lena[i+1][j+2]+lena[i+2][j+2]-lena[i][j]-2*lena[i+1][j]-lena[i+2][j]
            grad=(p1**2+p2**2)**0.5
            if grad<theshold:
                res[i][j]=255
    return res
```

1.4 Frei and Chen's Gradient Operator

```
def frei(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=lena[i+2][j]+(2**0.5)*lena[i+2][j+1]+lena[i+2][j+2]-lena[i][j]-(2**0.5)*lena[i][j+1]-lena[i][j+2]
            p2=lena[i][j+2]+(2**0.5)*lena[i+1][j+2]+lena[i+2][j+2]-lena[i][j]-(2**0.5)*lena[i+1][j]-lena[i+2][j]
            grad=(p1**2+p2**2)**0.5
            if grad<theshold:
                res[i][j]=255
    return res
```

1.5 Kirsch's Compass Operator

```
def kirsch(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=5*(lena[i][j+2]+lena[i+1][j+2]+lena[i+2][j+2])
            p1+=3*(-lena[i][j]-lena[i][j+1]-lena[i+1][j]-lena[i+2][j]-lena[i+2][j+1])
            p2=5*(lena[i][j]+lena[i+1][j]+lena[i+2][j])
            p2+=3*(-lena[i][j+2]-lena[i][j+1]-lena[i+1][j+2]-lena[i+2][j+2]-lena[i+2][j+1])
            p3=5*(lena[i][j]+lena[i][j+1]+lena[i][j+2])
            p3+=3*(-lena[i+1][j]-lena[i+2][j]-lena[i+1][j+2]-lena[i+2][j+2]-lena[i+2][j+1])
            p4=5*(lena[i+2][j]+lena[i+2][j+1]+lena[i+2][j+2])
            p4+=3*(-lena[i][j]-lena[i+1][j]-lena[i][j+2]-lena[i+1][j+2]-lena[i][j+1])
            p5=5*(lena[i][j]+lena[i+1][j]+lena[i][j+1])
            p5+=3*(-lena[i][j+2]-lena[i+1][j+2]-lena[i+2][j+2]-lena[i+2][j+1]-lena[i+2][j])
            p6=5*(lena[i][j+2]+lena[i+1][j+2]+lena[i][j+1])
            p6+=3*(-lena[i][j]-lena[i+1][j]-lena[i+2][j]-lena[i+2][j+1]-lena[i+2][j+2])
            p7=5*(lena[i+2][j+2]+lena[i+1][j+2]+lena[i+2][j+1])
            p7+=3*(-lena[i][j]-lena[i][j+1]-lena[i+1][j]-lena[i+2][j]-lena[i][j+2])
            p8=5*(lena[i+2][j]+lena[i+1][j]+lena[i+2][j+1])
            p8+=3*(-lena[i][j]-lena[i][j+1]-lena[i][j+2]-lena[i+1][j+2]-lena[i+2][j+2])
            grad=max(p1,p2,p3,p4,p5,p6,p7,p8)
            if grad<theshold:
                res[i][j]=255
    return res
```

1.6 Robinson's Compass Operator

```
def robinson(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=(lena[i][j+2]+2*lena[i+1][j+2]+lena[i+2][j+2])
            p1+=(-lena[i][j]-2*lena[i+1][j]-lena[i+2][j])
            p1=abs(p1)
            p3=(lena[i][j]+2*lena[i][j+1]+lena[i][j+2])
            p3+=(-lena[i+2][j]-lena[i+2][j+2]-2*lena[i+2][j+1])
            p3=abs(p3)
            p5=2*lena[i][j]+lena[i+1][j]+lena[i][j+1]
            p5+=(-lena[i+1][j+2]-2*lena[i+2][j+2]-lena[i+2][j+1])
            p5=abs(p5)
            p6=2*lena[i][j+2]+lena[i+1][j+2]+lena[i][j+1]
            p6+=(-lena[i+1][j]-2*lena[i+2][j]-lena[i+2][j+1])
            p6=abs(p6)
            grad=max(p1,p3,p5,p6)
            if grad<theshold:
                res[i][j]=255
    return res
```

1.7 Nevatia-Babu 5x5 Operator

```
def badu(lena,theshold):
    m=len(lena)
    n=len(lena[0])
    lena=expan(lena)
    lena=expan(lena)
    res=[[0]*(n) for i in range(m)]
    for i in range(m):
        for j in range(n):
            p1=100*(lena[i][j]+lena[i][j+1]+lena[i][j+2]+lena[i][j+3]+lena[i][j+4])
            p1+=100*(lena[i+1][j]+lena[i+1][j+1]+lena[i+1][j+2]+lena[i+1][j+3]+lena[i+1][j+4])
            p1-=100*(lena[i+3][j]+lena[i+3][j+1]+lena[i+3][j+2]+lena[i+3][j+3]+lena[i+3][j+4])
            p1-=100*(lena[i+4][j]+lena[i+4][j+1]+lena[i+4][j+2]+lena[i+4][j+3]+lena[i+4][j+4])

            p2=100*(lena[i][j+4]+lena[i+1][j+4]+lena[i+2][j+4]+lena[i+3][j+4]+lena[i+4][j+4])
            p2+=100*(lena[i][j+3]+lena[i+1][j+3]+lena[i+2][j+3]+lena[i+3][j+3]+lena[i+4][j+3])
            p2-=100*(lena[i][j+1]+lena[i+1][j+1]+lena[i+2][j+1]+lena[i+3][j+1]+lena[i+4][j+1])
            p2-=100*(lena[i][j]+lena[i+1][j]+lena[i+2][j]+lena[i+3][j]+lena[i+4][j])

            p3=100*(lena[i][j]+lena[i][j+1]+lena[i][j+2]+lena[i][j+3]+lena[i][j+4])
            p3+=100*(lena[i+1][j]+lena[i+1][j+1]+lena[i+1][j+2]+lena[i+2][j])
            p3+=78*lena[i+1][j+3]+92*lena[i+2][j+1]+32*lena[i+3][j]
            p3-=100*(lena[i+4][j]+lena[i+4][j+1]+lena[i+4][j+2]+lena[i+4][j+3]+lena[i+4][j+4])
            p3-=100*(lena[i+3][j+2]+lena[i+3][j+3]+lena[i+3][j+4]+lena[i+2][j+4])
            p3-=78*lena[i+3][j+1]+92*lena[i+2][j+3]+32*lena[i+1][j+4]


            p4=100*(lena[i][j]+lena[i+1][j]+lena[i+2][j]+lena[i+3][j]+lena[i+4][j])
            p4+=100*(lena[i][j+1]+lena[i+1][j+1]+lena[i+1][j+2]+lena[i+2][j+2])
            p4+=78*lena[i+3][j+1]+92*lena[i+1][j+2]+32*lena[i][j+3]
            p4-=100*(lena[i][j+4]+lena[i+1][j+4]+lena[i+2][j+4]+lena[i+3][j+4]+lena[i+4][j+4])
            p4-=100*(lena[i+2][j+3]+lena[i+3][j+3]+lena[i+4][j+3]+lena[i+4][j+2])
            p4-=78*lena[i+1][j+3]+92*lena[i+3][j+2]+32*lena[i+4][j+1]

            p5=100*(lena[i][j+4]+lena[i+1][j+4]+lena[i+2][j+4]+lena[i+3][j+4]+lena[i+4][j+4])
            p5+=100*(lena[i][j+3]+lena[i+1][j+3]+lena[i+2][j+3]+lena[i][j+2])
            p5+=78*lena[i+3][j+3]+92*lena[i+1][j+2]+32*lena[i][j+1]
            p5-=100*(lena[i][j]+lena[i+1][j]+lena[i+2][j]+lena[i+3][j]+lena[i+4][j])
            p5-=100*(lena[i+2][j+1]+lena[i+3][j+1]+lena[i+4][j+1]+lena[i+4][j+2])
            p5-=78*lena[i+1][j+1]+92*lena[i+3][j+2]+32*lena[i+4][j+3]

            p6=100*(lena[i][j]+lena[i][j+1]+lena[i][j+2]+lena[i][j+3]+lena[i][j+4])
            p6+=100*(lena[i+1][j+2]+lena[i+1][j+3]+lena[i+1][j+4]+lena[i+2][j+4])
            p6+=78*lena[i+1][j+1]+92*lena[i+2][j+3]+32*lena[i+3][j+4]
            p6-=100*(lena[i+4][j]+lena[i+4][j+1]+lena[i+4][j+2]+lena[i+4][j+3]+lena[i+4][j+4])
            p6-=100*(lena[i+3][j]+lena[i+3][j+1]+lena[i+3][j+2]+lena[i+2][j])
            p6-=78*lena[i+3][j+3]+92*lena[i+2][j+1]+32*lena[i+1][j]



            grad=max(p1,p2,p3,p4,p5,p6)
            if grad<theshold:
                res[i][j]=255
    return res
```

2 Result

thresholds:12

thresholds:24

thresholds:38

thresholds:30

thresholds:135

thresholds:43

thresholds:12500