convolution-opg

September 27, 2024

1 Forståelse af Convolution

Find mindst 3 digitale billeder, gerne i din egen samling.

Gerne med varierende kontrast i billederne.

Konverter dem alle til sorthvid med eksempelprogrammet.

Afprøv de to kernels i programmet.

Design implementer og test : * en lodret kantdetektering. * en vandret kantdetektering * prøv at lave og teste en 36 punkts smudge. * Kan du lave en kantdetektering som giver en tykkere kantlinje.

[1]:
```
from Convolutions import processImage, convolve2D
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
# process images
#image1 = processImage('misha.JPG')
image3 = processImage('skov.jpg')
#image3 = processImage('ski.jpg')

plt.imshow(image3, cmap='Greys_r', interpolation='nearest')  #besværgelser for␣
 ↪at få imshow til ikke at bruge farver
plt.axis('off')  # Hide axes
plt.show()
```

[3]:
```
kernel_3x3 = np.array([[-1, -1, -1],
                       [-1,  8, -1],
                       [-1, -1, -1]])
```

[4]:
```
kernel_5x5 = np.array([[-1, -1, -1, -1, -1],
                       [-1,  1,  2,  1, -1],
                       [-1,  2,  4,  2, -1],
                       [-1,  1,  2,  1, -1],
                       [-1, -1, -1, -1, -1]])
```

[5]:
```
kernel_7x7 = np.array([[-1, -1, -1, -1, -1, -1, -1],
                       [-1, -1,  1,  1,  1, -1, -1],
                       [-1,  1,  1,  2,  1,  1, -1],
                       [-1,  1,  2,  4,  2,  1, -1],
                       [-1,  1,  1,  2,  1,  1, -1],
                       [-1, -1,  1,  1,  1, -1, -1],
                       [-1, -1, -1, -1, -1, -1, -1]])
```

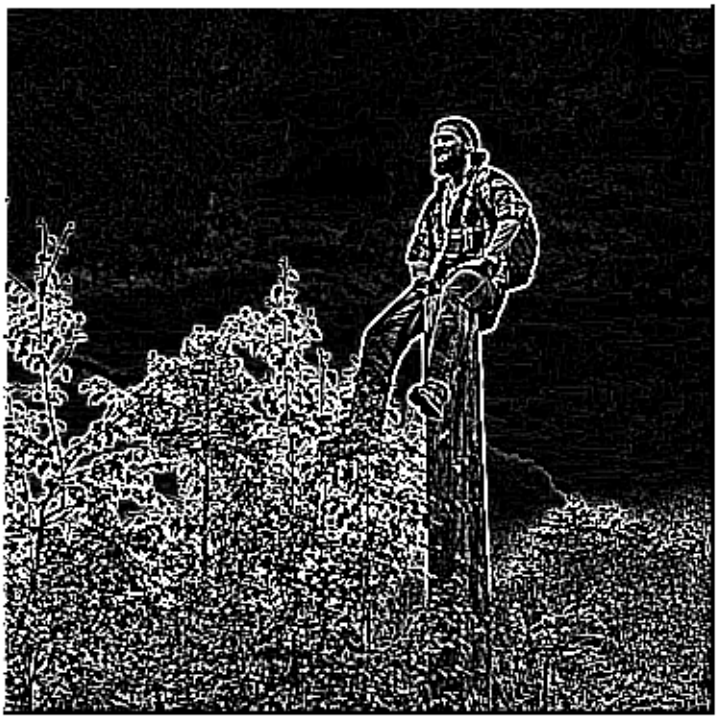
[6]:
```
def conv_show(image, kernel, padding=2):
    output = convolve2D(image, kernel, padding=padding, strides=1)
```

```
    plt.imshow(output, cmap='gray', vmin=0, vmax=127) #cmap='Greys_r',␣
↪interpolation='nearest')  #besværgelser for at få imshow til ikke at bruge␣
↪farver
    plt.axis('off')  # Hide axes
    plt.show()
```

[7]:
```
conv_show(image3, kernel_3x3)
```

[8]:
```
conv_show(image3, kernel_5x5)
```

[9]:
```
conv_show(image3, kernel_7x7, padding=3)
```



[10]:
```
def conv_l2_show(image, kernel, padding=2):
    output = convolve2D(image, kernel, padding=padding, strides=1)
    output = convolve2D(output, kernel, padding=padding, strides=1)

    plt.imshow(output, cmap='gray', vmin=0, vmax=127) #cmap='Greys_r',␣
 ↪interpolation='nearest')  #besværgelser for at få imshow til ikke at bruge␣
 ↪farver
    plt.axis('off')  # Hide axes
    plt.show()

def conv_l3_show(image, kernel, padding=2):
    output = convolve2D(image, kernel, padding=padding, strides=1)
    output = convolve2D(output, kernel, padding=padding, strides=1)
    output = convolve2D(output, kernel, padding=padding, strides=1)

    plt.imshow(output, cmap='gray', vmin=0, vmax=127) #cmap='Greys_r',␣
 ↪interpolation='nearest')  #besværgelser for at få imshow til ikke at bruge␣
 ↪farver
    plt.axis('off')  # Hide axes
    plt.show()
```

[11]:
```
# first attempt at vertical lines
kernel_vertical_3x3 = np.array([[-1, 0, 1],
                                [-1, 0, 1],
                                [-1, 0, 1]])

kernel_horisontal_3x3 = kernel_vertical_3x3.T
```

[12]:
```
# first attempt at vertical lines
kernel_vertical_5x5 = np.array([[-.5, -1, 0, 1, .5],
                                [-.5, -1, 0, 1, .5],
                                [-.5, -1, 0, 1, .5],
                                [-.5, -1, 0, 1, .5],
                                [-.5, -1, 0, 1, .5],])

kernel_horisontal_5x5 = kernel_vertical_5x5.T
```

[13]:
```
conv_show(image3, kernel_vertical_3x3)
```

[14]:
```
conv_show(image3, kernel_vertical_5x5)
```

[15]:
```
conv_show(image3, kernel_horisontal_3x3)
```

[16]:
```
conv_show(image3, kernel_horisontal_5x5)
```

[17]:
```
conv_l2_show(image3, kernel_vertical_3x3)
```

[18]:
```
conv_l2_show(image3, kernel_vertical_5x5)
```

[19]:
```
conv_l2_show(image3, kernel_horisontal_3x3)
```

[20]:
```
conv_l2_show(image3, kernel_horisontal_5x5)
```

[21]:
```
conv_l3_show(image3, kernel_vertical_3x3)
```

[22]:
```
conv_l3_show(image3, kernel_horisontal_3x3)
```