Volume 1 | Issue 1
2021/02/27

Blur
Convolution
Edge Detection

Blur

Spatial Filtering

It's neighborhood operation
The value of a pixel is based on pixels in a small range (neighbor)

$$\begin{bmatrix} f(x-1,\,y-1) & f(x,\,y-1) & f(x+1,\,y-1) \\ f(x-1,\,y) & f(x,y) & f(x+1,y) \\ f(x-1,y+1) & f(x,y+1) & f(x+1,\,y+1) \end{bmatrix}$$

The value of pixel f(x,y) is effected by neighbors

Kernel – filter mask

It's matrix $K^{m\,x\,n}$

- m : the number of rows
- n : the number of column
- m & n is often odd number

Ex: average filter

$$\frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{9}\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Average filter

Each pixel will get the average value of pixel neighbors
Given image I and kernel K^{3*3}

$$I = \begin{bmatrix} 10 & 11 & 10 & 0 & 0 & 1 \\ 9 & 10 & 11 & 1 & 0 & 1 \\ 10 & 9 & 10 & 0 & 2 & 1 \\ 11 & 10 & 9 & 10 & 9 & 11 \end{bmatrix} \quad K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Calculate each pixel with formula:

$$I'(x,y) = \sum_{i=-m/2}^{i=m/2} \sum_{j=-n/2}^{j=n/2} I(x+i, y+j) * \mathrm{K}(i,j)$$

When apply kernel K we will get new images as below
I'(1,1) = 1/9 * (10 * 1 + 11 * 1 + 10 * 1 + 9 * 1 + 10 * 1+ 11 * 1 + 10 * 1 + 9 * 1 + 10 * 1) = 10

$$I' = \begin{bmatrix} x & x & x & x & x & x \\ x & 10 & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix}$$

Mean filter

Each pixel will get the value of mean pixel neighbors
Given image I

$$I = \begin{bmatrix} 10 & 11 & 10 & 0 & 0 & 1 \\ 9 & 10 & 11 & 1 & 0 & 1 \\ 10 & 9 & 10 & 0 & 2 & 1 \\ 11 & 10 & 9 & 10 & 9 & 11 \end{bmatrix}$$

I'(1,1) = 10 , 11, 10, 9, 10, 11, 10, 9, 10 →ASC SORT →
9,9,10,10,10,10,10,11,11 = 10

Gauss filter

The pixel which near center will get more effect, the pixel which further center will get less effect
Gauss formular : σ : standard deviation

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Calculate Gauss kernel
Given m: the number of kernel rows, n: the number of kernel column

$$K(x,y) = G(x,y) \,/\, \{ x \in [-m/2; m/2], y \in [-n/2; n/2]$$

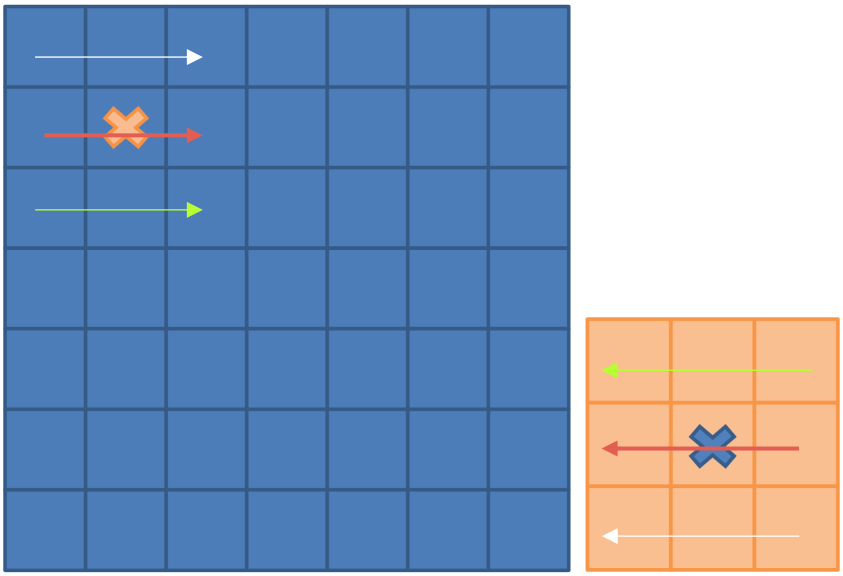
$$G - kernel = \begin{bmatrix} 0.059 & 0.095 & 0.059 \\ 0.095 & 0.15 & 0.095 \\ 0.059 & 0.095 & 0.059 \end{bmatrix}$$

Convolution

Formular

Given image I and kernel K^{m*n}

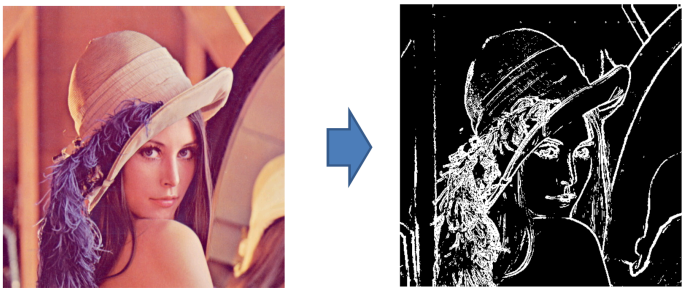
$$I'(x,y) = \sum_{i=-m/2}^{i=m/2} \sum_{j=-n/2}^{j=n/2} I(x - i, y - j) * K(i,j)$$



Pixel(1,1) = I(white arrow) * K(white arrow) + I(green arrow) * K(green arrow) + I(red arrow) * K(red arrow)

The order of multiply follows on direction of arrow

Edge Detection



Method: Gradient image

$$\nabla f = \left[\frac{\partial f(x,y)}{\partial x} \quad \frac{\partial f(x,y)}{\partial y}\right]$$

$$\frac{\partial f(x,y)}{\partial x} = \frac{I(x + dx, y) - I(x,y)}{dx}$$

$$\frac{\partial f(x,y)}{\partial y} = \frac{I(x, y + dy) - I(x,y)}{dy}$$

$$|\nabla f| = \sqrt{\frac{\partial f(x,y)^2}{\partial x} + \frac{\partial f(x,y)^2}{\partial y}}$$

After finished calculate gradient, you can use threshold to make edge more visible

$$I'(x,y) = \begin{cases} 0 \text{ if } |\nabla f| < \text{threshold} \\ 255 \text{ if } |\nabla f| >= \text{threshold} \end{cases}$$

Method: Sobel kernel

Instead of do gradient you can convolute image with kernel

$$hx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$hy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

⊗ : convolution

```
function sobel(A : as two dimensional image array)
       Gx = [-1 0 1; -2 0 2; -1 0 1]
       Gy = [-1 -2 -1; 0 0 0; 1 2 1]

       rows = size(A, 1)
       columns = size(A, 2)
       mag = zeros(A)

       for i=1:rows-2
              for j=1:columns-2
                     S1 = sum(sum(Gx ⊗ A(i:i+2,j:j+2)))
                     S2 = sum(sum(Gy ⊗ A(i:i+2,j:j+2)))
```

```
                    mag(i+1, j+1) = sqrt(S1.^2+S2.^2)
                end for
        end for

        threshold = 70 %varies for application [0 255]
        output_image = max(mag, threshold)
        output_image(output_image == round(threshold)) = 0;
        return output_image
end function
```

Method: Prewitt

Instead of do gradient you can convolute image with kernel

$$hx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$hy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$