|  |  |
| --- | --- |
| 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)  The value of pixel f(x,y)is effected by neighbors  Kernel – filter mask  It’s matrix Km x n   * m : the number of rows * n : the number of column * m & n is often odd number   Ex: average filter    Average filter  Each pixel will get the average value of pixel neighbors  Given image I and kernel K3 \* 3    Calculate each pixel with formula:  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    Mean filter  Each pixel will get the value of mean pixel neighbors  Given image I  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 : : standard deviation  Calculate Gauss kernel  Given m: the number of kernel rows, n: the number of kernel column |
| **Convolution** | Formular  Given image I and kernel Km\*n    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  After finished calculate gradient, you can use threshold to make edge more visible  Method: Sobel kernel  Instead of do gradient you can convolute image with kernel    : 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 |
|  |  |