Computer Vision I \_2018

Homework assignment #9

R07522717機械所製造組碩一 林温雅

Part1（此次作業僅one part）

Description:

Write programs to generate the following gradient magnitude images and choose proper thresholds to get the binary edge images:

* Roberts operator
* Prewitt edge detector
* Sobel edge detector
* Frei and Chen gradient operator
* Kirsch compass operator
* Robinson compass operator
* Nevatia-Babu 5X5 operator

Algorithm:

根據不同的edge detector所使用的kernel，對影像做convolution。

在roberts、perwitt、sobel、frei & chen中，將不同kernel的convolution值進行平方後相加開根號處理，設為新影像的值。而在kirsch、robinson、nevatia-babu中，將不同kernel的convolution值做比較，挑選max值設為新影像的值。

最後自行挑選合適的threshold，對影像做reverse thresholding(黑白轉換一下方便看)。

Parameters:

i,j #迴圈計數用參數

original\_img #原始圖檔

ker\_XX #各式不同kernel

list\_XX #儲存kernel用的list

rows, cols #圖檔的長與寬

temp\_img #擴大後的圖檔，為了convolution邊界所製作

temp #在迴圈中擷取影像中和kernel一樣大的矩陣，以方便計算

new\_img #用來接收新data的輸出圖檔

max #用來計算最大值的參數

Principal code fragment:

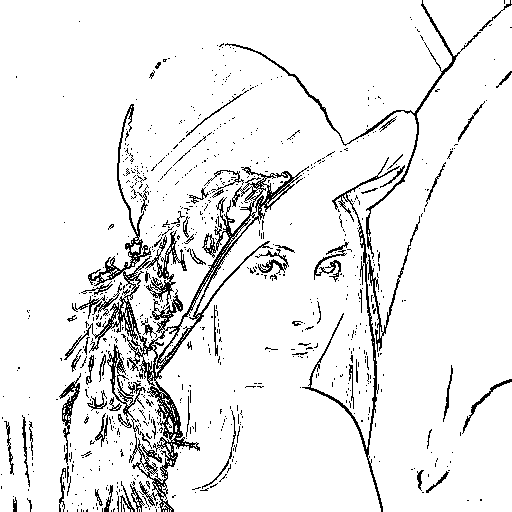
*def* roberts(*img*):  
 ker\_r1 = np.array([[-1, 0],[0, 1]])  
 ker\_r2 = np.array([[0, -1],[1, 0]])  
 rows, cols = *img*.shape  
 # for center在左上角  
 temp\_img = cv2.copyMakeBorder(src=*img*, top=0, bottom=1, left=0, right=1, borderType=cv2.BORDER\_REPLICATE)  
 new\_img = *img*.copy().astype(float)  
 *for* i *in* range(rows):  
 *for* j *in* range(cols):  
 temp = temp\_img[i:i+2, j:j+2]  
  
 new\_img[i,j] = np.sqrt(np.sum(np.multiply(ker\_r1, temp))\*\*2 + np.sum(np.multiply(ker\_r2, temp))\*\*2)  
 #new\_img[i,j] = np.abs(np.sum(np.multiply(ker\_r1, temp))) + np.abs(np.sum(np.multiply(ker\_r2, temp)))  
  
 *return* new\_img

*def* krisch(*img*):  
 ker\_k0 = np.array([[-3,-3,5], [-3,0,5], [-3,-3,5]])  
 ker\_k1 = np.array([[-3,5,5], [-3,0,5], [-3,-3,-3]])  
 ker\_k2 = np.array([[5,5,5], [-3,0,-3], [-3,-3,-3]])  
 ker\_k3 = np.array([[5,5,-3], [5,0,-3], [-3,-3,-3]])  
 ker\_k4 = np.array([[5,-3,-3], [5,0,-3], [5,-3,-3]])  
 ker\_k5 = np.array([[-3,-3,-3], [5,0,-3], [5,5,-3]])  
 ker\_k6 = np.array([[-3,-3,-3], [-3,0,-3], [5,5,5]])  
 ker\_k7 = np.array([[-3,-3,-3], [-3,0,5], [-3,5,5]])  
 list\_kn = [ker\_k0, ker\_k1, ker\_k2, ker\_k3, ker\_k4, ker\_k5, ker\_k6, ker\_k7]  
 rows, cols = *img*.shape  
 temp\_img = cv2.copyMakeBorder(src=*img*, top=1, bottom=1, left=1, right=1, borderType=cv2.BORDER\_REPLICATE)  
 new\_img = *img*.copy().astype(float)  
 *for* i *in* range(rows):  
 *for* j *in* range(cols):  
 temp = temp\_img[i:i+3, j:j+3]  
 max=0 # 初始化max值  
 *for* ker *in* list\_kn:  
 temp\_sum = np.sum(ker \* temp)  
 *if* temp\_sum > max:  
 max = temp\_sum  
 new\_img[i, j] = max  
  
 *return* new\_img

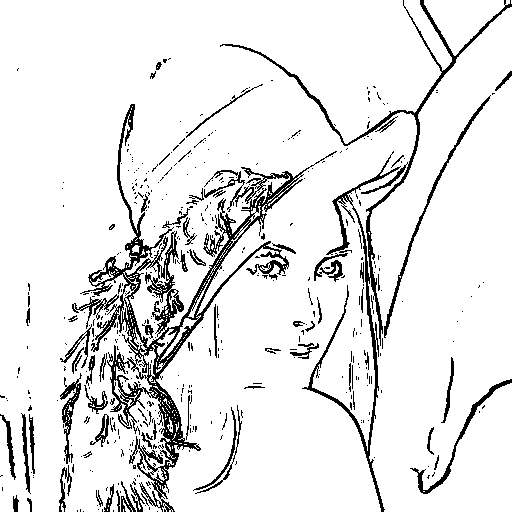
*def* reverse\_thresholding(*img*, *threshold*=128):  
 new\_img = np.empty(*img*.shape)  
 new\_img.fill(255)  
 mask = *img* >= *threshold* new\_img[mask] = 0  
 *return* new\_img

Resulting Image:

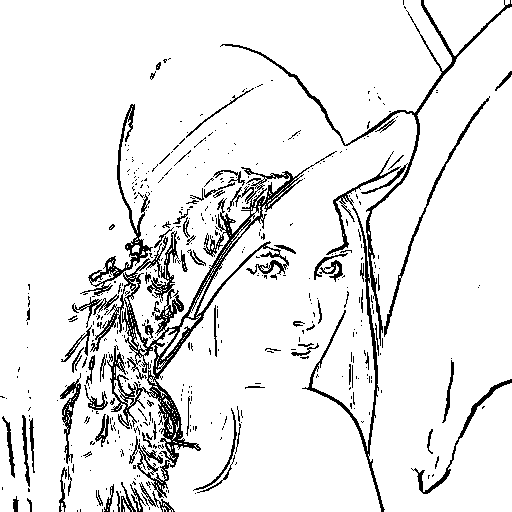
roberts\_30



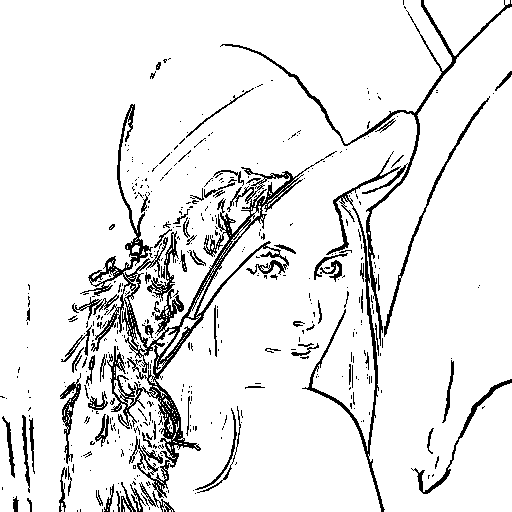
perwitt\_90



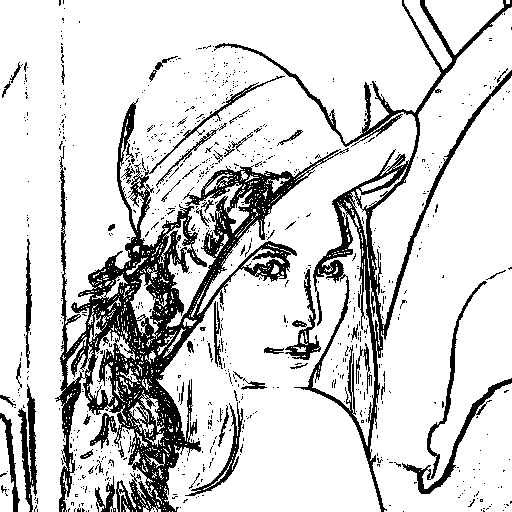
sobel\_130

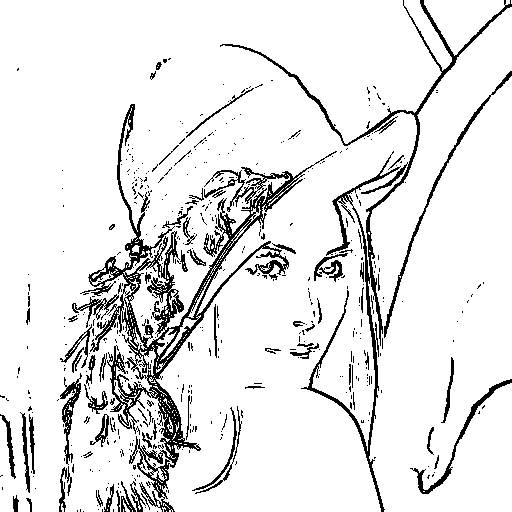


frei\_chen\_110



krisch\_230



robinson\_120

nevatia\_babu\_30000

