**Computer Vision HW9: General Edge Detection**

**R10741015 鄭傑鴻**

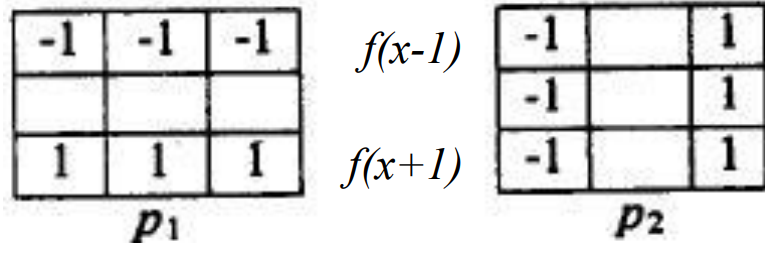
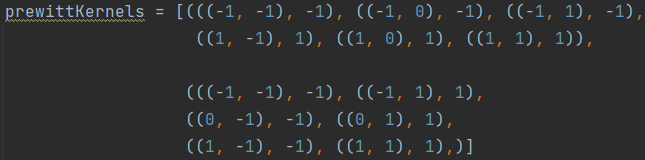
**Nov. 21, 2022**

**Implement the following edge detectors with thresholds:**

* **Description**

Since all of the edge detectors requires convolution over kernels, I wrote a general function to derive the results based on the kernel and threshold.

* **Method / Algorithm**

1. Translate the kernels into list of tuples: for example, the kernels for Prewitt Operator will be translated from , to . (Each element consists of the position (e.g. (1, 0)) and the weight (e.g., 1))
2. Pad the input image based on the kernel size

* For kernel size 2, do padding on the right and bottom boarders by 1
* For kernel size 3, do padding on all four borders by 1
* For kernel size 5, do padding on all four borders by 2

1. Iterate over the padded image:
   * For each kernel, derive the corresponding convolution result
   * Record the results in a list
2. Derive the gradient magnitude based on different operators:

* For Roberts, Prewitt, Sobel, Frei & Chen: derive by summing the squares and then take the square root
* For Kirsch, Robinson, Nevatia-Babu: derive by taking the maxima

1. Compare the gradient magnitude with the threshold

* Greater or equal: assign 0 (black pixel)
* Lesser: assign 255 (white pixel)
* **Main Code Segment**

def DetectEdge(self, inPic, kernelLst, thr, method, ksize=3):  
 retPic = np.zeros\_like(inPic)  
 paddedPic = None  
 if ksize==3: paddedPic = cv2.copyMakeBorder(inPic, 1, 1, 1, 1, cv2.BORDER\_REPLICATE)  
 elif ksize==5: paddedPic = cv2.copyMakeBorder(inPic, 2, 2, 2, 2, cv2.BORDER\_REPLICATE)  
 elif ksize==2: paddedPic = cv2.copyMakeBorder(inPic, 0, 1, 0, 1, cv2.BORDER\_REPLICATE)  
 assert paddedPic is not None  
  
 for i in range(inPic.shape[0]):  
 for j in range(inPic.shape[1]):  
 valLst = []  
 for kernel in kernelLst:  
 val = 0  
 if ksize==3: val = self.ApplyKernel(paddedPic, kernel, i+1, j+1)  
 elif ksize==5: val = self.ApplyKernel(paddedPic, kernel, i+2, j+2)  
 elif ksize==2: val = self.ApplyKernel(paddedPic, kernel, i, j)  
 valLst.append(val)  
 if method == 'SQRT':  
 grad = np.sqrt(np.sum(np.power(valLst, 2)))  
 elif method == 'MAX':  
 grad = np.max(valLst)  
 if grad>=thr: retPic[i][j]=0  
 else: retPic[i][j]=255  
 return retPic

* **Results**
  1. Robert's Operator: 12



* 1. Prewitt's Edge Detector: 24



* 1. Sobel's Edge Detector: 38



* 1. Frei and Chen's Gradient Operator: 30



* 1. Kirsch's Compass Operator: 135



* 1. Robinson's Compass Operator: 43



* 1. Nevatia-Babu 5x5 Operator: 12500

