

電腦視覺 作業九


指導老師 傅楸善



學生 蔡宇晴

學號 R08945050

	原圖	Robert
圖 片		
程 式 碼	<pre> 10 def Robert(img,threshold): 11 kernel1 = np.array([[1,0],[0,-1]]) 12 kernel2 = np.array([[0,1],[-1,0]]) 13 h,w = img.shape 14 edge_ = np.zeros((h,w),dtype=int) 15 16 17 for height in range(0,h-1): 18 for weight in range(0,w-1): 19 slice_img = img[height:height+2,weight:weight+2] 20 gx = np.sum(slice_img * kernel1) 21 gy = np.sum(slice_img * kernel2) 22 g = (gx **2 + gy **2) ** 0.5 23 if g >= threshold: 24 edge_[height,weight] = 255 25 else: 26 edge_[height,weight] = 0 27 28 edge_ = np.uint8(edge_) 29 edge_ = 255-edge_ 30 return edge_ </pre>	
描 述	<p>邊緣定位准，但是對噪聲敏感。適用於邊緣明顯且噪聲較少的圖像分割。影像的雜點看起來很多。</p> <p>在邊緣線條上的粗度相較於其他 6 個檢測來說較細。</p>	



	原圖	Prewitt
圖 片		
程 式 碼	<pre> 31 def Prewitt(img,threshold): 32 kernel1 = np.array([[-1,0,1],[-1,0,1],[-1,0,1]]) 33 kernel2 = np.array([[-1,-1,-1],[0,0,0],[1,1,1]]) 34 h,w = img.shape 35 edge_ = np.zeros((h,w),dtype=int) 36 for height in range(0,h-2): 37 for weight in range(0,w-2): 38 slice_img = img[height:height+3,weight:weight+3] 39 gx = np.sum(slice_img * kernel1) 40 gy = np.sum(slice_img * kernel2) 41 g = (gx **2 + gy **2) ** 0.5 42 if g >= threshold: 43 edge_[height,weight] = 255 44 else: 45 edge_[height,weight] = 0 46 47 edge_ = np.uint8(edge_) 48 edge_ = 255-edge_ 49 return edge_ </pre>	
描 述	<p>對噪聲有抑制作用，抑制噪聲的原理是通過像素平均，但是像素平均相當於對圖像的低通濾波。邊緣檢測結果較 robert 粗。</p>	

	原圖	Sobel
圖 片		
程 式 碼	<pre> 50 def Sobel(img,threshold): 51 kernel1 = np.array([[[-1,0,1],[-2,0,2],[-1,0,1]]) 52 kernel2 = np.array([[1,2,1],[0,0,0],[-1,-2,-1]]) 53 h,w = img.shape 54 edge_ = np.zeros((h,w)) 55 56 for height in range(0,h-2): 57 for weight in range(0,w-2): 58 slice_img = img[height:height+3,weight:weight+3] 59 gx = np.sum(slice_img * kernel1) 60 gy = np.sum(slice_img * kernel2) 61 g= (gx**2+gy**2) **0.5 62 if g >= threshold: 63 edge_[height, weight] = 255 64 else: 65 edge_[height, weight] = 0 66 edge_ = np.uint8(edge_) 67 edge_ = 255-edge_ 68 return edge_ </pre>	
描 述	<p>Sobel 和 Prewitt 都是加權平均，但是 sobel 對於距離不同的像素給予不同的權重。結果圖差不多，但噪聲少了許多。</p>	

	原圖	FreiAndChen
圖 片		
程 式 碼	<pre> 70 def FreiAndChen(img,threshold): 71 sqrt_2 = 2**0.5 72 kernel1 = np.array([[-1,-sqrt_2,-1],[0,0,0],[1,sqrt_2,1]]) 73 kernel2 = np.array([[-1,0,1],[-sqrt_2,0,sqrt_2],[-1,0,1]]) 74 h,w = img.shape 75 edge_ = np.zeros((h,w)) 76 for height in range(0,h-2): 77 for weight in range(0,w-2): 78 slice_img = img[height:height+3,weight:weight+3] 79 gx = np.sum(slice_img * kernel1) 80 gy = np.sum(slice_img * kernel2) 81 g = (gx **2 + gy **2) ** 0.5 82 if g >= threshold:edge_[height,weight] = 255 83 else:edge_[height,weight] = 0 84 edge_ = np.uint8(edge_) 85 edge_ = 255-edge_ 86 return edge_ </pre>	
描 述	<p>Frei-Chen 比 Sobel 看起來更好，因為它對噪聲不太敏感。</p> <p>整體的結果差不多，但是噪聲少了一些。</p>	

	原圖	Kirsch
圖 片		
程 式 碼	<pre> 87 def Kirsch(img, threshold): 88 kernel1 = np.array([[-3, -3, 5], [-3, 0, 5], [-3, -3, 5]]) 89 kernel11 = np.rot90(kernel1, 2) 90 kernel2 = np.array([[-3, 5, 5], [-3, 0, 5], [-3, -3, -3]]) 91 kernel22 = np.rot90(kernel2, 2) 92 kernel3 = np.array([[5, 5, 5], [-3, 0, -3], [-3, -3, -3]]) 93 kernel33 = np.rot90(kernel3, 2) 94 kernel4 = np.array([[5, 5, -3], [5, 0, -3], [-3, -3, -3]]) 95 kernel44 = np.rot90(kernel4, 2) 96 kernel_list = [kernel1, kernel11, kernel2, kernel22, kernel3, kernel33, kernel4, kernel44] 97 h, w = img.shape 98 Gradient = np.zeros((h, w)) 99 100 for height in range(0, h-2): 101 for weight in range(0, w-2): 102 slice_img = img[height:height+3, weight:weight+3] 103 compare_ = [] 104 for _ in kernel_list: 105 gx = np.sum(slice_img * _) 106 compare_.append(gx) 107 max_gx = max(compare_) 108 if max_gx >= threshold: Gradient[height, weight] = 255 109 else: Gradient[height, weight] = 0 110 Gradient = np.uint8(Gradient) 111 Gradient = 255-Gradient 112 return Gradient </pre>	
描 述	<p>相較於 FreiAndChen，噪點減少許多。因為考慮的方向比 FreiAndChen 多了許多方向的檢測。</p>	

	原圖	Robinson
圖 片		
程 式 碼	<pre> 113 def Robinson(img,threshold): 114 h,w = img.shape 115 East = np.array([[-1,0,1],[-2,0,2],[-1,0,1]]) 116 West = np.rot90(East,2) 117 Northeast = np.array([[0,1,2],[-1,0,1],[-2,-1,0]]) 118 Southwest =np.rot90(Northeast,2) 119 North =np.array([[1,2,1],[0,0,0],[-1,-2,-1]]) 120 South =np.rot90(North,2) 121 Northwest =np.array([[2,1,0],[1,0,-1],[0,-1,-2]]) 122 Southeast = np.rot90(Northwest,2) 123 directions = [East ,West ,Northeast ,Southwest,North ,South ,Northwest ,Southeast] 124 Gradient = np.zeros((h, w)) 125 for height in range(0,h-2): 126 for weight in range(0,w-2): 127 slice_img = img[height:height + 3, weight:weight + 3] 128 compare_ = [] 129 for _ in directions: 130 gx = np.sum(slice_img * _) 131 compare_.append(gx) 132 max_gx = max(compare_) 133 if max_gx >= threshold:Gradient[height, weight] = 255 134 else:Gradient[height, weight] = 0 135 Gradient = np.uint8(Gradient) 136 Gradient = 255-Gradient 137 return Gradient </pre>	
描 述	<p>和 Kirsch 結果類似，但產生的噪點又更少了，在髮尾部分的白色斑點比 Kirsch 還多。</p>	

	原圖	Nevatia_Bubu
圖 片		

程式碼	<pre> 139 def Nevatia_Babu(img,threshold): 140 kernel1 = np.array([[100,100,100,100,100], 141 [100,100,100,100,100], 142 [0,0,0,0,0], 143 [-100,-100,-100,-100,-100], 144 [-100,-100,-100,-100,-100]]) 145 146 kernel2 = np.array([[100,100,100,100,100], 147 [100,100,100,78,-32], 148 [100,92,0,-92,-100], 149 [32,-78,-100,-100,-100], 150 [-100,-100,-100,-100,-100]]) 151 152 kernel3 = np.array([[100,100,100,32,-100], 153 [100,100,92,-78,-100], 154 [100,100,0,-100,-100], 155 [100,78,-92,-100,-100], 156 [100,-32,-100,-100,-100]]) 157 158 kernel4 = np.array([[-100,-100,0,100,100], 159 [-100,-100,0,100,100], 160 [-100,-100,0,100,100], 161 [-100,-100,0,100,100], 162 [-100,-100,0,100,100]]) 163 164 kernel5 = np.array([[-100,-100,0,100,100], 165 [-100,-100,0,100,100], 166 [-100,-100,0,100,100], 167 [-100,-100,0,100,100], 168 [-100,-100,0,100,100]]) 169 170 kernel6 = np.array([[100,100,100,100,100], 171 [-32,78,100,100,100], 172 [-100,-92,0,92,100], 173 [-100,-100,-100,-78,32], 174 [-100,-100,-100,-100,-100]]) 175 kernel_list = [kernel1, kernel2, kernel3, kernel4, kernel5, kernel6] 176 h,w = img.shape 177 Gradient = np.zeros((h, w)) 178 for height in range(0,h-4): 179 for weight in range(0,w-4): 180 slice_img = img[height:height+5,weight:weight+5] 181 compare_ = [] 182 for _ in kernel_list: 183 gx = np.sum(slice_img * _) 184 compare_.append(gx) 185 max_gx = max(compare_) 186 if max_gx >= threshold:Gradient[height, weight] = 255 187 else:Gradient[height, weight] = 0 188 189 Gradient = np.uint8(Gradient) 190 Gradient = 255-Gradient 191 return Gradient </pre>
描述	<p>Nevatia_Bubu 的噪聲是所有邊緣偵測最少的，但在帽子邊緣的部分檢測的不太好，已經被閾值判定成非邊緣了。</p>