





電腦視覺 作業四

指導老師 傅楸善

學生 蔡宇晴


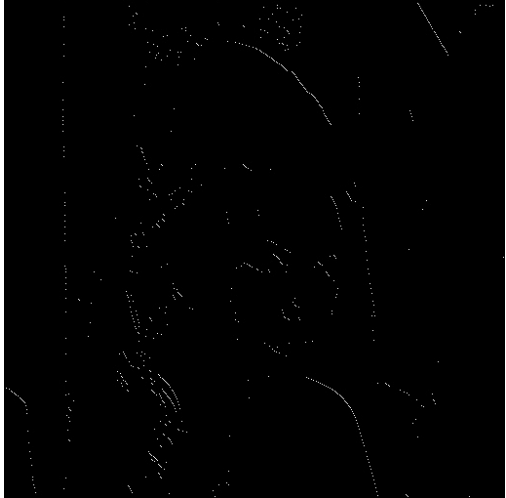
學號 R08945050

	原圖	Dilation
圖 片		
程 式 碼	<pre> 73 for height in range(h): 74 for width in range(w): 75 if img_pad[height+2,width+2] != 0: 76 slice_pad = img_pad[height:height+5,width:width+5] 77 slice_pad_and_masks_or = cv2.bitwise_or(slice_pad,masks) 78 final_ans = cv2.bitwise_or(slice_pad_and_masks_or,temp_img[height:height+5,width:width+5]) 79 temp_img[height:height+5,width:width+5] = final_ans </pre> <p>當遇到白點時（物體），以白點位置，開啟一個以白點為中心的 5x5kernel, 接著和 3-5-5-5-3 的八角形 kernel 做聯集，接著為了避免蓋掉原本在 temp_img 上的白點，於是再和 temp_img 相同位置、大小的 kernel 做聯集。</p> <p>接著才把 final_ans 賦值到 temp_img。</p>	
描 述	<p>膨脹會讓影像 object 變厚、變寬。其變化的方式，取決於先前決定好的 kernel,在此為 3-5-5-5-3 的八角形。</p> <p>文字斷裂或有裂痕，適當的使用膨脹或許可以讓閱讀更清楚。</p> <p>此張膨脹後的 lena 圖的髮尾部分可看出明顯加粗。</p>	

	原圖	Erosion
圖 片		
程 式 碼	<pre> for height in range(2,h+2): for width in range(2,w+2): slice_pad = img_pad[height-2:height+3,width-2:width+3] # 21 = 3+5+5+5+3=>八角形的mask if np.sum(cv2.bitwise_and(slice_pad,masks)) == 21: temp_img[height-2,width-2] = 255 </pre> <p>侵蝕的代碼較單純，只要判斷與 3-5-5-5-3 的八角形的 kernel 與原圖上對應的 kernel，取交集後，八角形 kernel 沒有出現值的改變（也就是八角形內沒有黑點），則把八角形的中心點(l,j)，在相對應的 temp_img[l,j]上標示為有值。</p>	
描 述	<p>侵蝕會讓物體變薄。其變化的方式取決於先前決定好的 kernel,在此為 3-5-5-5-3 的八角形。</p> <p>在想取出感興趣的物件時，可以利用侵蝕把特定的部分（例如：細線）去除，可以看出侵蝕後的 lena 圖，髮尾部分已經被去除。</p>	

	原圖	Opening
圖 片		
程 式 碼	<pre> 107 for height in range(2,h+2): 108 for width in range(2,w+2): 109 slice_pad = img_pad[height-2:height+3,width-2:width+3] 110 if np.sum(np.bitwise_and(slice_pad,masks)) == 21: 111 temp_img[height,width] = 255 112 '''做完erotion了 剩下dilasion''' 113 ans = np.uint(np.zeros((h+4,w+4))) 114 temp_img = np.uint8(temp_img)//255 115 116 for height in range(2,h+2): 117 for width in range(2,w+2): 118 if temp_img[height,width] != 0: 119 slice_pad = temp_img[height-2:height+3,width-2:width+3] 120 slice_pad_and_masks_or = np.bitwise_or(slice_pad,masks) 121 final_ans = np.bitwise_or(slice_pad_and_masks_or,ans[height-2:height+3,width-2:width+3]) 122 ans[height-2:height+3,width-2:width+3] = final_ans </pre> <p>第一步為侵蝕，與上述的侵蝕型相同 接著第二步再做膨脹。</p>	
描 述	<p>因侵蝕後再膨脹，依然會造成部分物體消失，例如細線消失後，就算再膨脹，也不會再出現。</p> <p>經過 opening 後，此張 lena 圖的髮尾被去除，且臉的輪廓變得平滑。</p>	

	原圖	Closing
圖 片		
程 式 碼	<pre> 146 img = np.pad(img,(2,2)) 147 img_complement = 255-img 148 masks = np.uint(mask(5)) 149 temp_img = np.uint(np.zeros((h + 4, w + 4))) 150 for i in range(2,h+2): 151 for j in range(2,w+2): 152 if np.sum(np.bitwise_and(masks,img_complement[i-2:i+3,j-2:j+3])) == 21: 153 final_bitwise = np.bitwise_or(masks,img_complement[i-2:i+3,j-2:j+3]) 154 temp_img[i-2:i+3,j-2:j+3] = final_bitwise </pre> <p>Closing 的幾何意義為，先把物件 A 取補集，在對補集做 3-5-5-5-3 的八角形的平移，再不與物件 A 交集的前提下，紀錄下所有經過的 kernel 位置。並把 kernel 經過的位置，都當作物件(值為 255)。</p> <p>接著，在對 A 的補集在取一次補集則還原至 A 物件。</p>	
描 述	<p>經過 closing 的物件輪廓會變得圓滑，但不像 opening 會保留隙縫處。closing 會把窄隙縫去除，填補細長缺口及小於 kernel 的洞。</p>	

	原圖	Hit and miss
圖 片		
程 式 碼	<pre> 179 mask_A = [[0,0,0],[1,1,0],[0,1,0]] 180 mask_A = np.array(mask_A) 181 182 mask_AC=[[0,1,1],[0,0,1],[0,0,0]] 183 mask_AC = np.array(mask_AC) 184 185 186 for i in range(1,h+1): 187 for j in range(1,w+1): 188 if img[i,j] != 0: 189 slice_pad = img[i-1:i+2,j-1:j+2] 190 if np.sum(np.bitwise_and(slice_pad,mask_A)) == 3: 191 temp_img[i,j] = 1 192 193 for i in range(1,h+1): 194 for j in range(1,w+1): 195 slice_pad = img_complement[i-1:i+2,j-1:j+2] 196 if np.sum(np.bitwise_and(slice_pad,mask_AC)) != 3: 197 temp_img_com[i,j] = 0 198 temp_img_com =np.uint(temp_img_com) 199 temp_img = np.uint(temp_img) 200 ans = cv2.bitwise_and(temp_img_com,temp_img) </pre>	
	<ol style="list-style-type: none"> 1.首先，開啟兩個右上角 kernel 2.接著對 img 做 mask_A 的 hit。也就是紀錄與 mask_A 交集後依然為 mask_A 的位置。 3.之後對 img 的補集做 mask_AC 的 hit。依然紀錄位置。 4.之後對兩個紀錄好的位置取交集，即可得到符合右上角的位置。 	

描 述	<p>Hit and miss 能夠辨識出特定的像素結構(kernel)，可分離前景像素或線段像素。</p> <p>此例為標示出具有右上角圖案的位置。</p> <p>因標示位置為右上角，因此在此圖的帽子右上處和肩膀右上處有明顯的標記。</p>
--------	----------------------------------------------------------------------------------------------------------------------------------