DIP HW2

洪郡辰
R11944050

March 26, 2023

1 Problem 1

1.1 P1.a

(a) sample1 (b) result1 (c) result2

Figure 1: Problem1.a.sample and result

(a) threshold 50 (b) threshold 150 (c) threshold 250

Figure 2: Problem1.a.different threshold

在 Fig.2 中的圖片分別為 threshold 為 50、150、250 的圖片。可以觀察到隨著 threshold 增加，圖片中的線條越少。這代表 threshold 越高則越少線段被認定為 edge。在 threshold 為 50 的圖片中可

以發現有許多部分被認定為 edge，這種情況下代表容易將雜訊考慮成 edge。在 threshold 為 250 的圖片中可以發現相較於 threshold 為 50 的圖片少了許多 edge，在如此高的 threshold 下也造成了 edge 的不連續，但在某些部分仍有著些許雜訊。

在 Fig.1 中為我的 result，我選擇 threshold 為 150 作為 result1，在這個情況下的 edge 的清晰度及雜訊達到了一個較好的平衡，edge image 看起來較為完整，因此我選擇其作為 result。

1.2 P1.b

(a) sample1 (b) result3

Figure 3: Problem1.b.sample and result

我透過調整以下三個參數對結果進行調整，以下為我使用在 result3 的參數:

(1) gaussian kernel size = 3x3 (2) low threshold = 150 (3) high threshold = 200

(a) 3x3 (b) 5x5 (c) 7x7

Figure 4: Problem1.b.different kernel size

在 Fig.4 中是不同 gaussian kernel size 下的結果，可以看到在 3x3kernel 下會有較少的部分被認定為 edge，這個可能是因為圖片在 3x3kernel 下經過 gaussian blur 後相較其他兩者仍不會太過平滑，5x5 及 7x7 可以看到有著比較多的細節及雜訊。我認為 3x3 作用下的圖片雜訊與 edge 的清晰度達成了一個較好的平衡，因此我選擇 3x3 作為 kernel size。

(a) 150 (b) 200 (c) 250

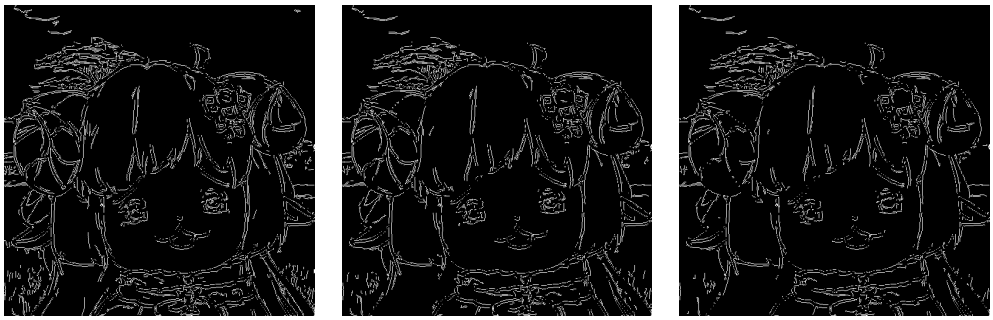

Figure 5: Problem1.b.different high threshold

在 Fig.5 中是不同 high threshold 的結果，low threshold 都是 100。我認為在調整 high threshold 時對於圖片中主體部分的紋路有較大的影響，在三張圖片中細節紋路並沒有隨著 high threshold 調整而有太大的影響。在主體部分的紋路上，high threshold 在 150 時太過明顯而 high threshold 為 200 及 250 時整體看起來差不多，但 250 的圖片中有許多 edge 都呈現斷斷續續的，因此我選擇 200 作為 high threshold。

(a) 50 (b) 100 (c) 150

Figure 6: Problem1.b.different low threshold

在 Fig.6 中是不同 low threshold 的結果，high threshold 都是 200。我認為在進行 low threshold 調整時能夠對細節的紋路進行很好地去除，可以看到在 Fig.6 的三張圖片中細部的紋路隨著 low threshold 上升而減少，我認為 low threshold 為 150 時的效果是最好的，因此我選擇 150 作為 low threshold。

1.3 P1.c

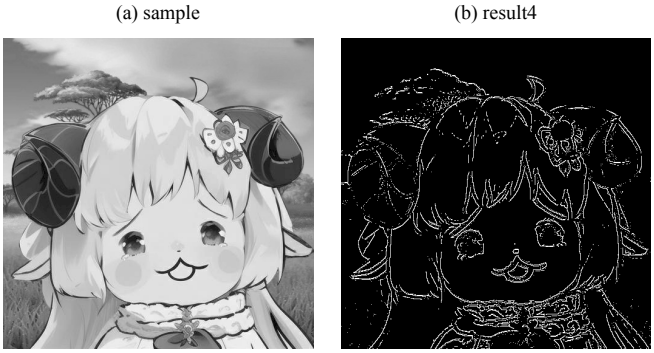

Figure 7: Problem1.c.sample and result

在 LOG 中我透過以下兩個參數對結果進行調整，以下為我使用在 result4 的參數：

(1) gaussian kernel size = 7x7 (2) threshold = 10

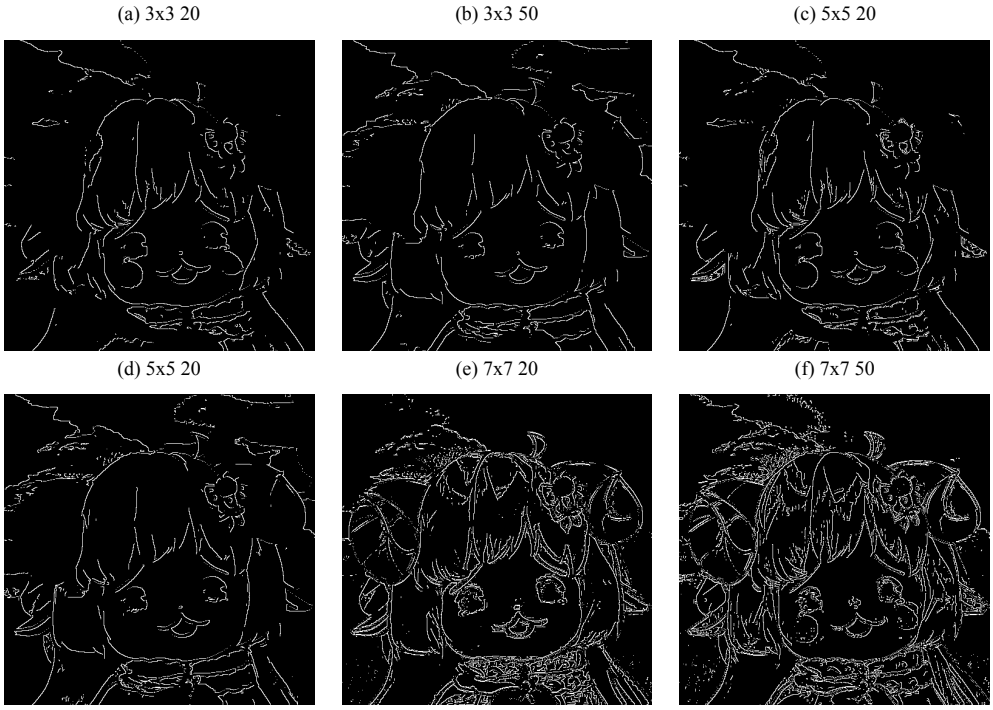

Figure 8: Problem1.c.different kernel size and threshold

在 Fig.8 中，我交叉比對不同 kernel 及不同 threshold 的影響。我發現隨著 kernel size 的增加及 threshold 的提升，edge 及雜訊越容易被偵測到。在 kernel size 為 3x3 及 5x5 的情況下，提升 threshold 仍無法有效地將邊緣檢測出來，部分邊緣沒有發生 zero-crossing，可能是平滑程度不夠。在將 kernel size 變成 7x7 時，主要的邊緣都有被檢測到，但也能發現有不少雜訊一起被偵測到，在

threshold 為 20 及 50 時都能發現有不少白點散布在 edge map 上。

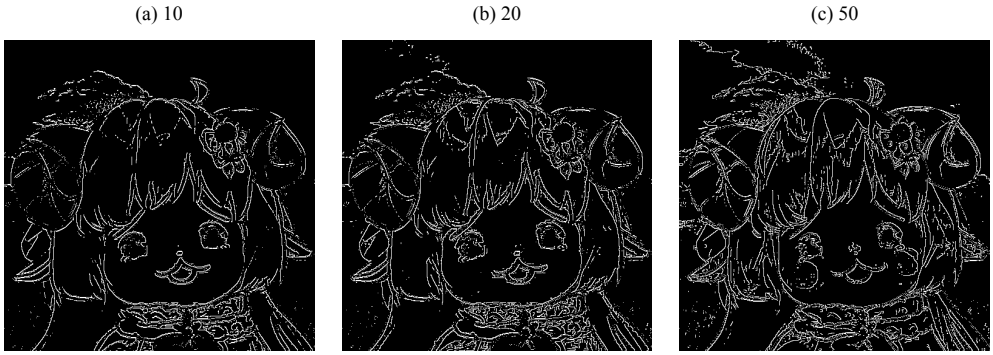

Figure 9: Problem1.c.different threshold

在 Fig.9 中，我選定 kernel size 為 7x7 再對不同 threshold 進行比較。在三張圖中我認為 threshold 為 10 的圖片效果最好，雜訊與邊緣取得一個較好的平衡，相較其他兩張雜訊白點少了許多。

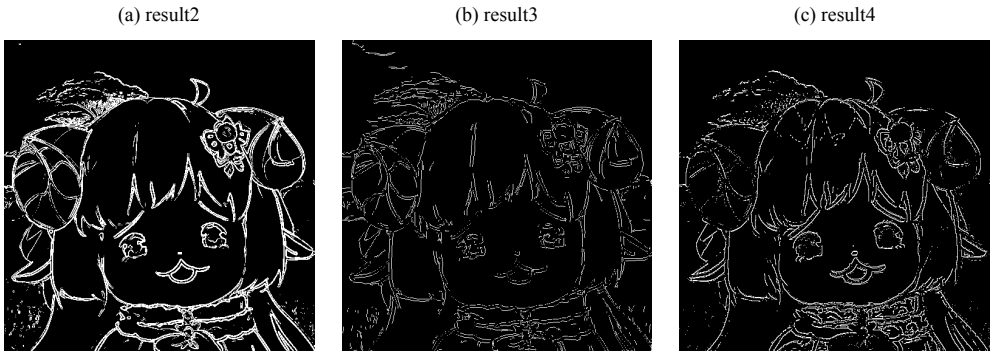

Figure 10: Problem1.c.different result

Fig.10 中為三張不同 edge detection 的 result，我發現 result2 相較其他兩者的 edge 較寬，可能是因為 sobel 沒有 canny 中的 non-maximal suppression 及 LOG 中的 zero-crossing detection，但其在視覺效果上我認為優於 canny。在對比一階 threshold 方法的 sobel 和 canny 以及二階 zero-crossing 方法的 LOG，我發現二階方法明顯對於噪音更加敏感但對於細部的邊緣也有更好的檢測能力，尤其是在頭髮的部分有最明顯的差距。我認為應該依據使用條件選用這三種 edge detection，每一種方法都有其特點，例如如果追求細部紋路特徵就使用 LOG，如果只需要大致上的邊緣就使用 canny，如果追求較粗的邊框就使用 sobel。

1.4 P1.d

(a) sample (b) result5

Figure 11: Problem1.d.sample and result

在 edge crispening 中我設定了一個參數，以下為我使用在 result5 的參數：

(1)c:0.6

(a) 0.6 (b) 0.7 (c) 0.8

Figure 12: Problem1.d.different c

在 Fig.12 中，我透過調整 c 對 edge crispening 的效果進行調整，c 代表 edge crispening 過程中原圖所佔的比例。我發現隨著 c 的上升 result 看起來越接近原圖，我認為是因為每個像素值之間透過 edge crispening 拉大的差距變小了，因此在邊緣即為像素值有較大梯度的位置，其周遭的像素值變化梯度隨著 c 上升而下降，造成邊緣沒有那麼清晰。我認為在 c 為 0.6 時，達到最好的效果，相較於原圖，result5 有著更清晰的圖片，圖片中的線條及窗戶看起來更有立體感，窗戶上的線條也清晰了許多。

2 Problem 2

2.1 P2.a

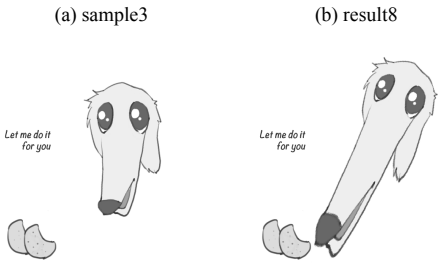

Figure 13: Problem2.a.sample and result

在 Fig.13 中為轉換後的結果，我透過以下步驟進行調整:

(1)translation (2)scaling (3)rotation (4)scaling

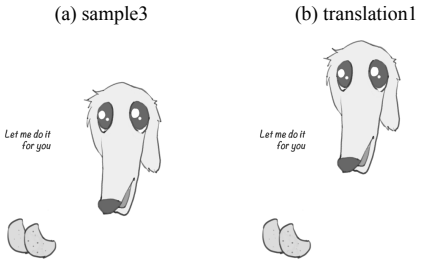

Figure 14: Problem2.a.translation1

在這個步驟我對 Borzoi 進行 translation，設定 tx=-40、ty=110，並針對 x 大於 200 的區域進行調整。我讓 Borzoi 移動到上方且稍微靠左，以方便於接下來進行拉長及旋轉的動作。

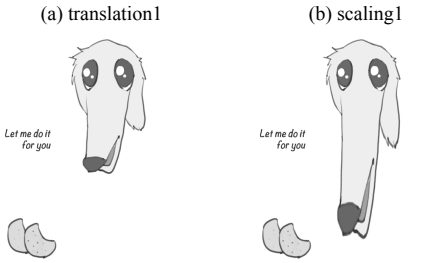

Figure 15: Problem2.a.scaling1

在這個步驟我對 Borzoi 進行 scaling，設定 xscal=1、yscal=2.1，並針對 x 大於 200 及 200<y<599 的區域進行調整，將 Borzoi 的嘴巴往下拉長至靠近 chips。

(a) scaling1 (b) rotation1

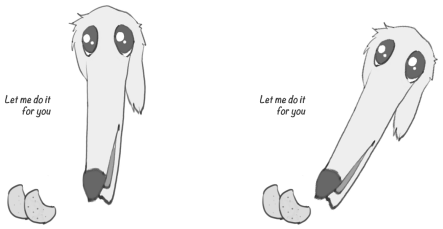

Figure 16: Problem2.a.rotation and translation 1

在這個步驟我對 Borzoi 進行 rotation 及 translation，我設定 theta=-25、針對 x>180 的部分進行調整。在旋轉過後我發現有部分嘴巴會跑至圖片外部，因此我在 rotation 中加入了 translation，使 Borzoi 能完整地回到圖片中且到一個適當地位置，我設定 tx=-85、ty=-130 讓 Borzoi 嘴巴在 chip 附近。我發現相較於 sample4 還離 chip 有點距離，因此我在下一步再加入 scaling。

(a) rotation1 (b) result8

Figure 17: Problem2.a.scaling2

在這個步驟我對 Borzoi 進行 scaling，我設定 xscal=1、yscal=1.1 並針對 x>180 及 200<y<599 的區域進行調整，將 Borzoi 的嘴巴拉至更靠近 chip 的位置。

2.2 P2.b

(a) sample5 (b) result9

Figure 18: Problem2.b.sample and result

在觀察 sample5 及 sample6 後我發現這是一個魚眼特效，因此我查詢了相關魚眼特效的物理效果，我了解到其是在作用範圍內進行圖像變形並且用靠近中心點位置的變形越強，隨著離中心點的距

離變大而變形變弱，也相當於凸透鏡的效果。

(a) deg0.8 (b) deg1.0 (c) deg1.2

Figure 19: Problem2.b.different degree

在一開始我先找出中心點及設定作用半徑，我設定 midx=150、midy=175、r=90，這個設定的中心點剛好位於嘴巴內且作用半徑到狗的眼睛下緣處附近，我認為 sample6 大約作用到這個地方。我設計出一個隨著距離變遠而作用變小的函式:

(1)u = int(midx+((x - midx) * (d / r)**deg))

(2)v = int(midy+((y - midy) * (d / r)**deg))

(x,y) 代表目前所求的座標位置、(u,v) 代表轉換前的座標位置、d 代表所求座標位置與中心點的距離、(d/r) 代表距離係數、deg 代表距離係數的次方。距離係數使得函數在距離中心點越遠時變形效果越弱，在 d=r 時不產生變形。

在一開始我並沒有設計出 deg 的部分，但我在觀察 deg=1 時，我發現我的 result 相較於 sample6 在嘴巴中心附近的變形量不夠，因此我想提高中心點附近的變形量，所以我想到能夠透過次方使得 (d/r) 較大時有更大的影響力，在 d=r 時同樣保持不變形。

在 Fig.19 中是不同 deg 的結果。我發現 deg 越大時中心點附近的變形量越大，deg 越小時中心點附近的變形量越小。在測試後我認為 deg=1.2 時最接近 sample6。

我認為 result9 及 sample6 在眼睛附近有比較大的不同，在 result9 中眼睛變形成月亮狀，在 sample6 中則保持在 sample5 中的形狀。其他部份我認為幾乎一模一樣。