







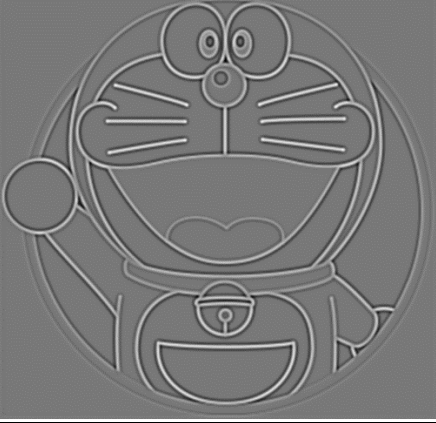

Computer Vision HW1 Report

Student ID: R10943117

Name: 陳昱仁

Part 1.

- Visualize the DoG images for 1.png.

	DoG Image		DoG Image
DoG1-1.png		DoG2-1.png	
DoG1-2.png		DoG2-2.png	
DoG1-3.png		DoG2-3.png	
DoG1-4.png		DoG2-4.png	

- Use three thresholds (2, 5, 7) on 2.png and describe the difference.

Threshold	Image with detected keypoints on 2.png
2	
5	
7	

(describe the difference)

觀察不同 DoG 的圖，可以看到第二個 octave 線條及模糊度都比第一個 octave 來得高，做越多
次 blur 得到的 DoG 在取得 keypoints 有較佳表現，圖片中亮暗變化也較為明顯

觀察不同 threshold 所得到的 keypoints，若 threshold 越高，得到的點越少，設定 threshold 的目
的是為了剔除一些絕對值不夠大的 outliers，這些被剔除的點很有可能是 noise 所導致的，故設定越
高的 threshold，得到的 keypoints 越少，但效果會更 robust。






Part 2.

- Report the cost for each filtered image.

Gray Scale Setting	Cost (1.png)
cv2.COLOR_BGR2GRAY	1207799
$R*0.0+G*0.0+B*1.0$	1439568
$R*0.0+G*1.0+B*0.0$	1305961
$R*0.1+G*0.0+B*0.9$	1393620
$R*0.1+G*0.4+B*0.5$	1279697
$R*0.8+G*0.2+B*0.0$	1127913


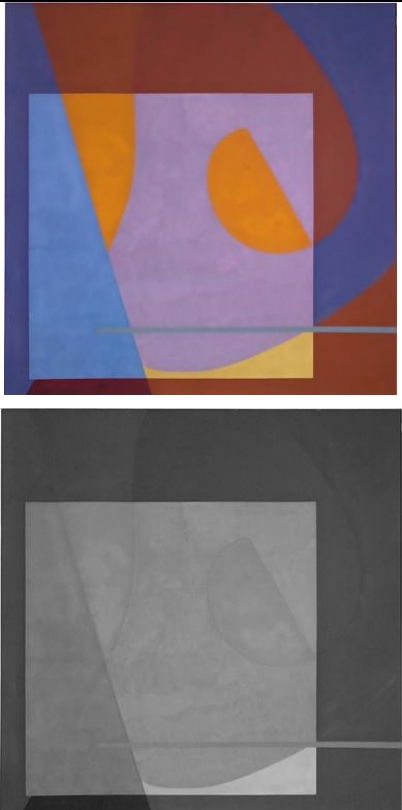
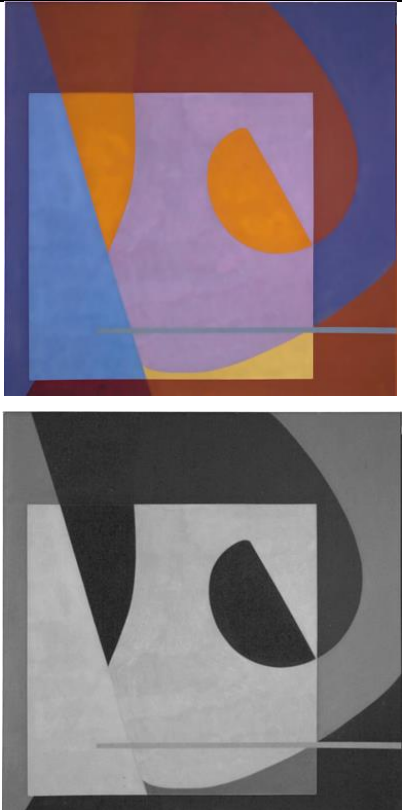
Gray Scale Setting	Cost (2.png)
cv2.COLOR_BGR2GRAY	183851
$R*0.1+G*0.0+B*0.9$	77883
$R*0.2+G*0.0+B*0.8$	86023
$R*0.2+G*0.8+B*0.0$	188019
$R*0.4+G*0.0+B*0.6$	128341
$R*1.0+G*0.0+B*0.0$	110862

- Show original RGB image / two filtered RGB images and two grayscale images with highest and lowest cost.

Original RGB image (1.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
	 	 

(Describe the difference between those two grayscale images)

Cost 較高的灰階圖，參數組合是 $R*0.0+G*0.0+B*1.0$ ，亮度皆由藍色所決定，紅色樹葉和旁邊綠色小草的顏色較無法分辨出來，相比之下 cost 較低的灰階圖，參數組合是 $R*0.8+G*0.2+B*0.0$ ，參數分布與比例和圖片內容相關性較高，紅色樹葉和綠色小草就能明顯辨認。此外和預期相同，cost 越高，灰階圖越無法辨認物體。

Original RGB image (2.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		

(Describe the difference between those two grayscale images)

Cost 較高的灰階圖，參數組合是 $R*0.2+G*0.8+B*0.0$ ，區塊已經無法容易辨識出來，中間橘色和紫色相接區看起來顏色一樣，外圍深紅色和深藍色也變成一大塊無法辨識，相比之下 cost 低的灰階圖，參數組合是 $R*0.1+G*0.0+B*0.9$ ，比例全分給紅藍兩色，並且沒分給圖中沒有的綠色，不同顏色的區塊明顯很多，可辨認出不同顏色區塊。

- Describe how to speed up the implementation of bilateral filter.

1. 因 Gr 的值皆在 $[\frac{1}{255}, \frac{2}{255} \dots \frac{255}{255}]$ 之間，且要經過 exp，若每次都重新取 exp，時間會過久，

因此先建立好 $[\frac{1}{255}, \frac{2}{255} \dots \frac{255}{255}]$ 經過 exp 的 table，之後需要用到在查表。

2. Gs 的 kernel，不論在算哪個 pixel 時，值皆不會改變，因此只須算一次即可，不論算哪個 pixel 皆拿同樣的 Gs。
3. 減少 for loop 運算，如下圖，我的 for loop 只用來給值，中間沒其他運算，所有變數運算皆是使用 numpy vector computation 的運算，一口氣算完整張圖，降低 for loop 運行來加速。

```
for h in range(H):
    for w in range(W):
        Tq[h, w] = padded_guidance[h:h+self.wndw_size, w:w+self.wndw_size]
        Iq[h, w] = padded_img[h:h+self.wndw_size, w:w+self.wndw_size]

Gr = self.get_Gr(guidance, Tq, guidance.ndim, r_table)
G = np.expand_dims(Gs * Gr, axis = -1)
output = (G * Iq).sum(axis = 2).sum(axis = 2) / G.sum(axis = 2).sum(axis = 2)

return np.clip(output, 0, 255).astype(np.uint8)
```