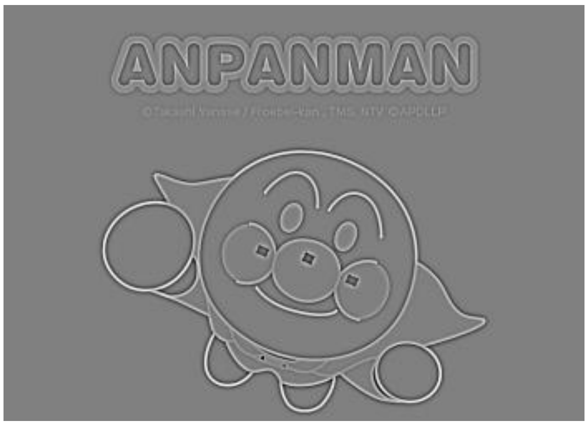
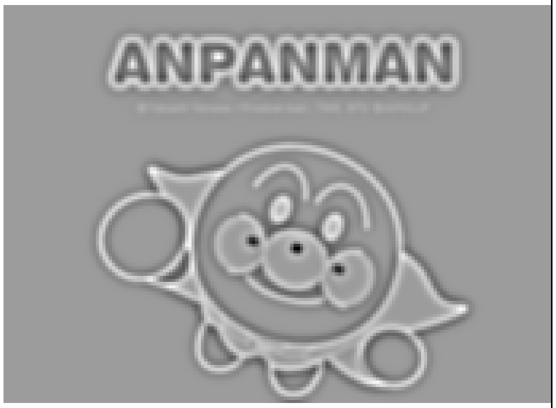
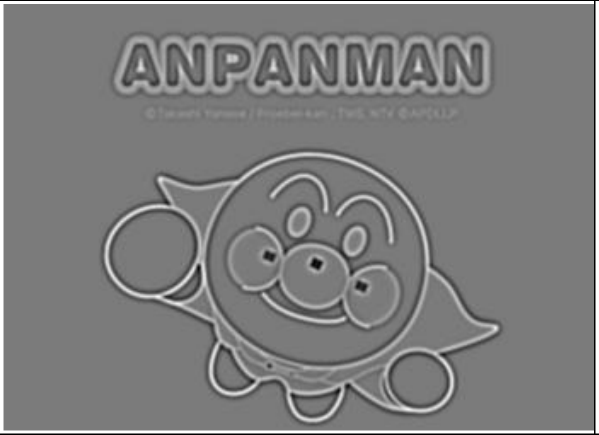
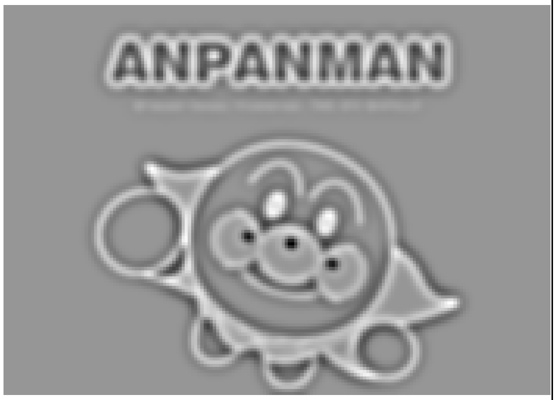
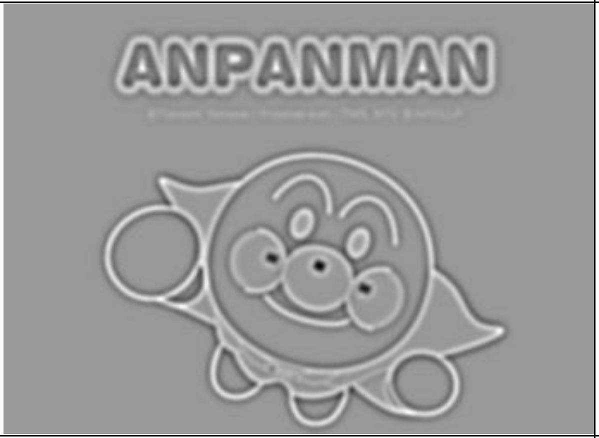
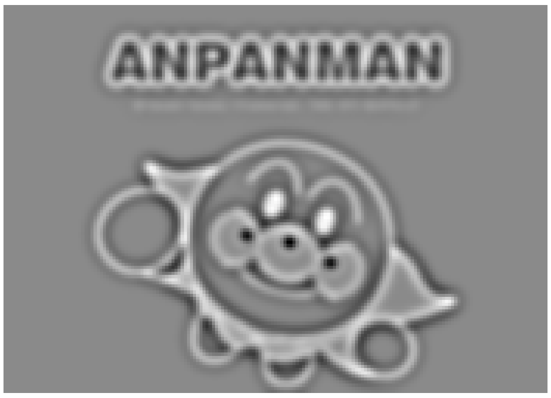
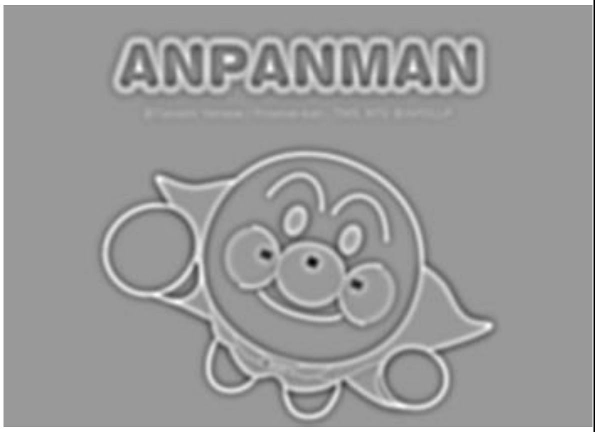
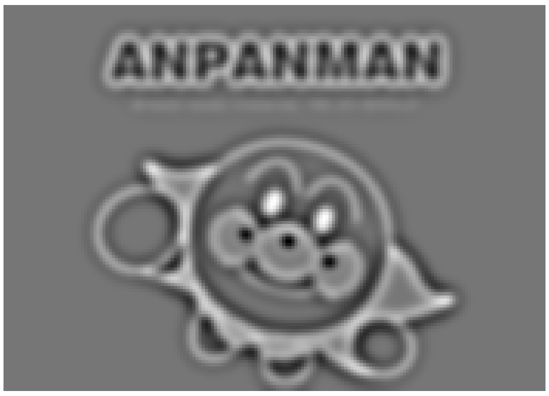
Computer Vision HW1 Report

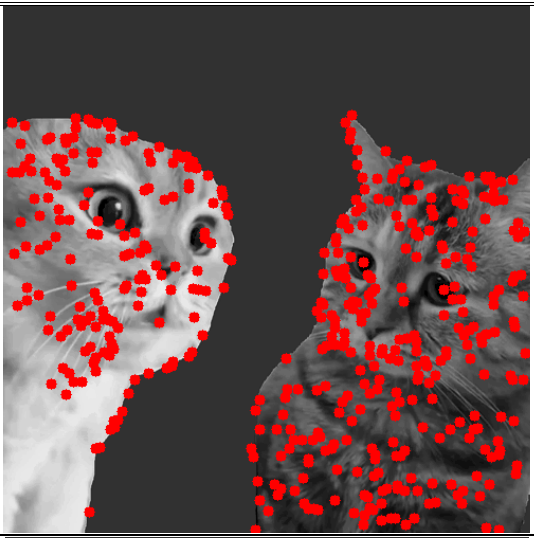
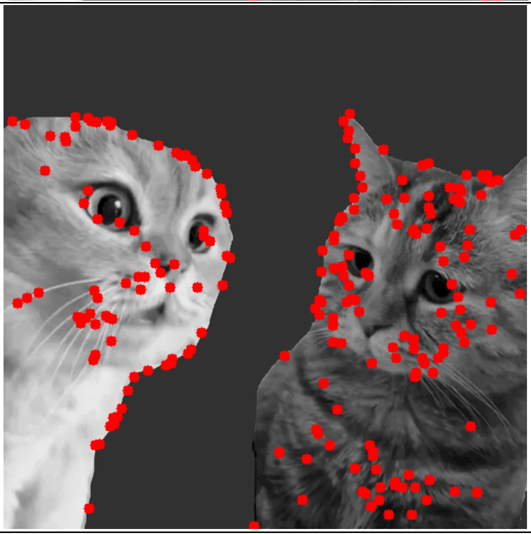
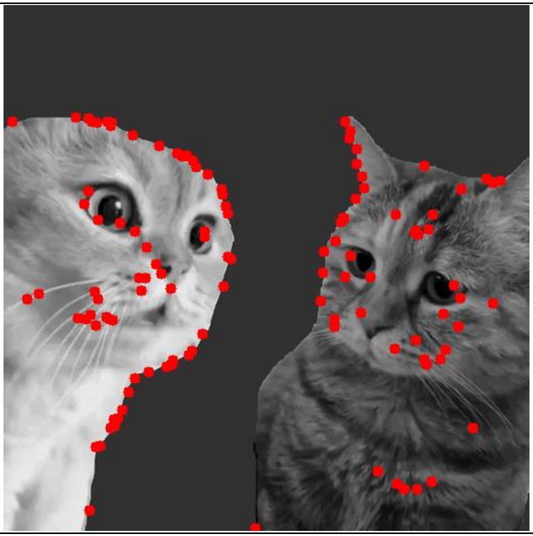
Student ID: B10901074
Name: 曾柏穎

Part 1.

- **Visualize the DoG images of 1.png.**

	DoG Image (threshold = 3)		DoG Image (threshold = 3)
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 (1,2,3) on 2.png and describe the difference.**

Threshold	Image with detected keypoints on 2.png
1	
2	
3	

(describe the difference)

隨著 threshold 上升，選取到的 keypoints 數量減少，並且留下的 keypoints 在那些變化更為明顯的邊界上。

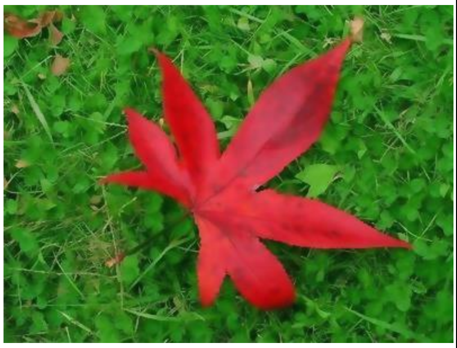
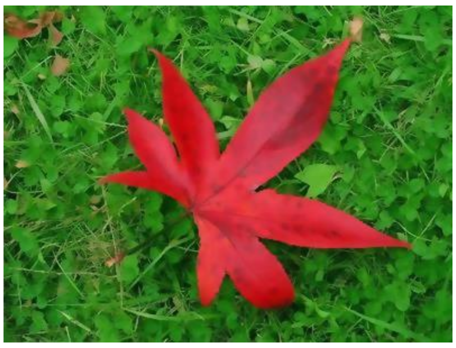
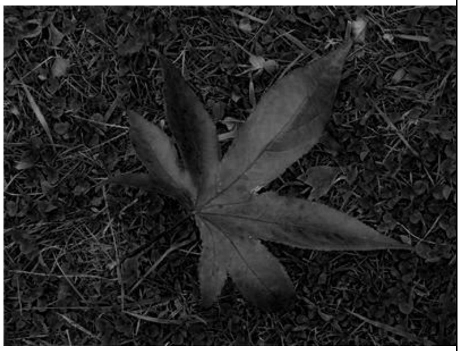
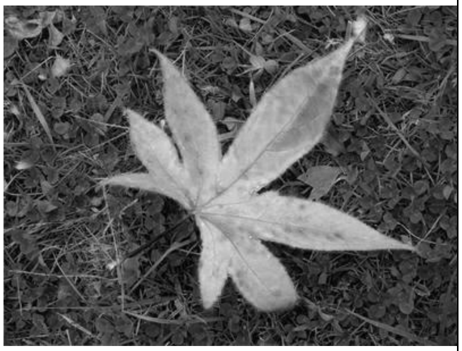
Part 2.

- **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

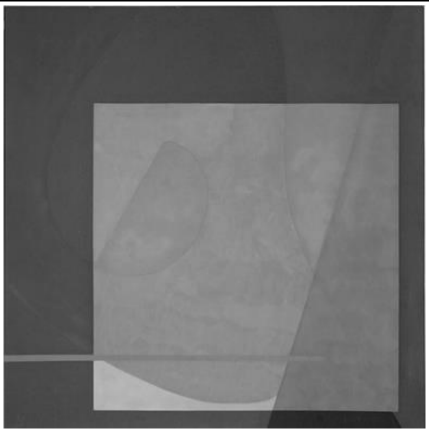
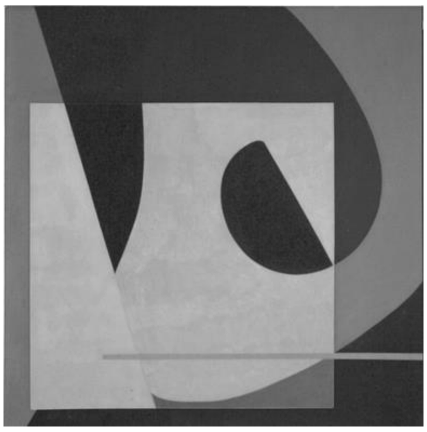
Gray Scale Setting	Cost (2.png)
cv2.COLOR_BGR2GRAY	183851
R*0.1+G*0.0+B*0.9	77884
R*0.2+G*0.0+B*0.8	86023
R*0.2+G*0.8+B*0.0	188019
R*0.4+G*0.0+B*0.6	128341
R*1.0+G*0.0+B*0.0	110862

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

Original RGB image (1.png)	Filtered RGB image and Grayscale image of Highest cost	Filtered RGB image and Grayscale image of Lowest cost
		
		

(Describe the difference between those two grayscale images)
Cost 最低的 guidance image 在邊界及不同區塊都有更明顯的分別，而 cost 最高的則讓整體有點混合在一起，與原圖的特徵相差較大。

Original RGB image (2.png)	Filtered RGB image and Grayscale image of Highest cost	Filtered RGB image and Grayscale image of Lowest cost

(Describe the difference between those two grayscale images)
Cost 最高的 guidance image 在對應原圖不同顏色的邊界顯得很不明顯，沒辦法清楚分出區塊，而 cost 最低的 guidance image 則還能有明顯邊緣與對比，因此做為 guidance 效果最好。

- **Describe how to speed up the implementation of bilateral filter.**
  在 jbf 中，每個 pixel 需要計算 window size*window size 個 pixel 疊加的資訊才能得到，因此我將每個 pixel 在 filter 的 window 中，對應到相同位移的 pixel 一起計算，如此一來就可以平行算出整張圖片對應 filter 中同樣位移的圖片，最後再將這些圖片疊加即可，過程中僅需用到 2 個 for loop 來算出 window size*window size 數量的圖片。