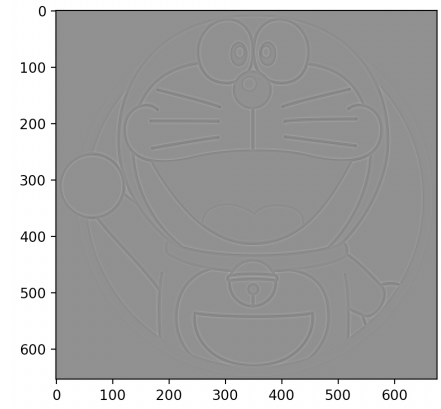
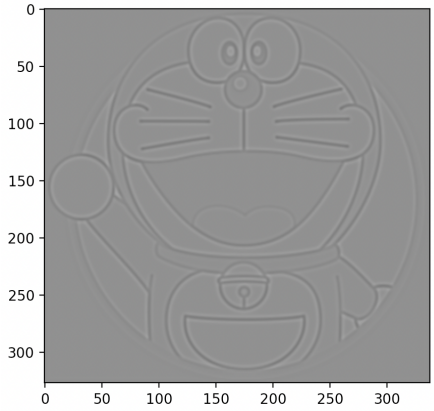
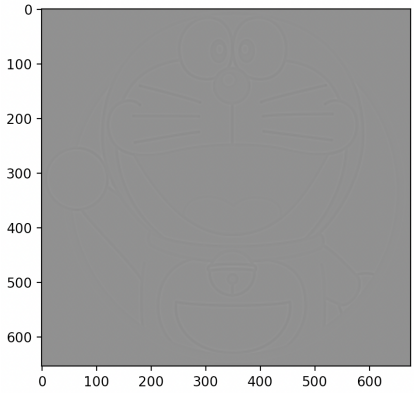
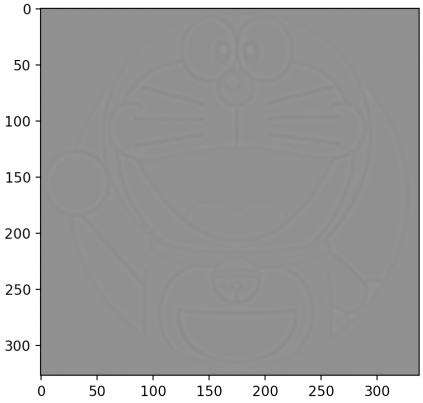
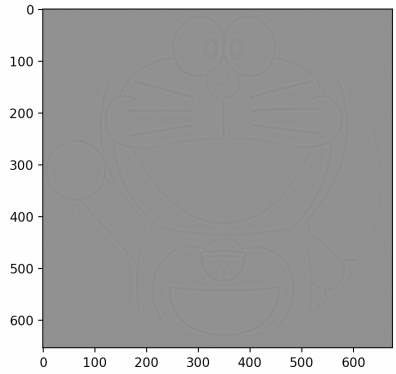
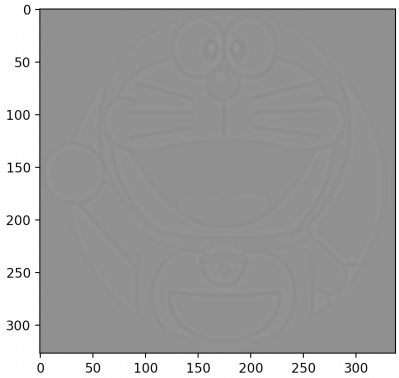
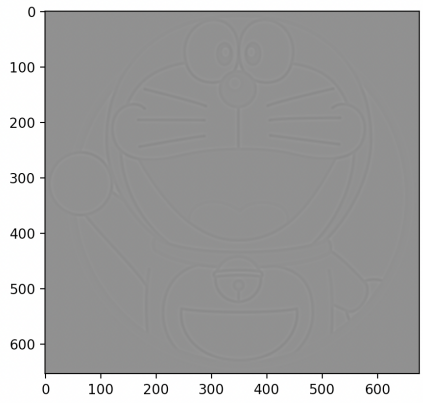
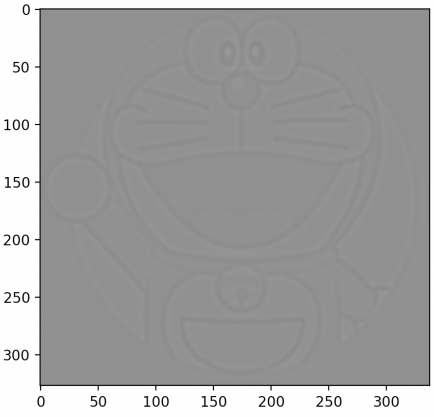
Computer Vision HW1 Report

Student ID: B07901036
Name: 陳俊廷

Part 1.

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

	DoG Image (threshold = 5)		DoG Image (threshold = 5)
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	基本上有顏色差異的地方都有被檢測出來
5	介於中間
7	只有顏色真的差很多的地方有被檢測出來(像是相鄰是全黑跟全白)

(describe the difference)

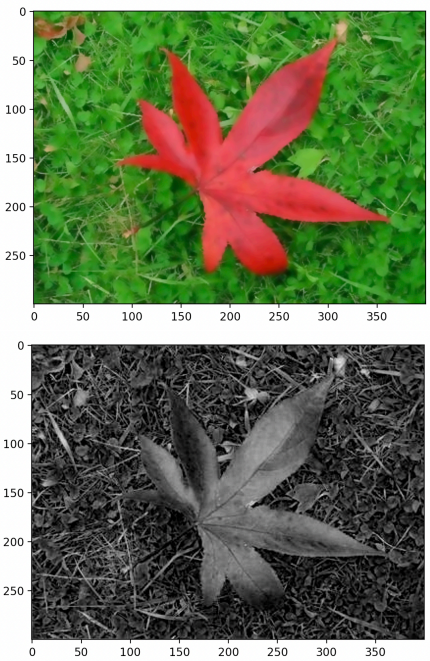
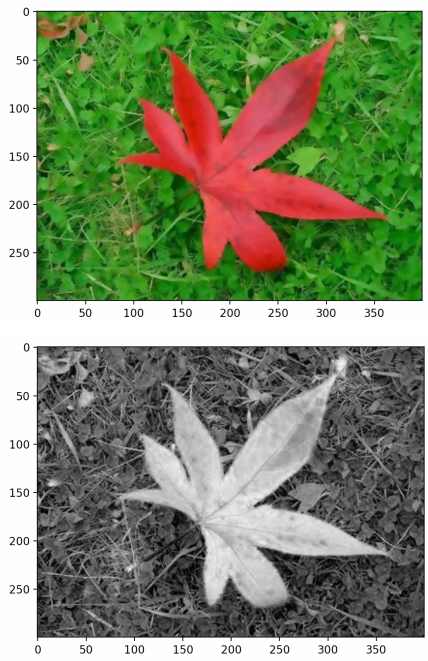
Part 2.

- **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

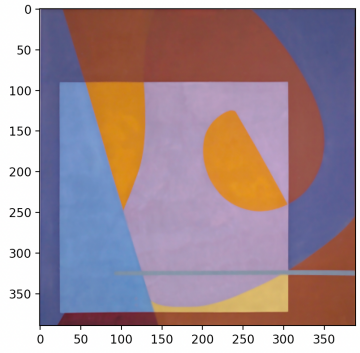
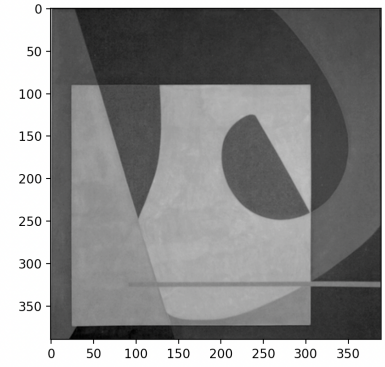
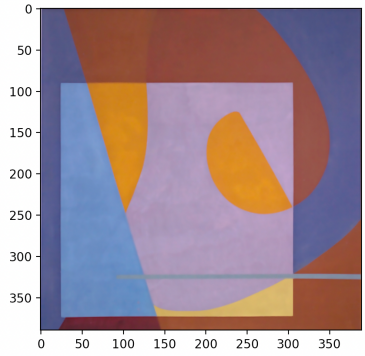
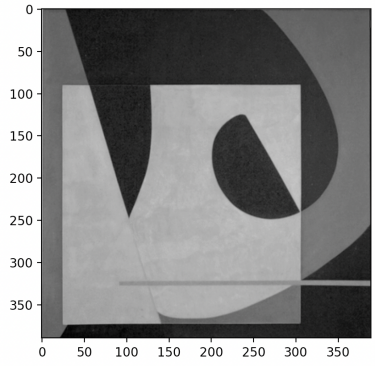
- **Show original RGB image / two filtered RGB images and two grayscale images with highest and lowest cost.**

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)

因為使用左邊的 gray scale setting 會讓楓葉跟草地太像，所以沒辦法很清楚的表示楓葉的邊緣。故拿來做 guidance 所產生的 JBF performance 會比較差(像是右邊數來第二個葉緣就很模糊)。

相比之下右邊的 setting 就讓楓葉跟草地在灰階有很大的差異，所以邊緣就很清楚。所以拿來當作 guidance 產生的 JBF 就會比較好。 **(guidance 的用途就是產生邊緣資訊)**

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)

其實同 1.png，左邊的 gray scale setting 會讓灰階圖的邊緣不是那麼清楚，所以拿來做成 guidance 的效果就沒有右邊好。

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

1. 只跑 window_size**2 的 for loop，也就是說單個迴圈內拿來運算的是整張圖的維度。原則上在使用 numpy 的情況下，單次運算的矩陣越大就會越快，所以這樣會比跑整張圖的 iteration 快
2. 在 range kernel 使用 LUT，因為 input img, guidance 會被轉成 uint8，之後再轉成 int32，所以其實在算 range kernel 的時候 intensity 差值只會有-255~255 的情況，後面又會平方，所以其實可以取絕對值，範圍再縮小成 255。故只要先算好 intensity 差值是 0~255 的 kernel weight，就可以避免在 for loop 裡面每次都做 exponential 的運算。(spatial kernel 只需要算一次，所以就不用 LUT 了)
3. 原則上在 for loop 裡面做的運算都可以用 numpy 實現，矩陣運算都比一個個 element 運算快
4. 在 BF 的部分，exponential 指數部分相加可以拆成 exponential 後相乘，所以就 RGB 三個 channel 分別算 intensity diff, 個別算出 exp 值後再相乘即可，還是可以用 2.建立好的 LUT