

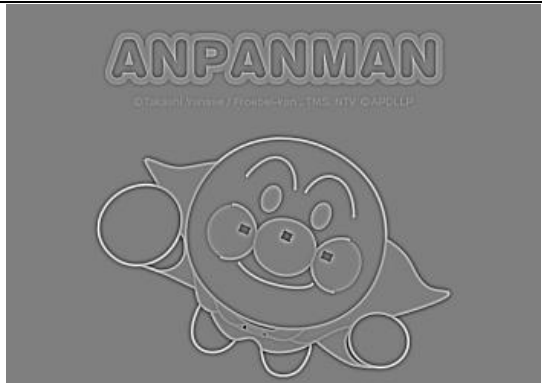

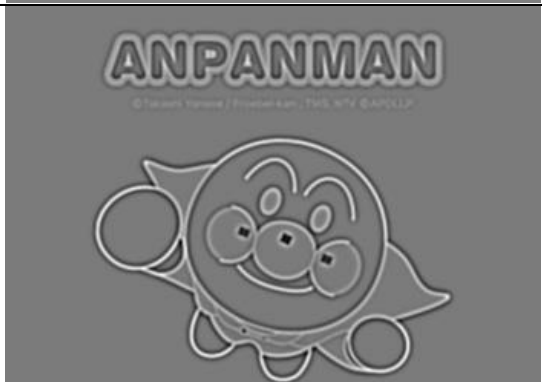

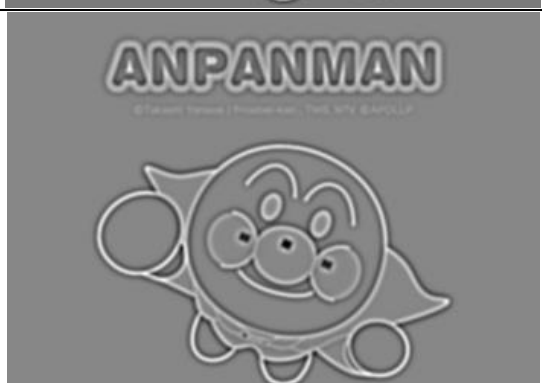

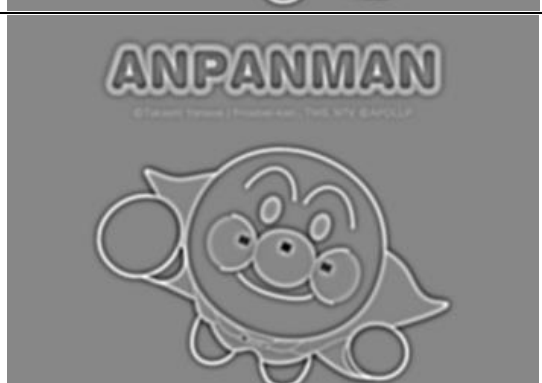

# Computer Vision HW1 Report

Student ID: R11921103

Name: 張銘軒

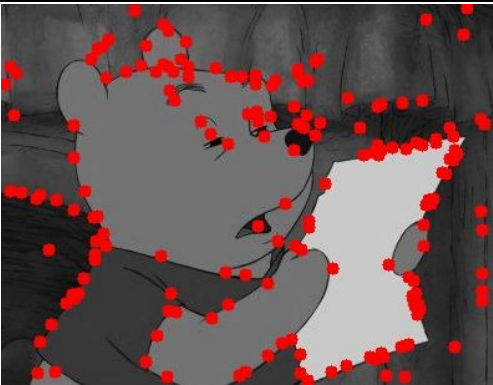


## Part 1.

- 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)






當 threshold 提高，key points 便會變少，這當然是因為有更多不是很清楚的邊界被篩選掉了。其中 threshold 是 1 的時候只要色塊亮度差異較大，就很容易被認為是邊界，因此有很多誤判的點；threshold 是 2 的時候效果好很多，能看出原圖輪廓或是辨認出原圖上比較明顯的線；threshold 是 3 的時候則是標的點正確率很高且標的點數較少。

## **Part 2.**

- **Report the cost for each filtered image.**





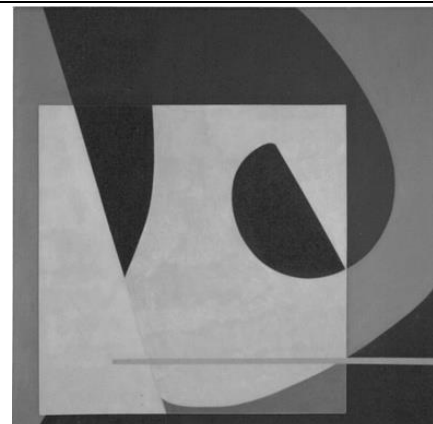
| Gray Scale Setting  | Cost (1.png) | Gray Scale Setting  | Cost (2.png) |
|---------------------|--------------|---------------------|--------------|
| cv2.COLOR_BGR2GRAY  | 1207799      | cv2.COLOR_BGR2GRAY  | 183850       |
| $R*0.0+G*0.0+B*1.0$ | 1439568      | $R*0.1+G*0.0+B*0.9$ | 77883        |
| $R*0.0+G*1.0+B*0.0$ | 1305961      | $R*0.2+G*0.0+B*0.8$ | 86023        |
| $R*0.1+G*0.0+B*0.9$ | 1393620      | $R*0.2+G*0.8+B*0.0$ | 188019       |
| $R*0.1+G*0.4+B*0.5$ | 1279697      | $R*0.4+G*0.0+B*0.6$ | 128341       |
| $R*0.8+G*0.2+B*0.0$ | 1127913      | $R*1.0+G*0.0+B*0.0$ | 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<br><u>Grayscale image</u> of<br>Highest cost         | Filtered <u>RGB image</u> and<br><u>Grayscale image</u> of<br>Lowest cost           |
|---|--|---|
|  |  |  |
|   |  |  |

(Describe the difference between those two grayscale images)

Highest cost 的灰階圖很暗，非常難以辨認原圖中的楓葉與其背景草地的差異性。反之，lowest cost 的灰階圖很明顯的看出楓葉，並由明亮差異可以更清晰的看出背景草地的紋路與材質。

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

(Describe the difference between those two grayscale images)

Highest cost 的灰階圖不同色塊混在一起，幾乎變成只有中間的亮色塊以及旁邊的暗色塊，非常難以辨認原圖中不同的色塊。反之，lowest cost 的灰階圖很明顯的看出色塊的明暗以及邊界，可以更清晰的看出原圖中的不同色塊。

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

在實作中，由於距離 $\pm 3\sigma_s$ 的 window 所構成的 spatial kernel 每一次迴圈都會是 $e^{-\frac{x^2+y^2}{2\sigma_s^2}}$ ，所以可以預先建立 spatial kernel 不用每次重算；我也建立了 look up table 存 $-\frac{n^2}{2(255*\sigma_r^2)}$ ， $n \in 0, 1, \dots, 255$ 的值，可以減少之後在算 range kernel 指數部分時的計算。為了再讓他快個幾毫秒我預先判斷 guidance 為灰階或彩色，以減少在迴圈中多餘的判斷式。最後透過 numpy.roll 的幫忙，我將原先移動 window 的程式改寫為移動 image 的版本，可大幅下降迴圈次數從 image size 至 window size。