



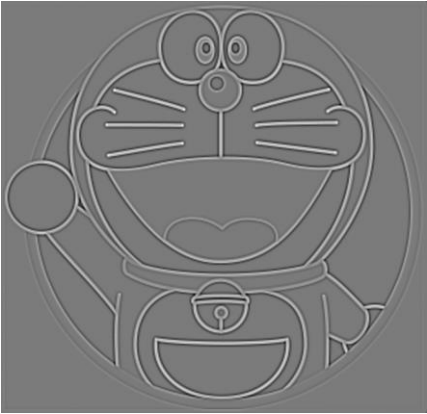



# Computer Vision HW1 Report



Student ID: R08222017

Name: 陳韋辰

## 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		keypoints: 200	
5		keypoints: 52	
7		keypoints: 23	

(describe the difference)

較低的 threshold(像是 threshold=2 時)可以找到較多的特徵點，在布丁和盤子的邊界、人的五官以及字體上都有很多的特徵點，完整地把 corner 和變化大的點找出來，但是過多的特徵點反而有很多重複，甚至抓到不太算是有足夠變化的地方(像是 threshold=2 時的字體上有一些不合理的點)。相反的較高的 threshold(像是 threshold=5、7 時)每個點都點在特徵很明顯的 corner 或是 edge 上，雖然

有些特徵抓不太到，但每個特徵點都很精確。






## 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


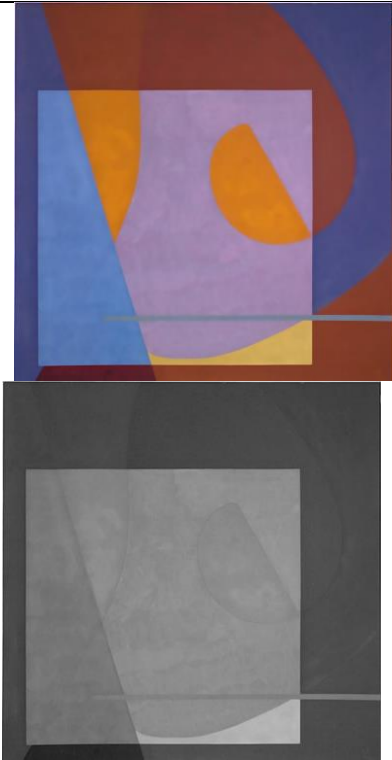
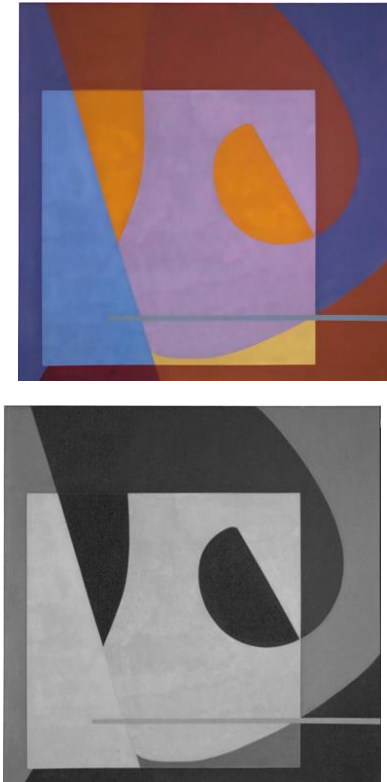
Gray Scale Setting	Cost (2.png)
cv2.COLOR_BGR2GRAY	183851
$R*0.1+G*0.0+B*0.9$	77883
$R*0.2+G*0.0+B*0.8$	86023
$R*0.2+G*0.8+B*0.0$	188019
$R*0.4+G*0.0+B*0.6$	128341
$R*1.0+G*0.0+B*0.0$	110862

- 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)

在 cost 高的 grayscale image，對比度較差，整張都黑黑的，也因此用在 guidance 上可能會造成 range kernel 的效果比較不好，可能沒辦法準確地在保留邊界的情況下模糊圖片。相對的，cost 低的 grayscale image 黑白很明顯，可以產生較佳的 range kernel，以保留邊界，像是在紅葉上方的綠草，低 cost 的 JBF 圖就可以看到草的邊緣，相對的 cost 高的 JBF 圖就很模糊。

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)

在 cost 高的 grayscale image，黃色跟藍色區塊都白白的，沒有很明顯的對比，而在低 cost 的 grayscale image 就差異很大，每個區域都有很明顯的黑白差異。從 JBF 的效果也可以很明顯的看出來，在 cost 高的 JBF 圖中邊界是模糊的，相較之下低 cost 的 JBF 圖有很清晰的邊界。至於在同色的區塊，不論 cost 高低，區塊內的紋路都由 JBF 的 spatial kernel 很好的模糊掉，達到 JBF 的效果。

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

#### 1. 使用 look-up-table :

在 range kernel 的計算上因為 pixel 的差只會在 -255 到 255 之間，所以會使用到大量重複的 exponential function，因此創建 Exponential function 的 look-up-table 會降低 range kernel weight 的計算 cost，從指數計算的  $O(r^2)/\text{pixel}$  的計算量降為  $O(1)$  的指數計算和  $(r^2)/\text{pixel}$  次查表。在 Ubuntu(64-bit)的模擬器 Intel(R) Core(TM) i5-8250 + 2GB RAM 的環境下，可以從不用 LUT 的 4.02 s 提升至 3.16 s，可以提升約 20% 的速度(平均 10 次)。

#### 2. 使用 NumPy 運算：

利用 NumPy 的公式，漸少 for-loop 的使用，可以大幅加速程式運算。

