**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