**Computer Vision HW1 Report**

Student ID: R12921107

Name: 王冠傑

**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 (1,2,3) on 2.png and describe the difference.**

|  |  |
| --- | --- |
| Threshold | Image with detected keypoints on 2.png |
| 2 |  |
| 5 |  |
| 7 |  |

(describe the difference)

When the threshold number grow up, keypoints number will gradually decrease. In other words, the larger threshold will gain more important features, and filter the redundant one.

**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 | 436673 |
| R\*0.1+G\*0.0+B\*0.9 | 120674 |
| R\*0.2+G\*0.0+B\*0.8 | 401344 |
| R\*0.2+G\*0.8+B\*0.0 | 143870 |
| R\*0.4+G\*0.0+B\*0.6 | 230122 |
| R\*1.0+G\*0.0+B\*0.0 | 359603 |

* **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)

|  |  |  |
| --- | --- | --- |
| 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)

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

1. **借鑑於參考資源，比起pixel-by-pixel的處理，一次計算整張圖，再做window size次的相加效率會更高。**
2. **計算Gs\_LUT和Gr\_LUT，一次性計算完所有的Gs, Gr window 各相對位置的數值，之後再用查表的方式計算特定點實際的Gs, Gr。**
3. **藉由Gaussion的特性，僅需要1D的Gs即可快速計算2D的Gs。**

**其他：嘗試使用broadcast, np.prod 等numpy優化程式碼算法，但因實測後效率反而降低。原因可能是過程涉及內存使用，因此棄用方法。**

**Reference:**

[**https://github.com/bettyteng21/1122-CV**](https://github.com/bettyteng21/1122-CV)

[**NTUEE-CV-2022Spring/homework1/part2/JBF.py at main · Offliners/NTUEE-CV-2022Spring · GitHub**](https://github.com/Offliners/NTUEE-CV-2022Spring/blob/main/homework1/part2/JBF.py)

[**bilateralfilter-numpy/bilateralfilter.py at master · wctu/bilateralfilter-numpy · GitHub**](https://github.com/wctu/bilateralfilter-numpy/blob/master/bilateralfilter.py)

[**Joint-Bilateral-Filter/JBF.py at main · Spheluo/Joint-Bilateral-Filter · GitHub**](https://github.com/Spheluo/Joint-Bilateral-Filter/blob/main/JBF.py)

[**numpy as\_strided 入门教程（啰嗦版） - 知乎**](https://zhuanlan.zhihu.com/p/597292224)

[**Joint-Bilateral-Filter/JBF.py at main · Spheluo/Joint-Bilateral-Filter · GitHub**](https://github.com/Spheluo/Joint-Bilateral-Filter/blob/main/JBF.py)