**Computer Vision HW1 Report**

Student ID: R12921059

Name: 鄧雅文

**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 |  |

因為Threshold越大，會被列入「不考慮」的點就越多，導致keypoint的candidate變少，所以可以從上面三張圖看出當threshold越大，keypoint越少，這個現象是合理的。

**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 | 183850 |
| R\*0.1+G\*0.0+B\*0.9 | 77882 |
| 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)

|  |  |  |
| --- | --- | --- |
| 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.**

參考Reference 6的作法

不是用傳統pixel-by-pixel的去loop，是用另一個視角去看，一次看整張圖。

以下都以19\*19的kernel size為例：

1. 算Gs:

先做單一軸的LUT，e.g. x方向位置差的結果只可能是0~19。

之後算Gs就依照公式把x跟y部分乘起來。

1. 算Gr:

一樣先做LUT，intensity range會落在(0~255)/255

算Gr時要站在整張圖的角度對圖做roll，再減掉不動的圖，之後查LUT。

1. 算分子分母:
2. 累加公式的分子Gs\*Gr\*roll(padded\_img) :

在傳統做法每次都處理一個window的加總，然後掃完一整張圖。這裡只是把這步顛倒過來，一次算整張圖，然後window size次的相加(能縮減到loop kernel size的主因)。

1. 累加公式的分母Gs\*Gr :

概念同上

1. 算output:

依照公式相除，同時crop掉padding的部分(因為前面的計算都是拿padded算的)

**Reference:**

[1] <https://github.com/Offliners/NTUEE-CV-2022Spring>

[2] <https://github.com/Louislar/NTU_CV_HW>

[3] <https://github.com/wctu/bilateralfilter-numpy/blob/master/bilateralfilter.py>

[4] <https://tree.rocks/convolution-with-numpy-matrix-dot-d4bc338b9142>

[5] <https://zhuanlan.zhihu.com/p/597292224>

[6] <https://github.com/Spheluo/Joint-Bilateral-Filter>