**Computer Vision HW1 Report**

Student ID: R10625016

Name: 許致銓

**Part 1.**

* **Visualize the DoG images for 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 |  |
| 5 |  |
| 7 |  |

(describe the difference)

**Due to the adjustment of the threshold, we can find the higher the standard of “keypoints” is, the more the “keypoints” only lie on the edge which are drastically change in color.**

**You can see there are many points on the picture with threshold=2, and points are even drawn on the plate’s edge. Then the pictures with threshold=5, there are only few points on the edge of plates and the bottom of the human’s face. The last pictures with threshold=7, we can see points only lie on edges with high color contrast.**

**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)

**The two grayscale pictures are totally different in colors. The one with highest cost only takes the blue channel, and the other with lowest cost takes 0.8 red channel and 0.2 green. In my opinion, the original picture mainly consists in red and green, which cause the high contrast of the manipulation factor between “with red and green” & “Without red and green”.**

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

**The original picture has a bigger part of blue. Subsequently, the picture with highest cost consists of 0.2 red and 0.8 green, and the picture with lowest cost consists of 0.1 red and 0.9 blue. As written above, the main difference is the blue. As a result, the channel of the pixel value is really important to lower the cost.**

* **Describe how to speed up the implementation of bilateral filter.**
* **First of all, following the TA’s suggestions, I build the Gs table to map the Gs index. Secondly, I try to build a Gs table for calculating the Gs formula. In so doing, I don’t need to run the three nested for loop. However, it is not that easy to figure it out. After receiving TA’s suggestion and the discussion with classmates, I understand that if I construct the “kernel value cube”, I can O(1) take the value of p-q. Moreover, there is a fabulous solution to solve the problem. That is, storing the p-q value by subtract the pictures which starts from the kernel’s [0, 0] to [18, 18], which means do the subtraction with all the pixels in original pictures to each kernel’s [0, 0]. In so doing, I only need to subtract the picture for 361 times, and my runtime can drastically be reduced.**