**Computer Vision HW1 Report**

Student ID: B10505047

Name: 邱郁喆

**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 |  |

(describe the difference)

當threshold = 1時，可得到非常多的keypoint，兩隻貓的臉部多處都有被偵測

當threshold = 2時， keypoint數量有略微減少，尤其兩隻貓的臉部及身體上都少很多

當threshold = 3時，keypoint數量明顯減少很多，主要剩下兩隻貓的臉形邊緣處以及臉上的零星部分

**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)

這兩張灰階圖的深淺有很明顯的差異，highest cost的灰階圖整體都偏暗，而lowest cost的灰階圖則是葉子的部分特別亮，跟旁邊的草地形成對比。highest cost的灰階圖RGB係數組成為(0, 0, 1)，也就是只有blue channel；lowest cost的灰階圖RGB係數組成為(0.8, 0.2, 0)，也就是側重於red channel。由於原圖片大致由紅色與綠色組成，因此以RGB係數組成為(0, 0, 1)的灰階圖作為guidance固然會是highest cost；而以RGB係數組成為(0.8, 0.2, 0)的灰階圖作為guidance因與原圖顏色組成較相近，因此是lowest cost。

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

highest cost的灰階圖較無法看出原圖有顏色差異的地方，而lowest cost的灰階圖則相反。highest cost的灰階圖RGB係數組成為(0.2, 0.8, 0)，也就是側重於green channel；lowest cost的灰階圖RGB係數組成為(0.1, 0, 0.9)，也就是側重於blue channel。由於原圖綠色的成分相對較少，因此以RGB係數組成為(0.2, 0.8, 0)的灰階圖作為guidance固然會是highest cost；而以RGB係數組成為(0.1, 0, 0.9)的灰階圖作為guidance因與原圖一樣有不少藍色成分，因此是lowest cost。

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

首先，由於spacial kernel gs的計算只跟距離有關，因此可以不用寫在考慮每個window中心點的for loop中，以減少在迴圈中重複計算所花的時間。Range kernel gr則是因為跟guidance的pixel value有關，因此較適合寫在for loop。而對於gs和gr的計算我都採用了numpy的broadcast特性，我想如此一來能夠加速矩陣的運算，減少重新調整矩陣大小所需的時間。此外，對於程式中需計算總和的部分，我發現使用np.sum() 的運算速度比直接使用python內建的 .sum() 來得慢不少。我原本都是使用np.sum()來計算總和，在改成 .sum() 後時間約減少了0.5~1秒。經過上網搜尋相關文章，我得知這可能是因為對於處理比較小的數組，python內建的 .sum() 不需額外的數據轉換及函式調用的時間，因此在此程式中可得到較好的運算速度。