

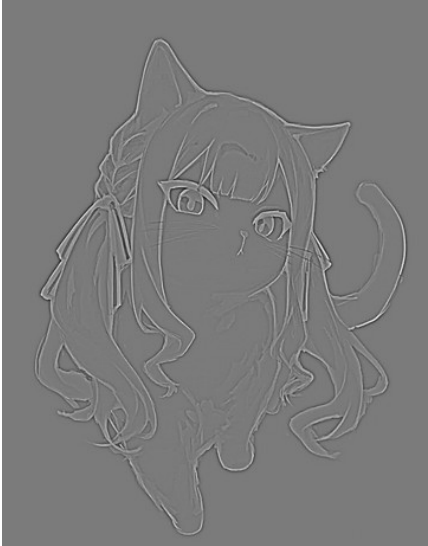



Computer Vision HW1 Report





Student ID: B10901151
Name: 林祐群

Part 1.


- Visualize the DoG images of 1.png.

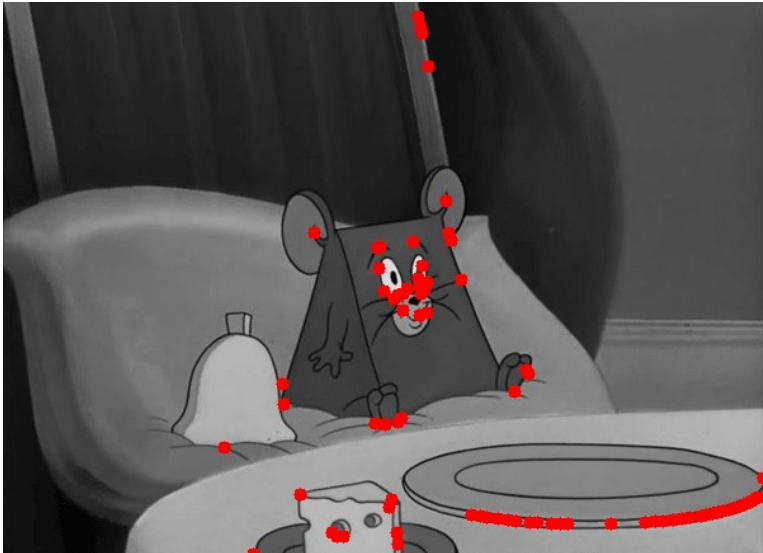

As visualized below, it's clearly shown that the Difference of Gaussian method applied in the problem includes two scales. One is those depicted in the left-side column, and the other is on the right-side column.

	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)

The differences between the three images lie in several aspects. For instance, the number of the selected key-points varies. It is obviously displayed that higher the threshold is chosen, less the number of key-points is shown. This is directly corresponded to the algorithm of our method that filters out those points that are not significant enough. In addition, the key-points which are still kept in the images in the images with 7 threshold are considered more important, or at least to implicitly inherit more information and higher contrast in DoG methods.






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.




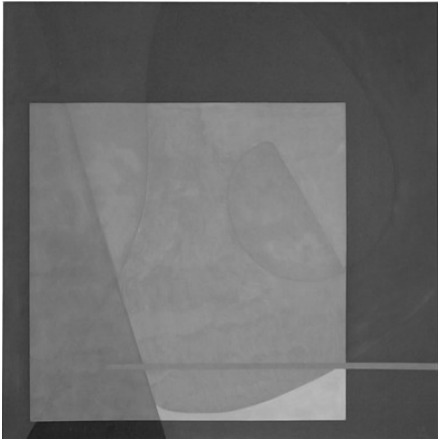

Original RGB image (1.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		
		

(Describe the difference between those two grayscale images)

As shown above, the origin RGB images on the leftmost column is comprised mainly of color red and green. That is, if the given guidance in grayscale has more weight on red and green, the L1-norm of the such setting will be lower. With this in mind, we compare the two image, one is of the highest cost ($R*0.0+G*0.0+B*1.0$) and the other is of the lowest cost ($R*0.8+G*0.2+B*0.0$). The lowest cost is achieved when giving significant weight to the red channel (0.8) and some weight to green (0.2), while completely ignoring the blue channel.

Conversely, the highest cost occurs when only the blue channel is considered ($R0.0+G0.0+B*1.0$). The results directly illustrate that the one with higher weight on color red and green leads to lower cost, as I stated intuitively.

In addition, we can discover that even though the grayscale images in the two setting are highly different, especially with respect to the luminance and edge preserving ability, the resulting RGB images look visually alike. With joint bilateral filter, we can attain better edge preserving ability with the guidance with clear edge structures. Thus, we can observe such conclusion with the images provided above.

Original RGB image (2.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		
		

(Describe the difference between those two grayscale images)

Similar to the aforementioned intuition, the highest L1-norm cost happens with $R*0.2+G*0.8+B*0.0$ and the lowest takes place with $R*0.1+G*0.0+B*0.9$. This is directly related to the color that mainly comprises the given image. In this case, color red and blue make their dominance, while green is negligible. That is to say, with the setting of $R*0.1+G*0.0+B*0.9$ (where color blue also dominates color red), we can obtain the better

result.

Additionally, there are significant difference in the grayscale images of the two setting. For the setting with higher L1-norm cost, the image is lighter and the edge is more obscure than the one with lower cost. Thus, the effectiveness of the guidance will definitely affect the results after joint bilateral filter, which can be reconfirmed through the illustration above.

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

First of all, I apply look-up table pre-computation to greatly reduce the time for the reason of eliminating redundant exponential calculations, which are computationally expensive. This approach allows constant-time access to Gaussian values, replacing $O(n)$ exponential calculations with $O(1)$ table lookups.

In addition, I exploit the symmetry property of Gaussian, storing only half of the spatial kernel table and accessing elements using absolute indices. This reduces memory usage by 50% while maintaining computational accuracy.

Last but not least, I utilize NumPy's vectorized operations instead of Python loops to further improve the efficiency of my code, such as `np.roll` and `np.prod`. These vectorized operations execute in optimized C code rather than Python's interpreter, providing substantial performance gains.

With these techniques and algorithms, I reduce the computation time from few seconds to about 0.25 seconds, which is a huge difference once the image size or pixel number are extremely large.