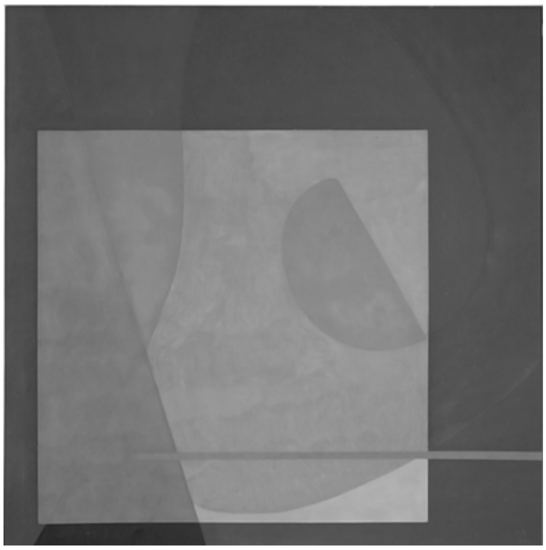
Computer Vision Homework 1

資工四 B05902115 陳建丞

- **Conventional RGB2GRAY conversion**

  $Y = 0.299R + 0.587G + 0.114B$

  Simply apply `np.dot()` to process the formula.

1a.png	Result
	

1b.png	Result
	

1c.png	Result
	

- **Joint bilateral filter**

  First, Check if the guidance image is a single channel image or a colored image. Next, use `cv2.copyMakeBorder()` to expand the border of guidance image and input image with type `BORDER_REFLCET`. And just follow the formula in the homework slide.

$$F^T(I) = \frac{\sum_{q\in\Omega_p} G_s(p,q)G_r(T_p,T_q)I_q}{\sum_{q\in\Omega_p} G_s(p,q)G_r(T_p,T_q)}$$

  Since the spatial kernel values are the same for every pixel, I just have to calculate one time and store the value window. Follow the formula below.

$$G_s(p,q) = e^{-\frac{(x_p-x_q)^2+(y_p-y_q)^2}{2\sigma_s^2}}$$

  And for the range kernel, applying the corresponding formula for single channel image and colored image. In this part, I use `np.multiply` and `[:]` of np array to implement some matrix operation. Follow the formula below.

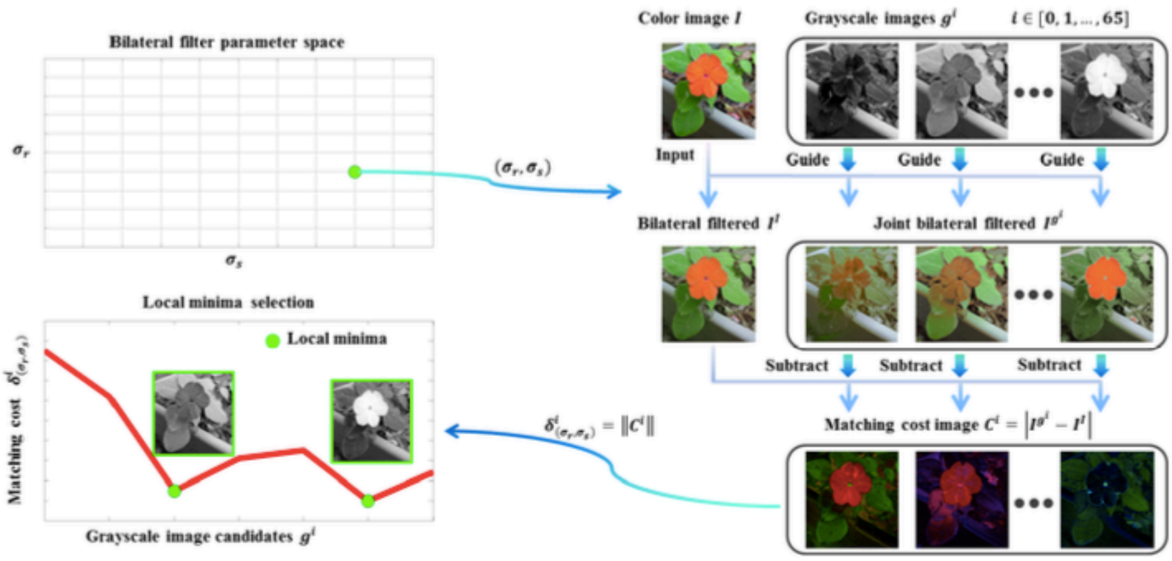
  - *single channel*

$$G_r(T_p,T_q) = e^{-\frac{(T_p-T_q)^2}{2\sigma_r^2}}$$

  - *colored*

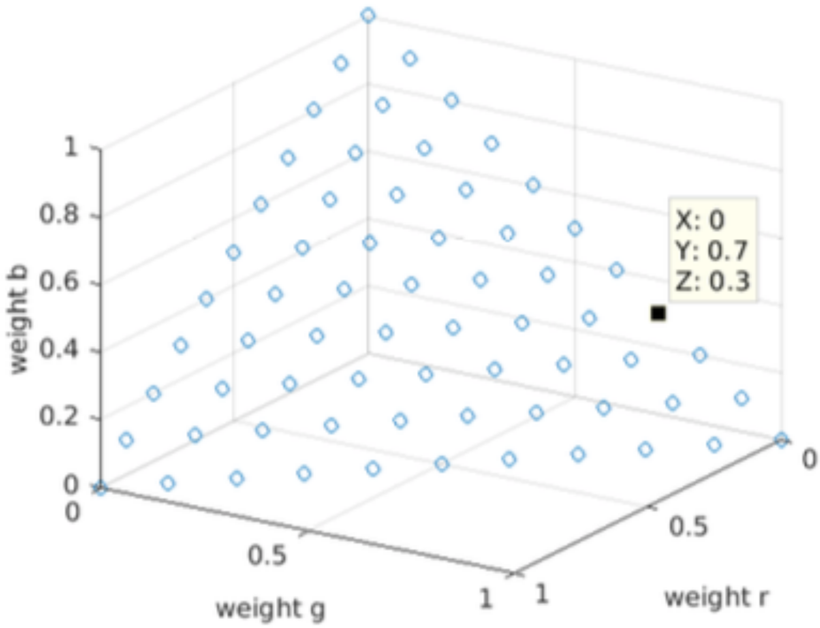
$$G_r(T_p,T_q) = e^{-\frac{(T_p^r-T_q^r)^2+(T_p^g-T_q^g)^2+(T_p^b-T_q^b)^2}{2\sigma_r^2}}$$

- **Local Minimum Selection**



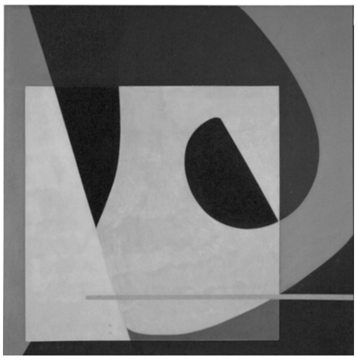
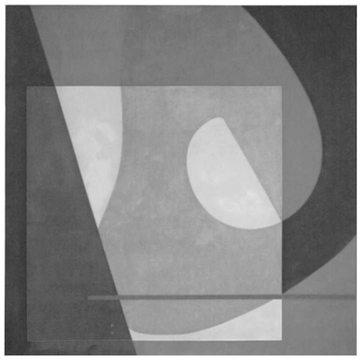
The whole process is the same as the picture above.

First, I run for loops to find all 66 possible combinations of (w_r, w_g, w_b) where $w_r, w_g, w_b >= 0$ and $w_r + w_g + w_b = 1$. Next, feed all the 66 candidates into my `Rgb2gray()` funtion to generate gray scale image as guidance for the joint bilateral filter. After filtering, compute the cost between bilateral filtered image and joint bilateral filtered image, which is $|I^{g^i} - I^I|$.



Each candidate (w_r, w_g, w_b) has at most 6 neighbors (w'_r, w'_g, w'_b) who has a 0.1 difference in one of (w_r, w_g, w_b). If the cost of (w_r, w_g, w_b) is the minimum among the its neighbors, then it's a local minum. And the corresponding value in vote table will be added by 1 if it's a local minumum. After processing this all $\sigma_S \in \{1, 2, 3\}$ and $\sigma_r \in \{0.05, 0.1, 0.2\}$ and all the voting, I use `np.max()` to find the max vote in the vote table and use `np.where()` to get the index of them, which stands for (w_r, w_g, w_b).

- **Result**

1a.png	$(w_r, w_g, w_b) = (0.0, 0.0, 1.0)$	$(w_r, w_g, w_b) = (1.0, 0.0, 0.0)$
		
Vote	9	9

1b.png	$(0.0, 0.9, 0.1)$	$(0.9, 0.0, 0.1)$	$(0.0, 0.2, 0.8)$
			
Vote	2	2	1

1c.png	$(0.0, 1.0, 0.0)$	$(0.7, 0.3, 0.0)$	$(1.0, 0.0, 0.0)$
			
Vote	3	3	3

  - Note: If there are candidates with same vote number, I will pick the one with smaller w_r. If w_r are the same, then pick the one with smaller w_g.

- **Requirement**
  - `cv2`
  - `numpy`
  - `argparse`
  - Run code `python joint_bilateral_filter.py -i input_path`