Assignment 1:

Advanced Color-to-Gray Conversion

Computer Vision

National Taiwan University

Fall 2019

Color Conversion

- RGB2YUV
  - Read https://en.wikipedia.org/wiki/YUV for more details

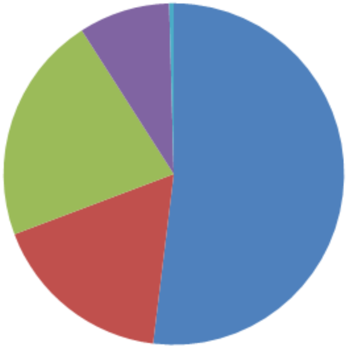
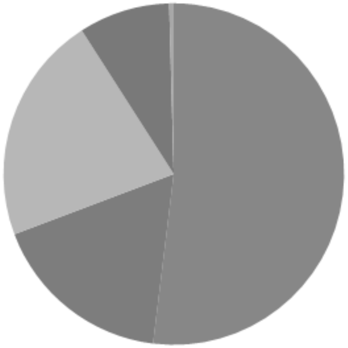
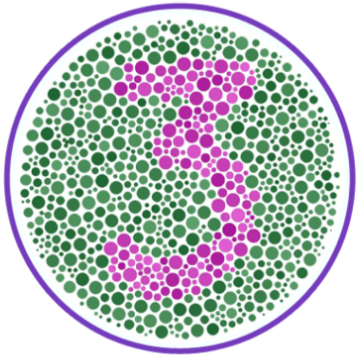
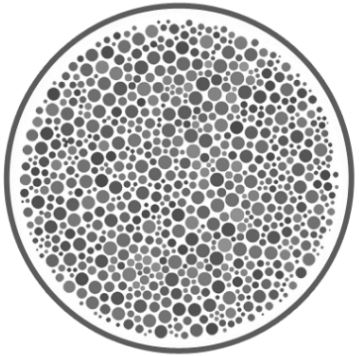
$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}.$$

- Many vision systems only take the Y channel (luminance) as input to reduce computations

RGB to Gray

Problems

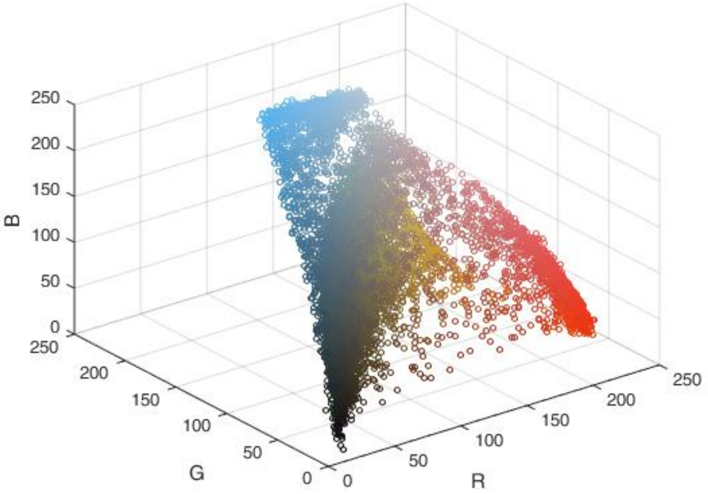






What happened?

- Dimensionality reduction

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

- Another view:
  - The conversion is actually a plane equation! All colors on the same plane are converted to the same grayscale value.



Finding a better conversion

- The general form of linear conversion:

$$Y = w_r \cdot R + w_g \cdot G + w_b \cdot B$$

$$w_r, w_g, w_b >= 0$$

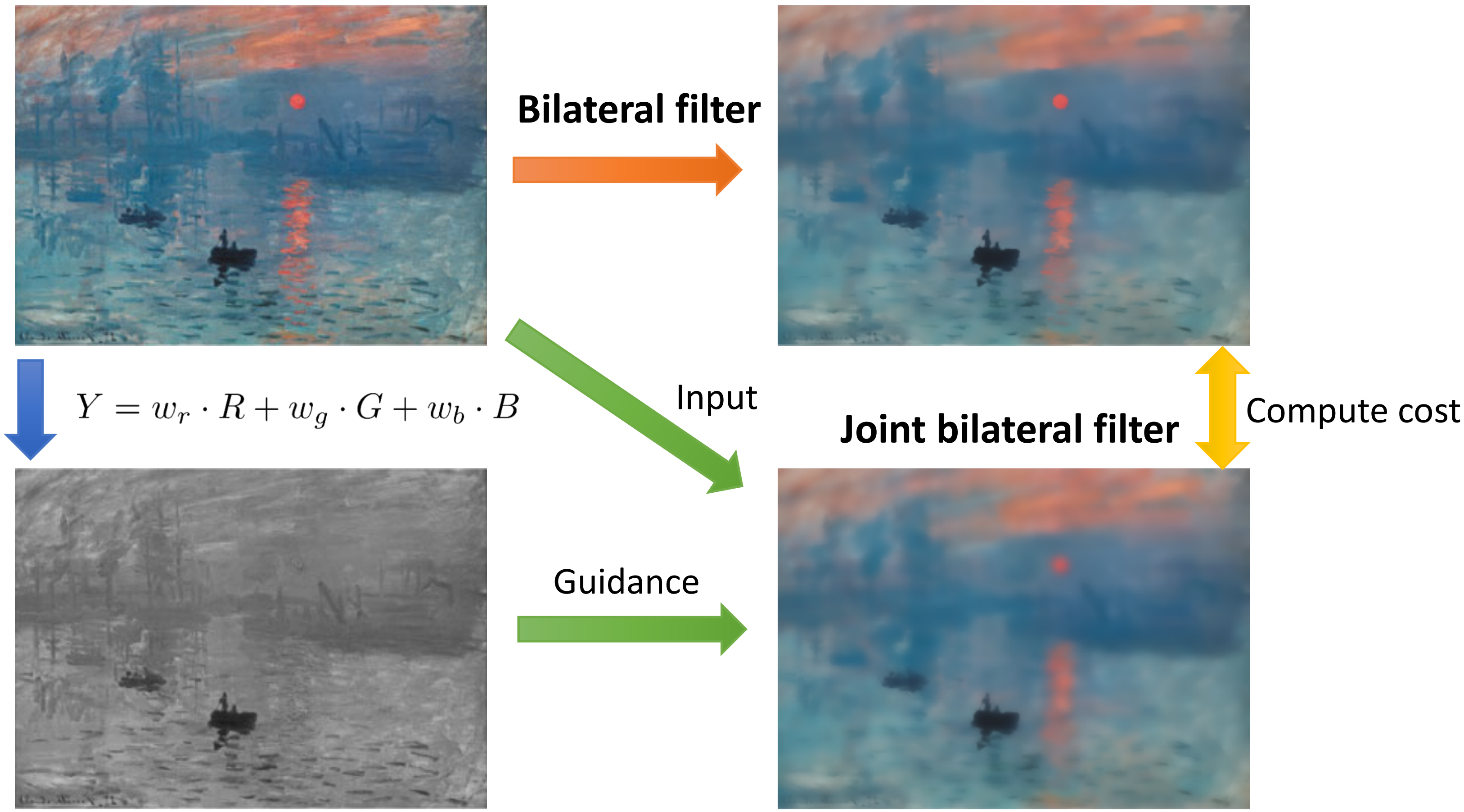
$$w_r + w_g + w_b = 1$$

- Let's consider the quantized weight space $w \in \{0, 0.1, 0.2, ..., 1\}$
  - For example: $(w_r, w_g, w_b) = (0, 0, 1)$
    $(w_r, w_g, w_b) = (0, 0.1, 0.9)$
  - Given a color image, a set of weight combination corresponds to a grayscale image candidate.
  - We are going to identify which candidate is better!

Yibing Song, Linchao Bao, Xiaobin Xu, and Qingxiong Yang. **Decolorization: is rgb2gray() out?**. In *SIGGRAPH Asia 2013 Technical Briefs*

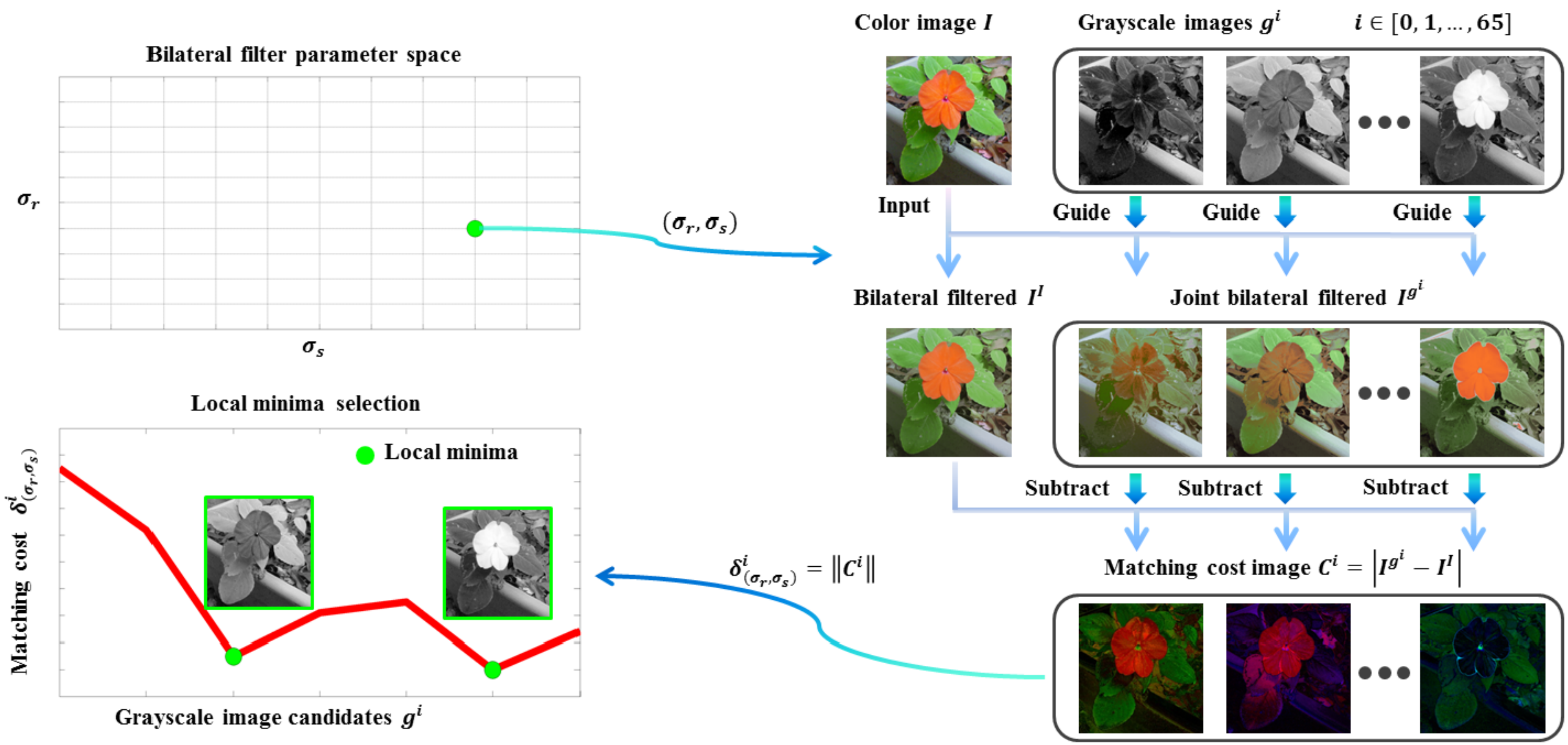
Measuring the perceptual similarity

- Joint bilateral filter (JBF) as the similarity measurement



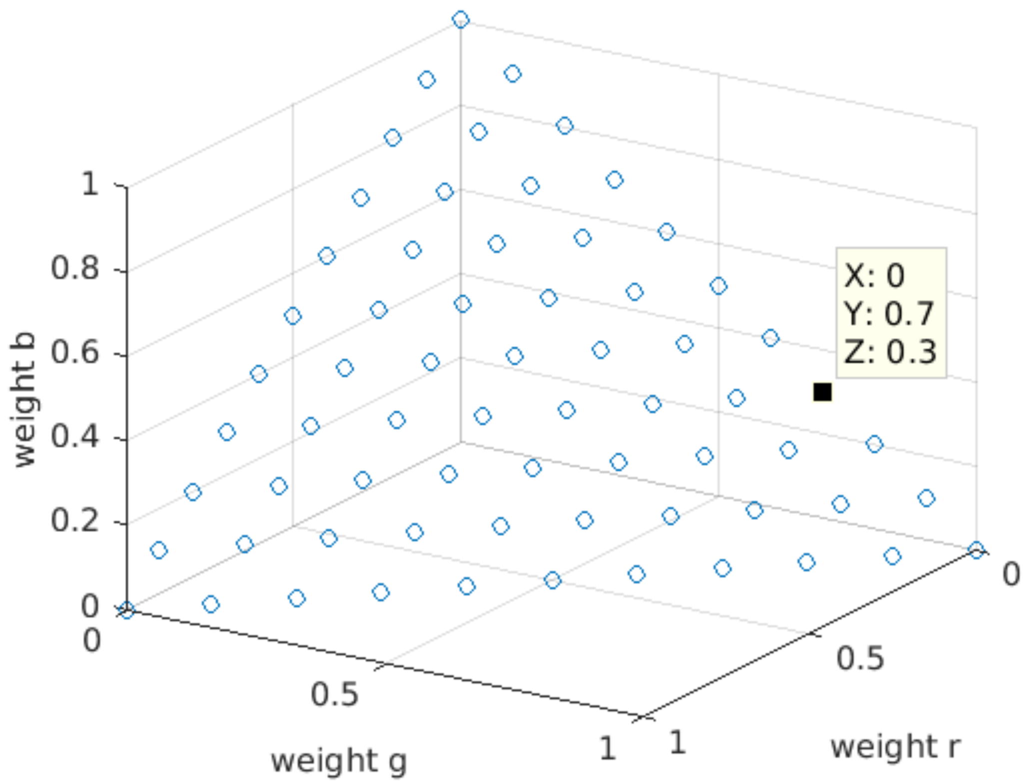
Measuring the perceptual similarity

- Joint bilateral filter (JBF) as the similarity measurement



Measuring the perceptual similarity

- Find local minimum
  - The actual weight space looks like this:

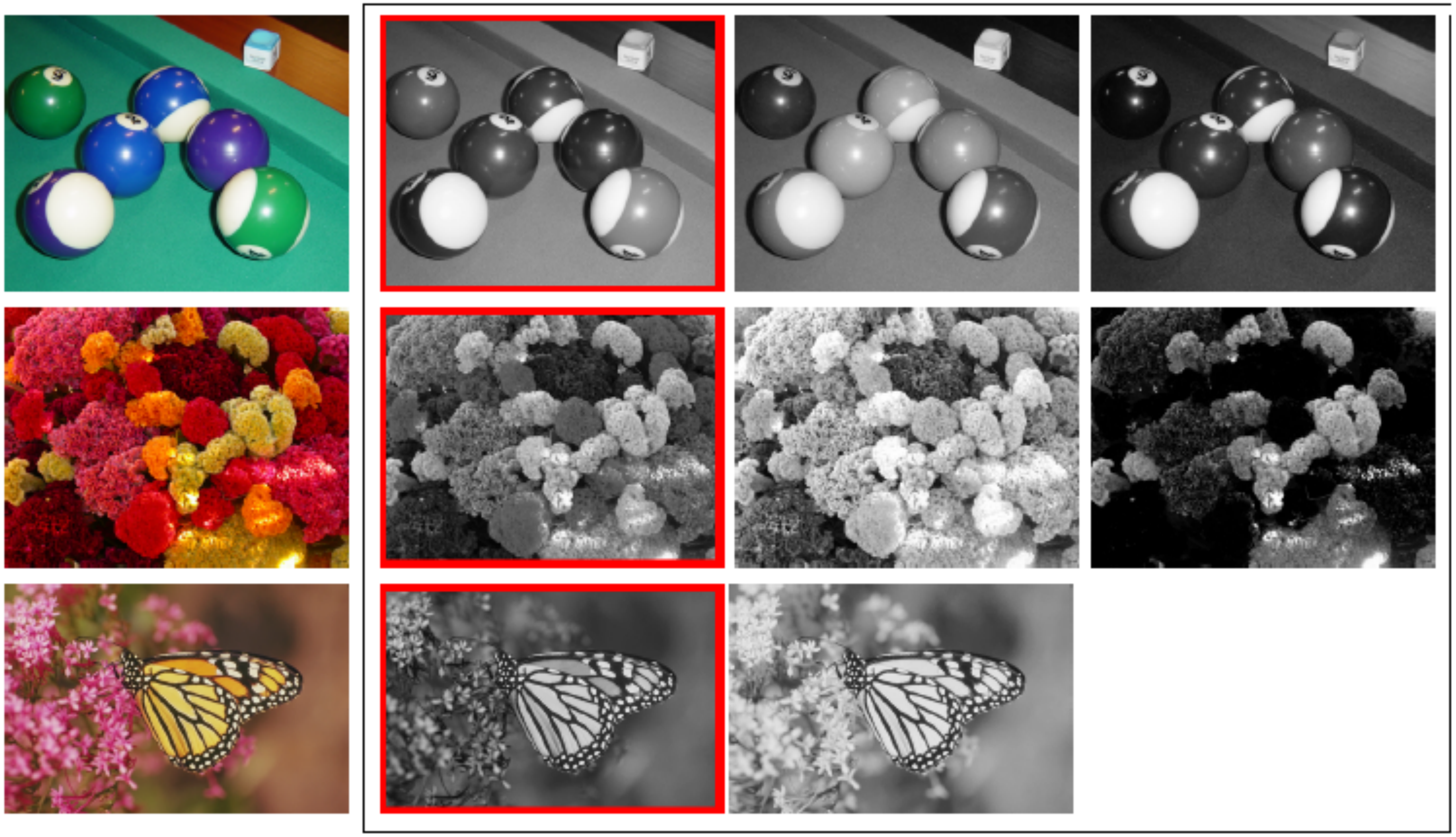


$$w_r, w_g, w_b >= 0$$

$$w_r + w_g + w_b = 1$$

Multiple Local Minima

- Keep the 3 most voted



Color Image Guided Bilateral Filter

- Given T as the guidance, the bilateral filter is written as:

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

- If T is a single-channel image:

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

- If T is a color image:

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

- Note : We should use the pixel values between 0 and 1 to construct the range kernel.

Color Image Guided Bilateral Filter

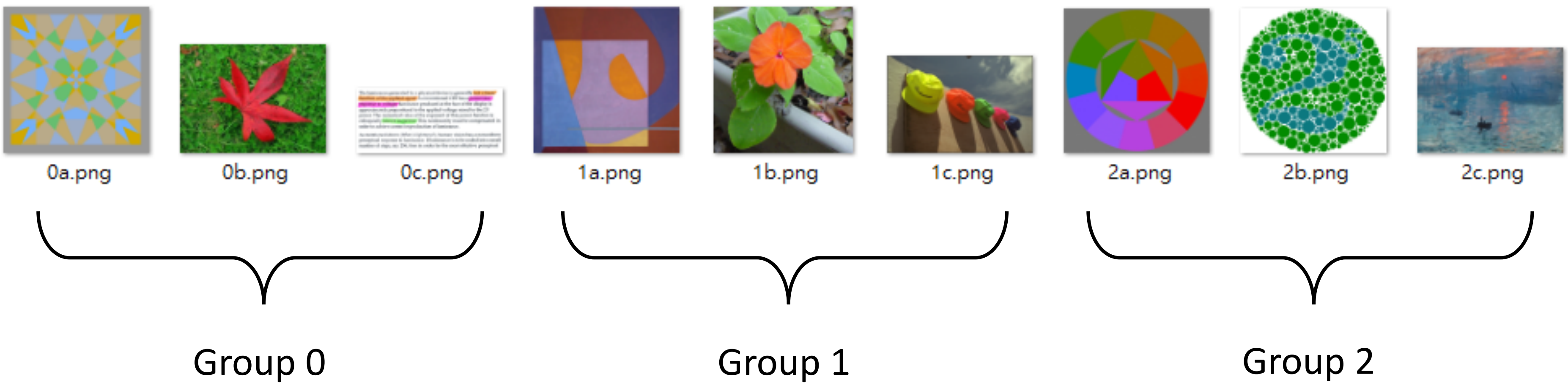
- For the spatial kernel :

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

- How to calculate the window size of kernels?
  - r = $3\sigma_S$
  - Window size = 2r + 1

Assignment Description

- Test images
  - 學號末三碼除以三之餘數



Assignment Description

- Implement the conventional rgb2gray conversion
- Implement the joint bilateral filter
- Implement the advanced rgb2gray described above
    - Quantize the weight space as in p6 (hint: totally 66 combinations)
    - Consider the 9 bilateral parameters $\sigma_s \in \{1, 2, 3\}$ and $\sigma_r \in \{0.05, 0.1, 0.2\}$
    - Find the cost local minima on the 2D plane $w_r + w_g + w_b = 1$
    - Vote the candidates for each set of bilateral parameter
    - Return the top 3 most voted candidates for each input image

Submission

- Code: *.py (Python 3.5+)
- A PDF report, containing
  - Your student ID, name
  - Describe how you design your joint bilateral filter
  - Describe how you implement the local minima selection
  - Show your input/output images and the corresponding weight combinations
    - Ex. 0a.png, 0a_gray.png (conventional rgb2gray conversion),
      0a_y1.png,0a_y2.png,0a_y3.png (the advanced rgb2gray)
  - Any other trick you want to share or comments to this assignment are welcome
- Compress all above files in a zip file named StudentID.zip
  - e.g. R07654321.zip
- Submit to CEIBA
- Deadline: 10/15 11:00 pm

Note

- If you use numpy.pad for padding, you should use the ”symmetric” argument; if you use opencv, you should use BORDER_REFLECT.
- The precision in kernel is float64.

Code Evaluation

- We provide a template class in "joint_bilateral_filter.py"

```python
class Joint_bilateral_filter(object):
    def __init__(self, sigma_s, sigma_r, border_type='reflect'):

        self.border_type = border_type
        self.sigma_r = sigma_r
        self.sigma_s = sigma_s

    def joint_bilateral_filter(self, input, guidance):
        ## TODO
        return output
```

Use this argument to decide the type of padding for input images.

Code Evaluation

We will run eval.py to test your function and the execution time.

```
from joint_bilateral_filter import Joint_bilateral_filter

def main():
    parser = argparse.ArgumentParser(description='JBF evaluation')
    parser.add_argument('--sigma_s', default=3, type=int, help='sigma of spatial kernel')
    parser.add_argument('--sigma_r', default=0.1, type=float, help='sigma of range kernel')
    parser.add_argument('--input_path', default='./testdata/ex.png', help='path of input image')
    parser.add_argument('--gt_bf_path', default='./testdata/ex_gt_bf.png', help='path of gt bf image')
    parser.add_argument('--gt_jbf_path', default='./testdata/ex_gt_jbf.png', help='path of gt jbf image')

    args = parser.parse_args()

    img = cv2.imread(args.input_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    guidance = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # create JBF class
    JBF = Joint_bilateral_filter(args.sigma_s, args.sigma_r, border_type='reflect')

    bf_out = JBF.joint_bilateral_filter(img_rgb, img_rgb).astype(np.uint8)
    jbf_out = JBF.joint_bilateral_filter(img_rgb, guidance).astype(np.uint8)

    bf_gt = cv2.cvtColor(cv2.imread(args.gt_bf_path), cv2.COLOR_BGR2RGB)
    jbf_gt = cv2.cvtColor(cv2.imread(args.gt_jbf_path), cv2.COLOR_BGR2RGB)

    bf_error = np.sum(np.abs(bf_out-bf_gt))
    jbf_error = np.sum(np.abs(jbf_out-jbf_gt))
    print('%d %d'%(bf_error, jbf_error))
```

Bilateral Filter

Joint Bilateral Filter

Code Evaluation

- We give you an example image and the ground truth images for your self-checking.
  (ex.png , ex_gt_bf.png, ex_gt_jbf.png)
- You can simply run "**python3 eval.py**" to test your function, and the errors which will be printed on the screen must be zeros.
- For testing your code on our computer, we will assign different arguments for sigma_s, sigma_r, input_path and the ground truth paths.

Grading (Total 15%)

- Report : 10%
- Code : 5%
  - If the errors are all zeros on our test image
    - If it runs within 1 min: 5 %
    - If it runs within 10 mins but larger than 1 min : 3 %
    - Others : 0 %
  - Others : 0 % (TBD)

TA information

- 吳禹澄 (Wu, Yu-Cheng)
  e-mail : yuchengwu@media.ee.ntu.edu.tw
  TA time : Thu. 13:30 - 15:00
  Location : 明達-431

- 劉致廷 (Liu, Chih-Ting)
  e-mail : jackieliu@media.ee.ntu.edu.tw
  TA time : Thu. 11:00 - 12:30
  Location : 明達-431