Assignment 1:

Image Corner Detection and Filtering

Computer Vision

National Taiwan University

Spring 2021

Outline

Part 1: Image Corner Detection

- Implement Harris corner detector

Part 2: Image Filtering

- Implement bilateral filter
- Advanced color-to-gray conversion

Part 1:

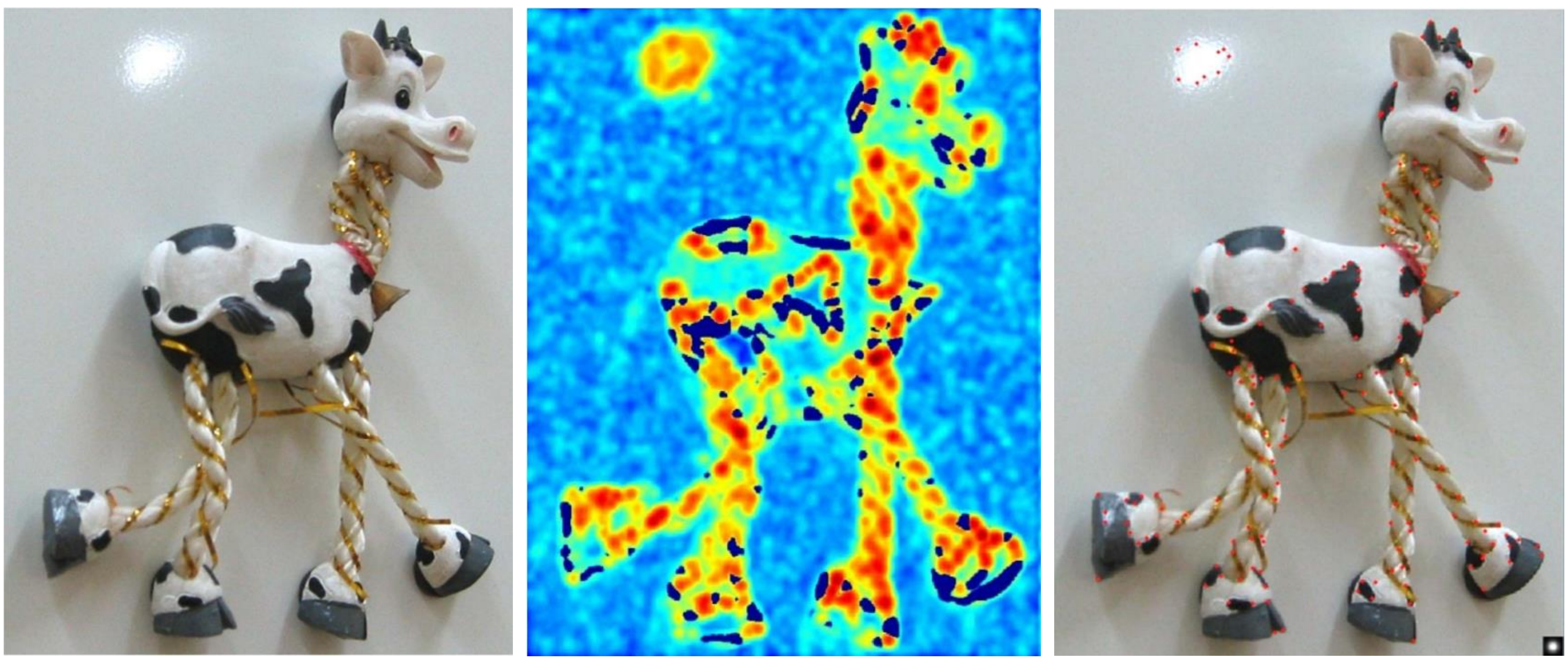
Image Corner Detection

Harris Corner Detector

A COMBINED CORNER AND EDGE DETECTOR

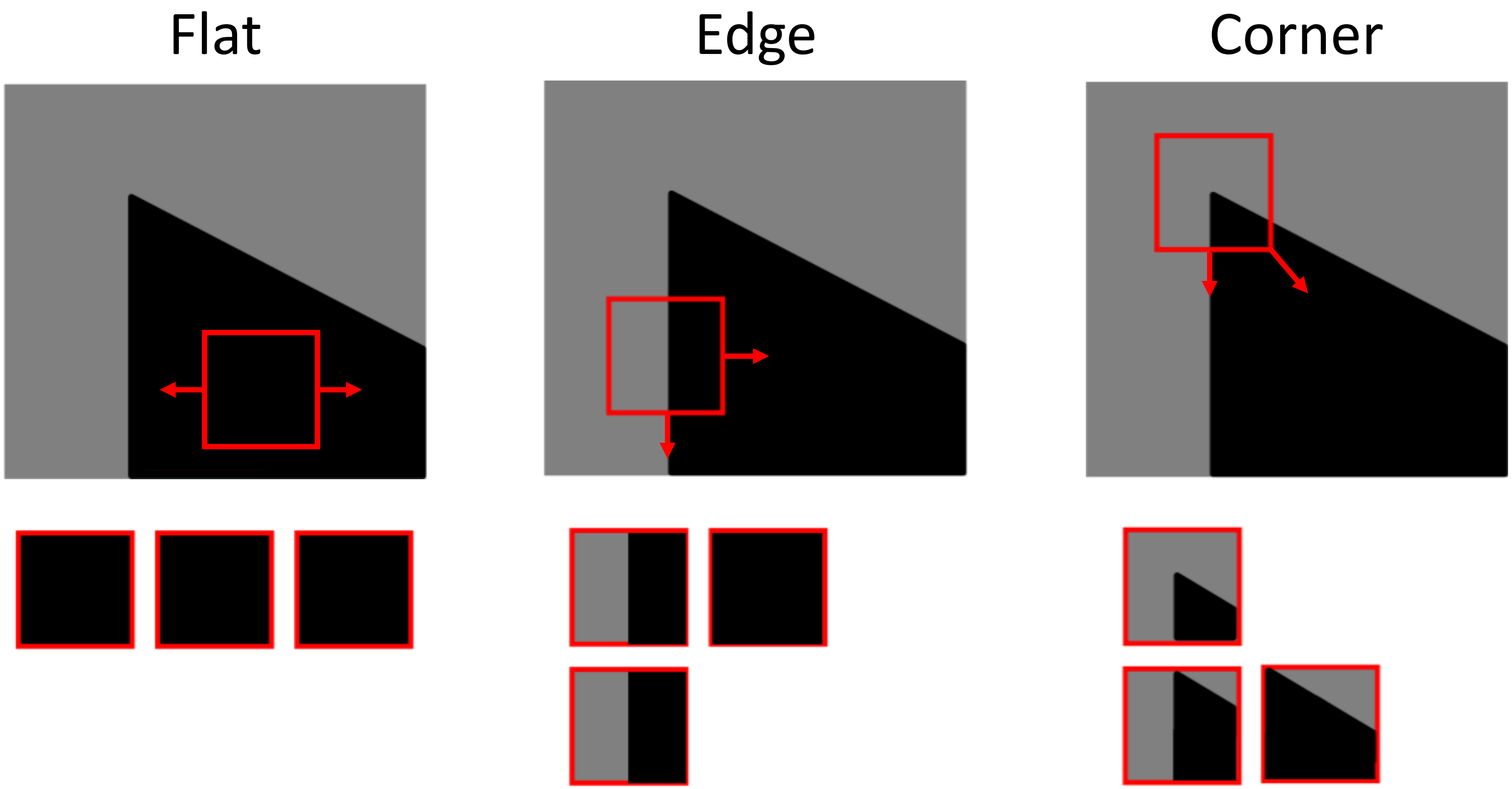
Chris Harris & Mike Stephens

Plessey Research Roke Manor, United Kingdom



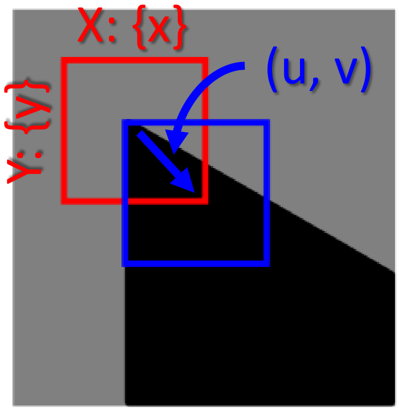
Moravec Corner Detector

- For a corner, shifting a window in **any direction** should give a **large change in intensity**



Moravec Corner Detector

- For a given patch (x, y) and displacement (u, v), the difference function can be written as



$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \, [\underbrace{I(x+u, y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

- Corner response for the center pixel is defined as

$$R = \min_{(u,v)} E(u,v)$$

(u, v) = {(1,0), (1,1), (0,1), (-1, 1)} for Moravec corner detector

Harris Corner Detector

- Moravec model only considers a set of shifts at every 45 degree, while Harris model considers all small shifts by using Taylor's expansion.

$$E(u,v) = \sum_{x,y} w(x,y) \quad [I(x+u, y+v) - I(x,y)]^2$$

Small motion assumption + Taylor Series expansion

*See reference for more details

Derivative of intensity along x or y axis

$$E(u,v) \approx [\,u \quad v\,]\, M \begin{bmatrix} u \\ v \end{bmatrix} \qquad M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

Definition of I_x and I_y in this assignment:

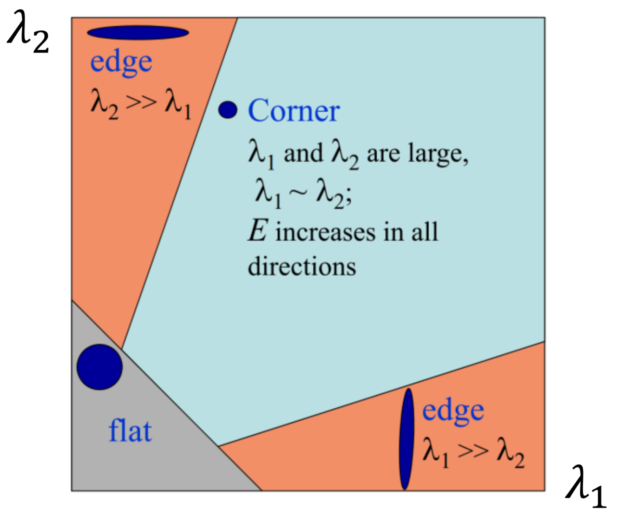
62	75	38
26	54	97
57	27	5

For the center pixel:

$$I_x = 1 \cdot 26 + (-1) \cdot 97 = -71$$
$$I_y = 1 \cdot 75 + (-1) \cdot 27 = \ \ 48$$

Harris Corner Detector

- Important property of 2x2 matrix M
  - let λ_1 and λ_2 as eigenvalues of M



  - λ_1 and λ_2 are small, the region is flat
  - $\lambda_1 \gg \lambda_2$ or vice versa, the region is edge
  - λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, the region is a corner

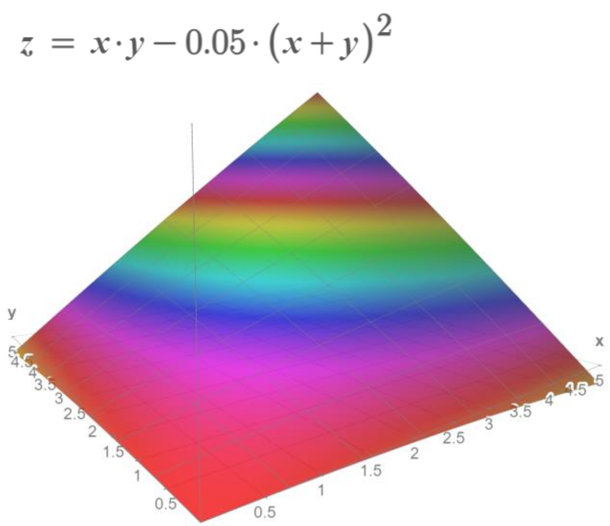
Harris Corner Detector

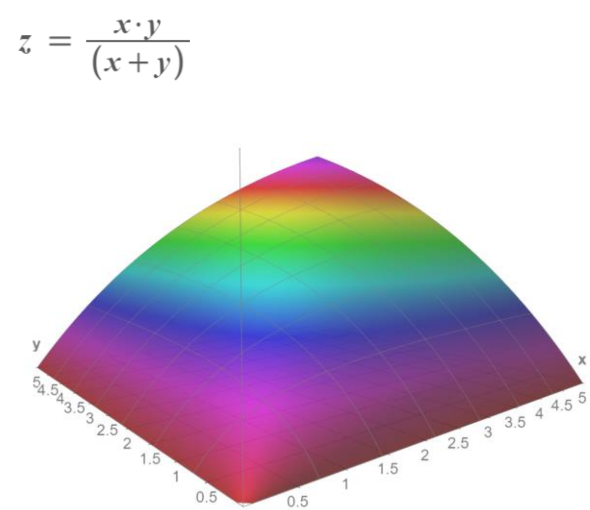
Principle:
For any matrix M and its eigenvalues λ_i

$$\mathrm{trace}(M) = \sum \lambda_i$$
$$\det(M) = \prod \lambda_i$$

- Harris corner response equation
  - $R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{trace}(M)^2$
  - $R = \lambda_1 \lambda_2 \,/\, (\lambda_1 + \lambda_2) = \det(M)/\mathrm{trace}(M)$

We use this response equation in this assignment





Harris Corner Detector

```
# Step 1: Smooth the image by Gaussian kernel
# - Function: cv2.GaussianBlur (kernel = 3, sigma = 1.5)
```

```
# Step 2: Calculate Ix, Iy (1st derivative of image along x and y axis)
# - Function: cv2.filter2D (kernel = [[1.,0.,-1.]] for Ix or [[1.],[0.],[-1.]] for Iy)
```

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

```
# Step 3: Compute Ixx, Ixy, Iyy (Ixx = Ix*Ix, ...)
```

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

```
# Step 4: Compute Sxx, Sxy, Syy (weighted summation of Ixx, Ixy, Iyy in neighbor pixels)
# - Function: cv2.GaussianBlur (kernel = 3, sigma = 1.)
```

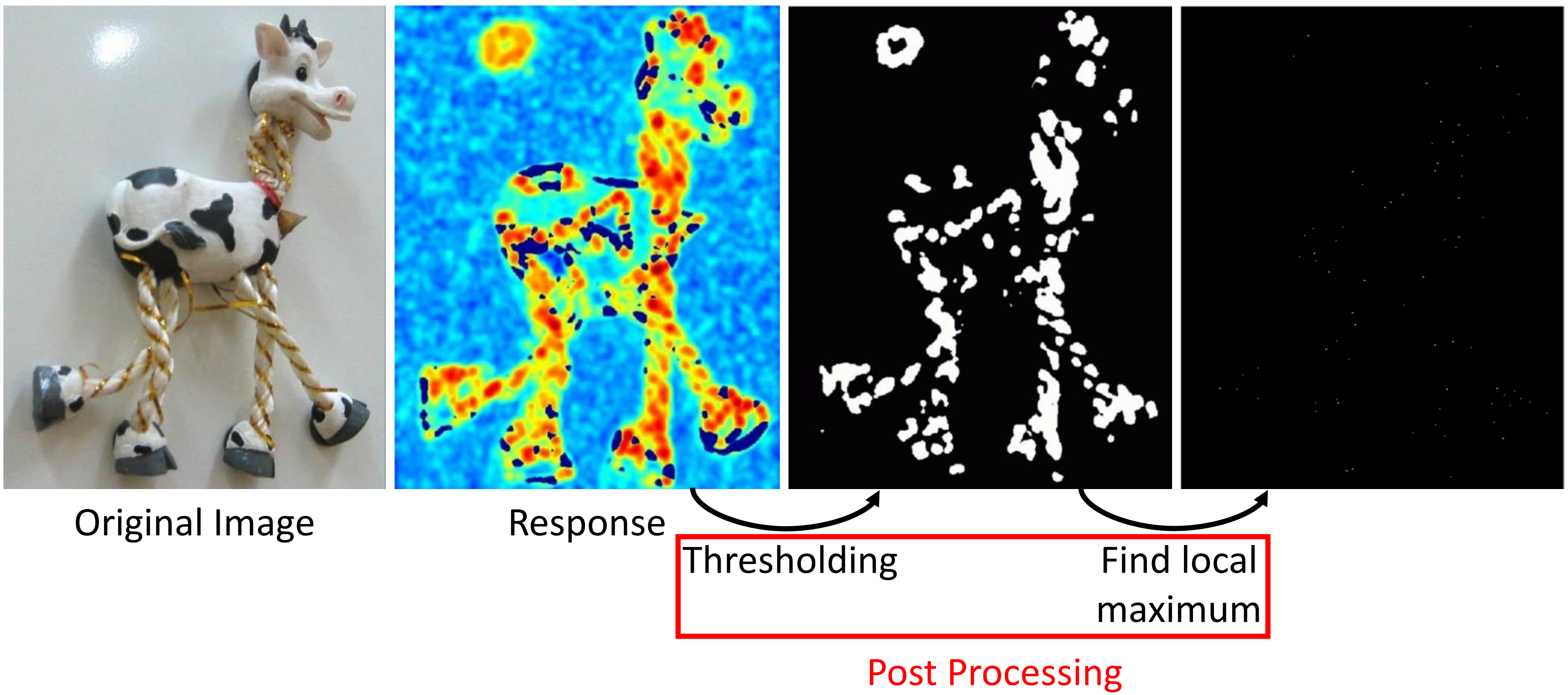
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix}$$

```
# Step 5: Compute the det and trace of matrix M (M = [[Sxx, Sxy], [Sxy, Syy]])
```

```
# Step 6: Compute the response of the detector by det/(trace+1e-12)
```

Harris Corner Detector

- Post Processing



Harris Corner Detector

- Threshold
  - The pixels whose responses > threshold would be selected as candidates
- Find local maximum
  - The candidates whose responses > all its 5x5 neighbors (24 pixels totally) are recognized as final corner
    - Need zero padding (width of) 2 to maintain the output size

Assignment Description

- part1/main.py
  - Read image, execute Harris corner detector, visualize results, … etc.
- part1/HCD.py

```
class Harris_corner_detector(object):
    def __init__(self, threshold):
        self.threshold = threshold

    def detect_harris_corners(self, img):
        ### TODO ####
```

  - Implement Harris corner detector, including two functions
  - detect_harris_corners: compute the corner response for input grayscale image
  - post_processing: detect the corner for the giving response map
    - The output format should be "list of list" (please refer p15)

Assignment Description

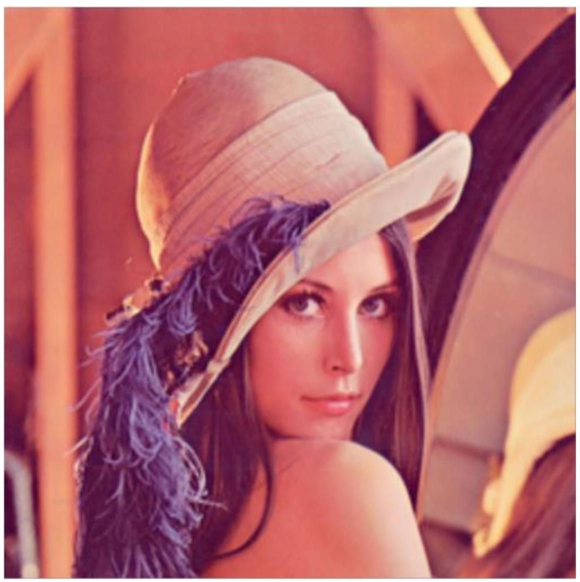
- part1/eval.py (DO NOT EDIT this file)
  - Evaluate the correctness of the output of Harris corner detector

```
def main():
    parser = argparse.ArgumentParser(description='evaluation function of Harris corner detector')
    parser.add_argument('--threshold', default=100., type=float, help='threshold value to determine corner')
    parser.add_argument('--image_path', default='./testdata/ex.png', help='path to input image')
    parser.add_argument('--gt_path', default='./testdata/ex_gt.pkl', help='path to ground truth pickle file')
    args = parser.parse_args()

    img = cv2.imread(args.image_path)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY).astype(np.float64)

    # create HCD class
    HCD = Harris_corner_detector(args.threshold)

    response = HCD.detect_harris_corners(img_gray)
    result = HCD.post_processing(response)
```

  - TAs will run this file to score upload code.
  - When testing your code, we will assign different arguments, like threshold, and corresponding ground truth file.

Assignment Description

- part1/testdata/
  - Include 1 example image (w/ ground truth) + other 3 images (w/o gt)

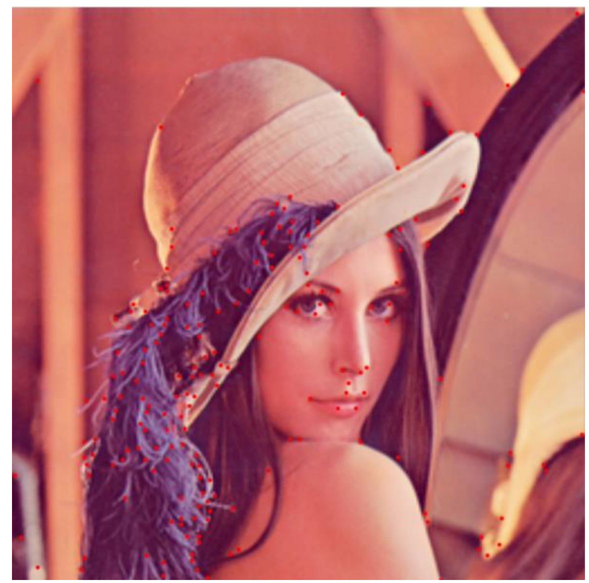


Example Image

Y X

```
[[  2 252]
 [ 27 227]
 [ 32  11]
 [ 34 221]
 [ 37 255]
 [ 38 147]
 [ 42 185]
 [ 47 155]
 [ 53  66]
 [ 55 195]
 [ 56 207]
 [ 63 168]
 [ 64 183]
 [ 71 172]
 [ 79 201]
 [ 84 124]
```

Ground truth (pickle file)



Example visualization results

Assignment Description

- Recommended steps
  - Implement Harris corner detector in HCD.py
  - Use eval.py to evaluate your HCD.py
    - By

```
python3 eval.py --image_path 'testdata/ex.png' --gt_path 'testdata/ex_gt.pkl'
```

    - The Result and Ground truth unmatch should be both 0, as

```
[Error] Result unmatch: 0
[Error] Ground truth unmatch: 0
```

  - Finish remaining code in main.py if needed

Reference

- Harris, Christopher G., and Mike Stephens. "A combined corner and edge detector", 1988.
- NTU VFX course slide
  - https://www.csie.ntu.edu.tw/~cyy/courses/vfx/19spring/lectures/handouts/lec06_feature.pdf
- OpenCV-Python Tutorial: Harris Corner Detection
  - https://docs.opencv.org/master/dc/d0d/tutorial_py_features_harris.html
- Wikipedia: Corner detection
  - https://en.wikipedia.org/wiki/Corner_detection

Supplementary:

Advanced Color-to-Gray Conversion

Color Conversion

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

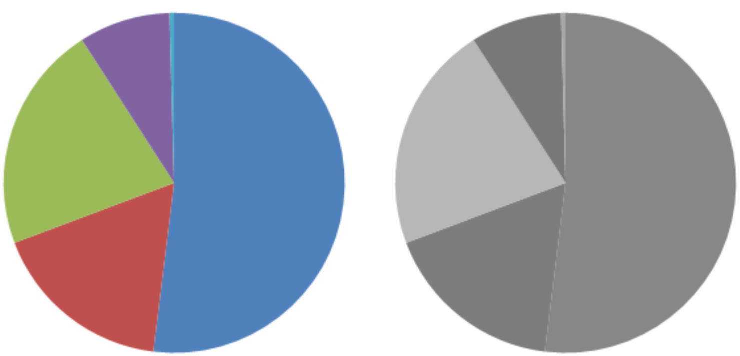
$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

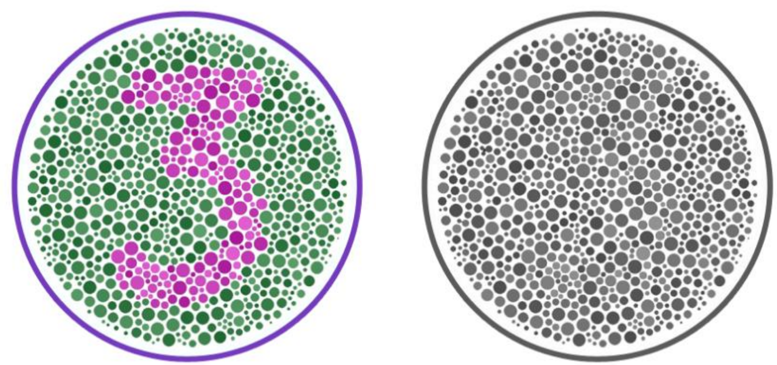
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}.$$

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

RGB to Gray

Problems



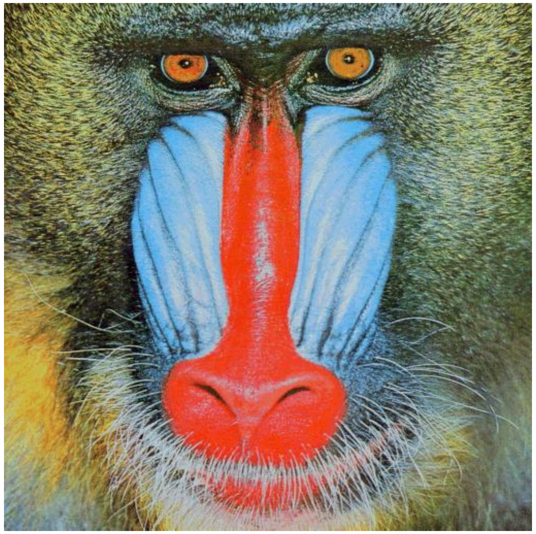


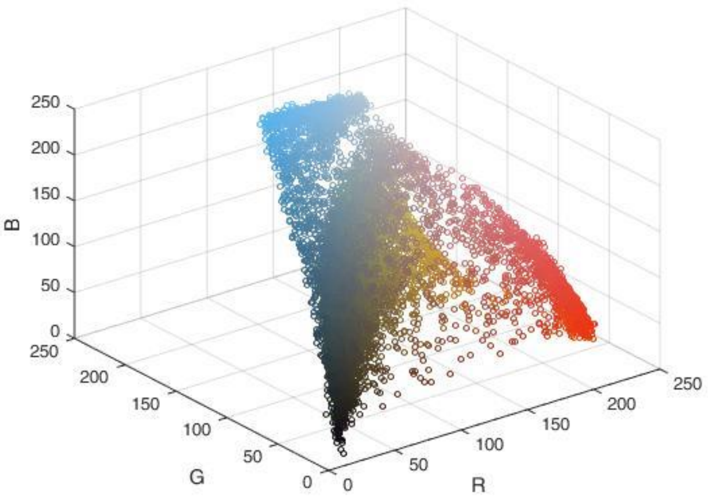
What happened?

- 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:

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

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

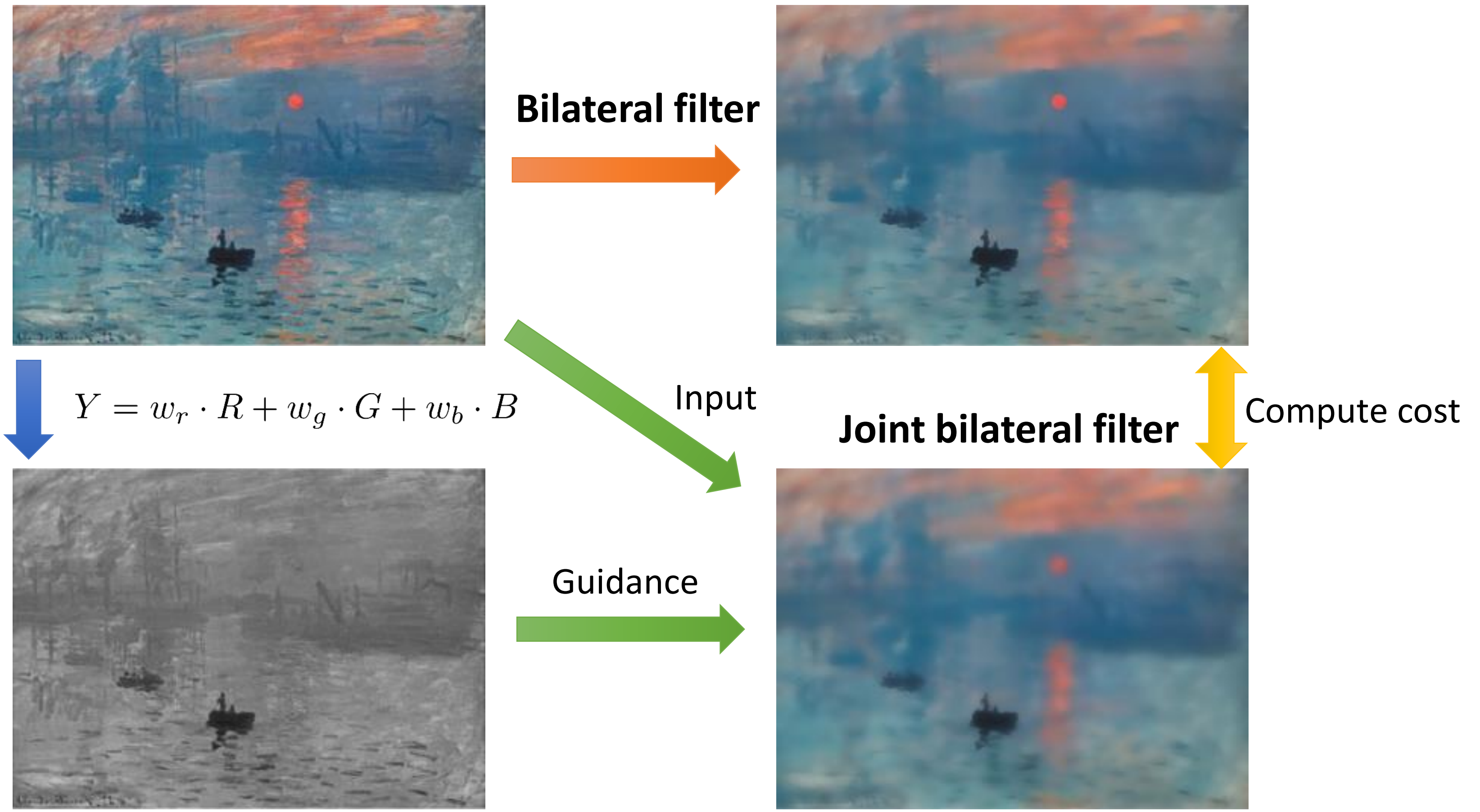
$$w_r + w_g + w_b = 1$$

- 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*

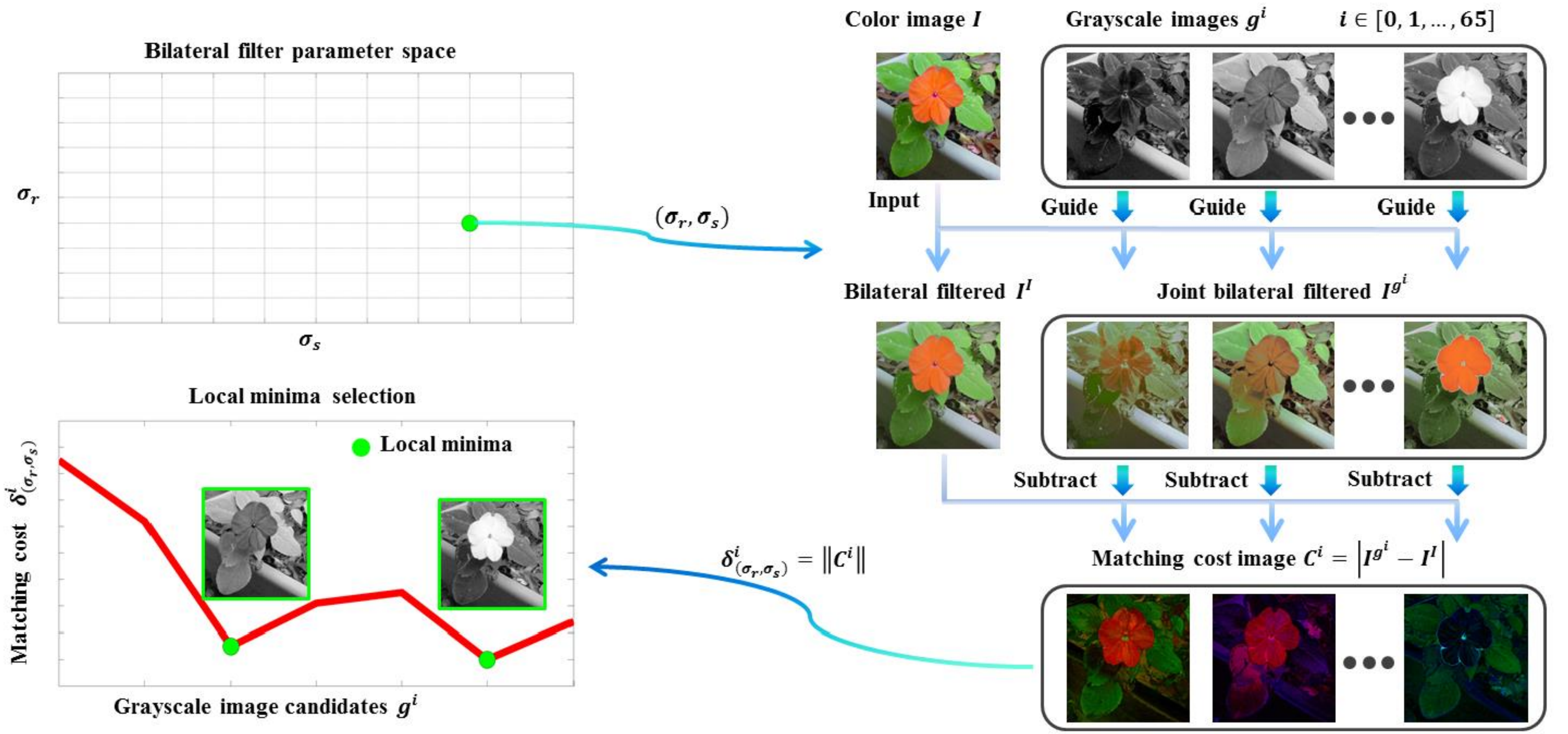
Measuring the perceptual similarity

- Joint bilateral filter (JBF) as the similarity measurement



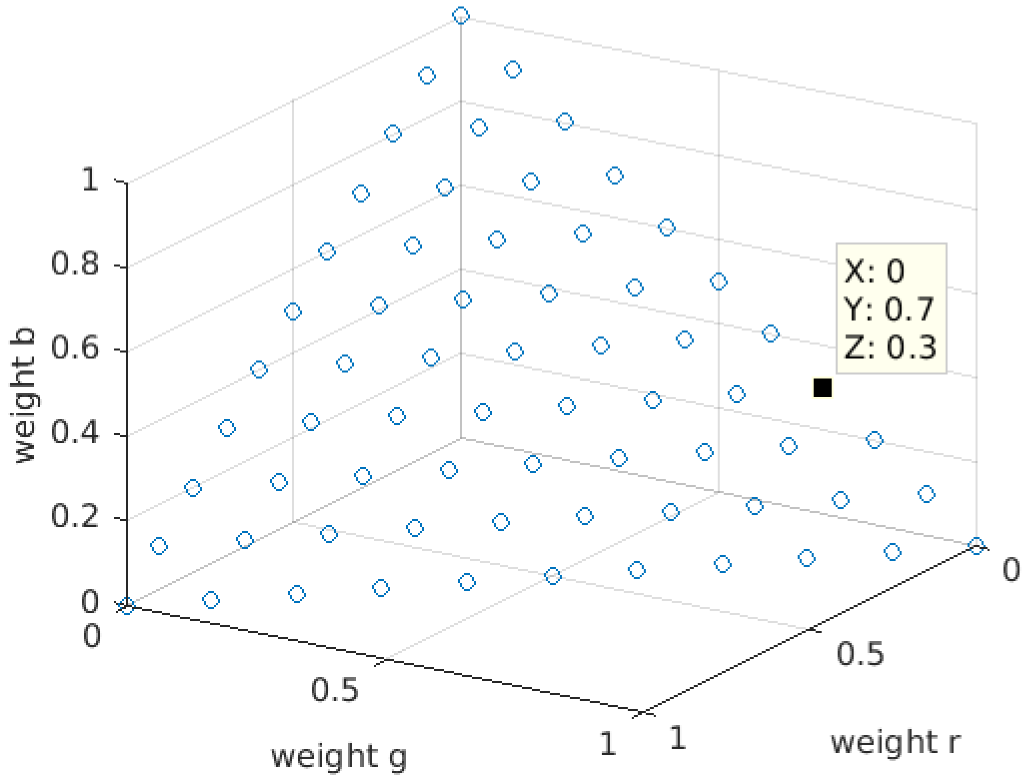
Measuring the perceptual similarity

- Joint bilateral filter (JBF) as the similarity measurement



Measuring the perceptual similarity

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



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

$$w_r + w_g + w_b = 1$$

Part 2:

Image Filtering

Bilateral Filter

- Given input image I and guidance T, the bilateral filter is written as:

$$I'_p = \frac{\sum_{q \in \Omega_p} G_s(p, q) \cdot G_r(T_p, T_q) \cdot I_q}{\sum_{q \in \Omega_p} G_s(p, q) \cdot G_r(T_p, T_q)}$$

  - I_p: Intensity of pixel p of original image I
  - I'_p: Intensity of pixel p of filtered image I'
  - T_p: Intensity of pixel p of guidance image T
  - Ω_p: Window centered in pixel p
  - G_s: Spatial kernel
  - G_r: Range kernel

Bilateral Filter

- For the spatial kernel G_s:

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

- For the range kernel G_r:
  - 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:

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

  - Pixel values should be normalized to [0, 1] (divided by 255) to construct range kernel.

Assignment Description

- part2/main.py
  - Read image, execute joint bilateral filter, read setting file, select the best grayscale conversion… etc.
- part2/JBF.py
  - Implement joint bilateral filter

```
class Joint_bilateral_filter(object):
    def __init__(self, sigma_s, sigma_r):
        self.sigma_r = sigma_r
        self.sigma_s = sigma_s
        self.wndw_size = 6*sigma_s+1
        self.pad_w = 3*sigma_s

    def joint_bilateral_filter(self, img, guidance):
        BORDER_TYPE = cv2.BORDER_REFLECT
        padded_img = cv2.copyMakeBorder(img, self.pad_w, self.pad_w, self.pad_w, self.pad_w, BORDER_TYPE)
        padded_guidance = cv2.copyMakeBorder(guidance, self.pad_w, self.pad_w, self.pad_w, self.pad_w, BORDER_TYPE)

        ### TODO ###

        return np.clip(output, 0, 255).astype(np.uint8)
```

Define window size

Pad the input and guidance image

Output image should be in format of uint8

Assignment Description

- part2/eval.py (DO NOT EDIT this file)
  - Evaluate the correctness of the output of joint bilateral filter

```python
def main():
    parser = argparse.ArgumentParser(description='evaluation function of joint bilateral filter')
    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('--image_path', default='./testdata/ex.png', help='path to input image')
    parser.add_argument('--gt_bf_path', default='./testdata/ex_gt_bf.png', help='path to ground truth bf image')
    parser.add_argument('--gt_jbf_path', default='./testdata/ex_gt_jbf.png', help='path to ground trut jbf image')
    args = parser.parse_args()

    img = cv2.imread(args.image_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

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

    bf_out = JBF.joint_bilateral_filter(img_rgb, img_rgb).astype(np.uint8)
    t0 = time.time()
    jbf_out = JBF.joint_bilateral_filter(img_rgb, img_gray).astype(np.uint8)
    print('[Time] %.4f sec'%(time.time()-t0))
```

We will test your inference duration of joint bilateral filter.

  - TAs will run this file to score upload code.
  - When testing your code, we will assign different arguments, like σ_s and σ_r, and corresponding ground truth file.

Assignment Description

- part2/testdata/
  - One example image with bf and jbf ground truth
  - Two images with respective setting files

1.png

1_setting.txt

```
R,G,B
0.0,0.0,1.0
0.0,1.0,0.0
0.1,0.0,0.9
0.1,0.4,0.5
0.8,0.2,0.0
sigma_s,2,sigma_r,0.1
```

  - Setting file gives σ_s, σ_r and five kinds of gray conversion
  - You need to use those five and also original cv2 gray conversions (six in total) as guidance to run joint bilateral filter and compute the perceptual similarity.
    - Refer p24 and p25 for details (we use L1-norm as our cost function).
    - Note: need to cast the image into np.int32 to avoid overflow for subtraction.

Assignment Description

- Recommended steps
  - Implement joint bilateral filter in JBF.py
  - Use eval.py to evaluate your JBF.py
    - By

```
python3 eval.py --image_path './testdata/ex.png' --gt_bf_path './testdata/ex_gt_bf.png' --gt_jbf_path './testdata/ex_gt_jbf.png'
```

    - The error of bilateral and joint bilateral filter should be both 0

```
[Error] Bilateral: 0
[Error] Joint bilateral: 0
```

  - Finish remaining code in main.py if needed
  - Improve the inference speed of joint bilateral filter

Assignment Description

- About the speed test of JBF…
  - For fair comparison, you CAN ONLY use basic functions (e.g. cannot use cv2.filter2D, cv2.GaussianBlur) in JBF.py
  - Reference time of TA code
    - Intel Core i7-8700K CPU + 64GB RAM ⇒ 1.41 sec
    - Intel Core i9-7900X CPU + 128GB RAM ⇒ 1.04 sec
  - Some useful tips
    - Build look-up-table for both spatial and range gaussian kernels
    - Reduce the usage of for-loop to enhance parallel processing
      - We only use one for-loop (in range(1, window_size**2)) in entire bilateral filtering
      - Reference: "Unrolled Inner Product"

Submission

- Code: HCD.py and JBF.py (Python 3.5+)
  - Package: Python standard library, numpy, cv2, matplotlib
  - https://docs.python.org/3.5/library/
- Report: report.pdf
- Do NOT copy homeworks (including code and report) from others

- Put all above files in a directory (named StudentID) and compress the directory into zip file (named StudentID.zip)
  - e.g. After TAs run “unzip R07654321.zip”, it should create one directory named “R07654321”
- Submit to NTU COOL
- Deadline: 4/1 11:59 pm
  - Late policy: http://media.ee.ntu.edu.tw/courses/cv/21S/hw/cv2021_delay_policy.pdf

Report

- Your student ID, name
- Part1: Harris corner detector
  - Visualize the detected corner for 1.png, 2.png, 3.png (refer p15 as example)
  - Use three thresholds (i.e. 25, 50, 100) on 2.png and describe the difference
- Part2: Joint bilateral filter
  - For 1.png and 2.png:
    - Report the cost for each filtered image (by using 6 grayscale images as guidance)
    - Show original RGB image / two filtered RGB images and two grayscale images with highest and lowest cost (five images in total for each input image)
    - Describe the difference between those two grayscale images
  - Describe how you speed up the implementation of bilateral filter

Grading (Total 15%)

- Part 1 Code: 30%
  - 30%, no error (both result and GT unmatch = 0)
  - 0%, others
- Part 2 Code: 30%
  - 30%, runs within 5 mins and no error (both bf and jbf error = 0)
  - 0%, others
- Report : 30%
- Inference time: 10%
  - 10%, Top ~20%
  - 6%, Top ~50%
  - 3%, Top ~80%
  - 0%, others

TA information

- Tsai-Shien Chen (陳在賢)
  E-mail: tschen@media.ee.ntu.edu.tw
  TA time: Thu. 15:30 - 17:00
  Location: 博理 421

- Wen-Tsung Hsieh (謝汶璁)
  E-mail: wthsieh@media.ee.ntu.edu.tw
  TA time: Tue. 13:30 - 15:00
  Location: 博理 421

- Chih-Ting Liu (劉致廷)
  E-mail: jackieliu@media.ee.ntu.edu.tw
  TA time: Fri. 13:00 - 14:30
  Location: 博理 421