影像處理、電腦視覺及深度學習概論
(Introduction to Image Processing, Computer Vision and Deep Learning)

Homework 1

TA: 吳又成

Gmail: nckubot65904@gmail.com

Office Hour: 14:00~16:00, Mon.

10:00~12:00, Thu.

At CSIE 9F Robotics Lab.

Notice (1/2)

- Copying homework is strictly prohibited!! Penalty: Both individuals will receive a score of 0!!
- Due date => 09:00:00, 2024/11/05 (Tue.)

  Do not submit late, or the following points will be deducted:
  - Submit within seven days after the deadline, and your score will be reduced by half.
  - If you submit after this period, you will receive a score of 0.
- You must attend the demonstration, otherwise your score will be 0. The demonstration schedule will be announced on NCKU Moodle.
- You must create GUI, otherwise your point will be deducted.
- Upload to => 140.116.154.28 -> Upload/Homework/Hw1
  - User ID: opencvdl2024 Password: RL2024opencvdl
- Format
  - Filename: Hw1_StudentID_Name_Version.rar
    - Ex: Hw1_F71234567_林小明_V1.rar
    - If you want to update your file, you should update your version to be V2,
    - Ex: Hw1_F71234567_林小明_V2.rar
  - Content: Project folder *(Excluding the pictures)

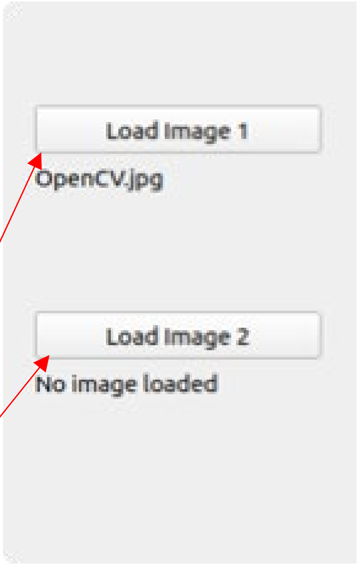
    *Note: Remove your "Debug" folder to reduce file size.

Notice (2/2)

- Python (recommended):
  - Python 3.8
  - Opencv-contrib-python (4.10.0)
  - Matplotlib (3.7.3)
  - Numpy (1.23.5)
  - UI framework: pyqt5 (5.15.11)

Assignment scoring (Total: 100%)

1. Image Processing (出題： Takeru)
   - 1.1 Color Separation
   - 1.2 Color Transformation
   - 1.3 Color Extraction
2. Image Smoothing (出題：Liu)
   - 2.1 Gaussian filter
   - 2.2 Bilateral filter
   - 2.3 Median filter
3. Edge Detection (出題： Ying)
   - 3.1 Sobel X
   - 3.2 Sobel Y
   - 3.3 Combination and Threshold
   - 3.4 Gradient Angle
4. Transforms (出題：Tina)
   - 4.1 Rotation
   - 4.2 Scaling
   - 4.3 Translate



* Don’t fix your image path
(There is another dataset for demonstration)

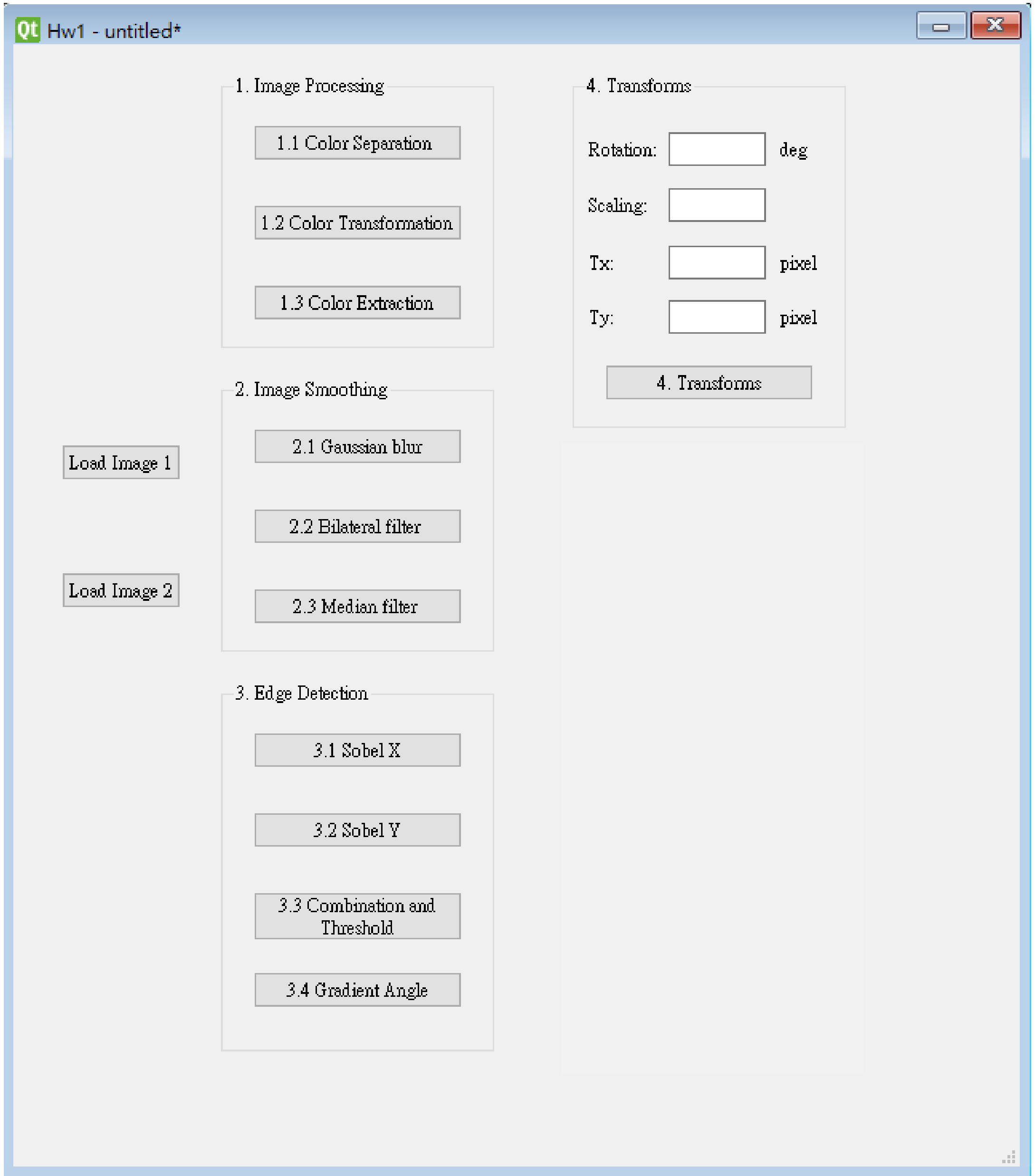
Load image 請用下面Function 來讀取路徑

[QFileDialog.getOpenFileName](#)

[獲取打開的檔路徑](#)

Assignment scoring (Total: 100%)

- Use one UI to present 5 questions.



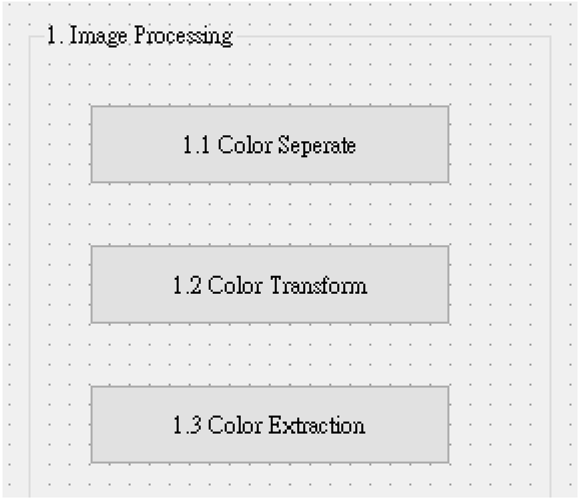
1. Image Processing

(出題：Takeru)

1.1 Color Separation

1.2 Color Transformation

1.3 Color Extraction



1.1 Color Separation

(出題：Takeru)

1. Given: a BGR image, “rgb.jpg”

   Q: Extract 3 channels of the image BGR and show the result color images.

   1) Use cv2.split() to get R G B gray scale images.

O/P O/P O/P I/P

b, g, r = cv2.split(image)

image: A BGR image “rgb.jpg” (640×452×3)
b: The Blue grayscale image (640×452×1)
g: The Green grayscale image (640×452×1)
r: The Red grayscale image(640×452×1)

   2) Use cv2.merge() to turn each gray scale image back to BGR image, individually

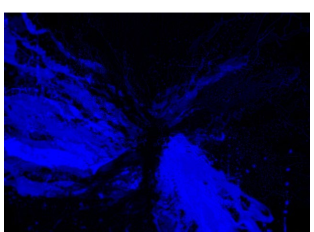
O/P **I/P**

b_image = cv2.merge([b, zeros, zeros])
g_image = cv2.merge([zeros, g, zeros])
r_image = cv2.merge([zeros, zeros, r])

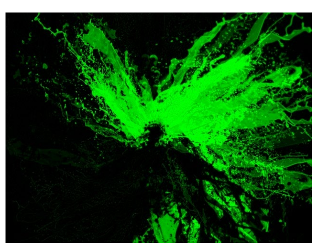
zeros : A blank (black) image (2D array filled with zeros) (640×452)
b: The blue grayscale image (640×452× 1)
g: The green grayscale image (640×452× 1)
r: The red grayscale image (640×452× 1)
b_image: A BGR image showing only the Blue channel, with Green and Red channels set to black. (640×452×3)
g_image: A BGR image showing only the Green channel, with Blue and Red channels set to black. (640×452×3)
r_image: A BGR image showing only the Red channel, with Green and Red channels set to black. (640×452×3)

Please show each R ,G ,B image as Figure

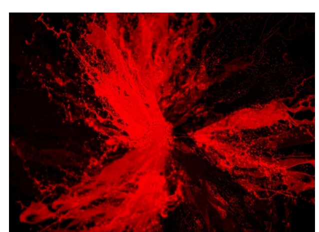
image (rgb.jpg)



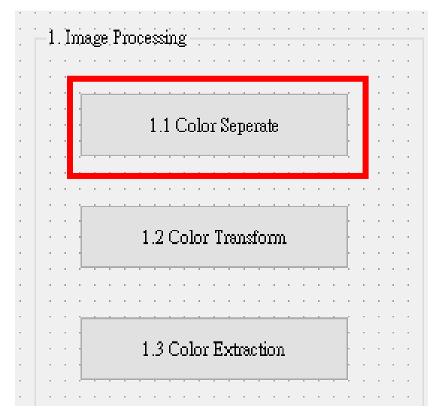
b_image



g_image



r_image



1.2 Color Transformation

(出題：Takeru)

2. Given: A BGR image, "rgb.jpg"

Q: Transform "rgb.jpg" into grayscale by

Q1): Call OpenCV function cv2.cvtColor(…, cv2.COLOR_BGR2GRAY) on rgb.jpg to generate Image *cv_gray*

O/P I/P

cv_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

image: A BGR image, rgb.jpg (640×452×3)
cv_gray: A grayscale image (640×452× 1) Fig. Q1

Q2): Merge BGR separated channel images from problem 1.1 to generate avg_gray = $(r+g+b)/3$.

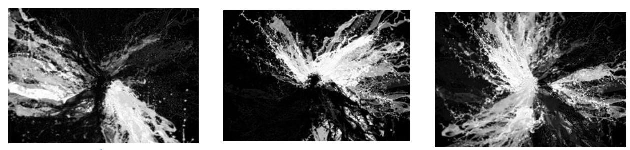
O/P I/P I/P I/P

avg_gray = (b/3 + g/3 + r/3).astype(np.uint8)

Convert to an 8-bit unsigned integer format (values between 0 and 255)

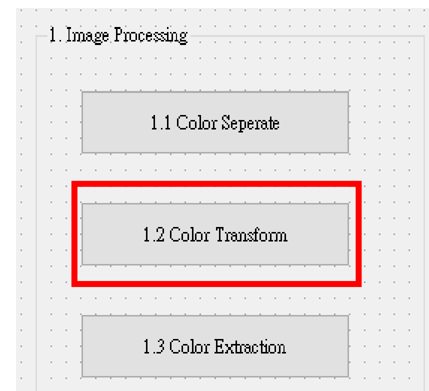
b: The blue grayscale image (640×452× 1)
g: The green grayscale image (640×452× 1)
r: The red grayscale image (640×452× 1)
avg_gray: A grayscale image (640×452× 1)Fig. Q2

image (rgb.jpg)



b g r

Result from problem 1.1 (1)



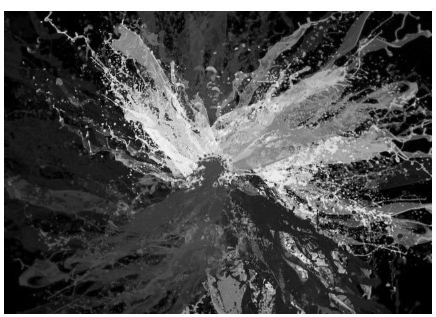


Fig. Q1) *cv_gray*

Perceptually weighted formula:
cv_gray= 0.299r + 0.587g + 0.114 b

Fig. Q2) *avg_gray*

Average weighted formula
avg_gray= (b+g+r)/3

1.3 Color Extraction

(出題：Takeru)

3. Given: A BGR image, “rgb.jpg”

Q: Show the Yellow-Green mask I_1 and the image with Yellow and Green colors removed I_2.

1) Transform “rgb.jpg” from BGR format to HSV format: cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

O/P hsv_image = cv2.cvtColor(I/P image, cv2.COLOR_BGR2HSV)

image: A BGR image, rgb.jpg (640×452×3)
hsv_image: A HSV image (640×452×3)

2) Yellow-Green **mask** I_1 by calling : **cv2.inRange(hsv_image , lower_bound , upper_bound)** Please refer to the next slide

O/P mask = cv2.inRange(I/P hsv_image, I/P lower_bound, I/P upper_bound)
I_1

hsv_Image: A HSV image (640×452×3)
lower_bound:The lower bound of the Yellow-Green HSV range
upper_bound: The upper bound of the Yellow-Green HSV range
mask: The Yellow-Green binary mask (640×452× 1) I_1

3) Invert the mask I_1 by calling: **cv2.bitwise_not(mask)**

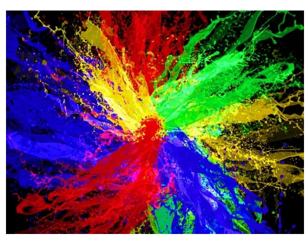
O/P mask_inverse = cv2.bitwise_not(I/P mask)

mask: The Yellow-Green binary mask (640×452× 1)
mask_inverse: The inversed mask(640×452× 1)

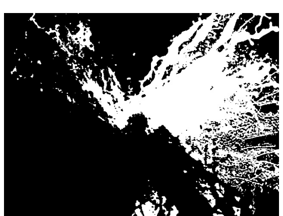
4) Remove Yellow and Green color in the image to generate I_2 by calling : **cv2.bitwise_and(image, image ,mask_inverse)**

O/P extracted_image = cv2.bitwise_and(I/P image, I/P image, I/P mask=mask_inverse)
I_2

image: A BGR image, rgb.jpg (640×452×3)
mask_inverse: The inversed mask(640×452× 1)
extracted_image: A BGR image where Yellow and Green colors are removed (640×452×3) I_2



rgb.jpg



Yellow-Green mask I_1

The inversed mask

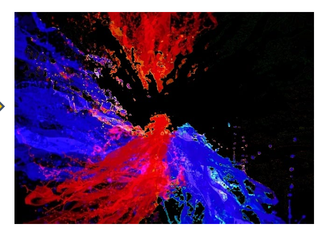
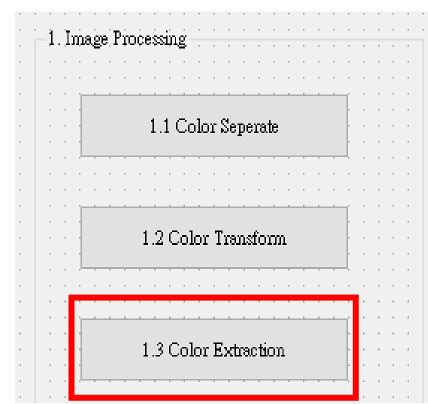


Image without yellow and green I_2



1.3 Color Extraction

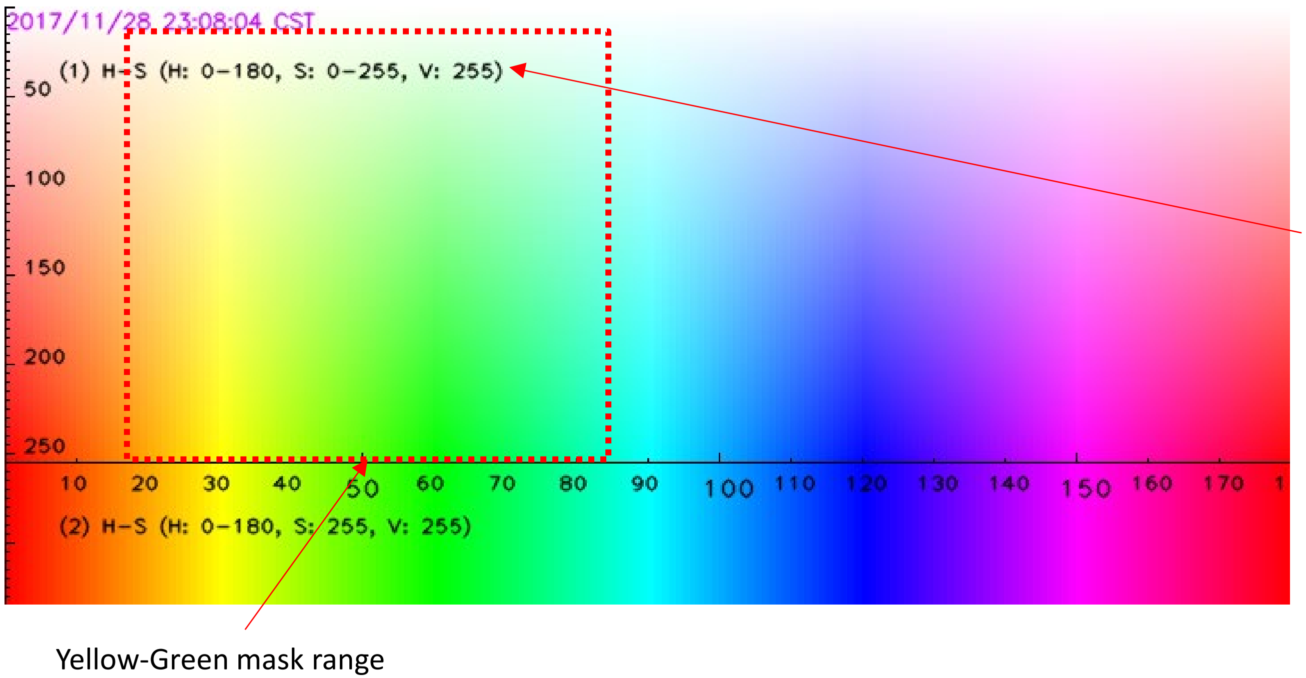
(出題：Takeru)

Yellow-Green HSV range

Hue (H): 18-85

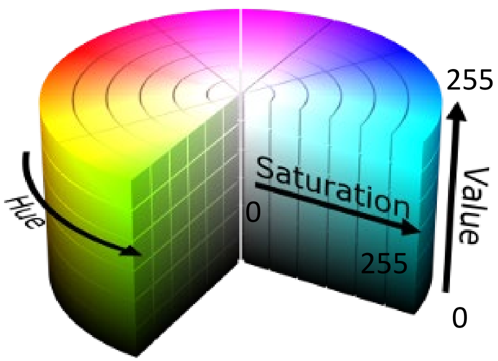
Saturation (S): 0-255

Value (V): 25-255



HSV values ranges between (h:0–180, s:0–255, v:0–255)

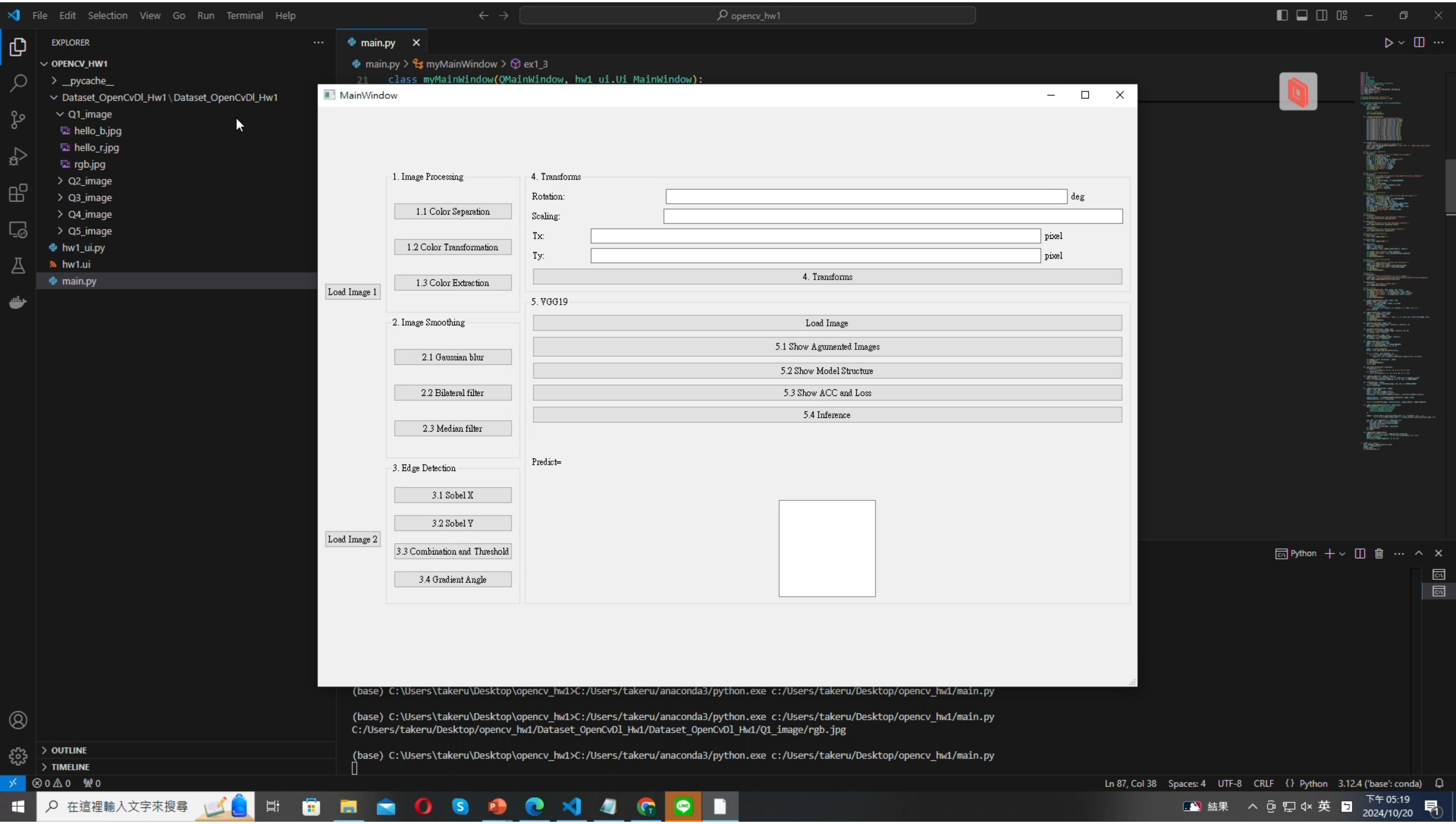
H(Hue) : x axis
S(Saturation) : y axis
V(Value) : 255



REF: https://medium.com/@gowtham180502/how-to-detect-colors-using-opencv-python-98aa0241e713

1. Image Processing– Demo Video

(出題：Takeru)



2. Image Smoothing

(出題：Liu)

2.1 Gaussian blur

2.2 Bilateral filter

2.3 Median filter

- Hint
  1) Textbook Chapter 3, p. 50 ~ 52, p.109~115

2.1 Gaussian Filter

(出題：Liu)

image: original image “image1.jpg”
kernal: kernal size of gaussian filter, which is (2m + 1) x (2m + 1)
sigmaX: standard deviation in the X direction
sigmaY: standard deviation in the Y direction
blur: image with gaussian blur

1. Given: 1 color image, “image1.jpg”
2. Q:
   1) Apply “Load image 1” button to load image.
   2) Click “2.1 Gaussian Blur” to show the popup window.

      O/P I/P I/P I/P I/P I/P
      blur = cv2.GaussianBlur(image, (kernal, kernal), sigmaX, sigmaY)

   3) Apply cv2.createTrackbar() to create a trackbar on popup window.
      - Or you can use cv2.imshow() to show these images (m=1, 2, 3, 4, 5).
   4) Apply trackbar to change the kernal radius (m).
   5) The range of radius size is m=[1, 5].
   6) Apply gaussian filter (cv2.GaussianBlur()) which kernel size is $(2m + 1) \times (2m + 1)$ to “image1.jpg” and show result on popup window.

O/P

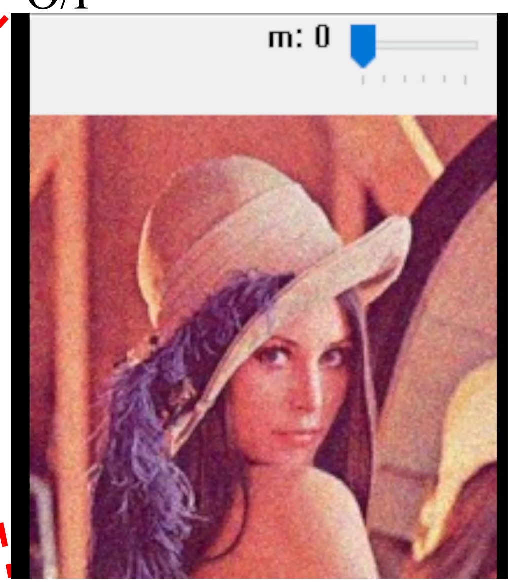
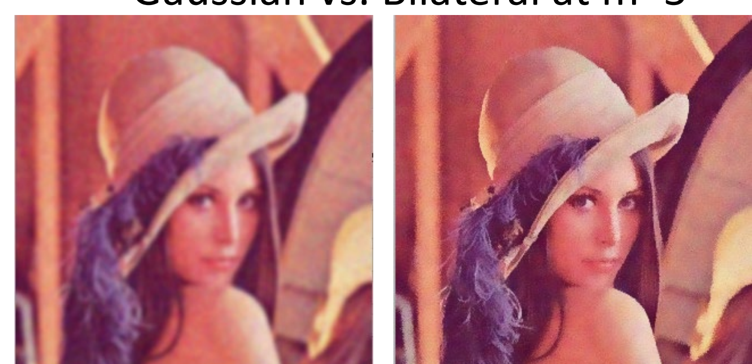


image1.jpg

Gaussian vs. Bilateral at m=5



Gaussian Bilateral

2.2 Bilateral Filter

(出題：Liu)

1. Given: 1 color image, “image1.jpg”
2. Q:
   1) Apply “Load image 1” button to load image.
   2) Click “2.2 Bilateral Filter” to show the popup window.

      O/P I/P I/P I/P I/P
      bilateral = cv2. bilateralFilter(image, d, sigmaColor, sigmaSpace)

   3) Apply cv2.createTrackbar() to create a trackbar on popup window.
      - Or you can use cv2.imshow() to show these images (m=1, 2, 3, 4, 5).
   4) Apply trackbar to change the kernal radius (m).
   5) The range of radius size is m=[1, 5].
   6) Apply bilateral filter (cv2. bilateralFilter()) which kernel size is $(2m+1) \times (2m+1)$ to “image1.jpg” and show result on popup window.

image: original image “image1.jpg”
d: Diameter of Pixel Neighborhood. ex: d=5 means 5x5 pixel area ➔ m=2 for (2m + 1) x (2m + 1)
sigmaColor: Filter Sigma in Color Space
像素之間的顏色差異對濾波結果的影響程度
sigmaSpace: Filter Sigma in Coordinate Space
像素之間的距離對濾波結果的影響程度，當d>0時此參數無作用，但因輸入需要此參數，所以將其設置與 sigmaSpace 相等就好。
bilateral: image after applying bilateral filter

- Hint
  1) simgaColor = 90, sigmaSpace = 90

O/P

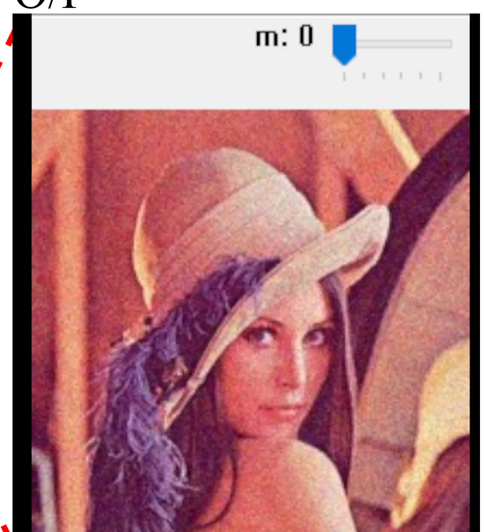


image1.jpg

sigmaColor 範例: 假設有個3x3的bilateral filter，sigmaColor設為100

針對第1個pixel，中心與其差距為240-150=90 < 100 (SigmaColor)，則第1個pixel會參與濾波

150	10	60
80	240	110
40	200	90

針對第7個pixel，中心與其差距為240-40=200 > 100 (SigmaColor)，則第7個pixel不會參與濾波

2.3 Median Filter

(出題：Liu)

1. Given: 1 color image, "image2.jpg"
2. Q:
   1) Apply "Load image 2" button to load image.
   2) Click "2.3 Median Filter" to show the popup window.

      median (O/P) = cv2. medianBlur(image (I/P), kernal (I/P))

   3) Apply cv2.createTrackbar() to create a trackbar on popup window.
      - Or you can use cv2.imshow() to show these images (m=1, 2, 3, 4, 5).
   4) Apply trackbar to change the kernal radius (m).
   5) The range of radius size is m=[1, 5].
   6) Apply median filter (cv2. medianBlur()) which kernel size is $(2m+1) \times (2m+1)$ to "image2.jpg" and show result on popup window.

image: original image "image1.jpg"
kernal: kernal size of median filter, which is (2m + 1) x (2m + 1)
median: image after applying median filter

O/P

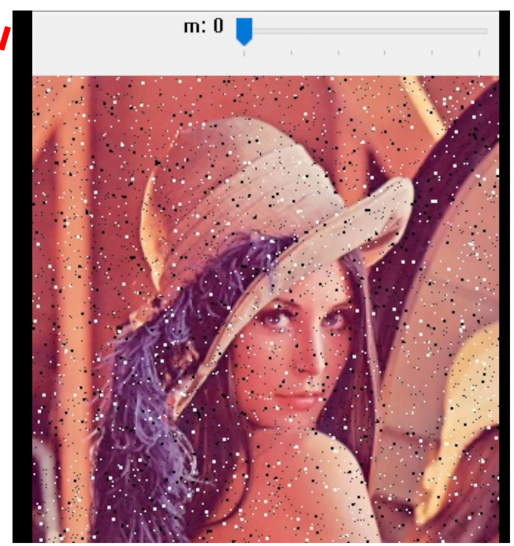


image2.jpg

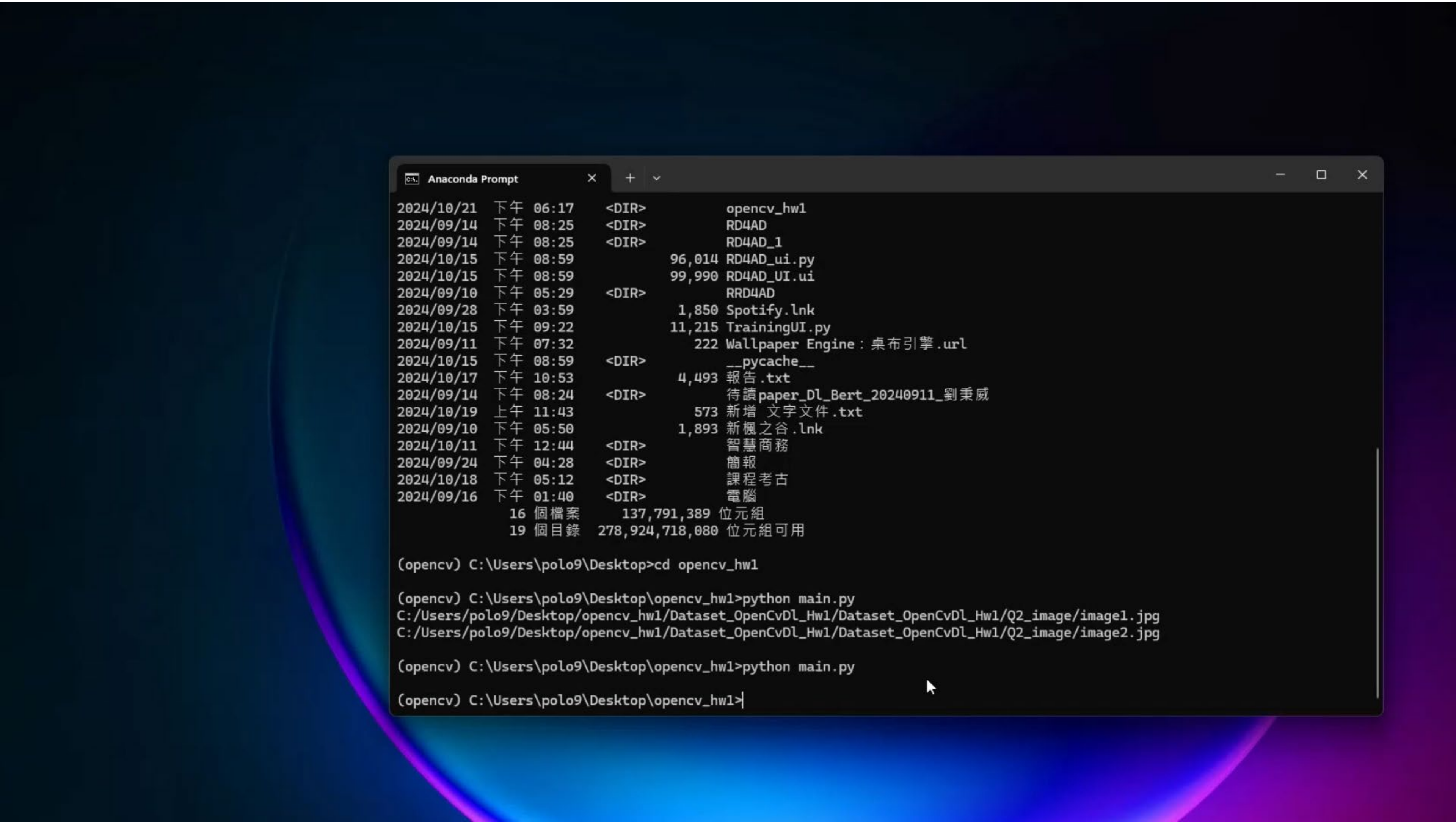
Median filter example

223	186	114
204	161	106
219	194	138

106 114 138 161 $\boxed{186}$ 194 204 219 223

2. Image Smoothing– Demo Video

(出題：Liu)



3. Edge Detection

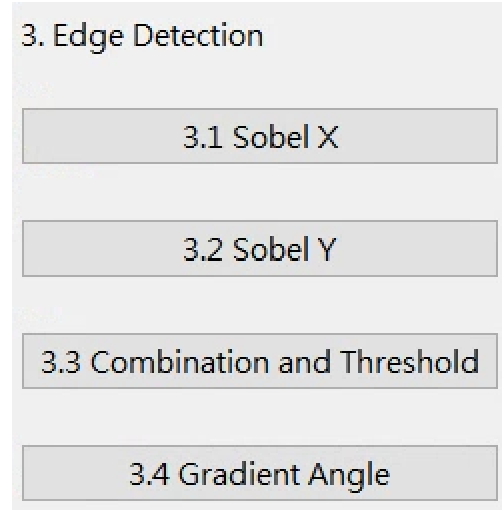
(出題：Ying)

3.1 Sobel X

3.2 Sobel Y

3.3 Combination and Threshold

3.4 Gradient Angle



3.1 Sobel x

(出題： Ying)

1. Given: A RGB image, “building.jpg”
2. Q: Generate Sobel x image for “building.jpg”
   1) Convert the RGB image into a grayscale image

      gray (O/P) = cv2.cvtColor(image (I/P), cv2.COLOR_BGR2GRAY)

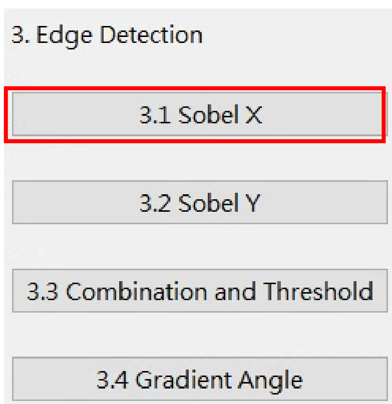
   1) Smooth grayscale image with Gaussian smoothing filter.

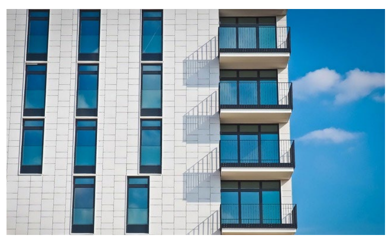
      blur (O/P) = cv2.GaussianBlur(gray (I/P), (kernal (I/P), kernal (I/P)), sigmaX (I/P), sigmaY (I/P))

   2) Apply Sobel edge detection to detect vertical edge by your own 3x3 Sobel x operator. (Can not use OpenCV Function cv2.Sobel and cv2.filter2D.)
   3) Please show the result with cv2.imshow function.

- Hint: Textbook Chapter 6, p.144 ~ 149

image: original image “building.jpg”
gray: grayscale image
kernal: kernal size of gaussian filter, which is (2m + 1) x (2m + 1)
sigmaX: standard deviation in the X direction
sigmaY: standard deviation in the Y direction
blur: image with gaussian blur



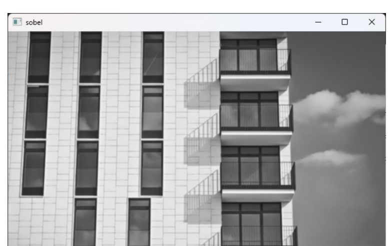


I/P: building.jpg

Convert into grayscale image

Grayscale Image

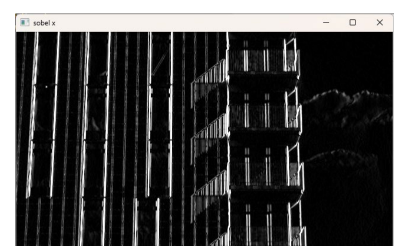
Gaussian smoothing



Blur image

-1	0	1
-2	0	2
-1	0	1

Apply Sobel x filter on blur image



O/P: Sobel x

3.2 Sobel y

(出題： Ying)

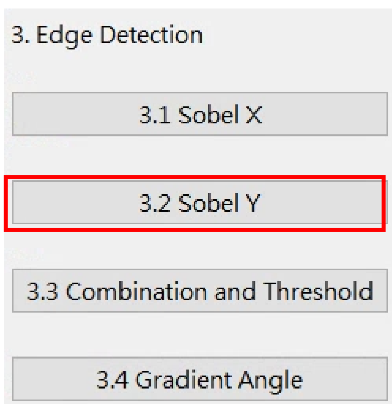
1. Given: A RGB image, “building.jpg”
2. Q: Generate Sobel x image for “building.jpg”
   1) Convert the RGB image into a grayscale image

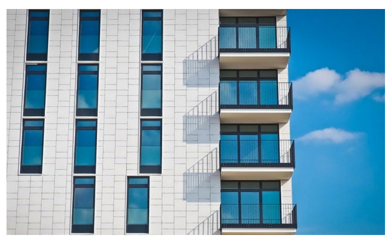
      O/P I/P
      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   2) Smooth grayscale image with Gaussian smoothing filter.

      O/P I/P I/P I/P I/P I/P
      blur = cv2.GaussianBlur(gray, (kernal, kernal), sigmaX, sigmaY)
   3) Apply Sobel edge detection to detect horizontal edge by your own 3x3 Sobel x operator. (Can not use OpenCV Function cv2.Sobel and cv2.filter2D.)
   4) Please show the result with cv2.imshow function.

- Hint: Textbook Chapter 6, p.144 ~ 149

image: original image “building.jpg”
gray: grayscale image
kernal: kernal size of gaussian filter, which is (2m + 1) x (2m + 1)
sigmaX: standard deviation in the X direction
sigmaY: standard deviation in the Y direction
blur: image with gaussian blur



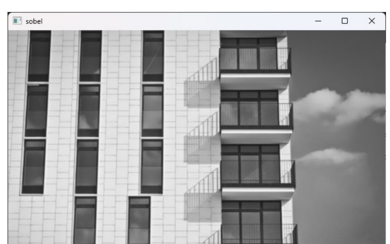


I/P: building.jpg

Convert into grayscale image

Grayscale Image

Gaussian smoothing



Blur image

-1	-2	-1
0	0	0
1	2	1

Apply Sobel y filter on blur image

O/P: Sobel y

3.3 Combination and Threshold

(出題： Ying)

1. Given: The result of 3.1) Sobel x and 3.2) Sobel y
2. Q: Combine Sobel x and Sobel y, then set threshold for result
   1) New value of pixel = $\sqrt{Sobel_X{}^2 + Sobel_Y{}^2}$
   2) Normalize combination result to 0~255

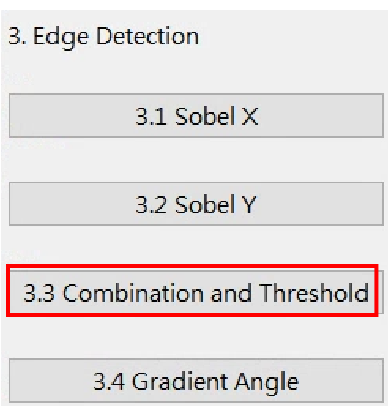
      normalized (O/P) = cv2.normalize(combination (I/P), None, min (I/P, 0), max (I/P, 255), cv2.NORM_MINMAX)
   3) Given threshold (1)128, (2) 28. Set to 0 if pixel value is lower than threshold, otherwise, set to 255.

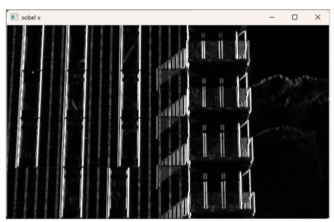
      _, result (O/P) = cv2.threshold(normalized (I/P), thresh (I/P, 128), maxval (I/P, 255), cv2.THRESH_BINARY)
   4) Show both **combination** and **threshold result** with cv2.imshow function. Two results should be shown together.

- Hint: Textbook Chapter 6, p.148 ~ 149

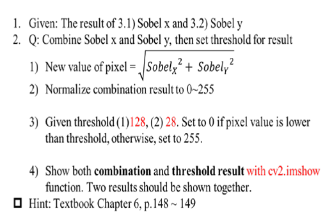
combination: image combined with Sobelx & Sobely
min: the minimum value for normalization
max: the maximum value for normalization
normalized: the resulting image after normalization

thresh: the threshold value
maxval: the maximum value in the grayscale range
result: the resulting image after applying the threshold to "normalized"





I/P: Sobel x



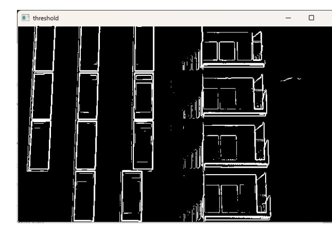
1. Given: The result of 3.1) Sobel x and 3.2) Sobel y
2. Q: Combine Sobel x and Sobel y, then set threshold for result
   1) New value of pixel = $\sqrt{Sobel_X{}^2 + Sobel_Y{}^2}$
   2) Normalize combination result to 0~255
   3) Given threshold (1)128, (2) 28. Set to 0 if pixel value is lower than threshold, otherwise, set to 255.
   4) Show both **combination** and **threshold result** with cv2.imshow function. Two results should be shown together.

- Hint: Textbook Chapter 6, p.148 ~ 149

I/P: Sobel y

$\sqrt{Sobel_X{}^2 + Sobel_Y{}^2}$

Combination of Sobel x and Sobel y



O/P (1): Threshold=128

O/P (2): Threshold=28

3.4 Gradient Angle

(出題：Ying)

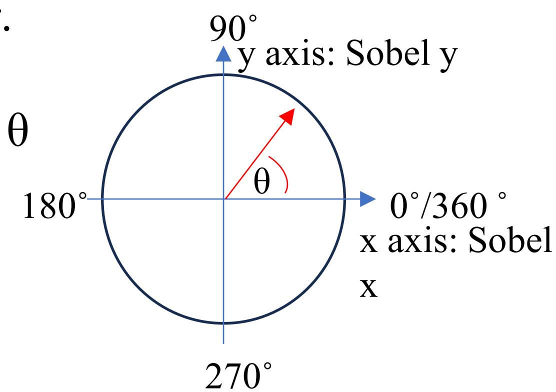
1. Given: The result of 3.1) Sobel x and 3.2) Sobel y
2. Q: Calculate the gradient angle θ and show specific range of angle.
   1) Calculate the gradient angle θ by result of Sobel x and Sobel y
   2) Generate **two different masks** by given two different range of angle (1)170°~ 190°(2) 260°~ 280°. Set to 255 if pixel value is in range, otherwise set to 0.
   3) Generate results by calling cv2.bitwise_and

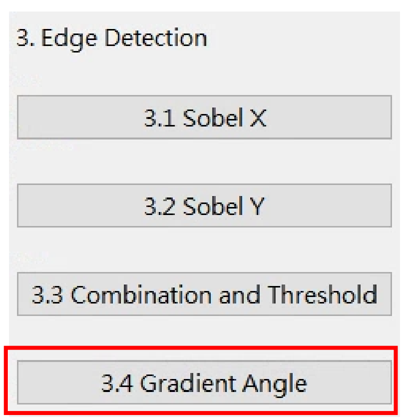
      output = cv2.bitwise_and(img1, img2)

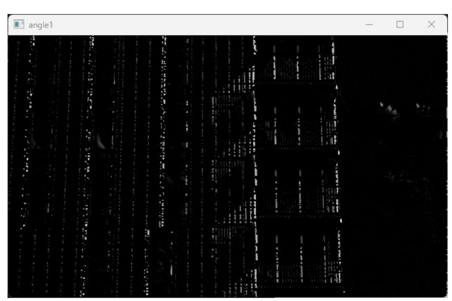
      (output: O/P; img1: I/P; img2: I/P)
   4) Show both results with cv2.imshow function. Two results should be shown together.

img1: The first input image (combination image from 3.3)
img2: The second input image (your masks), which is used for the bitwise AND operation with img1
output: result image

□ Hint: Gradient angle θ





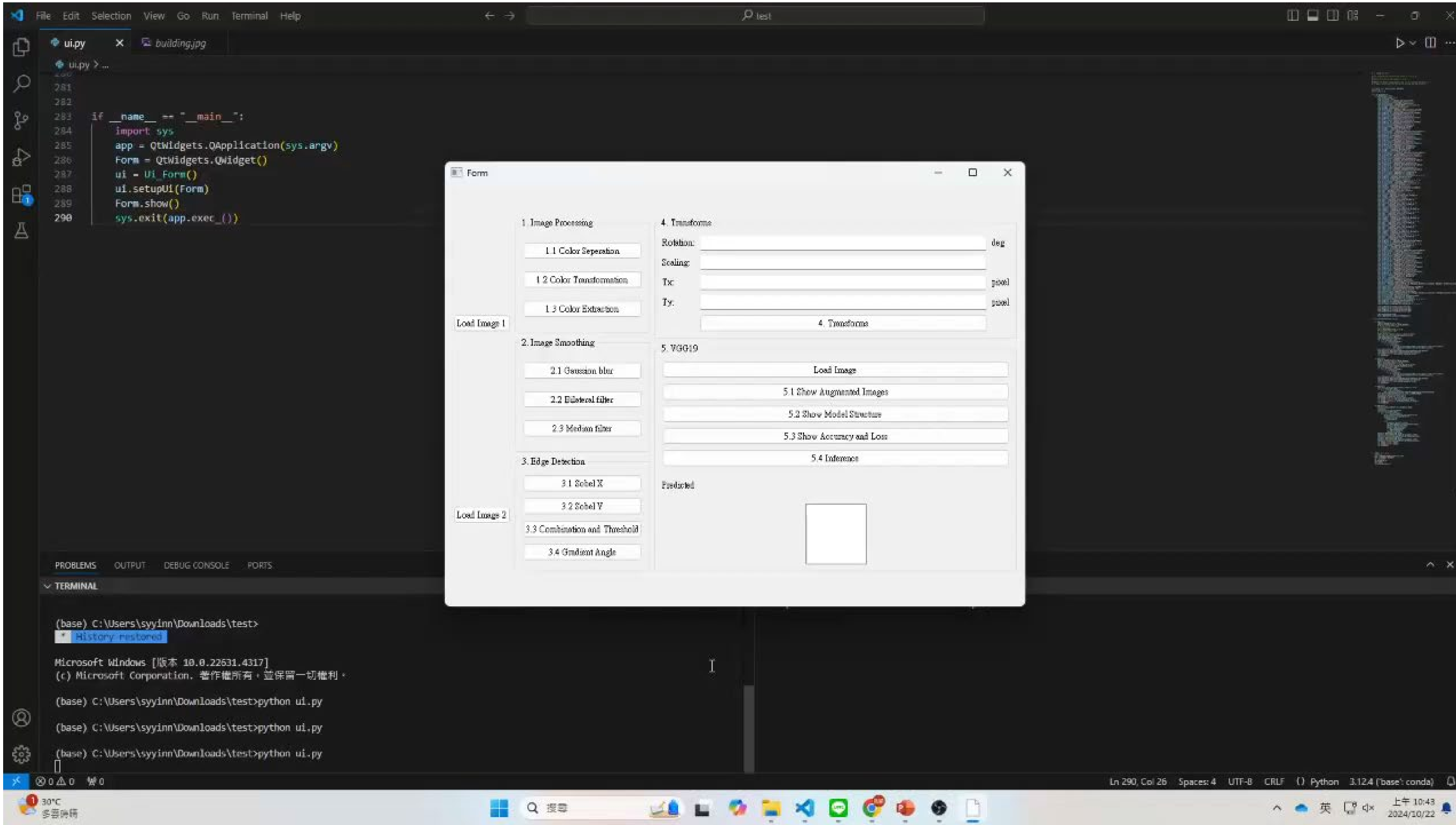


O/P (1)

O/P (2)

3. Edge Detection - DEMO

(出題：Ying)



4. Transforms

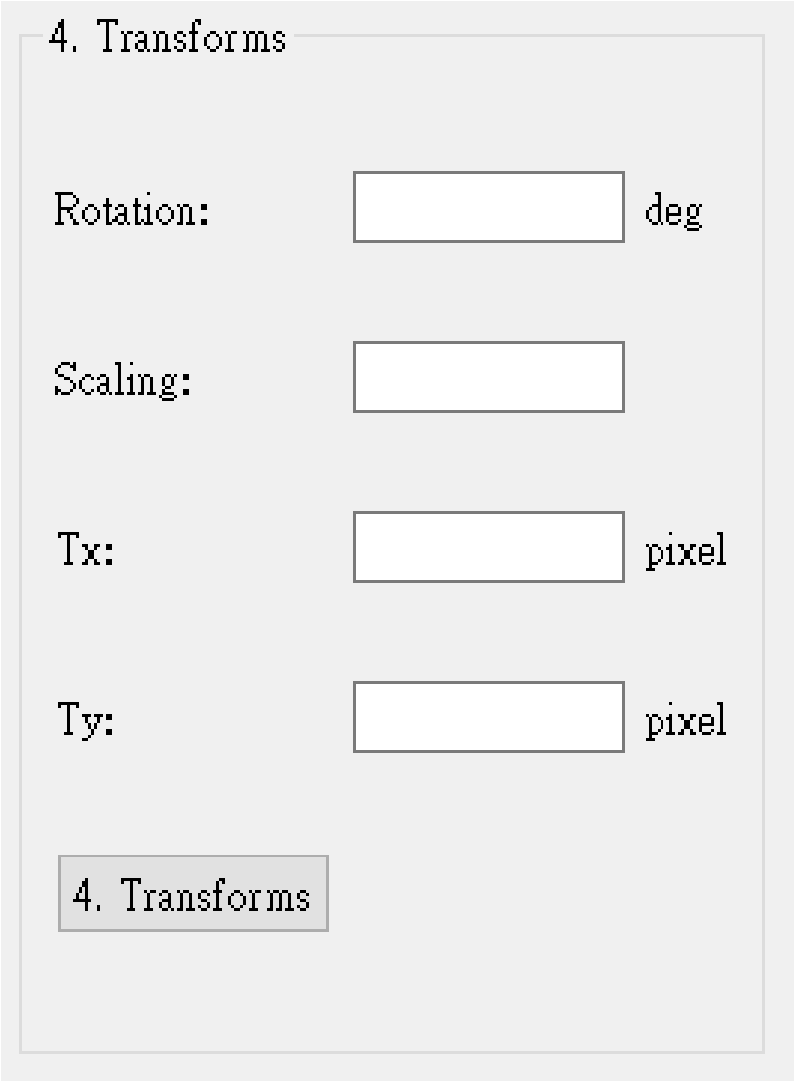
(出題：Tina)

4.1 Rotation

4.2 Scaling

4.3 Translate

UI Demo:



4. Transforms

(出題：Tina)

1. Given: "burger.png"
2. Q: 1) Click button "4. Transforms", burger.png should be showed.

   2) Please rotate, scale and translate the burger (as image below) using

   $$\overset{O/P}{result} = cv2.warpAffine(\overset{I/P}{img}, \overset{I/P}{M}, \overset{I/P}{(w,h)})$$

   function with following parameters

   (set default values 0, should be manually adjusted in the GUI)

   (1) Angle = 30° (positive degree ➔ counter-clockwise)

   (2) Scale = 0.9,

   (3) Translation with:

   - $X_{new} = X_{old} + 535$ pixels $= 240 + 535 = 775$
   - $Y_{new} = Y_{old} + 335$ pixels $= 200 + 335 = 535$

   - Point C (240, 200) is center point of burger in original image
   - Point C'(775, 535) is center point of burger in result image

img: image burger.png.

M: affine transformation matrix(3x3).

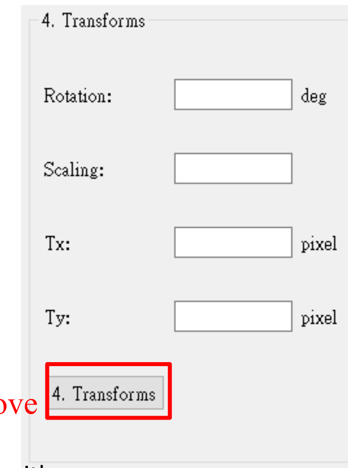
(w, h): width and height from image(1920x1080).

result: image after affine transformation

$$M_{(Scale=0.9)} = \begin{bmatrix} 0.9 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{(Angle = 30^o)} = \begin{bmatrix} \cos(30°) & -\sin(30°) & 0 \\ \sin(30°) & \cos(30°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{(Translation=535,335)} = \begin{bmatrix} 1 & 0 & 535 \\ 0 & 1 & 335 \\ 0 & 0 & 1 \end{bmatrix}$$



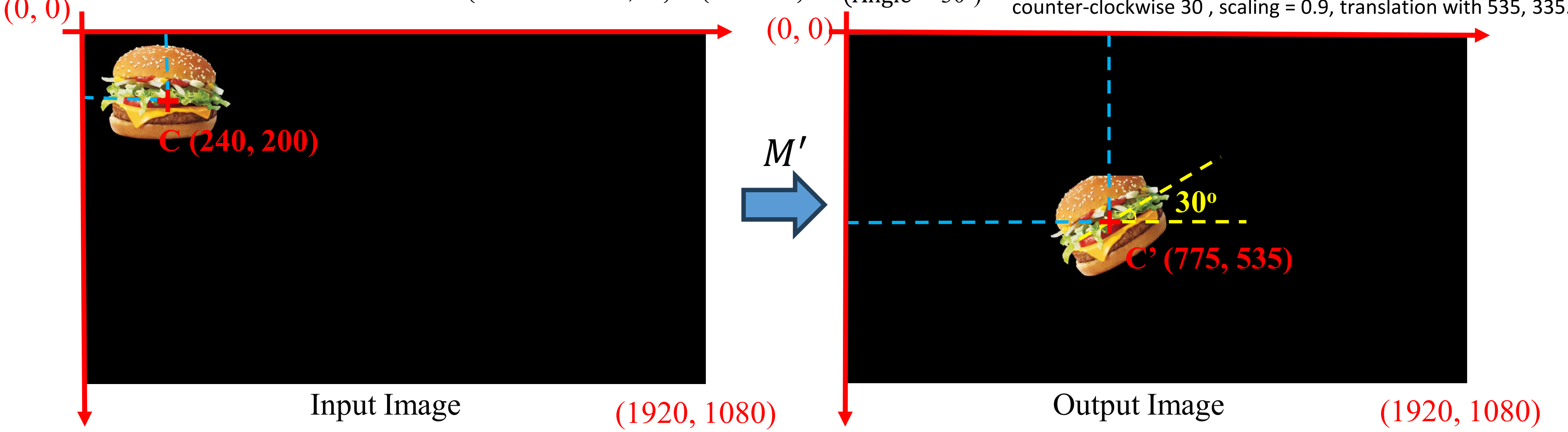
➢ Hint: Textbook Chapter 12, (p.407 ~ 412)

- Rotation, Scale: Object center not move
- Translation: Object center move

Result Demo:

$$M' = M_{(Translation=535,335)} \times M_{(Scale=0.9)} \times M_{(Angle=30^o)}$$

M': affine transformation with counter-clockwise 30 , scaling = 0.9, translation with 535, 335.



4. Transforms - DEMO

(出題：Tina)