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 2

TA:

彥博: nckubot65904@gmail.com

Office Hour: 17:00~19:00, Mon.

10:00~12:00, Fri.

At CSIE 9F Robotics Lab.

Notice (1/2)

- Copying homework is strictly prohibited!! Penalty: Grade will be zero for both persons!!
- If the code can't run, you can come to our Lab within one week and show that your programming can work. Otherwise you will get zero!!
- Due date => Midnight 23:59:59 on 2021/12/26 (Sun.)
  - No delay. If you submit homework after deadline, you will get 0.
- Upload to => 140.116.154.1 -> Upload/Homework/Hw2
  - User ID: opencvdl2021 Password: opencvdl2021
- Format
  - Filename: Hw2_StudentID_Name_Version.rar
    - Ex: Hw2_F71234567_林小明_v1.rar
    - If you want to update your file, you should update your version to be v2, ex: Hw2_F71234567_林小明_v2.rar
  - Content: project folder*(including the pictures)
    *note: remove your "Debug" folder to reduce file size

Notice (2/2)

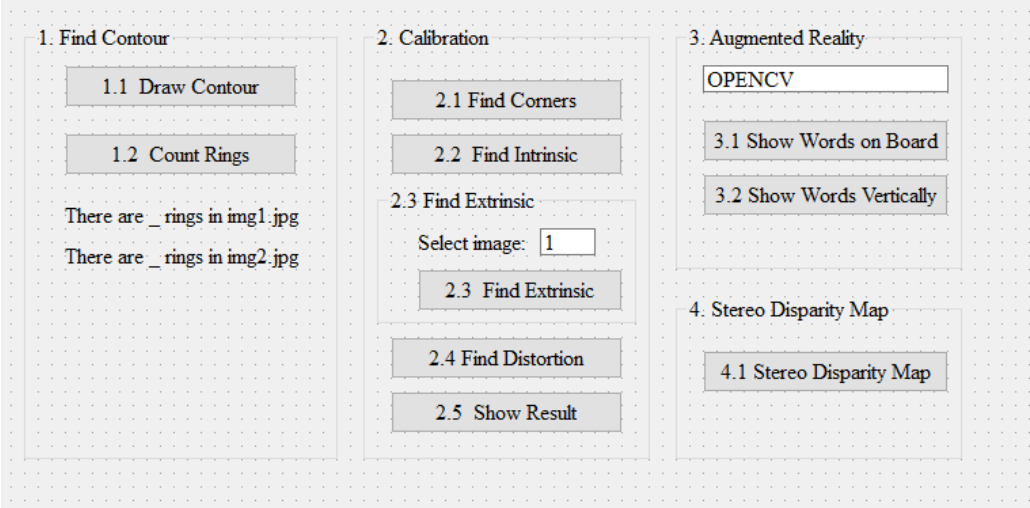
- Python (recommended)
  - Python 3.7 (https://www.python.org/downloads/)
  - opencv-contrib-python (3.4.2.17)
  - Matplotlib 3.1.1
  - UI framework: pyqt5 (5.15.1)

- C++ (check MFC guide in ftp)
  - OpenCV 3.3.1 (https://opencv.org/release.html)
  - Visual Studio 2015 (download from http://www.cc.ncku.edu.tw/download/)
  - UI framework: MFC

Assignment scoring (Total: 100%)

1. (20%) Find Contour (出題：Ray)
   - 1.1 (10%) Draw Contour
   - 1.2 (10%) Count Coins
2. (20%) Camera Calibration (出題：West)
   - 2.1 (4%) Corner detection
   - 2.2 (4%) Find the intrinsic matrix
   - 2.3 (4%) Find the extrinsic matrix
   - 2.4 (4%) Find the distortion matrix
   - 2.5 (4%) Show the undistorted result

UI Example



3. (20%) Augmented Reality (出題：Keter)
   - 3.1 (10%) Show words on board
   - 3.2 (10%) Show words vertically
4. (20%) Stereo Disparity Map (出題：Aaron)
   - 4.1 (10%) Compute disparity map
   - 4.2 (10%) Checking the Disparity Value
5. (20%) Dogs and Cats classification Using ResNet50 (出題：育成)

1. (20%) Find Contour

(出題：Ray)

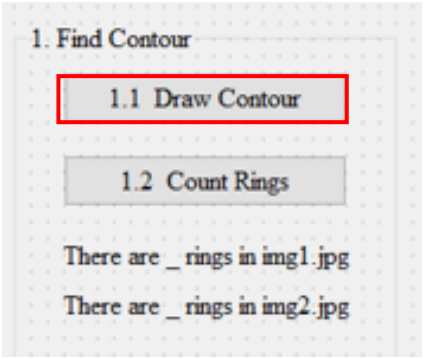
1.1 (15%) Draw Contour

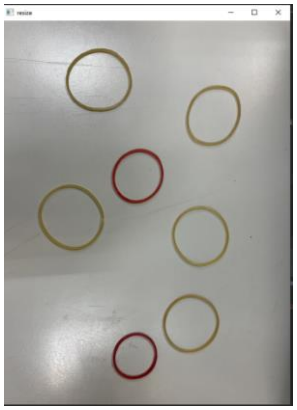
1.2 (5%) Count Rings

1.1 Find Contour – Draw Contour

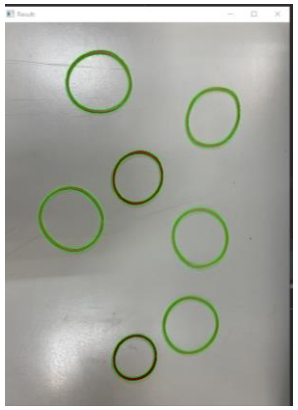
(出題：Ray)

- Given: two color images, “img1.jpg” and “img2.jpg”
- Q: 1) **Draw Contour**: Using OpenCV functions to find the contours of rings in two images.
- Hint: Textbook Chapter 8, p.234 ~ p.241
  1. RGB ➔ Resize(1/2) ➔ Grayscale ➔ Binary
  2. Remember to remove the noise. (use **Gaussian Blur** & **other function**)
  3. Using some **edge detection functions** to get better results. (Ex: cv2.Canny)



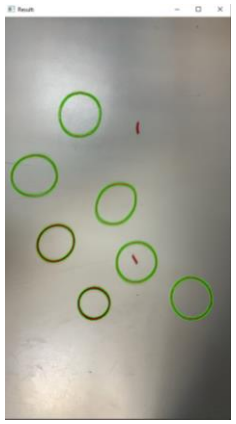


img1.jpg



Draw Contours

img2.jpg

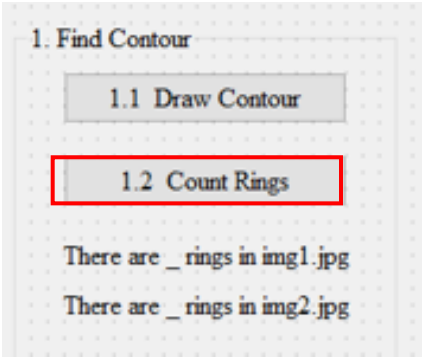


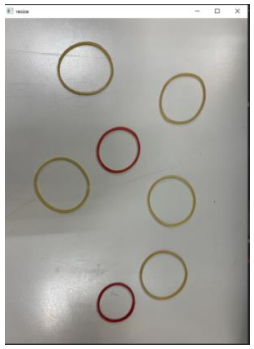
Draw Contours

1.2 Find Contour – Count Rings

(出題：Ray)

- Given: two color images, “img1.jpg” and “img2.jpg”
- Q: 2) **Count Rings**: Using OpenCV functions to find how many rings in two images.
- Hint: Textbook Chapter 8, p.234 ~ p.241
  Calculate how many rings (contour/2)





7 rings

img1.jpg

7 rings

img2.jpg

(出題：West)

2. (20%) Camera Calibration

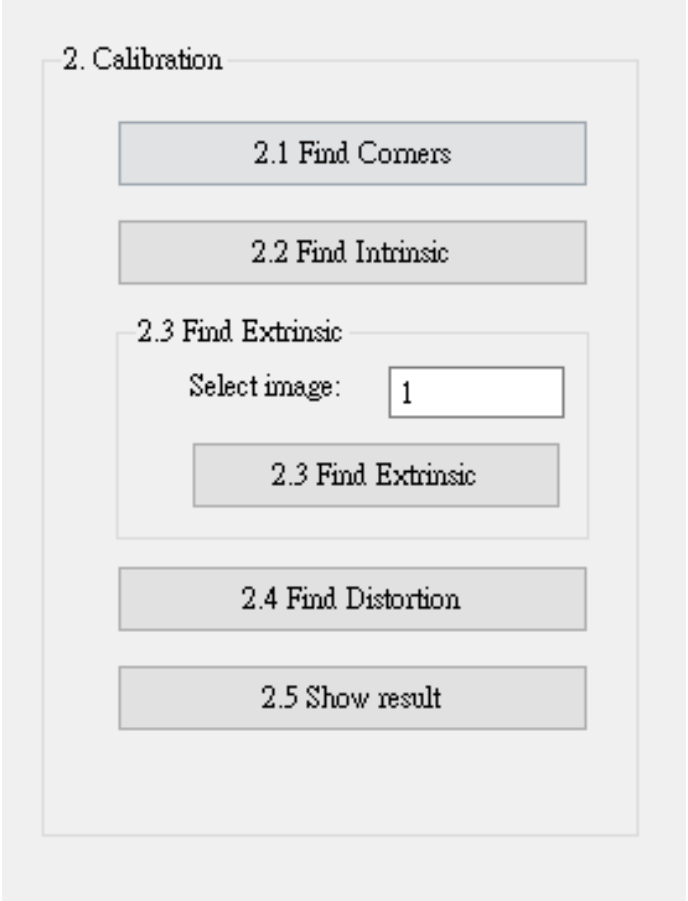
2.1 (4%) Corner detection

2.2 (4%) Find the intrinsic matrix

2.3 (4%) Find the extrinsic matrix

2.4 (4%) Find the distortion matrix

2.5 (4%) Show the undistorted result



2.1 Corner Detection (4%)

(出題：West)

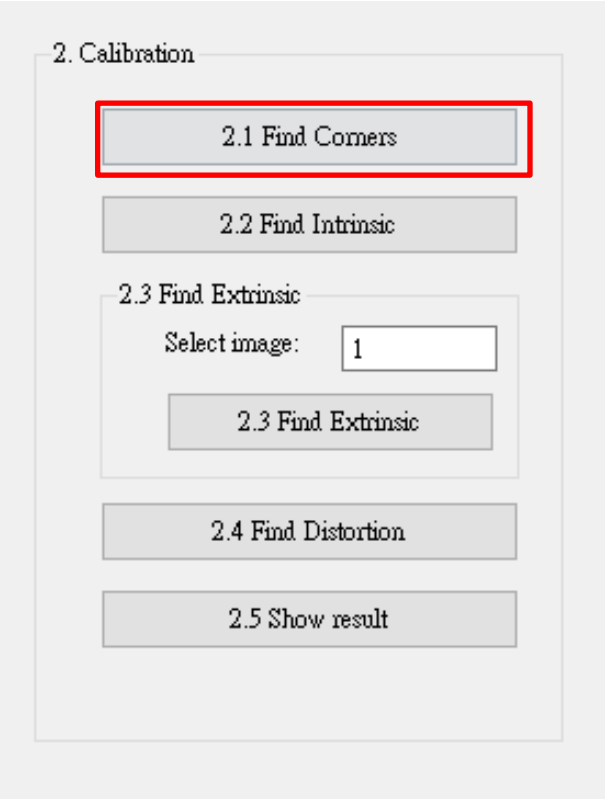
- Given: 15 images, 1.bmp ~ 15.bmp
- Q: 1) Find and draw the corners on the chessboard for each image.
  2) Click button “2.1” to show each picture 0.5 seconds.

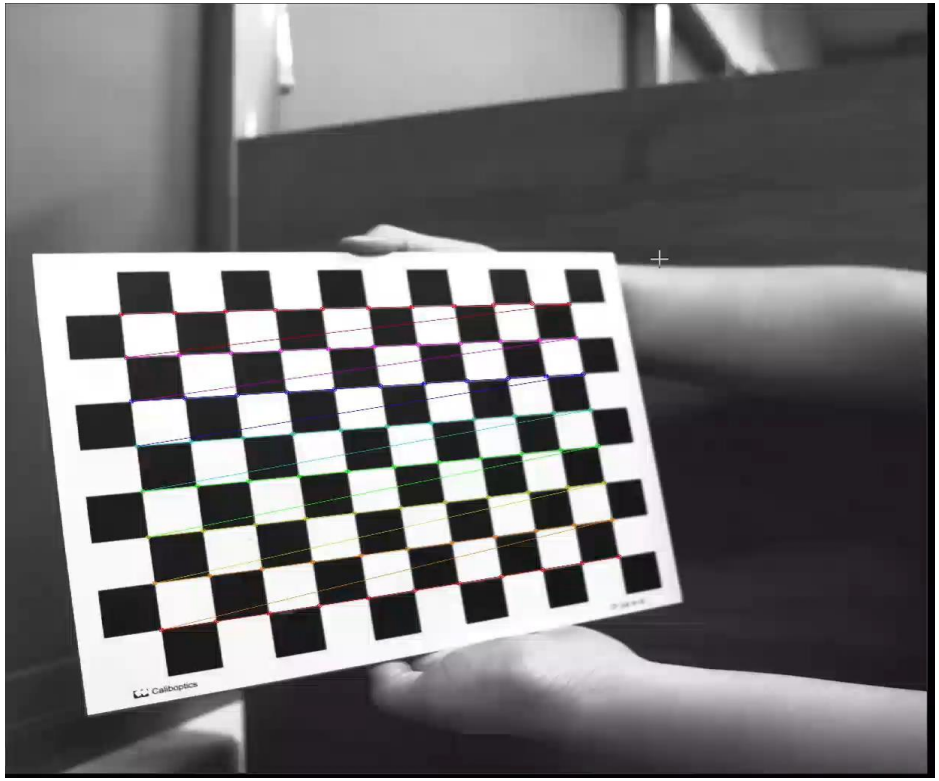
- Hint :

  OpenCV Textbook Chapter 11 (p. 398 ~ p. 399)

  cv.findChessboardCorners(…)

- Ex:





2.2 Find the Intrinsic Matrix (4%)

(出題：West)

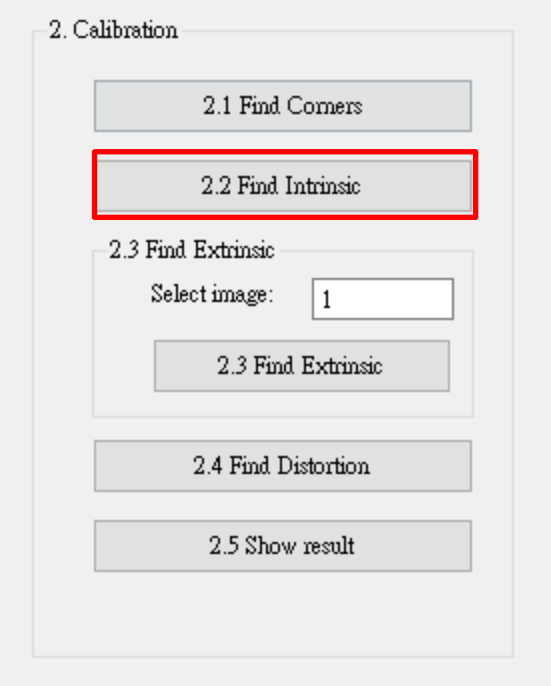
- Given: 15 images, 1.bmp ~ 15.bmp
- Q: 1) Find the intrinsic matrix (): $\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$

  2) Click button “2.2” and then show the result on the console window.
- Output format:

```
Intrinsic:
[[2.22370244e+03 0.00000000e+00 1.03021663e+03]
 [0.00000000e+00 2.22296836e+03 1.03752624e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

(Just an example)

❑ Hint: OpenCV Textbook Chapter 11 (P.398 ~ p.400)



2.3 Find the Extrinsic Matrix (4%)

(出題：West)

- Given: Intrinsic parameters, distortion coefficients, and the list of 15 images
- Q: 1) Find the extrinsic matrix of the chessboard for each of the 15 images, respectively:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}$$

2) Click button “2.3” and then show the result on the console window.

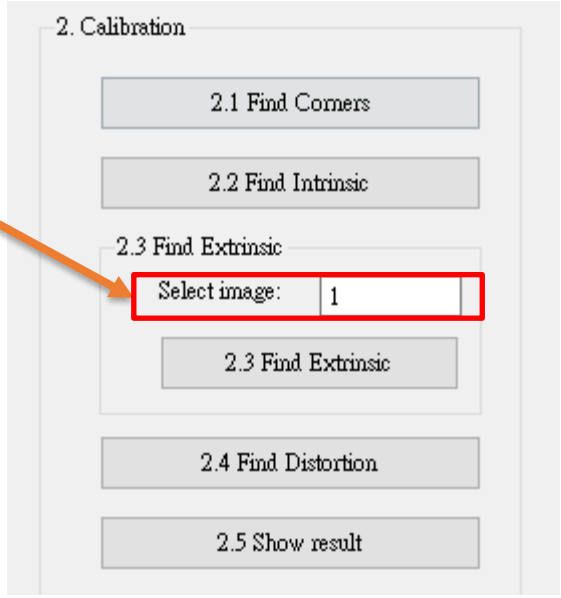
- Output format:

```
Extrinsic:
[[-0.8767247  -0.23001438  0.4224301   4.39838495]
 [ 0.19727469 -0.97293475 -0.12033563  0.68022105]
 [ 0.43867585 -0.02216645  0.89837194 16.22126   ]]
```

(Just an example)

❑ Hint: OpenCV Textbook Chapter 11, p.370~402

(1) List of numbers: 1~15
(2) Select 1, then 1.bmp will be applied, and so on



2.4 Find the Distortion Matrix (4%)

(出題：West)

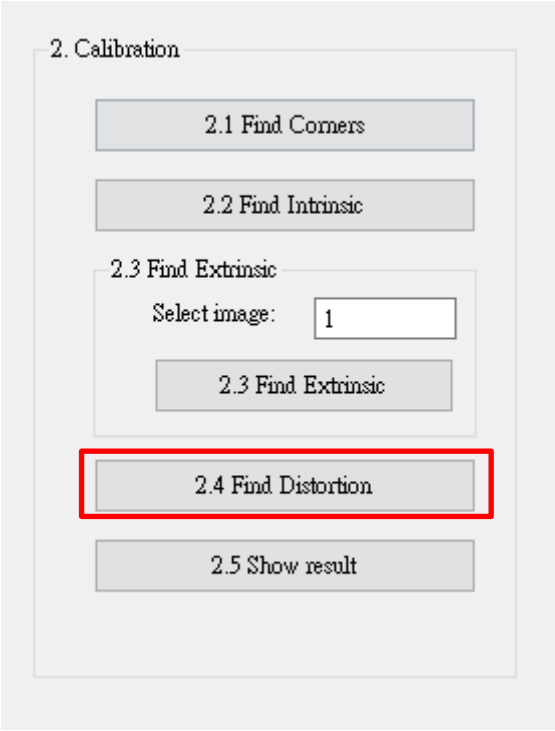
- Given: 15 images
- Q: 1) Find the distortion matrix: $[k_1, k_2, p_1, p_2, k_3]$

  2) Click button “2.4” to show the result on the console window.
- Output format:

```
Distortion:
[[-0.11868112  0.02776881 -0.00092036  0.00047227  0.11793646]]
```

(Just an example)

- Hint:
  - Distortion coefficients can be obtained simultaneously with intrinsic parameters
  - OpenCV Textbook Chapter 11 (P.398 ~ p.400)

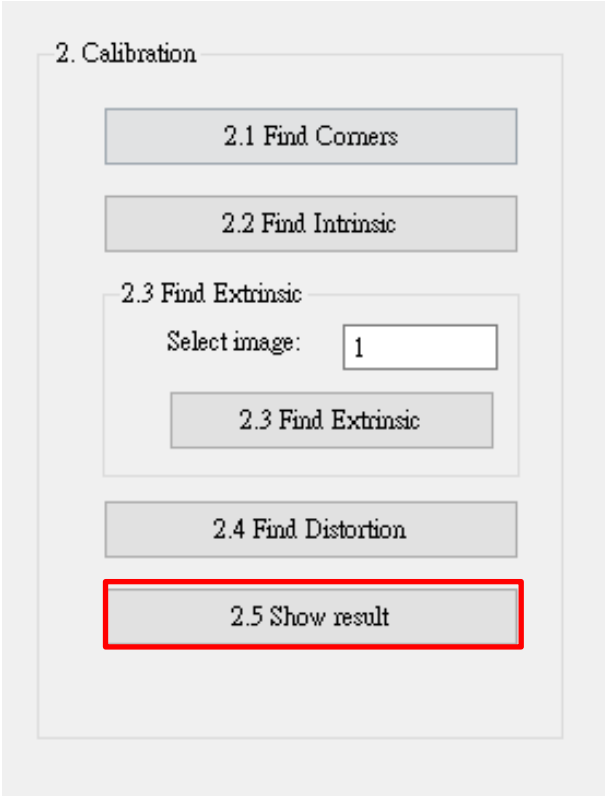


2.5 Show the undistorted result (4%)

(出題：West)

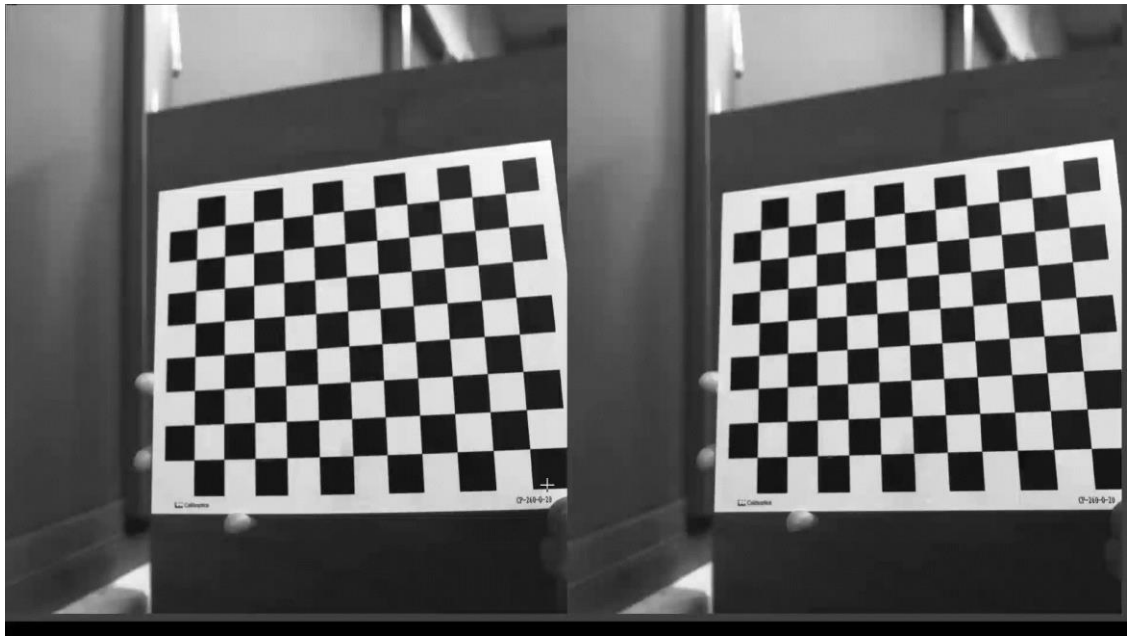
- Given: 15 images
- Q: 1) Undistort the chessboard images.

  2) Show distorted and undistorted images.
- Hint:
  - cv::undistort(…) or cv::initUndistortRectifyMap(…)
  - OpenCV Textbook Chapter 11 (P.398 ~ p.400)



Distorted image

Undistorted image

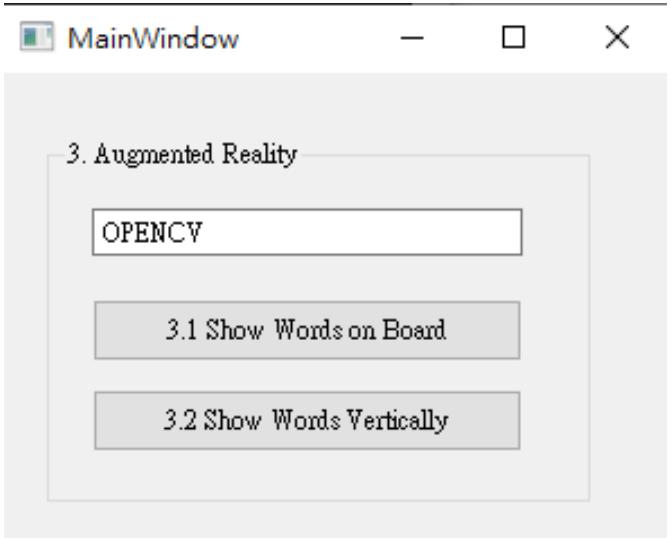


(出題：Keter)

3. (20%) Augmented Reality

3.1 (10%) Show words on board

3.2 (10%) Show words vertically

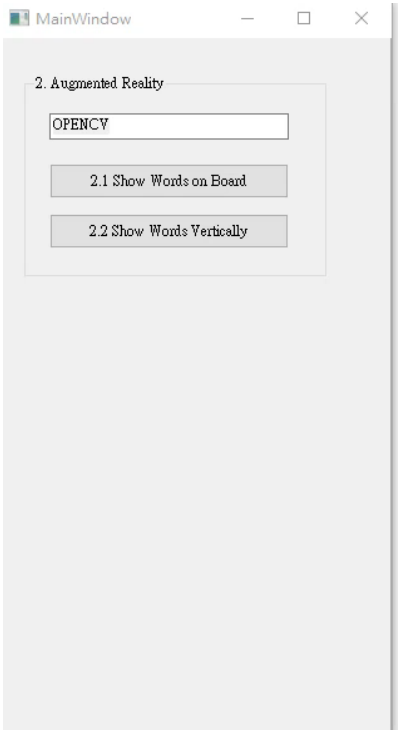


3. (20%) Augmented Reality

(出題：Keter)

- Given: 5 images: 1~5.bmp
- Q:
  1) Calibrate 5 images to get intrinsic, distortion and extrinsic parameters
  2) Input a “Word” less than 6 char in English in the textEdit box
  3) Derive the shape of the “Word” by using the provided library
  4) Show the “Word” on the chessboards images(1.bmp to 5.bmp)
  5) Show the “Word” vertically on the chessboards images(1.bmp to 5.bmp)
  6) Click the button to show the “Word” on the picture. Show each picture for 1 second (total 5 images)

Demo:



Hint : Textbook Chapter 11,
p.387~395 Calibration
p.405~412 Projection

cv2.calibrateCamera()
cv2.projectPoints()

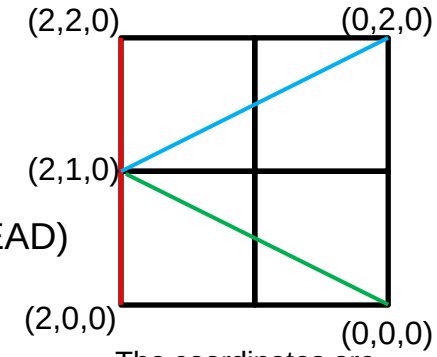
3. (20%) Augmented Reality

(出題：Keter)

- Guides and Requirements:
  1) How to use the library: (alphabet_lib_onboard.txt, alphabet_lib_vertical.txt)
     - Use OpenCV function to read and derive the array or matrix of the char
       Here take 'K' in 'alphabet_lib_onboard.txt' for example

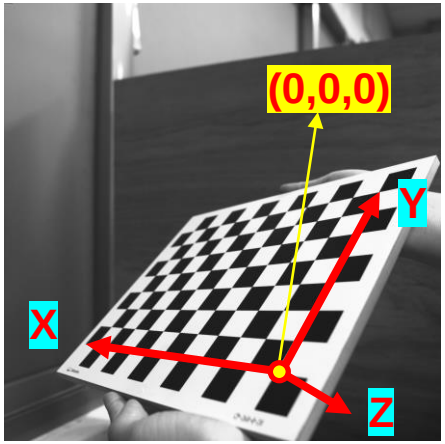
       Ex (Python):
       fs = cv2.FileStorage('alphabet_lib_onboard.txt', cv2.FILE_STORAGE_READ)
       ch = fs.getNode('K').mat() ➔ get the lines of 'K'

       ch = [[[2, 2, 0], [2, 0, 0]],
             [[0, 2, 0], [2, 1, 0]],
             [[2, 1, 0], [0, 0, 0]]]

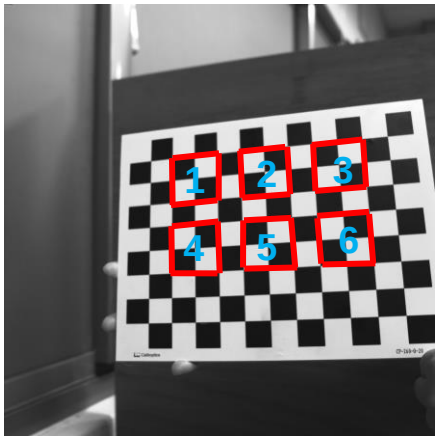


The coordinates are in Word Frame

     - 'K' consist of 3 lines, so the 'ch array' consists 3 pairs of 3D coordinates in Word Frame representing two ends of the line shown in the upper right image.
  2) Chessboard Coordinates
     - The chessboard x, y, z axis and (0,0,0) coordinate are shown in the bottom left image
     - Each Char should be place in the order and position shown in the bottom right image



Chessboard Frame



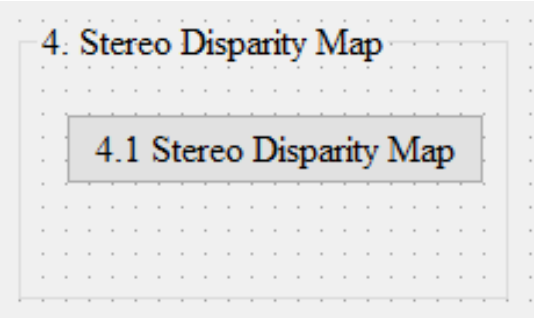
Position and Order

4. (20%) Stereo Disparity Map

(出題：Aaron)

4.1 (10%) Stereo Disparity Map

4.2 (10%) Checking the Disparity Value



4.1 (10%) Stereo Disparity Map

(出題：Aaron)

- Given: a pair of images, imL.png and imR.png (have been rectified)
- Q:
  - Find the disparity map/image based on Left and Right stereo images
- Guides:

(1) Window Size: Must be odd and within the range [5, 255]

(2) Search range and direction:

- Disparity range:
  - Must be positive and divisible by 16.
  - Map disparity range to gray value range 0~255 for the purpose of visualization.
- If the left image is the reference image (the one used to cal. depth info for each pixel of that img), then the search direction at right image will go from the right to left direction.

Camera information: baseline=342.789mm,
focal length=4019.284 pixel,
$c_x^{right} - c_x^{left}$=279.184 pixel

➢ Hint: OpenCV Textbook Chapter 12 (P.451)
StereoBM::create(256, 25)

OpenCV Textbook Chapter 11 (P.372-373) & OpenCV Textbook Chapter 12 (P.436)

imL.png

Left Image (Reference Image)

imR.png

Right Image

Result

4.2 (10%) Checking the Disparity Value

(出題：Aaron)

- Given: a pair of images, imL.png and imR.png and disparity map from Q4.1.
- Q:
  - Click at left image and draw the corresponding dot at right image.
- 1) Click at left image and draw the dot on the right image at accurate position.
- 2) User should allow to repeat 1).

➢ Note: Click at gray position at disparity map result from Q.4.1, ignore the position with 0 disparity(e.g. Failure case).

Result

5.0 (20%)Dogs and Cats classification Using ResNet50 (出題：育成)

1) Dataset introduction:

(1) ASIRRA (Animal Species Image Recognition for Restricting Access) is a HIP(Human Interactive Proof) that works by asking users to identify photographs of cats and dogs, that's supposed to be easy for people to solve, but difficult for computers. Now we can use artificial intelligence technology to achieve this goal.

(2) The dataset includes 12501 photos of cats and 12501 photos of dogs. You need to download them in Reference below(R2), and split the training set, validation set and test set using 8:1:1 ratio in both dogs and cats directory.

2) Environment Requirement

(1) Python
(2) Tensorflow
(3) Opencv-contrib-python
(4) Matplotlib

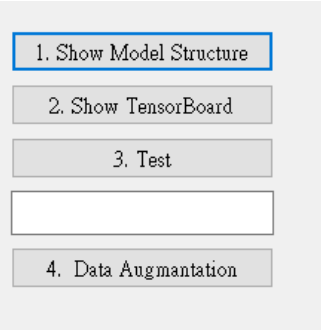


Fig. *Example UI*

layer name	output size	18-layer	34-layer	50-layer
conv1	112×112			7×7, 64, stride 2
				3×3 max pool, stride 2
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1			average pool, 1000-d fc, softm
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9

Fig. *ResNet's Network Architecture*

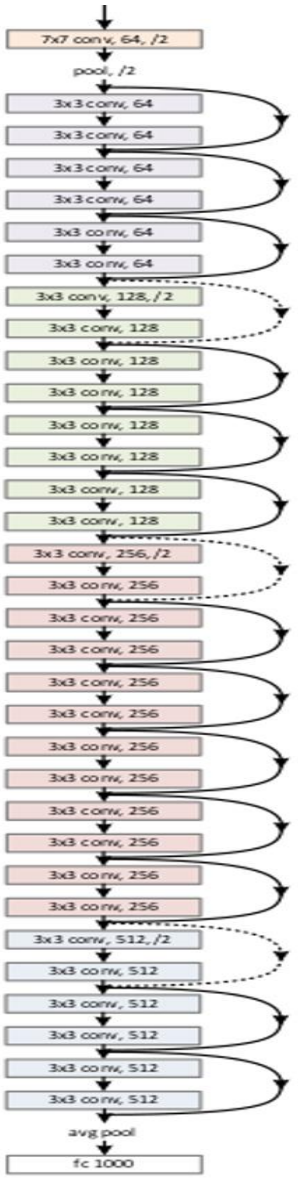


Fig. *ResNet50's Schematic Diagram*

R. Reference

R1) Deep Residual Learning for Image Recognition
R2) https://www.microsoft.com/en-us/download/details.aspx?id=54765 (ASIRRA)

5.1 Construct and show summary of your model structure by printing out on the terminal. **(5%)**

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]

Fig. *Summary of ResNet50*

5.2 Training at home at least 5 epochs and use TensorBoard to monitor, then save the final **screenshot of TensorBoard** (**5%**, Use other tools can get 3%).

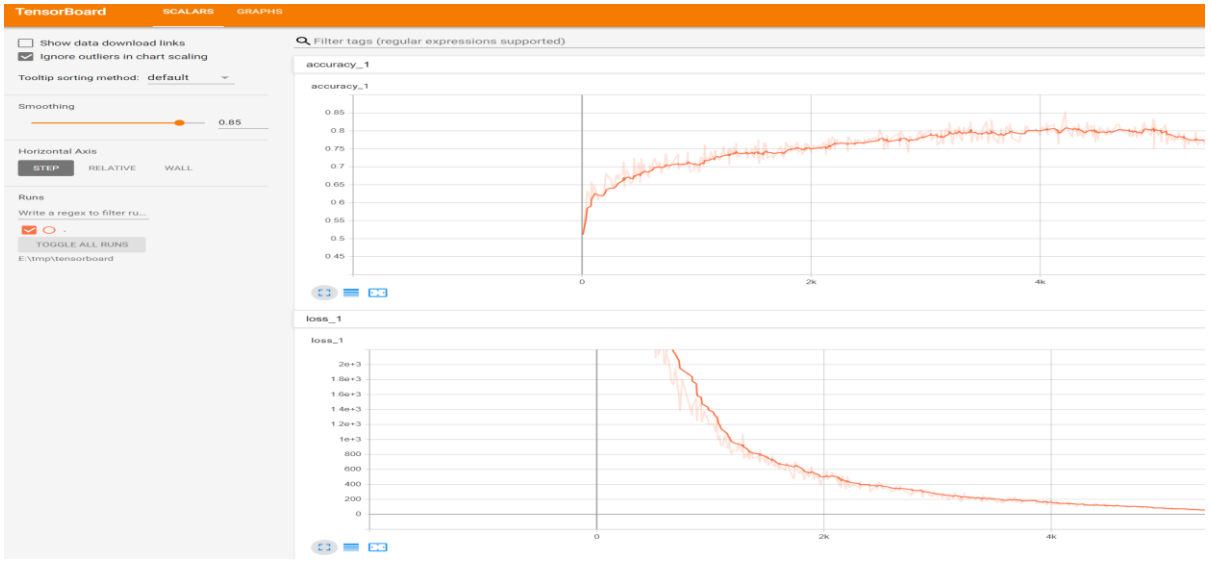


Fig. *Example of training with TensorBoard*

5.3 **Randomly select** a picture from the test set and mark its predicted category.(**5%**)

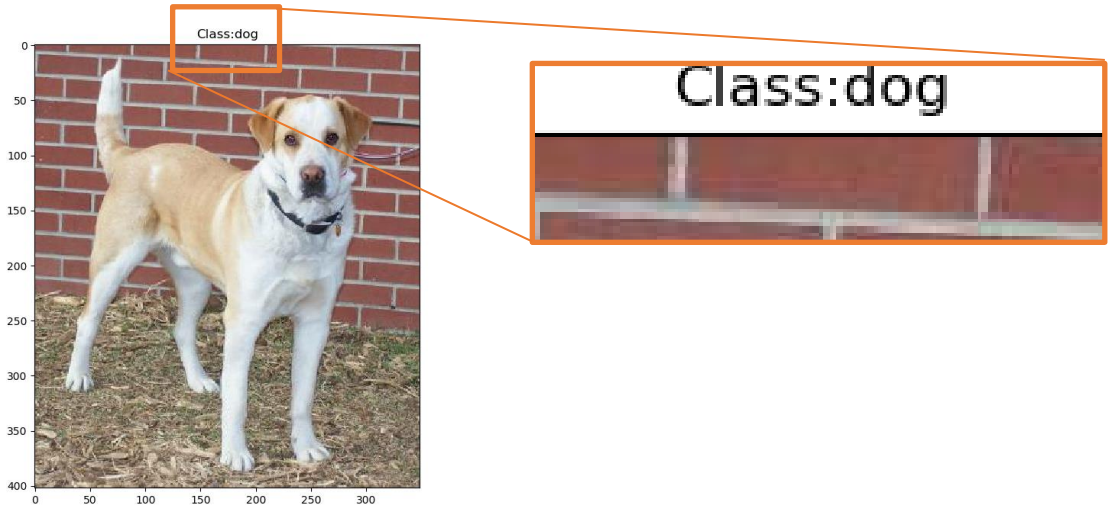


Fig. *Classification display example*

Algorithm 1: Random Erasing Procedure

Input : Input image I;
Image size W and H;
Area of image S;
Erasing probability p;
Erasing area ratio range s_l and s_h;
Erasing aspect ratio range r_1 and r_2.
Output: Erased image I^*.
Initialization: $p_1 \leftarrow$ Rand (0, 1).

if $p_1 \geq p$ **then**
  $I^* \leftarrow I$;
  return I^*.
else
  while *True* **do**
    $S_e \leftarrow$ Rand $(s_l, s_h) \times S$;
    $r_e \leftarrow$ Rand (r_1, r_2);
    $H_e \leftarrow \sqrt{S_e \times r_e}$, $W_e \leftarrow \sqrt{\frac{S_e}{r_e}}$;
    $x_e \leftarrow$ Rand $(0, W)$, $y_e \leftarrow$ Rand $(0, H)$;
    if $x_e + W_e \leq W$ *and* $y_e + H_e \leq H$ **then**
      $I_e \leftarrow (x_e, y_e, x_e + W_e, y_e + H_e)$;
      $I(I_e) \leftarrow$ Rand (0, 255);
      $I^* \leftarrow I$;
      return I^*.
    end
  end
end

Fig. *Random-Erasing algorithm*

5.4 Train another model with Random-Erasing or any other data augmentation method. **Write the code of your augmentation method(3%)** and **draw the comparison table of accuracy**, save it as a picture.**(2%)**

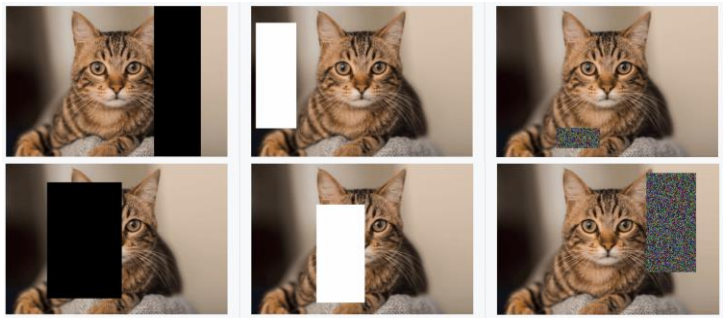


Fig. *Examples of the use of Random-Erasing*

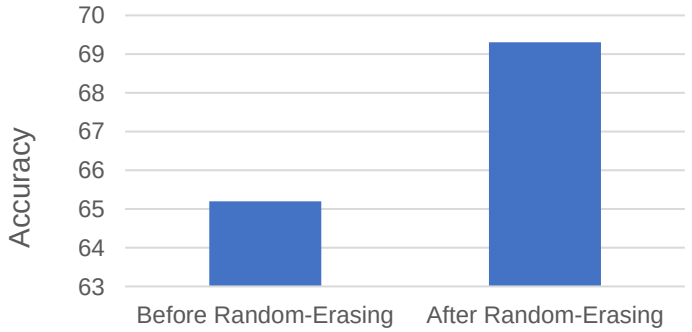


Fig. *Random-Erasing effect comparison example*

R. Reference

Random Erasing Data Augmentation