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:

少鈞: nckubot65904@gmail.com

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

10:00~12:00, Fri.

At CSIE 9F Robotics Lab.

Notice (1/2)

- Copying homework is strictly prohibited!! Penalty: Score will be zero for both persons!!
- Due date => 09:00:00, 2022/12/12 (Mon.)

  No delay. Penalties for late homework:
  - ➢ Up to 7 days late, loss of 50% of the score awarded
  - ➢ After 7 days, the score will be marked as 0.
- You must attend the demonstration, or your score will be 0. The demonstration schedule will be announced on NCKU moodle.
- You must make a GUI and follow the format, or you will get some penalties.
- Upload to => 140.116.154.1 -> Upload/Homework/Hw2

  Upload/Homework/Hw2_05
  - ➢ User ID: opencvdl2022 Password: opencvdl2022
- Format
  - ➢ Filename: Hw2_StudentID_Name_Version.rar

    Hw2_05_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*(excluding the pictures, only source code)

    *note: remove your “Debug” folder to reduce file size

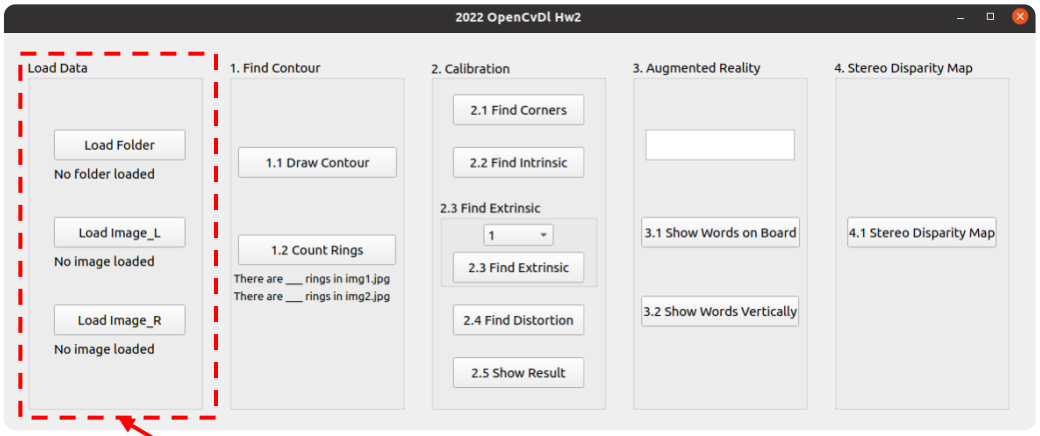
Notice (2/2)

- Python
  - 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)

Assignment scoring (Total: 100%)

1. (20%) Image Processing (出題：Mei)
   - 1.1 (15%) Draw Contour
   - 1.2 (5%) Count Rings
2. (20%) Camera Calibration (出題：Jessica)
   - 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
3. (20%) Augmented Reality (出題：Ming)
   - 3.1 (10%) Show words on board
   - 3.2 (10%) Show words vertically
4. (20%) Stereo Disparity Map (出題：Maton)
   - 4.1 (10%) Stereo Disparity Map
   - 4.2 (10%) Checking the Disparity Value
5. (20%) Train a Cat-Dog Classifier Using ResNet50 (出題：Benjamin)
   - 5.1 (3%) Load the dataset and resize images
   - 5.2 (3%) Plot class distribution of training dataset
   - 5.3 (3%) Show the structure of ResNet50 model
   - 5.4 (3%) Set up 2 kinds of loss functions to train 2 ResNet50 models
   - 5.5 (3%) Compare the accuracies of 2 ResNet50 models on validation dataset
   - 5.6 (4%) Use the better-trained model to run inference and show the predicted class label

Question 5 needs to upload separately.



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

1. (20%) Find Contour

(出題：Mei)

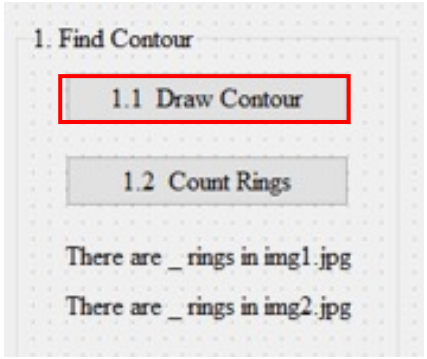
1.1 (15%) Draw Contour

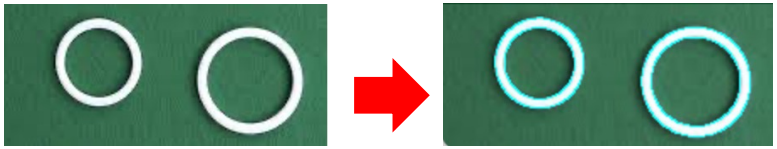
1.2 (5%) Count Rings

1.1 Find Contour – Draw Contour

(出題：Mei)

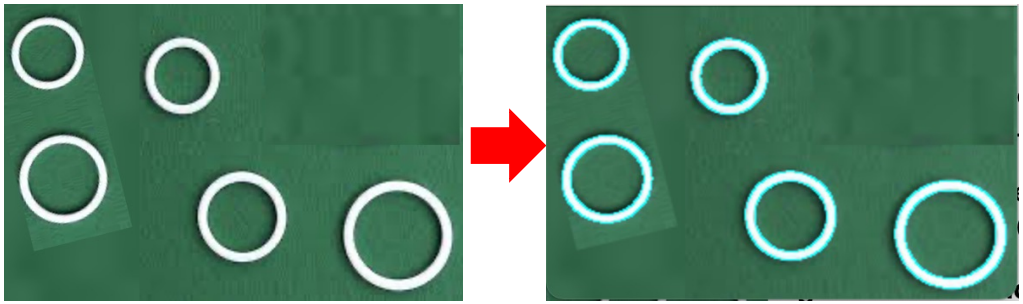
- Given: two color images, “img1.jpg” and “img2.jpg”
- Q: 1) **Draw Contour**: Using OpenCV functions to find the contours of rings in two images.
- Hint: Textbook Chapter 8, p.234 ~ p.241
  1. RGB ➔ Resize(1/2) ➔ Grayscale
  2. Remember to remove the noise. (use **Gaussian Blur**)
  3. Using some **edge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





img1.jpg

Draw Contours



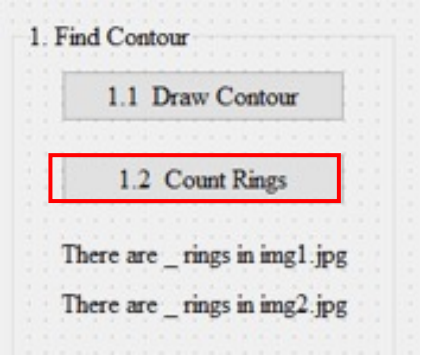
img2.jpg

Draw Contours

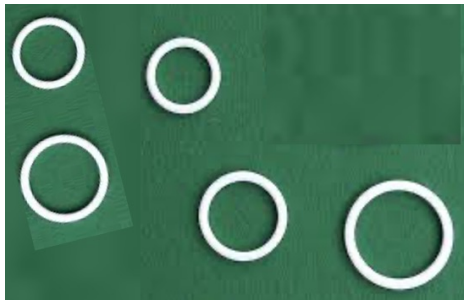
1.2 Find Contour – Count Rings

(出題：Mei)

- 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



img1.jpg



img2.jpg

2. (20%) Camera Calibration

(出題：Jessica)

2.1 (4%) Corner detection

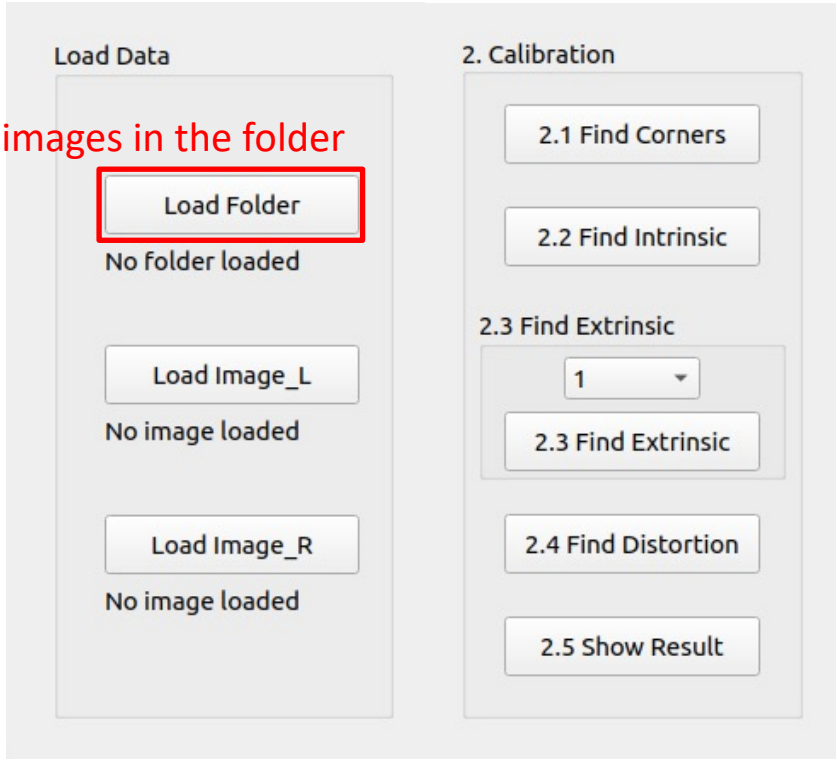
2.2 (4%) Find the intrinsic matrix

2.3 (4%) Find the extrinsic matrix

2.4 (4%) Find the distortion matrix

2.5 (4%) Show the undistorted result

Load all images in the folder



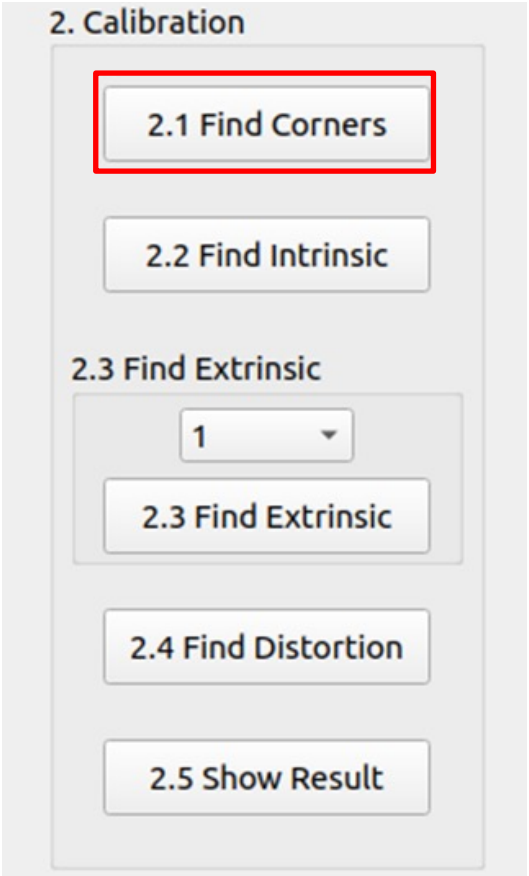
2.1 Corner Detection (4%)

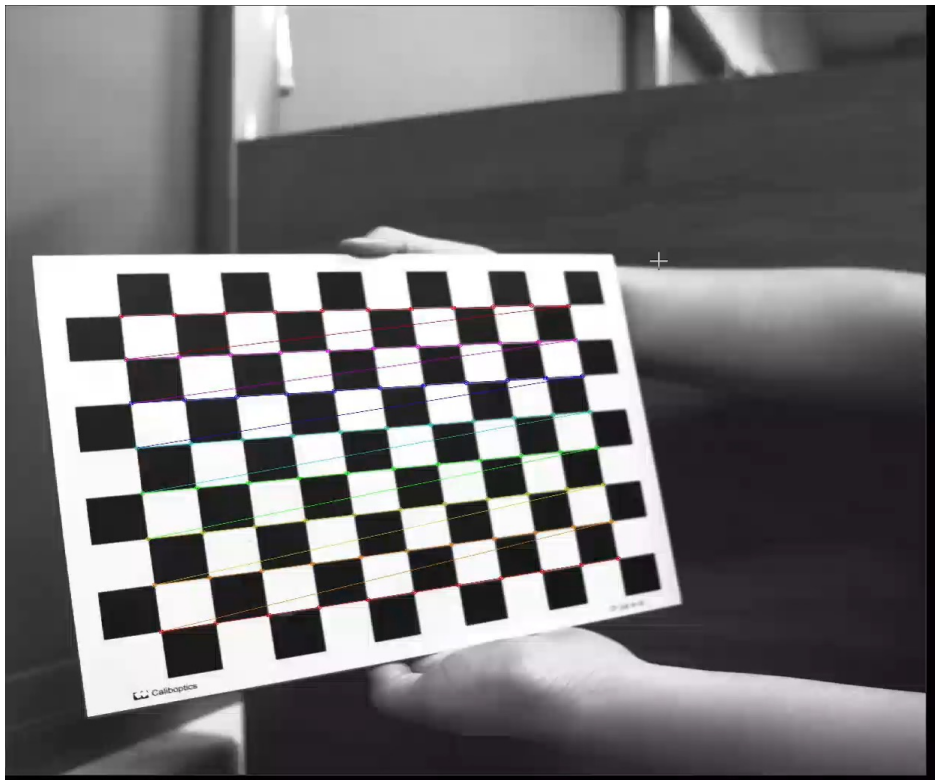
(出題：Jessica)

- 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：Jessica)

- Given: 15 images, 1.bmp ~ 15.bmp
- Q: 1) Find the intrinsic matrix ():

$$\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

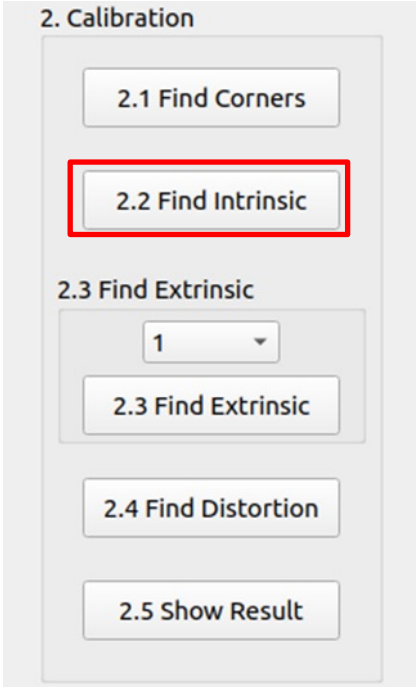
2) Click button “2.2” and then show the result on the console window.

- 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：Jessica)

- 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.

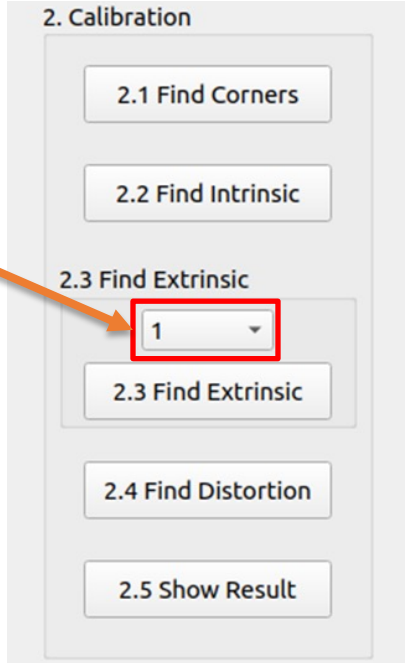
- Output format:

```
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%)

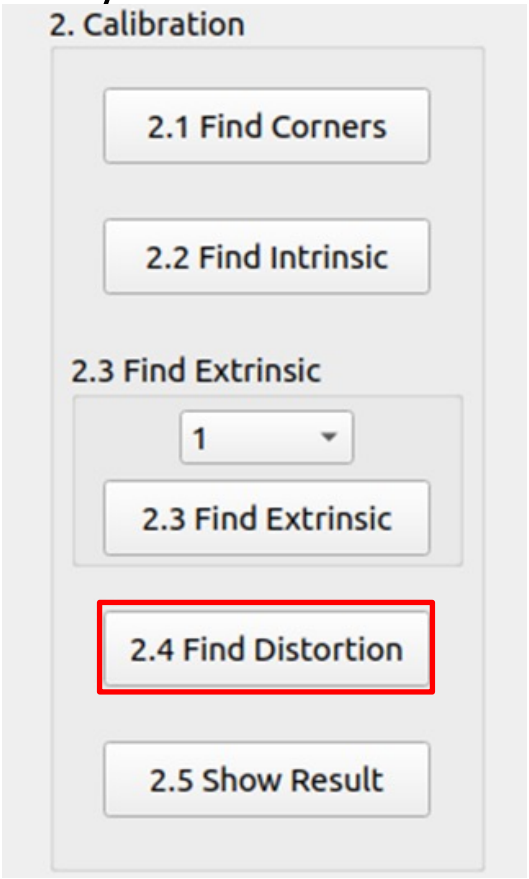
(出題：Jessica)

- 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)

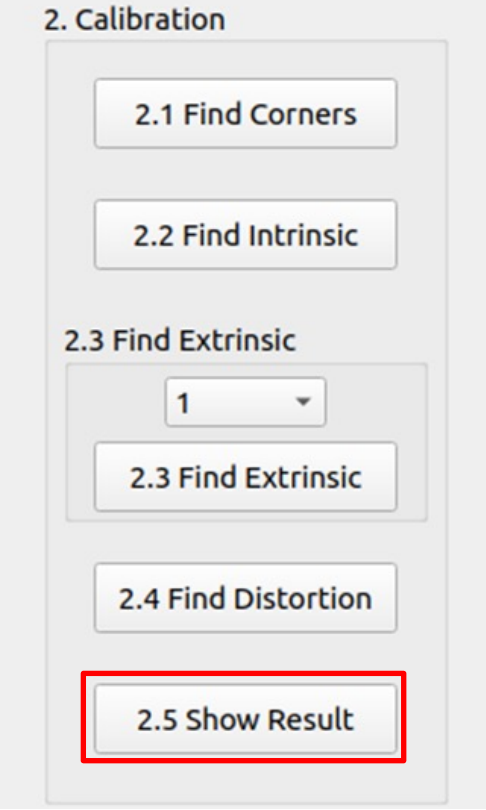


2.5 Show the undistorted result (4%)

(出題：Jessica)

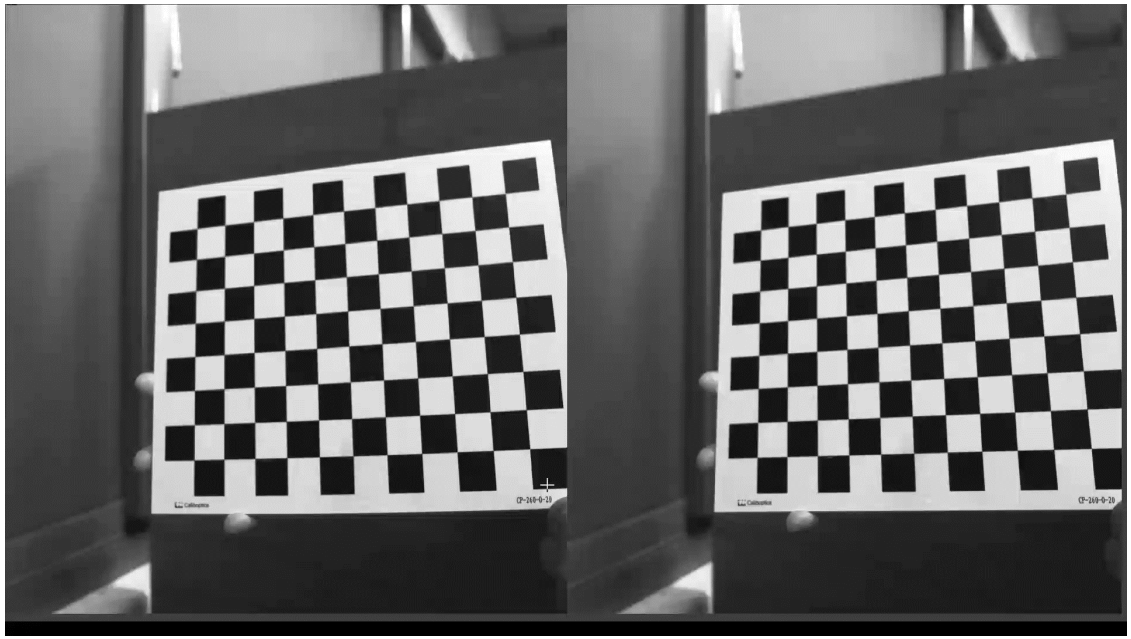
- Given: 15 images
- Q: 1) Undistort the chessboard images.
  2) Show each distorted and undistorted images 0.5 seconds.

- Hint:
  - cv::undistort(…) or cv::initUndistortRectifyMap(…)
  - OpenCV Textbook Chapter 11 (P.398 ~ p.400)



Distorted image

Undistorted image

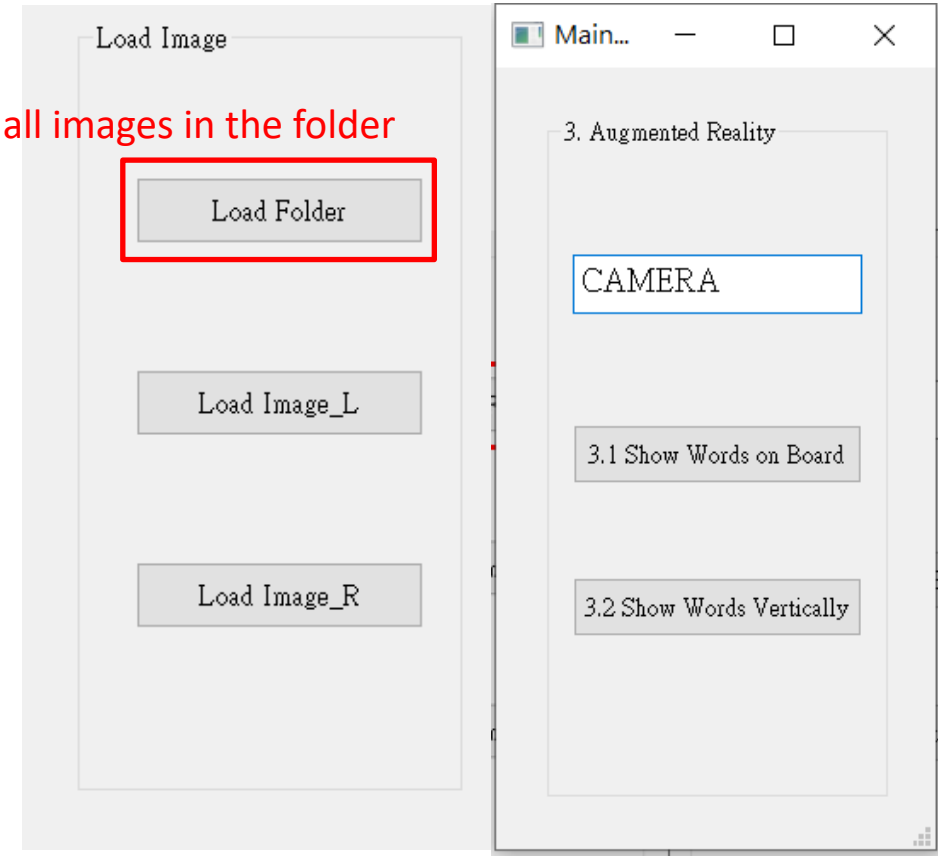


3. (20%) Augmented Reality

(出題：Ming)

3.1 (10%) Show words on board

3.2 (10%) Show words vertically

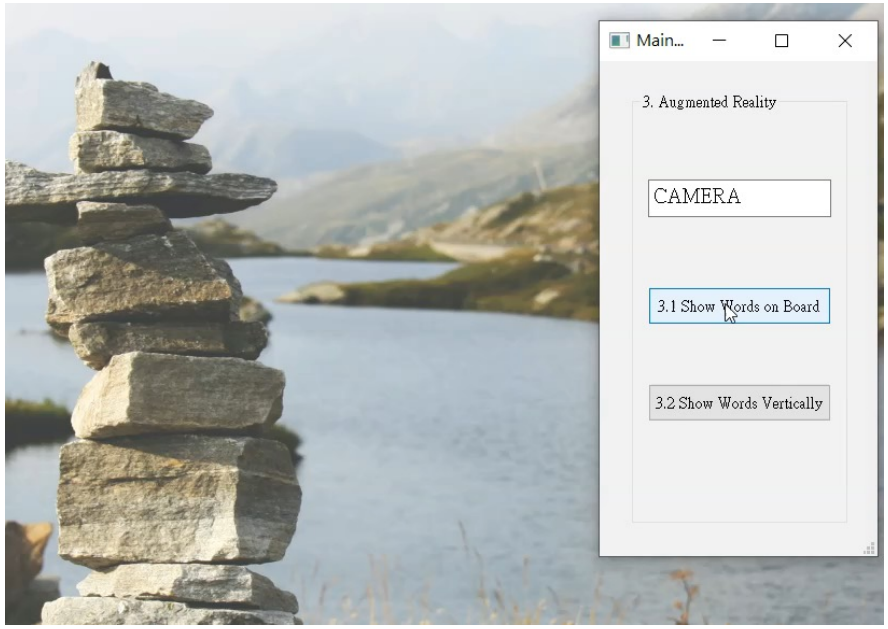


3. (20%) Augmented Reality

(出題：Ming)

- 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

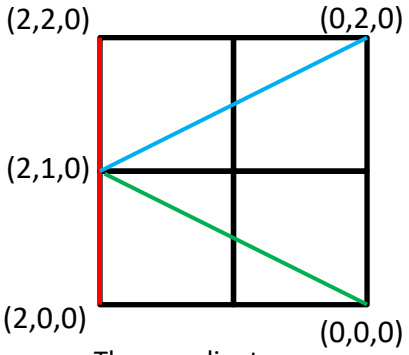
(出題：Ming)

- 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'

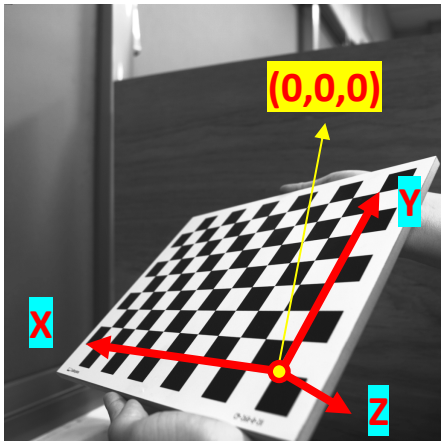
       ch = [[[2, 2, 0], [2, 0, 0]],
             [[0, 2, 0], [2, 1, 0]],
             [[2, 1, 0], [0, 0, 0]]]

     - '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.

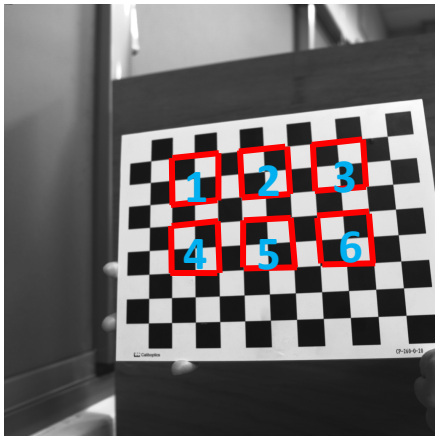


The coordinates are in Word Frame

  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



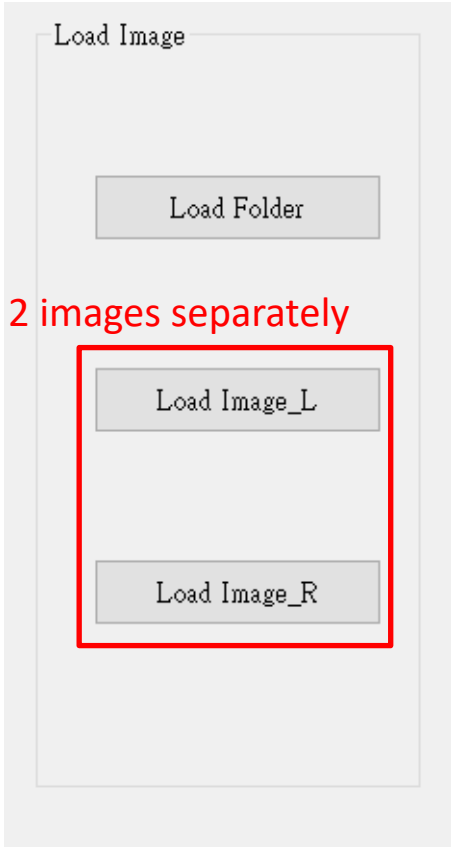
Position and Order

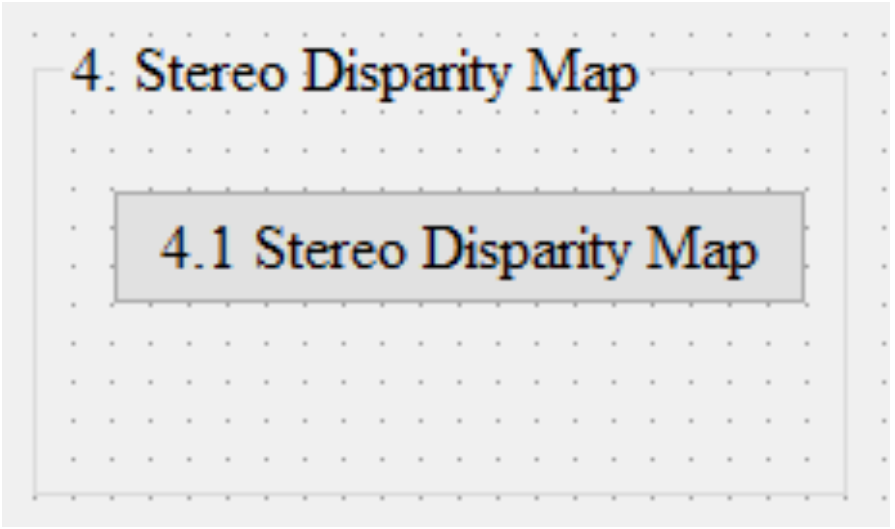
4. (20%) Stereo Disparity Map

(出題：Maton)

4.1 (10%) Stereo Disparity Map

4.2 (10%) Checking the Disparity Value





4.1 (10%) Stereo Disparity Map

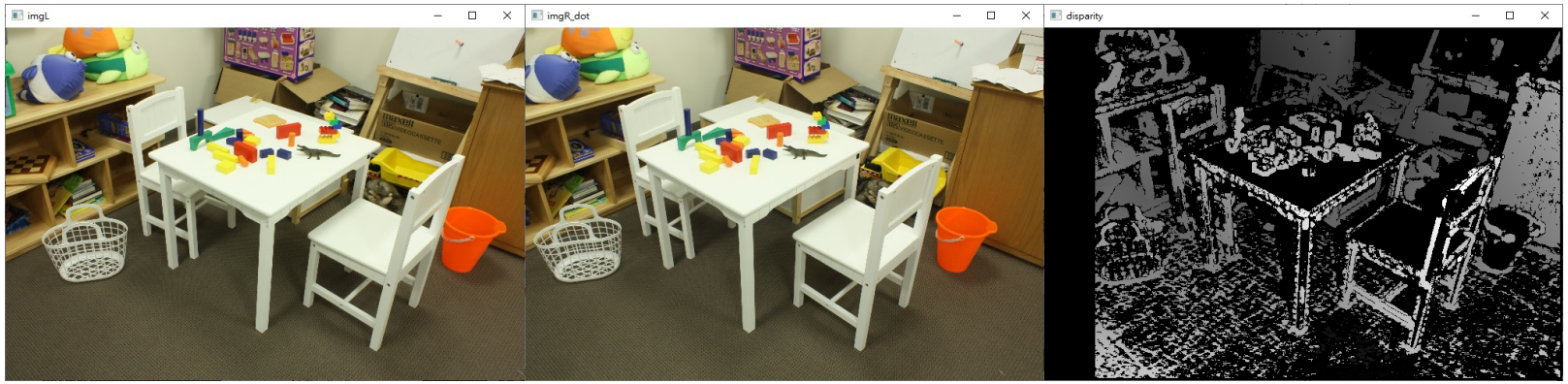
(出題：Maton)

- 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: 1) baseline=342.789mm,
2) focal length=4019.284 pixel,
3) $c_x^{right} - c_x^{left}$=279.184 pixel

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

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



imL.png
Left Image (Reference Image)

imR.png
Right Image

Result

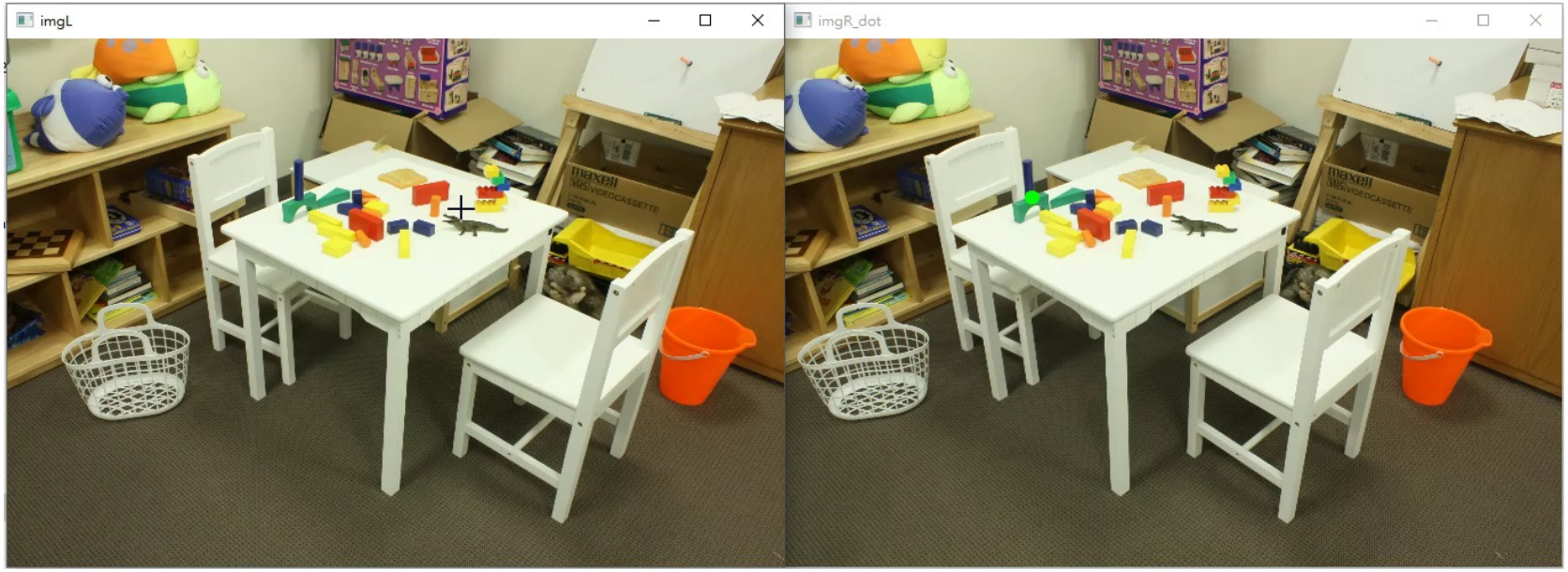
4.2 (10%) Checking the Disparity Value

(出題：Maton)

- ❑ 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 Video:



5. Train a Cat-Dog Classifier Using ResNet50 (20%)

(出題：Benjamin)

5.1 (3%) Load the dataset and resize images

5.2 (3%) Plot class distribution of training dataset

5.3 (3%) Show the structure of ResNet50 model

5.4 (3%) Set up 2 kinds of loss functions to train 2 ResNet50 models

5.5 (4%) Compare the accuracies of 2 ResNet50 models on validation dataset

5.6 (4%) Use the better-trained model to run inference and show the predicted class label

5.0 Train a Cat-Dog Classifier Using ResNet50

(出題：Benjamin)

1. Objective
   1) Learn how to train a ResNet50 model to classify images of cats and dogs
   2) Learn how to deal with imbalanced data
2. ResNet50
   1) Residual learning: avoid degradation problems
   2) Bottleneck: build a deeper network without additional parameters
3. Cats and Dogs Dataset (modified from Kaggle Cats and Dogs Dataset)
   1) Data type: JPG images
   2) 2 classes: Cat and Dog
   3) Datasets
      (1) Path: /Download/Homework/Hw2/Dataset_OpenCvDl_Hw2_Q5.zip
      (2) Training dataset: 16,200 JPG images in total. However, the dataset is imbalanced.
      (3) Validation dataset: 1,800 JPG images in total.
      (4) Inference dataset: 10 JPG images in total. It is for testing the inference function in your GUI program.

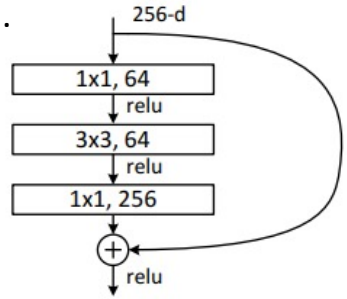


Figure: Residual Block (with bottleneck)

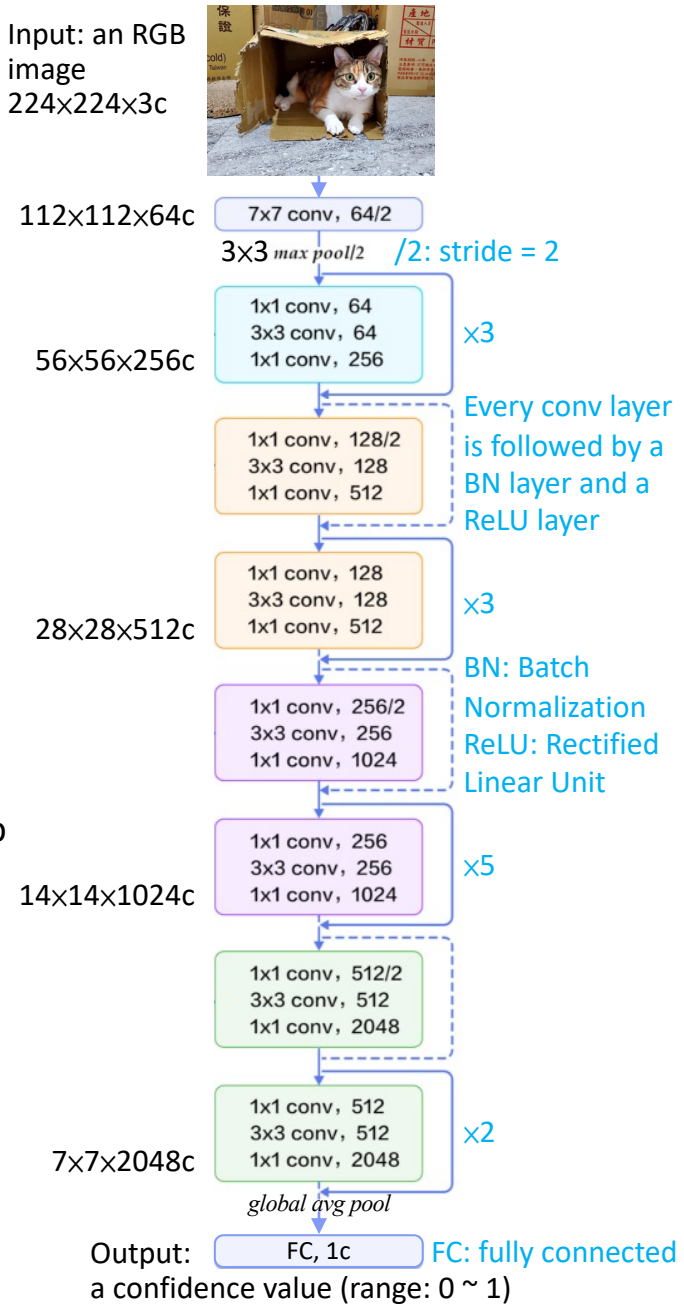


Figure: ResNet50 model structure

R. Reference

1) Deep Residual Learning for Image Recognition
2) Kaggle Cats and Dogs Dataset

5.0 Train a Cat-Dog Classifier Using ResNet50

(出題：Benjamin)

4. Requirements

1) Train ResNet50 models with TensorFlow or PyTorch
2) Every chart should have a chart title and axis titles
3) Organize the files in this structure:

```
Hw2_05_StudentID_Name_Version // project folder
|-- model            // folder to put trained models
|-- inference_dataset
    |-- Cat
    |-- Dog
|-- main.py          // codes for your GUI program
|-- train.py         // codes for model training
|--  ⋮               // other files or folders you need
```

Notice: Please include the inference dataset in your homework file.

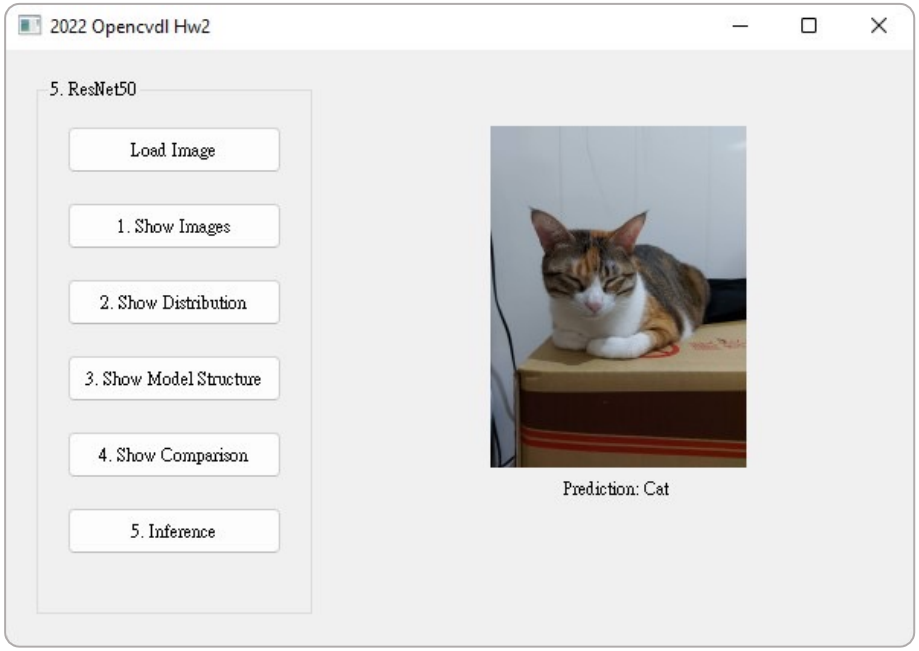
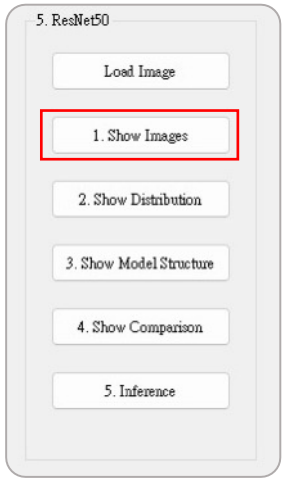


Figure: GUI example

5.1 (3%) Load the dataset and resize images

(出題：Benjamin)

1. Objective
   1) At home:
      (1) Load the training dataset and validation dataset
      → Hint:
      (a) TensorFlow (tutorial): tf.keras.utils.image_dataset_from_directory()
      (b) PyTorch (tutorial): torch.utils.data.Dataset
      (2) Resize images to 224×224×3c (RGB)
      → Hint:
      (a) TensorFlow: tf.keras.utils.image_dataset_from_directory(image_size=(?, ?))
      (b) PyTorch (tutorial): torchvision.transform
   2) When the demo:
      (1) Click the button "1. Show Images"
      (2) Load the inference dataset
      (3) Resize images to 224×224×3c (RGB)
      (4) Get 1 image from each class in the inference dataset
      (5) Show images in a new window
      → Hint: use matplotlib.pyplot functions to show images (tutorial):
      (a) figure()
      (b) imshow()
      (c) subplot()
      (d) title()



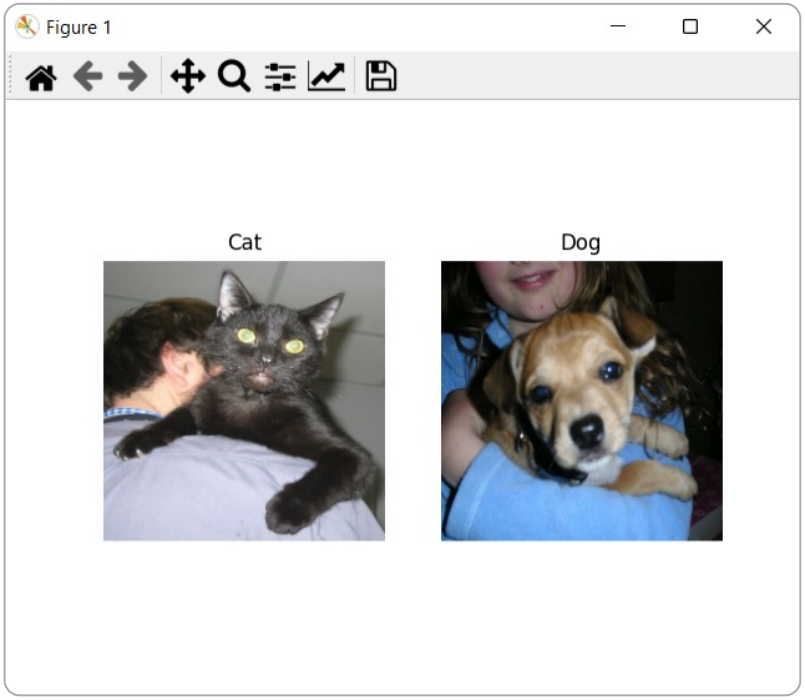


Figure: 1 image from each class
Notice: this is an example, the images might differ

5.2 (3%) Plot class distribution of training dataset (出題：Benjamin)

1. Objective
   1) At home:
      (1) Load the training dataset
      (2) Iterate through the dataset and count the number of images of each class
      (3) Plot the class distribution
          → Hint: use matplotlib.pyplot functions to plot
            (a) figure()
            (b) bar()
            (c) xticks()
            (d) ylabel()
      (4) Save the figure
   2) When the demo:
      (1) Click the button "2. Show Distribution"
      (2) Show the saved figure of class distribution in a new window

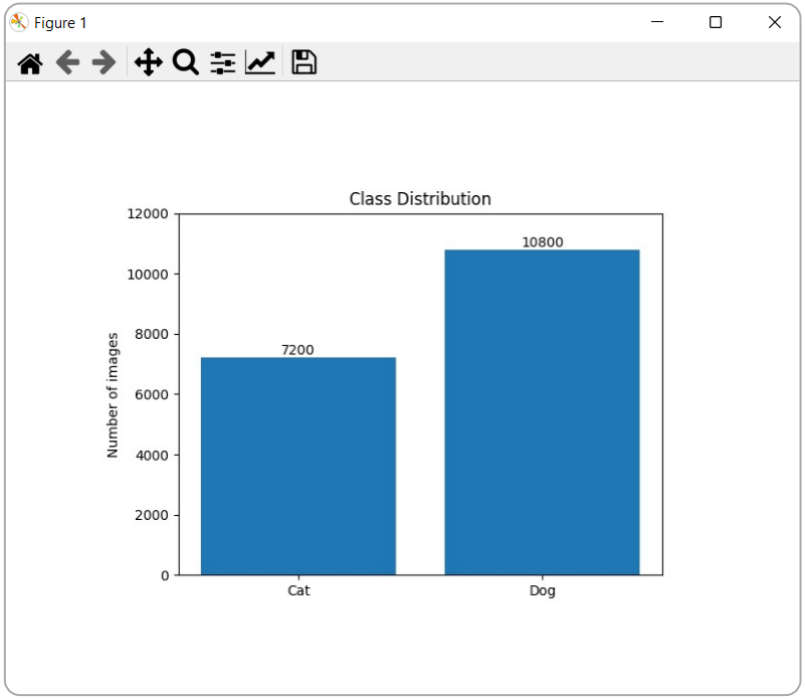


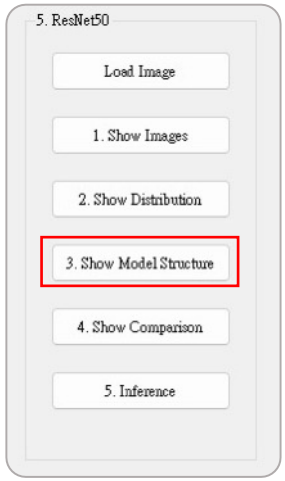
Figure: Class Distribution
Notice: this is an example, the numbers might differ

5.3 (3%) Show the structure of ResNet50 model

(出題：Benjamin)

1. Objective

1) At home:
   (1) Build a ResNet50 model
   → Hint:
   (a) TensorFlow: tf.keras.applications.resnet50.ResNet50()
   (b) PyTorch: torchvision.models.resnet50()
   (2) Replace the output layer to a FC (Fully Connected) layer of 1 node with a Sigmoid activation function
   → Hint:
   (a) TensorFlow (tutorial): tf.keras.layers.Dense(1, activation='sigmoid')
   (b) PyTorch (tutorial): torch.nn.Linear(2048, 1), torch.nn.Sigmoid
   If the class label of Cat is 1, the output value (range: 0 ~ 1) should be close to 1 for cat images, and vice versa.
   (3) Run the function to show the structure in the terminal
   → Hint:
   (a) TensorFlow: tf.keras.Model.summary()
   (b) PyTorch: torchsummary
2) When the demo:
   (1) Click the button "3. Show Model Structure"
   (2) Run the function to show the structure in the terminal



Layer (type)	Feature map shape (B, H, W, C)	Num. of param.	Layer connecting 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 (BatchNormalization)	(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]']

Figure: the structure of ResNet50 model

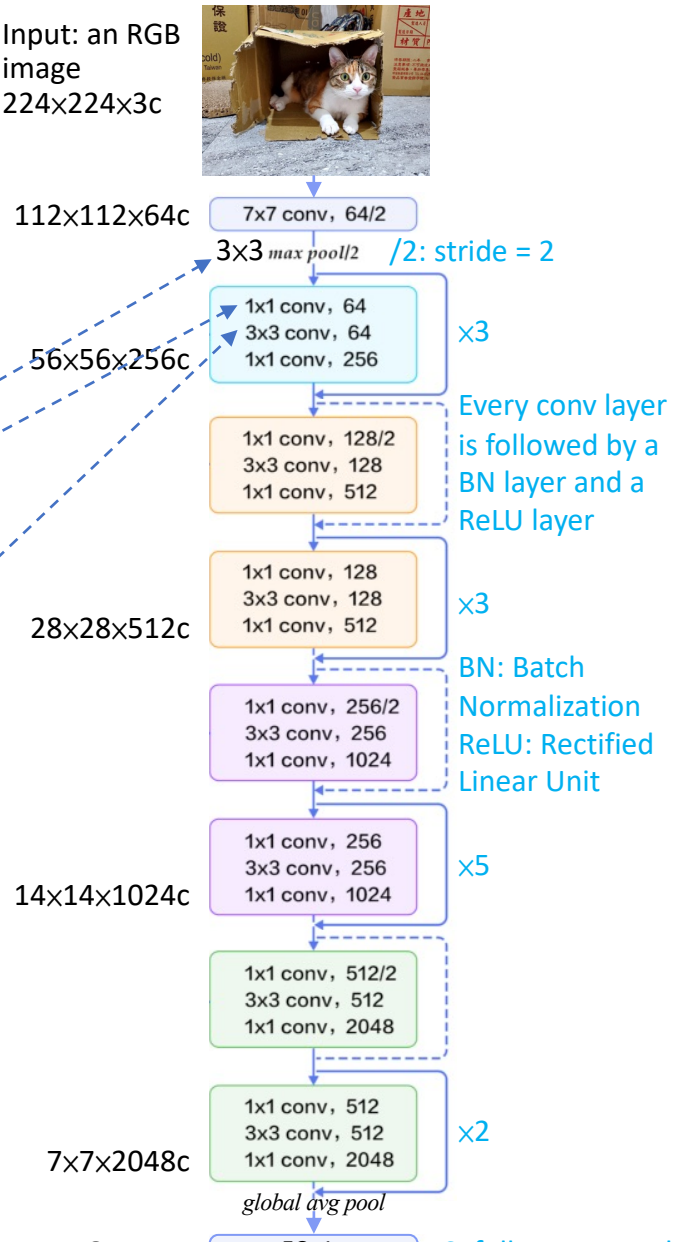
5.3 (3%) Show the structure of ResNet50 model

(出題：Benjamin)

2. How to read the structure

Layer (type)	Feature map shape (B, H, W, C)	Num. of param.	Layer connecting 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 (BatchNormalization)	(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 (BatchNormalization)	(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]']
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	['pool1_pool[0][0]']
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	['conv2_block1_2_relu[0][0]']
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_0_conv[0][0]']
conv2_block1_3_bn (BatchNormalization)	(None, 56, 56, 256)	1024	['conv2_block1_3_conv[0][0]']

Figure: the structure of ResNet50 model



5.4 (3%) Set up 2 kinds of loss functions to train 2 ResNet50 models

(出題：Benjamin)

1. Objective
   1) At home:
      (1) Set up Focal Loss (α-balanced) and Binary Cross Entropy in codes for model training (train.py)
      → Hint:
      (a) TensorFlow: tfa.losses.SigmoidFocalCrossEntropy(), tf.keras.losses.BinaryCrossEntropy()
      (b) PyTorch: torchvision.ops.sigmoid_focal_loss(), torch.nn.function.binary_cross_entropy()
      Because you have to use Focal Loss, which accepts 1 confidence value, you may replace the output layer of ResNet50 model to a FC layer of 1 node with a Sigmoid activation function.
      (2) Train 2 ResNet50 models with training dataset using 2 loss functions
      → Hint:
      (a) TensorFlow: tf.keras.Model.fit()
      (b) PyTorch (tutorial): write a for loop to train the model
   2) When the demo:
      (1) Show your codes about loss functions in train.py

```
60  ### Q5.4 Set up 2 kinds of loss functions to train 2 ResNet50 models
61  # 1st loss function
62  loss_function = tfa.losses.SigmoidFocalCrossEntropy(alpha=0.4, gamma=1.0)
63
64  # 2nd loss function
65  # loss_function = tf.keras.losses.BinaryCrossentropy()
66
67  optimizer = tf.keras.optimizers.Adam(learning_rate=8e-5)
68
69  # configure loss function and optimizer for training
70  model.compile(optimizer=optimizer, loss=loss_function, metrics=['accuracy'])
```

Train a model with Focal Loss

Train another model with Binary Cross Entropy

Figure: Code example (you can follow the hyperparameter settings)

5.4 (3%) Set up 2 kinds of loss functions to train 2 ResNet50 models

(出題：Benjamin)

2. What is Focal Loss?

- $\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$

1) $p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$

p: model's output confidence value for the input with class label y = 1 (range: $0 \sim 1$)

y: ground-truth class label (0 or 1)

2) $\alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1-\alpha & \text{otherwise,} \end{cases}$

α: class imbalance weighting factor (Usually set to $1 - \textit{frequency of } y = 1 \textit{ examples}$)

3) γ: focusing parameter (Usually set to 2)

- Focal Loss mitigates class imbalance by assigning more weights to hard examples and less weights to easy examples.
- Focal Loss is for binary classification, ex: Foreground and Background, Face and Non-Face, or Cat and Dog.
- Reference: Focal Loss for Dense Object Detection

Ex: we have

1) 30 examples with $y = 0$

2) 70 examples with $y = 1$

→ Frequency of $y = 1$ examples is $\frac{70}{100} = 0.7$

→ $\alpha = 1 - 0.7 = 0.3$

5.5 (4%) Compare the accuracies of 2 ResNet50 models on validation dataset

(出題：Benjamin)

1. Objective
   1) At home:
      (1) Validate 2 ResNet50 models with validation dataset
      → Hint:
      (a) TensorFlow: tf.keras.Model.evaluate()
      (b) PyTorch (tutorial): write a for loop to validate the model
      (2) Plot the accuracy values with a bar chart
      (3) Save the figure
   2) When the demo:
      (1) Click the button "4. Show Comparison"
      (2) Show the saved figure of accuracy comparison in a new window

Figure: Accuracy Comparison
Notice: this is an example, the numbers might differ

5.6 (4%) Use the better-trained model to run inference and show the predicted class label

(出題：Benjamin)

1. Objective

1) At home:

(1) Load the trained model

→ Hint:

(a) TensorFlow: tf.keras.Model.load_weights()

(b) PyTorch: torch.nn.Module.load_state_dict()

(2) Click the button "Load Image" to select 1 image arbitrarily

→ Hint: PyQt5.QtWidgets.QFileDialog.getOpenFileName()

(3) Show the loaded image in the GUI

(4) Resize the loaded image to 224×224×3c (RGB)

(5) Click the button "5. Inference" to run inference on the resized image

→ Hint:

(a) TensorFlow: tf.keras.Model.predict_step()

(b) PyTorch: pass an image when calling torch.nn.M object to run inference, ex: trained_model(img)

(6) Show the predicted class label

→ Hint: decide the class label with a threshold of the output value.

$$\text{Ex: class label} = \begin{cases} \text{Cat}, & output < thresh \\ \text{Dog}, & output \geq thresh \end{cases}$$

$$thresh = 0.5$$

2) When the demo: repeat the process