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:

West: player210421@gmail.com

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

10:00~12:00, Wed.

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 2020/12/31 (Thur.)
  - No delay. If you submit homework after deadline, you will get 0.
- Upload to => 140.116.154.1 -> Upload/Homework/OpenCv_Hw2
  - User ID: opencvdl2020 Password: opencvdl2020
- 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)

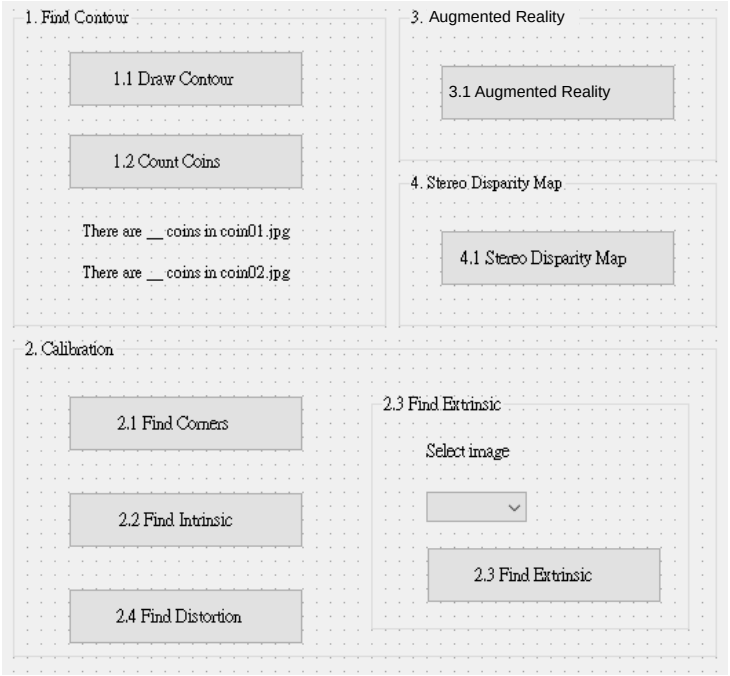
- Python (recommended)
  - Python 3.7 (https://www.python.org/downloads/)
  - opencv-contrib-python (3.4.2.17)
  - Matplotlib 3.1.1
  - UI framework: pyqt5 (5.15.1)

- 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： Shan)
   - 1.1 (10%) Draw Contour
   - 1.2 (10%) Count Coins
2. (20%) Camera Calibration （出題： Max)
   - 2.1 (5%) Corner detection
   - 2.2 (5%) Find the intrinsic matrix
   - 2.3 (5%) Find the extrinsic matrix
   - 2.4 (5%) Find the distortion matrix
3. (20%) Augmented Reality （出題： Oran)
4. (20%) Stereo Disparity Map（出題： Oran)
   - 4.1 (10%) Compute disparity map
   - 4.2 (10%) Calculate the depth
5. (20%) Dogs and Cats classification Using ResNet50 （出題： Bill)

UI Example



1. (20%) Find Contour

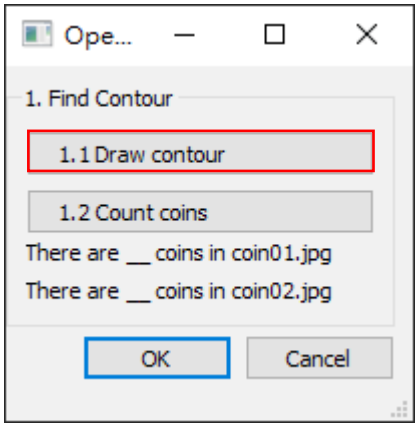
(出題： Shan)

1.1 (10%) Draw Contour

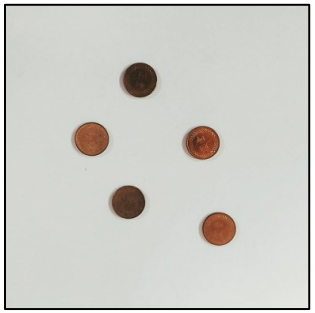
1.2 (10%) Count Coins

1.1 Find Contour – Draw Contour

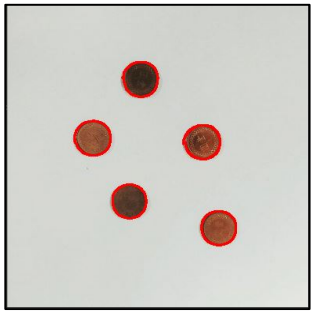
(出題： Shan)



- Given: two color images, “coin01.jpg” and “coin02.jpg”
- Q: 1) **Draw Contour**: Using OpenCV functions to find the contours of coins in two images.
- Hint: Textbook Chapter 8, p.234 ~ p.241
  1. RGB ➔ Grayscale ➔ Binary
  2. Remember to use **Gaussian Blur** to remove the noise.
  3. Using some **edge detection functions** to get better results. (Ex: cv2.Canny)

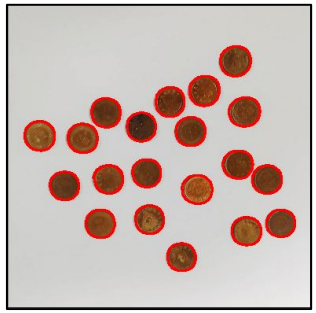


Coin01.jpg



Draw Contours

Coin02.jpg

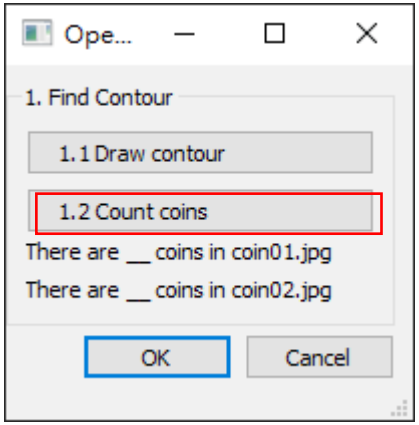


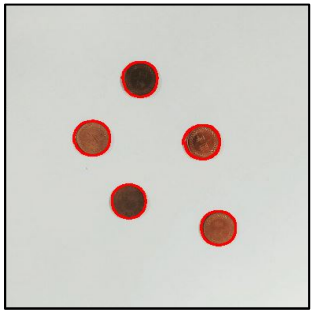
Draw Contours

1.2 Find Contour – Count Coins

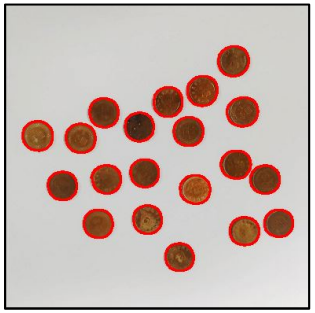
(出題： Shan)

- Given: two color images, “coin01.jpg” and “coin02.jpg”
- Q: 2) **Count Coins**: Using OpenCV functions to find how many coins in two images.
- Hint: Textbook Chapter 8, p.234 ~ p.241
  Calculate how many contours returned by `cv2.findContours`

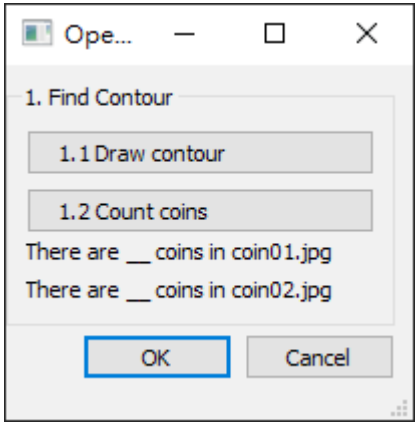




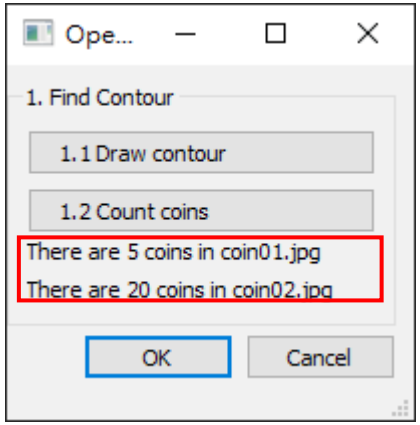
5 coins



20 coins



Original UI

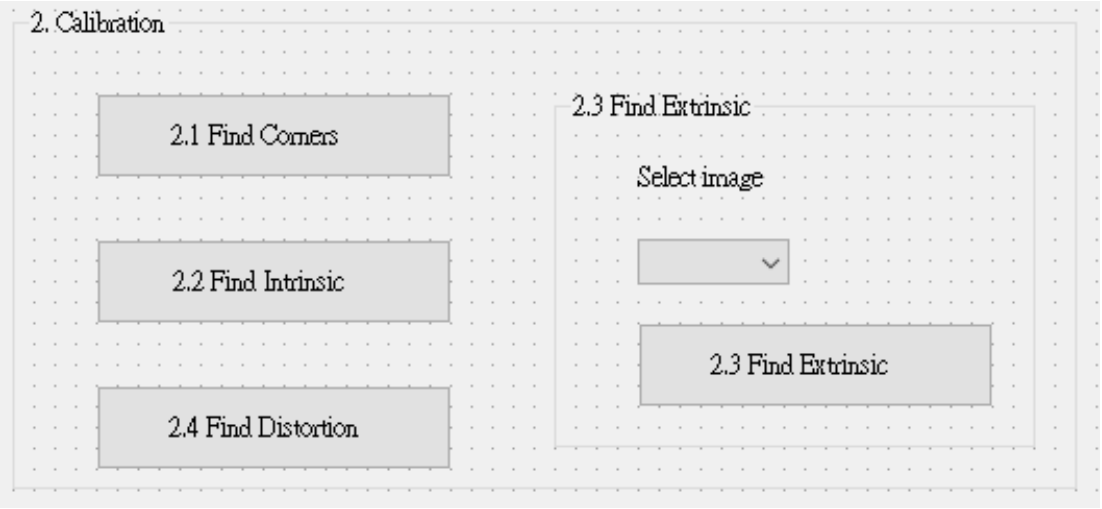


Show num.

2. (20%) Camera Calibration

(出題： Max)

2.1 (5%) Corner detection
2.2 (5%) Find the intrinsic matrix
2.3 (5%) Find the extrinsic matrix
2.4 (5%) Find the distortion matrix

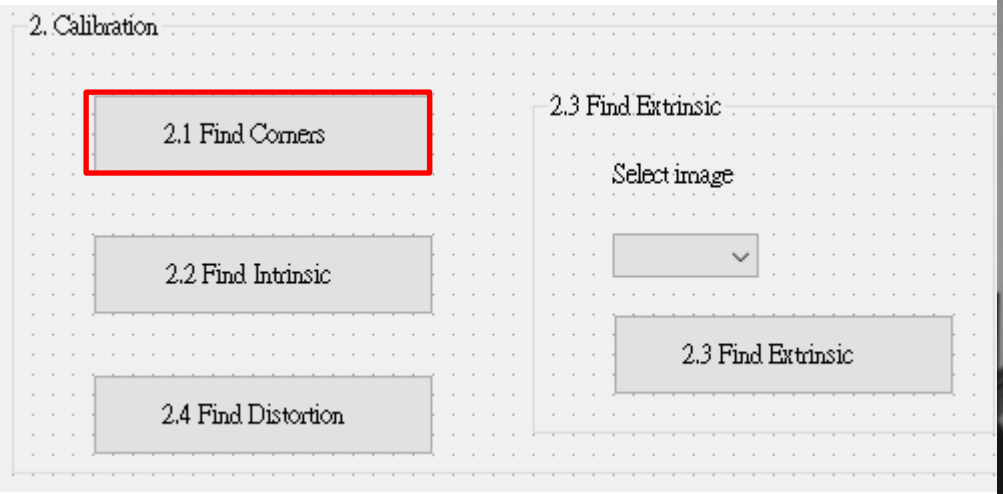


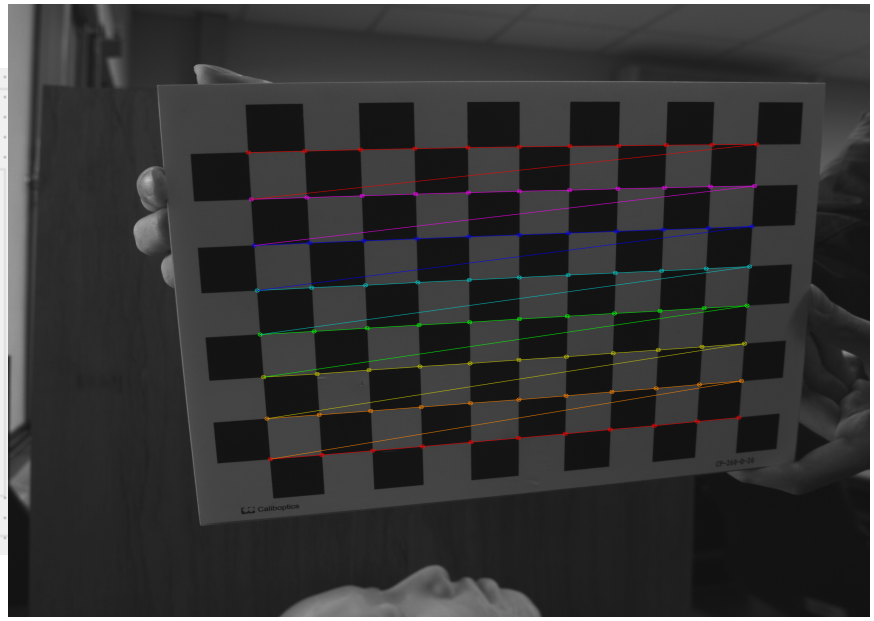
2.1 (5%) Corner Detection

- Given: 15 images, 1.bmp ~ 15.bmp
- Q: 1) Find and draw the corners on the chessboard for each image.
  2) Click button “1.1” to show the result.
- Hint :

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

  cvShowImage(…);
- Ex:





2.2 (5%) Find the Intrinsic Matrix

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

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

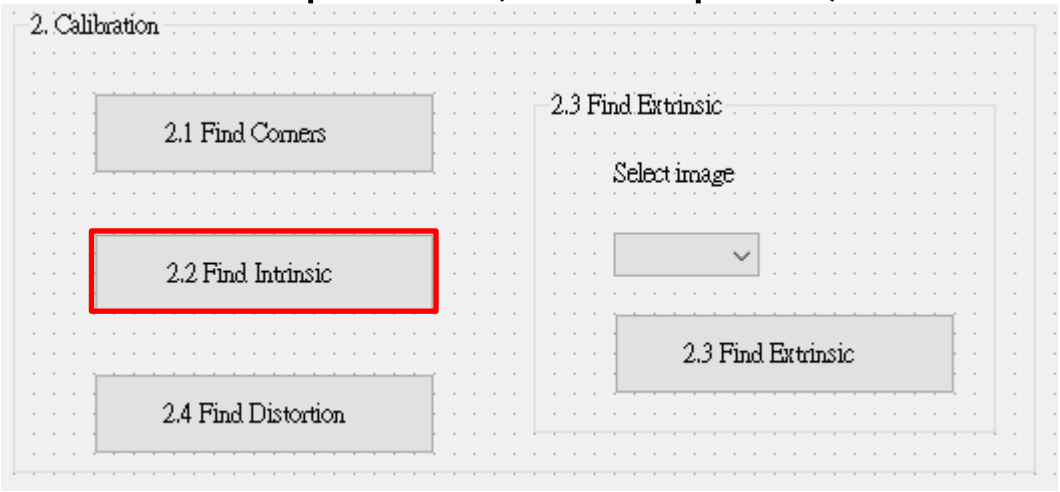
2) Click button “1.2” and then show the result on the console window.

- Output format:

```
[2227.333008, 0.000000, 384.186066;
0.000000, 2226.654541, 299.351746;
0.000000, 0.000000, 1.000000]
```

(Just an example)

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



2.3 (5%) Find the Extrinsic Matrix

- 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 “1.3” and then show the result on the console window.

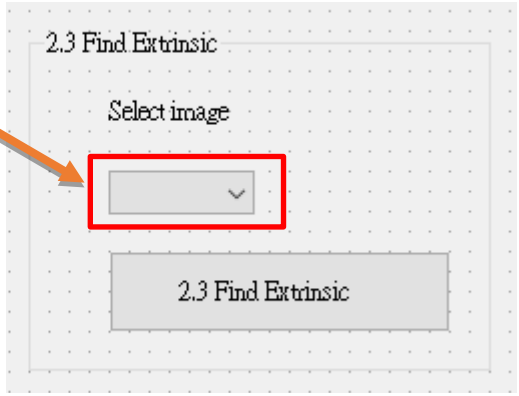
- Output format:

```
[-0.128827  ,0.991169  ,-0.031426  ,-1.969988 ;
0.983549  ,0.131755  ,0.123583  ,-1.105037 ;
0.126632  ,-0.014988  ,-0.991836  ,49.121323 ;  ]
```

(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



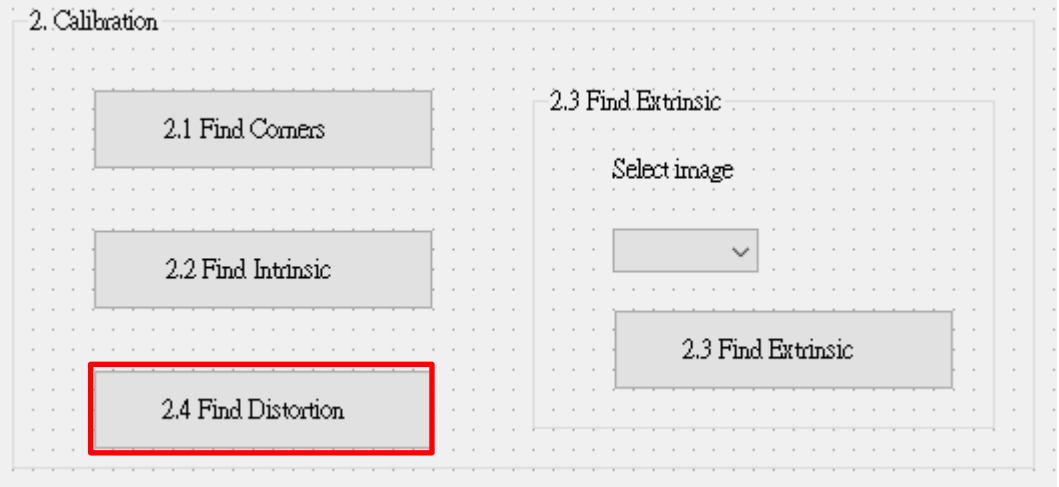
2.4 (5%) Find the Distortion Matrix

- Given: 15 images
- Q: 1) Find the distortion matrix: $[k_1, k_2, p_1, p_2, k_3]$
  2) Click button “1.4” to show the result on the console window.
- Output format:

```
[-0.072230, -0.261944, -0.000024, -0.003354, 4.228090]
```

(Just an example)

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



3. (20%) Augmented Reality

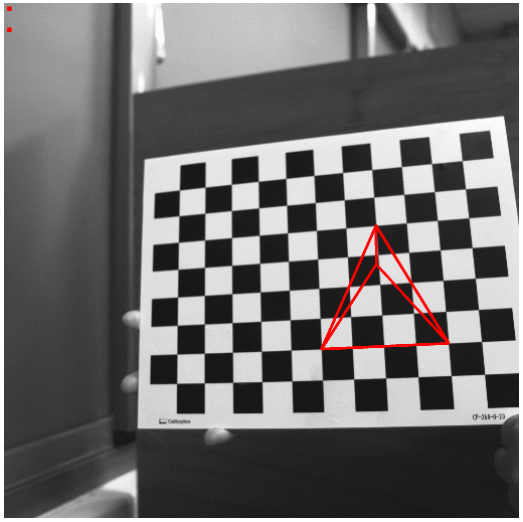
(出題： Oran)

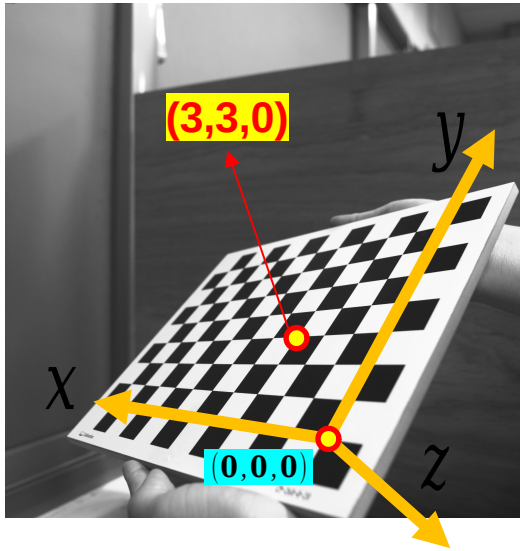
- Given: 5 images: 1~5.bmp
- Q:
  - 1) Calibrate 5 images to get intrinsic, distortion and extrinsic parameters
  - 2) Draw a “pyramid” on the chessboards images(1.bmp to 5.bmp)
  - 3) Click the button to show the tetrahedron on the picture. Show each picture 0.5 seconds (total 5 images)

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

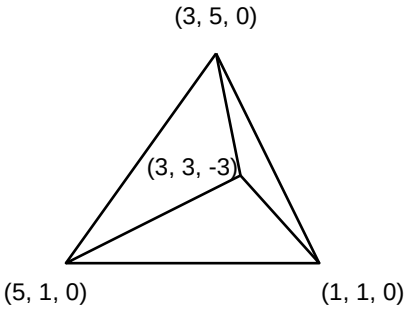
```
cv::calibrateCamera()
cv::projectPoints()
```

Demo





- 3D Object coordinates:
  Vertex (3, 3, -3)
  Corners(1, 1, 0)(3, 5, 0)(5, 1, 0)



4. (20%) Stereo Disparity Map

(出題： Oran)

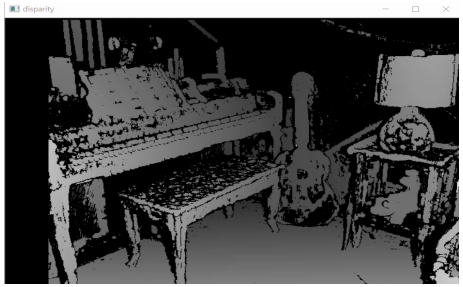
- Given: a pair of images, imL.png and imR.png (have been rectified)
- Q:
  1) (10%) Click button “4.1 Stereo Disparity Map” and compute the disparity map/image based on Left and Right stereo images
  2) (10%) Select a point on disparity map from 4.1, calculate the depth and show both the disparity value and the depth on the image. (Note: user can repeat 2) (EX: Result Video)
- Hint: Textbook Chapter 12 (P.451): StereoBM::create(numDisparities, blockSize) or StereoSGBM::create()
  1) Block 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.

Camera information: Baseline=178 mm, focal length=2826 pixels, =123 pixels
OpenCV Textbook Chapter 11 (P.372-373) & OpenCV Textbook Chapter 12 (P.436)

$$Z = \frac{f * B}{d}$$

imgL.png
Left Image (Reference Image)

imgR.png
Right Image



Result Video

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

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 by yourself.

2) Objective:

- You need to use python to write the ResNet network and complete the questions on the next few pages.

3) Environment Requirement

(1) Python 3.6
(2) Tensorflow 1.14
(3) Opencv-contrib-python 4.1.1 (for image show and write)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. *ResNet's Network Architecture*

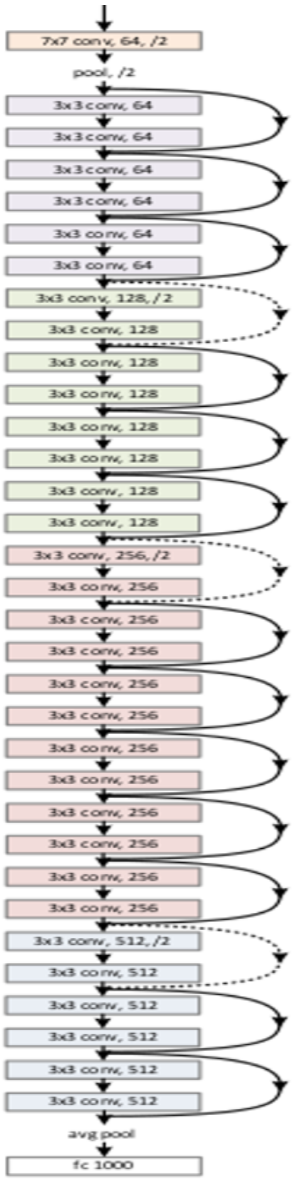


Fig. *ResNet50's Schematic Diagram*

R. Reference

R1) Deep Residual Learning for Image Recognition

R2) https://www.microsoft.com/en-us/download/details.aspx?id=54765 (ASIRRA)

5.1 Training by your computer at least **5 epochs** and show(or upload) your source code of ResNet50. (**5%**)

```
train loss = 0.67, train accuracy = 62.50%
train loss = 0.67, train accuracy = 62.50%
train loss = 0.70, train accuracy = 50.00%
train loss = 0.62, train accuracy = 87.50%
```

Fig. *Accuracy display during training*

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

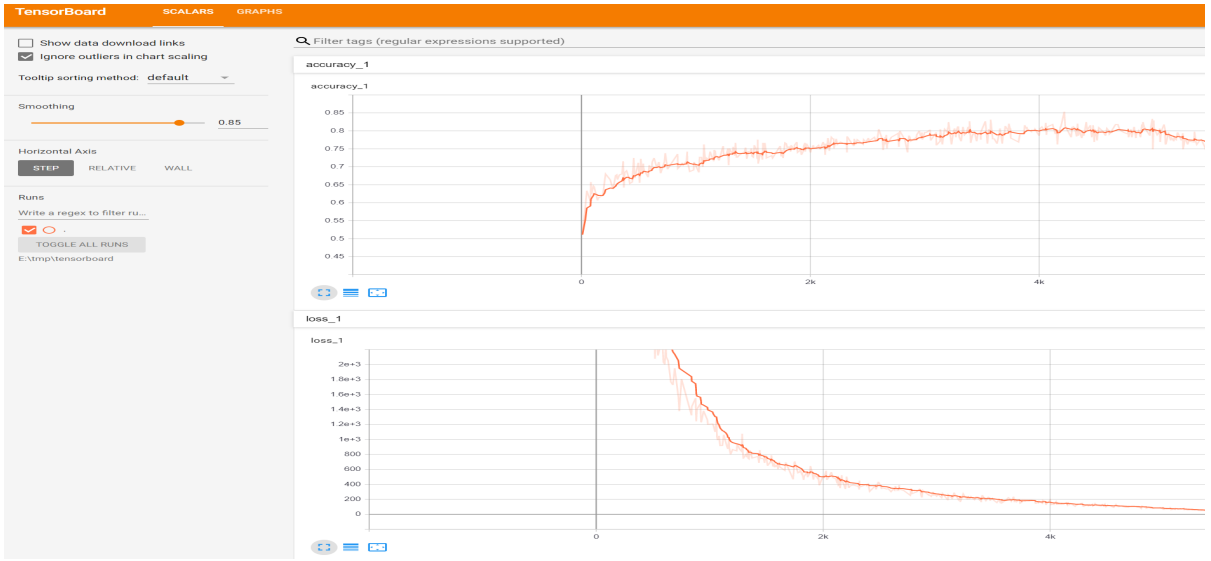


Fig. *Example of training with TensorBoard*

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

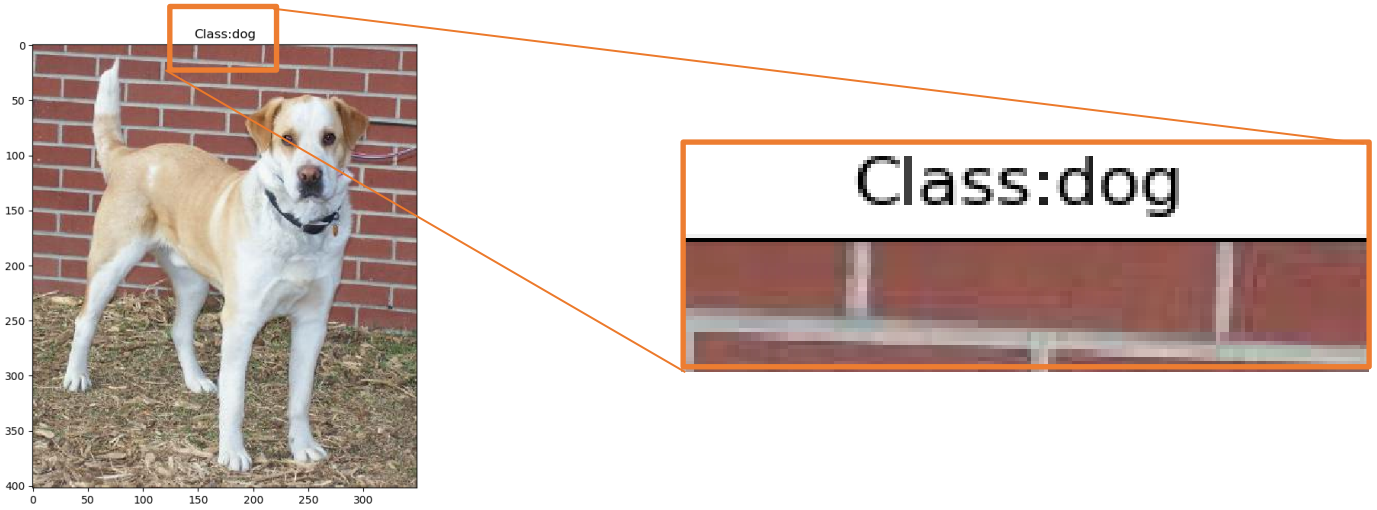


Fig. *Classification display example*

5.4 Try to resize the pictures in the dataset to expand the amount of training data, **write the code**(**3%**) and **draw the comparison table of accuracy**, save it as a picture.(**2%**)

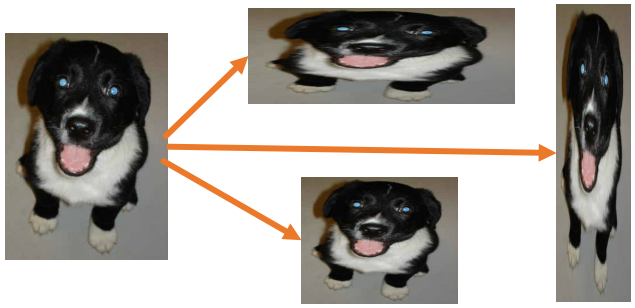


Fig. *Example of resize operation*

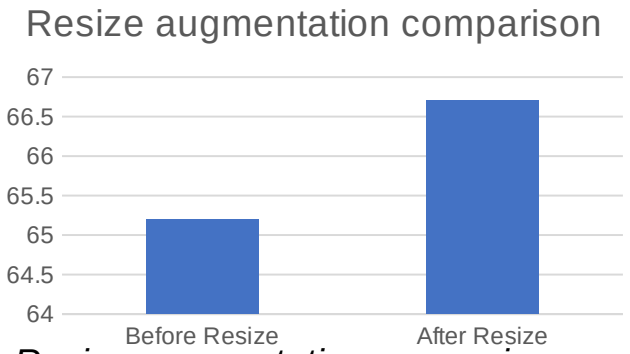


Fig. *Resize augmentation comparison example*