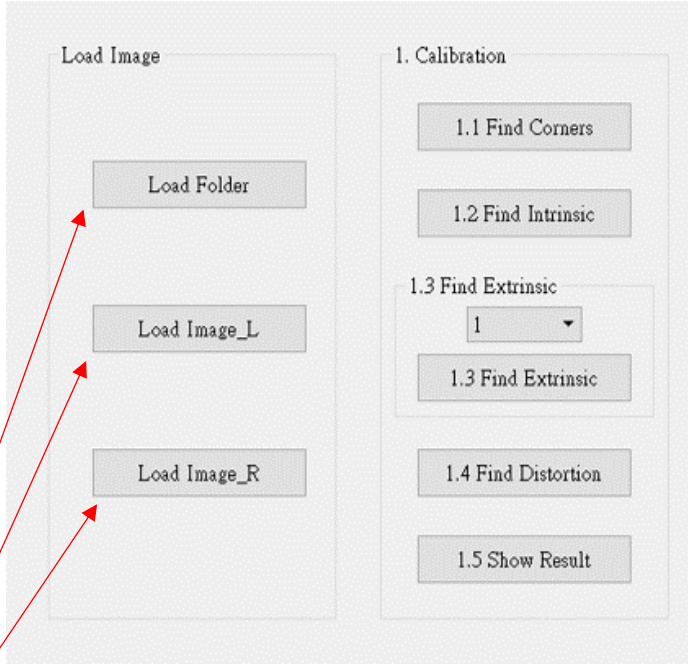
Notice (2/2)

- Python (recommended):
  - **Python 3.8**
  - **Opencv-contrib-python (4.10.0)**
  - **UI framework: pyqt5 (5.15.11)**

Assignment scoring (Total: 100%)

1. Camera Calibration (出題：Kerwin)
   - 1.1 Corner detection
   - 1.2 Find the intrinsic matrix
   - 1.3 Find the extrinsic matrix
   - 1.4 Find the distortion matrix
   - 1.5 Show the undistorted result
2. Augmented Reality (出題：Yiyu)
   - 2.1 Show words on board
   - 2.2 Show words vertically
3. Stereo Disparity Map (出題： Tien)
   - 3.1 Stereo Disparity Map
4. SIFT
   - 4.1 Keypoints (出題： Ian)
   - 4.2 Matched Keypoints
5. ~~Image Transformation~~ (出題： Yiyu)
   - ~~5.1 Rotation, scaling, translation~~
   - ~~5.2 Perspective transform~~

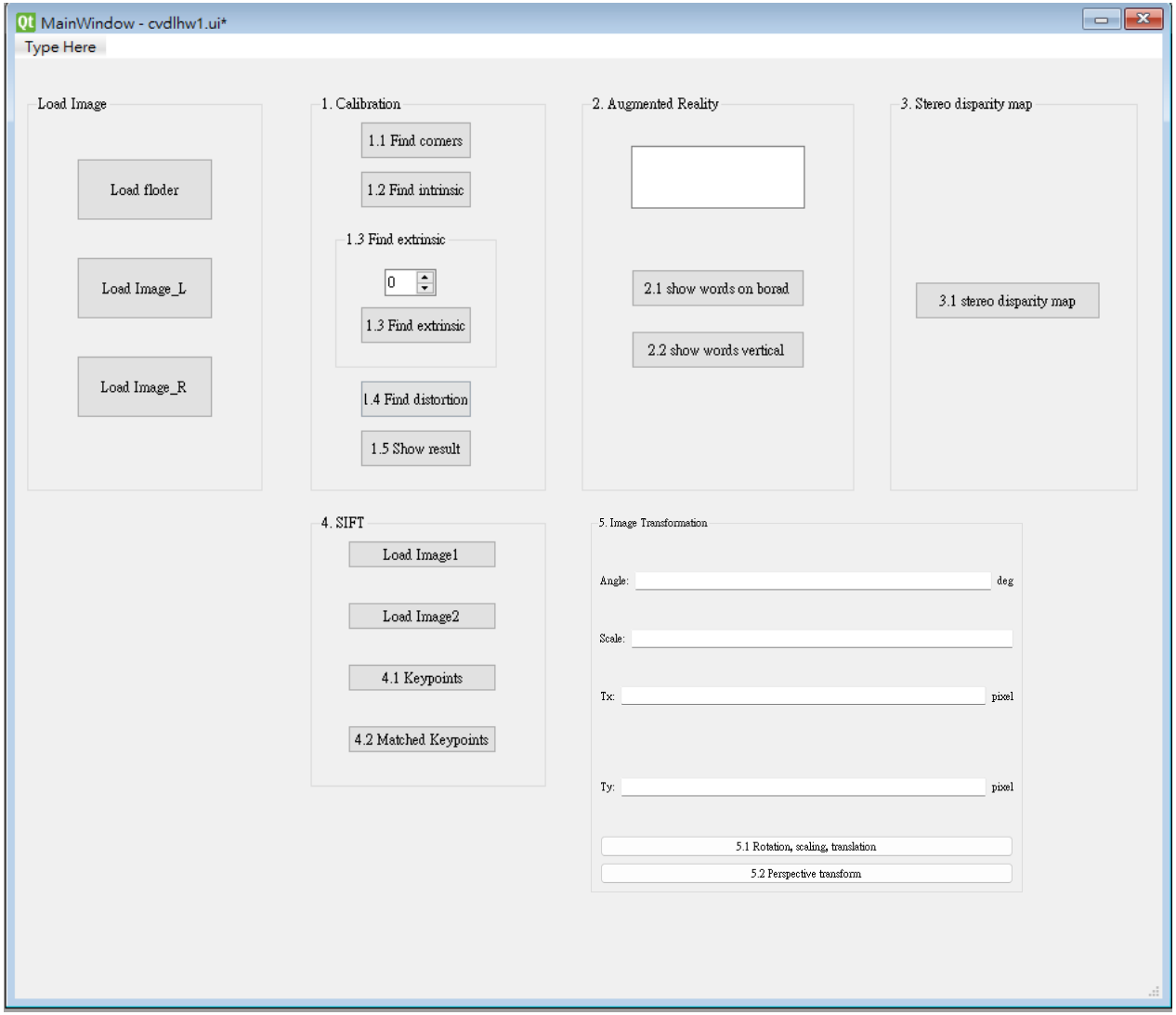


* Don’t fix your image path
  (There is another dataset for demonstration)

Load image please use the following function to read the path.
QFileDialog.getOpenFileName

Assignment scoring (Total: 100%)

- Use one UI to present 4 questions.



1. Camera Calibration

(出題： Kerwin)

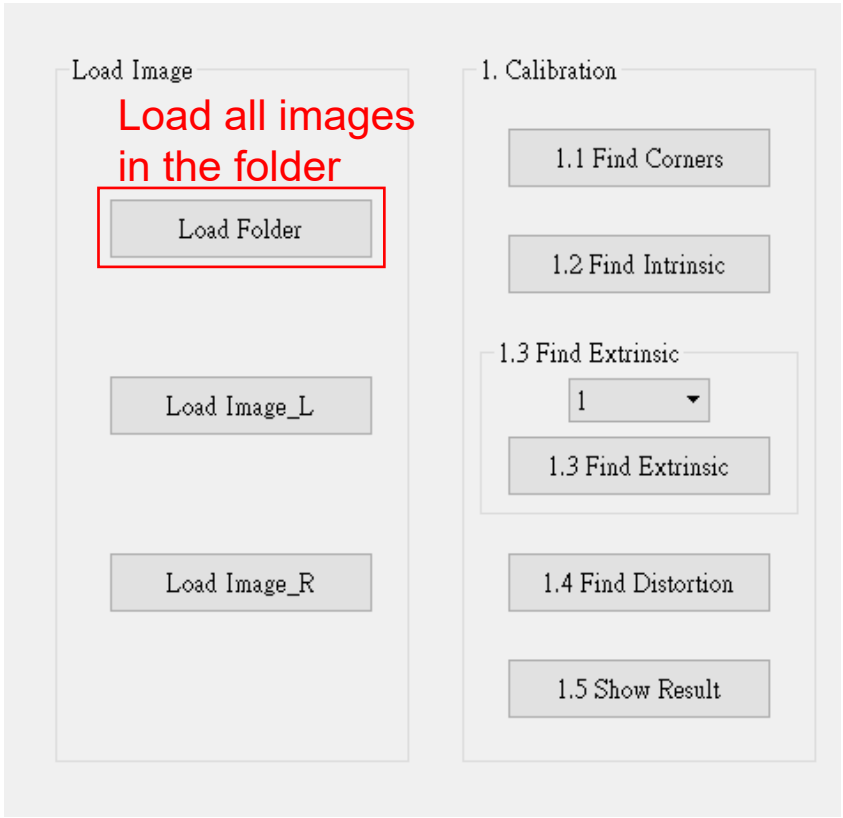
1.1 Corner detection

1.2 Find the intrinsic matrix K

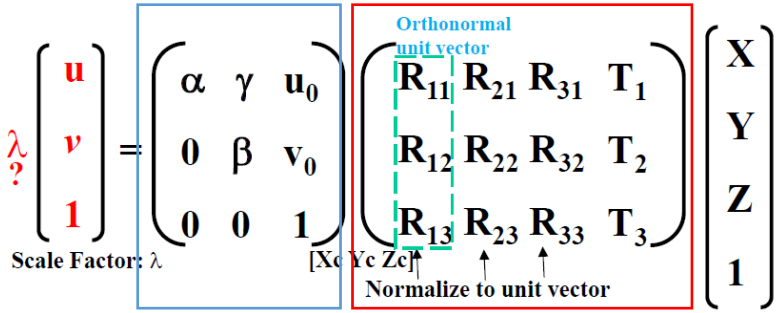
1.3 Find the extrinsic matrix $[R, T]$

1.4 Find the distortion matrix D

1.5 Show the undistorted result



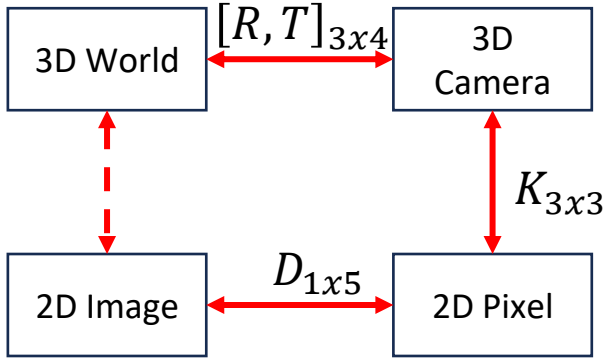
$[R, T]_{3x4}$:
Extrinsic Matrix



K_{3x3}: Intrinsic Matrix

$$D_{1x5} = [k_1, k_2, p_1, p_2, k_3]$$

D_{1x5}: Distortion Matrix



1.1 Corner Detection

(出題： Kerwin)

grayimg: input image. (2048x2048)
(width, height): width and high of corner in chessboard (11x8)
corners: corners points (1x2) on chessboard in image coord.

- Given: 15 images, 1.bmp ~ 15.bmp
- Q1:

1) Find and draw the corners on the chessboard for each image.

ret, corners (O/P) = cv2.findChessboardCorners(grayimg (I/P), (width, height) (I/P)) ➔ in order to detect the corner of chessboard.

corners (O/P) = cv2.cornerSubPix(grayimg (I/P), corners (I/P), winSize (I/P), zeroZone (I/P), criteria (I/P)) ➔ in order to increase accuracy.

winSize = (5, 5), the range of the search area near the corner point.
zeroZone = (-1, -1), window size prevent from focusing on edge of image, (-1, -1) means not to set a dead zone
criteria = (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS, 30, 0.001), termination optimization criteria which is OpenCV recommend.

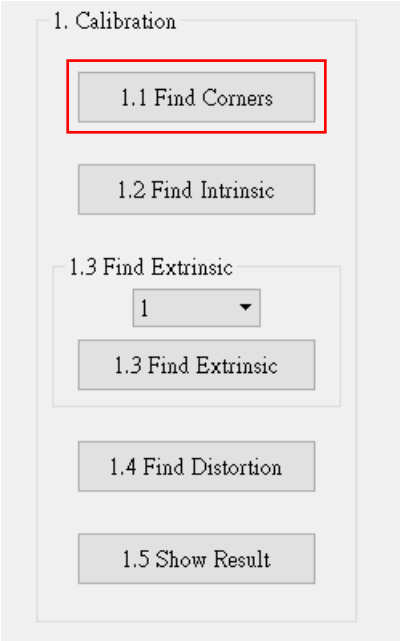
max iteration time

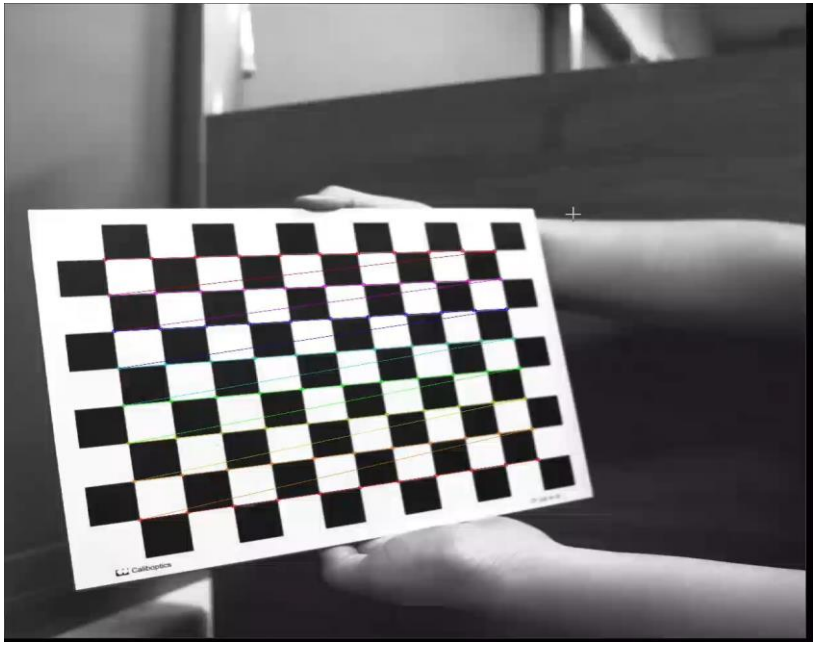
Precision of sliding window

2) Click button “1.1 Find Corners” to show each picture.

- Hint:

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





1.2 Find the Intrinsic Matrix

(出題： Kerwin)

- Given: 15 images, 1.bmp ~ 15.bmp
- Q2:

  1) Find the intrinsic matrix:

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

ins: intrinsic matrix(K: $3x3$)
dist: distortion matrix(D: $1x5$)
rvec: rotation vector(R: $1x3$)
tvec: translation vector(T: $1x3$)

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

ins, dist, rvec, tvec=cv2.calibrateCamera (3DObjectPoints, 2DImagePoints=corners, (w, h))

3x3 1x5 1x3 1x3

➔ in order to get R, T, K, D

3DobjectPoints: corners points of chessboard in 3D coordinate.(unit: 0.02m), (11x8x1)
(w,h): image size(2048, 2048)

Totoal points:
(0,0,0),…, (11,0,0)
⋮ ⋱ ⋮
(0,7,0),…, (11,7,0)

  2) Click button "1.2 Find Intrinsic" and then show the result on the console window.

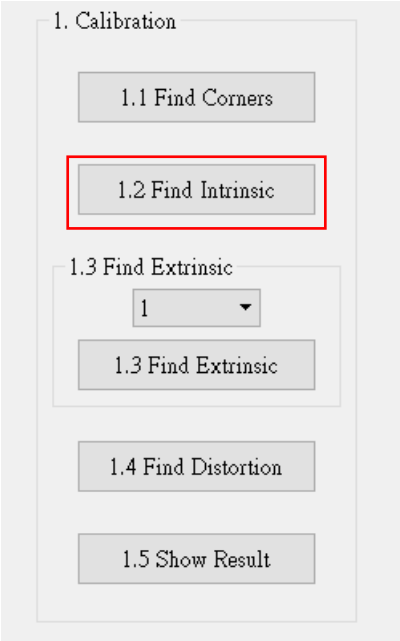
  Output format:

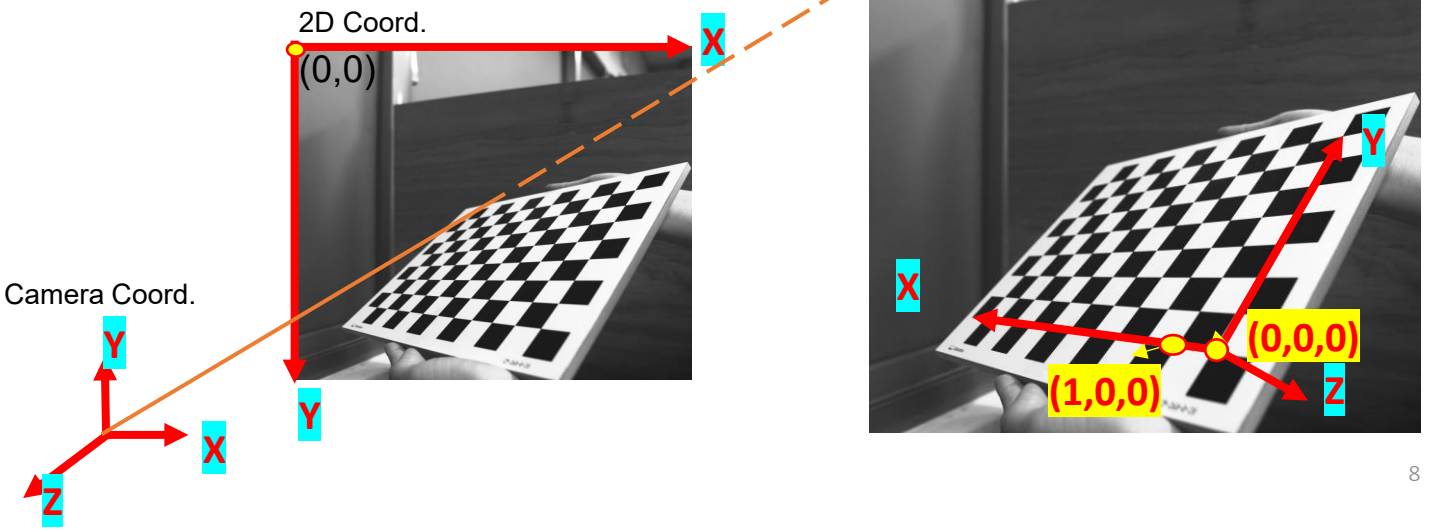
```
Intrinsic:
[[2.22370244e+03 0.00000000e+00 1.03021663e+03]
 [0.00000000e+00 2.22296836e+03 1.03752624e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
```

(Just an example)

- Hint:

  OpenCV Textbook Chapter 11 (p.398 ~ p.400)





1.3 Find the Extrinsic Matrix

(出題： Kerwin)

extrinsic_matrix

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & T_X \\ r_{21} & r_{22} & r_{23} & T_Y \\ r_{31} & r_{32} & r_{33} & T_Z \end{bmatrix}$$

- Given: Intrinsic parameters, distortion coefficients, and the list of 15 images
- Q3:
  1) Find the extrinsic matrix of the chessboard for each of the 15 images, respectively:

     You can get rvec, tvec from (1.2) in cv2.calibrateCamera.

     rotation_matrix (O/P) = cv2.Rodrigues(rvec (I/P))[0] ➔ Rodrigues transformation: transform rotation vector into rotation matrix(3x3).

     extrinsic_matrix (O/P) = np.hstack(rotation_matrix (I/P) , tvec (I/P)) ➔ Merge R and T in order to get extrinsic matrix(3x4).

  2) Click button "1.3 Find Extrinsic" and then show the result on the console window.

     Output format:

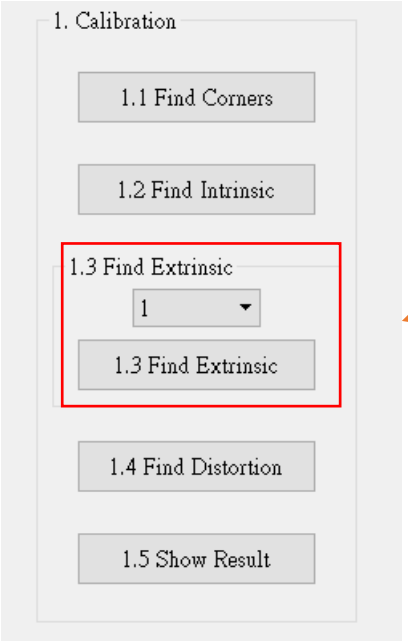
```
Extrinsic:
[[-0.8767247  -0.23001438  0.4224301   4.39838495]
 [ 0.19727469 -0.97293475 -0.12033563  0.68022105]
 [ 0.43867585 -0.02216645  0.89837194 16.22126   ]]
```

(Just an example)

- Hint:

  Extrinsic matrix can be obtained simultaneously with intrinsic.

  OpenCV Textbook Chapter 11, (p.370 ~ p.402)



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

rvec, tvec get from (1.2) in cv2.calibrateCamera

1.4 Find the Distortion Matrix

(出題： Kerwin)

- Given: 15 images
- Q4:

  1) Find the distortion matrix: $[k_1, k_2, p_1, p_2, k_3]$

  You can get distortion matrix from (1.2) in cv2.calibrateCamera.

  2) Click button "1.4 Find Distortion" to show the result on the console window.

  Output format:

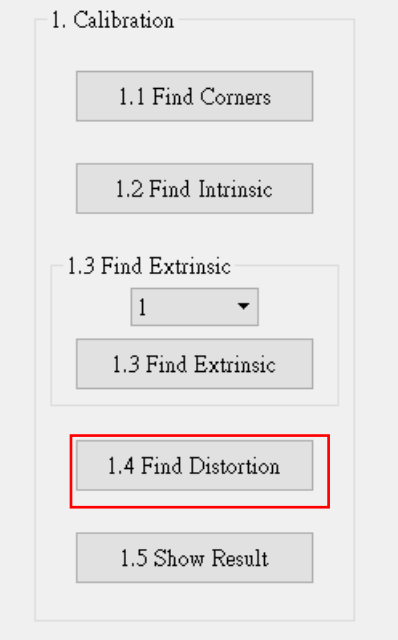
```
Distortion:
[[-0.11868112  0.02776881 -0.00092036  0.00047227  0.11793646]]
```

- Hint:

  (Just an example)

  Distortion coefficients can be obtained simultaneously with intrinsic.

  OpenCV Textbook Chapter 11 (p.398 ~ p.400)



1.5 Show the Undistorted Result

(出題： Kerwin)

- Given: 15 images
- Q5:

  1) Undistort the chessboard images.

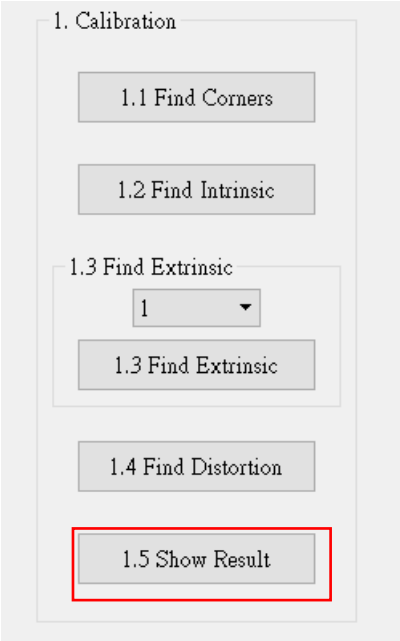
  You can get intrinsic matrix and distortion matrix from (1.2) in cv2.calibrateCamera.

  result_img (Q/P) = cv2.undistort(grayimg (I/P), ins (I/P), dist (I/P)) ➔ Undistort the image by intrinsic matrix and distortion matrix

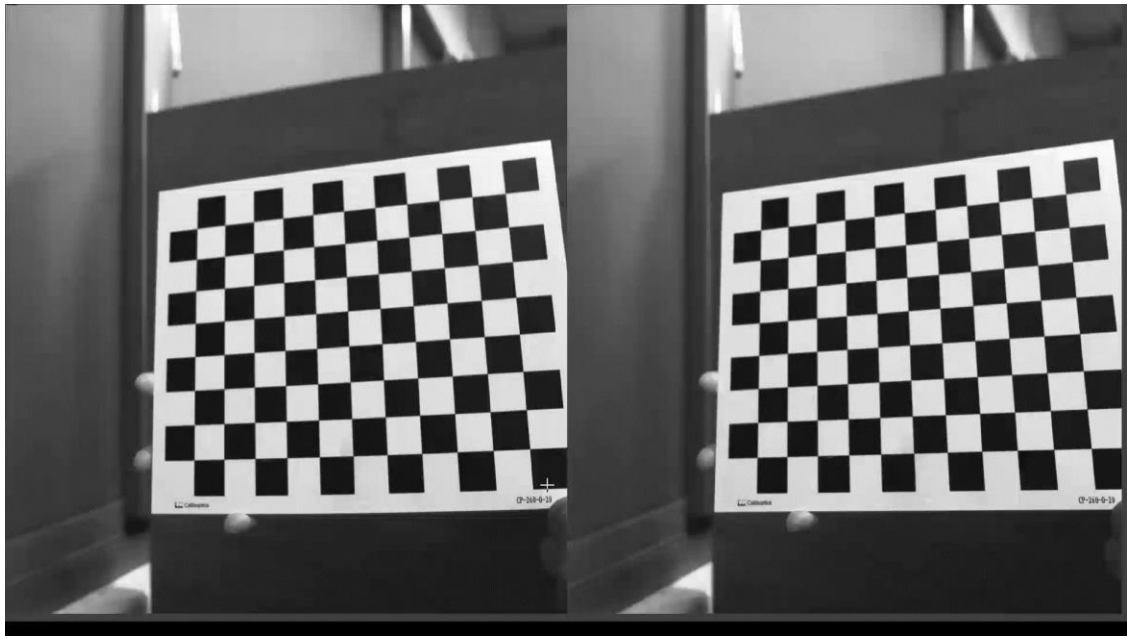
  2) Click button "1.5 Show Result" to show distorted and undistorted images

- Hint:

  OpenCV Textbook Chapter 11 (p.398 ~ p.400)

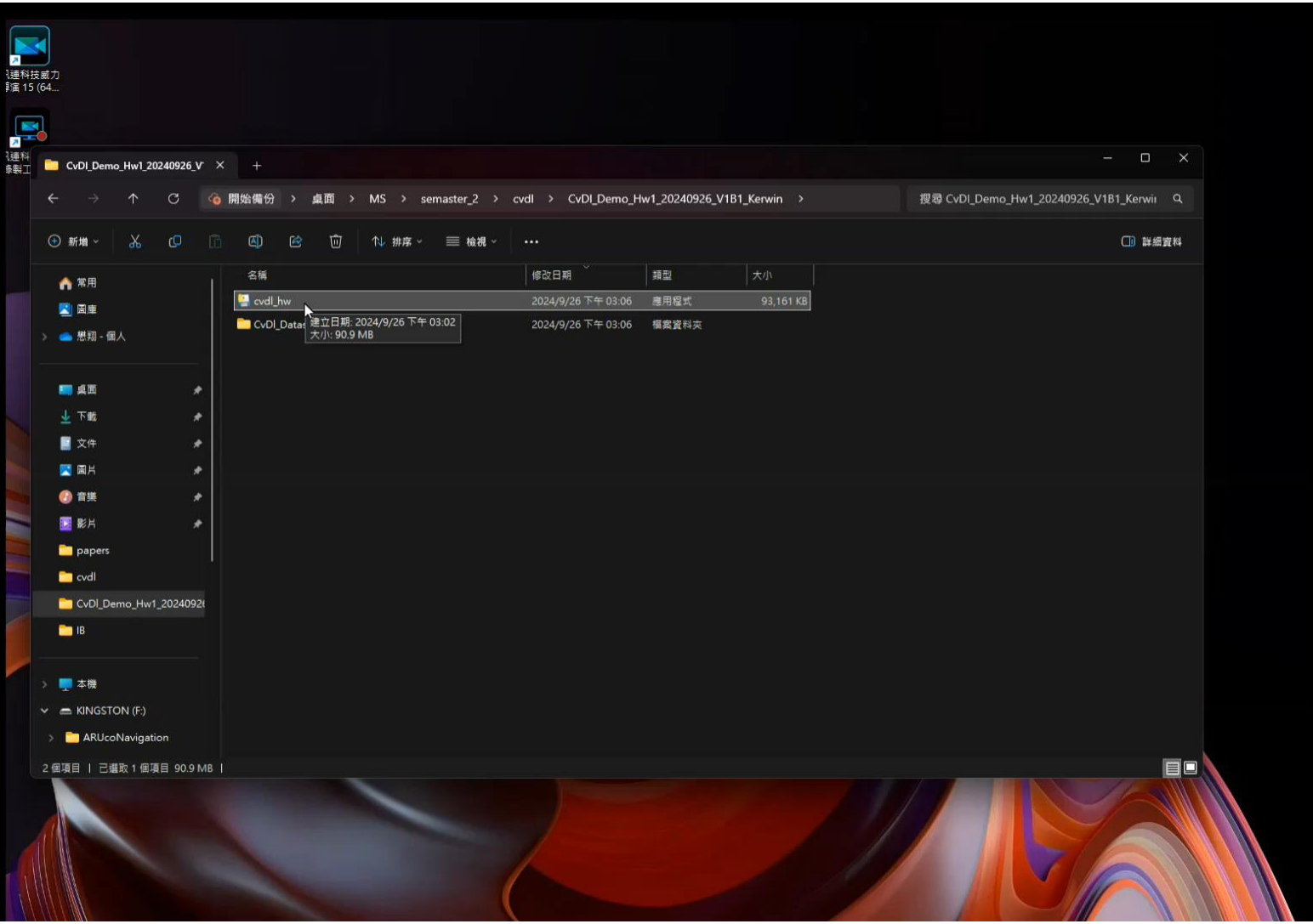


Distorted image | Undistorted image



1.6 Camera Calibration – Demo Video

(出題： Kerwin)

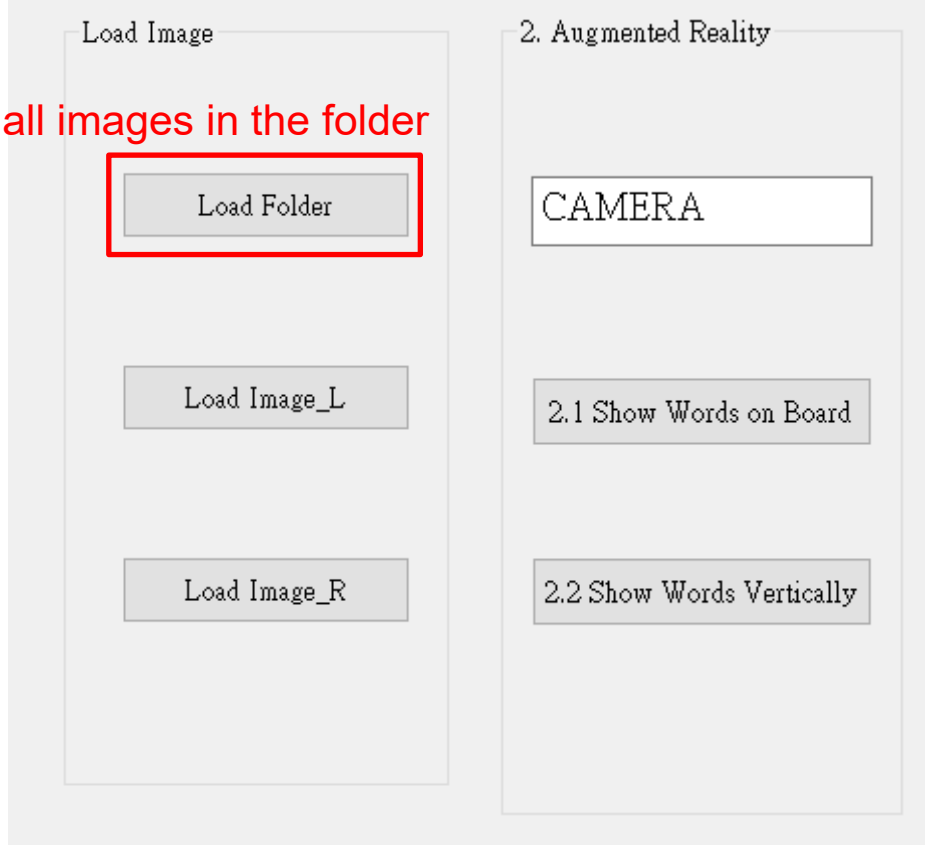


2. Augmented Reality

(出題： Yiyu)

2.1 Show words on board

2.2 Show words vertically



2. Augmented Reality

(出題： Yiyu)

➢ Guides and Requirements:

1) How to use the database: (alphabet_db_onboard.txt, alphabet_db_vertical.txt)

– Inside the database:
(1) It contains the 3D world coordinates of letter A to Z
(2) Each letter represents an object

– Use OpenCV function to read and derive the array or matrix of the char
Here take 'K' in 'alphabet_db_onboard.txt' for example
e.g. (Python):

O/P: created file reader I/P: file name I/P: specify mode

fs = cv2.FileStorage('alphabet_db_onboard.txt', cv2.FILE_STORAGE_READ) ➔ read data from database

charPoints = fs.getNode('K').mat() ➔ convert it into to a matrix; Node K: Six 3D points as below

O/P: Object coordinates (3x2x3) I/P: specify the letter

```
charPoints = [ [[2, 2, 0], [2, 0, 0]],
               [[0, 2, 0], [2, 1, 0]],
               [[2, 1, 0], [0, 0, 0]]] Unit: 0.02m (square size of the checkerboard)
```

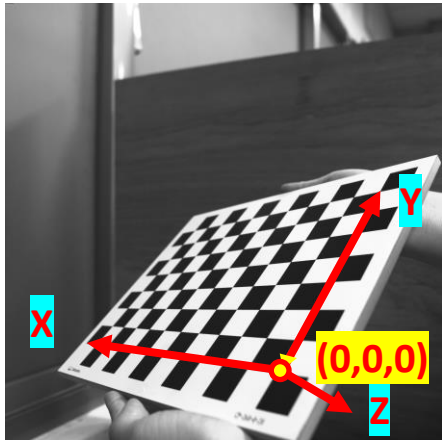
– Letter 'K' consist of 3 lines, so the 'charPoints' consists 3 pairs of 3D coordinates in World Coordinate representing two ends of the line shown in the upper right image.

2) Chessboard Coordinates

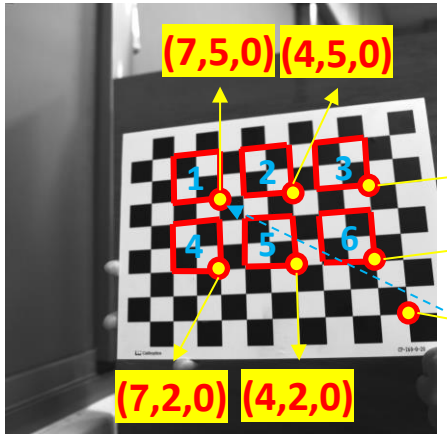
– The chessboard x, y, z axis and (0,0,0) origin coordinate are shown in the bottom left image

– Each character should be placed in the order and position shown in the bottom right image
- Apply translation to 3D object coordinates to move to the designated position (add value to coord.)

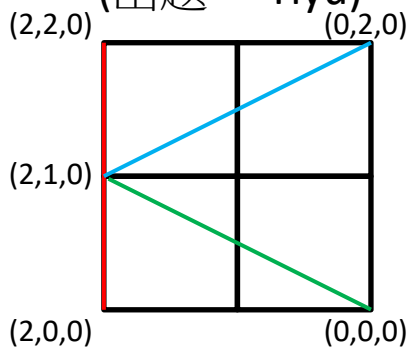
3D World Coordinate = 3D Chessboard Coordinate



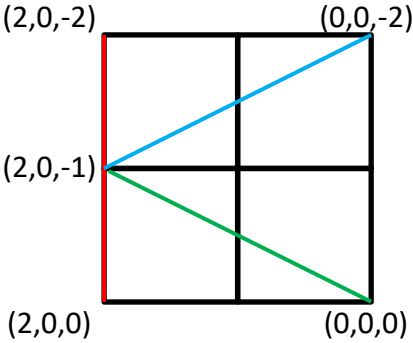
Chessboard Coordinate



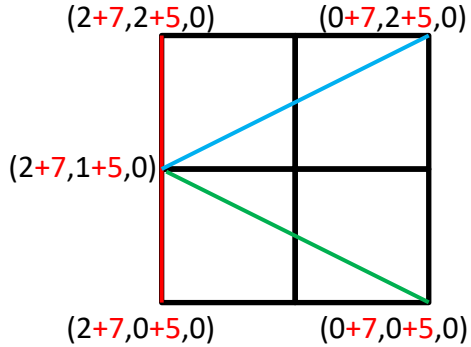
Position and Order



Object Points in Onboard.txt: The 3D world coordinates = 3D object coordinates (X, Y, Z=0)



Object Points in Vertical.txt: The 3D world coordinates = 3D object coordinates (X, Y=0, Z)



Example: Object points of Onboard.txt at position 1 (x-y plane of 3D world coordinates)

2.1 Show words on board

(出題： Yiyu)

ins: intrinsic matrix(K: $3x3$)
dist: distortion matrix(D: $1x5$)
rvec: rotation vector(R: $1x3$)
tvec: translation vector(T: $1x3$)
(w, h): image size

➢ Given: 5 images (1~5.bmp), alphabet_db_onboard.txt

Q1: **Show a Word (e.g. CAMERA) on chessboard**

1) Calibrate 5 images to get intrinsic, extrinsic, distortion, rotation vector, and translation vector parameters.

Link for camera calibration example

O/P O/P O/P O/P I/P: corner points of the chessboard in 3D world coordinate(11x8x1) for 5 images $I_{i=1\sim5}$ I/P

ins, dist, rvec, tvec=cv2.calibrateCamera (objectPoints, imagePoints=corners, (w, h))

3x3 1x5 (1x3) X5 images (1x3) X5 images

I/P: corner points of the chessboard in 2D image coordinate for 5 images

2) Input a word less than 6 character in English in textEdit box.

3) Derive the shape of the word by using the provided database (alphabet_db_onboard.txt) and project it on image.

O/P: created file reader I/P: file name I/P: specify mode

fs = cv2.FileStorage('alphabet_db_onboard.txt', cv2.FILE_STORAGE_READ)➔ read data from database

charPoints = fs.getNode('K').mat() ➔ convert it into a matrix

O/P: Object 3D world coordinates (3x2x3 for 'K') I/P: input the letter from UI text box

charPoints = [[[2+7, 2+5, 0], [2+7, 0+5, 0]],
[[0+7, 2+5, 0], [2+7, 1+5, 0]],
[[2+7, 1+5, 0], [0+7, 0+5, 0]]]

pointA pointB

newCharPoints = [[[949, 1054], [831, 1194]],
[[1083, 1026], [891, 1122]],
[[891, 1122], [962, 1176]]] Unit: 0.02m (square size of the checkerboard)

4) Project points on chessboard for each image

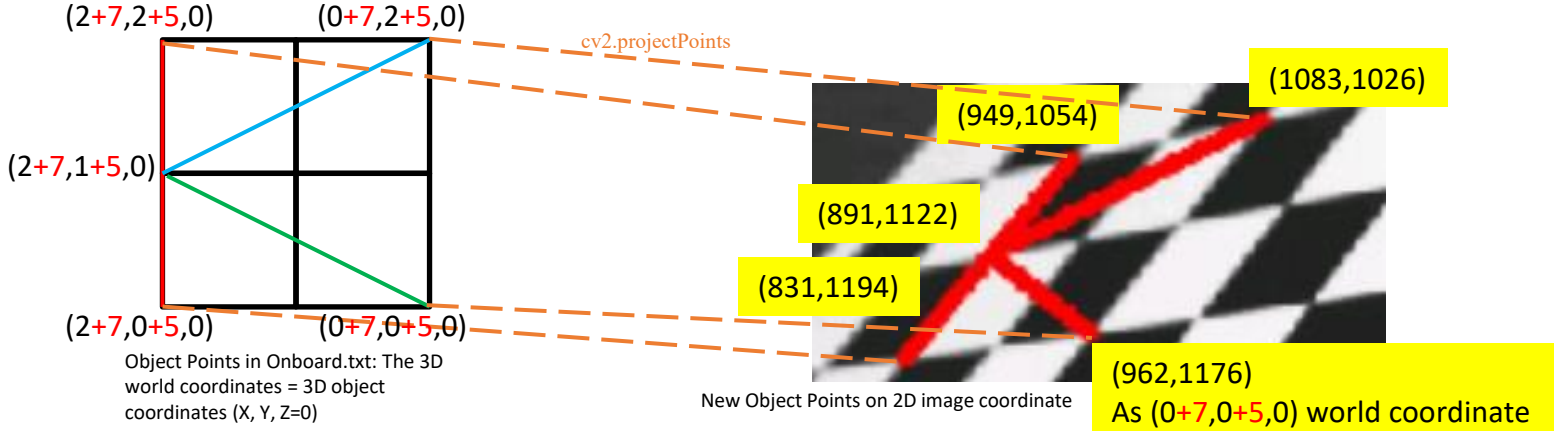
O/P: new object 2D coordinates I/P: object 3D world coordinates I/P I/P I/P I/P

newCharPoints = cv2.projectPoints(charPoints, ins, dist, rvec, tvec)

➔ Get transformed 2D corner points of each charater. Total 6 characters. The position of each character is in order over 6 positions.

5) Click button "2.1 Show Words on Board" to show the result of each image for 1 second (5 images in total).

cv2.line(image, pointA, pointB)➔ Draw a line on image from point A to B (points aquired from newCharPoints)



Object Points in Onboard.txt: The 3D world coordinates = 3D object coordinates (X, Y, Z=0)

New Object Points on 2D image coordinate

2.2 Show words vertically

(出題： Yiyu)

ins: intrinsic matrix(K: $3x3$)
dist: distortion matrix(D: $1x5$)
rvec: rotation vector(R: $1x3$)
tvec: translation vector(T: $1x3$)
(w, h): image size

➢ Given: 5 images (1~5.bmp), alphabet_db_onboard.txt

Q1: **Show a Word (e.g. CAMERA) vertically on chessboard**

1) Calibrate 5 images to get intrinsic, extrinsic, distortion, rotation vector, and translation vector parameters.

Link for camera calibration example

O/P O/P O/P O/P I/P: corner points of the chessboard in 3D world coordinate(11x8x1) for 5 images $I_{i=1\sim5}$ I/P

ins, dist, rvec, tvec=cv2.calibrateCamera (objectPoints, imagePoints=corners, (w, h))

3x3 1x5 (1x3) X5 images (1x3) X5 images

I/P: corner points of the chessboard in 2D image coordinate for 5 images

2) Input a word less than 6 character in English in textEdit box.

3) Derive the shape of the word by using the provided database (alphabet_db_vertical.txt) and project it on image.

O/P: created file reader I/P: file name I/P: specify mode

fs = cv2.FileStorage('alphabet_db_vertical.txt', cv2.FILE_STORAGE_READ)➔ read data from database

charPoints = fs.getNode('K').mat() ➔ convert it into a matrix

O/P: Object 3D world coordinates (3x2x3 for 'K') I/P: input the letter from UI text box

```
charPoints = [[[2+7, 0+5, -2], [2+7, 0+5, 0]],
              [[0+7, 0+5, -2], [2+7, 0+5, -1]],
              [[2+7, 0+5, -1], [0+7, 0+5, 0]] ]
                 pointA          pointB
newCharPoints = [[[949, 1054], [831, 1194]],
                 [[1083, 1026], [891, 1122]],
                 [[891, 1122], [962, 1176]] ] Unit: 0.02m (square size of the checkerboard)
```

4) Project points on chessboard for each image

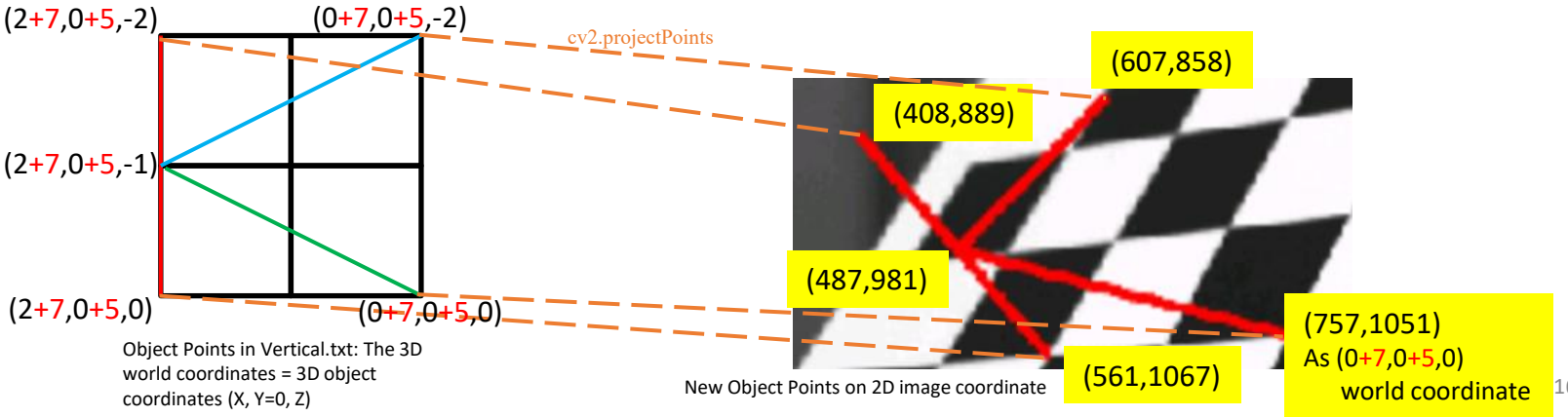
O/P: new object 2D coordinates I/P: object 3D world coordinates I/P I/P I/P I/P

newCharPoints = cv2.projectPoints(charPoints, ins, dist, rvec, tvec)

➔ Get transformed 2D corner points of each charater. Total 6 characters. The position of each character is in order over 6 positions.

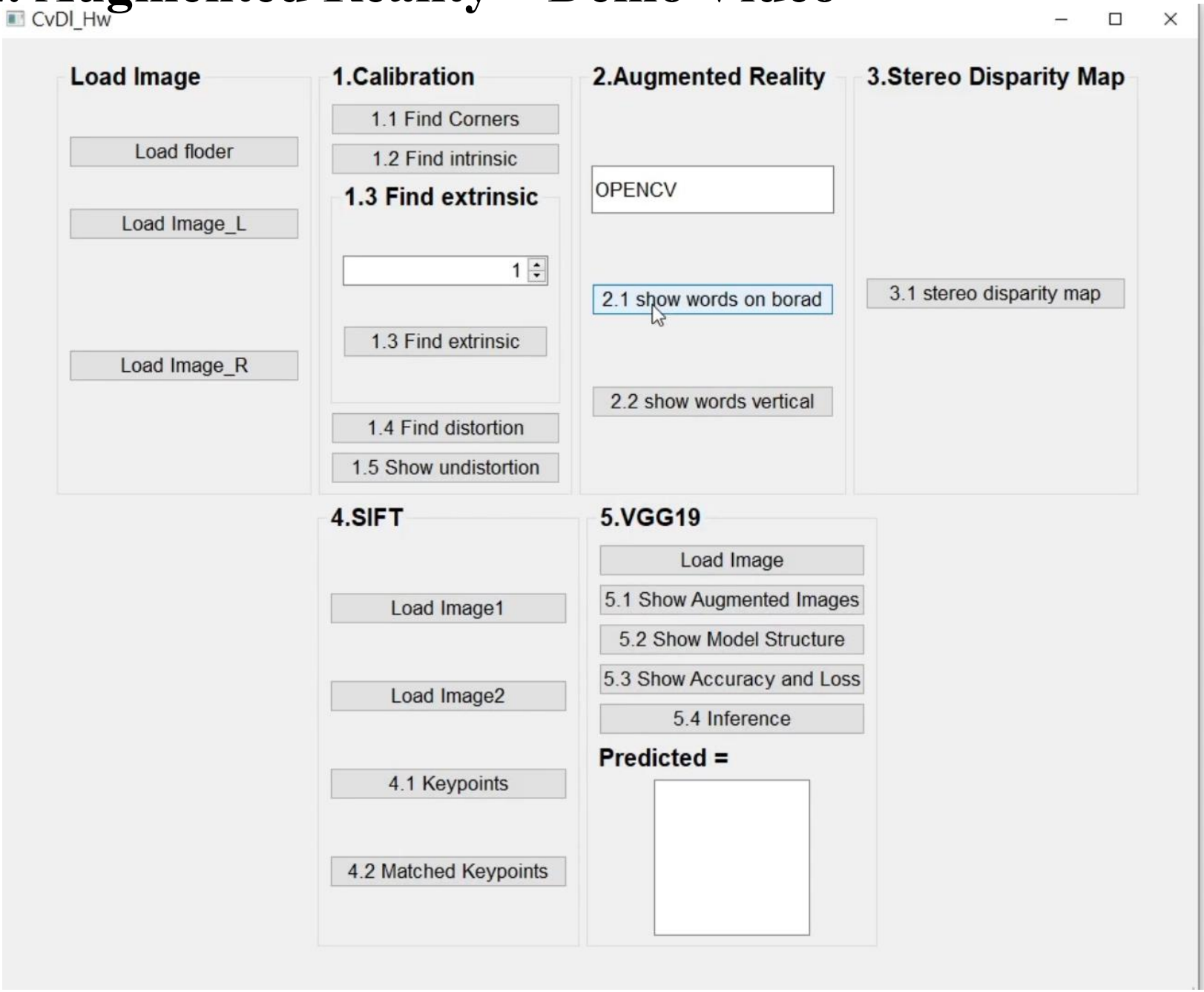
5) Click button "2.1 Show Words vertically" to show the result of each image for 1 second (5 images in total).

cv2.line(image, pointA, pointB)➔ Draw a line on image from point A to B (points aquired from newCharPoints)



2. Augmented Reality – Demo Video

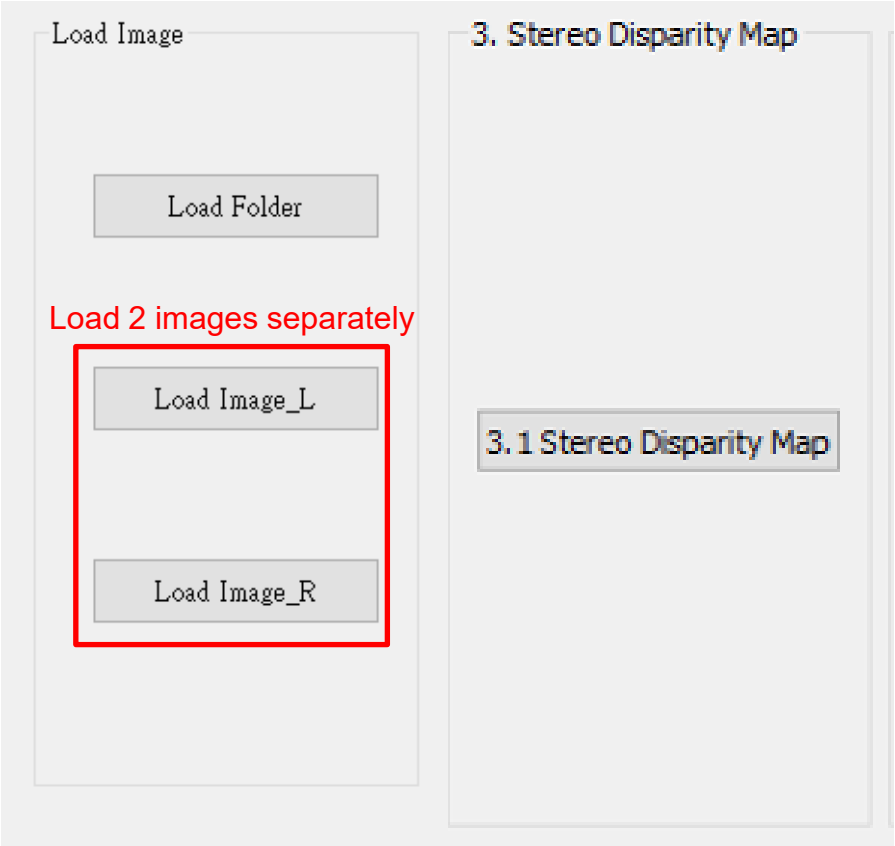
(出題： Yiyu)



3. Stereo Disparity Map

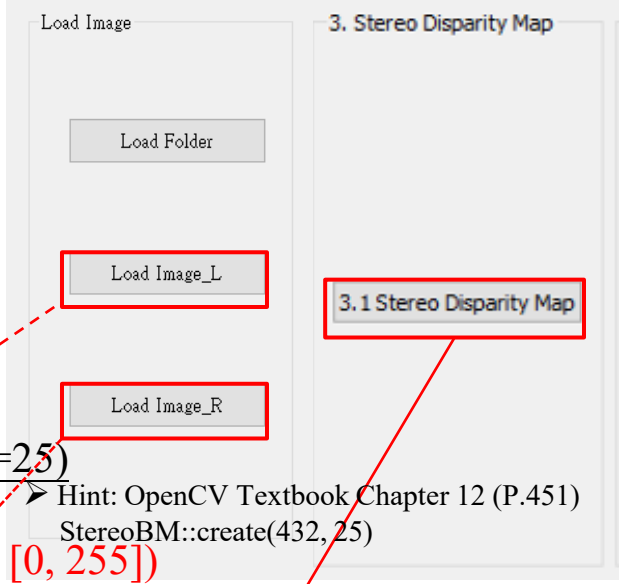
(出題：Tien)

3.1 Stereo Disparity Map



3.1 Stereo Disparity Map

(出題：Tien)



1. 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**
   1) Load imgL.png (click "Load Image_L"). (Input)
   2) Load imgR.png (click "Load Image_R"). (Input)
   3) Click button "3.1 Stereo Disparity Map" (Output) to show result.

Hint:

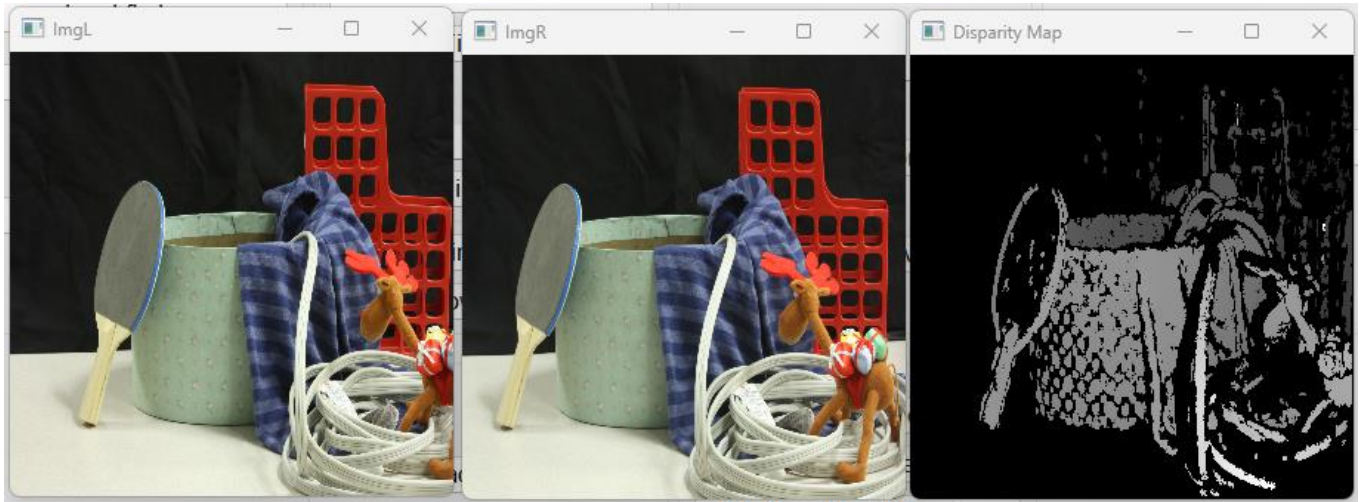
Usually max disparity is less than 64 pixels

(BM: Block Matching)

1. Use OpenCV StereoBM class to build StereoBM objects.
2. class stereo = cv2.StereoBM.create(**numDisparities**=432, **blockSize**=25)
3. disparity (Output) = stereo.compute(**imgL**,**imgR**) (Input)
4. Show the Disparity Map (Normalize [Min, Max] of disparity value to [0, 255])
   - The above parameters can be freely changed according to the following rules.
     - **numDisparities (int)**: The disparity search range must be positive and divisible by 16.
     - **blockSize (int)**: The window (block) size compared by the algorithm, must be odd and within the range [3, 51]. BlockSize = Window size
       - Larger block size implies smoother but less accurate disparity map.
       - Smaller block size gives finer disparity details, yet increase the likelihood of algorithmic misalignment.

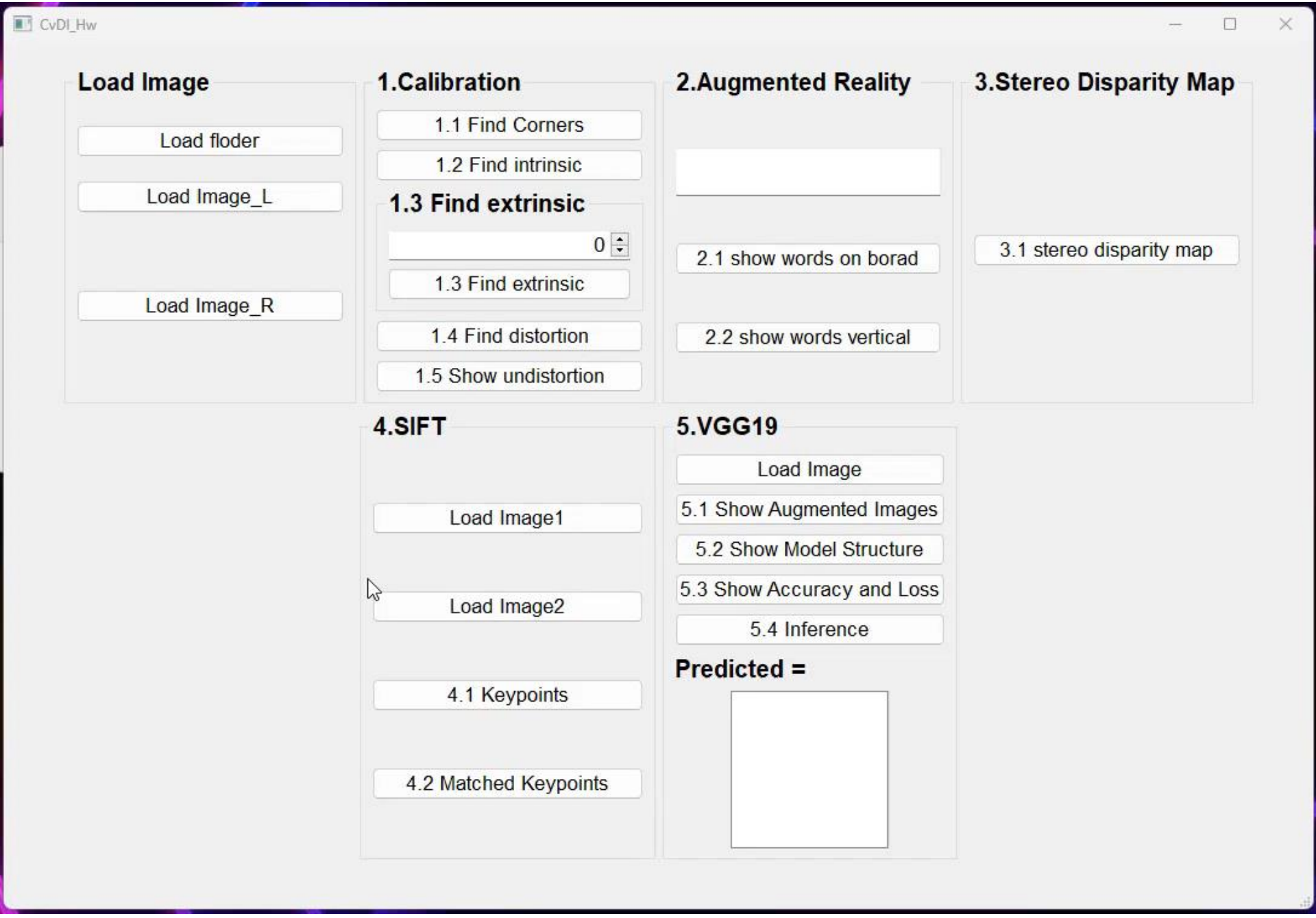
➢ Hint: OpenCV Textbook Chapter 12 (P.451)
StereoBM::create(432, 25)

Both input image are W: 2816 pixel, H: 1916 pixel



3. Stereo Disparity Map - Demo Video

(出題：Tien)

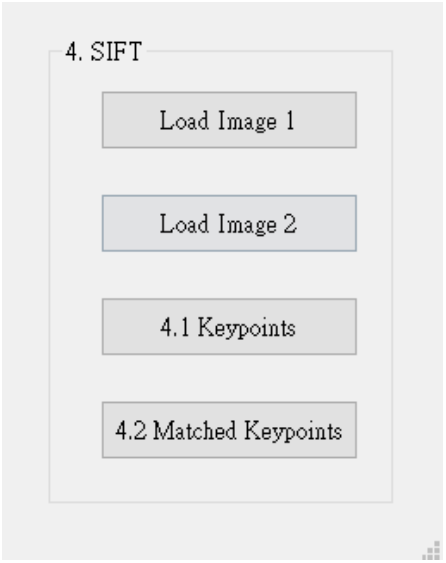


4. SIFT

(出題：Ian)

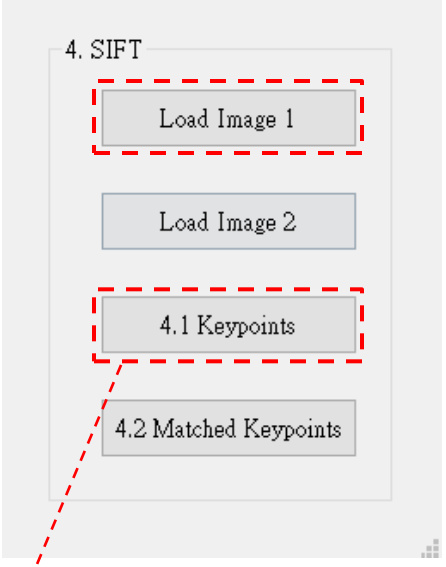
4.1 SIFT Keypoints

4.2 Matched Keypoints



4.1 SIFT Keypoints

(出題： Ian)



1. - Click button “Load Image 1” to load Left.jpg.

   - Click button “4.1 Keypoints” to show:

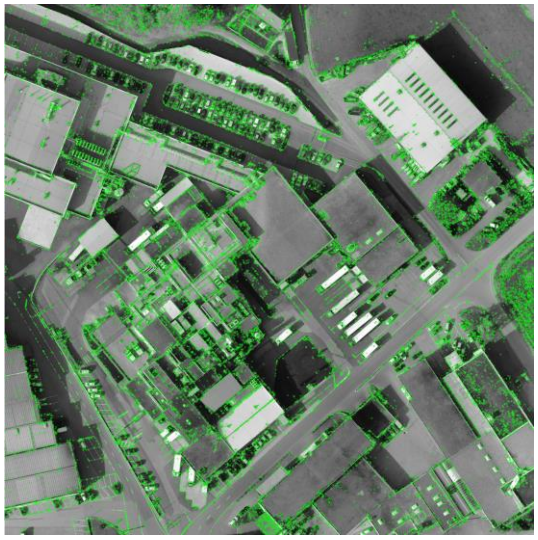
   1) Convert image to grayscale image.
      - gray (O/P) = cv2.cvtColor(img (I/P), cv2.COLOR_BGR2GRAY)
   2) Based on SIFT algorithm, find keypoints on Left.jpg.
      - Use OpenCV SIFT detector to detect keypoints and descriptors.
      - sift = cv2.SIFT_create() # Create a SIFT detector
      - keypoints (O/P), descriptors (O/P) = sift.detectAndCompute(gray (I/P), None)

        # Many SIFT keypoints (contain coordinate, size, angle, etc.),
        each keypoint has its descriptor(1x128)
   3) Then draw the keypoints of Left.jpg as I_1.
      - img = cv2.drawKeypoints(gray (I/P), keypoints (I/P), None, color=(0,255,0)) (keypoints color: b, g, r)
   4) Please show image I_1

Input:

Left.jpg

Output:



I_1

Reference : OpenCV Textbook SIFT, p321, 464, 524

4.2 Matched Keypoints

(出題： Ian)

1. - Click button “Load Image 1” to load Left.jpg.

   - Click button “Load Image 2” to load Right.jpg.

   - Click button “4.2 Matched Keypoints”,

   1) Based on SIFT algorithm, find the keypoints and descriptors at Image 1 and Image 2 (same as question 4.1)

   2) Find match keypoints of two images
      - matches (O/P) = cv2.BFMatcher().knnMatch(descriptors1 (I/P), descriptors2 (I/P), k=2)

   matches[m,n]: two (k=2) closest matched keypoints for each keypoint of Left.jpg. m is the closest, n is the second closest.

   3) Extract Good Matches from 2) result:
      - Hint:

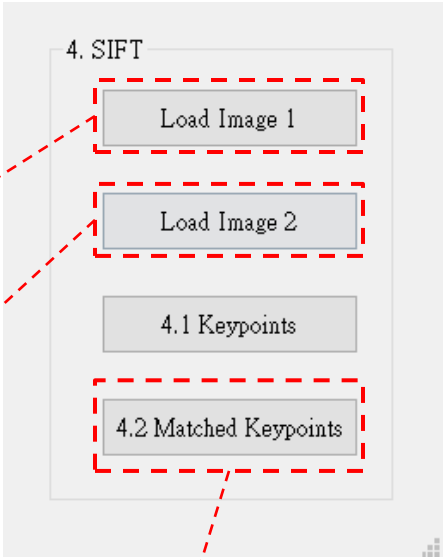
```
for m, n in matches:
    if m.distance < 0.75*n.distance:
        good_matches.append(m)
```

   Ratio test: If the distance to the closest match m is much smaller than the distance to the second closest n, the match is considered good.

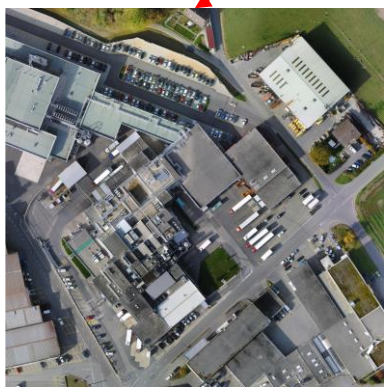
   4) Draw the matched feature points between two image
      - Hint: Use “cv2.drawMatchesKnn()” to draw the matches.

```
cv2.drawMatchesKnn(gray1 (I/P), keypoints1 (I/P), gray2 (I/P), keypoints2 (I/P), good_matches (I/P),
    None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

   5) Please show image I_2



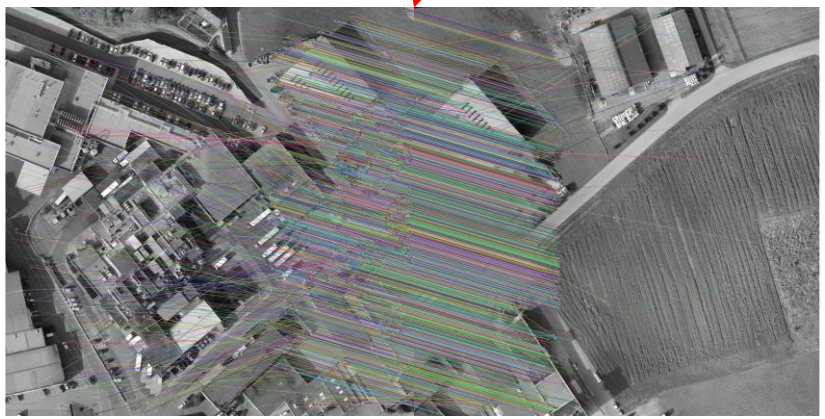
Input:



Left.jpg

Right.jpg

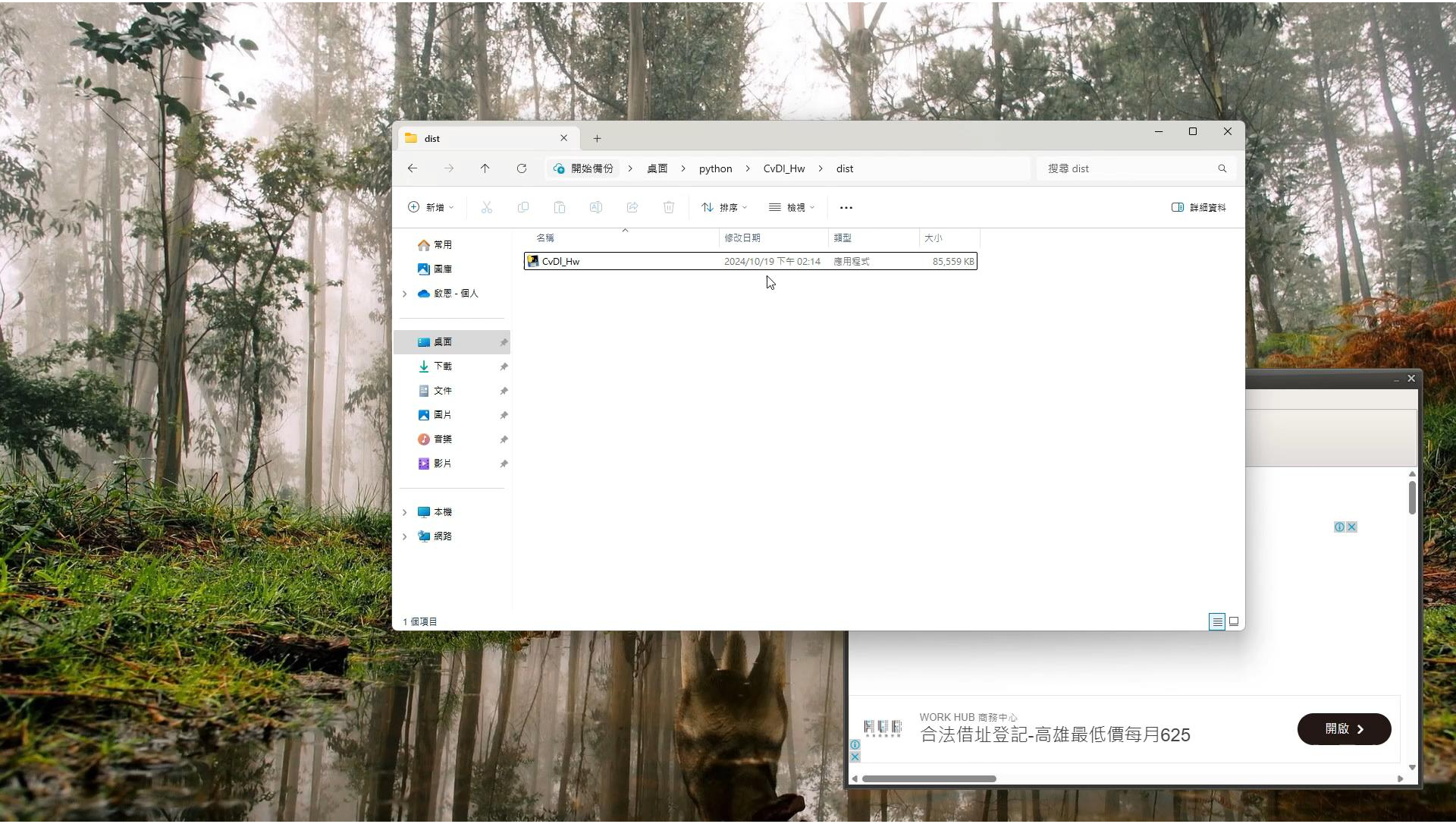
Output:



I_2

Reference: OpenCV Textbook SIFT, p321, 464, 524

4. SIFT - DEMO


dist
開始備份
桌面
python
CvDl_Hw
dist
搜尋 dist
新增
排序
檢視
詳細資料
常用
圖庫
啟恩 - 個人
桌面
下載
文件
圖片
音樂
影片
本機
網路
名稱
修改日期
類型
大小
CvDl_Hw
2024/10/19 下午 02:14
應用程式
85,559 KB
1 個項目
WORK HUB 商務中心
合法借址登記-高雄最低價每月625
開啟