電腦視覺應用
Applications of Computer Vision

擴增實境 i

孫士韋
Shih-Wei Sun
swsun@newmedia.tnua.edu.tw

HMD with AR

- AR Helmet, Microsoft Hololens, 2021
  - https://www.youtube.com/watch?v=Yfp0ajL8n_c
- AR smart Glass, Epson BT-300, 2016
  - https://www.youtube.com/watch?v=hhYPqF3aHUs
- O-Displaying, 2017 (SA, tnua)
  - https://youtu.be/GUoxKL4NXFI

- AR survey, 1997 https://www.cs.unc.edu/~azuma/ARpresence.pdf

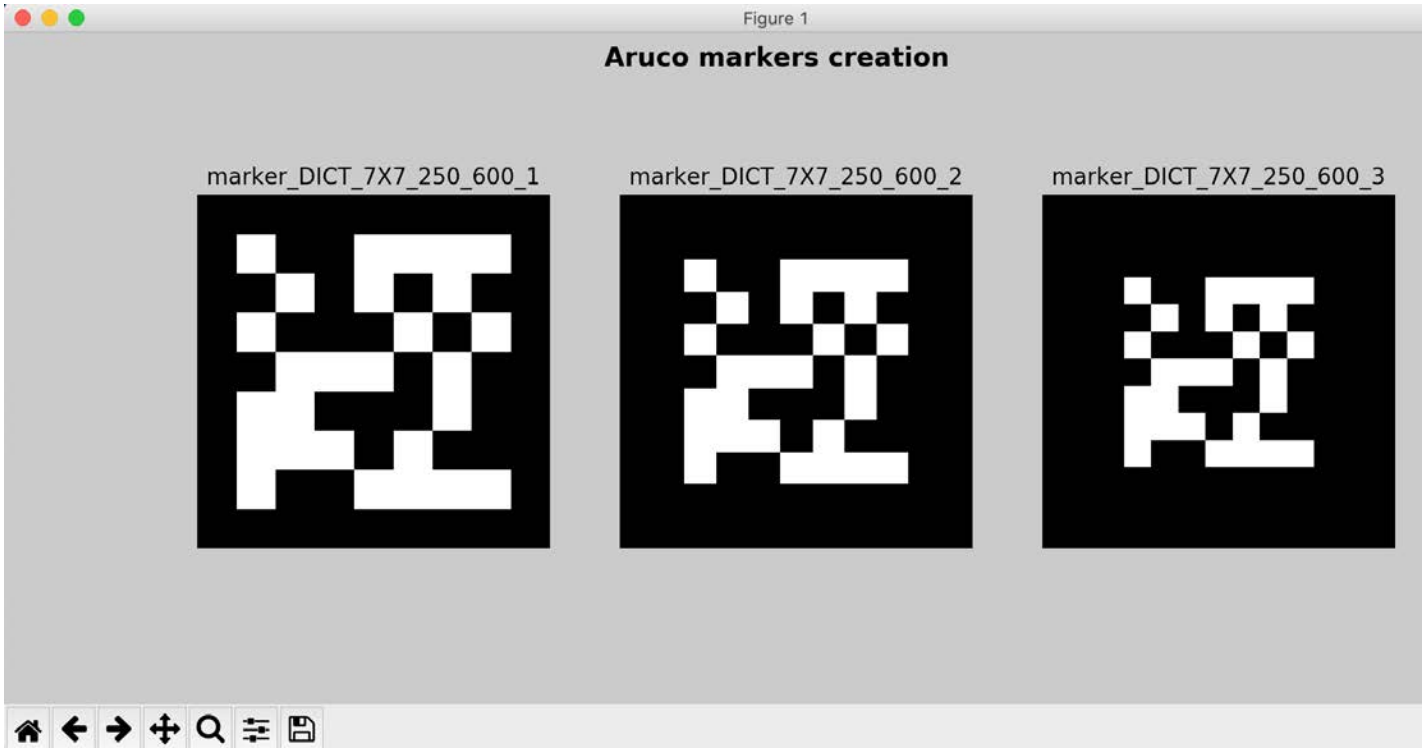
Figure 12: Two optical see-through HMDs, made by Hughes Electronics

Figure 16: External view of the ARGOS system, an example of monitor-based AR. (Courtesy David Drascic and Paul Milgram, U. Toronto.)

Outline

- Creating markers
- Detecting markers
- Camera calibration
- Camera pose estimation

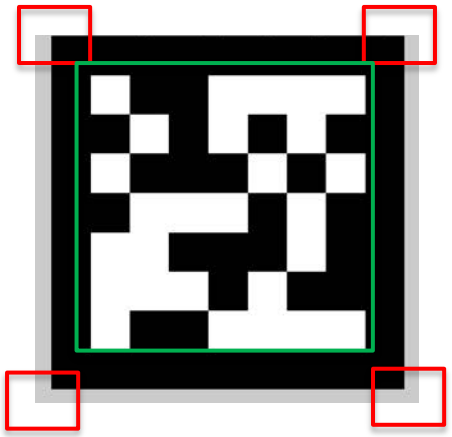
Creating markers



Marker-based AR

- ArUco marker
  - [Garrido-Jurado et al., 2014] (Pattern Recogn.)

S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. 2014. "Automatic generation and detection of highly reliable fiducial markers under occlusion", Pattern Recogn. 47, 6 (June 2014), 2280-2292.



  - Detect markers: 4 corners
  - Correct possible errors
    - Occlusion problem
  - External cell: black
  - Internal cell: marker codes

https://docs.opencv.org/master/d5/dae/tutorial_aruco_detection.html
http://www.uco.es/investiga/grupos/ava/node/26

Importing libraries, plot preparing

```
1  import cv2
2  from matplotlib import pyplot as plt
3
4
5  def show_img_with_matplotlib(color_img, title, pos):
6      # Convert BGR image to RGB
7      img_RGB = color_img[:, :, ::-1]
8
9      ax = plt.subplot(1, 3, pos)
10     plt.imshow(img_RGB)
11     plt.title(title)
12     plt.axis('off')
13
14 # Create the dimensions of the figure and set title:
15 fig = plt.figure(figsize=(12, 5))
16 plt.suptitle("Aruco markers creation", fontsize=14,
       fontweight='bold')
17 fig.patch.set_facecolor('silver')
18
```

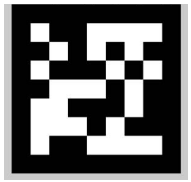
importing classes of ***cv2, pyplot***

Define a function for plotting

Setting figure size, title texts, font size, color

ArUco

- First step:
  - Creation of markers and dictionaries
- Second step
  - Generating markers
    - With desired number/number of internal cells
- Predefined dictionaries with markers
  - Print the markers / show in on your phone
  - Ex:



```
DICT_7X7_250 = 14,
```

```
19 aruco_dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
20
```

Returns the marker ready to be printed

Get the dictionary,
Compose of 250 markers, each marker has a size of 7x7

DrawMarker

- Draw Marker
  - cv2.aruco.drawMarker( )

```
21  aruco_marker_1 =
        cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
        sidePixels=600, borderBits=1)
```

  - 1st parameter: dictionary (output of L20)
  - 2nd parameter: marker id, 0~249 (total 250)
  - 3rd parameter: pixel size/marker image
  - 4th parameter: borderBits/number of bits (cells)

Generating Markers

- Draw 3 Markers
  - cv2.aruco.drawMarker( )
  - The same id
  - Different borderBits

```
21  aruco_marker_1 =
        cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
        sidePixels=600, borderBits=1)
22  aruco_marker_2 =
        cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
        sidePixels=600, borderBits=2)
23  aruco_marker_3 =
        cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
        sidePixels=600, borderBits=3)
```

Save the Markers

- cv2.imwrite( )
- Save the images to disk

Target file path for saving

```
25 # We save the created markers using 'cv2.imwrite()':
26 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_1.png", aruco_marker_1)
27 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_2.png", aruco_marker_2)
28 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_3.png", aruco_marker_3)
```

Plot the markers

- Plot the images
  - show_img_with_matplotlib( )
    - written before

```
29
30 # Plot the images:
31 show_img_with_matplotlib(cv2.cvtColor(aruco_marker_1,
       cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_1", 1)
32 show_img_with_matplotlib(cv2.cvtColor(aruco_marker_2,
       cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_2", 2)
33 show_img_with_matplotlib(cv2.cvtColor(aruco_marker_3,
       cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_3", 3)
34
35 # Show the Figure:
36 plt.show()
```

Show the figures on the screen

Full code (1/3)

```
1  import cv2
2  from matplotlib import pyplot as plt
3
4
5  def show_img_with_matplotlib(color_img, title, pos):
6      # Convert BGR image to RGB
7      img_RGB = color_img[:, :, ::-1]
8
9      ax = plt.subplot(1, 3, pos)
10     plt.imshow(img_RGB)
11     plt.title(title)
12     plt.axis('off')
13
14 # Create the dimensions of the figure and set title:
15 fig = plt.figure(figsize=(12, 5))
16 plt.suptitle("Aruco markers creation", fontsize=14,
       fontweight='bold')
17 fig.patch.set_facecolor('silver')
18
19 aruco_dictionary =
       cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
20
```

Full code (2/3)

```
21 aruco_marker_1 =
       cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
       sidePixels=600, borderBits=1)
22 aruco_marker_2 =
       cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
       sidePixels=600, borderBits=2)
23 aruco_marker_3 =
       cv2.aruco.drawMarker(dictionary=aruco_dictionary, id=2,
       sidePixels=600, borderBits=3)
24
25 # We save the created markers using 'cv2.imwrite()':
26 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_1.png", aruco_marker_1)
27 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_2.png", aruco_marker_2)
28 cv2
       .imwrite
       ("/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       marker_DICT_7X7_250_600_3.png", aruco_marker_3)
```

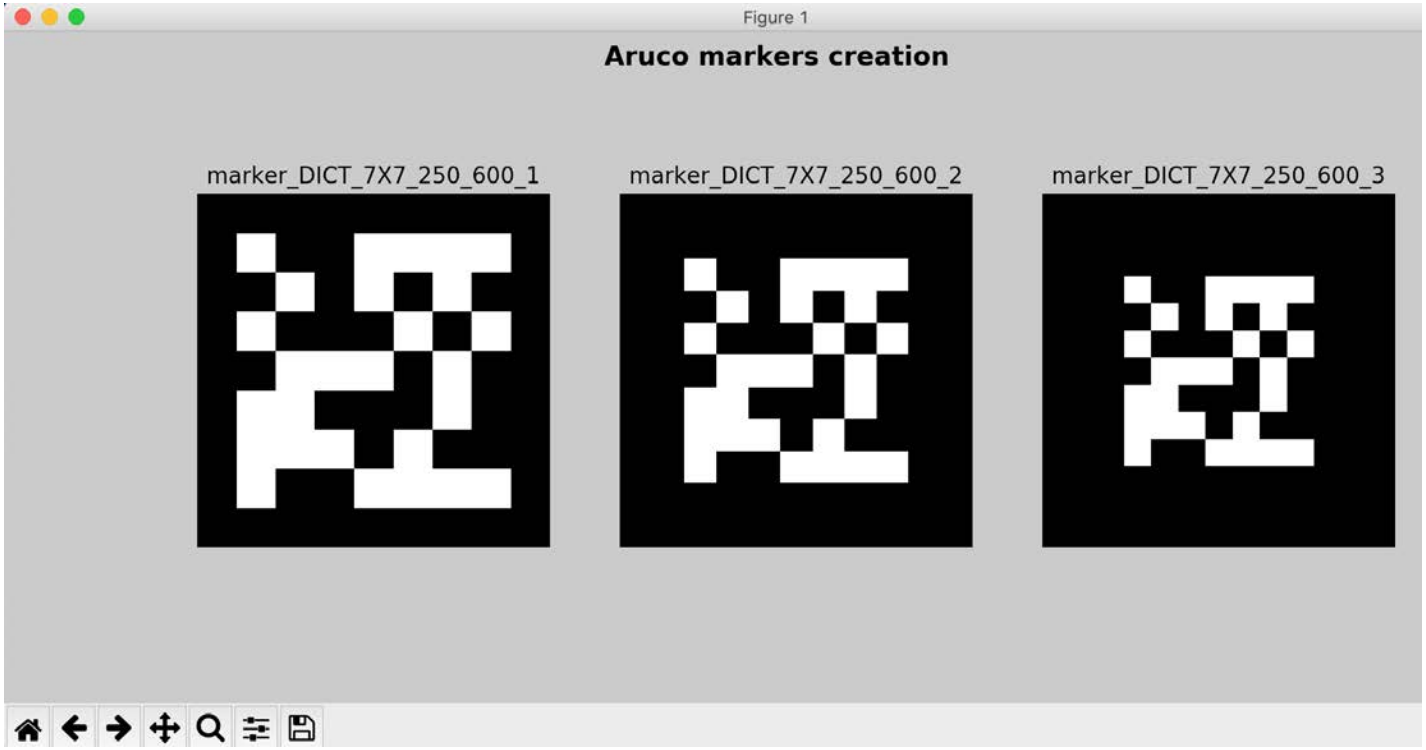
Full code (3/3)

```
29
30  # Plot the images:
31  show_img_with_matplotlib(cv2.cvtColor(aruco_marker_1,
        cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_1", 1)
32  show_img_with_matplotlib(cv2.cvtColor(aruco_marker_2,
        cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_2", 2)
33  show_img_with_matplotlib(cv2.cvtColor(aruco_marker_3,
        cv2.COLOR_GRAY2BGR), "marker_DICT_7X7_250_600_3", 3)
34
35  # Show the Figure:
36  plt.show()
```

Practice 2

- Creating the marker:

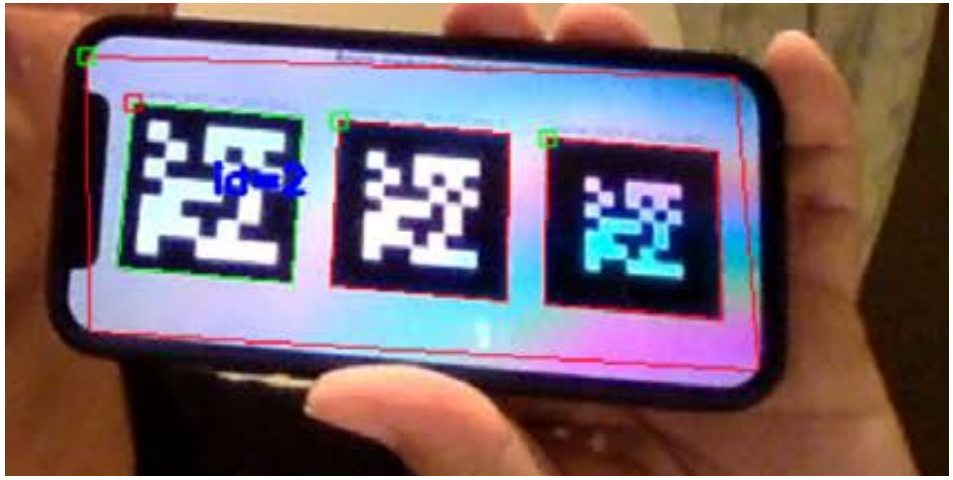
Results:



Saved files:

Detecting markers

- Take a picture for the markers
  - from the screen
- Show it on your phone!



Detect the markers

- Use the function
  - cv2.aruco.detectMarkers( )

```
13      corners, ids, rejected_corners =
            cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
            parameters=parameters)
```

  - 1st parameter: gray scale image
  - 2nd parameter: dictionary object
  - 3rd parameter: parameters

```
3   aruco_dictionary =
        cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
4
5   parameters = cv2.aruco.DetectorParameters_create()
```

Returned items

- From the function
  - cv2.aruco.detectMarkers( )

```
13      corners, ids, rejected_corners =
            cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
            parameters=parameters)
```

  - 1st parameter: detected corners
    - 4 corners (top-left, top-right, bottom-right, bottom-left)
  - 2nd parameter: id
  - 3rd parameter: rejected corners
    - For debugging purpose; 4 corners

Detect markers from a camera

- Enable the web camera
  - From your computer

```
1  import cv2
7  capture = cv2.VideoCapture(0)
8  while True:
27 capture.release()
28 cv2.destroyAllWindows()
```

Enable the camera

Codes L9 – L24

Release the camera

Draw markers

- From the function
  - cv2.aruco.drawdDetectMarkers( )
- Detected markers
- Rejected markers

image of the current frame

```
15  # Draw detected markers:
16  frame = cv2.aruco.drawDetectedMarkers(image=frame,
        corners=corners, ids=ids, borderColor=(0, 255, 0))
17
18  # Draw rejected markers:
19  frame = cv2.aruco.drawDetectedMarkers(image=frame,
        corners=rejected_corners, borderColor=(0, 0, 255))
20
```

corners id B,G,R

rejected corners B,G,R

Full code (1/3)

```
1  import cv2
2
3  aruco_dictionary =
       cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
4
5  parameters = cv2.aruco.DetectorParameters_create()
6
7  capture = cv2.VideoCapture(0)
```

Full code (2/3)

```
8  while True:
9      ret, frame = capture.read()
10
11     gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
12
13     corners, ids, rejected_corners =
           cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
           parameters=parameters)
14
15     # Draw detected markers:
16     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=corners, ids=ids, borderColor=(0, 255, 0))
17
18     # Draw rejected markers:
19     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=rejected_corners, borderColor=(0, 0, 255))
20
21     # Display the resulting frame
22     cv2.imshow('frame', frame)
23     if cv2.waitKey(1) & 0xFF == ord('q'):
24         break
```

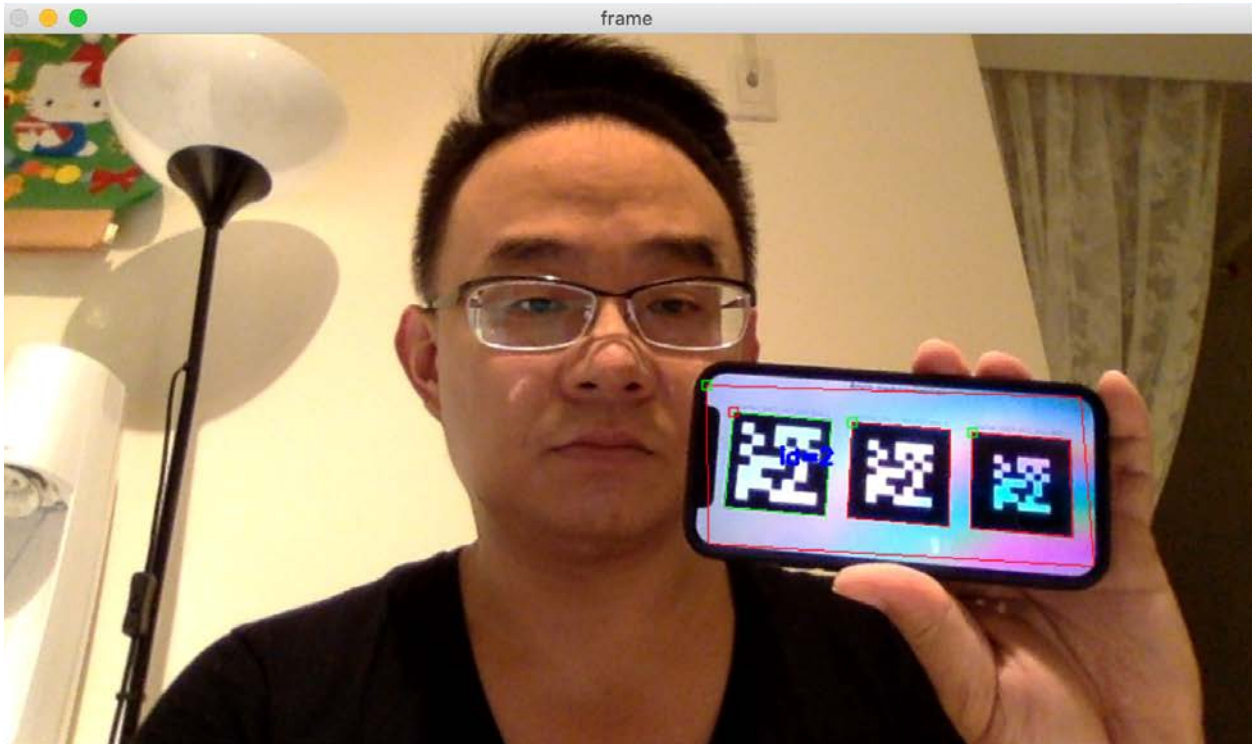
Full code (3/3)

```
25
26 # Release everything:
27 capture.release()
28 cv2.destroyAllWindows()
```

Practice 3

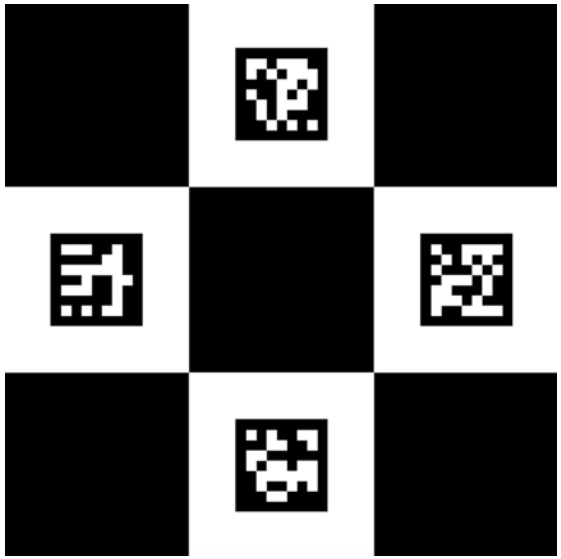
- Draw detected markers

Results:



Camera calibration

- Before obtaining the camera pose
  - Using the detected markers
- It is necessary to know
  - The calibration parameters of the camera
- Performed only once
  - Camera optics
    - Not modified



Calibration process

- From the calibration function
  - cv2.aruco.calibrateCameraCharuco( )
- Using a set of corners: from several views
  - Extracted from a board

```
42      cal = cv2.aruco.calibrateCameraCharuco(all_corners, all_ids,
            board, gray.shape, None, None)
```

- Returns:
  - Camera matrix (3x3 matrix)
    - Focal distances, camera center coordinates
    - intrinsic parameters
  - Vector with distortion coefficients

Calibration (1/2)

- Syntax:

```
calibrateCameraCharuco(charucoCorners, charucoIds, board, imageSize,
cameraMatrix, distCoeffs[, rvecs[, tvecs[, flags[, criteria]]]]) -> retval,
cameraMatrix, distCoeffs, rvecs, tvecs
```

- charucoCorners
  - A vector containing detected corners
- charucoIds: list of identifiers
- Board: board layout
- imageSize: input image size

Calibration (2/2)

- Output vectors:

```
calibrateCameraCharuco(charucoCorners, charucoIds, board, imageSize,
cameraMatrix, distCoeffs[, rvecs[, tvecs[, flags[, criteria]]]]) -> retval,
cameraMatrix, distCoeffs, rvecs, tvecs
```

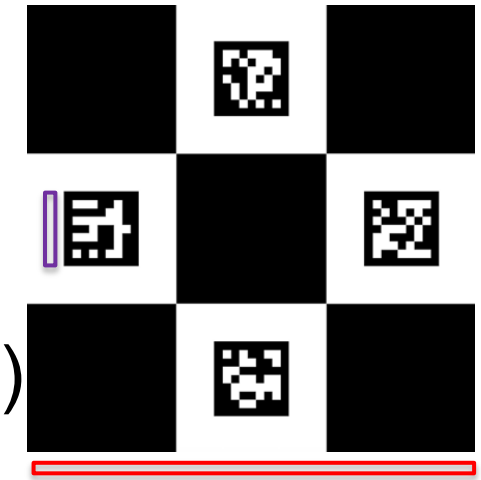
- rvecs: estimated rotation vectors
  - For each board view
- tvecs: estimated translation vectors
  - For pattern view
- cameraMatrix: board layout
- distCoeffs: distortion coefficients

items in cal

```
42      cal = cv2.aruco.calibrateCameraCharuco(all_corners, all_ids,
            board, gray.shape, None, None)

48  retval, cameraMatrix, distCoeffs, rvecs, tvecs = cal
```

Board



- From the calibration function
  - cv2.aruco.CharucoBoard_create( )

The signature is as follows:

```
CharucoBoard_create(squaresX, squaresY, squareLength, markerLength,
dictionary) -> retval
```

```
7 board = cv2.aruco.CharucoBoard_create(3, 3, .025, .0125,
    dictionary)
```

  - 1st parameter: # of squares in x-direction
  - 2nd parameter: # of squares in y-direction
  - 3rd parameter: square side length (meter)
  - 4th parameter: marker length (meter)
  - 5th parameter: dictionary

Assigning the board

- Dictionary and board

```
6 dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
7 board = cv2.aruco.CharucoBoard_create(3, 3, .025, .0125,
      dictionary)
8
9 image_board = board.draw((200 * 3, 200 * 3))
```

Save the vectors to a file

- Open file f: L51
- Close file f: L53
- Save for: cameraMatrix, disCoeffs

```
50 # Save the camera parameters:
51 f =
       open
       ('/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       calibration2.pckl', 'wb')
52 pickle.dump((cameraMatrix, distCoeffs), f)
53 f.close()
```

Full code (1/3)

```
1  import time
2  import cv2
3  import numpy as np
4  import pickle
5
6  dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
7  board = cv2.aruco.CharucoBoard_create(3, 3, .025, .0125,
       dictionary)
8
9  image_board = board.draw((200 * 3, 200 * 3))
10
11 cv2
       .imwrite
       ('/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       charuco.png', image_board)
12
13 cap = cv2.VideoCapture(0)
14
15 all_corners = []
16 all_ids = []
17 counter = 0
18
```

0. Target file path
for saving (1st time running)

Full code (2/3)

```
19 for i in range(300):
20     ret, frame = cap.read()
21     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
22                                     1. Detect markers, Draw detected markers
23     # Detect markers:
24     res = cv2.aruco.detectMarkers(gray, dictionary)
25
26     if len(res[0]) > 0:
27         res2 = cv2.aruco.interpolateCornersCharuco(res[0],
               res[1], gray, board)
28         if res2[1] is not None and res2[2] is not None and
               len(res2[1]) > 3 and counter % 3 == 0:
29             all_corners.append(res2[1])
30             all_ids.append(res2[2])
31
32         cv2.aruco.drawDetectedMarkers(gray, res[0], res[1])
33
34     cv2.imshow('frame', gray)
35     if cv2.waitKey(1) & 0xFF == ord('q'):
36         break
37
38     counter += 1
39
```

Full code (3/3)

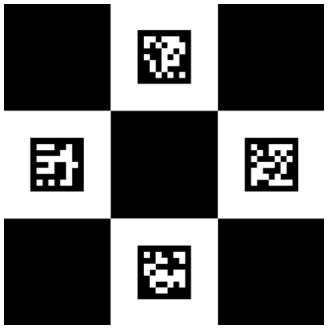
```
40 # Calibration can fail for many reasons:
41 try:
42     cal = cv2.aruco.calibrateCameraCharuco(all_corners, all_ids,
           board, gray.shape, None, None)
43 except:
44     cap.release()
45     print("Calibration could not be done ...")
46
47 # Get the calibration result:
48 retval, cameraMatrix, distCoeffs, rvecs, tvecs = cal
49
50 # Save the camera parameters:
51 f =
       open
       ('/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       calibration2.pckl', 'wb')
52 pickle.dump((cameraMatrix, distCoeffs), f)
53 f.close()
54
55 # Release everything:
56 cap.release()
57 cv2.destroyAllWindows()
```

2. Assign the detected marker information:
Matrix, rotation vector, translation vector

3. Target file path
for saving the pckl file
Camera matrix, disCoeffs

Practice 4, Calibration

- Frist run for the code:



charuco.png

Generate the file:

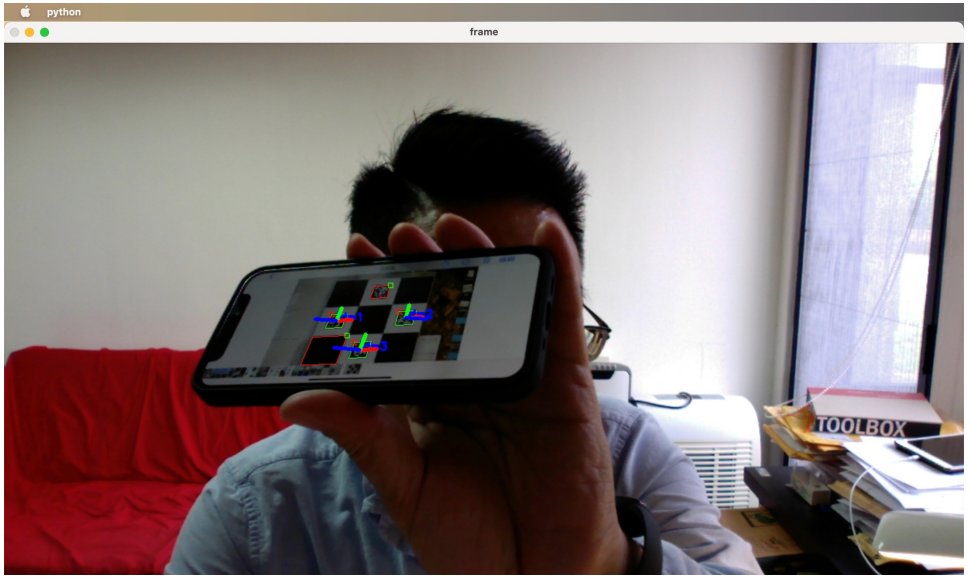
- Second run for the code

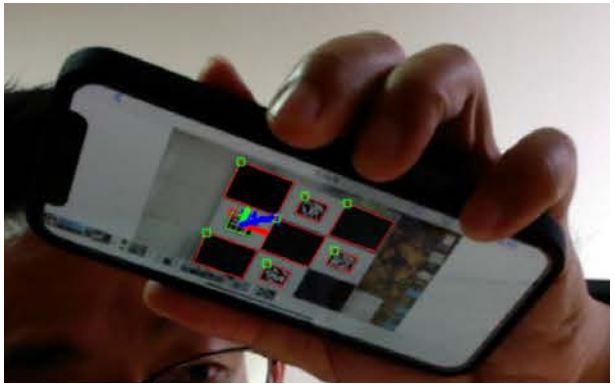
calibration2.pckl

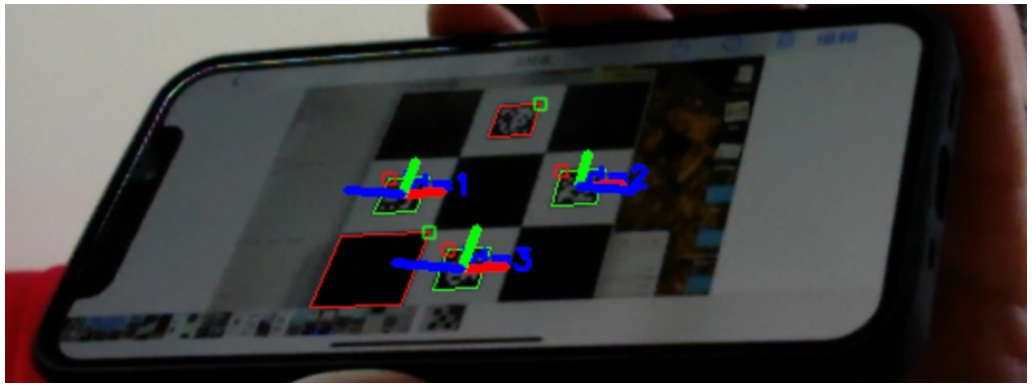
To be used later

Calibrated matrix used for camera pose estimation

Next week…







Textbook Reading

- Mastering OpenCV 4 with Python
  - Ch 9, Augmented Reality
    - Marker-based augmented reality
      - p. 268 -p. 273