電腦視覺應用
Applications of Computer Vision

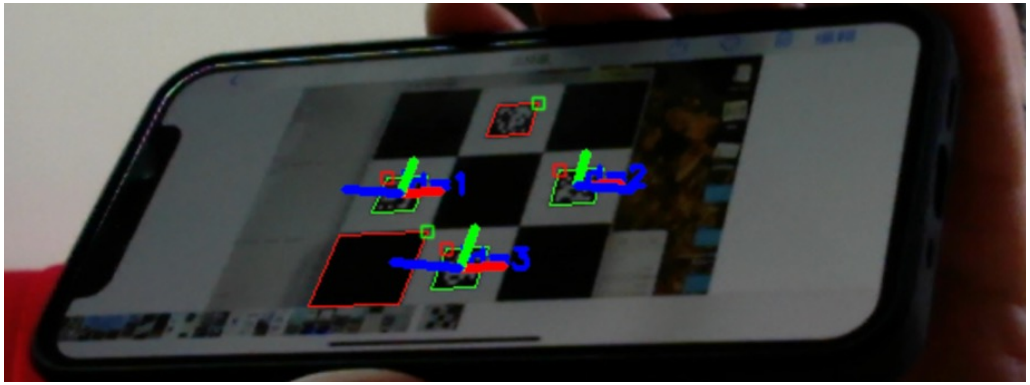
擴增實境 ii

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

Outline

- Camera pose estimation
- Camera pose estimation and basic augmentation
- Camera pose estimation and advanced augmentation

Camera pose estimation

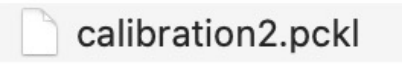


Preprocessing: Load the calibration file

```
1  import cv2
2  import os
3  import pickle
4
5  if not
       os.path
       .exists
       ('/Users/swsun/Documents/python/Chapter09/01-chapter-content/
       calibration2.pckl'):
6      print("You need to calibrate the camera you'll be using. See
           calibration script.")
7      exit()
8  else:
9      f =
           open
           ('/Users/swsun/Documents/python/Chapter09/01-chapter-cont
           ent/calibration2.pckl', 'rb')
10     cameraMatrix, distCoeffs = pickle.load(f)
11     f.close()
12     if cameraMatrix is None or distCoeffs is None:
13         print("Calibration issue. Remove ./calibration.pckl and
               recalibrate your camera")
14         exit()
15
```

Importing the class of ***cv2, os, pickle***

Try to remove the calibration file
Showing this information



Check the file by the path

Warning information

Sep 1: Open the file

Assigning the file path

Assign the data

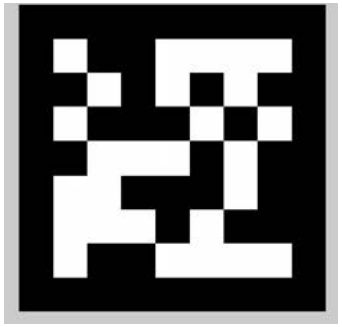
Step 2: Load the data

Step 3: close the file

Warning information, no data in the file

Assigning the Aruco Dictionary

- First step:
  - Get the markers and dictionaries
- Second step
  - Creating parameters
    - With desired number/number of internal cells
- Predefined dictionaries with markers



```
16 aruco_dictionary =
       cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
17
18 parameters = cv2.aruco.DetectorParameters_create()
```

The same as the previous week

DICT_7X7_250

7x7 cells, 0~249 codes

Read frames from a camera

- Enable the web camera
  - From your computer

Coding style in *Hello world, Camera*

```
1  import cv2

20 capture = cv2.VideoCapture(0)
21
22 while True:
43 # Release everything:
44 capture.release()
45 cv2.destroyAllWindows()
```

Enable the camera

Codes L23 – L42

```
23 ret, frame = capture.read()
24 gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

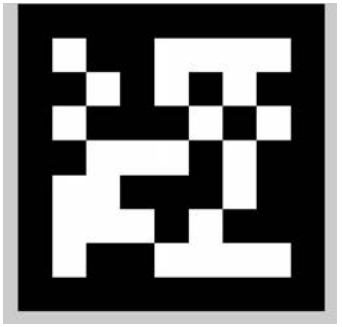
Read the frame
Convert to gray

```
39 cv2.imshow('frame', frame)
40 if cv2.waitKey(1) & 0xFF == ord('q'):
41     break
```

Show the frame
Break for pressing ‘q’

Release the camera

Detect the marker



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

The same coding style
In the previous week

```
26      corners, ids, rejectedImgPoints =
            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

Draw the detected markers

The same coding style
In the previous week

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

image of the current frame

```
28  frame = cv2.aruco.drawDetectedMarkers(image=frame,
        corners=corners, ids=ids, borderColor=(0, 255, 0))
29
30  frame = cv2.aruco.drawDetectedMarkers(image=frame,
        corners=rejectedImgPoints, borderColor=(0, 0, 255))
```

corners id B,G,R

rejected corners B,G,R

Key method

Camera Pose Estimation (1/2)

- With the detected *ids*, from the function
  - cv2.aruco.*estimatePoseSingleMarkers*( )
    - Estimates the pose for single markers

```
32  if ids is not None:
33      rvecs, tvecs, _ =
            cv2.aruco.estimatePoseSingleMarkers(corners, 1,
            cameraMatrix, distCoeffs)
```

  - 1st parameter: detected corners
    - 4 corners (top-left, top-right, bottom-right, bottom-left)
  - 2nd parameter: length of the marker side
  - 3rd, 4th parameters: read from the file
    - cameraMatrix, disCoeffs

pckl file in the last week

Key method

Camera Pose Estimation (2/2)

- Returned values from the function
  - cv2.aruco.*estimatePoseSingleMarkers*( )
    - Calculate the pose difference from the given
      - cameraMatrix, disCoeffs

```
32  if ids is not None:
33      rvecs, tvecs, _ =
            cv2.aruco.estimatePoseSingleMarkers(corners, 1,
            cameraMatrix, distCoeffs)
```

Scale *defined here!!*

  - 1st parameter, rvecs:
    - estimated ***rotation*** vectors
  - 2nd parameter, tvecs:
    - estimated ***translation*** vectors

How to define an object in a 3D space?
Rotation,
Scale,
position (translation)

Draw the 3 Axes

- Estimate from the function
  - cv2.aruco.*drawAxis*( )
    - By the detected and given matrix (from the file)

```
cv2.aruco.drawAxis(frame, cameraMatrix, distCoeffs,
    rvec, tvec, 1)
```

Draw to the frame

Marker length

  - 1st parameter: frame
  - 2nd and 3rd parameters: (read from the file)
    - cameraMatrix, disCoeffs
  - 4nd and 5th parameters: rvecs, tvecs (estimted)
  - 6th parameter: marker length (real space unit)

Full code (1/3)

```
import cv2
import os
import pickle

if not
    os.path
    .exists
    ('/Users/swsun/Documents/python/Chapter09/01-chapter-content/
    calibration2.pckl'):
    print("You need to calibrate the camera you'll be using. See
        calibration script.")
    exit()
else:
    f =
        open
        ('/Users/swsun/Documents/python/Chapter09/01-chapter-cont
        ent/calibration2.pckl', 'rb')
    cameraMatrix, distCoeffs = pickle.load(f)
    f.close()
    if cameraMatrix is None or distCoeffs is None:
        print("Calibration issue. Remove ./calibration.pckl and
            recalibrate your camera")
        exit()

```

Full code (2/3)

```
16 aruco_dictionary =
       cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
17
18 parameters = cv2.aruco.DetectorParameters_create()
19
20 capture = cv2.VideoCapture(0)
21
22 while True:
23     ret, frame = capture.read()
24     gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
25
26     corners, ids, rejectedImgPoints =
           cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
           parameters=parameters)
27
28     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=corners, ids=ids, borderColor=(0, 255, 0))
29
30     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=rejectedImgPoints, borderColor=(0, 0, 255))
31
```

The same space

Full code (3/3)

The same space

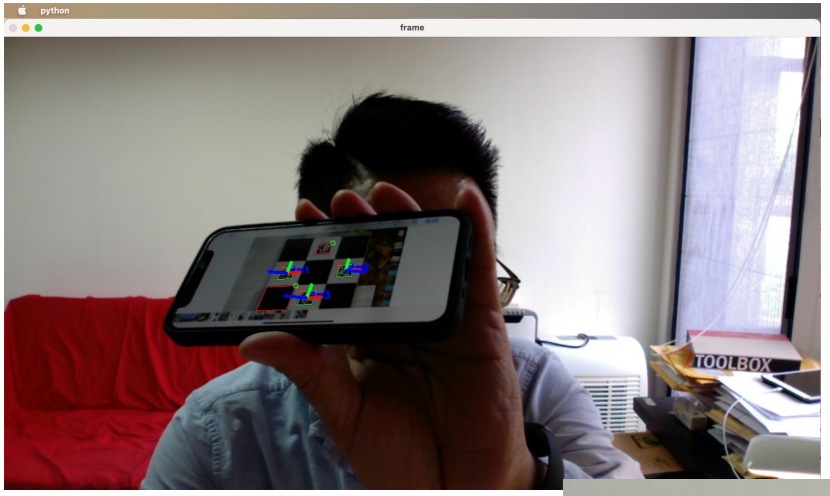
```
32      if ids is not None:
33          rvecs, tvecs, _ =
                cv2.aruco.estimatePoseSingleMarkers(corners, 1,
                cameraMatrix, distCoeffs)
34
35          for rvec, tvec in zip(rvecs, tvecs):
36              cv2.aruco.drawAxis(frame, cameraMatrix, distCoeffs,
                    rvec, tvec, 1)
37
38      # Display the resulting frame
39      cv2.imshow('frame', frame)
40      if cv2.waitKey(1) & 0xFF == ord('q'):
41          break
42
43  # Release everything:
44  capture.release()
45  cv2.destroyAllWindows()
```

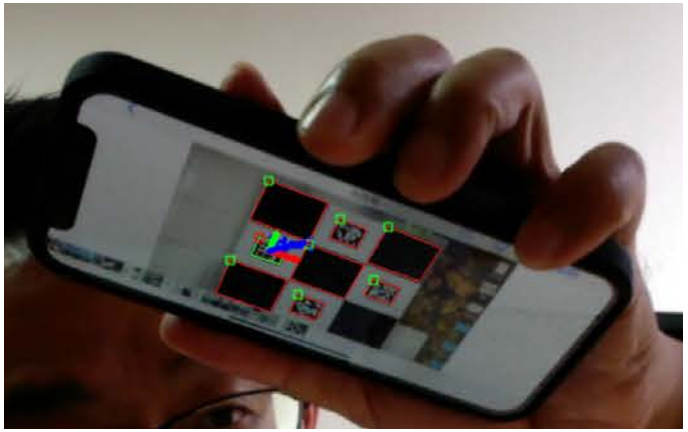
zip: parallel iteration
https://realpython.com/python-zip-function/

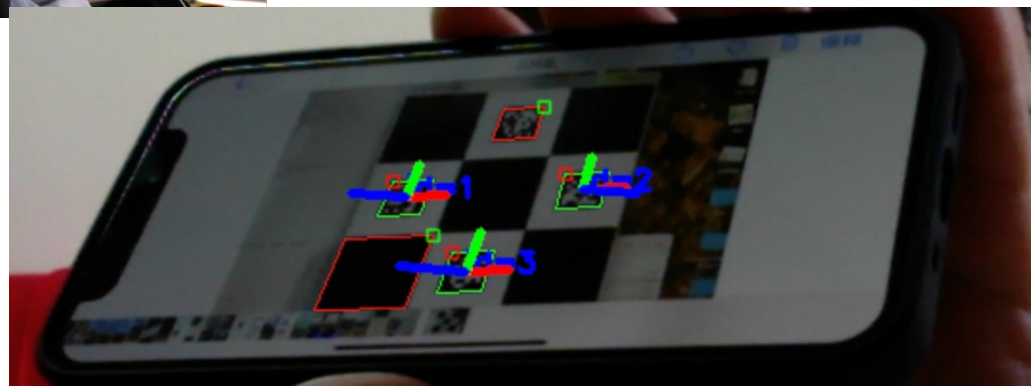
Practice 1

- Draw the results:

calibration2.pckl







Camera pose estimation and basic augmentation

Assigning 4 corners

- Marker Coordinate system
  - Centered on the middle of the marker
    - Coordinates of the 4 corners of the marker
      - `(-markerLength/2, markerLength/2, 0)`
      - `(markerLength/2, markerLength/2, 0)`
      - `(markerLength/2, -markerLength/2, 0)`
      - `(-markerLength/2, -markerLength/2, 0)`
    - Assign 4 corners to the array

```
51          desired_points = np.float32([[-1 / 2, 1 / 2, 0], [1 /
                2, 1 / 2, 0], [1 / 2, -1 / 2, 0], [-1 / 2, -1 / 2,
                0]]) * OVERLAY_SIZE_PER
```

Project the Points

- Project by the function
  - cv2.aruco.*projectPoints*( )

```
54              # Project the points:
55              projected_desired_points, jac =
                    cv2.projectPoints(desired_points, rvecs, tvecs,
                    cameraMatrix, distCoeffs)
```

  - 1st parameter: desired_points
  - 2nd and 3rd parameters: rvecs, tvecs (estimted)
  - 4nd and 5th parameters: (read from the file)
    - cameraMatrix, disCoeffs

Draw the Projected Points

- Project by the function
  - cv2.aruco.*projectPoints*( )

```
58            draw_points(frame, projected_desired_points)
```

  - 1st parameter: frame
  - 2nd parameter: projected_desired_points

```
19 def draw_points(img, pts):
20     """ Draw the points in the image"""
21
22     pts = np.int32(pts).reshape(-1, 2)
23
24     img = cv2.drawContours(img, [pts], -1, (255, 255, 0), -3)
25
26     for p in pts:
27         cv2.circle(img, (p[0], p[1]), 5, (255, 0, 255), -1)
28
29     return img
```

https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html
draw all the contours": -1, 3

B,G,R cyan color

Circle center radius

B,G,R pink color

https://www.geeksforgeeks.org/python-opencv-cv2-circle-method/

Full code (1/4)

```
1  import cv2
2  import os
3  import pickle
4  import numpy as np
5
6  OVERLAY_SIZE_PER = 1
7
8  if not
       os.path
       .exists('/Users/sunshih-wei/Documents/python/CV_07/calibration2
       .pckl'):
9      print("You need to calibrate the camera you'll be using. See
           calibration script.")
10     exit()
11 else:
12     f =
           open
           ('/Users/sunshih-wei/Documents/python/CV_07/calibration2
           .pckl', 'rb')
13     cameraMatrix, distCoeffs = pickle.load(f)
14     f.close()
15     if cameraMatrix is None or distCoeffs is None:
16         print("Calibration issue. Remove ./calibration.pckl and
               recalibrate your camera")
17         exit()
18
```

Add a new variable

Full code (2/4)

Add a new method

```
19 def draw_points(img, pts):
20     """ Draw the points in the image"""
21
22     pts = np.int32(pts).reshape(-1, 2)
23
24     img = cv2.drawContours(img, [pts], -1, (255, 255, 0), -3)
25
26     for p in pts:
27         cv2.circle(img, (p[0], p[1]), 5, (255, 0, 255), -1)
28
29     return img
30
31 aruco_dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
32
33 parameters = cv2.aruco.DetectorParameters_create()
34
35 capture = cv2.VideoCapture(0)
36
```

Full code (3/4)

```
37 while True:
38     ret, frame = capture.read()
39     gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
40
41     corners, ids, rejectedImgPoints =
           cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
           parameters=parameters)
42
43     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=corners, ids=ids, borderColor=(0, 255, 0))
44
45     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=rejectedImgPoints, borderColor=(0, 0, 255))
46
```

Full code (4/4)

```
47      if ids is not None:
48          rvecs, tvecs, _ =
                cv2.aruco.estimatePoseSingleMarkers(corners, 1,
                cameraMatrix, distCoeffs)
49
50          for rvec, tvec in zip(rvecs, tvecs):
51              desired_points = np.float32([[-1 / 2, 1 / 2, 0], [1 /
                    2, 1 / 2, 0], [1 / 2, -1 / 2, 0], [-1 / 2, -1 / 2,
                    0]]) * OVERLAY_SIZE_PER
52
54              # Project the points:
55              projected_desired_points, jac =
                    cv2.projectPoints(desired_points, rvecs, tvecs,
                    cameraMatrix, distCoeffs)
56
                # Draw the projected points (debugging):
58              draw_points(frame, projected_desired_points)
59
60      # Display the resulting frame
        cv2.imshow('frame', frame)
62      if cv2.waitKey(1) & 0xFF == ord('q'):
63          break
64
65  # Release everything:
66  capture.release()
    cv2.destroyAllWindows()
```

Add new lines of codes

Except the new method Only

3 new Lines of new codes

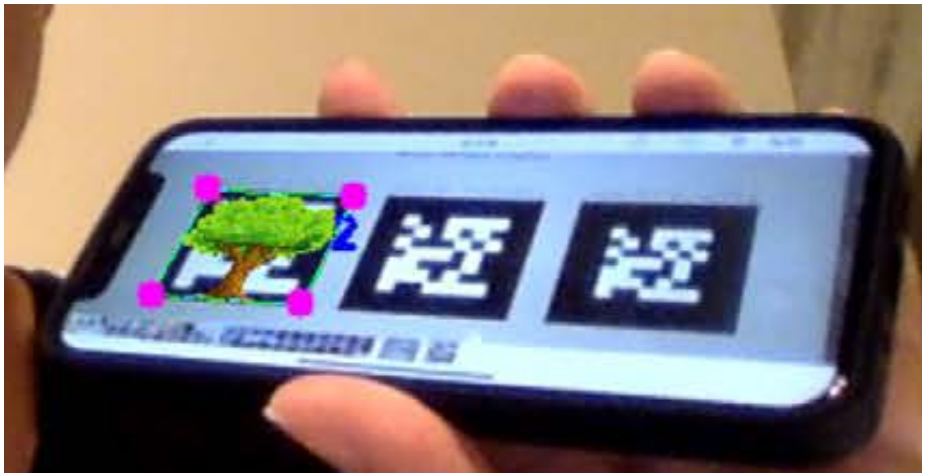
Practice 2

- Try to run the code:

You need the file generated in the last week

Results:

Camera pose estimation and advanced augmentation



Assigning an overlay image

- Read a image: overlay

```
19 overlay =
       cv2
       .imread
       ("/Users/sunshih-wei/Documents/python/Chapter09/01-chapter-cont
       ent/tree_overlay.png")
```

- Draw the overlay image

```
82             frame =
                   draw_augmented_overlay(projected_desired_points,
                   overlay, frame)
```

- Call the method: draw_augmented_overlay( )

```
33 def draw_augmented_overlay(pts_1, overlay_image, image):
34     """Overlay the image 'overlay_image' onto the image 'image'"""
35     pts_2 = np.float32([[0, 0], [overlay_image.shape[1], 0],
           [overlay_image.shape[1], overlay_image.shape[0]],
36                         [0, overlay_image.shape[0]]])
37
38     cv2.rectangle(overlay_image, (0, 0), (overlay_image.shape[1],
           overlay_image.shape[0]), (255, 255, 0), 10)
39
40     M = cv2.getPerspectiveTransform(pts_2, pts_1)
41     dst_image = cv2.warpPerspective(overlay_image, M,
           (image.shape[1], image.shape[0]))
42
43     # Create the mask:
44     dst_image_gray = cv2.cvtColor(dst_image, cv2.COLOR_BGR2GRAY)
45     ret, mask = cv2.threshold(dst_image_gray, 0, 255,
           cv2.THRESH_BINARY_INV)
46
47
48     image_masked = cv2.bitwise_and(image, image, mask=mask)
49
50     result = cv2.add(dst_image, image_masked)
51     return result
52
53
```

Overlay an image

- Overlay the image of a tree

- 1.Define the squares of the overlay image
- 2. transform matrix
- 3. mask
- 4. perform addition

1. Define the squares of the overlay image

- The input of the function

Projected corner points

overlay image

current frame

```
33 def draw_augmented_overlay(pts_1, overlay_image, image):
34     """Overlay the image 'overlay_image' onto the image 'image'"""
35     pts_2 = np.float32([[0, 0], [overlay_image.shape[1], 0],
           [overlay_image.shape[1], overlay_image.shape[0]],
           [0, overlay_image.shape[0]]])
36
37
38     cv2.rectangle(overlay_image, (0, 0), (overlay_image.shape[1],
           overlay_image.shape[0]), (255, 255, 0), 10)
39
```

Corner points of the Overlay image

overlay image

top-left corner

bottom-right corner

B,G,R cyan

thickness of 10 px

https://docs.opencv.org/master/dc/da5/tutorial_py_drawing_functions.html

2. Transform matrix

- Perspective Transformation
  - warping

Perspective Transform: Homography; H

Corner points of the Overlay image

Projected corner points

```
40      M = cv2.getPerspectiveTransform(pts_2, pts_1)
41      dst_image = cv2.warpPerspective(overlay_image, M,
            (image.shape[1], image.shape[0]))
```

To the Destination image

Warp the overlay image

3. Mask

- Generating the mask

```
43      # Create the mask:
44      dst_image_gray = cv2.cvtColor(dst_image, cv2.COLOR_BGR2GRAY)
45      ret, mask = cv2.threshold(dst_image_gray, 0, 255,
            cv2.THRESH_BINARY_INV)
```

Convert to gray-level image

Get the foreground (binarize)

mask: a foreground image
foreground: 1
background: 0

4. Perform addition

- bitwise and operation

bitwise_and: 0, 1

current frame

```
48      image_masked = cv2.bitwise_and(image, image, mask=mask)
49
50      result = cv2.add(dst_image, image_masked)
51      return result
```

Put the overlay image on the frame

mask:
foreground: 1
background: 0

image

bitwise_and

https://docs.opencv.org/master/d0/d86/tutorial_py_image_arithmetics.html

Full code (1/5)

```
1  import cv2
2  import os
3  import pickle
4  import numpy as np
5
6  OVERLAY_SIZE_PER = 1
7
8  if not
       os.path
       .exists
       ('/Users/sunshih-wei/Documents/python/CV_07/calibration2
       .pckl'):
9      print("You need to calibrate the camera you'll be using. See
           calibration script.")
10     exit()
11 else:
12     f =
           open
           ('/Users/sunshih-wei/Documents/python/CV_07/calibration2
           .pckl', 'rb')
13     cameraMatrix, distCoeffs = pickle.load(f)
14     f.close()
15     if cameraMatrix is None or distCoeffs is None:
16         print("Calibration issue. Remove ./calibration.pckl and
               recalibrate your camera")
17         exit()
18
```

Full code (2/5)

Read the image

```
19 overlay =
       cv2
       .imread
       ("/Users/sunshih-wei/Documents/python/Chapter09/01-chapter-cont
       ent/tree_overlay.png")
20
21 def draw_points(img, pts):
22     """ Draw the points in the image"""
23
24     pts = np.int32(pts).reshape(-1, 2)
25
26     img = cv2.drawContours(img, [pts], -1, (255, 255, 0), -3)
27
28     for p in pts:
29         cv2.circle(img, (p[0], p[1]), 5, (255, 0, 255), -1)
30
31     return img
32
```

Full code (3/5)

Add new codes

```
33  def draw_augmented_overlay(pts_1, overlay_image, image):
34      """Overlay the image 'overlay_image' onto the image 'image'"""
35      pts_2 = np.float32([[0, 0], [overlay_image.shape[1], 0],
            [overlay_image.shape[1], overlay_image.shape[0]],
36                          [0, overlay_image.shape[0]]])
37
38      cv2.rectangle(overlay_image, (0, 0), (overlay_image.shape[1],
            overlay_image.shape[0]), (255, 255, 0), 10)
39
40      M = cv2.getPerspectiveTransform(pts_2, pts_1)
41      dst_image = cv2.warpPerspective(overlay_image, M,
            (image.shape[1], image.shape[0]))
42
43      # Create the mask:
44      dst_image_gray = cv2.cvtColor(dst_image, cv2.COLOR_BGR2GRAY)
45      ret, mask = cv2.threshold(dst_image_gray, 0, 255,
            cv2.THRESH_BINARY_INV)
46
47
48      image_masked = cv2.bitwise_and(image, image, mask=mask)
49
50      result = cv2.add(dst_image, image_masked)
51      return result
52
53
```

Full code (4/5)

```
54 aruco_dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_7X7_250)
55
56 parameters = cv2.aruco.DetectorParameters_create()
57
58 capture = cv2.VideoCapture(0)
59
60 while True:
61     ret, frame = capture.read()
62     gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
63
64     corners, ids, rejectedImgPoints =
           cv2.aruco.detectMarkers(gray_frame, aruco_dictionary,
           parameters=parameters)
65
66     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=corners, ids=ids, borderColor=(0, 255, 0))
67
68     frame = cv2.aruco.drawDetectedMarkers(image=frame,
           corners=rejectedImgPoints, borderColor=(0, 0, 255))
69
70     if ids is not None:
71         rvecs, tvecs, _ =
               cv2.aruco.estimatePoseSingleMarkers(corners, 1,
               cameraMatrix, distCoeffs)
72
```

Full code (5/5)

```
73      for rvec, tvec in zip(rvecs, tvecs):
74          desired_points = np.float32([[-1 / 2, 1 / 2, 0], [1 /
                2, 1 / 2, 0], [1 / 2, -1 / 2, 0], [-1 / 2, -1 / 2,
                0]]) * OVERLAY_SIZE_PER
75
76          # Project the points:
77          projected_desired_points, jac =
                cv2.projectPoints(desired_points, rvecs, tvecs,
                cameraMatrix, distCoeffs)
78
79          # Draw the projected points (debugging):
80          #draw_points(frame, projected_desired_points)
81          # Overlay the image:
82          frame =
                draw_augmented_overlay(projected_desired_points,
                overlay, frame)
83
84      # Display the resulting frame
85      cv2.imshow('frame', frame)
86      if cv2.waitKey(1) & 0xFF == ord('q'):
87          break
88
89  # Release everything:
90  capture.release()
91  cv2.destroyAllWindows()
```

Remove 1 line,
Add 1 new line of codes

Practice 3

- Try to run the code:

You need the file generated in the last week

Results:

You need the image file 

Textbook Reading

- Mastering OpenCV 4 with Python
  - Ch 9, Augmented Reality
    - Marker-based augmented reality
      - Camera pose estimation
      - Camera pose estimation and basic augmentation
      - Camera pose estimation and more advanced augmentation
    - p. 274 - p. 279