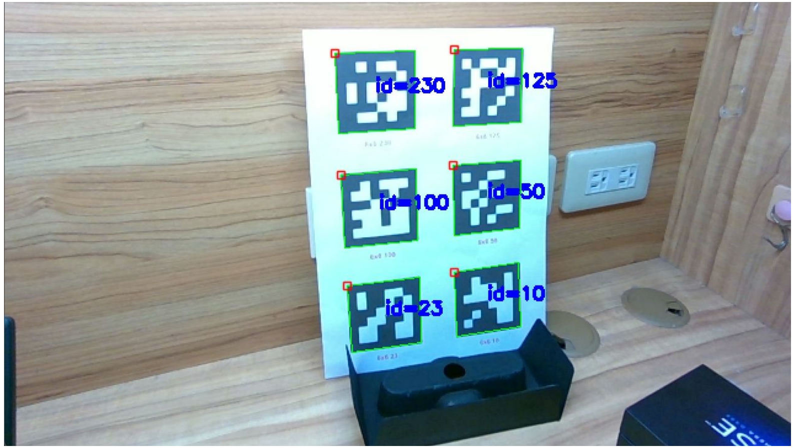
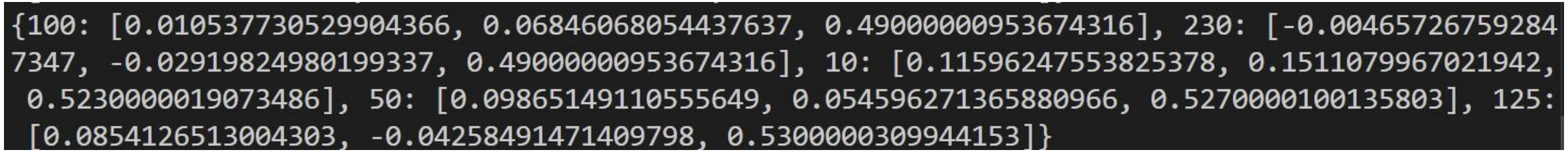
Lab 1 - Marker Detection

Realsense ArUco Detection

What we will do



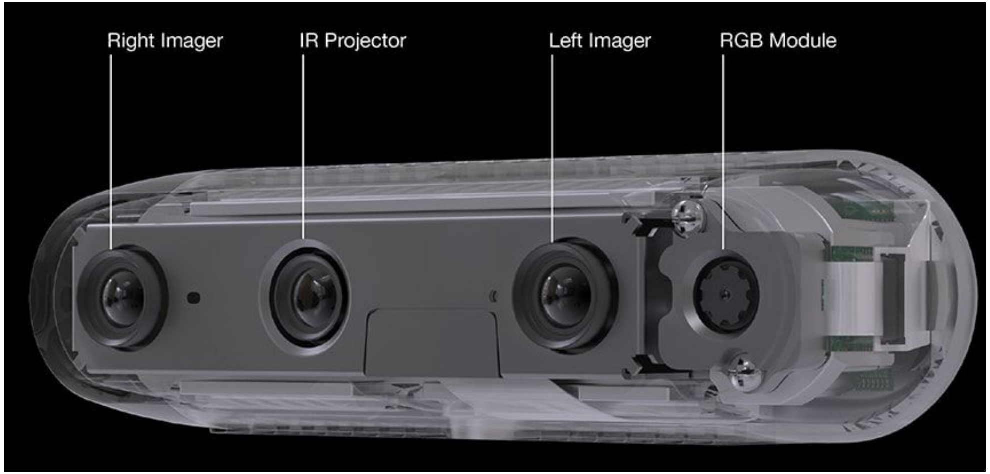


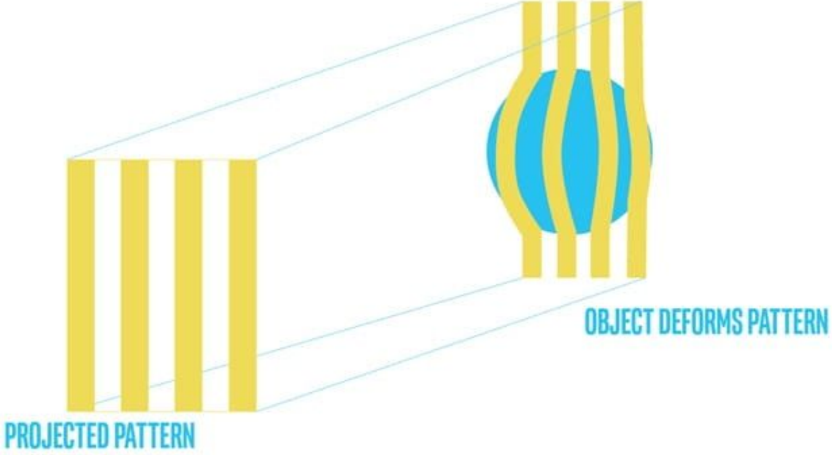
```
{100: [0.010537730529904366, 0.06846068054437637, 0.49000000953674316], 230: [-0.00465726759284
7347, -0.02919824980199337, 0.49000000953674316], 10: [0.11596247553825378, 0.1511079967021942,
 0.5230000019073486], 50: [0.09865149110555649, 0.054596271365880966, 0.5270000100135803], 125:
 [0.0854126513004303, -0.04258491471409798, 0.5300000309944153]}
```

Introduction

Intel® RealSense™ Depth Camera D455

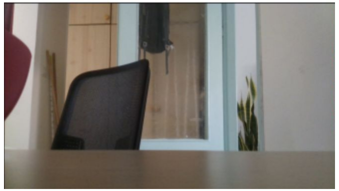
- Ideal Range: 60 cm ~ 6m
- Depth Accuracy: <2% error at 4 m
- Captures RGB image and depth

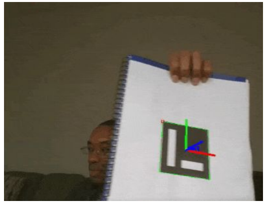




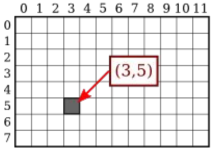
Marker Detection

(1) RGB Image



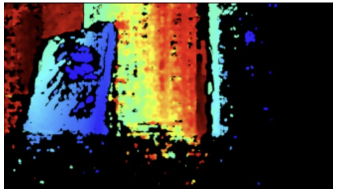


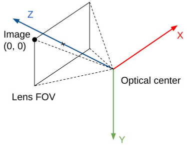
(2) Find ArUco using OpenCV functions



(3) Obtain ArUco pixel coordinates

(4) Depth Image



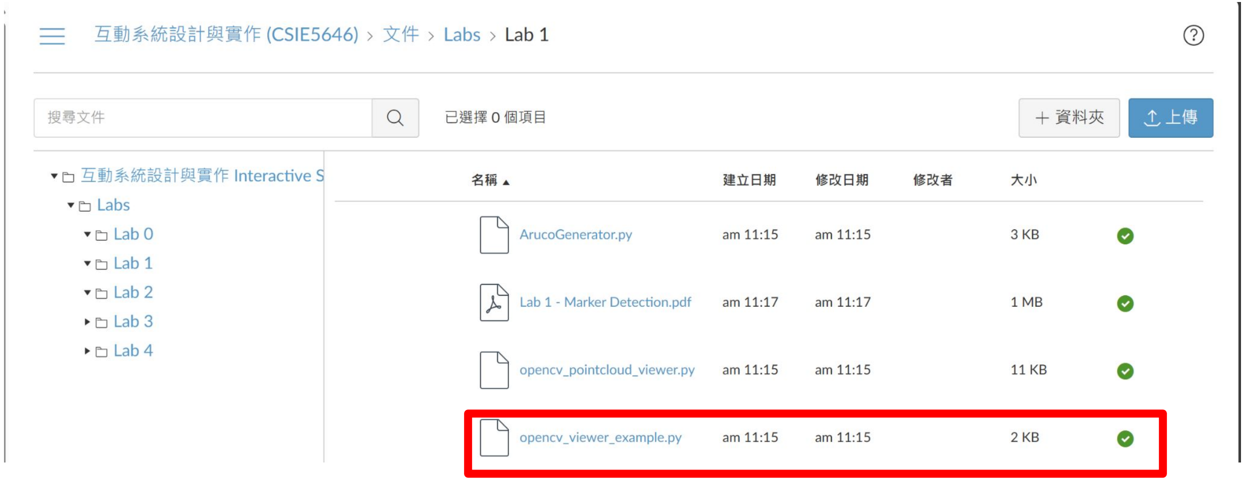


(5) Find ArUco 3D coordinates

Realsense Sample Code

RGB and Depth Streaming

- How to setup Realsense RGB and Depth Streaming with OpenCV


互動系統設計與實作 (CSIE5646) › 文件 › Labs › Lab 1
搜尋文件
已選擇 0 個項目
＋資料夾
上傳
互動系統設計與實作 Interactive S
Labs
Lab 0
Lab 1
Lab 2
Lab 3
Lab 4

名稱 ▲	建立日期	修改日期	修改者	大小	
ArucoGenerator.py	am 11:15	am 11:15		3 KB	✓
Lab 1 - Marker Detection.pdf	am 11:17	am 11:17		1 MB	✓
opencv_pointcloud_viewer.py	am 11:15	am 11:15		11 KB	✓
opencv_viewer_example.py	am 11:15	am 11:15		2 KB	✓

Enable Streaming

config.enable_stream(stream_type, width, height, format, frame_rate)

- **stream_type:** Types of data provided by RealSense devices
  - We will use depth and color
- **format:** Identifies how binary data is encoded within a frame:
  - **z16** for grayscale depth represented with 16 bits
  - **bgr8** for Blue Green Red color represented with 8 bits

Start Streaming

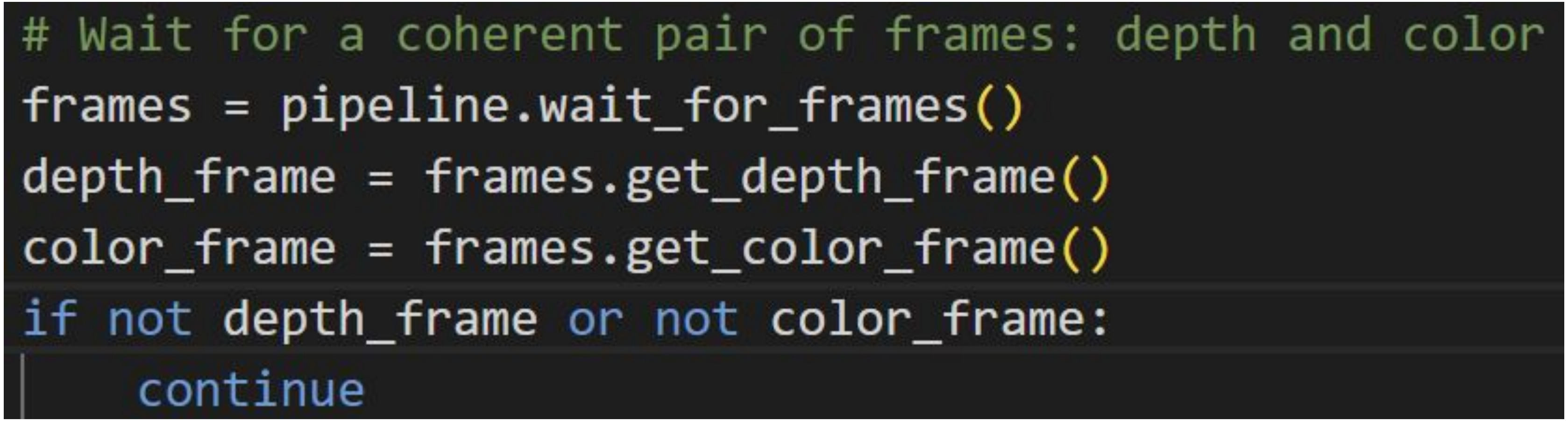
pipeline.start(config)

Start the pipeline streaming according to the configuration.

Get color and depth frame

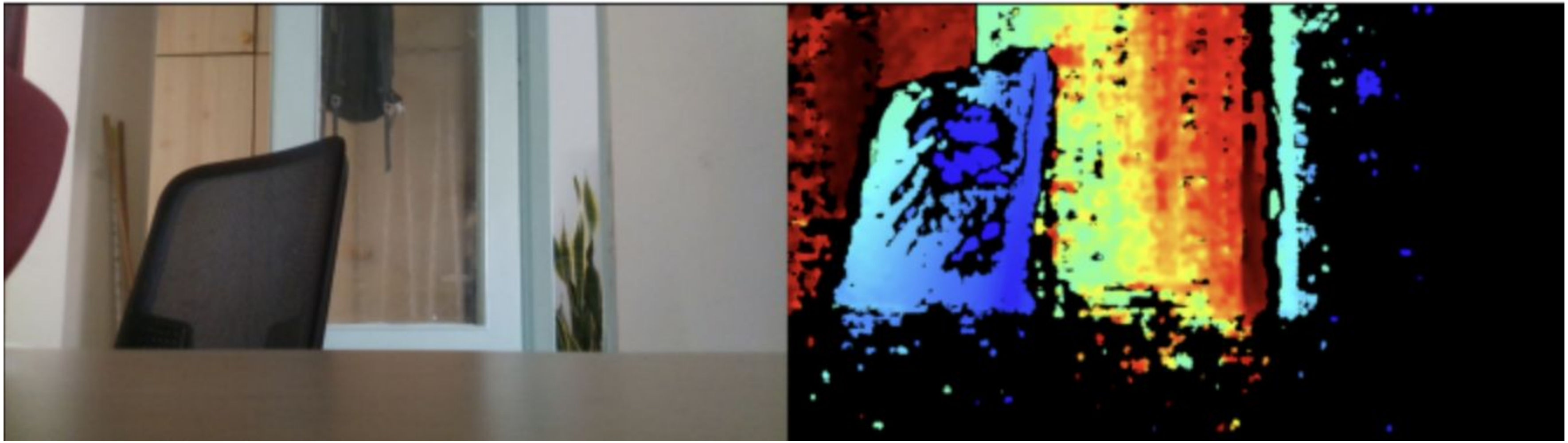
pipeline.wait_for_frames()

- Wait until a new set of frames becomes available. The frames set includes time-synchronized frames of each enabled stream in the pipeline.



```
# Wait for a coherent pair of frames: depth and color
frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()
color_frame = frames.get_color_frame()
if not depth_frame or not color_frame:
    continue
```

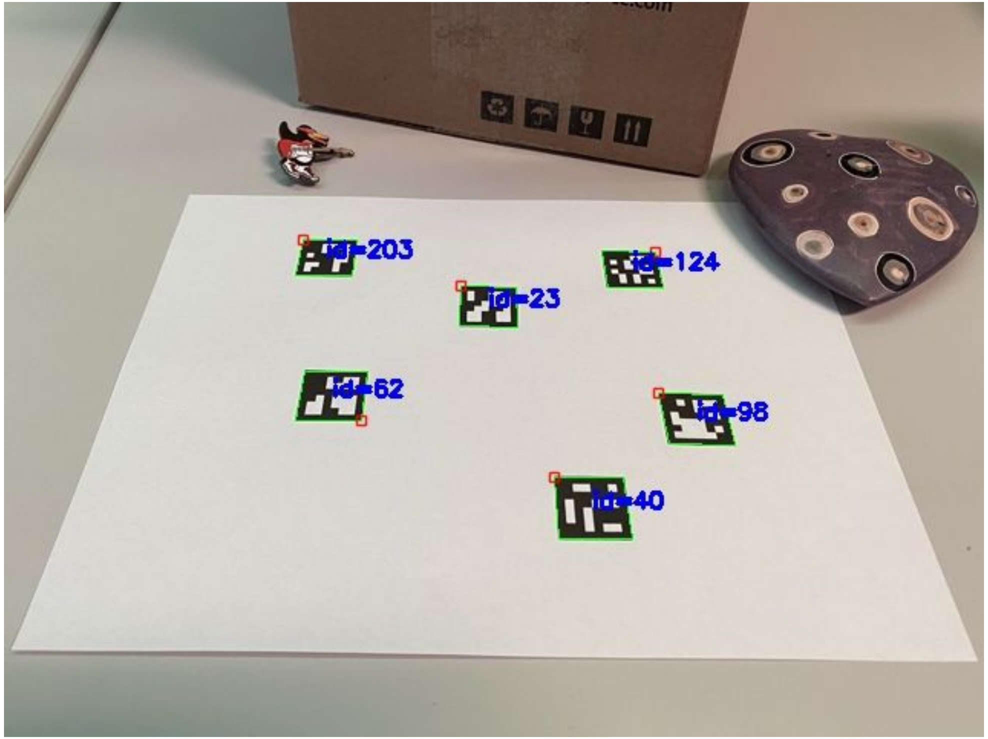
Show color and depth image through openCV



ArUco Detection

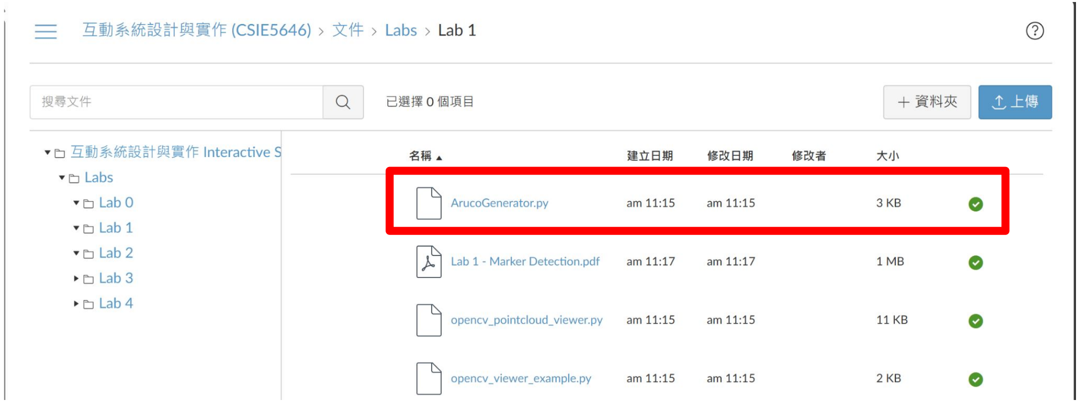
OpenCV ArUco Markers Detection

cv2.aruco



Create Markers

- Download ArucoGenerator.py
- Run command
  - python3 ArucoGenerator.py -o [output file name] -i [ArUco Id] -t [ArUco Type]
- **With this you can generate ArUco codes!**



互動系統設計與實作 (CSIE5646) › 文件 › Labs › Lab 1

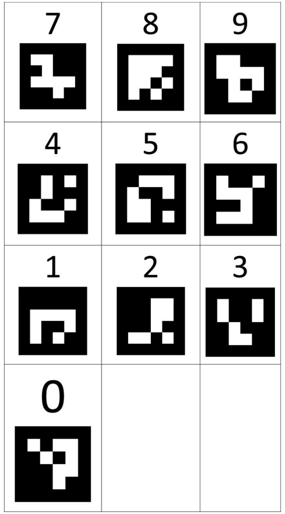
搜尋文件

已選擇 0 個項目

＋資料夾 上傳

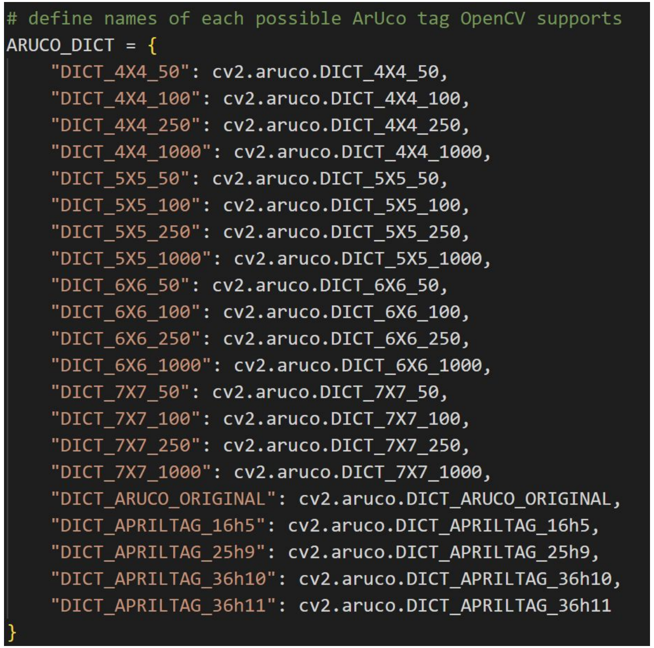
- 互動系統設計與實作 Interactive S
  - Labs
    - Lab 0
    - Lab 1
    - Lab 2
    - Lab 3
    - Lab 4

名稱 ▲	建立日期	修改日期	修改者	大小	
ArucoGenerator.py	am 11:15	am 11:15		3 KB	✓
Lab 1 - Marker Detection.pdf	am 11:17	am 11:17		1 MB	✓
opencv_pointcloud_viewer.py	am 11:15	am 11:15		11 KB	✓
opencv_viewer_example.py	am 11:15	am 11:15		2 KB	✓



Setup Detector

- **Set the type of ArUco to detect**
  - *getPredefinedDictionary(ARUCO_DICT[type])*
- **Set default detector parameters**
  - *DetectorParameters()*
- **Create detector**
  - *ArucoDetector(arucoDict, arucoParams)*



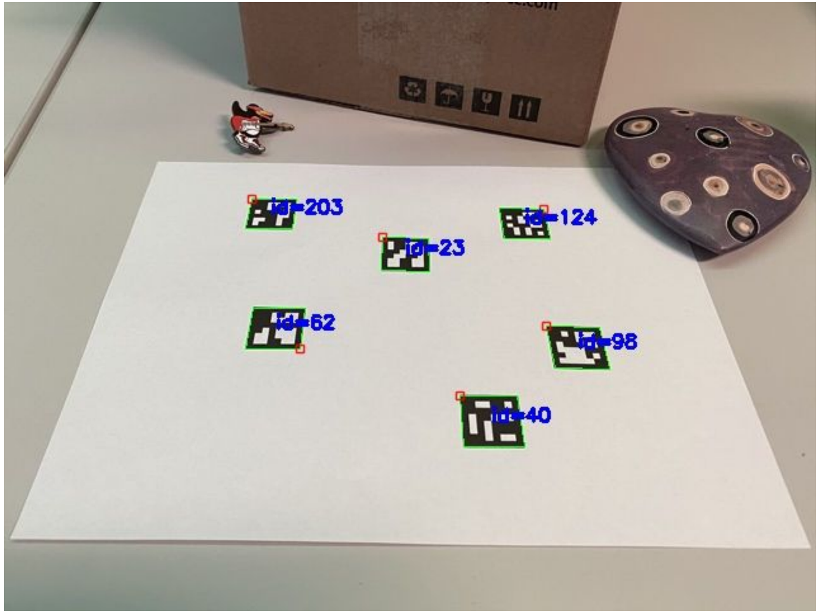
```
# define names of each possible ArUco tag OpenCV supports
ARUCO_DICT = {
    "DICT_4X4_50": cv2.aruco.DICT_4X4_50,
    "DICT_4X4_100": cv2.aruco.DICT_4X4_100,
    "DICT_4X4_250": cv2.aruco.DICT_4X4_250,
    "DICT_4X4_1000": cv2.aruco.DICT_4X4_1000,
    "DICT_5X5_50": cv2.aruco.DICT_5X5_50,
    "DICT_5X5_100": cv2.aruco.DICT_5X5_100,
    "DICT_5X5_250": cv2.aruco.DICT_5X5_250,
    "DICT_5X5_1000": cv2.aruco.DICT_5X5_1000,
    "DICT_6X6_50": cv2.aruco.DICT_6X6_50,
    "DICT_6X6_100": cv2.aruco.DICT_6X6_100,
    "DICT_6X6_250": cv2.aruco.DICT_6X6_250,
    "DICT_6X6_1000": cv2.aruco.DICT_6X6_1000,
    "DICT_7X7_50": cv2.aruco.DICT_7X7_50,
    "DICT_7X7_100": cv2.aruco.DICT_7X7_100,
    "DICT_7X7_250": cv2.aruco.DICT_7X7_250,
    "DICT_7X7_1000": cv2.aruco.DICT_7X7_1000,
    "DICT_ARUCO_ORIGINAL": cv2.aruco.DICT_ARUCO_ORIGINAL,
    "DICT_APRILTAG_16h5": cv2.aruco.DICT_APRILTAG_16h5,
    "DICT_APRILTAG_25h9": cv2.aruco.DICT_APRILTAG_25h9,
    "DICT_APRILTAG_36h10": cv2.aruco.DICT_APRILTAG_36h10,
    "DICT_APRILTAG_36h11": cv2.aruco.DICT_APRILTAG_36h11
}
```

Detect Markers

- **arucoDetecor.detectMarkers(image)**
  - Return markerCorners, markerIds, rejectedCandidates in image.
- **corners:** list of the four corners of the detected markers.
- **ids:** list of ids of each of the detected markers in **markerCorners**.
- **rejected (rejectedImgPoints):** list of shapes that were found and considered but did not contain a valid marker.

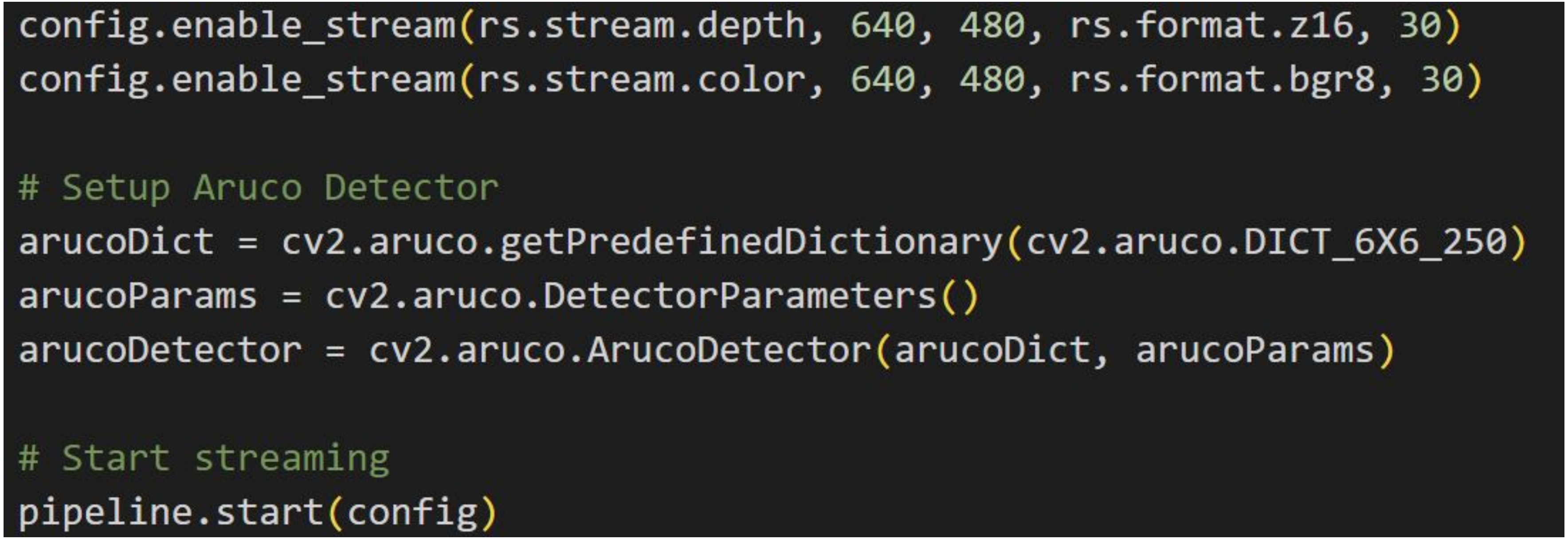
Draw Markers

- **drawDetectedMarkers(image, corners, ids)**
    - Return image with markers drawn.



Combine with RealSense

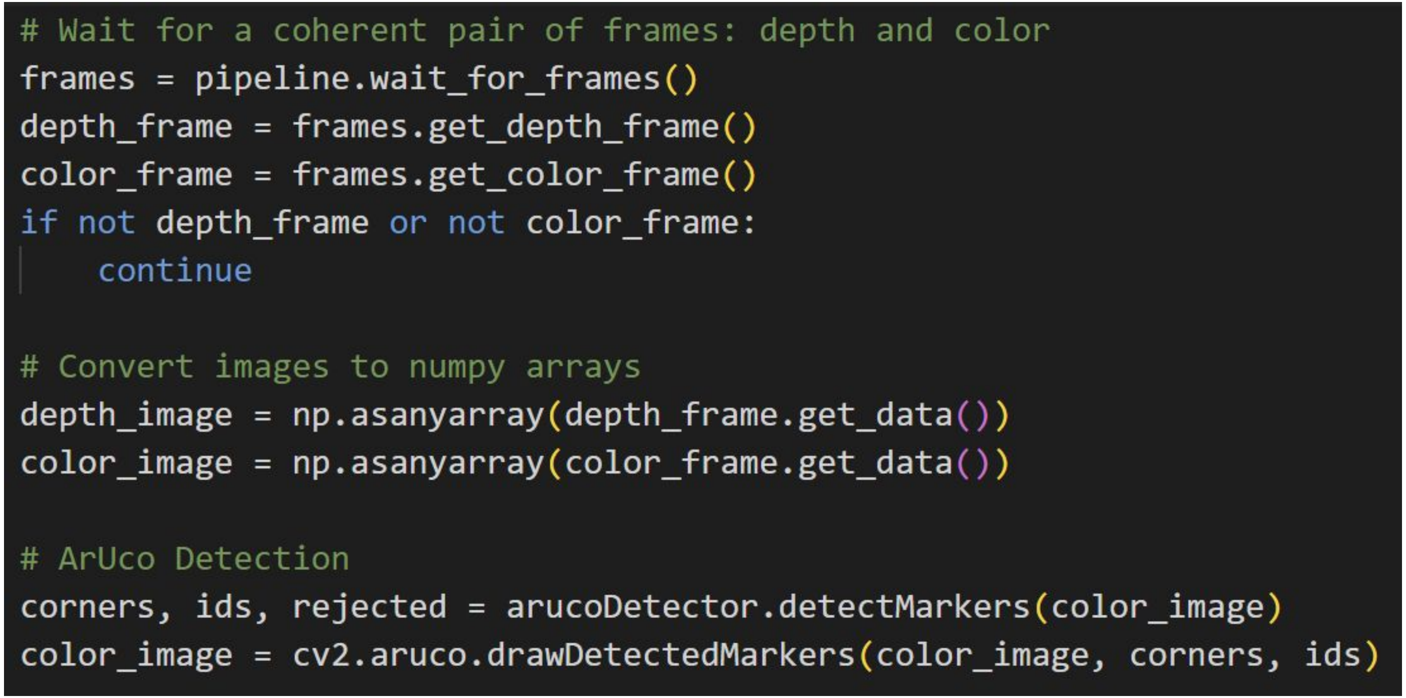
Setup detector before streaming.



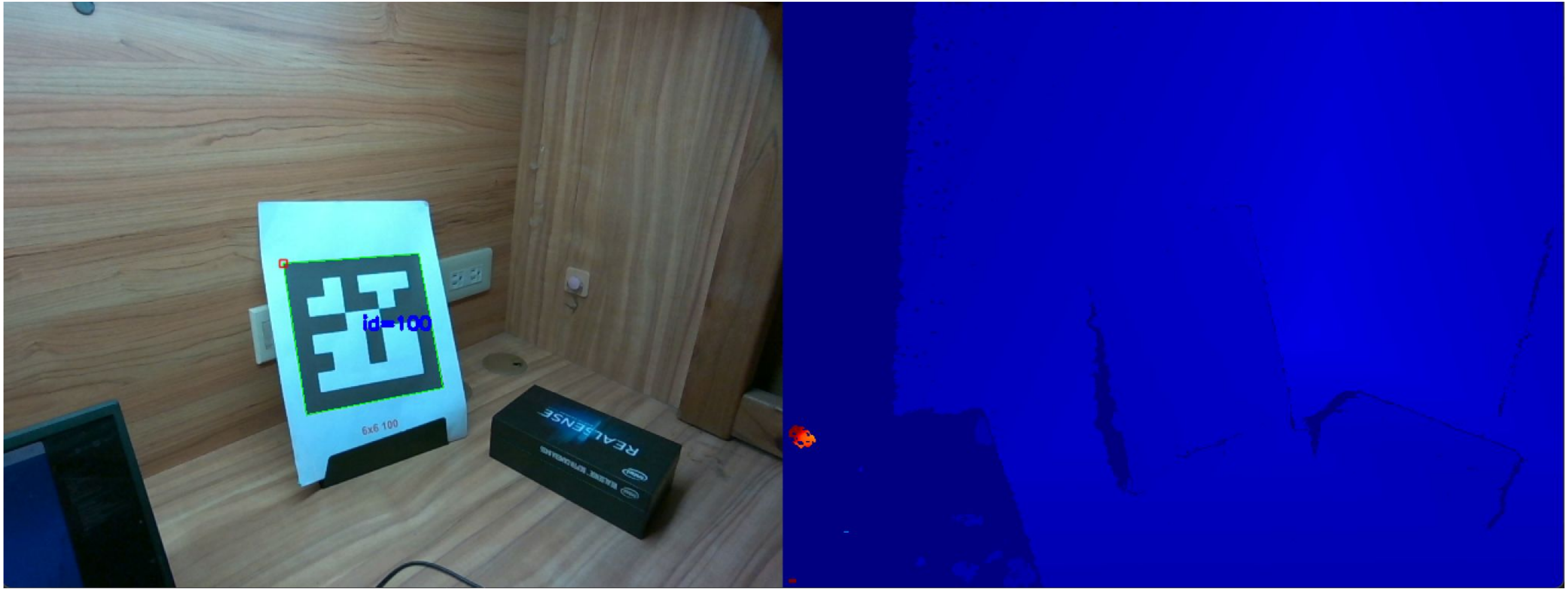
```
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

# Setup Aruco Detector
arucoDict = cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_6X6_250)
arucoParams = cv2.aruco.DetectorParameters()
arucoDetector = cv2.aruco.ArucoDetector(arucoDict, arucoParams)

# Start streaming
pipeline.start(config)
```

Combine with RealSense

Detect ArUco and draw result to each frame.

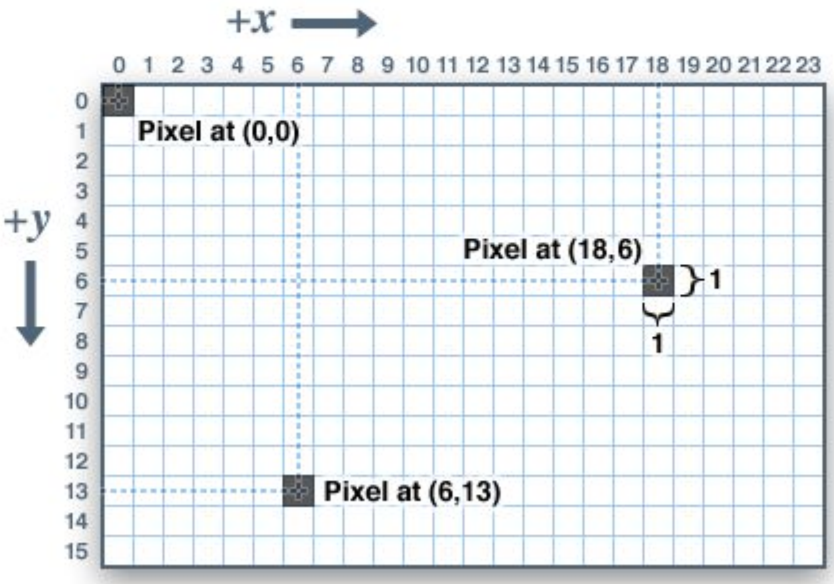


```
# Wait for a coherent pair of frames: depth and color
frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()
color_frame = frames.get_color_frame()
if not depth_frame or not color_frame:
    continue

# Convert images to numpy arrays
depth_image = np.asanyarray(depth_frame.get_data())
color_image = np.asanyarray(color_frame.get_data())

# ArUco Detection
corners, ids, rejected = arucoDetector.detectMarkers(color_image)
color_image = cv2.aruco.drawDetectedMarkers(color_image, corners, ids)
```


id=100
6x6 100
REALSENSE

To Do

What we obtained

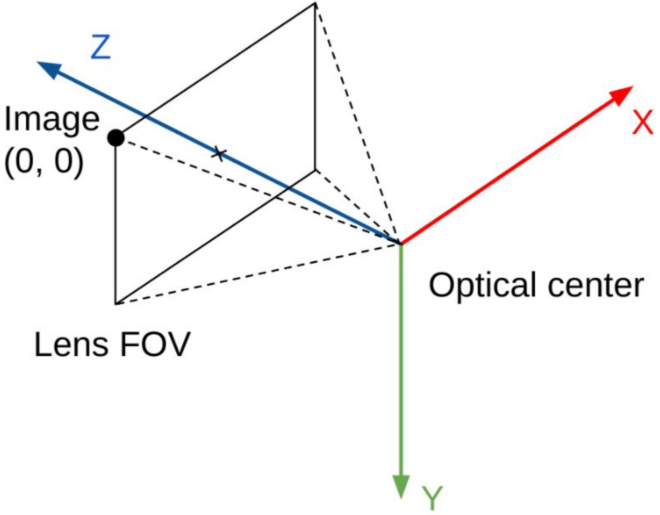


2D pixel coordinates in the frame space

(x, y)

(The origin is the top-left of the frame!)

What we want for MR applications



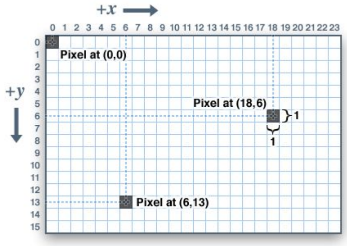
3D coordinates with respect to the camera in real-world meters

(X, Y, Z)

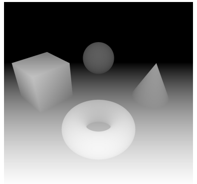
(The origin is the camera's optical center!)

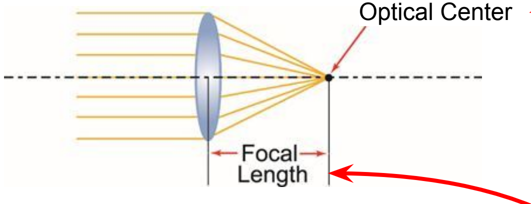
How to obtain 3D coordinates?

Using the ArUco position in the 2D coordinates as the baseline for X and Y...



... and the depth value at the ArUco position as a baseline for Z...





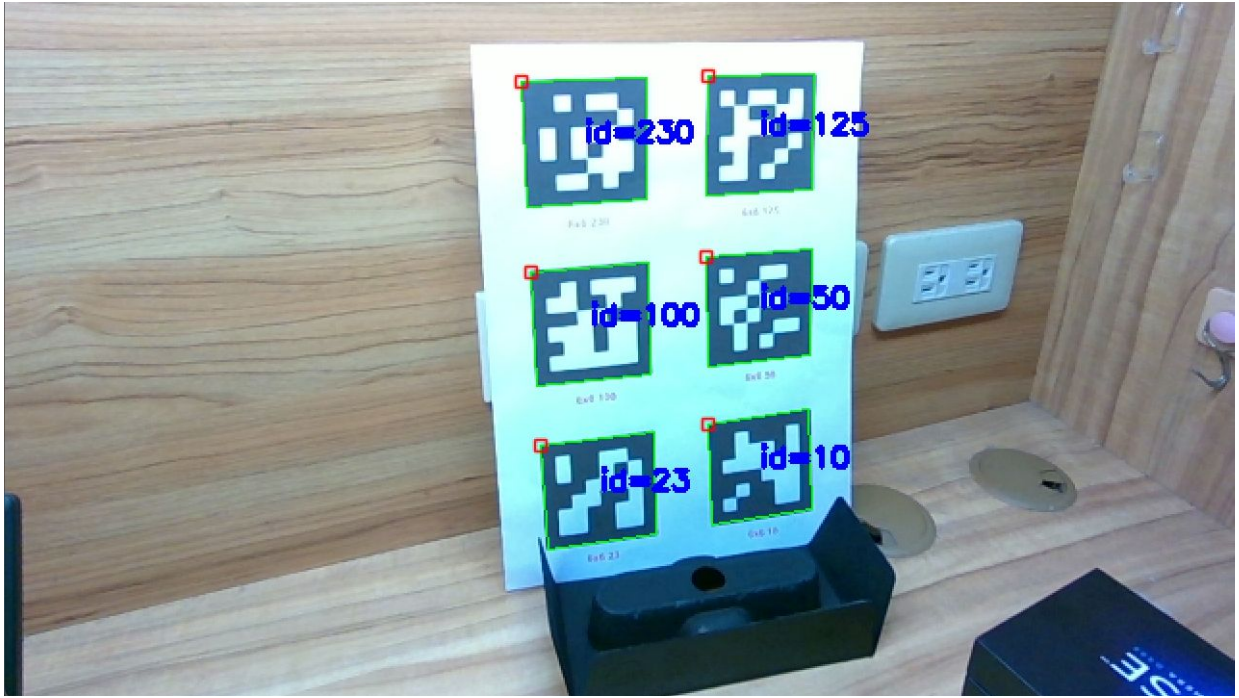
more or less around the middle of the image

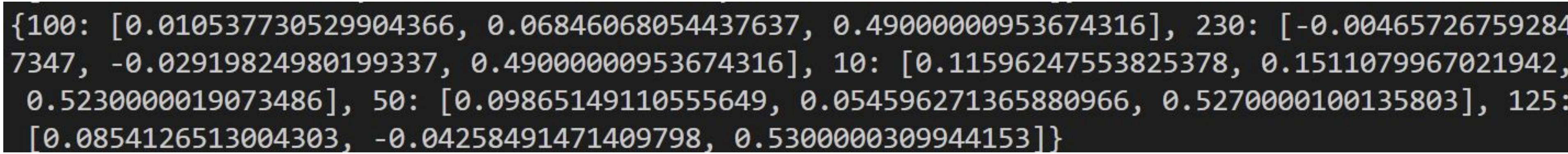
...and the camera intrinsics (optical center and focal length) to apply a proper transformation

Getting 3D coordinates

Get the 3D coordinates of the ArUco markers.

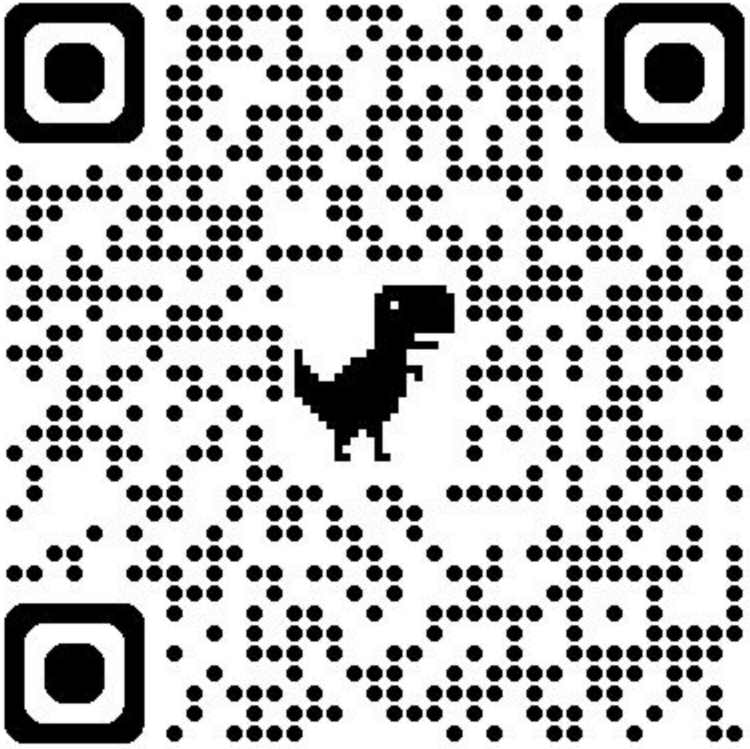
- **rs2_deproject_pixel_to_point(depth_intrinsics, [x, y], depth)**
  - Use pixel coordinates and depth in an image to get the corresponding point in 3D space.
- **depth:** depth_frame.get_distance(x, y)
  - Get the depth at pixel coordinates.
- **depth_intrinsics:** depth_frame.profile.as_video_stream_profile().intrinsics





```
{100: [0.010537730529904366, 0.06846068054437637, 0.49000000953674316], 230: [-0.00465726759284
7347, -0.02919824980199337, 0.49000000953674316], 10: [0.11596247553825378, 0.1511079967021942,
 0.5230000019073486], 50: [0.09865149110555649, 0.054596271365880966, 0.5270000100135803], 125:
 [0.0854126513004303, -0.04258491471409798, 0.5300000309944153]}
```

Feedback !



https://forms.gle/9ukjeotnWWAPTdLH9