Documentation for Assignment 2 – Intro to Open CV

Eric Sobczak

Answers to Assignment Questions

(1) How is an OpenCV image stored in memory?

OpenCV stores image data within an array. This array can store many different types of data, but we are mainly storing either luminance values for grayscale or RGB pixels for these assignments.

(2) What does image.shape return? Here image is the variable name that stores the image you are referring to.

Image.shape returns the dimensions of the array holding the image which turns out to be the height, width, and number of color channels.

(3) What do the parameters, fx and fy refer to in cv2.resize?

The fx and fy parameters in cv2.resize refer to scale factors along the horizontal and vertical axis.

(4) What would happened if, cols/2, and rows/2 in the following function were changed to cols/4 and rows/4? M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)

The center of rotation of the image would go from the center to the top left quadrant.

(5) What function must be used after cv2.getRotationsMatrix2D to actually perform the rotation? What parameters do you pass to this function?

You must run warpAffine on the image to actually apply the rotation. This function accepts the image to modify, the destination of the transform, the matrix to apply, size of the output image, and flags, how to extrapolate pixels at the border, and a value if a constant border, as parameters.

(6) How does Aruco marker detection work? What are the steps taken by aruco.detectMarkers() to detect an Aurco marker?

Aruco marker detection works by finding markers in an image, and comparing them to a dictionary of existing markers. The aruco.detectMarker function does this by running adaptive thresholding to detect rectangular features. The corners are then extracted from the features and used to correct any perspective warp of the marker. The marker can then be compared to the dictionary by overlaying a grid over the detected marker. If nothing in the dictionary matches, it is rejected.

(7) What is the dimension of the arrays ids and corners from the output of detectMarkers() when it sees 2 aruco markers?

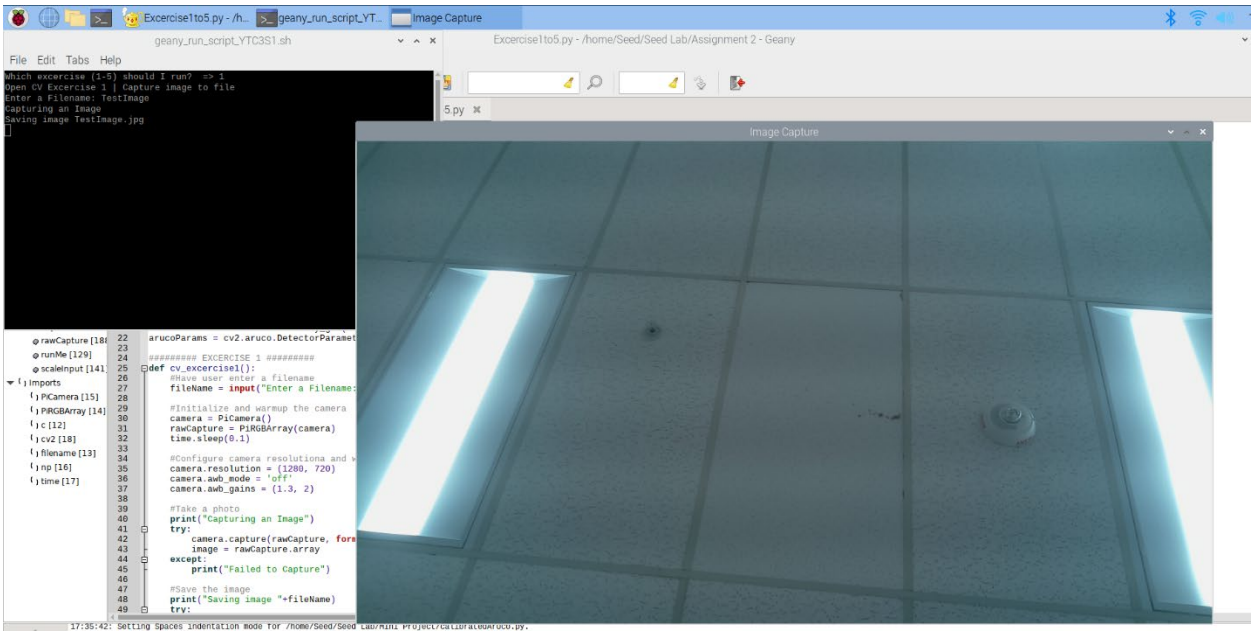
When two markers are detected, the dimension of the ID array will be 1x2 and the dimension of the corner array will be 1x2x4x2. ArUco packages everything within an array, so that the starting 1x, and then there are two markers, so the following 2x. Corners then have points (4) and x/y.

(8) detectMarkers() has adjustable parameters. From aruco.detectMarkers() you can click on this parameter to get more detail. What parameter determines the minimum perimeter for a marker contour to be detected? How does this affect the size of markers that can be detected? I would encourage you to try to change this parameter on an example image to verify your answer. Note that in python, the code parameters = aruco.DetectorParameters_create() creates the parameter object, and elements of the object can be changed using the dot notation, e.g. parameters.adaptiveThreshWinSizeMin = 1. You can get a list of all attributes of an object via dir(parameters).

The parameter **minMarkerPerimeterRate** determines the minimum perimeter for a marker contour to be detected. It is a double that scales the maximum dimension of the input image. A lower rate will tell the Aruco detection to consider contours with lower perimeters. While this does mean smaller markers can possibly be detected, it has a huge performance penalty as it considers jargon contours.

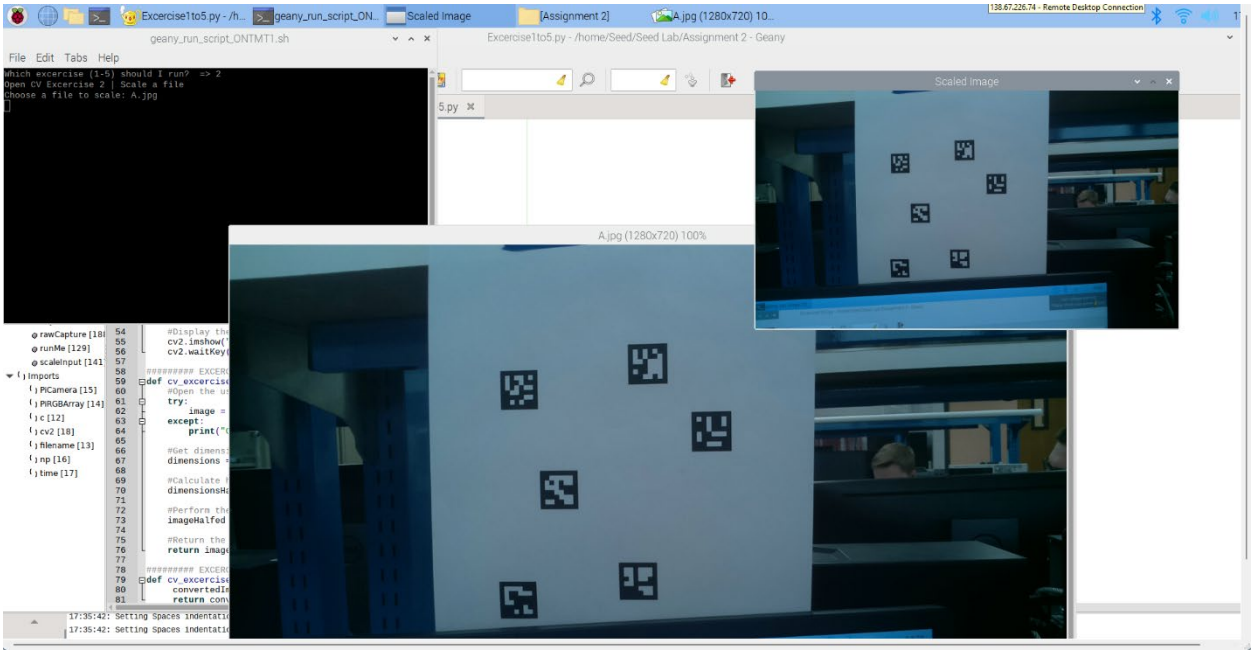
Example Execution

Exercise 1



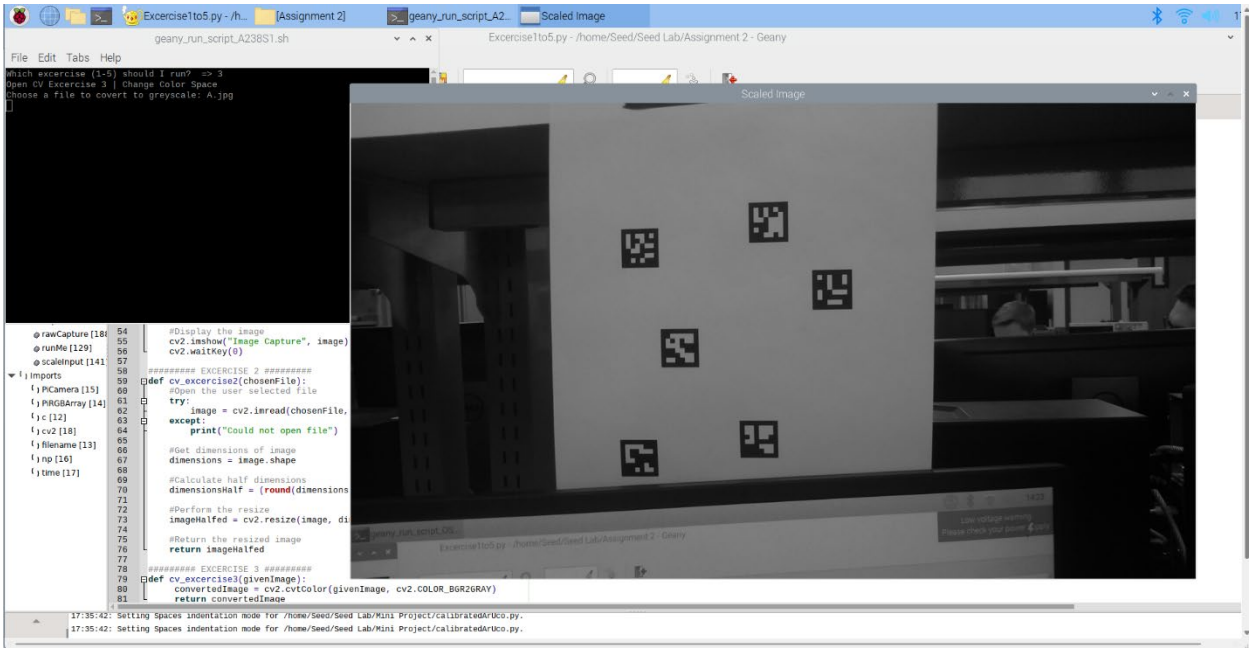
The above image demonstrates asking the user for a file name, taking and storing a photo, and then displaying that photo.

Exercise 2



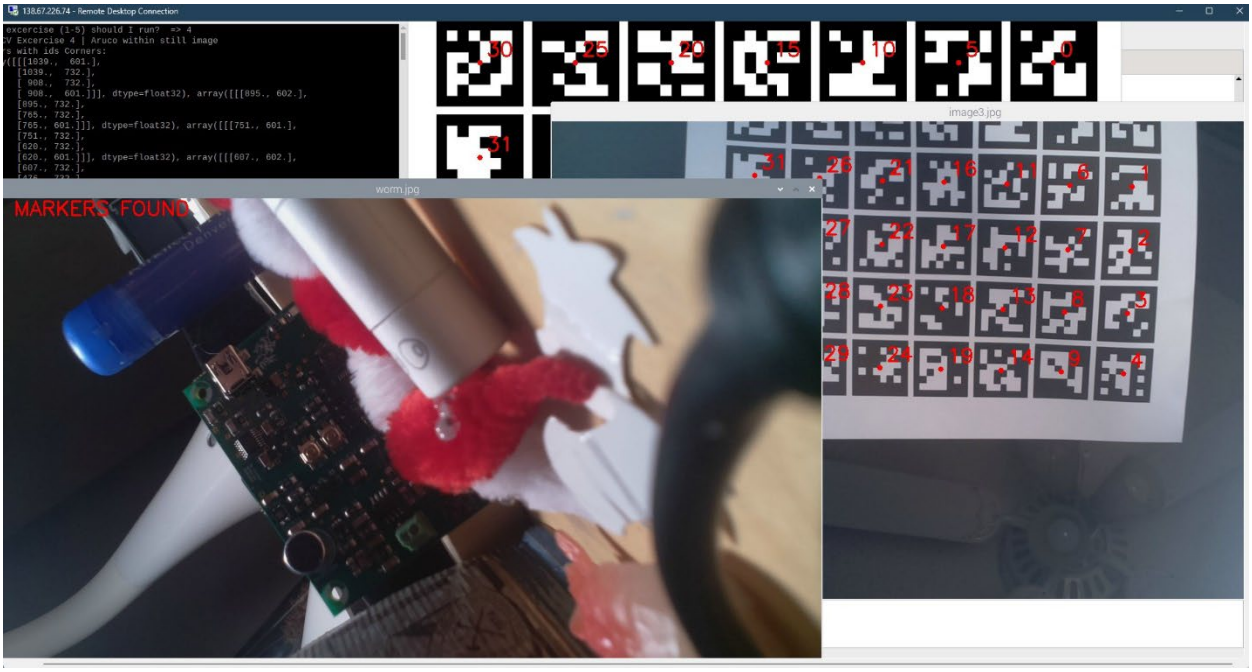
The above image demonstrates taking an original image (A) and resizing it to half its size.

Exercise 3



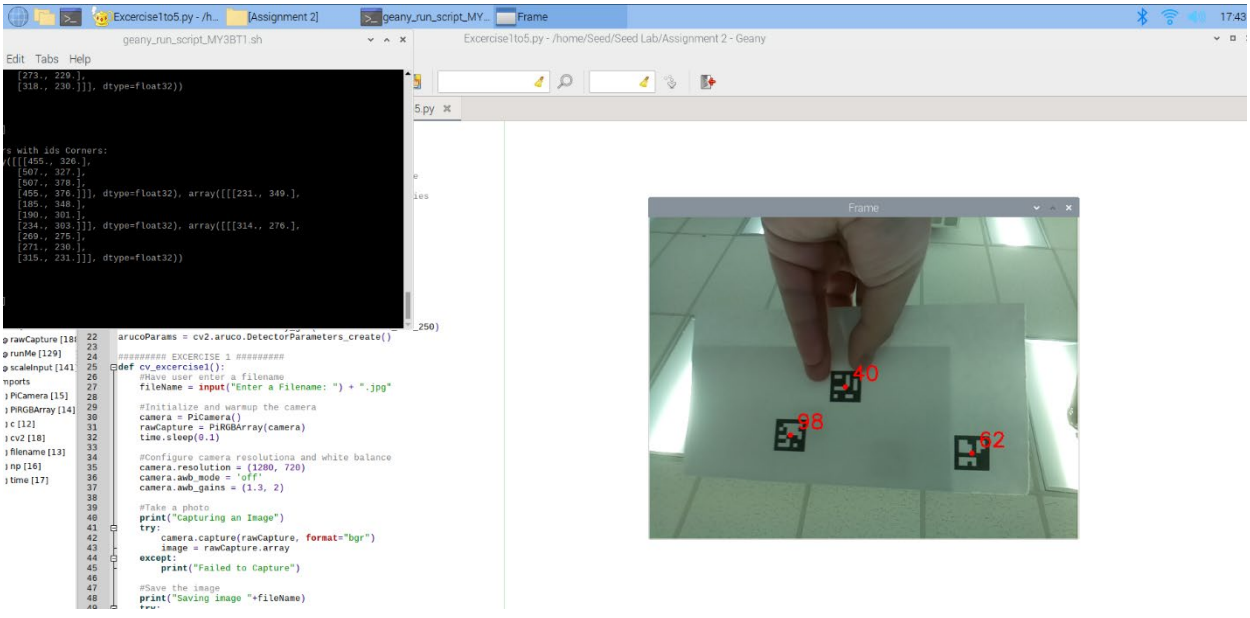
The above image demonstrates taking a stored image and switching it to grayscale.

Exercise 4



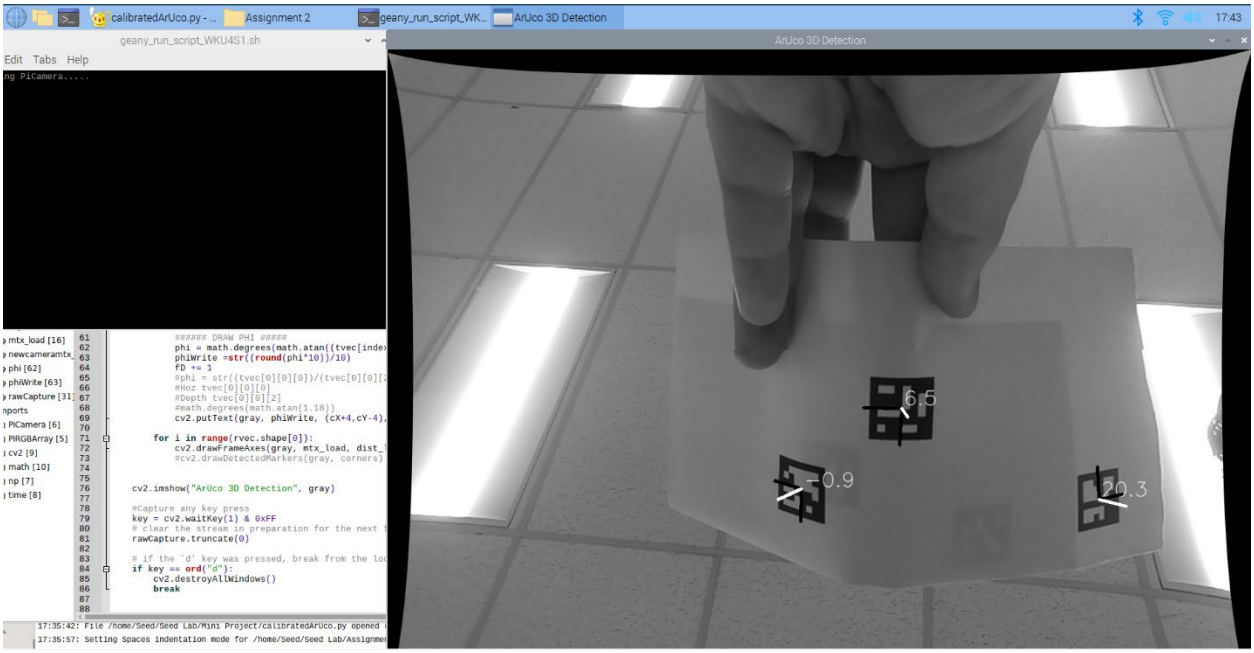
The above image demonstrates detecting ArUco markers in a series of photos

Exercise 5



The above image shows a ArUco markers being detected in a live feed.

Exercise 6



The above image shows calculating the angular offset of ArUco markers. This script uses uses distortion correction get an accurate location of the markers.

Code for Exercise 1-5:

```
#   Eric Sobczak
#   9/6/2022
#   SEED Lab Python Assignment 2, Exercises 1-5

# The following code run through exercises 1 through 5.
# Select the exercise to run by entering a value on the
# keyboard and hitting enter. All these exercises use
# Open CV to process the image, and the PICamera libraries
# to capture the image. Open CV is also used as File
# management, keyboard capture, and image preview.

from calendar import c
from fileinput import filename
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time
```

```
import cv2

#Constants for ArUco detection
arucoDict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
arucoParams = cv2.aruco.DetectorParameters_create()

######### EXCERCISE 1 #########
def cv_excercise1():
    #Have user enter a filename
    fileName = input("Enter a Filename: ") + ".jpg"

    #Initialize and warmup the camera
    camera = PiCamera()
    rawCapture = PiRGBArray(camera)
    time.sleep(0.1)

    #Configure camera resolutiona and white balance
    camera.resolution = (1280, 720)
    camera.awb_mode = 'off'
    camera.awb_gains = (1.3, 2)

    #Take a photo
    print("Capturing an Image")
    try:
        camera.capture(rawCapture, format="bgr")
        image = rawCapture.array
    except:
        print("Failed to Capture")

    #Save the image
    print("Saving image "+fileName)
    try:
        cv2.imwrite(fileName, image)
    except:
        print("Couldn't save "+fileName)

    #Display the image
    cv2.imshow("Image Capture", image)
    cv2.waitKey(0)

######### EXCERCISE 2 #########
def cv_excercise2(chosenFile):
    #Open the user selected file
    try:
        image = cv2.imread(chosenFile, cv2.IMREAD_COLOR)
```

```
    except:
        print("Could not open file")

    #Get dimensions of image
    dimensions = image.shape

    #Calculate half dimensions
    dimensionsHalf = (round(dimensions[1]/2),round(dimensions[0]/2))

    #Perform the resize
    imageHalfed = cv2.resize(image, dimensionsHalf, interpolation =
cv2.INTER_AREA)

    #Return the resized image
    return imageHalfed

######### EXCERCISE 3 #########
def cv_excercise3(givenImage):
     convertedImage = cv2.cvtColor(givenImage, cv2.COLOR_BGR2GRAY)
     return convertedImage

######### EXCERCISE 4 & 5 #########
def cv_excercise45(givenImage, fileName):
    #Run ArUco Detection
    (corners, ids, rejected) = cv2.aruco.detectMarkers(cv2.cvtColor(givenImage,
cv2.COLOR_BGR2GRAY), arucoDict, parameters=arucoParams)

    #Check if any marker was found
    if len(corners) > 0:
        #Start print statement
        print("Markers with ids", end=" ")

        #Loop through all the markers
        for index, cornerInfo in enumerate(corners):
            #Print corner id
            print(ids[index][0], end=", ")

            #Calculate center of marker
            cX = int((cornerInfo[0][0][0] + cornerInfo[0][2][0]) / 2.0)
            cY = int((cornerInfo[0][0][1] + cornerInfo[0][2][1]) / 2.0)

            #Add dot to center of marker image
            cv2.circle(givenImage, (cX, cY), 4, (0, 0, 255), -1)

            #Write the ID of the marker on the image
```

```
            cv2.putText(givenImage, str(ids[index][0]), (cX+10, cY-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        #End Print Statement
        print("\b\b were detected in " + fileName + "\n")

    else:
        #No DICT_6X6_250 markers were detected in the image
        print("No markers detected in " + fileName + "\n")

        #Write on image
        cv2.putText(givenImage, "NO MARKERS FOUND", (0, 25),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        #cv2.putText(givenImage, "NO MARKERS DETECTED", (3, 28),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    #Return image
    return givenImage


#Initialization
runMe = input("Which excercise (1-5) should I run?  => ")

#Excercise 1
if ("1"==runMe):
    print("Open CV Excercise 1 | Capture image to file")
    cv_excercise1()


#Excercise 2
if ("2"==runMe):
    print("Open CV Excercise 2 | Scale a file")
    #Get a file name from user
    scaleInput = input("Choose a file to scale: ")
    #Scale and show the image
    cv2.imshow("Scaled Image", cv_excercise2(scaleInput))
    cv2.waitKey(0)


#Excercise 3
if ("3"==runMe):
    print("Open CV Excercise 3 | Change Color Space")
    #Get a file to work with
    fileToGrey = input("Choose a file to covert to greyscale: ")
    #Load that file
```

```
    try:
        imageToGrey = cv2.imread(fileToGrey, cv2.IMREAD_COLOR)
    except:
        print("Could not open file")
    #Send image to function and process it, then show it
    cv2.imshow("Scaled Image", cv_excercise3(imageToGrey))
    cv2.waitKey(0)


#Excercise 4
if ("4"==runMe):
    print("Open CV Excercise 4 | Aruco within still image")

    #Files to loop through
    allFiles = ["image.jpg", "image2.jpg", "image3.jpg"]
    #Loop through each file
    for markerFile in allFiles:
        #Open file
        try:
            detectInMe = cv2.imread(markerFile, cv2.IMREAD_COLOR)
        except:
            print("Could not open file")

        #Perform ArUco Detection and show the image
        cv2.imshow(markerFile, cv_excercise45(detectInMe, markerFile))
        cv2.waitKey(0)


#Excercise 5
if ("5"==runMe):
    print("Open CV Excercise 5 | Aruco within video")
    #Setup Camera
    camera = PiCamera()
    camera.resolution = (640, 480)
    camera.framerate = 32
    rawCapture = PiRGBArray(camera, size=(640, 480))

    # allow the camera to warmup
    time.sleep(0.1)

    # capture frames from the camera
    for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    # grab the raw NumPy array representing the image, then initialize the
timestamp and occupied/unoccupied text
```

```
        imageVideo = frame.array

        #Perform the ArUco detection and display the video
        cv2.imshow("Frame", cv_excercise45(imageVideo, "Video"))
        key = cv2.waitKey(1) & 0xFF

        # clear the stream in preparation for the next frame
        rawCapture.truncate(0)

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break



#Excercise 1
#   Getting Started with Picamera
#       https://pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-
with-opencv-and-python/
#   Configure White Balance
#       https://raspberrypi.stackexchange.com/questions/22975/custom-white-
balancing-with-picamera
#   take_picture.py example file on Canvas
#   Cool info on camera config for future
#       https://python.hotexamples.com/examples/picamera/PiCamera/awb_gains/pytho
n-picamera-awb_gains-method-examples.html
#
#Excercise 2
#   Opening an Image file
#       https://www.geeksforgeeks.org/reading-image-opencv-using-python/
#   Using the resize function
#       https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/
#   Getting image parameters
#       https://www.tutorialkart.com/opencv/python/opencv-python-get-image-size/
#
#Excercise 3
#   How to use the cv2.cvtcolor
#       https://www.geeksforgeeks.org/python-opencv-cv2-cvtcolor-method/
#
#Excercise 4
#   Great tutorial on ArUco detection
#       https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-
and-python/
#   Gave me a stronger understanding of how ArUco markers have types
```

```
#       https://pyimagesearch.com/2020/12/28/determining-aruco-marker-type-with-opencv-and-python/
#
# No additional resources used for excercsie 5
```

Code for Exercise 6 - Calibrator:

```
#   Eric Sobczak
#   9/6/2022
#   SEED Lab Python Assignment 2+

# Performing camera calibration is vital to accurately find the location of
# ArUco markers in 3D space. This script generates a .yaml calibration file
# that contains calibration data for any PiCamera. This scripts begins by
# running a photo application to collect calibration data. Simply aim the
# camera at a checkerboard and press "c" multiple times from different angles.
# After atleast 6 images, press the d key. The program will look for the
# checkboard in each image and note the locations of all the corners.
# Afterwards opencv camera calibration script will run and generate a new
# calibration. This is save to a yaml file, and then reloaded. A preview of
# the calibration results is then shown to the user.


from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time
import cv2

# Set the parameters for finding sub-pixel corners, max 30 cycles, max error tolerance 0.001
subPixelCriteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
#Set Size of Chessboard
gridW = 9   # 10 - 1
gridH = 6   # 7  - 1
checkerSpacing = 18.1 #Size of checkerboard points in mm
#Define the checkerboard in world coordanites
checkerPoints = np.zeros((gridW*gridH, 3 ), np.float32)  #Create numpy array full of zeros for all checkerboard points
checkerPoints[:,: 2 ] = np .mgrid[ 0 :gridW, 0 :gridH].T.reshape( -1 , 2 )  #Fill the grid with coordainted of points
checkerPoints = checkerPoints * checkerSpacing #Multiply out grid by spacing

#Create holders 3D and 2D Coordanites
```

```
worldCordChess = [] # 3D points in the world coordinate system
imgCordChess = []   # 2D points in the image plane


#Capture images for calibration
camera = PiCamera()
camera.resolution = (1296, 976)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size=(1296, 976))

# allow the camera to warmup
print("Starting PiCamera.....")
time.sleep(0.1)

#Idex for taking images
photoIndex = 0

print("Press 'c' to take a photo")
print("Press 'd' to finish and calibrate")

#Capture photos for calibration
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    imageVideo = frame.array

    #Display the video
    cv2.imshow("Frame", imageVideo)
    key = cv2.waitKey(1) & 0xFF

    # clear the stream in preparation for the next frame
    rawCapture.truncate(0)

    # if the `d' key was pressed, break from the loop
    if key == ord("d"):
        cv2.destroyAllWindows()
        break

    # if the 'c' key was pressed, capture an image
    if key == ord("c"):
        fileName = "calibrationImage" + str(photoIndex) + ".jpg"
        cv2.imwrite(fileName, imageVideo)
        print ("Saving " + fileName)
        photoIndex += 1

#Loop through all the captured photos and perform calibration steps
```

```
calibrateIndex = 0
while calibrateIndex < photoIndex:
    #Load from captured photo
    fileName = "calibrationImage" + str(calibrateIndex) + ".jpg"
    img = cv2.imread(fileName)

    #Get image parameters
    imgHeight, imgWidth = img.shape[0], img.shape[1]

    #Get grayscale image
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    #Perform scan for checkerbaord
    ret, corners = cv2.findChessboardCorners(gray, (gridW,gridH),None)

    if ret == True:
        print("Checkerboard found in " + fileName + ", running subpixel
correction")
        #Use subpixel constraints to get more accurate corners
        cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),subPixelCriteria)

        #Add corners to lists from earlier
        worldCordChess.append(checkerPoints) #? - Im still confused why we are
doing this as checkerPoints has no data from the image
        imgCordChess.append(corners)

        #Draw and display the corners on the chessbaord
        cv2.drawChessboardCorners(img, (gridW,gridH), corners, ret)
        cv2.namedWindow(('CheckerView - ' + fileName), cv2.WINDOW_NORMAL)
        cv2.resizeWindow(('CheckerView - ' + fileName), 640, 480)
        cv2.imshow(('CheckerView - ' + fileName),img)
        cv2.waitKey(2000)

    else:
        print("No checkerboard found in " + fileName)

    calibrateIndex += 1
cv2.destroyAllWindows()

#Generate the calibration
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(worldCordChess, imgCordChess,
gray.shape[::-1], None, None)
#Generate a new camera matrix to account for a different frame size with the
distortion
```

```
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, (imgWidth,
imgHeight), 1, (imgWidth, imgHeight))

#Save calibration to file
calibrationFile = "calibration_test2.yaml"
cv_file=cv2.FileStorage(calibrationFile, cv2.FILE_STORAGE_WRITE)
cv_file.write("camera_matrix", mtx)
cv_file.write("dist_coeff", dist)
cv_file.write("new_camera_matrix", newcameramtx)
cv_file.release()

#Load calibration file
calibrationFileLoad = "calibration_test.yaml"
cv_file_load = cv2.FileStorage(calibrationFileLoad, cv2.FILE_STORAGE_READ)
mtx_load = cv_file_load.getNode("camera_matrix").mat()
dist_load = cv_file_load.getNode("dist_coeff").mat()
newcameramtx_load = cv_file_load.getNode("new_camera_matrix").mat()
cv_file_load.release()

#Start video capture again
print("Running preview of distortion correction, press 'd' to exit")
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    imageVideo = frame.array

    dst1 = cv2.undistort(imageVideo, mtx_load, dist_load, None,
newcameramtx_load)

    #Perform the ArUco detection and display the video
    cv2.imshow("Corrected", dst1)
    cv2.imshow("Uncorrected", imageVideo)
    key = cv2.waitKey(1) & 0xFF

    # clear the stream in preparation for the next frame
    rawCapture.truncate(0)

    # if the `d' key was pressed, break from the loop
    if key == ord("d"):
        cv2.destroyAllWindows()
        break

# The following tutorial provide info on how to perform these calibration steps.
# I also utilized the resources from the other documents
#https://stackoverflow.com/questions/39432322/what-does-the-
getoptimalnewcameramatrix-do-in-opencv
```

```
#https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
```

Code for Exercise 6 – Angle Detection:

```
#   Eric Sobczak
#   9/12/2022
#   SEED Lab Python Assignment 2, Excercise 6

# The following script utilizes the calibration file generated in
# the other python script in order accuratley detect ArUco markers
# in 3D space, and provide their angular horizontal offset. The
# .yaml file is loaded, and distortion correction is applied to the
# image. Then ArUco markers are found, and pose estimation is used
# to find their location in 3D space. Trigonometry is then used to
# calculate the agnular offset from the camera based on the world
# coordanite.

from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time
import cv2
import math


#Load calibration file
calibrationFileLoad = "calibration_test.yaml"
cv_file_load = cv2.FileStorage(calibrationFileLoad, cv2.FILE_STORAGE_READ)
mtx_load = cv_file_load.getNode("camera_matrix").mat()
dist_load = cv_file_load.getNode("dist_coeff").mat()
newcameramtx_load = cv_file_load.getNode("new_camera_matrix").mat()
cv_file_load.release()

#Constants for ArUco detection
arucoDict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
arucoParams = cv2.aruco.DetectorParameters_create()

font = cv2.FONT_HERSHEY_SIMPLEX #font for displaying text (below)

#Set camera parameters
camera = PiCamera()
camera.resolution = (1296, 976)
#camera.framerate = 30
rawCapture = PiRGBArray(camera, size=(1296, 976))
```

```
# allow the camera to warmup
print("Starting PiCamera.....")
time.sleep(0.1)

#For drawing phi
fD = 0

for frame in camera.capture_continuous(rawCapture, format="rgb",
use_video_port=True):
    #Get image and convert
    image = frame.array
    grayOff = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.undistort(grayOff, mtx_load, dist_load, None, newcameramtx_load)

    #Run ArUco detection
    corners, ids, rejected = cv2.aruco.detectMarkers(gray, arucoDict,
parameters=arucoParams)
    #Check if any marker was found
    if len(corners) > 0:
        rvec, tvec, _ = cv2.aruco.estimatePoseSingleMarkers(corners, 0.05,
mtx_load, dist_load)
        (rvec-tvec).any()
        #Loop through all the markers
        for index, cornerInfo in enumerate(corners):
            #Print corner id
            #print(ids[index][0], end=", ")

            #Calculate center of marker
            cX = int((cornerInfo[0][0][0] + cornerInfo[0][2][0]) / 2.0)
            cY = int((cornerInfo[0][0][1] + cornerInfo[0][2][1]) / 2.0)

            ###### DRAW PHI #####
            phi =
math.degrees(math.atan((tvec[index][0][0])/(tvec[index][0][2])))
            phiWrite =str((round(phi*10))/10)
            fD += 1
            #phi = str((tvec[0][0][0])/(tvec[0][0][2]))
            #Hoz tvec[0][0][0]
            #Depth tvec[0][0][2]
            #math.degrees(math.atan(1.18))
            cv2.putText(gray, phiWrite, (cX+4,cY-4), font, 1,
(255,255,255),1,cv2.LINE_AA)

        for i in range(rvec.shape[0]):
```

```
            cv2.drawFrameAxes(gray, mtx_load, dist_load, rvec[i, :, :], tvec[i,
:, :], 0.03)
            #cv2.drawDetectedMarkers(gray, corners)


    cv2.imshow("ArUco 3D Detection", gray)

    #Capture any key press
    key = cv2.waitKey(1) & 0xFF
    # clear the stream in preparation for the next frame
    rawCapture.truncate(0)

    # if the `d' key was pressed, break from the loop
    if key == ord("d"):
        cv2.destroyAllWindows()
        break

# The following tutorial provide info on how to perform
# I also utilized the resources from the other documents
#https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
#https://stackoverflow.com/questions/10057854/inverse-of-tan-in-python-tan-1
#These sources had working example code, but where in Chinese
#They assisted me in getting the correct order of functions
# https://blog.dgut.top/2020/07/20/opencv-biaoding/
# https://blog.dgut.top/2020/07/15/python-aruco/
```