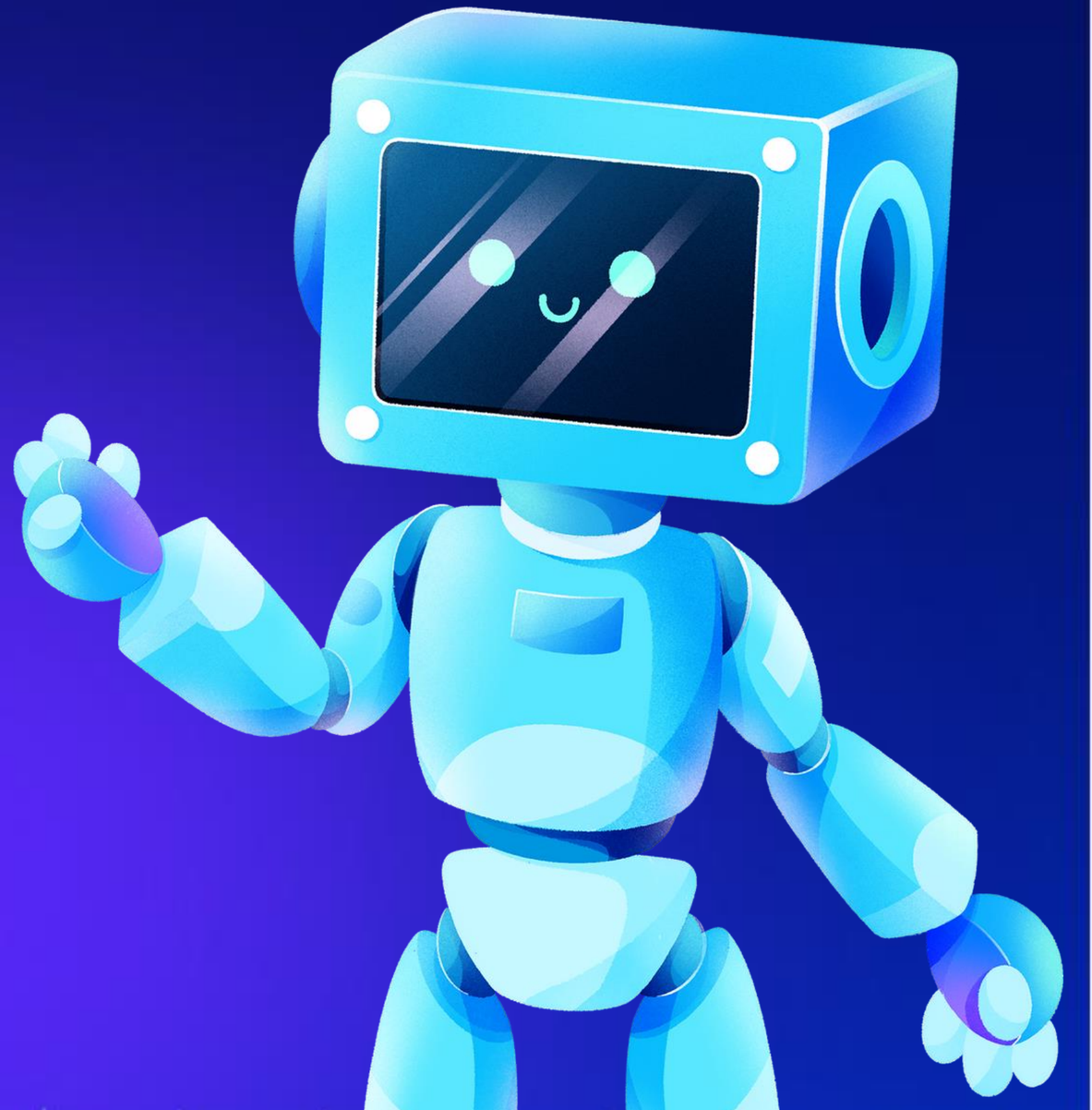
WEEKLY REPORT LAB FTDC TEAM 5

Team Members :
Abdul Aziz



Report on Abdul Aziz

1. Learning Camera Calibration and Trying to Run the Camera Calibration Program

This week, I have been exploring camera calibration through a variety of sources, including a reference guide and YouTube tutorials. I attempted to run a camera calibration program using pictures of chessboard patterns that I had taken myself as calibration material. However, I encountered some errors. Upon further investigation, I discovered that the issue was with the images I provided. Despite taking additional pictures, the error persisted.

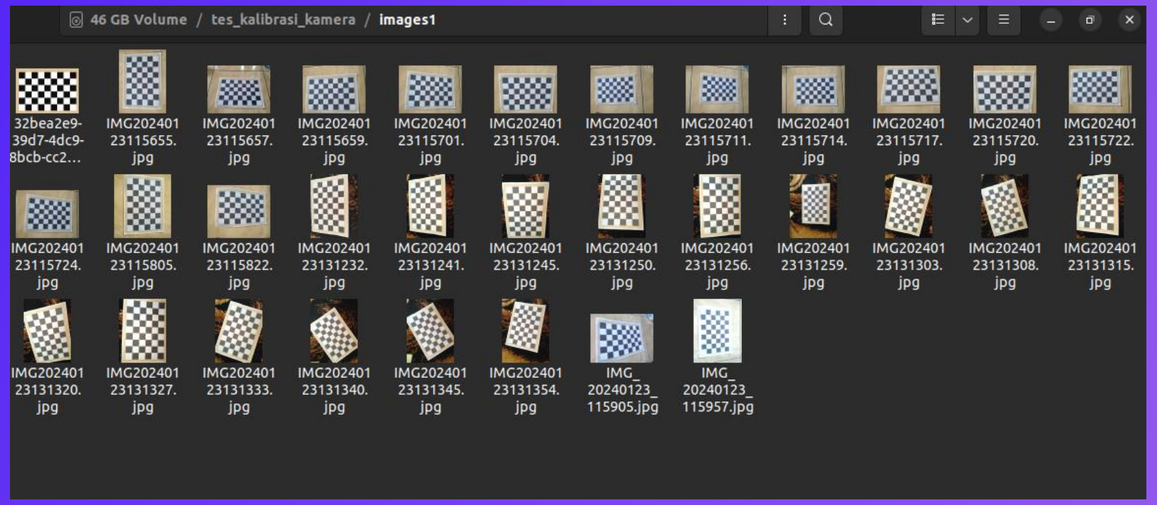
- Capture Error

```
ichika@ichika-VivoBook-14-ASUS-Laptop-X441MA-X441MA:/media/ichika/3F4DD044061E65E3/tes_kalibrasi_kamera$ python3 kalibras
i.py
Traceback (most recent call last):
  File "/media/ichika/3F4DD044061E65E3/tes_kalibrasi_kamera/kalibrasi.py", line 56, in <module>
    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],  None, None)
cv2.error: OpenCV(4.9.0) /io/opencv/modules/calib3d/src/calibration.cpp:3752: error: (-215:Assertion failed) nimages > 0
in function 'calibrateCameraRO'

ichika@ichika-VivoBook-14-ASUS-Laptop-X441MA-X441MA:/media/ichika/3F4DD044061E65E3/tes_kalibrasi_kamera$
* History restored
```

- Capture Program

```python
import cv2
import numpy as np
import os
import glob

# Defining the dimensions of checkerboard
CHECKERBOARD = (6, 9)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# Creating vector to store vectors of 3D points for each checkerboard image
objpoints = []

# Creating vector to store vectors of 2D points for each checkerboard image
imgpoints = []

# Defining the world coordinates for 3D points
objp = np.zeros((1, CHECKERBOARD[0] * CHECKERBOARD[1], 3), np.float32)
objp[0, :, :2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape(-1, 2)
prev_img_shape = None

# Extracting path of individual image stored in a given directory
images = glob.glob('./images1/*.jpg')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Find the chessboard corners
    # If desired number of corners are found in the image then ret = true
    ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
                                             cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK + cv2.CALIB_CB_NORMALIZE_IMAGE)
    """
    If desired number of corner are detected,
    we refine the pixel coordinates and display
    them on the images of checker board
    """
    if ret == True:
        objpoints.append(objp)
        # refining pixel coordinates for given 2d points.
        corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
        imgpoints.append(corners2)
        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, CHECKERBOARD, corners2, ret)
        cv2.imshow('img', img)
        cv2.waitKey(0)

# Ensure that the variable 'img' is defined outside the loop
h, w = img.shape[:2]

"""
Performing camera calibration by
passing the value of known 3D points (objpoints)
and corresponding pixel coordinates of the
detected corners (imgpoints)
"""
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],  None, None)
print("Camera matrix : n")
print(mtx)
print("dist : n")
print(dist)
print("rvecs : n")
print(rvecs)
print("tvecs : n")
print(tvecs)

cv2.destroyAllWindows()
```

- chess board pictures



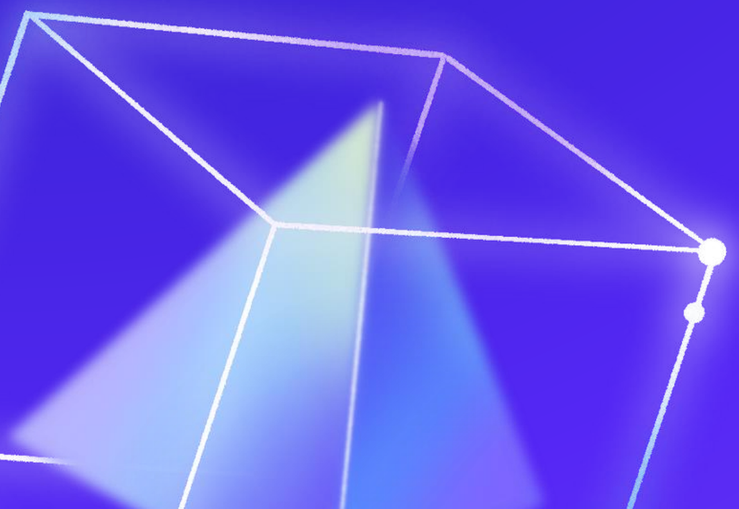
Report on Abdul Aziz

2. Learning Fundamentals of Python

This week, I had the opportunity to study basic Python, specifically functions, and I was able to use this knowledge to create a program that calculates the area and perimeter of a rectangle.

Source

- https://github.com/aziz-0110/weekly-report-1-FTDC
- https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-for-camera-calibration-in-computer-vision/



I am endeavoring to enhance my English language proficiency, so I kindly request your pardon for any imprecisions in my vocabulary selection.

- Capture Program

```python
import cv2
import numpy as np
import os
import glob

# Defining the dimensions of checkerboard
CHECKERBOARD = (6, 9)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# Creating vector to store vectors of 3D points for each checkerboard image
objpoints = []

# Creating vector to store vectors of 2D points for each checkerboard image
imgpoints = []

# Defining the world coordinates for 3D points
objp = np.zeros((1, CHECKERBOARD[0] * CHECKERBOARD[1], 3), np.float32)
objp[0, :, :2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape(-1, 2)
prev_img_shape = None

# Extracting path of individual image stored in a given directory
images = glob.glob('./images1/*.jpg')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Find the chessboard corners
    # If desired number of corners are found in the image then ret = true
    ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
                                             cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK + cv2.CALIB_CB_NORMALIZE_IMAGE)
    """
    If desired number of corner are detected,
    we refine the pixel coordinates and display
    them on the images of checker board
    """
    if ret == True:
        objpoints.append(objp)
        # refining pixel coordinates for given 2d points.
        corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
        imgpoints.append(corners2)
        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, CHECKERBOARD, corners2, ret)
        cv2.imshow('img', img)
        cv2.waitKey(0)

# Ensure that the variable 'img' is defined outside the loop
h, w = img.shape[:2]

"""
Performing camera calibration by
passing the value of known 3D points (objpoints)
and corresponding pixel coordinates of the
detected corners (imgpoints)
"""
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],  None, None)
print("Camera matrix : n")
print(mtx)
print("dist : n")
print(dist)
print("rvecs : n")
print(rvecs)
print("tvecs : n")
print(tvecs)

cv2.destroyAllWindows()
```