NGUYỄN HOÀNG THANH - 2151013087

**Part 1: Getting started**

Code:

import cv2

# videoPath = "datasets/video\_book.mp4"

imgPath = "datasets/logo-ou.png"

img = cv2.imread(imgPath, cv2.IMREAD\_UNCHANGED)

if img.shape[2] == 4:

    img = cv2.cvtColor(img, cv2.COLOR\_RGBA2RGB)

resized\_img = cv2.resize(img, (30, 30))

overlay = resized\_img

# cap = cv2.VideoCapture(videoPath)

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("Không thể kết nối camera. Vui lòng kiểm tra lại.")

    exit()

while True:

    ret, frame = cap.read()

    if not ret:

        break

    frame = cv2.resize(frame, (1024, 768))

    cv2.putText(

        frame,

        "Hoang Thanh - OU",

        (10, 100),

        fontFace=cv2.FONT\_HERSHEY\_SIMPLEX,

        fontScale=0.8,  # Adding the missing fontScale argument

        color=(255, 255, 255),  # Adding font color (white in this case)

        thickness=2,  # Adding text thickness

    )

    height, width, \_ = overlay.shape

    roi = frame[100 : 100 + height, 150 : 150 + width]

    overlay = cv2.resize(overlay, (roi.shape[1], roi.shape[0]))

    frame[25 : 25 + height, 25 : 25 + width] = overlay

    # Hiển thị frame

    cv2.imshow("Frame", frame)

    # Thoát nếu nhấn phím 'q'

    if cv2.waitKey(1) & 0xFF == ord("q"):

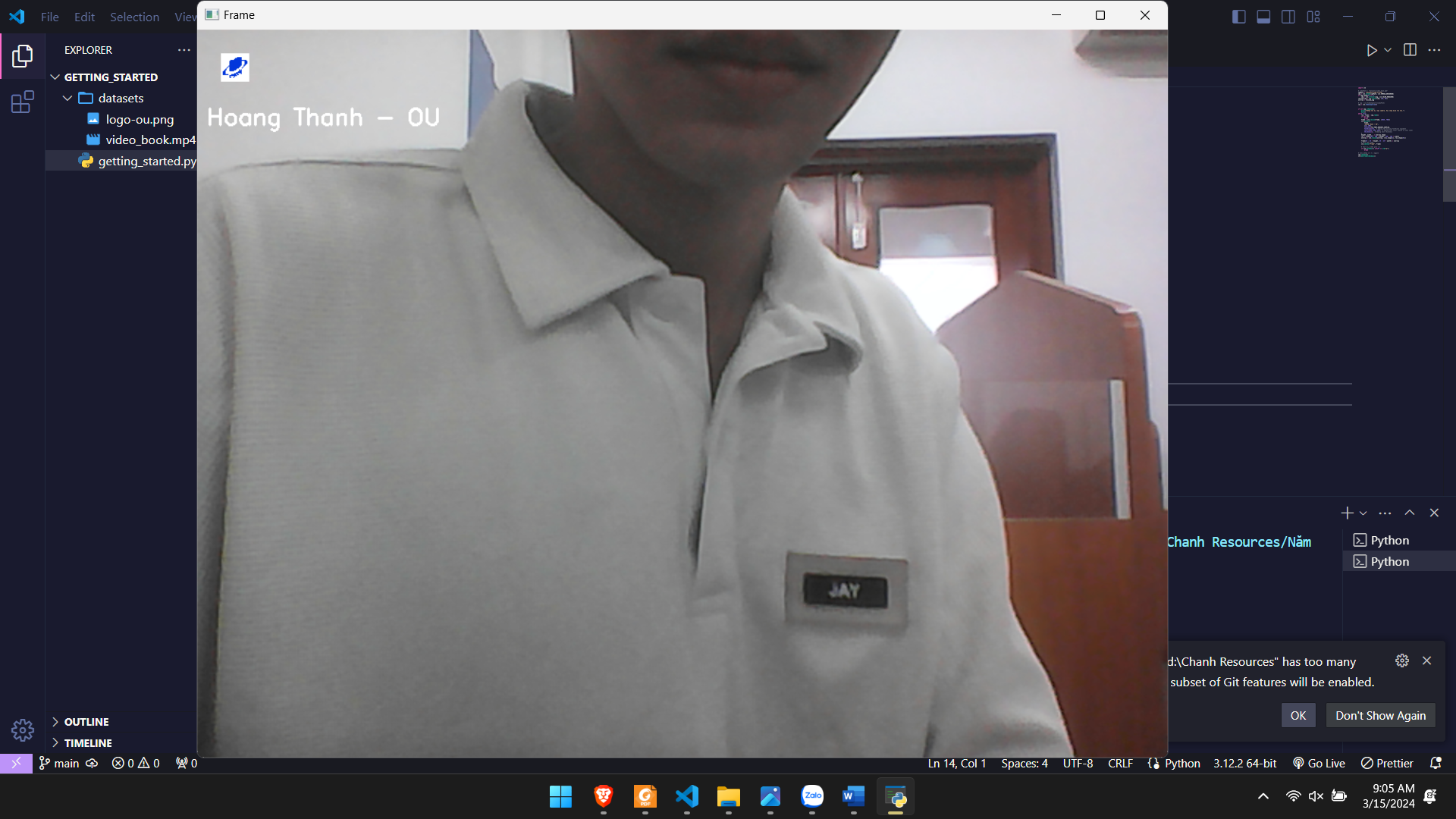
        break

# Giải phóng các tài nguyên

cap.release()

cv2.destroyAllWindows()

Result:



**Part 2: Candel in pinhole**

Code:

import cv2

import numpy as np

import matplotlib.pyplot as plt

def pinholeCamera(

    imageSize=(400, 400),

    pinholeSize=(3, 3),

    objectPosition=(200, 200),

    objectSize=(20, 100),

):

    image = np.zeros((imageSize[0], imageSize[1], 3), dtype=np.uint8)

    pinholeStart = (

        objectPosition[0] - pinholeSize[0] // 2,

        objectPosition[1] - pinholeSize[1] // 2,

    )

    pinholeEnd = (pinholeStart[0] + pinholeSize[0], pinholeStart[1] + pinholeSize[1])

    image[pinholeStart[1] : pinholeEnd[1], pinholeStart[0] : pinholeEnd[0]] = [

        225,

        225,

        225,

    ]

    candleStart = (

        objectPosition[0] - objectSize[0] // 2,

        objectPosition[1] - objectSize[1] // 2,

    )

    candleEnd = (candleStart[0] + objectSize[0], candleStart[1] + objectSize[1])

    image[candleStart[1] : candleEnd[1], candleStart[0] : candleEnd[0]] = [0, 165, 225]

    return image

def plotImages(images, titles):

    fig, axe = plt.subplots(1, len(images), figsize=(12, 4))

    for ax, image, title in zip(axe, images, titles):

        ax.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

        ax.set\_title(title)

        ax.axis("off")

    plt.show()

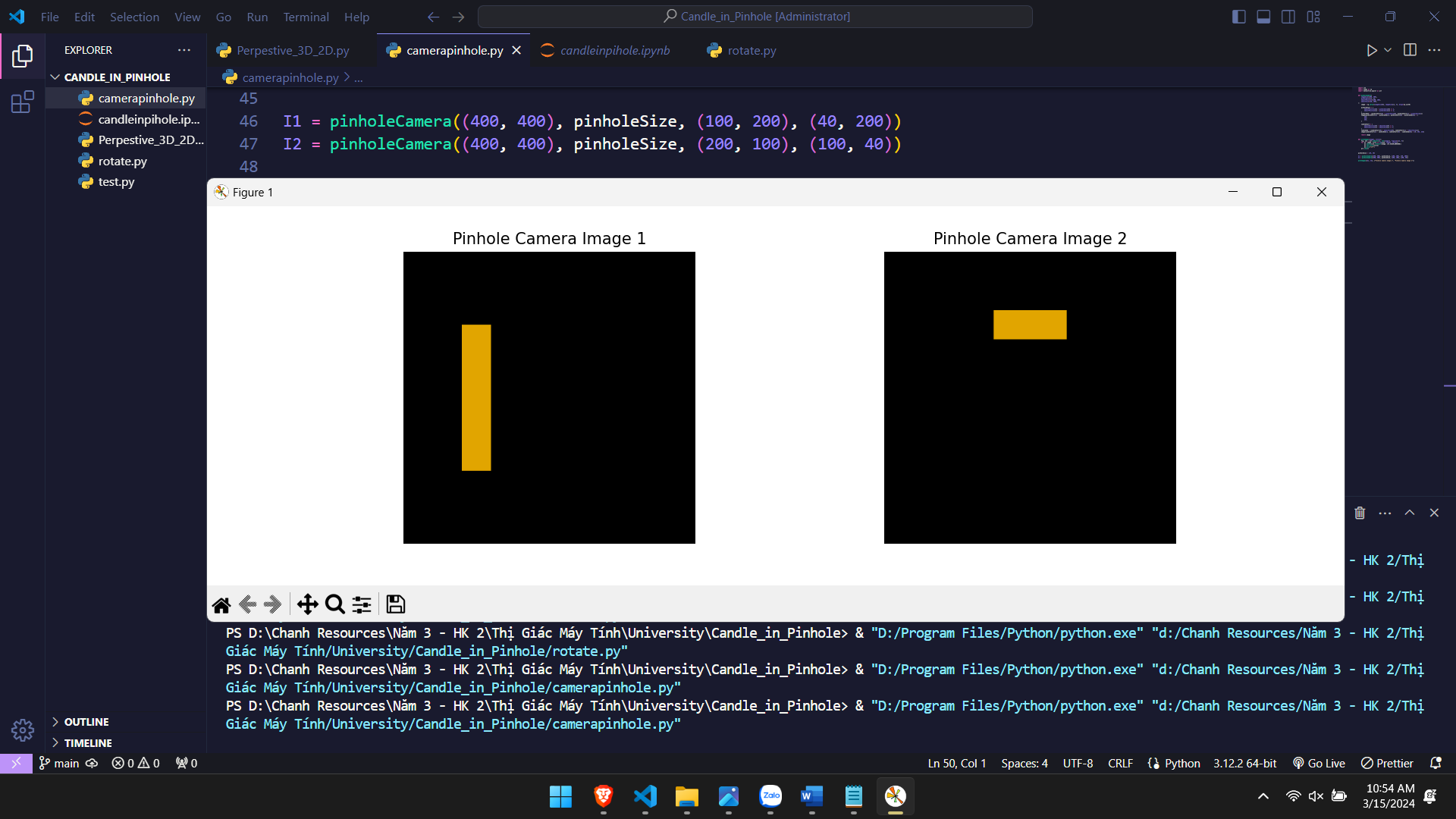
pinholeSize = (10, 10)

I1 = pinholeCamera((400, 400), pinholeSize, (100, 200), (40, 200))

I2 = pinholeCamera((400, 400), pinholeSize, (200, 100), (100, 40))

plotImages([I1, I2], ["Pinhole Camera Image 1", "Pinhole Camera Image 2"])

Result:



**Part 3: Perpestive cube 3D 🡪2D**

Code:

import cv2

import numpy as np

import matplotlib.pyplot as plt

A = np.array([1, 1, 1])

B = np.array([-1, 1, 1])

C = np.array([1, -1, 1])

D = np.array([-1, -1, 1])

E = np.array([1, 1, -1])

F = np.array([-1, 1, -1])

G = np.array([1, -1, -1])

H = np.array([-1, -1, -1])

camera = np.array([4, 2, 7])

Points = dict(zip("ABCDEFGH", [A, B, C, D, E, F, G, H]))

edges = ["AB", "CD", "EF", "GH", "AC", "BD", "EG", "FH", "AE", "CG", "BF", "DH"]

points = {k: v - camera for k, v in Points.items()}

def pinhole(v):

    x, y, z = v

    if z == 0:

        return np.array([float("inf"), float("inf")])

    return np.array([x / z, y / z])

uvs = {k: pinhole(p) for k, p in points.items()}

plt.figure(figsize=(10, 10))

for a, b in edges:

    ua, va = uvs[a]

    ub, vb = uvs[b]

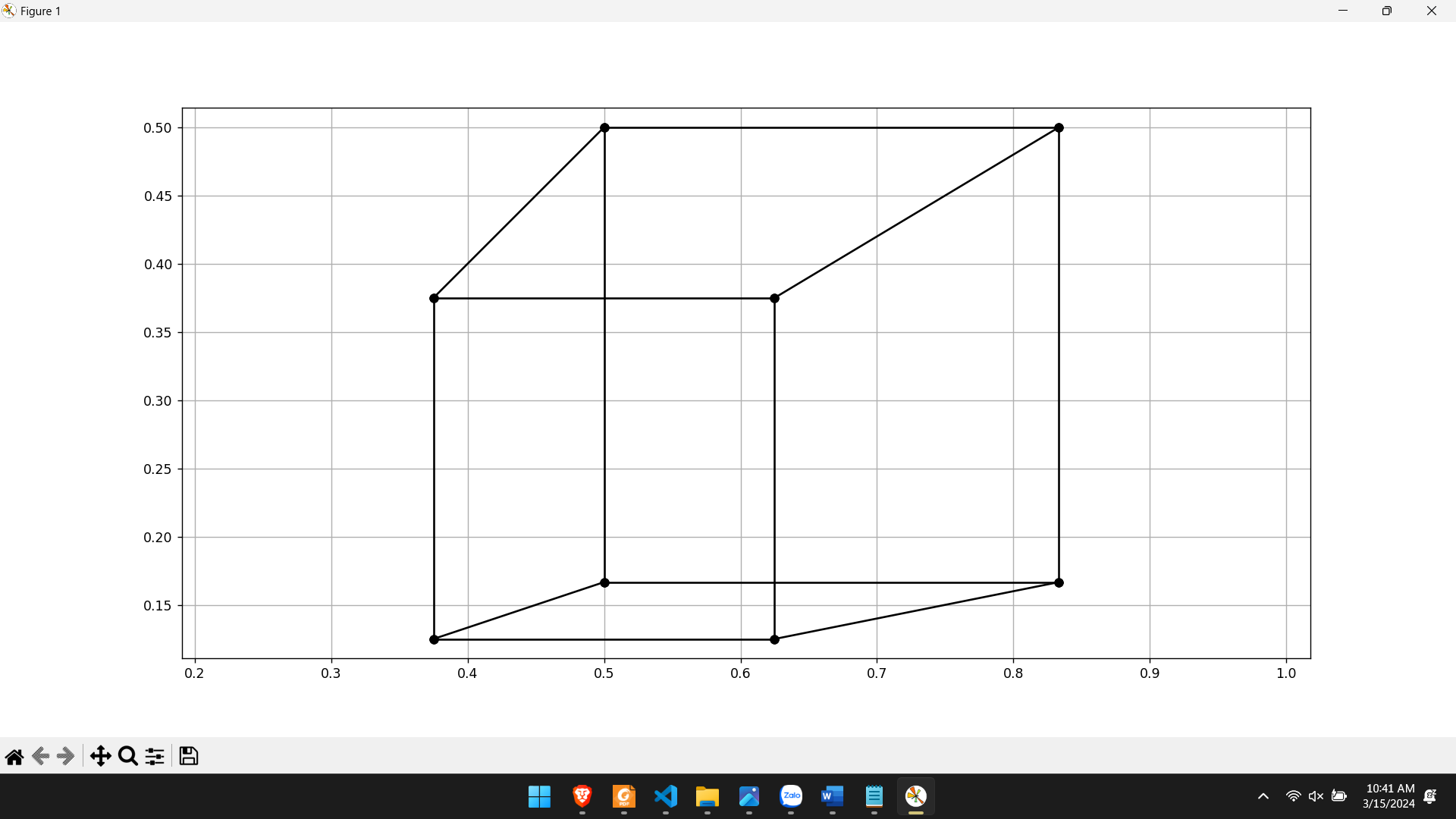
    plt.plot([ua, ub], [va, vb], "ko-")

plt.axis("equal")

plt.grid()

plt.show()

Result:



**Part 4: Rotate cube 3D 🡪 2D**

Code:

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Define points

A = np.array([1, 1, 1])

B = np.array([-1, 1, 1])

C = np.array([1, -1, 1])

D = np.array([-1, -1, 1])

E = np.array([1, 1, -1])

F = np.array([-1, 1, -1])

G = np.array([1, -1, -1])

H = np.array([-1, -1, -1])

camera = np.array([2, 3, 5])

Points = dict(zip("ABCDEFGH", [A, B, C, D, E, F, G, H]))

edges = ["AB", "CD", "EF", "GH", "AC", "BD", "EG", "FH", "AE", "CG", "BF", "DH"]

points = {k: v - camera for k, v in Points.items()}

def pinhole(v):

    x, y, z = v

    if z == 0:

        return np.array([float("inf"), float("inf")])

    return np.array([x / z, y / z])

def rotate(R, v):

    return np.dot(R, v)

def getRotX(angle):

    Rx = np.zeros((3, 3))

    Rx[0, 0] = 1

    Rx[1, 1] = np.cos(angle)

    Rx[1, 2] = -np.sin(angle)

    Rx[2, 1] = np.sin(angle)

    Rx[2, 2] = np.cos(angle)

    return Rx

def getRotY(angle):

    Ry = np.zeros((3, 3))

    Ry[0, 0] = np.cos(angle)

    Ry[0, 2] = -np.sin(angle)

    Ry[2, 0] = np.sin(angle)

    Ry[2, 2] = np.cos(angle)

    Ry[1, 1] = 1

    return Ry

def getRotZ(angle):

    Rz = np.zeros((3, 3))

    Rz[0, 0] = np.cos(angle)

    Rz[0, 1] = -np.sin(angle)

    Rz[1, 0] = np.sin(angle)

    Rz[1, 1] = np.cos(angle)

    Rz[2, 2] = 1

    return Rz

angles = [40, 50, 60]

# Plot for Z rotations

fig, ax = plt.subplots(1, 3, figsize=(15, 5))

for i, angle\_z in enumerate(angles):

    Rz = getRotZ(np.degrees(angle\_z))

    # Apply rotation to points

    ps = {key: rotate(Rz, value) for key, value in points.items()}

    uvs = {key: pinhole(value) for key, value in ps.items()}

    # Plot edges

    for a, b in edges:

        ua, va = uvs[a]

        ub, vb = uvs[b]

        ax[i].plot([ua, ub], [va, vb], "ko-")

    ax[i].set\_title(f"Z{angle\_z}")

    ax[i].axis("equal")

    ax[i].grid()

plt.show()

# Plot for Y rotations

fig, ax = plt.subplots(1, 3, figsize=(15, 5))

for i, angle\_y in enumerate(angles):

    Ry = getRotY(np.degrees(angle\_y))

    # Apply rotation to points

    ps = {key: rotate(Ry, value) for key, value in points.items()}

    uvs = {key: pinhole(value) for key, value in ps.items()}

    # Plot edges

    for a, b in edges:

        ua, va = uvs[a]

        ub, vb = uvs[b]

        ax[i].plot([ua, ub], [va, vb], "ko-")

    ax[i].set\_title(f"Y{angle\_y}")

    ax[i].axis("equal")

    ax[i].grid()

plt.show()

# Plot for X rotations

fig, ax = plt.subplots(1, 3, figsize=(15, 5))

for i, angle\_x in enumerate(angles):

    Rx = getRotX(np.degrees(angle\_x))

    # Apply rotation to points

    ps = {key: rotate(Rx, value) for key, value in points.items()}

    uvs = {key: pinhole(value) for key, value in ps.items()}

    # Plot edges

    for a, b in edges:

        ua, va = uvs[a]

        ub, vb = uvs[b]

        ax[i].plot([ua, ub], [va, vb], "ko-")

    ax[i].set\_title(f"X{angle\_x}")

    ax[i].axis("equal")

    ax[i].grid()

plt.show()

Result:

