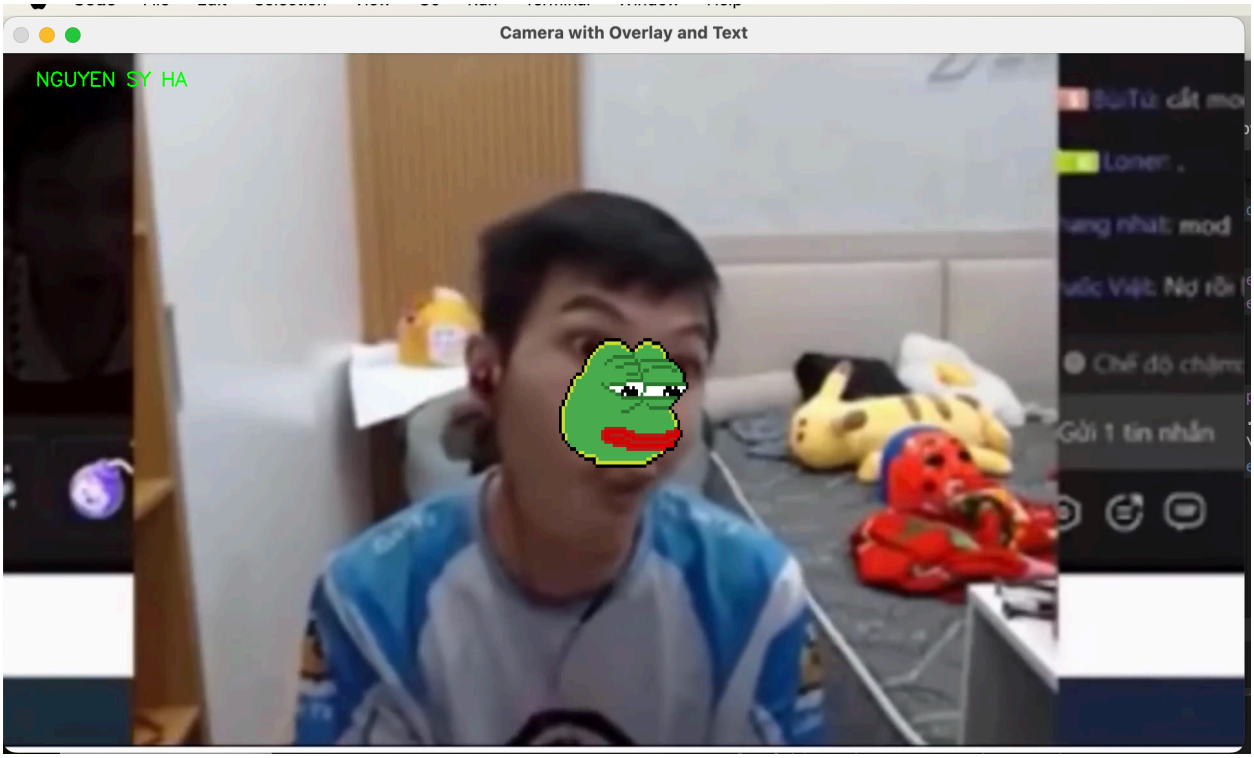
1. Getting started (0.5)

Capture frames from MP4 or true camera, overlay your text and logo (png/jpg) into frame



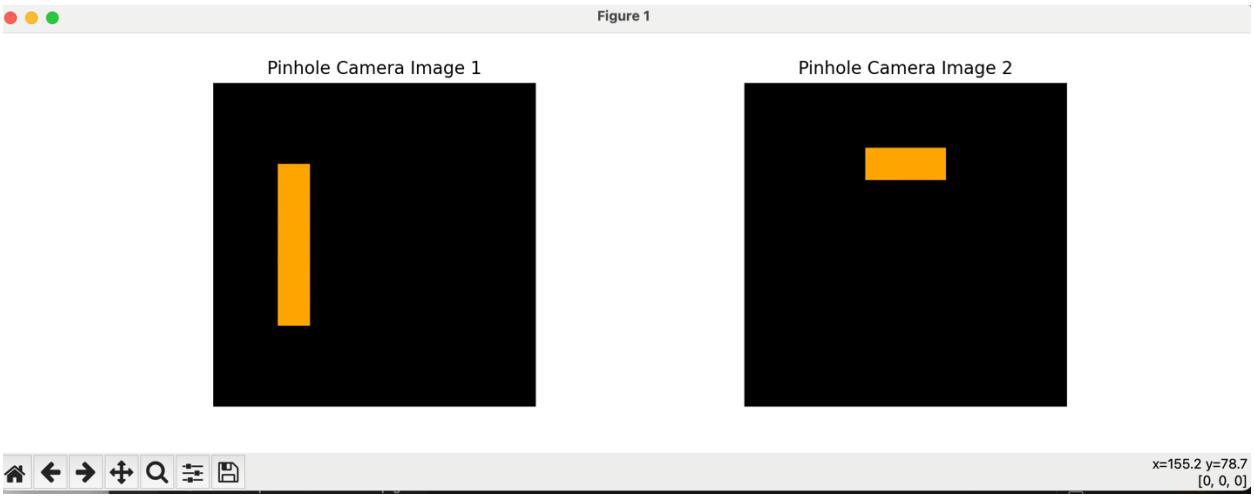
MÃ NGUỒN :

```
import cv2

# Đọc video
input_video_path = '/Users/syha/Downloads/CHETDOI.mp4'
cap = cv2.VideoCapture(input_video_path)

# Đọc hình ảnh bạn muốn chèn (hình nhỏ)
overlay = cv2.imread('/Users/syha/Downloads/pixel.png', cv2.IMREAD_UNCHANGED)

# Kích thước mới cho hình ảnh nhỏ
desired_width = 200
desired_height = 200

# # Mở camera
# cap = cv2.VideoCapture(0)

while True:
    # Đọc frame từ camera
```

```
    ret, frame = cap.read()
    ## Kiểm tra đã đọc frame thành công không
    if not ret:
        break
    # Resize overlay để phù hợp với kích thước frame
    overlay_resized = cv2.resize(overlay, (desired_width, desired_height))

    # Tính toán vị trí để đặt overlay giữa frame
    y_offset = (frame.shape[0] - overlay_resized.shape[0]) // 2
    x_offset = (frame.shape[1] - overlay_resized.shape[1]) // 2

    # Chèn overlay vào frame
    for c in range(0, 3):
        frame[y_offset:y_offset + overlay_resized.shape[0], x_offset:x_offset +
overlay_resized.shape[1], c] = \
            frame[y_offset:y_offset + overlay_resized.shape[0], x_offset:x_offset +
overlay_resized.shape[1], c] * \
            (1 - overlay_resized[:, :, 3] / 255.0) + \
            overlay_resized[:, :, c] * (overlay_resized[:, :, 3] / 255.0)

    # Chèn chữ vào frame
    text = "NGUYEN SY HA"
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 1
    font_thickness = 2
    text_color = (0, 255, 0)  # Màu chữ là xanh (BGR)
    text_position = (50, 50)

    cv2.putText(frame, text, text_position, font, font_scale, text_color,
font_thickness)

    # Hiển thị frame
    cv2.imshow('Camera with Overlay and Text', frame)

    # Thoát nếu nhấn phím 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Giải phóng tài nguyên
cap.release()
cv2.destroyAllWindows()
```

2. Object in pinhole (1.0)

Simulate image formation by drawing the object through [pinhole camera]



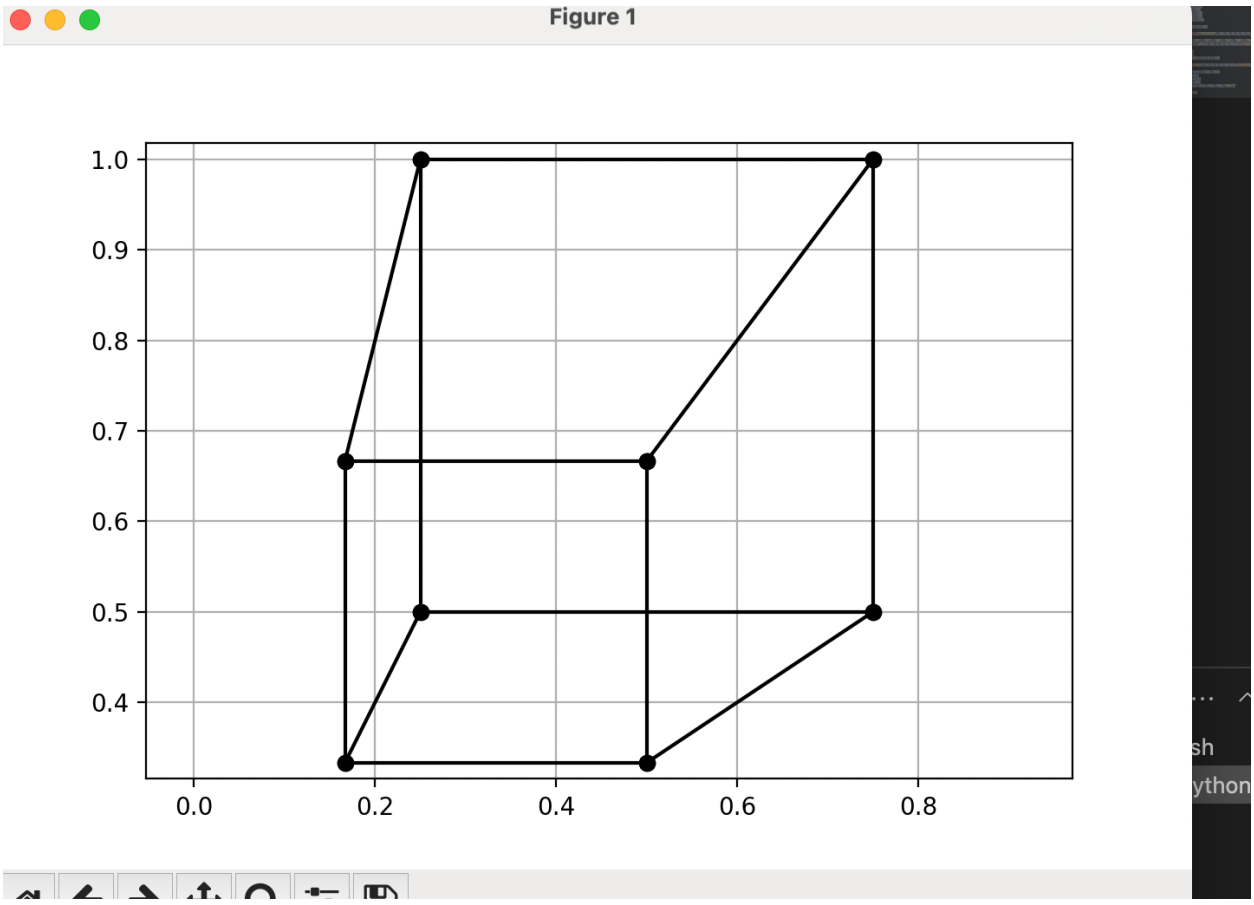
MÃ NGUỒN :

```
import numpy as np
import cv2
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]] = [255, 255,
255]

    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, 255]

    return image

def plotImages(images, titles):
    fig, axes = plt.subplots(1, len(images), figsize=(12, 4))

    for ax, image, title in zip(axes, 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'])
```

3. Perspective 3D cube (0.5)

Perspective 3D cube to image plane



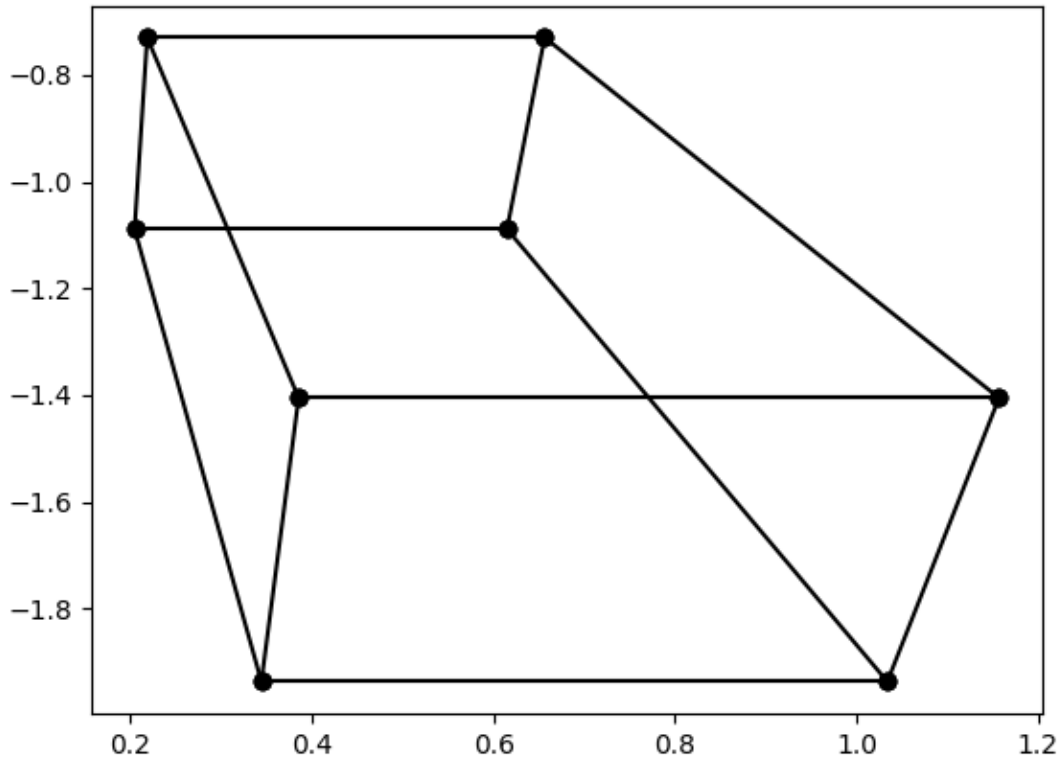
MÃ NGUỒN :

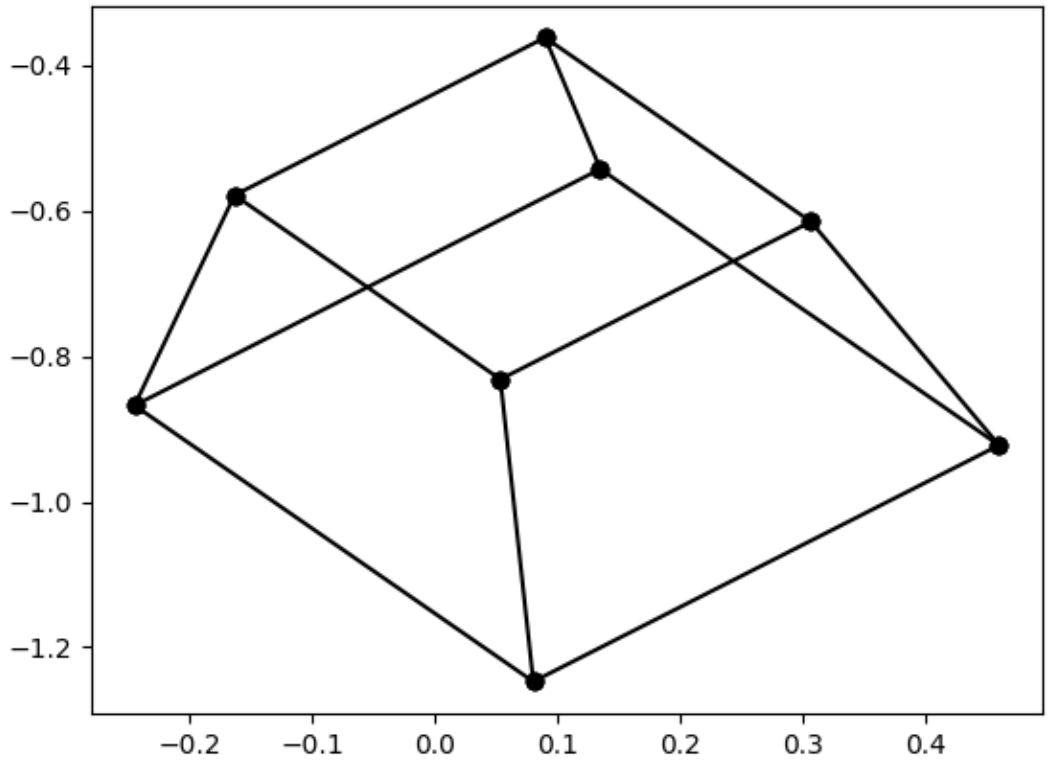
```
import numpy as np
import matplotlib.pyplot as plt

vec = np.array

A = vec([1, 1, 1])
```

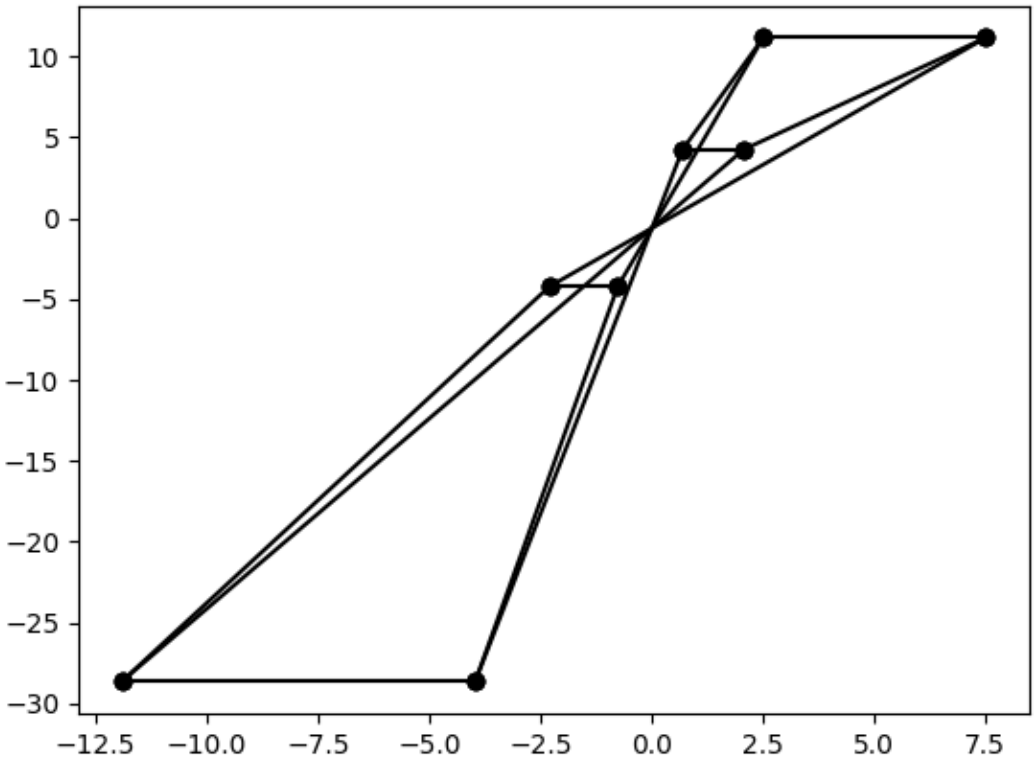
```
B = vec([-1, 1, 1])
C = vec([1, -1, 1])
D = vec([-1, -1, 1])
E = vec([1, 1, -1])
F = vec([-1, 1, -1])
G = vec([1, -1, -1])
H = vec([-1, -1, -1])

camera = vec([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: p - camera for k, p in Points.items()}

def pinhole(v):
    x, y, z = v
    return vec([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()
```

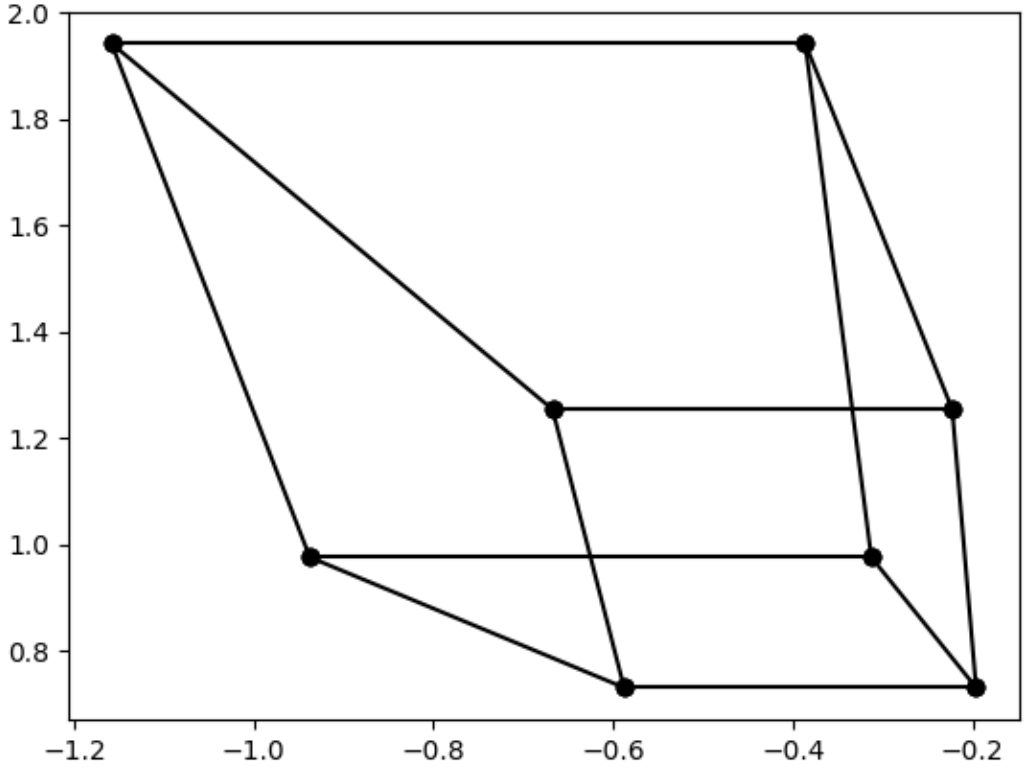
4. Rotate Cube 3D (1.0)

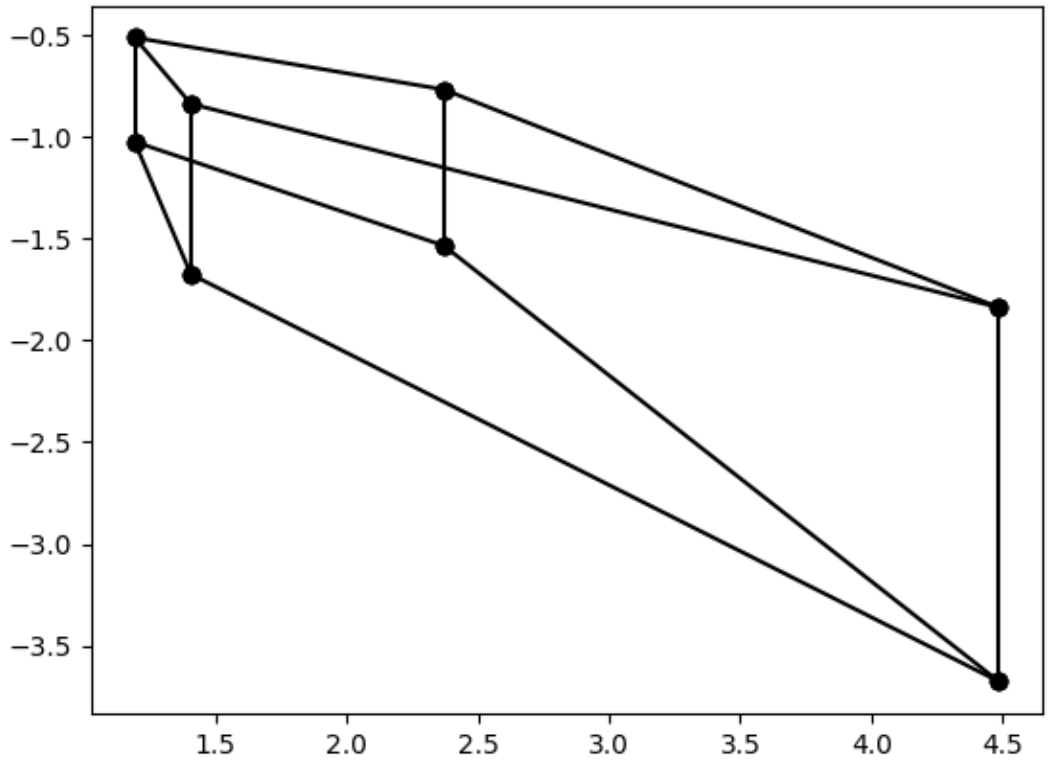
Rotate 3D cube by the axis (X, Y, Z), project rotated object to image plane

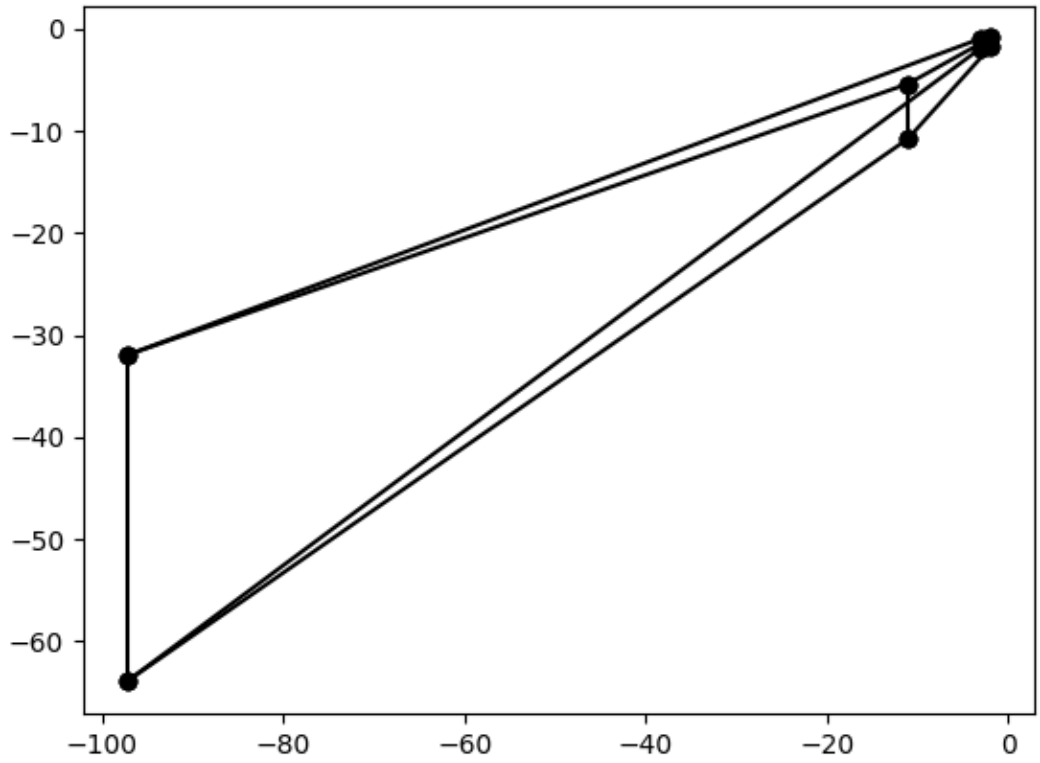

−0.8
−1.0
−1.2
−1.4
−1.6
−1.8
0.2
0.4
0.6
0.8
1.0
1.2

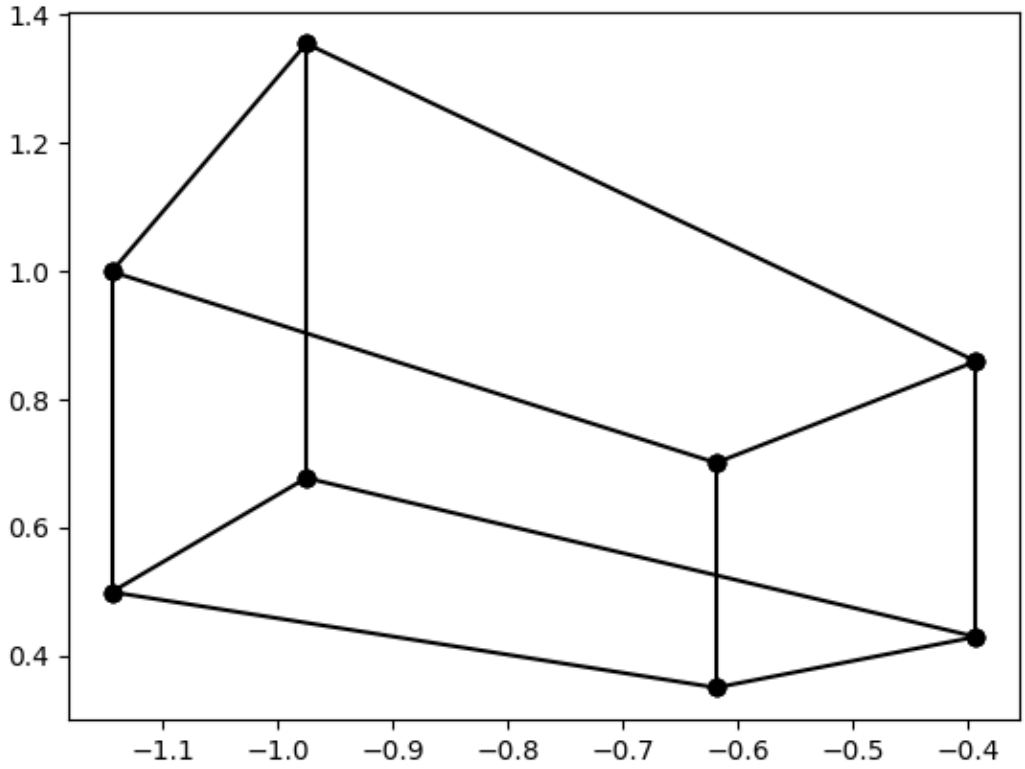

−0.4
−0.6
−0.8
−1.0
−1.2
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4

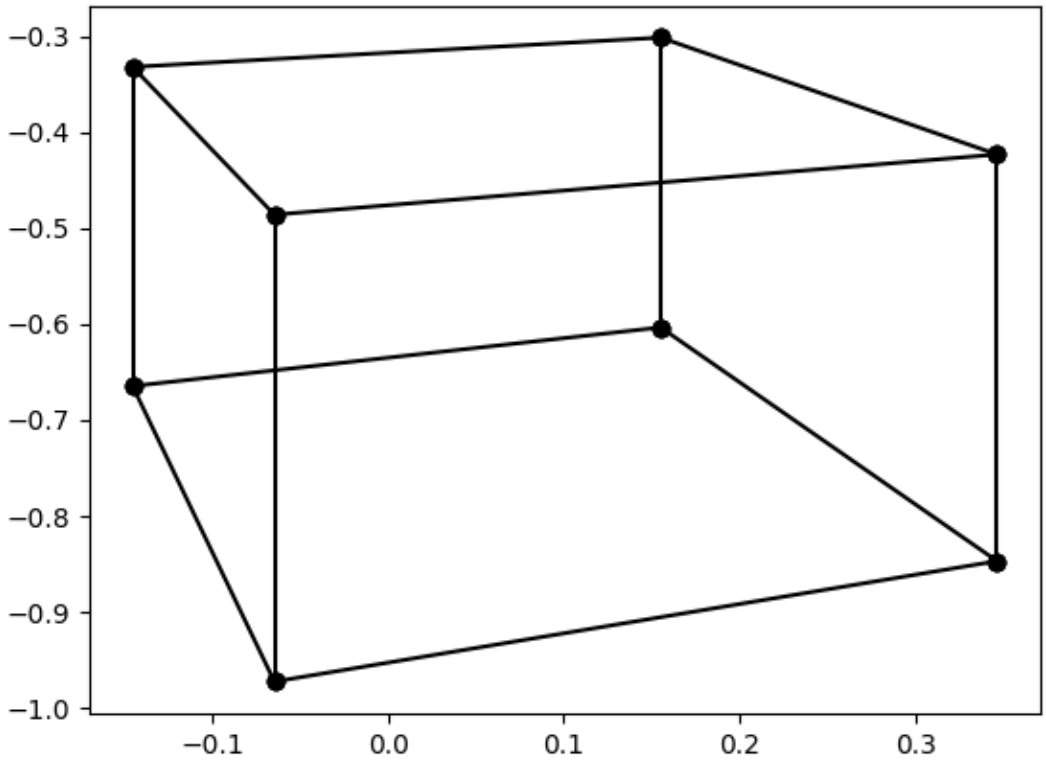

10
5
0
−5
−10
−15
−20
−25
−30
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

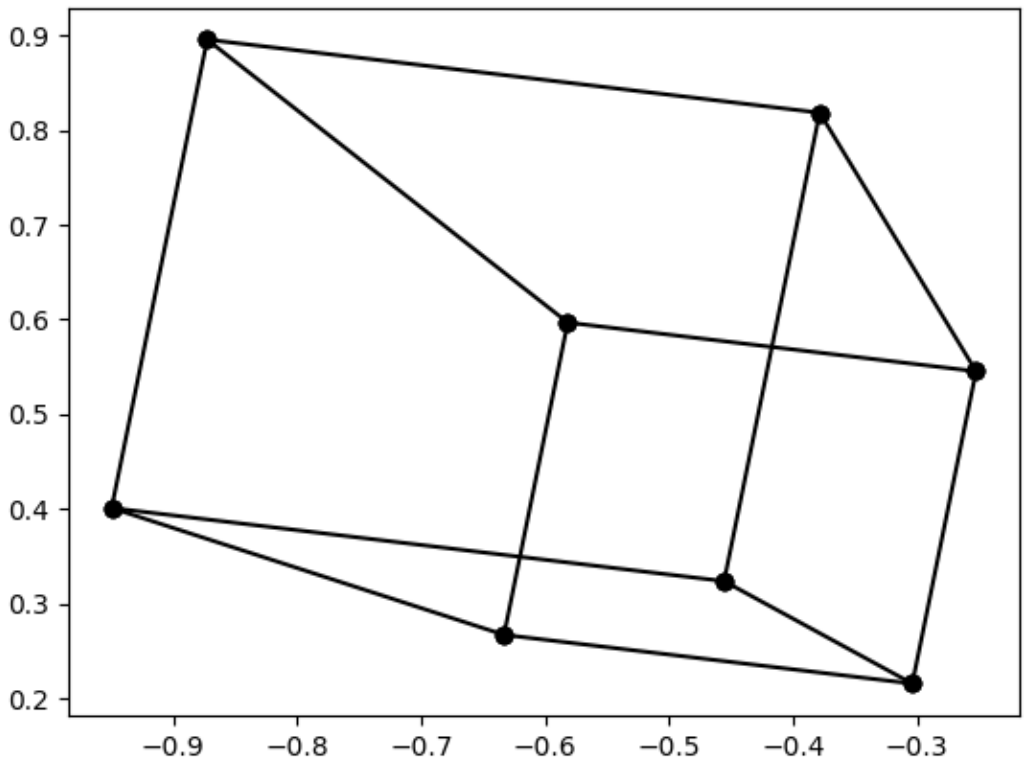

2.0
1.8
1.6
1.4
1.2
1.0
0.8
−1.2
−1.0
−0.8
−0.6
−0.4
−0.2

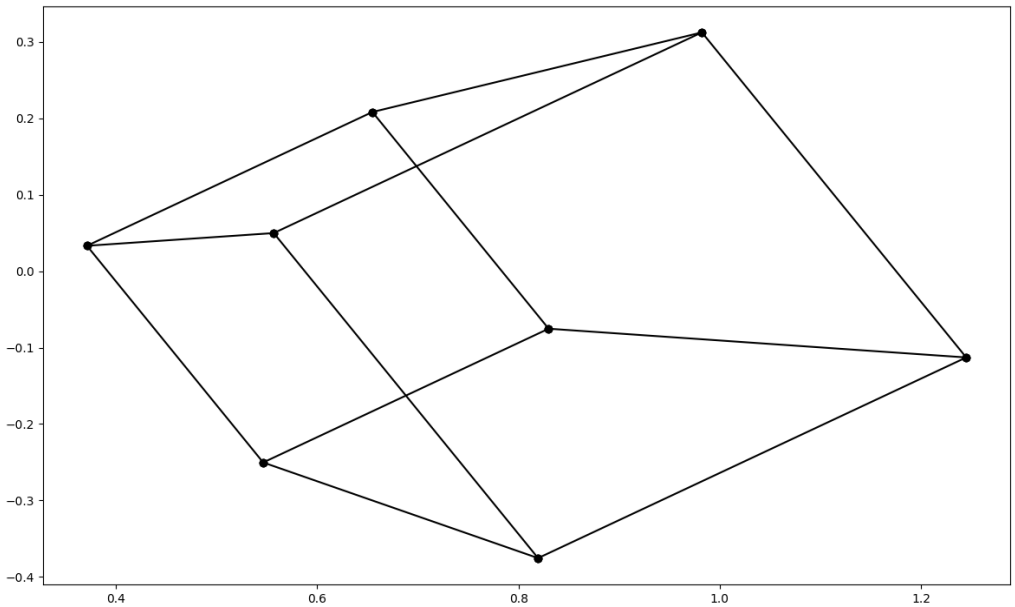

−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5


0
−10
−20
−30
−40
−50
−60
−100
−80
−60
−40
−20
0


1.4
1.2
1.0
0.8
0.6
0.4
−1.1
−1.0
−0.9
−0.8
−0.7
−0.6
−0.5
−0.4


−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
−1.0
−0.1
0.0
0.1
0.2
0.3


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
−0.9
−0.8
−0.7
−0.6
−0.5
−0.4
−0.3


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0.4
0.6
0.8
1.0
1.2

MÃ NGUỒN :

```
import numpy as np

vec = np.array

#Khai báo các điểm trong hệ tọa độ 3 chiều:
A = vec([1,1,1])
B = vec([-1,1,1])
C = vec([1,-1,1])
D = vec([-1,-1,1])
E = vec([1,1,-1])
F = vec([-1,1,-1])
G = vec([1,-1,-1])
H = vec([-1,-1,-1])

#Đặt camera tại vị trí:
camera = vec([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: p - camera for k, p in Points.items()} #k is key, p is value

#Hàm nhận input là 1 vector 3 tham số, trả về output là vector 2 tham số:
def pinhole(v):
    x, y, z = v
    return vec([x/z , y/z ])


#Hàm tạo ma trận xoay quanh trục X
def getRotX(angle):
    #Tạo ma trận 3 hàng 3 cột, gán giá trị ban đầu cho mọi giá trị trong ma trận là 0
    Rx = np.zeros(shape=(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

#Hàm tạo ma trận xoay quanh trục Y
def getRotY(angle):
    Ry = np.zeros(shape=(3,3))
    Ry[0,0] = np.cos(angle)
    Ry[0,2] = np.sin(angle)
    Ry[1,1] = 1
```

```
    Ry[2,0] = -np.sin(angle)
    Ry[2,2] = np.cos(angle)

    return Ry

#Hàm tạo ma trận xoay quanh trục Z
def getRotZ(angle):
    Rz = np.zeros(shape=(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

#Hàm xoay, nhận 2 giá trị: R là ma trận, v là vector
def rotate(R, v):
    return np.matmul(v, R)

import matplotlib.pyplot as plt

plt.figure(figsize = (5, 5))

angles = [15, 30, 45, 60]
for angle in angles:
    Rx = getRotX(angle)
    plt.show()

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

    for a, b in edges:
        ua, va = uvs[a]
        ub, vb = uvs[b]
        plt.plot([ua, ub], [va, vb], "ko-")

for angle in angles:
    Ry = getRotY(angle)
    plt.show()

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

    for a, b in edges:
        ua, va = uvs[a]
        ub, vb = uvs[b]
        plt.plot([ua, ub], [va, vb], "ko-")
```

```
for angle in angles:
    Rz = getRotZ(angle)
    plt.show()

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

    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()
```