Bài 01: Face Detection Haar

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

|  |
| --- |
| import numpy as np import cv2  videoPath = "pexels\_videos\_2912 (1080p).mp4" imgPath = "751190d924a888f6d1b9.jpg" overlay = cv2.imread(imgPath, cv2.IMREAD\_UNCHANGED)  cap = cv2.VideoCapture(videoPath)  while True:  ret, frame = cap.read()  if not ret:  break  cv2.putText(frame, "Hello, My name is Ka - 2151010266", (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2)  height, width, \_ = overlay.shape  roi = frame[50:50 + height, 100:25 + width]  overlay\_resized = cv2.resize(overlay, (int(roi.shape[1] \* 0.2), int(roi.shape[0] \* 0.2)))  frame[100:100 + overlay\_resized.shape[0], 200:200 + overlay\_resized.shape[1]] = overlay\_resized   cv2.imshow("video with text", frame)  if cv2.waitKey(25) & 0xFF == ord('q'):  break cap.release() cv2.destroyAllWindows() |
|  |

Bài 02: Object in pinhole (1.0)

Simulate image formation by drawing the object through pinhole camera

|  |
| --- |
| 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)):  # Create a black image  image = np.zeros((imageSize[0], imageSize[1], 3), dtype=np.uint8)   # Simulate pinhole by setting a region to white where light passes through  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]   # Simulate the object in the scene  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] # color for the object   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)) # Convert BGR to RGB for matplotlib  ax.set\_title(title)  ax.axis('off')  plt.show()  # Simulate pinhole camera image formation with a candle object pinholeSize = (10, 10)  I1 = pinholeCamera((400, 400), pinholeSize, (100, 200), (40, 200)) I2 = pinholeCamera((400, 400), pinholeSize, (200, 100), (100, 40))  # Display simulated images plotImages([I1, I2], ['Pinhole camera image 1', 'Pinhole camera image 2']) |
|  |

Bài 3: Perspective 3D cube (0.5)

Perspective 3D cube to image plane

|  |
| --- |
| import numpy as np 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()} import matplotlib.pyplot as plt 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() |
|  |

Bài 4: Rotate Cube 3D (1.0)

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

|  |
| --- |
| 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])  def getRotX(angle):  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  def getRotY(angle):  Ry = np.zeros(shape=(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(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   def rotate(R, v):  return np.matmul(v, R)  plt.figure(figsize=(10, 10))  angles = [15, 30, 45, 60] for angle in angles:  Rz = getRotZ(angle)  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.title("Rotation Rz {}".format(angle))  plt.plot([ua, ub], [va, vb], "ko-")  plt.pause(2)  plt.clf()  for angle in angles:  Rx = getRotX(angle)  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.title("Rotation Rx {}".format(angle))  plt.plot([ua, ub], [va, vb], "ko-")  plt.pause(2)  plt.clf()  for angle in angles:  Ry = getRotY(angle)  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.title("Rotation Ry {}".format(angle))  plt.plot([ua, ub], [va, vb], "ko-")  plt.pause(2)  plt.clf() plt.axis("equal") plt.grid()  plt.show() |
| |  | | --- | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |