```
# %%
import cv2
import mediapipe as mp
import numpy as np
import timeit

start = timeit.timeit()


LEFT_IRIS = [474, 475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]
LEFT_EYE = [362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385,
384, 398]
RIGHT_EYE = [33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160,
161, 246]

mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh

drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
cv2.namedWindow('Eye', cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty('Eye', cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)


def get_ratio(mesh_points):
    lw = mesh_points[263][0] - mesh_points[362][0]
    rw = mesh_points[133][0] - mesh_points[33][0]
    lh = mesh_points[374][1] - mesh_points[386][1]
    rh = mesh_points[145][1] - mesh_points[159][1]
    lxratio = (mesh_points[473][0] - mesh_points[362][0]) / lw
    rxratio = (mesh_points[468][0] - mesh_points[33][0]) / rw
    lyratio = (mesh_points[473][1] - mesh_points[386][1]) / lh
    ryratio = (mesh_points[468][1] - mesh_points[159][1]) / rh
    return lxratio, lyratio, rxratio, ryratio


def draw_pupil(img, mesh_points, w, h):
    cv2.circle(  # left
        img,
        np.round(mesh_points[473] * [w, h]).astype(int),
        3,
        (0, 255, 255),
        3,
    )
    cv2.circle(  # right
        img,
        np.round(mesh_points[468] * [w, h]).astype(int),
        3,
        (0, 255, 255),
        3,
    )


def draw_gaze(img, lxratio, lyratio, rxratio, ryratio, w, h):

    looking_x = round(w * (lxratio + rxratio) / 2)
```

```
    looking_y = round(h * (lyratio + ryratio) / 2)
    looking_x = min(max(looking_x, 0), w)
    looking_y = min(max(looking_y, 0), h)
    # print(looking_x, looking_y)
    cv2.circle(img, (looking_x, looking_y), 20, (0, 0, 255), 3)

    text = ''
    if (lxratio + rxratio) / 2 < 0.2:
        text = 'Left'
    elif (lxratio + rxratio) / 2 > 0.8:
        text = 'Right'
    else:
        text = 'None'
    text += '\n'
    end = timeit.timeit()

    try:
        with open(r'C:\Users\nbook\Desktop\output.txt', 'a') as f:
            f.writelines(text)
        with open(r'C:\Users\nbook\AppData\LocalLow\DefaultCompany\CSE 5546\
test.txt', 'a') as f:
            f.writelines(text)
    except:
        pass


mat_target = np.zeros((4, 2))
for i, x in enumerate([0, 1]):
    for j, y in enumerate([0, 1]):
        mat_target[i * 2 + j][0] = x
        mat_target[i * 2 + j][1] = y

with mp_face_mesh.FaceMesh(
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5,
    refine_landmarks=True,
    max_num_faces=1,
) as face_mesh:
    cal_left = []
    cal_right = []
    while cap.isOpened():
        success, img = cap.read()
        if not success:
            print("Ignoring empty camera frame")
            continue
        img = cv2.flip(img, 1)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_h, img_w = img.shape[:2]
        img.flags.writeable = False
        results = face_mesh.process(img)

        img.flags.writeable = True
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

        if results.multi_face_landmarks:
            face_landmarks = results.multi_face_landmarks[0]
            mp_drawing.draw_landmarks(
                image=img,
                landmark_list=face_landmarks,
```

```
                connections=mp_face_mesh.FACEMESH_CONTOURS,
                landmark_drawing_spec=drawing_spec,
                connection_drawing_spec=drawing_spec,
            )
            mesh_points = np.array([[p.x, p.y] for p in face_landmarks.landmark])
            draw_pupil(img, mesh_points, img_w, img_h)
            lxratio, lyratio, rxratio, ryratio = get_ratio(mesh_points)

            # calibration
            cal_num = len(cal_left)
            if cal_num == 4:  # calibrated
                # draw_gaze(img, lxratio, lyratio, rxratio, ryratio, img_w, img_h)

                # lxratio_adj, lyratio_adj = matl @ np.array([lxratio, lyratio, 1])
                # rxratio_adj, ryratio_adj = matr @ np.array([rxratio, ryratio, 1])
                # draw_gaze(img, lxratio_adj, lyratio_adj, rxratio_adj,
ryratio_adj, img_w, img_h)

                lxratio_adj = (lxratio - lxmin) / (lxmax - lxmin)
                rxratio_adj = (rxratio - rxmin) / (rxmax - rxmin)
                lyratio_adj = (lyratio - lymin) / (lymax - lymin)
                ryratio_adj = (ryratio - rymin) / (rymax - rymin)
                # draw_gaze(img, lxratio_adj, lyratio, rxratio_adj, ryratio, img_w,
img_h)
                draw_gaze(img, lxratio_adj, lyratio_adj, rxratio_adj, ryratio_adj,
img_w, img_h)

                if cv2.waitKey(1) & 0xFF == ord('r'):
                    cal_left = []
                    cal_right = []
            else:  # not calibrated
                cv2.putText(
                    img,
                    f'Please do eye calibration first: {cal_num}/4',
                    (50, 150),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    1,
                    (0, 255, 255),
                    2,
                    cv2.LINE_AA,
                )
                cv2.circle(img, (cal_num % 2 * 1280, cal_num // 2 * 720), 10, (255,
255, 0), 20)
                key = cv2.waitKey(32)
                if key == 32:
                    cal_left.append((lxratio, lyratio))
                    cal_right.append((rxratio, ryratio))
                if len(cal_left) == 4:
                    mat_source_l, mat_source_r = np.array(cal_left),
np.array(cal_right)
                    print(np.round(cal_left, 2), '\n\n', np.round(cal_right, 2))
                    matl, _ = cv2.estimateAffine2D(mat_source_l, mat_target)
                    matr, _ = cv2.estimateAffine2D(mat_source_r, mat_target)
                    lxmin = mat_source_l[:, 0].min()
                    lxmax = mat_source_l[:, 0].max()
                    lymin = mat_source_l[:, 1].min()
                    lymax = mat_source_l[:, 1].max()
                    rxmin = mat_source_r[:, 0].min()
                    rxmax = mat_source_r[:, 0].max()
```

```
                    rymin = mat_source_r[:, 1].min()
                    rymax = mat_source_r[:, 1].max()
                    # break

        cv2.imshow('Eye', img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()

# %%
```