Final Project Report

Stereovision
Object Detection
Action Tracking

Objectives

A lightweight software to track people and objects with live camera feed(s).

Goal is to optimize such that it should be able to run on edge devices.

Classical image processing techniques must be used to improve performance and in methodology used.

Surveillance objectives such as identification, tracking, monitoring and logging movements of people in one and two camera systems.

Performance objectives such as frame skipping, thresholding, tagging bounding boxes, etc. must be explored

Deliverables

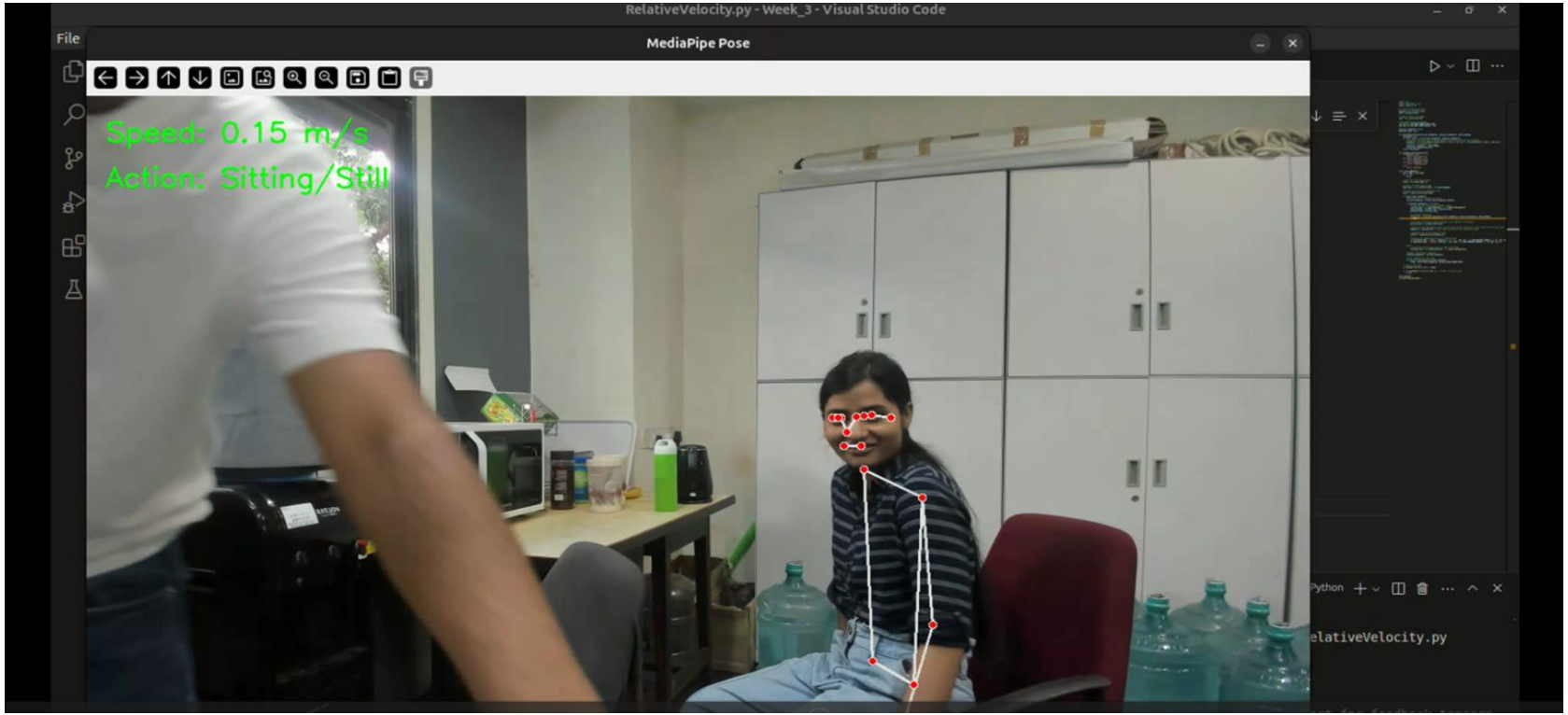
1. Gaze Detection
2. Face Detection
3. Pose Detection
4. Relative Velocity
5. Person Re Identification (2 Camera)
6. Depth Perception (2 Camera)
7. Face recognition
8. Face re identification

Phase 1

1 Camera Systems

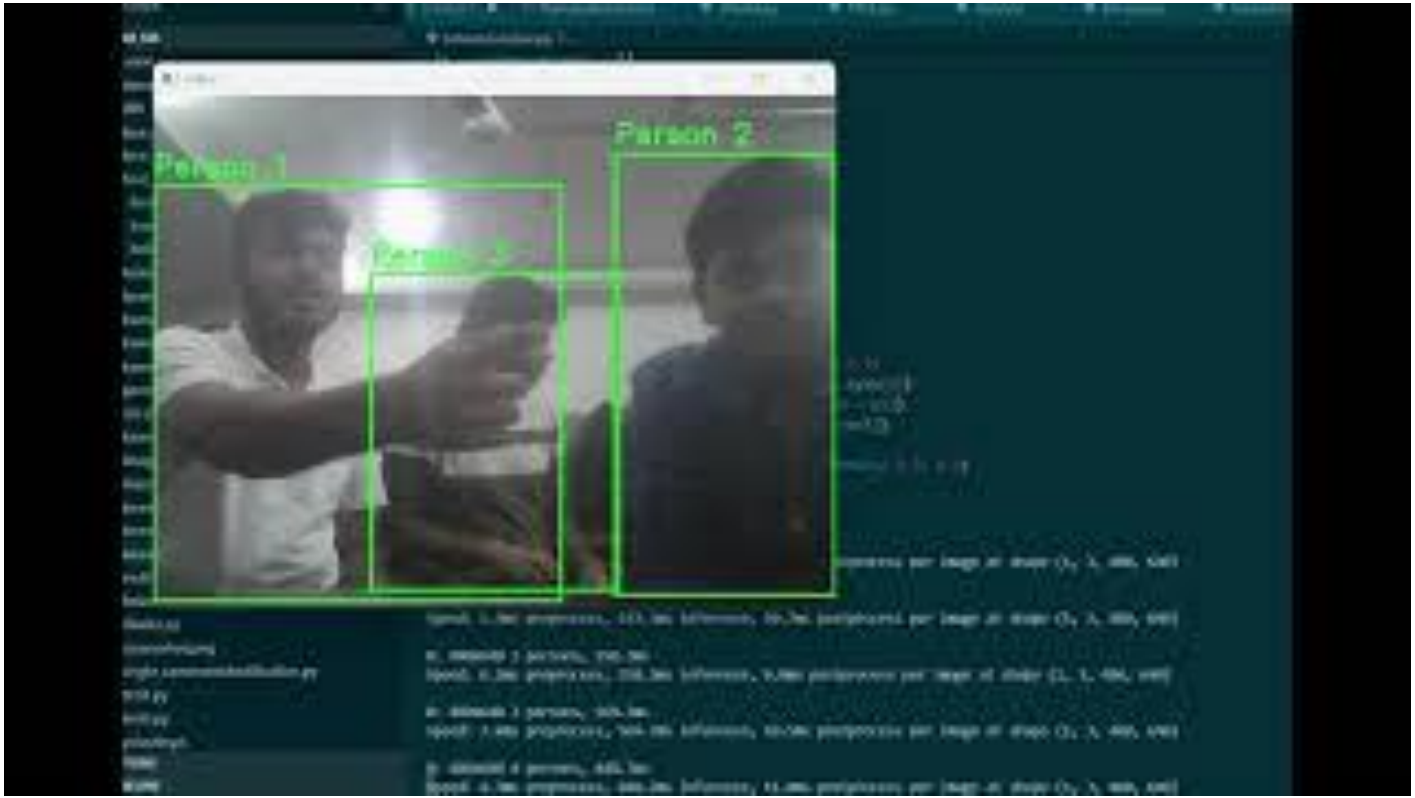
Demo - Gaze Detection

Demo - Relative Velocity



Demo - Face Detection

Person identification: Torso



Demo - Pose Detection

Phase 2

2 Camera Systems
[StereoVision]

Demo - Person Re Identification

1 Joint Set has:
17 Key Points
17 RGB Tuples
1 Bounding Box

1 Frame has:
Multiple Joint Sets

1 Camera Feed has:
A series of Frames

The RGB Matrix:
Map the Sum of the Euclidean Distance between each common keypoint between two joint sets of two separate frames.

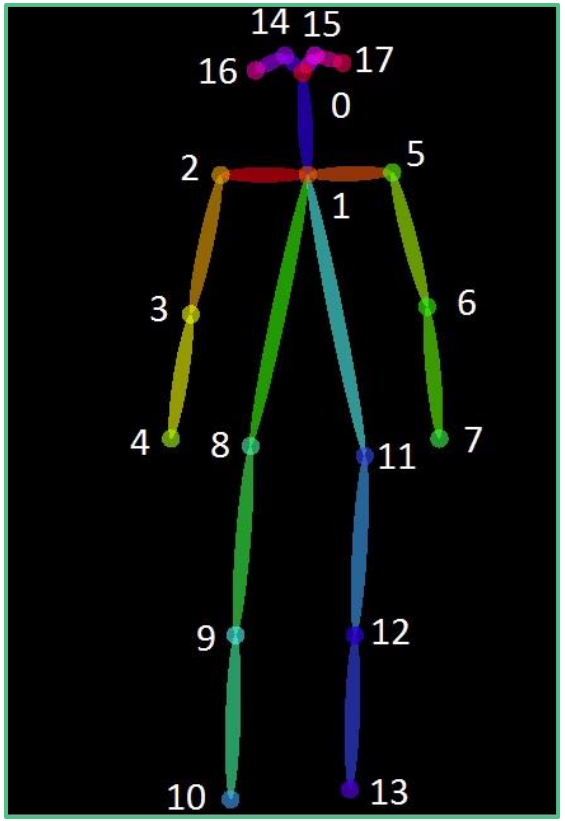
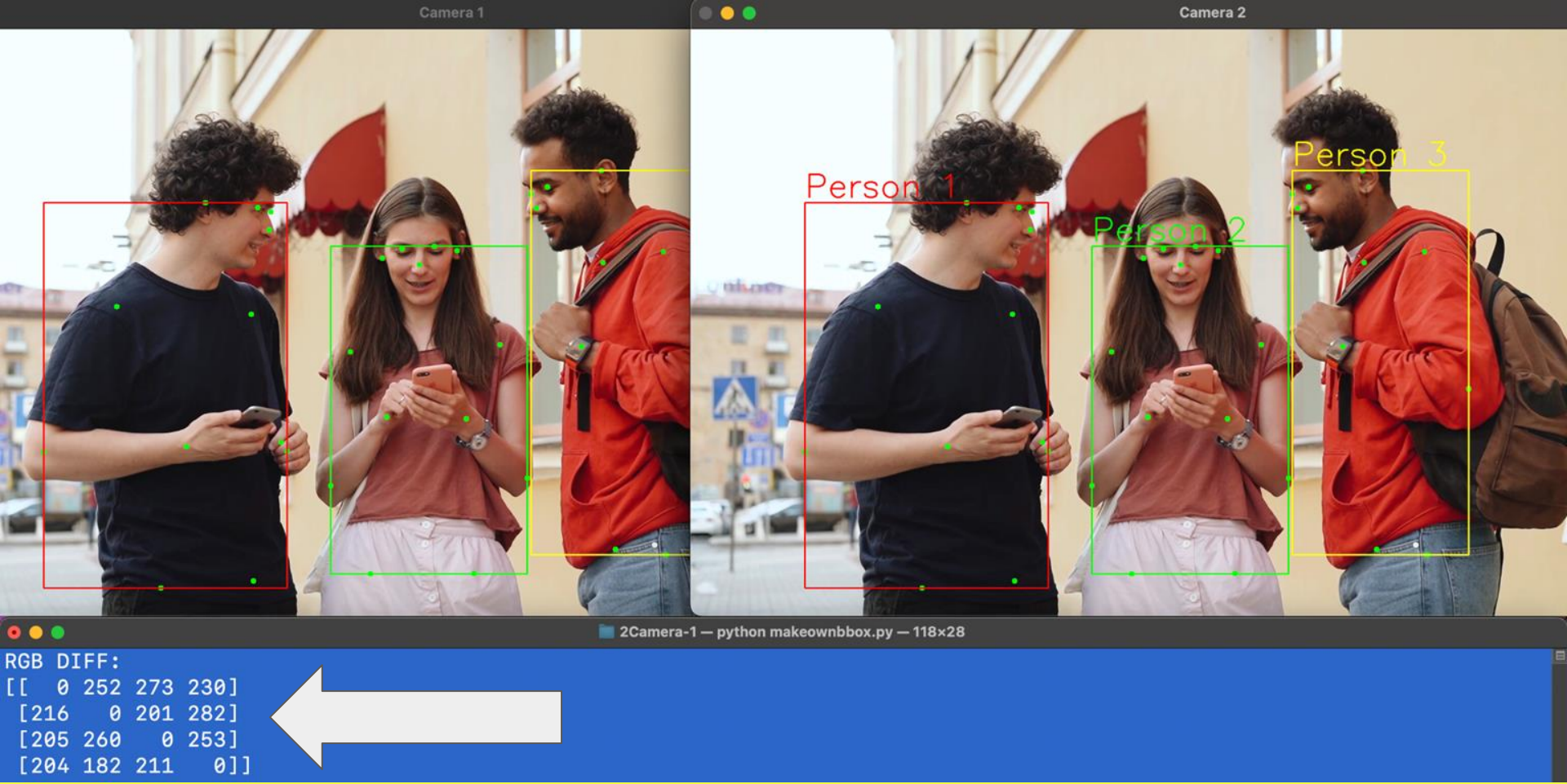


Fig. Key Points of a Joint Set



Gives us the two persons with the least RGB difference. Thresholding eliminates least worst option issue.


[986.0, 635.5,
[986.0, 635.5,
MATCH UP BOX
[725.0, 326.0,
Labels
[1212.0, 569.0
[1212.0, 569.0
MATCH UP BOX
[1869.0, 224.0
Labels
[253.0, 839.0,
[253.0, 839.0,
MATCH UP BOX
[241.0, 761.0,
0: 384x640 4 p
Speed: 2.4ms p
shape (1, 3,
stprocess per image at

Face - Recognition and Identification

Real-Time Face Recognition and Identification with Individual ID

- Main Goals:
  - Identify the face using joint sets, key points, RGB matrix and bounding box on live camera feed
  - Optimize frame processing while maintaining high accuracy in face detection and recognition.
  - Running the face-recognition on multiple cameras

Face- Recognition and Identification

Real-Time Face Recognition and Identification with individual ID

Full Code:

```python
import face_recognition
import cv2
import pickle
import os

os.environ['QT_QPA_PLATFORM'] = 'xcb'

KNOWN_FACES_DIR = 'Python Programs/Asim Tewari/Week 2/Known_faces'
#UNKNOWN FACES DIR = "unknown faces"
TOLERANCE = 0.6
FRAME_THICKNESS= 3
FONT_THICKNESS = 2
MODEL = "hog" #cnn

video1 = cv2.VideoCapture(0)
video2 = cv2.VideoCapture(4)


known_faces = []
known_names = []

for name in os.listdir(KNOWN_FACES_DIR):
    for filename in os.listdir(f"{KNOWN_FACES_DIR}//{name}"):
        #image = face_recognition.load_image_file(f" {KNOWN_FACES_DIR}/{name}/{filename}")
        #encoding = face_recognition. face_encodings (image)[0]
        encoding = pickle.load(open(f"{KNOWN_FACES_DIR}//{name}//{filename}", "rb"))
        known_faces.append(encoding)
        known_names.append(int(name))

if len(known_names) > 0:
    next_id = max(known_names) + 1
else:
    next_id = 0
print("processing")

while True:
    #image = face_recognition.load_image_file(f"{UNKNOWN_FACES_DIR}/{filename}")
    ret1, image1 = video1.read()
    ret2, image2 = video2.read()

    images=[image1,image2]
    n=0
    for idx, image in enumerate(images):
        if (ret1 and n//5==0) or (ret2 and n//5==0):
            locations=face_recognition.face_locations(image,model=MODEL)
            encodings = face_recognition.face_encodings(image, locations)
            #image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            for face_encoding, face_location in zip(encodings, locations):
                results = face_recognition.compare_faces (known_faces,face_encoding, TOLERANCE)
                match = None
                if True in results:
                    match = known_names[results.index(True)]
                    # print(f"Match found: {match}")
                else:
                    match = str(next_id)
                    next_id+=1
                    known_names.append(match)
                    known_faces.append(face_encoding)

                    folder_path = f"{KNOWN_FACES_DIR}//{match}"

                    if os.path.exists(folder_path):
                        while os.path.exists(folder_path):
                            match+=1
                            next_id+=1
                            folder_path = f"{KNOWN_FACES_DIR}//{match}"

                    os.mkdir(folder_path)
                    pickle.dump(face_encoding, open(f"{folder_path}//{next_id}.pkl", "wb"))

                    face_img = image[face_location[0]:face_location[2], face_location[3]:face_location[1]]
                    cv2.imwrite( f"{KNOWN_FACES_DIR}//{match}//{match}.jpg", face_img)

                top_left = (face_location [3], face_location[0])
                bottom_right = (face_location[1], face_location[2])
                color = [255, 255, 255]
                cv2. rectangle(image, top_left, bottom_right, color, FRAME_THICKNESS)

                top_left = (face_location [3], face_location[2])
                bottom_right = (face_location[1], face_location[2]+22)
                cv2. rectangle(image, top_left, bottom_right, color, cv2.FILLED)
                cv2.putText(image, match, (face_location[3]+10, face_location [2]+15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0),1, cv2.LINE_AA)
        else:
            n+=1

    cv2.imshow("Camera 1", image1)
    cv2.imshow("Camera 2", image2)

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

video1.release()
video2.release()
cv2.destroyAllWindows()
```

Face- Recognition and Identification

Real-Time Face Recognition and Identification with individual ID

Highlights of Code:

```
import face_recognition
import cv2
import pickle
import os
```

Libraries

This saves the encoding of the face append that encoding into known_face variable

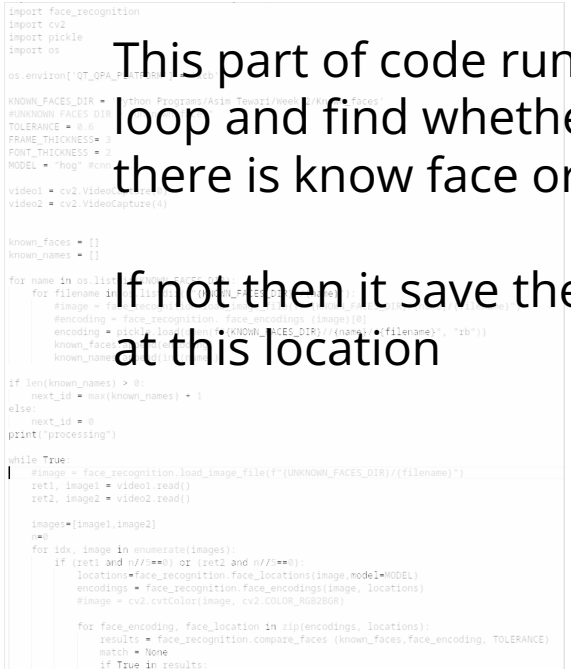
Face- Recognition and Identification

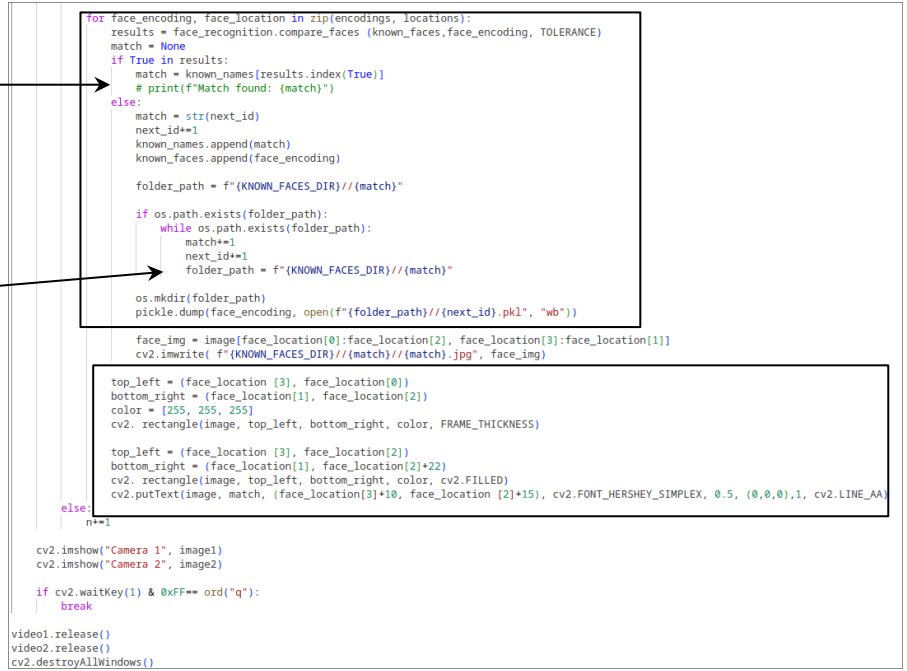
Real-Time Face Recognition and Identification with individual ID

Highlights of Code:

This part of code runs a loop and find whether there is know face or not.

If not then it save the face at this location





Face- Recognition and Identification

Real-Time Face Recognition and Identification with individual ID

Output:

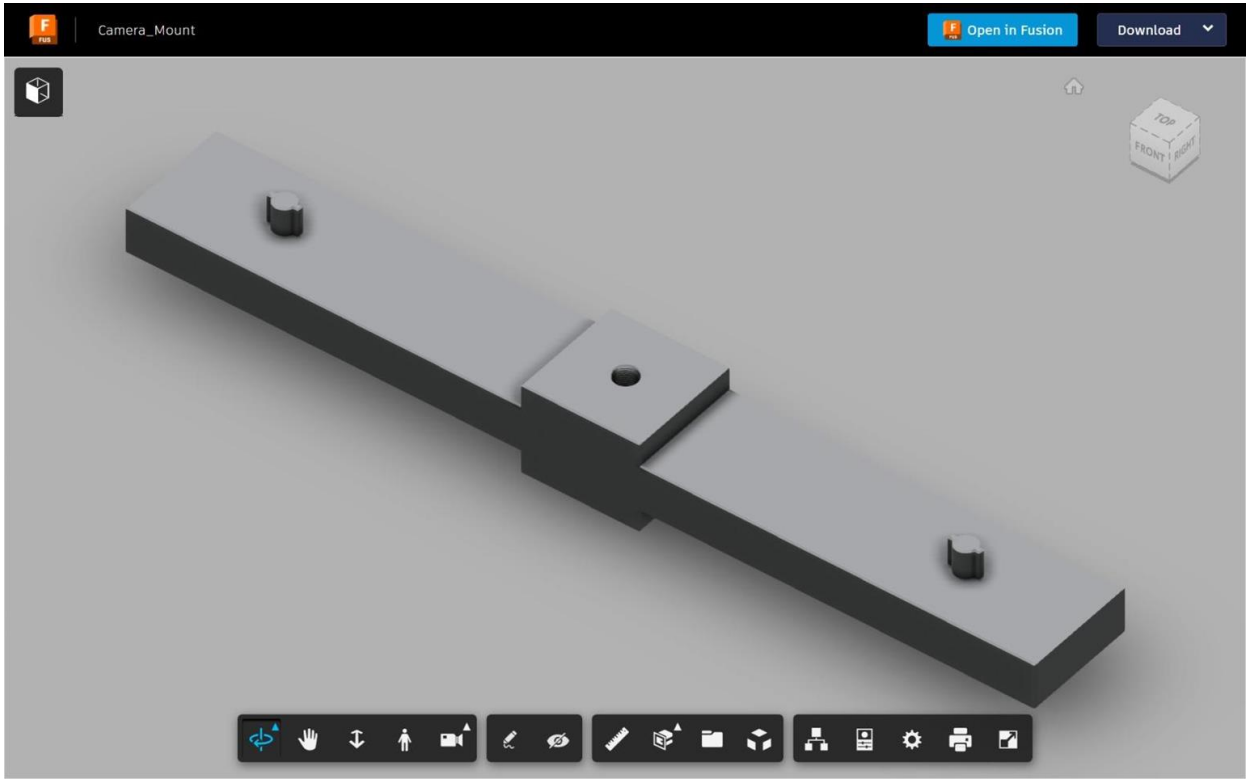
Face Tracking

Tracking of face once it is identified

- Goal
  - Once face is identified we should track that face without putting it in the model
  - To decrease the computation power in face recognition
  - To use less memory to store the data of faces

Demo - Depth Perception Using Re Identification [Parallel Identical Cameras]

3D Printed Camera Rig Model



Future Scope

1. Action Detection
2. Object Association
3. Logging Real Time Data
4. Depth Perception Using Re Identification [NonParallel NonIdentical Cameras]
5. Merging all modules into one lightweight software