**Computer Vision and Image Processing CSE473/573-B**

**Raghav Singh Chauhan**

**UB person number: 50486848**

**UBID: raghavsi**

**Project Name: Playing Game subway surfers using pose detection**

**Application Details:** Using PyAutoGUI and the Python Mediapipe module, I have created an application that does real-time posture detection. The software's objective is to recognize particular stances in the video game Subway Surfers and then execute different keyboard commands in response to the recognized pose.

**Input:**  Various Poses from the webcam field of the camera

Jump, Crouch, Left, Right

**Output:** The player in the game moving according to the detected pose.

**State of the Art:**  The state of the art in pose detection consists of various models being developed that can predict with reasonably good accuracy the poses.

**My Contribution to the project:** I have added triggers to the game from the pose detected from the mediapipe pose detection library in python using pyautogui to trigger the various keypresses(‘UP’, ‘DOWN’, ‘LEFT’, ‘RIGHT’)

**Approach**

**Details of the algorithm used and their pros and cons:**

The mediapipe library in python is a very popular library for pose detection. It is quite user friendly as it does not involve complicated pose detection algorithms to be implemented and the end user can just use the functions for pose detection.

Under the hood the mediapipe library is implemented as:

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames utilizing our [BlazePose](https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html) research that also powers the [ML Kit Pose Detection API](https://developers.google.com/ml-kit/vision/pose-detection). Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas our method achieves real-time performance on most modern [mobile phones](https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md#mobile), [desktops/laptops](https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md#desktop), in [python](https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md#python-solution-api) and even on the [web](https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md#javascript-solution-api).

The solution utilizes a two-step detector-tracker ML pipeline, proven to be effective in our [MediaPipe Hands](https://github.com/google/mediapipe/blob/master/docs/solutions/hands.md) and [MediaPipe Face Mesh](https://github.com/google/mediapipe/blob/master/docs/solutions/face_mesh.md) solutions. Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed, i.e., for the very first frame and when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame’s pose landmarks.

Source : <https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md>

**Pros and Cons of algorithms of mediapipe pose detection library:**

**Pros:**

1. **Accuracy:** In general, Mediapipe's pose detection algorithm yields accurate results, especially when there is adequate lighting and little background distraction. It can successfully recognize important body parts, including joints and landmarks.
2. **Real-time performance:** Because of the algorithm's real-time pose detection capabilities, it is appropriate for applications like gaming that demand quick response times. In order to attain quick processing times, it makes use of effective algorithms and optimizations.
3. **Multi-person pose detection:** The detection of numerous people in a single frame is supported by Mediapipe. This makes it possible for users to engage with one another in apps or games using poses. Although my application does not require multi-person pose detection but still it is a good feature to have in a pose detection library or model.
4. **Pre-trained models:** Pose detection models that have already been trained are provided by the library, which streamlines the implementation procedure. Due to the vast datasets used in their training, these models exhibit strong generalization and performance across a range of body shapes and poses.

The self-coded aspects of the algorithms that are being implemented such as triggering various key presses is relevant to the project as it helps in giving various triggers to the player in game to avoid obstacles and helps add to great user experience. Also the pose detection is a good way that people can do physical activity while enjoying the immersive experience of the video game.

I took some aspects of the algorithms from online sources. Mainly I referred to the following github repository that uses pose detection using mediapipe to detect various poses.

**Reference:** <https://github.com/pengzhangzhi/AI-assisted-subway-surfers/tree/main>

**What aspects of the algorithms have you coded on your own?**

I have used media pipe python pose detection library and consulted the github repository to implement the pose detection based on the poses given by user. I have also added keyboard press triggers based on the various poses detected using pyautogui.

**Demonstrate an understanding of how the self-coded aspects of the algorithms contribute to the project.**

The keyboard press triggers helps in ensuring that the intended move is being performed in the game and within reasonable low-latency.

**What aspects of the algorithms have you used from the online resources? Cite them appropriately.**

I have referred to github repository that implements pose detection in game subway surfer.

Reference: <https://github.com/pengzhangzhi/AI-assisted-subway-surfers/tree/main>

I have referred to this repository as well as mediapipe and pyautogui docs.

Citation: <https://github.com/pengzhangzhi/AI-assisted-subway-surfers/tree/main>

<https://developers.google.com/mediapipe>

<https://pyautogui.readthedocs.io/en/latest/>

**Experimental protocol:**

I did not feel the need to use any datasets for the model as mediapipe library in python is trained on several thousands of images of people to get various poses with good accuracy. Regarding computing resources, my project requires a computer with a webcam and decent processing power to run the Python code using Mediapipe and Pyautogui libraries.

**Evaluation of success:**

My approach to evaluation was to play the game with all poses a certain number of times and see whether the player in the game was able to detect my pose with a low latency.

I tried the pose detection library with multiple images of mine in different poses. I tried around 20 times the poses. The accuracy was the following:

Jumping: 15/20

Crouching: 17/20

Left: 20/20

Right: 20/20

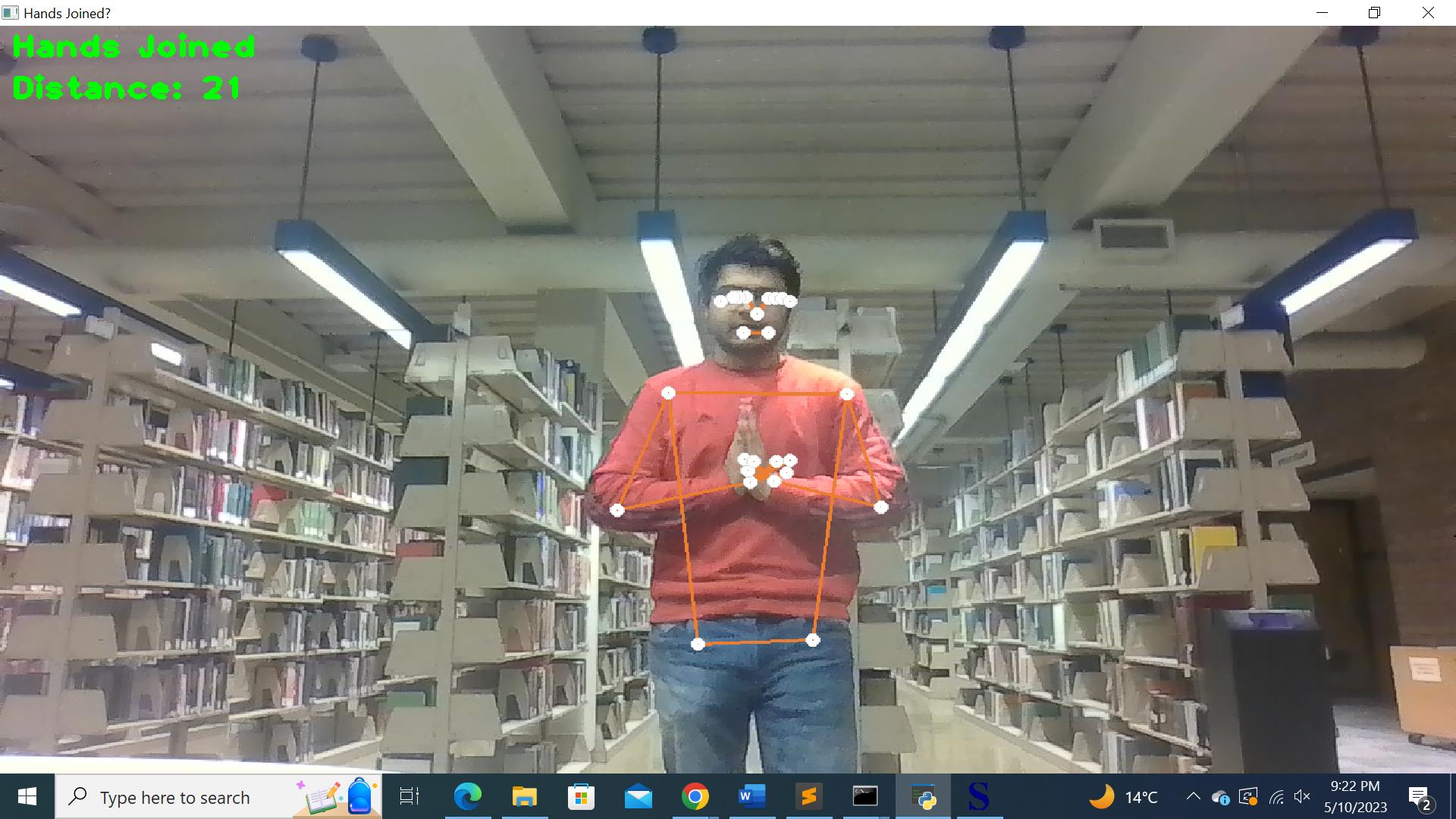
The slightly less accuracy in jumping and crouching is due to lapse in the timing of these poses as it requires slightly extra effort which can delay the model prediction especially when playing game at high frame rates.

**Results**

**Game can be accessed at https://poki.com/en/g/subway-surfers**

Qualitative results:

Game Start (Join Hands):



Hands NOT Joined:



Image 0 Idle State



Image 1 (Jumping)

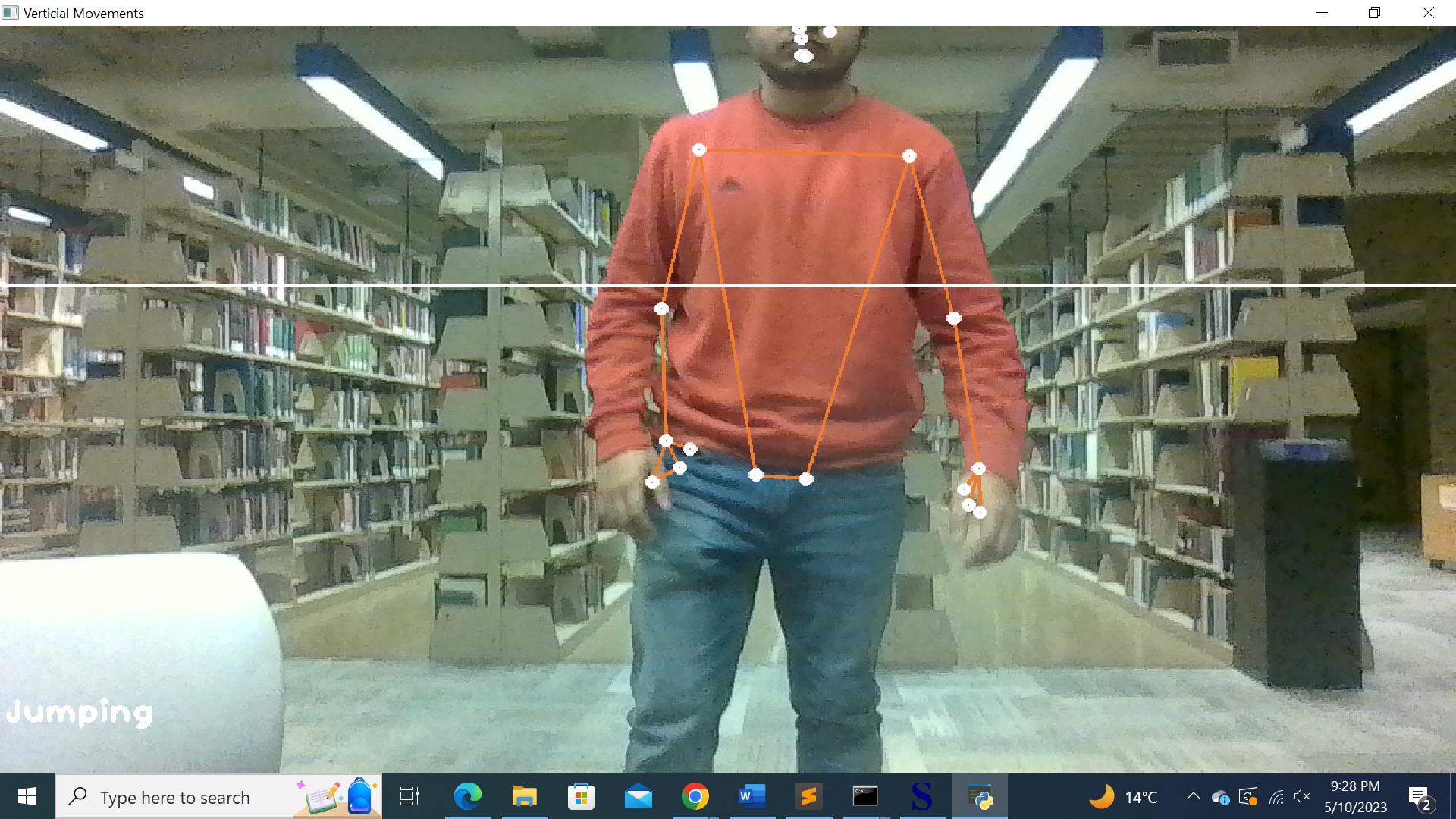


Image 2 (Crouching)

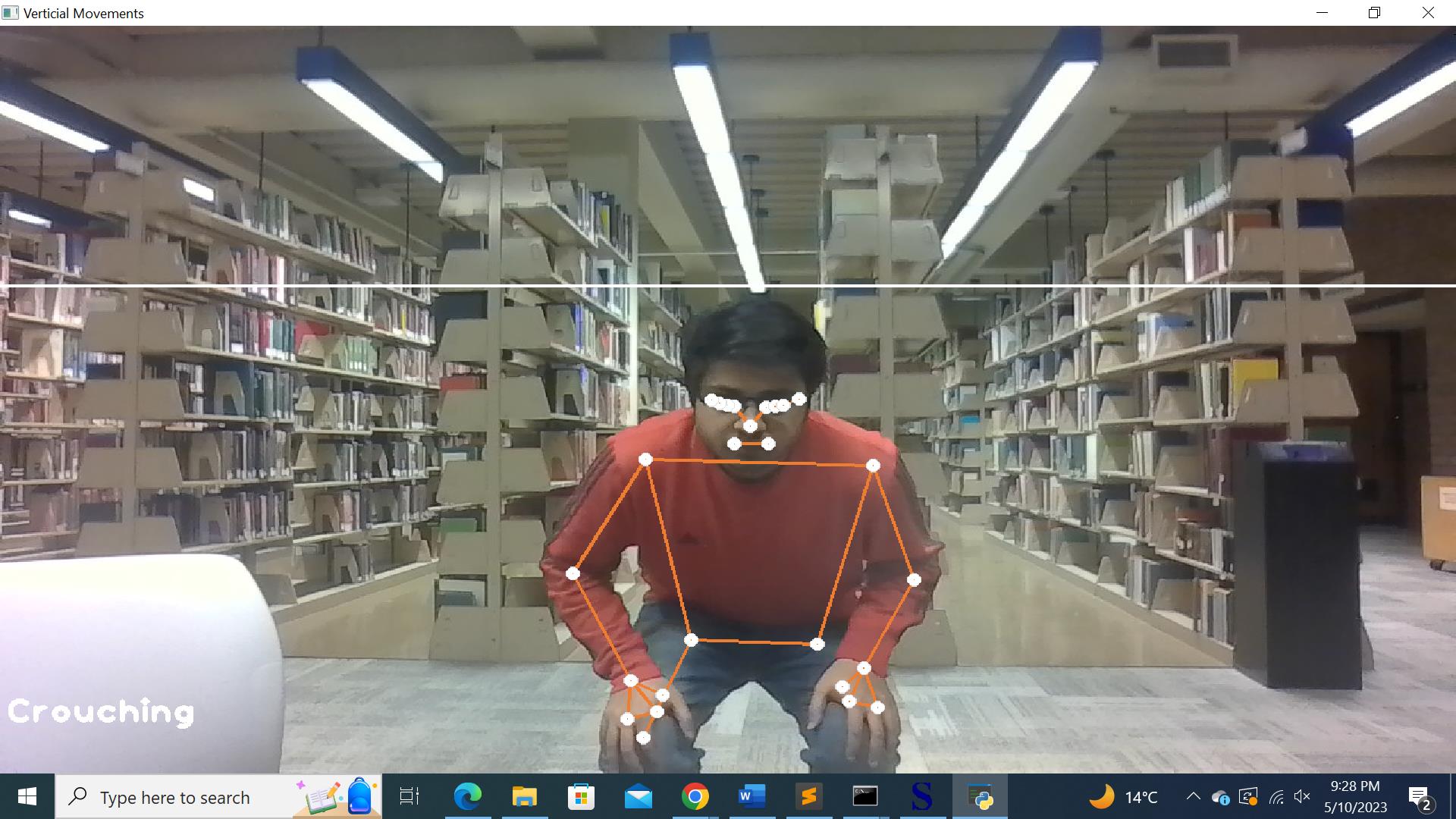


Image 3 (Left)

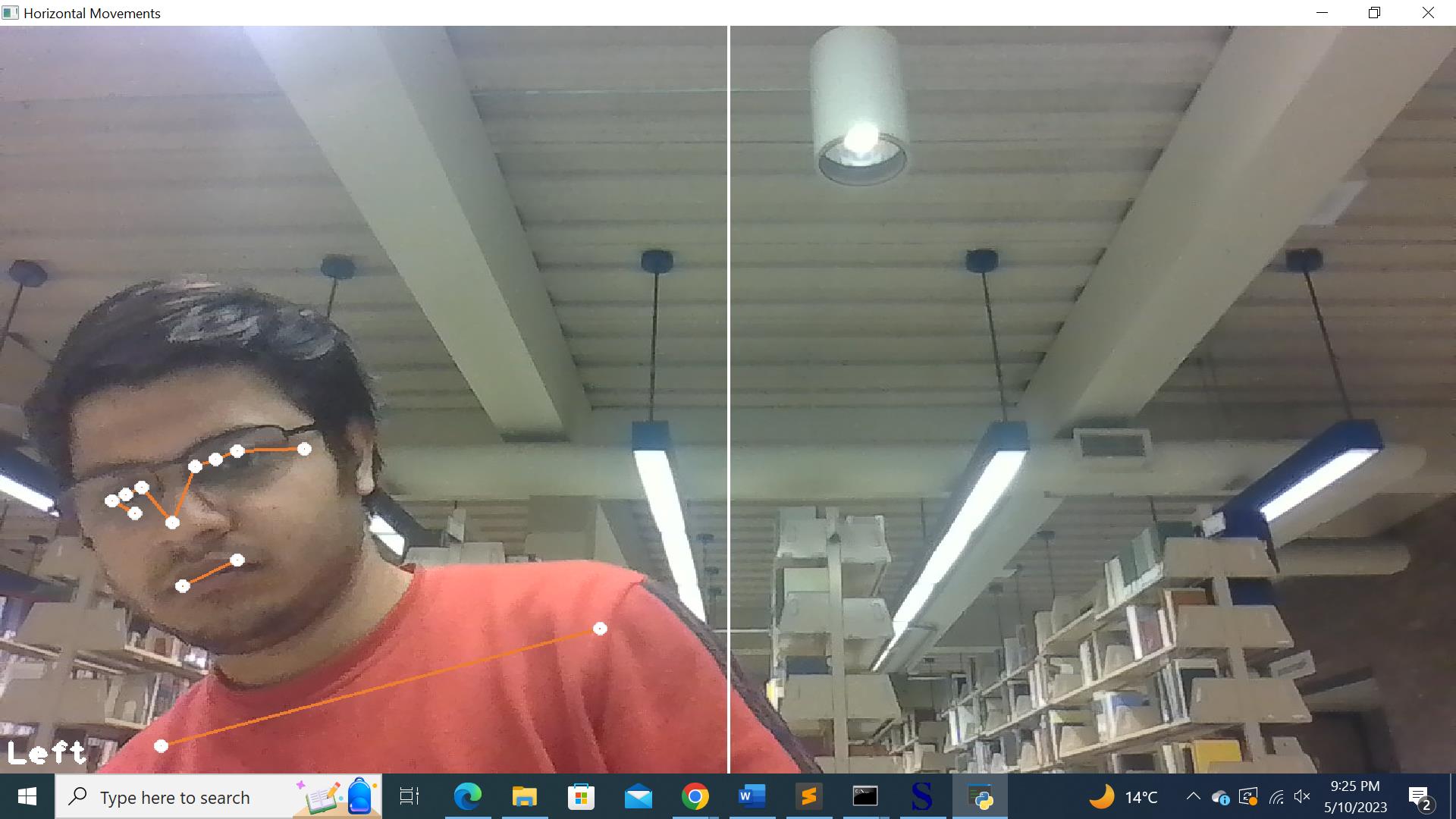


Image 4 (Right)

A person in a library

Description automatically generated with low confidence

**Interpretation of results**

Kindly check the video for the performance of pose detection while playing the game.

**Analysis:**

**Advantages** of the algorithm used for developing the application:

* The player experience may be improved by the pose detection method, which can offer a more intuitive and natural way to interact with the game.
* Pose recognition may lessen the chance of repetitive strain injuries and other physical pain brought on by standard keyboard/mouse inputs.
* The algorithm may be altered to recognize particular stances or motions, opening up new gaming elements and dynamics.

**Limitations** of the algorithm used for developing the application:

* Lighting, background clutter, and other elements that might obstruct the recognition of important body parts on the user may have an impact on the precision of the pose detection algorithm.
* The algorithm's versatility and adaptability may be constrained by its inability to recognize positions that do not exist in the training data or that are executed in an unconventional manner.
* The method may not be accessible on all devices due to the high computing power needed to perform it in real-time.

**Discussion and lessons learned**

Through this project, I gained knowledge on how to create a real-time pose detection system and utilize it to interface with a game using Python modules like PyAutoGUI and Mediapipe. I got experience using computer vision tools and applying machine learning models in real-time scenarios.

Future applications of the abilities and knowledge gathered from this project include gaming, fitness tracking, and human-computer interaction. Pose detection, for instance, can be used in fitness monitoring programs to watch a user's form and give them immediate feedback. Additionally, it enables more natural and intuitive interactions between humans and digital objects in virtual and augmented reality applications.

Overall, working on this project has given me a better grasp of computer vision, its uses, and the practical issues related to creating real-time applications. I intend to keep expanding on this knowledge and investigating fresh applications for pose detection to improve human-computer interaction.

**Bibliography**

[**https://developers.google.com/mediapipe**](https://developers.google.com/mediapipe)

[**https://opencv.org/**](https://opencv.org/)

[**https://www.python.org/**](https://www.python.org/)

[**https://github.com/pengzhangzhi/AI-assisted-subway-surfers/tree/main**](https://github.com/pengzhangzhi/AI-assisted-subway-surfers/tree/main)

[**https://pyautogui.readthedocs.io/en/latest/**](https://pyautogui.readthedocs.io/en/latest/)

**https://poki.com/en/g/subway-surfers**

**Video editing : https://www.Clideo.com**