Human Motion Interface for Video Game Control

1. Problem Description

The human-computer interface behind many modern video games, using either handheld joystick controllers or keyboard and mouse input, is an example of a typical human input device for computers, monitors, televisions, etc. There are two major drawbacks to traditional human input systems that are often overlooked: the input needed to achieve a certain effect has little to no correspondence to a physically-intuitive set of motions, and more importantly, inputs are heavily dependent on the user's fine motor skills. Fine motor skills used to interact with modern technology are heavily affected by factors such as muscular coordination disorders and old age. Computer vision, particularly the processes of motion tracking and human pose estimation, may allow us to create human-computer interfaces that link effect to physically-intuitive human motion inputs with little dependence on fine motor ability. We propose a vision-based computer-input system that maps gestures and motion of the user's body into computer input. We propose a car-racing simulator as a test subject for our system, and aim to intuitively control the motion of a player's virtual car with human body motion as the only action. Such a system would be instrumental in improving the accessibility of modern technology, with implementations far beyond the scope of computer games.

2. Related Work

2.1. Human Pose Estimation

Some of the largest components of our system fall into the category of so called Human Pose Estimation or the ability to properly:

- Identify users as sources of input, from a video stream
- Segment the user's body to isolate relevant portions of data
- Compute location information about the user's body from relevant portions of data.

The area of Human Pose Estimation is widely studied. Robust, optimized solutions exist that can help us with this step. One popular solution is derived from [1], and open-source code for the so called openpose library shows promising results for real-time human body segmentation.

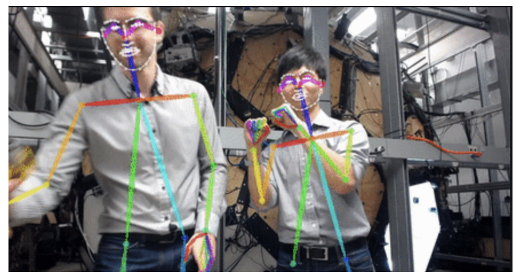

Figure 1. A demonstration of openpose performing human body segmentation.

2.2. Gesture Identification/Motion Tracking

[1] and Openpose give us promising ways to identify locations of human body features (hands, arms, etc.) in a video stream. However, this is not enough to categorize a user's motion as a particular input. To be able to extract feature motion over time as a useful input to our car racing video game, we will have to implement motion tracking over a video stream.

A number of guides exist for implementing feature tracking over video. [3] and [2] demonstrate simple object tracking across video frames, and [2] demonstrates a system shown to be accurate with motion of hands. Implementing a similar system for tracking arbitrary features of the human body obtained from human pose estimation should provide adequate results.

3. Methodology

Our human computer interface will consist of three major steps: Pose Estimation, or feature location, Feature Tracking and classification of motions, and video game interfacing. Each one is detailed below. All code will be written in Python.

3.1. Pose Estimation

The first step of our human computer interface pipeline is the pose estimation step, in which we use openpose to locate certain key features of the user. What constitutes a key feature will be determined when the set of video game controls we must mimic is properly defined. Possible key features include the location of the user's wrists, elbows, or the angles of various limbs.

During this step we will also determine the computational cost of openpose and make decisions on how to proceed based on the results. If openpose is able to run with sufficiently low latency, we move on to feature tracking, but if openpose demonstrates a high latency, we will explore other methods of pose estimation, or methods of improving the performance of openpose using by decreasing framerates or image resolutions.

3.2. Feature Tracking

Obtaining the locations of various important features is only the first step towards identifying physically-meaningful actions of the user. In order for the user to be able to signal various commands within the video game, we must track the previously obtained features over time and classify the user's behavior into one of a few possible actions. For example, mimicing a car racing game, the set of possible actions could be {No command, accelerate, decelerate, steer}. Each of these commands or classifications will also have a value associated with it corresponding to its magnitude, which will also be obtained directly from feature locations. For example, a steering command mapped to the user rotating their torso would have a steering value associated with it derived from the angle through which the user has turned.

3.3. Game Interface

The third and final major step of our methodology is to turn the classifications obtained from the Feature Tracking step and feed them to a video game as user input. We will simply map each corresponding classification from step 2 to an "intermediate control" which is simply the joystick/keyboard/mouse input that would produce the same output from the game, and implement simple keyboard control with Python to feed that input to the video game.

4. Timeline

We adopt an eight-week timeline for the development and documentation of this project, outlined more specifically below:

- Week 0: Configure openpose to identify feature points from front and/or side cameras.
- Week 1: Perform any necessary optimizations to run openpose with as low latency as possible.
- Week 2: Finalize the set of trackable and detectable gestures, and map each one to a particular keyboard mapping in a physically-intuitive way.
- Week 3-4: Implement a feature tracking system, and confirm that each gesture accurately corresponds to a specific function with sufficient accuracy.
- Week 5: Interface detections with video game.
- Week 6: Optimize vision video game interaction for low latency.
- Week 7: Write paper describing project methodology, and explore future works inspired by this project.

References

[1] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[2] Charlie Liu. Gesture detection with a raspberry pi. 2017.

[3] Adrian Rosebrock. Simple object tracking with opencv. 2018.