Human Pose Estimation for Video Game Control

Rakesh Johny, Tom Li, Aditya Narayanan, Albert Xia

Abstract

Current methods of interacting with computers are flawed in a couple of key ways: they fail to map physically-intuitive motions to their computer control counterparts, and they rely heavily on a user's fine-motor skills, which are heavily impacted by factors such as muscle coordination disabilities and old age. In this paper, we propose a system of computer control through gesture tracking via real-time human pose estimation on an embedded device, which is capable of addresssing these issues, by mapping general physically-intuitive motions into computer control. Furthermore, our device boasts low latency in the human-computer interaction, and is minimally intrustive to the computer being controlled. We test the viability of our system as a computer-control device by playing two video games using only gestures.

1. Introduction

The human-computer interfaces behind many modern video games use either handheld joystick controllers or keyboard and mouse input, which are examples of 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 human pose estimation and motion tracking may allow us to create human-computer interfaces that link computer control to physically-intuitive human motion inputs with little dependence on fine motor ability or additional physical input devices. We propose a vision-based computer-input system that maps gestures and motion of the user's body into computer input. We test the system's effectiveness with the playability of two simple video games as metrics. In order to demonstrate the versatility of our phased-approach to gesture-based control, we select a car-racing game and the Google Chrome Dino Racer game as test video games. Such a system, when expanded to replicating general keyboard + mouse control, 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 the location, in image coordinates, of keypoints on the user's body.

The area of Human Pose Estimation is widely studied. There are a wide array of robust, optimized solutions for gathering 2D image coordinates of keypoints from images of users.

2.1.1 Openpose

One popular solution to realtime human pose estimation, derived from [1], uses Part Affinity Fields (PAFs) to learn to assosciate body parts with individuals in the image, and open-source code for the so called openpose library shows promising results for real-time human body segmentation.

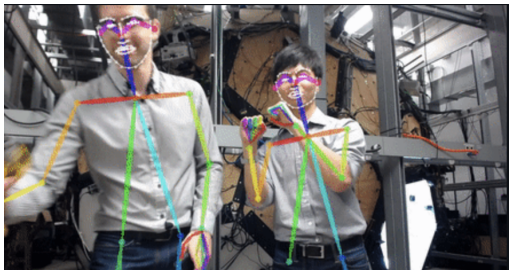

Figure 1. A demonstration of openpose performing human body segmentation.

2.1.2 MoveNet and PoseNet

PoseNet and its more recent counterpart MoveNet are fully convolutional neural network pose estimators built on existing ResNet or MobileNet backbones (citation here). Benchmarks of PoseNet and MoveNet show that it is viable to run either estimator in real time on embedded devices with appropriate model quantizations (citation here), with PoseNet being slightly faster when compared to MoveNet at identical levels of quantization. In comparison's with Openpose, PoseNet/MoveNet are shown to perform slightly better on instance segmentation of the person class from the COCO dataset.

2.2. Gesture Identification/Motion Tracking

Openpose and MoveNet/PoseNet 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 command to a computer system. 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. We drew on [3] and [2] in developing a tracking algorithm for the identification of a human user ducking or jumping from a keypoint cloud.

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. Results

5. Future Works

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.