Gameplay with moving hand gestures

Computer Vision(CS-763)

1 Introduction

With the advancement of virtual and augmented reality, gaming industry has evolved so much over a short period of time. Major tech companies like steam, facebook, Microsoft etc are coming up with exiting VR gaming technologies.
Following the similar zeal, we have developed a simple game controller which takes input from **moving hand gestures**. The project is developed on the principles of Computer Vision and Machine Learning.

2 Related Work

Motivation of this project came from ASL fingerspelling Interpretation[1], where they have developed a solution using which one can interpret the American Sign Language alphabets from images of hand gestures. Static hand images are used in this work. To extract the features, few image processing techniques are used such as *The histogram of centroid distances(HOCD)* and *gabor filters*, further machine learning is used to learn and predict the gestures.
Our work also follows a similar kind of pipeline, but instead of using static images, we have used moving hand videos as input. We have also used motion information to improve the predictions.

3 Google's MediaPipe Hands Solution

Google had came up with this great set of solution in the Computer Vision field which they termed as **MediaPipe**. One such solution is called **MediaPipe Hands**[2]
It is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. The keypoints capturing process doesn't depends on the lighting and the texture conditions of the images, as the ML model used here is very robust. It returns 21 keypoints landmark location information for every frame. We have used this information in our favour.

4 Our Learnings

While working on the project we have faced some challenges. These are listed below.

4.1 Capturing the flow

The first challenge was to use the flow information to increase the accuracy of the model. As we are not working with static images, we are working with a set of 20 video frames for every gesture, obtaining the motion of hand keypoints was easy to obtain. We have used the Lucas-kanade method(sparse flow) to obtain the flow information for these keypoints. Then we obtained the magnitude and angle information from this flow and finally we have appended this data for every frame along with the keypoint coordinate information. This finally made our feature set an array of **105 features** per frame. Out of which 63 features are the x,y,z coordinates of the 21 keypoints, and remaining 42 features are the flow magnitude and angle information for every keypoint.

4.2 Finding best machine learning module for learning and predictions

We have looked at some of the literature which works on moving frames features. Most of them uses a **LSTM architecture** due to the temporal similarity between the frames. We have used bidirectional lstm layers instead of unidirectional as it was giving higher test accuracy. But when we tried it in real time, it was not working very well.
Then we opted for **Decision Tree model**, with `max_depth=10`. This was also giving high test accuracy, and surprisingly it was also working fine with real time inputs.

4.3 Inferring from the model

In order to develop the end to end solution we had to combine the gesture prediction module with the user input capturing module, then based on the predictions we need to emulate the actions in the game. It is in this part we faced most of the challenges.
Firstly as the model is trained with 50 gesture samples per gesture, and for every gesture we are considering 20 frames. But while inferring, this data capture is not always perfect because we don't know when the user has started making the gesture and when he stopped. This imperfect data capture can lead to some inaccurate predictions.
Other challenge we faced was in emulating the key presses based on the gesture predictions. For e.g. when user makes a gesture of `forward` the system should simulate `key press 'w'`, similarly for all the other gestures namely `backward, left, and right`. For accomplishing this task we have used **pynput** library in python. The other major issue was, even when the hand is not moving (i,e, the

user is not making any gestur and keeping the hand still) then also our model is predicting some best suited gesture from the database of gestures. This is not ideal, to overcome this issue, we have again used the *motion* information of the frames while inferring. We have neglected the inference command when the hand motion is not greater than some *threshold* value.

5 Conclusions

We feel that the solution we developed is completely based on Computer vision techniques and it is not very accurate. The feature extraction part with the help of **mediapipe** library is flawless. But the issues while inferring from the model, and having a smooth control over the game character motion is yet to achieve. We believe that with some optimizations such as using a sliding window kind of approach while data capturing can improve the robustness of the project.
We belive that based on whatever we learnt in the Computer Vision course, we have applied well to this project, and we were able to deliver all the deliverable which we promised to do.

6 References

References

[1] Ewald, Hans Magnus, Ishan Patil, and Shalini Ranmuthu. "ASL fingerspelling Interpretation." University of Stanford, Reports (2016).

[2] "Hands." mediapipe, 23 Mar. 2022, google.github.io/mediapipe/solutions/hands.