Gameplay with moving hand gestures

Course Project for Computer Vision(CS-763)

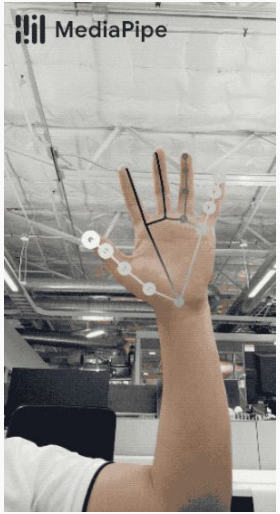
Project Members:

- Sanchar Palit(204070004)
- Animesh(21Q050015)
- Kiran C Ranebennur(21Q050017)

Project contribution split-up

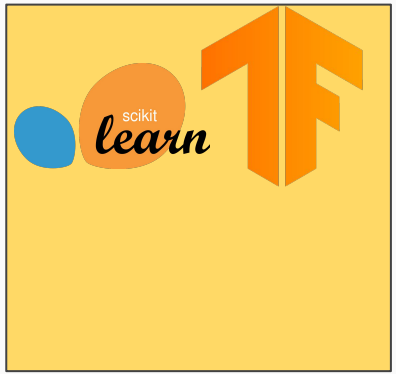
Member	Qualitative Contribution	Quantitative Contribution
Sanchar Palit(204070004)	Designing the machine learning model for prediction of gestures.	33.3%
Animesh(21Q050015)	Feature extraction for training and inference. Working with *mediapipe* data.	33.3%
Kiran C Ranebennur(21Q050017)	Designing the *game* and building the logic to control the game using gesture predictions. Improving the inference technique.	33.3%

Problem Statement



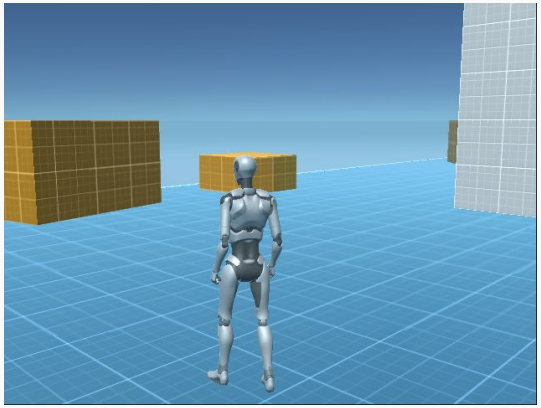
Feature extraction using *MediaPipe* library

Extracted Features



Machine Learning to predict the gestures

Gesture Input



Emulating key presses based on gesture predictions

Problem Statement

Using *moving hand gestures** to control a video game character movements.

Hand features are extracted with the help of Google's MediaPipe Hands solution.

Based on the gesture features, predict the gesture using a ML based pipeline.

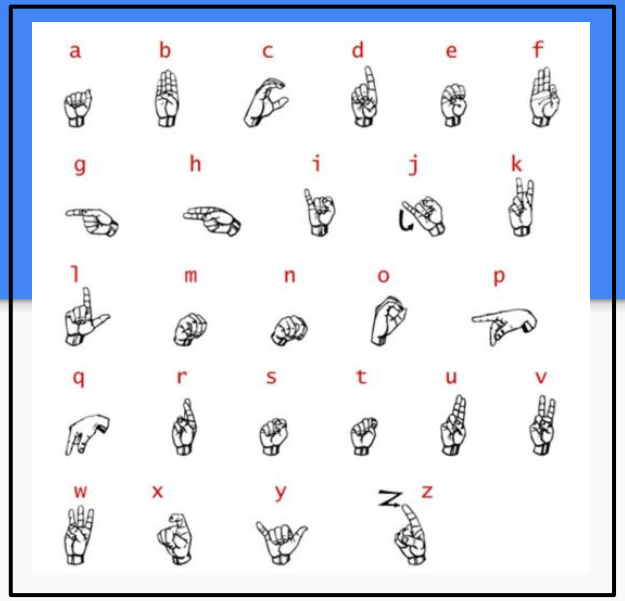
Use the predicted gestures to emulate the video-game inputs to control the character.

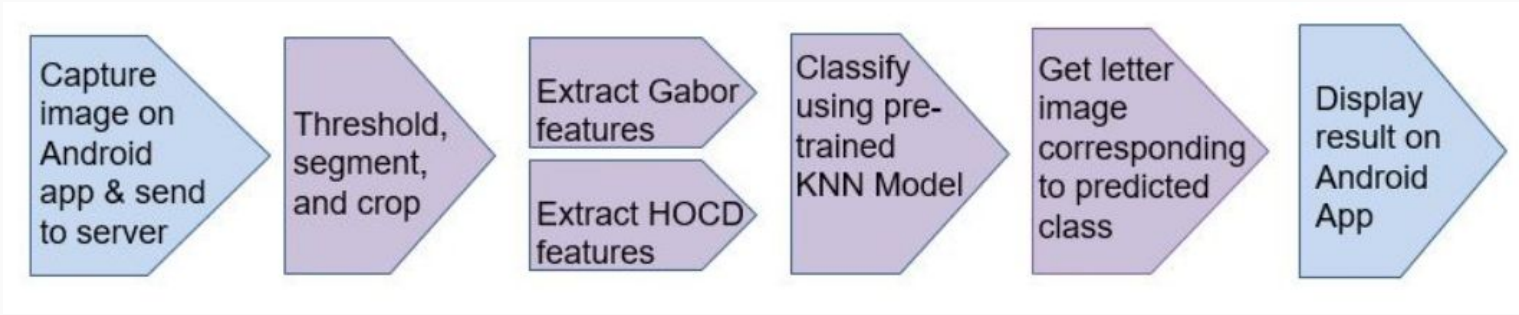
* gesture details are provided later

Motivation

"ASL fingerspelling Interpretation."

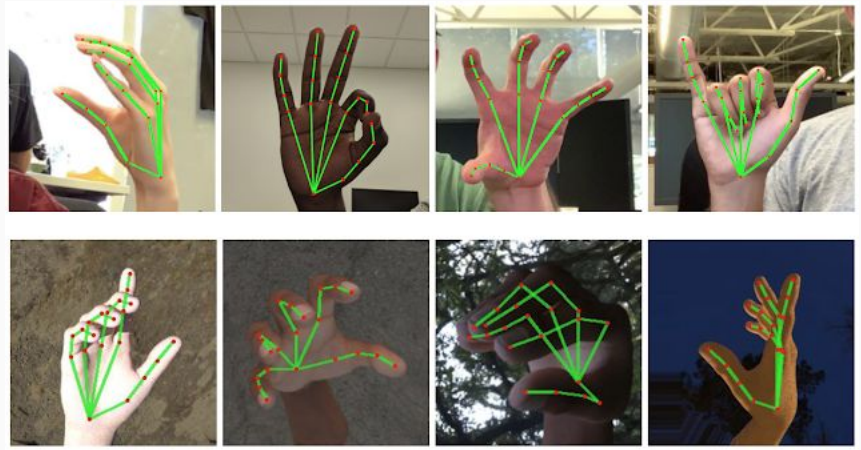
By, Ewald, Hans Magnus, Ishan Patil, and Shalini Ranmuthu. *University of Stanford, Reports* (2016).

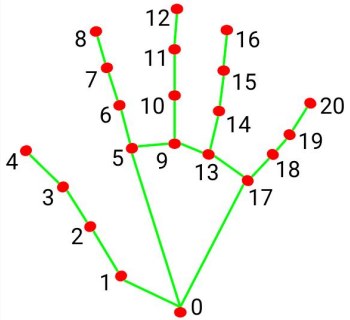




Google MediaPipe Hands

- Takes in hand images and returns hands information based on few key points.
- Returns many information regarding these keypoints such as,
  - <x,y,z> coordinates based on image coordinates
  - <x,y,z> coordinates based on world coordinates
  - Number of hands detected





0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

21 hand key points defined

Related Work

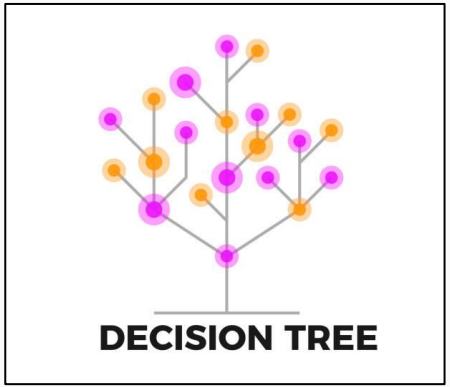
- A somewhat related solution is present here - “Action detection for sign language”
- In this work, *Google MediaPipe* library is used to extract hand gesture information.
- Further a LSTM network is used to learn and predict the gestures.
- The limitation to this implementation is, no **hand motion** information is considered as features.

Only considering **21 hand keypoints coordinates** as features

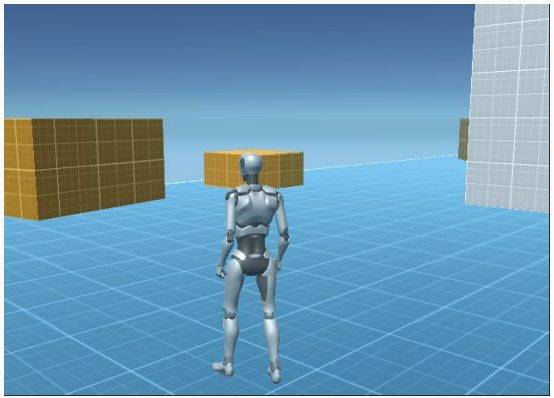
No motion information is considered as features

Extensions to original work

Using **optical flow** information for hand keypoints to *enhance* the *motion gesture prediction*



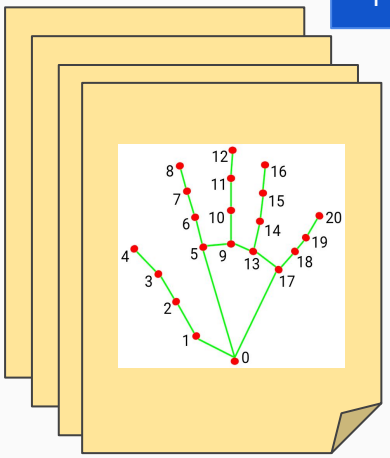
Using **Decision Tree** for learning and predicting from the gesture features



Developed a pipeline to use the gesture predictions to control the in-game character

Methodology

1. Feature Extraction



20 video frames are taken for every hand gesture.

Keypoint coordinates

Keypoint flow magnitudes

Keypoint flow angles

Frame 1 features

1_x 1_y 1_z 2_x 2_y 2_z . . . 21_x 21_y 21_z flow_mag_1 flow_mag_2 . . . . flow_mag_21 flow_ang_1 flow_ang_2 . . . flow_ang_21

Frame 2 features

1_x 1_y 1_z 2_x 2_y 2_z . . . 21_x 21_y 21_z flow_mag_1 flow_mag_2 . . . . flow_mag_21 flow_ang_1 flow_ang_2 . . . flow_ang_21

.
.
.

Frame 20 features

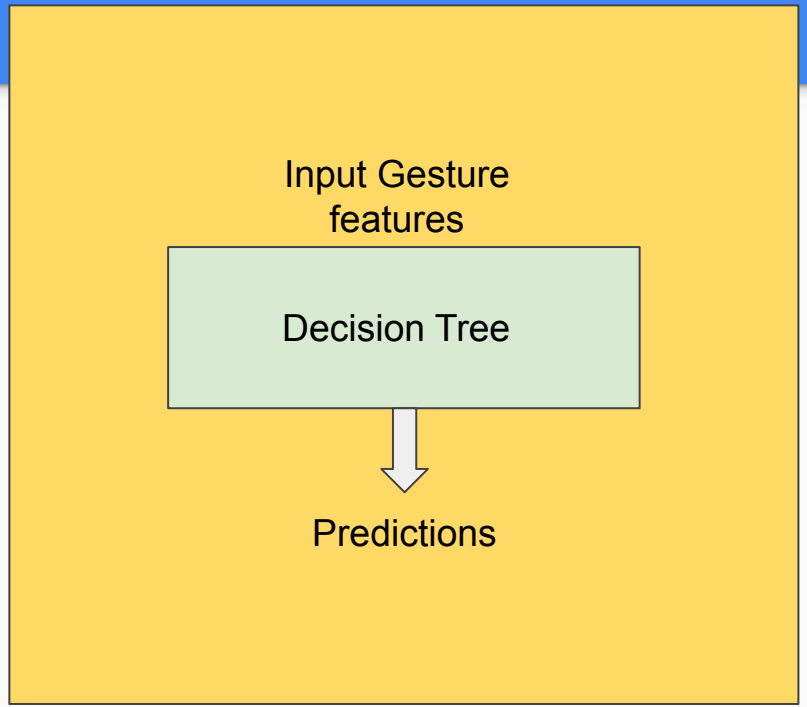
1_x 1_y 1_z 2_x 2_y 2_z . . . 21_x 21_y 21_z flow_mag_1 flow_mag_2 . . . . flow_mag_21 flow_ang_1 flow_ang_2 . . . flow_ang_21

(20x105)

For every frame, **'21 keypoints on hand'** locations is obtained using **mediapipe library** tool. Apart from that, motion information is also obtained by comparing those 21 points with 21 points in the previous frame using **Lucas-Kanade** method. Finally all these information is combined in a large 2D array with **20 rows**(1 for every frame) and **105 columns**(21*3 keypoint locations and 21*2 magnitude and angle for flow)

2. Learning the features

- For learning, **Decision Tree** with *max_depth = 10* is used.
- **Input** to the model is a numpy array having shape **(4*50, 20*105)**.
- Where 4*50 represents **4 gestures** to learn and **50 gesture samples** for every gesture.
- 20x105 is explained in previous slide.
- Using this architecture, we're able to achieve very high test accuracy.
- Thus **this model** is **used to predict the gestures** during the **inference phase**



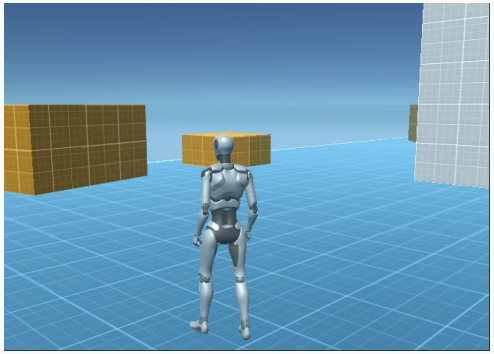
3. Controlling the game

User starts the game, and runs the **inference** script simultaneously

User's hand input is constantly captured and fed to the ML model for prediction of gestures

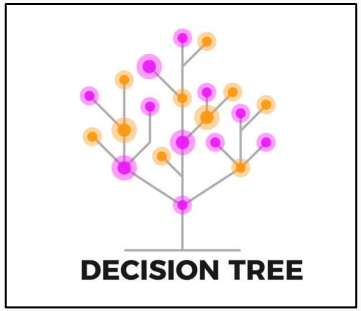
If some valid gesture is predicted, then the corresponding action is performed in the game

For performing the action in game, **pynput** library is used to emulate the keypresses.



Based on predicted gesture, appropriate keypress is emulated using **pynput** python library

Captured gesture information sent to ML model for predicting gestures



Screenshot of one group member with hand features

* Gesture details

4 basic gestures are used namely
forward, backward, left, right

Forward

Backward

Left

Right

All the gestures are made using the **Right hand**.

Deliverables assessment

What we promised

We had promised of achieving a certain level of accuracy in gesture prediction such that controlling the game character is smooth.

What we delivered

- We have achieved that level of accuracy in gesture prediction.
- Game is also playable using the hand gestures.
- We were able to extend the original work in order to include learnings from Computer Vision course. (viz. Motion detection)

Future Work

We can use some advanced machine learning techniques to predict the upcoming movements of the in-game object based on the previous gesture inputs in order to smooth the transition of movements.

Results

Demo Video