Team 10 - Final Project Report

Nishanth Kannan
ee19btech11024@iith.ac.in

Venkata Cheran Deep Pamuru
es19btech11012@iith.ac.in

Mylavarapu Sri Akhil
es19btech11014@iith.ac.in

Pranav Kumar Kota
ee19btech11051@iith.ac.in

Abstract

For this project we have made a painting application using a webcam. We will be using a palm detector model called BlazePalm by MediaPipe and OpenCV library functions for creating an interactive interface and grabbing image input from WebCam.

1. Introduction

The boundary between the virtual and the real world is slowly fading. We are in a time when the metaverse dream is being actively pursued. However, it is still in its infancy. With this project, we hope to demonstrate how we can interact with this invisible world.
In addtion, the ability to detect hand gestures in particular will be very useful in applications like translating sign language, gesture controls, AR/VR development, etc,. The main motivation for doing this project is for convenience and versatility. On a bigger scale, this application can be used for teaching and drawing, considering that these fields are becoming more and more digitized.

2. Problem Statement

To develop a painting interface by processing the video stream input from webcams.

3. Literature Review

MediaPipe is a cross-platform framework for building multimodal applied machine learning pipelines.

The Hand Tracking Solution we are interested in utilizes an ML Pipeline of two models:
1)A palm detector that operates on a bounding box reported by using a single shot multibox detector (BlazePalm).
2)A hand landmark model that takes the input image of a hand and returns a 21 point landmark graph of the hand.
The first model of palm detection outputs a cropped image of the hand which helps our training as the need for data augmentation (rotations, translation, scaling, etc) is drastically reduced and the resources can be allocated to identifying the landmarks (in the hand) and increase the training accuracy.
This is implemented using Blazepalm, a single shot detector model for palm detection. Palm detection is preferred over hand detection, as estimating a rigid body like a palm is much easier than the detection of fingers. On that we use an encoder-decoder feature extractor and minimize the focal loss during training.

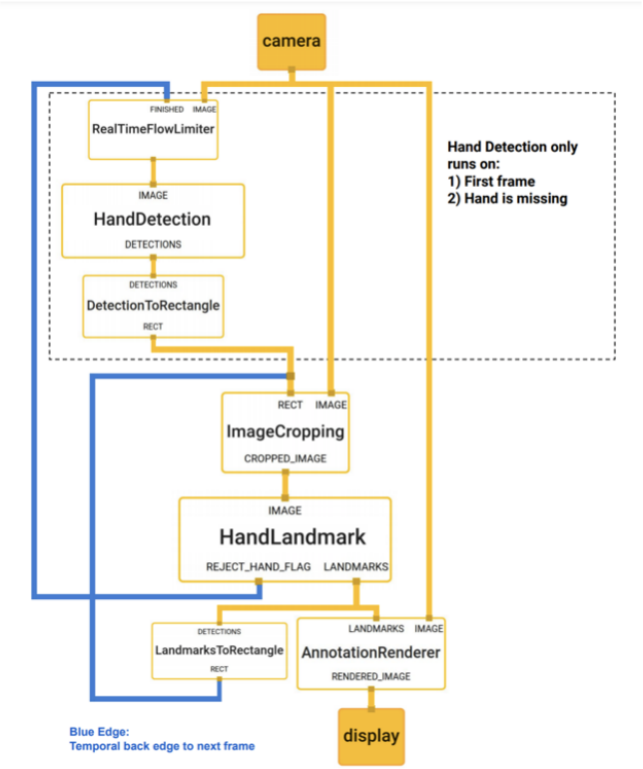

Figure 1. Pipeline

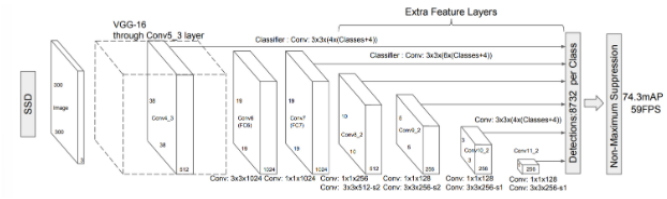

Figure 2. CNN

For the SSD Framework to work, it needs an input image and a ground truth box, and it returns the predicted bounding box. The ML Model used in SSD is a Convolutional Neural Network, with extra feature layers for accurate predictions.

Whilst there exist other models for object detection, SSD is preferred as it works on lower hardware, faster compile time and yields better accuracy.

The second model for the Hand Landmark Model should return 2 landmark points on the picture of the hand with the x, y, relative depth as coordinates, a flag indicating the presence of a hand, and binary classification for left or right handedness.

This is the framework we have used in the project for hand tracking.

4. Results

- Github link
- **Code Explanation**
  - The modules used:
    - find_hands
    - find_landmarks
    - fingers_up
  - Set resolution of the camera input
  - Define the drawing canvas
  - Read input video stream from webcam
  - perform hand tracking using the find_hands module
  - find the landmarks of tracked hand using find_landmarks
  - use fingers_up to determine the position of hand
  - according to our requirements,we have:
    - gesture1(selection mode)
    - gesture2(drawing mode)
    - gesture3(clear all mode)
  - depending on the hand gesture, track coordinates of fingers and apply the required effect onto the canvas

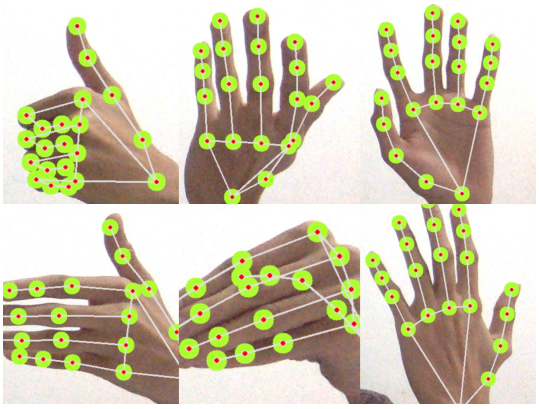

Figure 3. The 21 point hand graph results of the Model

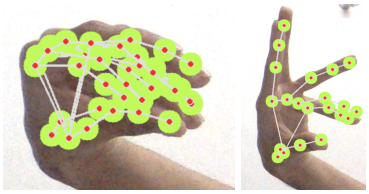

Figure 4. We observe that when clustering of fingers increases, model accuracy falls and it leads to false predictions.

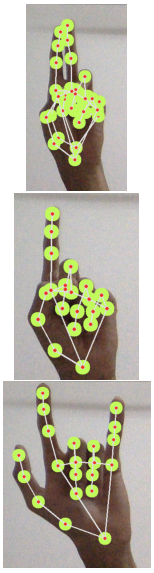

Figure 5. Select,Draw,Clear All

  - calculate fps using the time module and print it for display
  - show the canvas and hand-tracked image frame video stream

Video demonstration

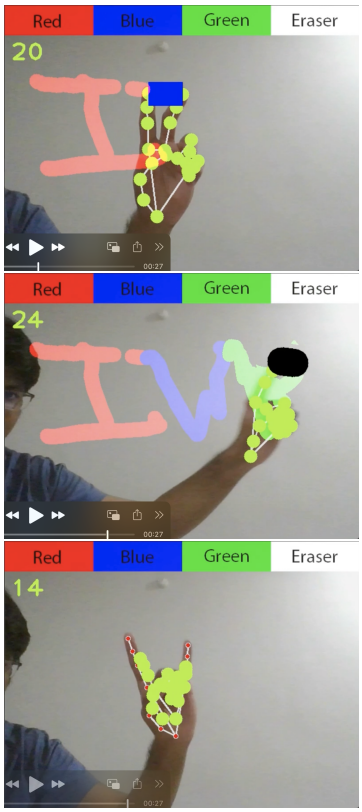

Figure 6. Demonstrating painting using different features

5. Discussion

The handtracking module enables us to be able to process different features required for gesture control. However, as we have seen above, the accuracy falls with increase in skewed clustering of fingers.

For the application, we have observed that the efficiency greatly varies with device specifications.The default frames per second(fps) without tracking was about 30 fps on average, while with tracking, for a laptop enabled with GPU, we were able to get 15 fps to 30 fps on average, whereas for others, it varied from 5 fps to 15 fps on average and a noticeable lag of at least 2 seconds was observed.

6. References

- (PDF) Real time finger tracking and contour detection for gesture recognition using OpenCV Tutorial: Webcam Paint Application Using OpenCV — by Akshay L Chandra
- (PDF) SSD: Single Shot MultiBox Detector
- On Device, Real-Time Hand Tracking using MediaPipe
- MediaPipe Hands: Real-Time Hand Tracking