Starting Soon...

Google Developer Student Clubs

OpenCV & Mediapipe

Computer Vision for Hackathons

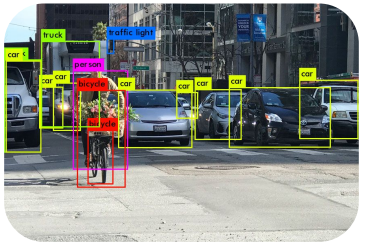
Benjamin Tsang

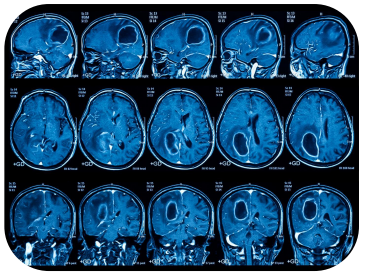
What is computer vision?

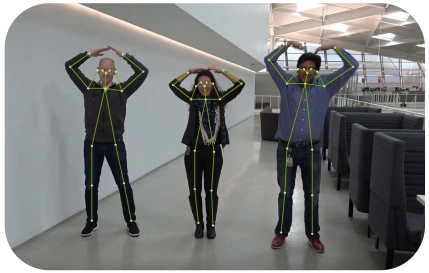
Definition: Allowing computers to interpret and understand images and video

Use cases: Self driving cars, Medical Imaging, AR (Augmented Reality)

Solutions: Object detection, face recognition, Gesture tracking, pose estimation



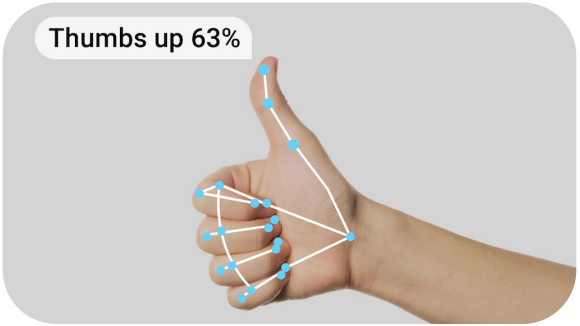




Gesture recognition & face detection

Gesture recognition: Tracking hand movements to recognise gesture and symbols.

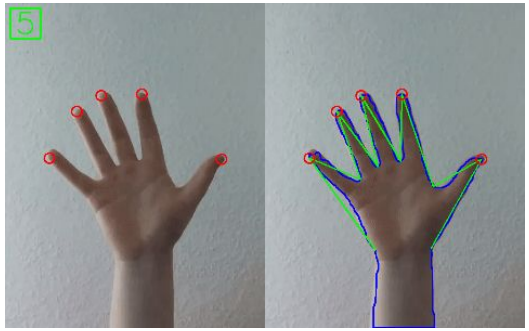
Touchless interfaces, using hand signals to interact with applications.

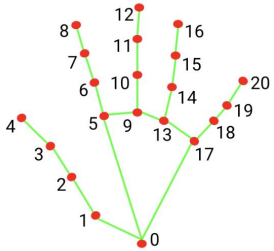


Gesture recognition & face detection

How does gesture recognition work?

1. Hand tracking - locate hands in a video
2. Landmark detection (fingertips, palms, joints)
3. Gesture classification (classify into thumbs up, 1, 2, 3, based on landmarks)



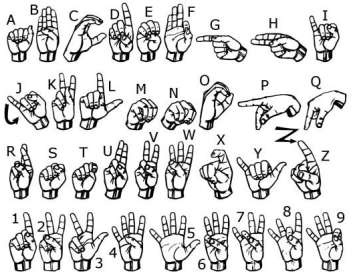


0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP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

Gesture recognition & face detection

Hackathon projects using gesture recognition?

1. Sign language translator app
2. Smart device gesture controller
3. Virtual avatars
4. Metaverse meeting room
5. AR experiment tools



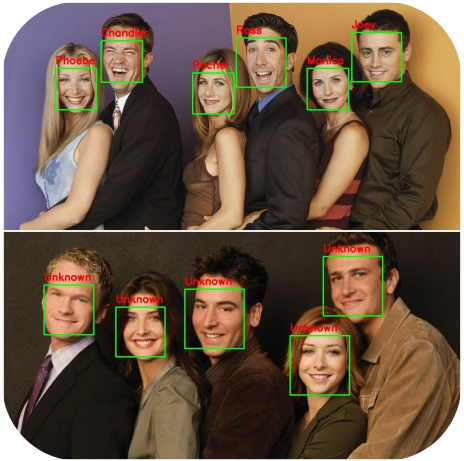
Gesture recognition & face detection

Face detection: Identifying, locating and analysing faces in image and video input

Preprocessing: convert frames to black and white for quicker computation

Feature Extraction: detect eyes, nose, mouth, etc

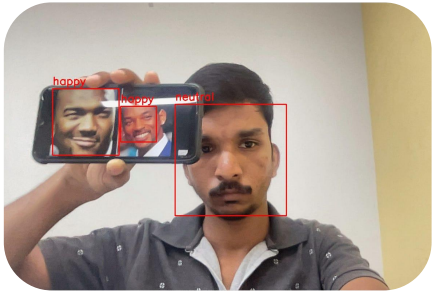
Create bounding boxes: draw a rectangle around detected faces.

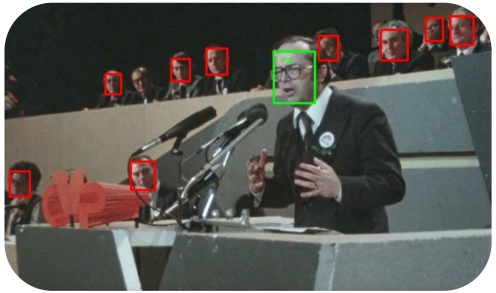


Gesture recognition & face detection

Hackathon projects using face recognition?

1. Automated attendance tracking
2. Face recognition password manager
3. Emotion detection for therapy and mental health
4. Public speaking trainer app





How would we do it without Mediapipe?

Collect Images for Training

Small scale (basic face/hand detection) ~ 5,000 to 10,000 images

Medium scale (face verification/gesture classification) ~ 50,000 to 100,000 images

Large scale (face recognition/complex gestures) ~ 500,000+ images

Preprocess and label the images

Convert images to grayscale to reduce noise

Resize images (ensure fixed size input for ML models)

Normalize pixel values to improve generalization in the ML models

Label the images

Train and Deploy the model

Split the dataset into training, validation and testing

Train the model - mostly convolutional neural nets (CNN)

Evaluate and fine tune hyperparameters

What is Mediapipe and why is it useful?

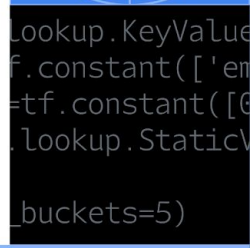
Open Source ML framework by Google for computer vision applications

We will use its pretrained models, which can run directly through a web browser/app. This allows us to skip the complex steps of training and deploying manually.

Solution	Android	Web	Python	iOS	Customize model
LLM Inference API	●	●		●	●
Object detection	●	●	●	●	●
Image classification	●	●	●	●	●
Image segmentation	●	●	●		
Interactive segmentation	●	●	●		
Hand landmark detection	●	●	●	●	
Gesture recognition	●	●	●	●	●
Image embedding	●	●	●		
Face detection	●	●	●	●	
Face landmark detection	●	●	●		
Face stylization	●	●	●		●
Pose landmark detection	●	●	●		
Image generation	●				●
Text classification	●	●	●	●	●
Text embedding	●	●	●		
Language detector	●	●	●		
Audio classification	●	●	●		

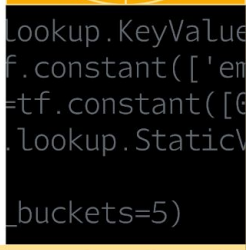
Google Developer Student Clubs

OpenCV + Mediapipe Demo



Google Developer Student Clubs

Integration into a web application



Tech Stack

- React + Node.js
- MediaPipe
- You choice of backend

Setup and installation

```
Install node.js @ https://nodejs.org/en/download

npx create-react-app myapp

npm install react-webcam @mediapipe/face_detection @mediapipe/camera_utils @mediapipe/drawing_utils
```

Importing MediaPipe Modules (face)

Index.html

```
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/camera_utils/camera_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/control_utils/control_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/drawing_utils/drawing_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/face_detection/face_detection.js" crossorigin="anonymous"></script>
```

App.js

```
import {FaceDetection} from '@mediapipe/face_detection';
import * as Facedetection from '@mediapipe/face_detection';
import * as cam from '@mediapipe/camera_utils';
import Webcam from 'react-webcam';
import {useRef, useEffect, use} from 'react';
import { drawRectangle, drawLandmarks } from "@mediapipe/drawing_utils";
```

Add in react-webcam and canvas

```
<Webcam style={{ position: "absolute", marginRight: "auto", marginLeft: "auto",
          left: 0,
          right: 0,
          textAlign: "center",
          zIndex: 9,
          width: 640,
          height: 480,
        }}></Webcam>
<canvas ref={canvasRef}style={{ position: "absolute", marginRight: "auto", marginLeft: "auto", left: 0,
          right: 0,
          textAlign: "center",
          zIndex: 9,
          width: 640,
          height: 480,
        }}></canvas>
```

Setup Face Detection Model from CDN

```
const webcamRef = useRef(null);
const canvasRef = useRef(null);
var camera = null;
ref={webcamRef}
ref={canvasRef}
 useEffect(() => {
   const faceDetection = new FaceDetection({locateFile: (file) => {
     return `https://cdn.jsdelivr.net/npm/@mediapipe/face_detection/${file}`;
   }});
 }, []);
faceDetection.setOptions({
    model:  'short',
    minDetectionConfidence: 0.5,
    minTrackingConfidence: 0.5
  });
```

Send livestream data to face_detection module

```
if (webcamRef.current) {
    camera = new cam.Camera(webcamRef.current.video, {
      onFrame: async () => {
        await faceDetection.send({image: webcamRef.current.video});
      },
      width: 640,
      height: 480
    });
    camera.start();
  }

faceDeteciton.onResults(onResults);
```

Draw bounding boxes on canvas

```
const onResults = (results) => {
  const canvas = canvasRef.current;
  canvas.width = 640;
  canvas.height = 480;
  const context = canvas.getContext("2d");
  context.save();
  context.clearRect(0, 0, canvas.width, canvas.height);
  context.drawImage(
    results.image, 0, 0, canvas.width, canvas.height);
  if (results.detections.length > 0) {
    drawRectangle(
        context, results.detections[0].boundingBox,
        {color: 'blue', lineWidth: 4, fillColor: '#00000000'});
    drawLandmarks(context, results.detections[0].landmarks, {
      color: 'red',
      radius: 5,
    });
  }
  context.restore();
}
```

Setup and installation (hand tracking)

```
Install node.js @ https://nodejs.org/en/download

npx create-react-app myapp

npm install react-webcam @mediapipe/hands @mediapipe/camera_utils @mediapipe/drawing_utils
```

Importing MediaPipe Modules (hands)

Index.html

```
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/camera_utils/camera_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/control_utils/control_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/drawing_utils/drawing_utils.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@mediapipe/hands/hands.js" crossorigin="anonymous"></script>
```

App.js

```
import {Hands} from '@mediapipe/hands';
import * as hands from '@mediapipe/hands';
import * as cam from '@mediapipe/camera_utils';
import Webcam from 'react-webcam';
import {useRef, useEffect, use} from 'react';
import { drawConnectors, drawLandmarks } from "@mediapipe/drawing_utils";
```

Add in react-webcam and canvas

```
<Webcam style={{ position: "absolute", marginRight: "auto", marginLeft: "auto",
          left: 0,
          right: 0,
          textAlign: "center",
          zIndex: 9,
          width: 640,
          height: 480,
        }}></Webcam>
<canvas ref={canvasRef}style={{ position: "absolute", marginRight: "auto", marginLeft: "auto", left: 0,
          right: 0,
          textAlign: "center",
          zIndex: 9,
          width: 640,
          height: 480,
        }}></canvas>
```

Setup Face Detection Model from CDN

```
const webcamRef = useRef(null);
const canvasRef = useRef(null);
var camera = null;
ref={webcamRef}
ref={canvasRef}
 useEffect(() => {
   const handTracker = new Hands({locateFile: (file) => {
     return `https://cdn.jsdelivr.net/npm/@mediapipe/hands/${file}`;
   }});
 }, []);
handTracker.setOptions({
    maxNumHands: 2,
    modelComplexity: 1,
    minDetectionConfidence: 0.5,
    minTrackingConfidence: 0.5
});
```

Send livestream data to face_detection module

```
if (webcamRef.current) {
    camera = new cam.Camera(webcamRef.current.video, {
      onFrame: async () => {
        await handTracker.send({image: webcamRef.current.video});
      },
      width: 640,
      height: 480
    });
    camera.start();
  }

handTracker.onResults(onResults);
```

Draw bounding boxes on canvas

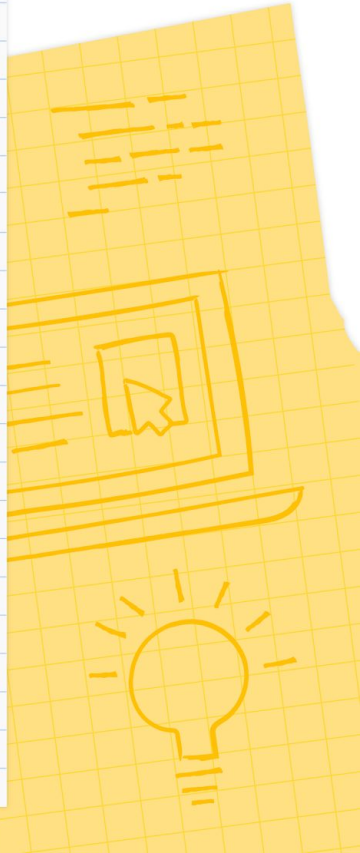
```
const onResults = (results)  => {
   const canvas = canvasRef.current;
   canvas.width =  640;
   canvas.height =  480;
   const context = canvas.getContext( "2d");
   context.save();
   context.clearRect( 0,  0, canvas.width, canvas.height);
   context.drawImage(
       results.image,  0,  0, canvas.width, canvas.height);
   if (results.multiHandLandmarks) {
     for (const landmarks  of results.multiHandLandmarks) {
       drawConnectors(context, landmarks, hands.HAND_CONNECTIONS,
                      {color:  '#00FF00', lineWidth:  5});
       drawLandmarks(context, landmarks, {color:   '#FF0000', lineWidth:  2});
     }
   }
   context.restore();
 }
```

Sending data to the backend

Add to onResult:

```
try {
    await fetch("http://127.0.0.1:5000/hand_landmarks", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ landmarks: results.multiHandLandmarks })
    });
  } catch (error) {
    console.error("Error sending data:", error);
  }
}
```

- Completed code repo https://github.com/tsangh5/deerhacks-workshop/tree/main
- Mediapipe documentation https://chuoling.github.io/mediapipe/, https://ai.google.dev/edge/mediapipe/solutions/guide
- OpenCV documentation https://docs.opencv.org/4.x/index.html



Google Developer Student Clubs

Thank you for attending!

Happy Hacking :)