**GESTURE BASED YOGA DISPLAY**

**CODE TO IDENTIFY POSE AND INDIVIDUAL MARKERS AND THEIR VISIBILITY**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Gesture Based Yoga Display</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background: #f0f0f0;

}

#canvas {

border: 1px solid black;

}

#poseInfo {

position: absolute;

top: 10px;

left: 10px;

background: rgba(255, 255, 255, 0.8);

padding: 10px;

border-radius: 5px;

}

</style>

</head>

<body>

<div id="poseInfo"></div>

<video id="video" width="640" height="480" autoplay></video>

<canvas id="canvas" width="640" height="480"></canvas>

<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>

<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/posenet"></script>

<script>

const video = document.getElementById('video');

const canvas = document.getElementById('canvas');

const ctx = canvas.getContext('2d');

const poseInfo = document.getElementById('poseInfo');

async function setupCamera() {

video.width = 640;

video.height = 480;

const stream = await navigator.mediaDevices.getUserMedia({

video: true

});

video.srcObject = stream;

return new Promise(resolve => {

video.onloadedmetadata = () => {

resolve(video);

};

});

}

async function detectPose() {

const net = await posenet.load();

while (true) {

const pose = await net.estimateSinglePose(video, {

flipHorizontal: false

});

drawPose(pose);

identifyPose(pose);

await new Promise(requestAnimationFrame);

}

}

function drawPose(pose) {

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.drawImage(video, 0, 0, canvas.width, canvas.height);

pose.keypoints.forEach(keypoint => {

if (keypoint.score > 0.5) {

ctx.beginPath();

ctx.arc(keypoint.position.x, keypoint.position.y, 5, 0, 2 \* Math.PI);

ctx.fillStyle = 'aqua';

ctx.fill();

}

});

}

function calculateAngle(a, b, c) {

const ab = Math.sqrt(Math.pow(b.x - a.x, 2) + Math.pow(b.y - a.y, 2));

const bc = Math.sqrt(Math.pow(b.x - c.x, 2) + Math.pow(b.y - c.y, 2));

const ac = Math.sqrt(Math.pow(c.x - a.x, 2) + Math.pow(c.y - a.y, 2));

return Math.acos((ab \* ab + bc \* bc - ac \* ac) / (2 \* ab \* bc)) \* (180 / Math.PI);

}

function identifyPose(pose) {

const keypoints = pose.keypoints;

const leftShoulder = keypoints.find(k => k.part === 'leftShoulder').position;

const rightShoulder = keypoints.find(k => k.part === 'rightShoulder').position;

const leftElbow = keypoints.find(k => k.part === 'leftElbow').position;

const rightElbow = keypoints.find(k => k.part === 'rightElbow').position;

const leftWrist = keypoints.find(k => k.part === 'leftWrist').position;

const rightWrist = keypoints.find(k => k.part === 'rightWrist').position;

const leftHip = keypoints.find(k => k.part === 'leftHip').position;

const rightHip = keypoints.find(k => k.part === 'rightHip').position;

const leftKnee = keypoints.find(k => k.part === 'leftKnee').position;

const rightKnee = keypoints.find(k => k.part === 'rightKnee').position;

const leftAnkle = keypoints.find(k => k.part === 'leftAnkle').position;

const rightAnkle = keypoints.find(k => k.part === 'rightAnkle').position;

const leftElbowAngle = calculateAngle(leftShoulder, leftElbow, leftWrist);

const rightElbowAngle = calculateAngle(rightShoulder, rightElbow, rightWrist);

const leftKneeAngle = calculateAngle(leftHip, leftKnee, leftAnkle);

const rightKneeAngle = calculateAngle(rightHip, rightKnee, rightAnkle);

poseInfo.innerHTML = `

<p>Left Elbow Angle: ${leftElbowAngle.toFixed(2)}</p>

<p>Right Elbow Angle: ${rightElbowAngle.toFixed(2)}</p>

<p>Left Knee Angle: ${leftKneeAngle.toFixed(2)}</p>

<p>Right Knee Angle: ${rightKneeAngle.toFixed(2)}</p>

`;

// Here you can add logic to identify specific yoga poses based on the angles.

}

async function main() {

await setupCamera();

video.play();

detectPose();

}

main();

</script>

</body>

</html>