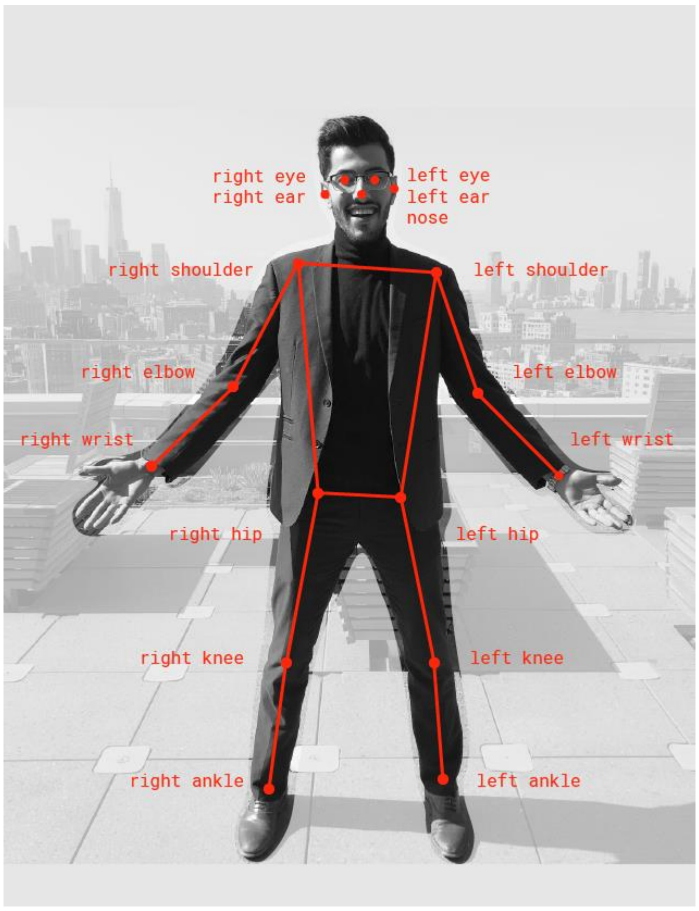
DETECTION & SCORING OF MULTIPLE FACES' ORIENTATION USING POSENET

TEAM (4-b)

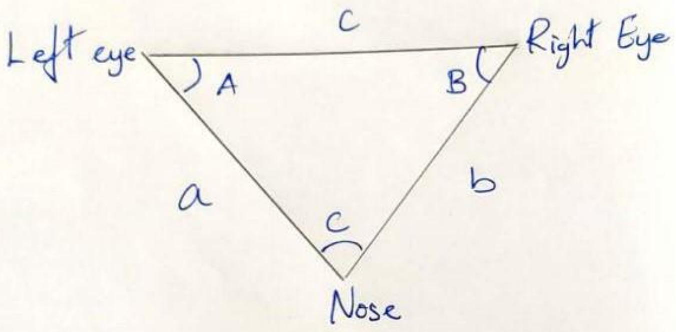
RAMPRASATH A, 17BCE1098
SHYAM SURESH, 17BCE1303

Abstract:

Given an image consisting of multiple faces, the model should detect the orientation of the faces and calculate a score based on the number of faces that are facing straight.



POSENET returns 17 key-points for all persons who are present in the given image, out of which we are going to work with the three key-points present in the face namely: left eye, right eye and the nose. The algorithm below is implemented w.r.t the pret-trained POSENET model to realize our objective.

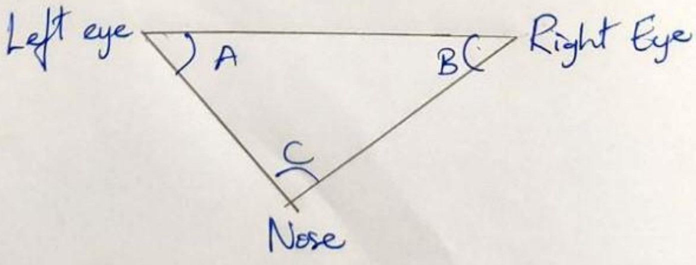


To find:

$$\angle A = \cos^{-1}\left(b^2 + c^2 - a^2 / 2bc\right)$$

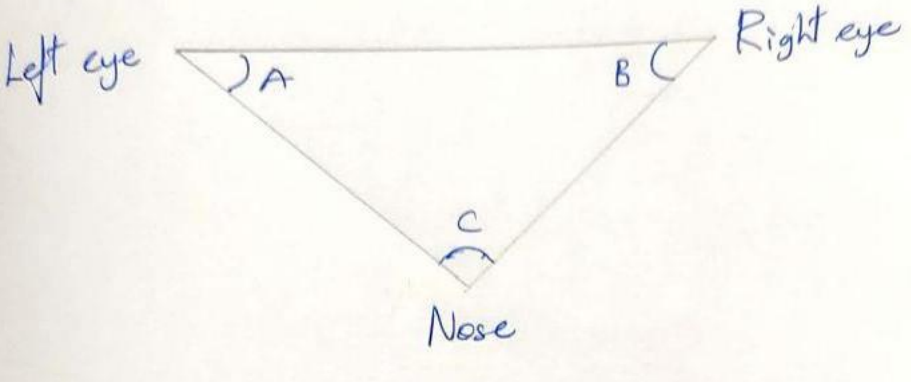
$$\angle B = \cos^{-1}\left(c^2 + a^2 - b^2 / 2ca\right)$$

Case 1: If a person turns his face left w.r.t. camera



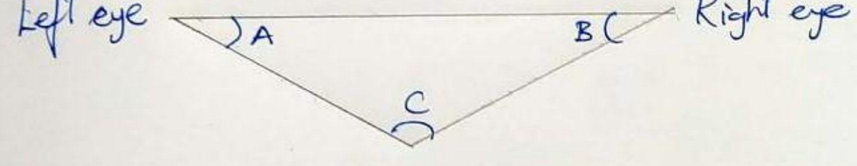
Here: $\angle A > \angle B$

Case 2 : If a person is facing right w.r.t. camera.



Here, $\angle A < \angle B$

Case 3: If a person is looking upwards w.r.t. camera



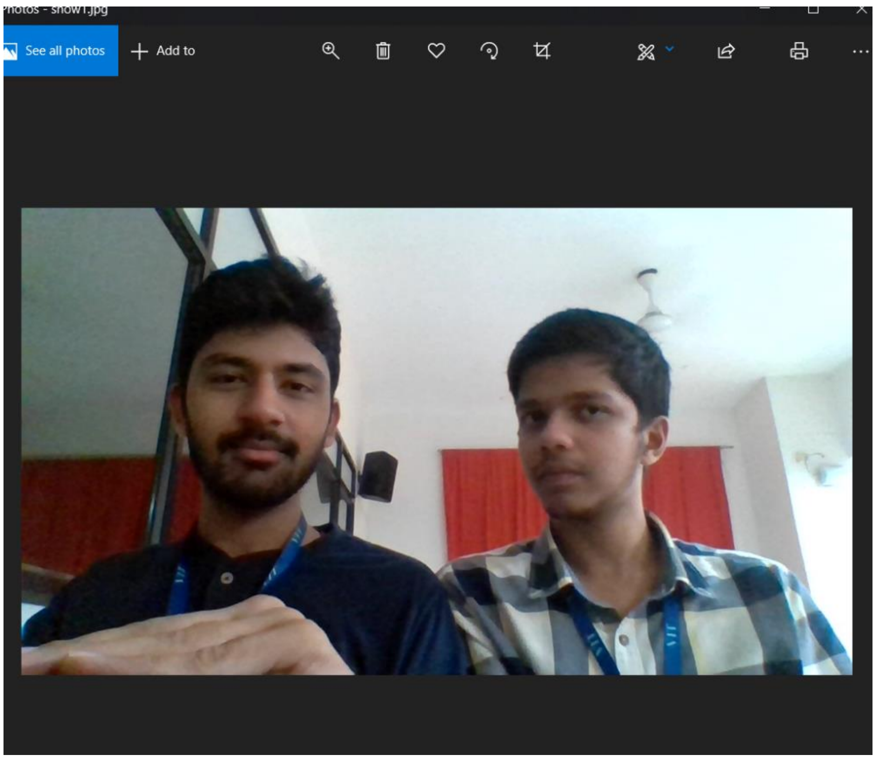
Here, $\angle A$ and $\angle B$ is very smaller than the normal threshold.

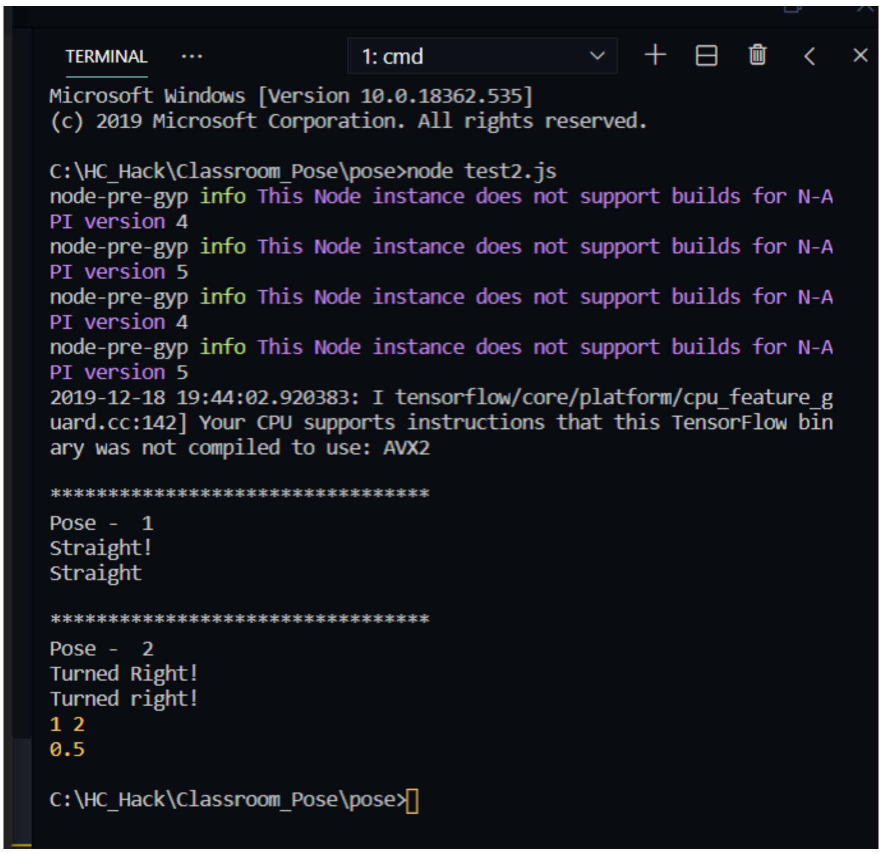
Code:

```
const tf = require('@tensorflow/tfjs-node');

const posenet = require('@tensorflow-models/posenet');

const {

    createCanvas, Image

} = require('canvas')

const imageScaleFactor = 0.5;

const outputStride = 16;

const flipHorizontal = false;


//DETECTS THE FACE ORIENTATION

const detect = (pose) => {

    var dc = []

    var points = []

    var parts = []

    for(var i = 0; i < 5; i++) {

        var data = null

        const pk = pose.keypoints[i]

        if(pk.score > 0.50){

            data = [pk.part, pk.score, pk.position.x, pk.position.y]

            parts.push(pk.part)

        }

        points.push(data)

    }


    if(!parts.includes("leftEar") && !parts.includes("rightEar")){

        console.log("Straight")

        dc.push("s")

    }

    else if(!parts.includes("leftEar")){
        console.log("Turned Left!")

        dc.push("l")

    }

    else if(!parts.includes("rightEar")){

        console.log("Turned Right!")

        dc.push("r")

    }

    else{

        console.log("Straight!")

        dc.push("s")

    }


    try{

            var a = Math.sqrt(Math.pow((points[0][2] - points[1][2]), 2) +
Math.pow((points[0][3] - points[1][3]), 2))

            var b = Math.sqrt(Math.pow((points[0][2] - points[2][2]), 2) +
Math.pow((points[0][3] - points[2][3]), 2))

            var c = Math.sqrt(Math.pow((points[1][2] - points[2][2]), 2) +
Math.pow((points[1][3] - points[2][3]), 2))


            var B = Math.acos((Math.pow(b,2) + Math.pow(c,2) - Math.pow(a,2))
/ (2*b*c)) * 180/Math.PI



            var A = Math.acos((Math.pow(c,2) + Math.pow(a,2) - Math.pow(b,2))
/ (2*c*a)) * 180/Math.PI



            if(A < 30 && B < 30){

                console.log("Looking up")

            }

            else if(A > B){

                if((B + 7.5) < A){

                    console.log("Turned left")
```

```javascript
                    dc.push("l")
                }
                else{
                    console.log("Straight")
                    dc.push("s")
                }
            }else if(B > A){
                if((A + 7.5) < B){
                    console.log("Turned right!")
                    dc.push("r")
                }
                else{
                    console.log("Straight")
                    dc.push("s")
                }
            }
            score = 0
            if(dc[0] == "s") score += 0.5
            if(dc[1] == "s") score += 0.5

    }catch(err){
        if(dc[0] == "s") score = 0.5
        else score = 0
    }
    return score
}


const tryModel = async() => {
    const img = new Image();
    imgName = '../data/show1.jpg'
    img.src = imgName;
    const canvas = createCanvas(img.width, img.height);

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

    ctx.drawImage(img, 0, 0);

    const input = tf.browser.fromPixels(canvas);


    //const pose = await net.estimateSinglePose(input, imageScaleFactor,
flipHorizontal, outputStride);

    //load model

    const net = await posenet.load()



    //get poses of each face

    const poses = await net.estimateMultiplePoses(input, {

        flipHorizontal: false,

        maxDetections: 100,

        scoreThreshold: 0.5,

        nmsRadius: 20

    })



    var points = []

    var parts = []



    var i = 1

    sc = 0



    //parse through the list of poses

    for(const pose of poses){

        if(pose.score > 0.3){
            console.log("\n**********************************\nPose - ", i)
            i++
            sc += detect(pose)
        }
    }

    console.log(sc, i-1)
    sc = sc / (i-1)
    console.log(sc)

    //open image
    var exec = require('child_process').exec
    child = await exec('start ' + imgName,
      function (error, stdout, stderr) {
        if (error !== null) {
          console.log('exec error: ' + error);
        }
    });
}

tryModel();
```

Output Screenshots:





```
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\HC_Hack\Classroom_Pose\pose>node test2.js
node-pre-gyp info This Node instance does not support builds for N-A
PI version 4
node-pre-gyp info This Node instance does not support builds for N-A
PI version 5
node-pre-gyp info This Node instance does not support builds for N-A
PI version 4
node-pre-gyp info This Node instance does not support builds for N-A
PI version 5
2019-12-18 19:44:02.920383: I tensorflow/core/platform/cpu_feature_g
uard.cc:142] Your CPU supports instructions that this TensorFlow bin
ary was not compiled to use: AVX2

*********************************
Pose -  1
Straight!
Straight

*********************************
Pose -  2
Turned Right!
Turned right!
1 2
0.5

C:\HC_Hack\Classroom_Pose\pose>
```