

SVEUČILIŠTE U SPLITU  
PRIRODOSLOVNO-MATEMATIČKI FAKULTET U SPLITU

**SEMINAR**

# **Detekcija položaja tijela osobe**

*Matea Lukić*

Mentor: *Dr.sc. Saša Mladenović*

Split, svibanj 2021.

# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Seminarski rad</b>	<b>2</b>
2.1. Ključni pojmovi . . . . .	2
2.2. Procjena ljudske poze u stvarnom vremenu u pregledniku s TensorFlow.js . . . . .	2
2.2.1. Početak rada s PoseNetom . . . . .	2
2.2.2. Procjena poze za jednu osobu . . . . .	4
2.2.3. Procjena više osoba . . . . .	5
2.2.4. Tehnički dio - algoritam . . . . .	6
2.3. Korištenje TensorFlow.js PoseNet modela i Pythona. Rješavanje zadataka s podudaranjem poza . . . . .	8
2.3.1. Raščlanjivanje rezultata PoseNeta . . . . .	8
2.3.2. Podudaranje poza . . . . .	10
2.4. Primjena . . . . .	12
2.4.1. Učitavanje PoseNet modela. Crtanje i spajanje ključnih točaka . . . . .	12
2.4.2. Filter za lice . . . . .	14
2.4.3. Igra . . . . .	15
<b>3. Zaključak</b>	<b>26</b>
<b>4. Literatura</b>	<b>27</b>
<b>5. Sažetak</b>	<b>28</b>

# 1. Uvod

Prepoznavanje poze osobe se odnosi na tehnike računalnog vida kojima se može otkriti ljudska figura na slikama ili video zapisima. Pomoću njih možemo utvrditi gdje se, primjerice, na slici nalazi lijevo rame. Ova se tehnika koristi u fitness industriji, interaktivnim instalacijama koje reagiraju na tijelo i sličnim područjima. Ovdje će primjena biti predstavljena na primjeru računalne igru u kojoj igrač kontrolira lopticu položajem tijela u realnom vremenu. Nakon ove motivacije, za početak će biti ukratko objašnjeni ključni pojmovi. Fokus će biti na detekciji položaja tijela pomoću TensorFlow biblioteke i klasifikaciji poza pomoću neuronskih mreža.

## 2. Seminarski rad

### 2.1. Ključni pojmovi

Za početak, upoznajmo se s terminima koji će biti navedeni u seminaru, a krucijalni su za razumijevanje.

- **TensorFlow**

TensorFlow je besplatna simbolička matematička biblioteka softvera otvorenog koda za strojno učenje. Ima raznovrsnu uporabu, ali je poseban fokus na treningu i zaključivanju dubokih neuronskih mreža. TensorFlow ime je dobio po **tenzorima**, koji su nizovi proizvoljne dimenzionalnosti. Korištenjem TensorFlow-a može se manipulirati tenzorima s vrlo velikim brojem dimenzija.

- **PoseNet**

PoseNet, iz biblioteke TensorFlow.js, je verzija modela strojnog učenja koji omogućuje procjenu ljudske poze u pregledniku u stvarnom vremenu.

### 2.2. Procjena ljudske poze u stvarnom vremenu u pregledniku s TensorFlow.js

Procjena poza je tehnika računalnogvida koja prepoznaje osobe na slikama i video zapisima. Algoritam jednostavno procjenjuje gdje su ključni zglobovi tijela. Ova tehnologija ne može identificirati osobu koja se nalazi na slici - dakle, ne povezuje pozu s bilo kakvim osobnim podacima. Detekcija poze ima široku primjenu, od interaktivnih instalacija koje reagiraju na tijelo, do proširene stvarnosti, animacije i još mnogo toga.

#### 2.2.1. Početak rada s PoseNetom

PoseNet model se može koristiti za procjenu jedne ili više poza, što znači da postoji verzija algoritma koja može otkriti samo jednu osobu na slici / video snimci i jedna

verzija koja može otkriti više osoba na slici / video snimci.

Na visokoj razini procjena poze odvija se u dvije faze:

1. Ulazna RGB slika dovodi se kroz konvolucijsku neuronsku mrežu.
2. Algoritam za otrivanje jedne ili više poza koristi za dekodiranje poza, postavljanje rezultata vjerojatnosti za položaje ključnih točaka i rezultata vjerojatnosti podudaranja s rezultatima iz modela.

Objašnjenje spomenutih pojmova:

- **Poza** - PoseNet će vratiti stavljeni objekt koji sadrži popis ključnih točaka i projiciju vjerojatnosti na razini instance za svaku otkrivenu osobu. PoseNet vraća vrijednosti vjerojatnosti za svaku otkrivenu osobu, kao i za svaku otkrivenu ključnu točku poze
- **Ocjena povjerenja u pozu** - ovo određuje cjelokupnu vjerojatnost u procjenu poze. Kreće se između 0,0 i 1,0.
- **Ključna točka** - dio poze osobe koji se procjenjuje, poput nosa, desnog uha, lijevog koljena, desnog stopala itd. Sadrži i položaj i ocjenu vjerojatnosti ključne točke. PoseNet trenutno otkriva 17 ključnih točaka PoseNeta prikazanih na sljedećem dijagramu:



**Slika 2.1:** Ključne točke i poza

- **Procjena vjerojatnosti ključne točke** - ovo određuje pouzdanost da je procijenjeni položaj ključne točke točan. Kreće se između 0,0 i 1,0.
- **Položaj ključne točke** - 2D koordinate x i y na izvornoj ulaznoj slici gdje je otkrivena ključna točka.

### 2.2.2. Procjena poze za jednu osobu

Algoritam za procjenu u jednoj pozici jednostavniji je i brži. Idealan je slučaj kada je samo jedna osoba u fokusu na ulaznoj slici ili video - snimci. Nedostatak je što će se, ako je na slici više osoba, ključne točke obje osobe možebitno procijeniti dijelom iste pozice - što znači, na primjer, da se lijeva ruka osobe 1 i desno koljeno osobe 2 mogu povezati algoritmom kao da pripada istoj pozici. Pregledajmo ulaze za algoritam procjene u jednoj pozici:

- **Ulazni element slike** - HTML element koji sadrži sliku za predviđanje poza, poput videozapisa ili oznake slike.

- **Faktor skale slike** - Broj između 0,2 i 1. Zadane vrijednosti 0,50. Ako se ovaj broj postavi na nižu vrijednost smanji se slika i poveća brzina prilikom prolaska mrežom po cijenu točnosti.
- **Horizontalno okretanje** - Služi za okretanje/zrcaljenje položaja.
- **Izlazni korak** - mora biti 32, 16 ili 8. Zadane postavke su 16. Ovaj parametar utječe na točnost i brzinu procjene pozne. Što je vrijednost izlaznog koraka niža - točnost je veća, ali je brzina manja et vice versa.

Izlazi algoritma za procjenu jedne pozne: **Poza**, koja sadrži i **ocjenu vjerojatnosti pozne** i niz od 17 **ključnih točaka**. Svaka ključna točka sadrži položaj ključne točke i ocjenu pouzdanosti (vjerojatnost) ključne točke.

### 2.2.3. Procjena više osoba



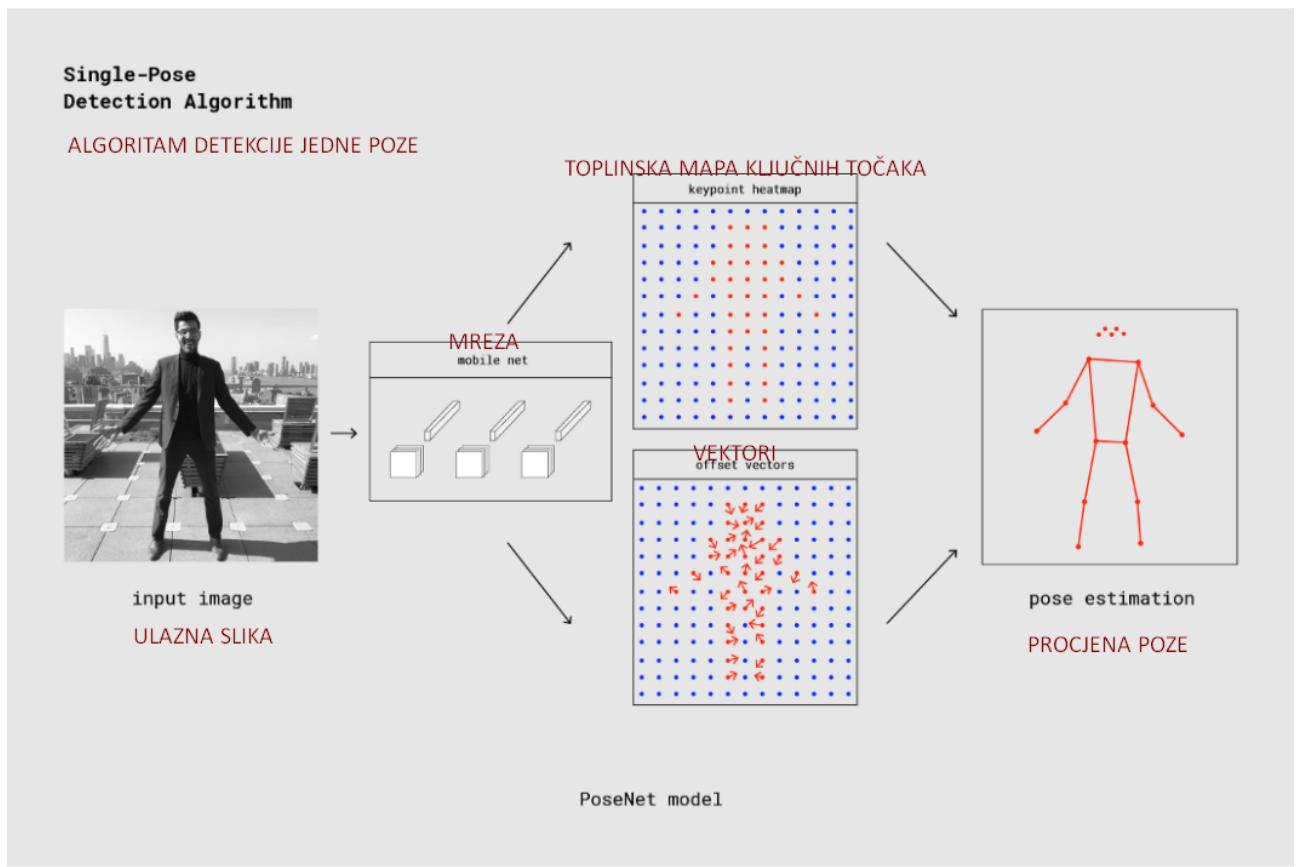
**Slika 2.2:** Detekcija pozne kad je više osoba na slici

Algoritam za procjenu pozne za više osoba može procijeniti mnogo pozne / osoba na slici. Složeniji je i nešto sporiji od algoritma s jednom pozom, ali prednost mu je u

tome što ako se na slici pojavi više ljudi, manja je vjerojatnost da će njihove otkrivene ključne točke biti povezane s pogrešnom pozom. Atraktivno svojstvo ovog algoritma je to što na performanse ne utječe broj osoba na ulaznoj slici. Bez obzira ima li manje ili više osoba na slici, vrijeme računanja bit će isto.

#### 2.2.4. Tehnički dio - algoritam

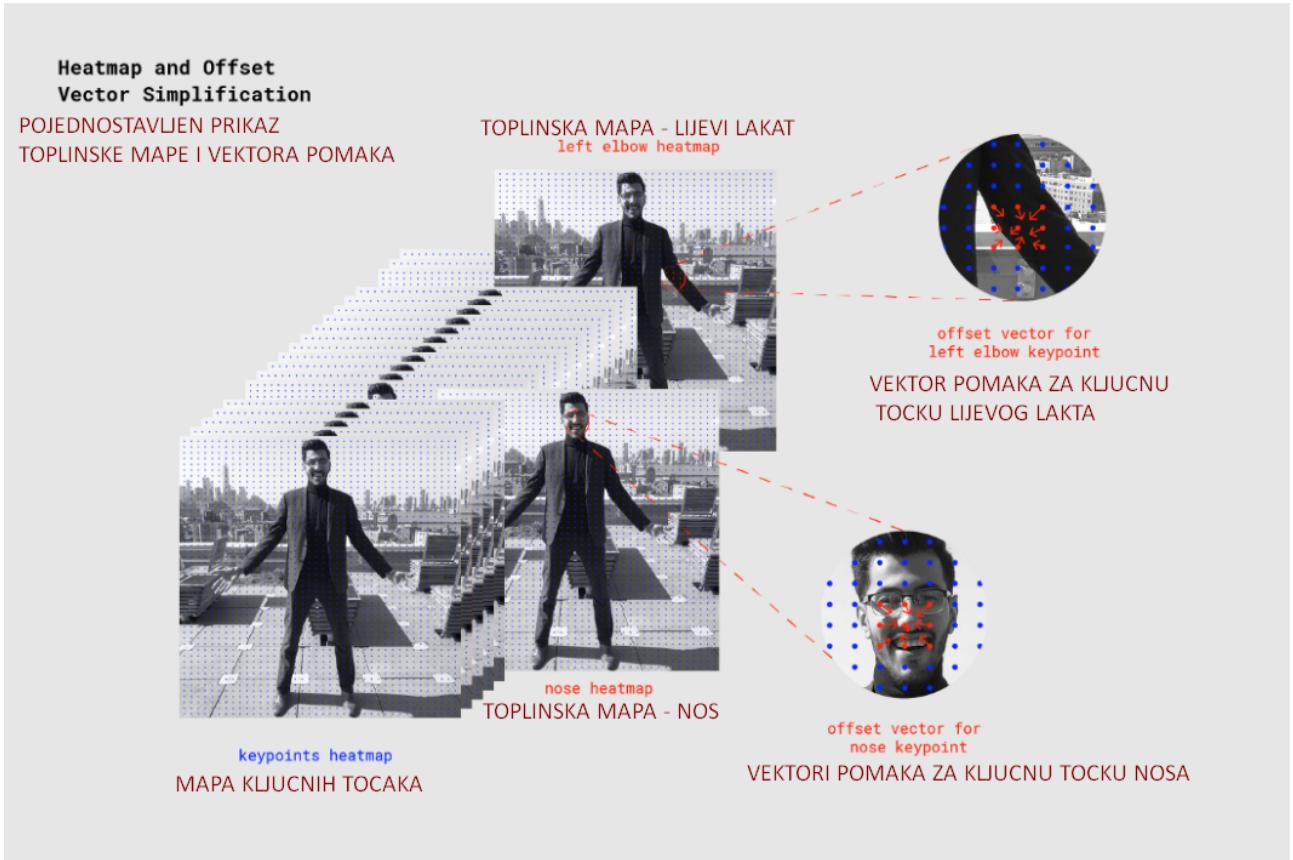
U ovom dijelu će biti pobliže objašnjeni dijelovi algoritma za prepoznavanje jedne poze. Postupak izgleda ovako:



**Slika 2.3:** Cjevovod detektora poza za jednu osobu pomoću PoseNet-a

#### Obrada ulaznih podataka modela i izlazi modela

Kada PoseNet obrađuje sliku, ono što se zapravo vraća je toplinska karta zajedno s vektorima pomaka koji se mogu dekodirati kako bi se na slici pronašla područja s visokom vjerojatnosti koja odgovaraju postavljanju ključnih točaka. Ilustracija u nastavku biloži kako je svaka od ključnih točaka poza povezana s jednim tenzorom toplinske karte i vektorom pomaka.



**Slika 2.4:** Toplinska mapa i vektori pomaka

Svaka od 17 ključnih točaka poze koje vraća PoseNet povezana je s jednim tenzorom toplinske karte i jednim tenzorom pomaknutog vektora koji se koristi za određivanje točnog mjesto ključne točke.

**Toplinske karte** Svaka je toplinska karta 3D tenzor veličine: razlučivost x razlučivost x 17, jer je 17 broj ključnih točaka koje je PoseNet otkrio. Svaka pozicija u toj toplinskoj mapi ima ocjenu pouzdanosti, što je vjerojatnost da dio te ključne točke postoji na tom položaju.

**Offset vektori (vektori pomaka)** Svaki vektor pomaka je 3D tenzor veličine: razlučivost x razlučivost x 34, gdje je 34 broj ključnih točaka \* 2. Budući da su toplinske mape približna vrijednost mesta na kojem se nalaze ključne točke, vektori pomaka odgovaraju položaju na mjestu točaka toplinske karte i koriste se za predviđanje točnog mesta ključnih točaka.

**Procjena položaja iz rezultata modela** Nakon što se slika provede kroz model, izvodimo nekoliko izračuna kako bismo procijenili pozu iz rezultata.

Dobivanje ključnih točaka poza:

1. **Sigmoidna aktivacija** vrši se na toplinskoj karti kako bi se dobili bodovi (ocjena

vjerojatnosti).

2. **argmax2d** se radi na rezultatima vjerojatnosti ključne točke kako bi se dobio x i y indeks u toplinskoj mapi s najvišom ocjenom za svaki dio, što je zapravo mjesto gdje će dio najvjerojatnije postojati.
3. **Vektor pomaka** za svaki dio dobiva se dobivanjem x i y iz pomaka koji odgovaraju indeksu x i y u toplinskoj mapi za taj dio. To daje tenzor veličine 17x2, pri čemu je svaki redak vektor pomaka za odgovarajuću ključnu točku. Na primjer, za dio s indeksom k, kada je položaj toplinske karte y i x, vektor pomaka je:

```
offsetVector = [offsets.get(y, x, k), offsets.get(y, x, 17 + k)]
```

**Slika 2.5:** Vektori pomaka

4. Da bi se dobila, **ključna točka** toplinska karta x i y svakog dijela množi se s izlaznim korakom, a zatim se dodaje odgovarajućem vektoru pomaka.

```
keypointPositions = heatmapPositions * outputStride + offsetVectors
```

**Slika 2.6:** Računanje pozicije ključne točke

5. Konačno, svaka ocjena vjerojatnosti ključne točke je ocjena vjerojatnosti svog položaja toplotne karte. Ocjena povjerenja u pozu srednja je vrijednost bodova ključnih točaka.

## 2.3. Korištenje TensorFlow.js PoseNet modela i Pythona. Rješavanje zadatka s podudaranjem poza

### 2.3.1. Raščlanjivanje rezultata PoseNeta

U nastavku će biti opisana obrada izlaznih podataka modela za vizualiziranje poze. Izlaz se sastoji od 2 dijela: Heatmaps (9,9,17) - odgovaraju vjerojatnosti pojave svake ključne točke u određenom dijelu slike (9,9). Koriste se za lociranje približnog položaja zglobova. Offset vektori (9,9,34) - oni se koriste za točniji izračun položaja ključne točke. Prvih 17 treće dimenzije odgovaraju x koordinatama, a drugih 17 odgovara y koordinatama

Slika ispod ima mrežu vrijednosti  $9 \times 9$  i jedna je od 17 toplotnih karata. Maksimalna vrijednost je u čeliji koja se nalazi na čelu i kosi. Očito tamo nema nosa,

ali pomoću toplinskih karata se može pronaći samo približan položaj ključnih točaka. Nakon pronalaska indeksa za maksimalnu vrijednost, možemo prilagoditi položaje pomaknutim vektorima.



**Slika 2.7:** Vjerojatnosti pojavljivanja nosa u različitim čelijama slike

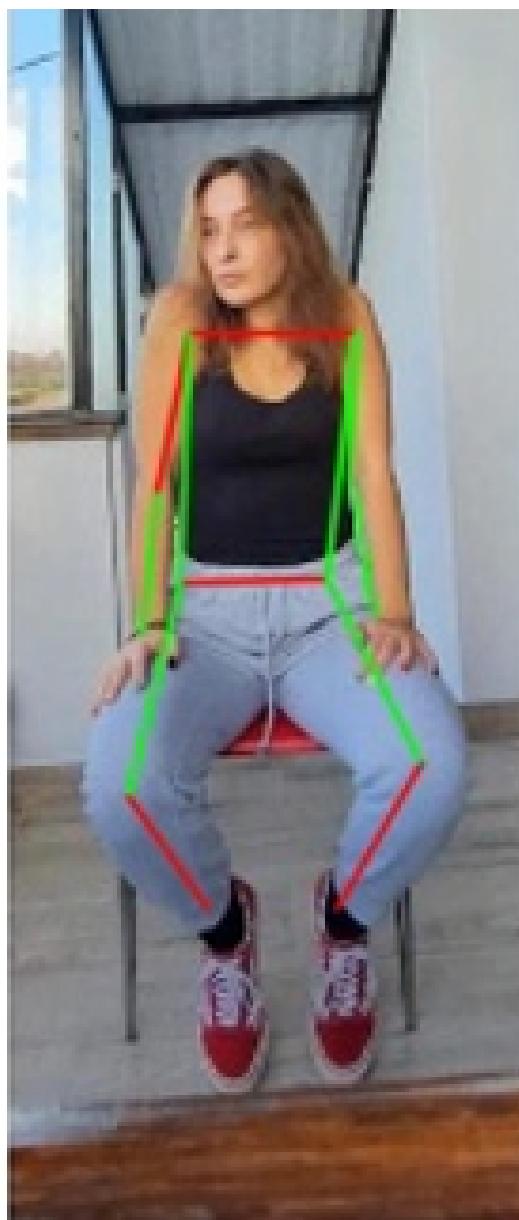
Dodata je zastavica u svaku ključnu točku kako bi se moglo filtrirati one u koje model nije siguran i one za koje predviđa da su izvan slike. Nakon toga se može vizualizirati rezultat crtanjem svih ključnih točka. Evo slika koje sadrže predložak i ciljne poze :



**Slika 2.8:** Previđene koordinate ključnih točaka - žute točke na slici

### 2.3.2. Podudaranje poza

Postoje različiti načini usporedbe položaja dijelova tijela i njihovih međusobnih odnosa na slici. Ideja ovog pristupa je usporediti kut između svakog dijela tijela i x osi za obje slike te izračunati proporcije između dijelova tijela za svaku sliku i usporediti ih. Prvo se računaju kutevi i veličine za svaki dio tijela.



**Slika 2.9:** Zelene linije prikazuju dio poze koji se podudara, a crvene linije dio koji se ne podudara

## 2.4. Primjena

### 2.4.1. Učitavanje PoseNet modela. Crtanje i spajanje ključnih točaka

```
let video;
let poseNet;
let pose;
let skeleton;

function setup() {
  createCanvas(640, 480);
  video= createCapture(VIDEO);
  video.hide();

  poseNet=ml5.poseNet(video, modelLoaded);
  poseNet.on('pose',gotPoses);
}

function gotPoses(poses) {
  console.log(poses);
  if(poses.length>0) {
    pose=poses[0].pose;
    skeleton= poses[0].skeleton;
  }
}

function modelLoaded() {
  console.log('poseNet ready');
}
```

Slika 2.10: Funkcije za dohvaćanje PoseNet modela i "hvatanje" poza u realnom vremenu s kamere

Na slici su prikazane funkcije setup(), gotPoses() i modelLoaded(). Funkcija setup() kreira pozadinu i uključuje video u realnom vremenu s kamere računala. Poziva funkciju poseNet() iz biblioteke ml5 u koju su uključeni argumenti video i modelLoaded(). Nadalje pozivamo metodu gotPoses() kao argument funkcije on(). Ona sprema poze i kostur u varijable pose i skeleton.

```
function draw() {
  background(220);
  translate(video.width,0);
  scale(-1,1);
  image(video,0,0,video.width,video.height);

  if(pose) {

    let eyeR=pose.rightEye;
    let eyeL=pose.leftEye;
    let d=dist(eyeR.x,eyeR.y,eyeL.x,eyeL.y);

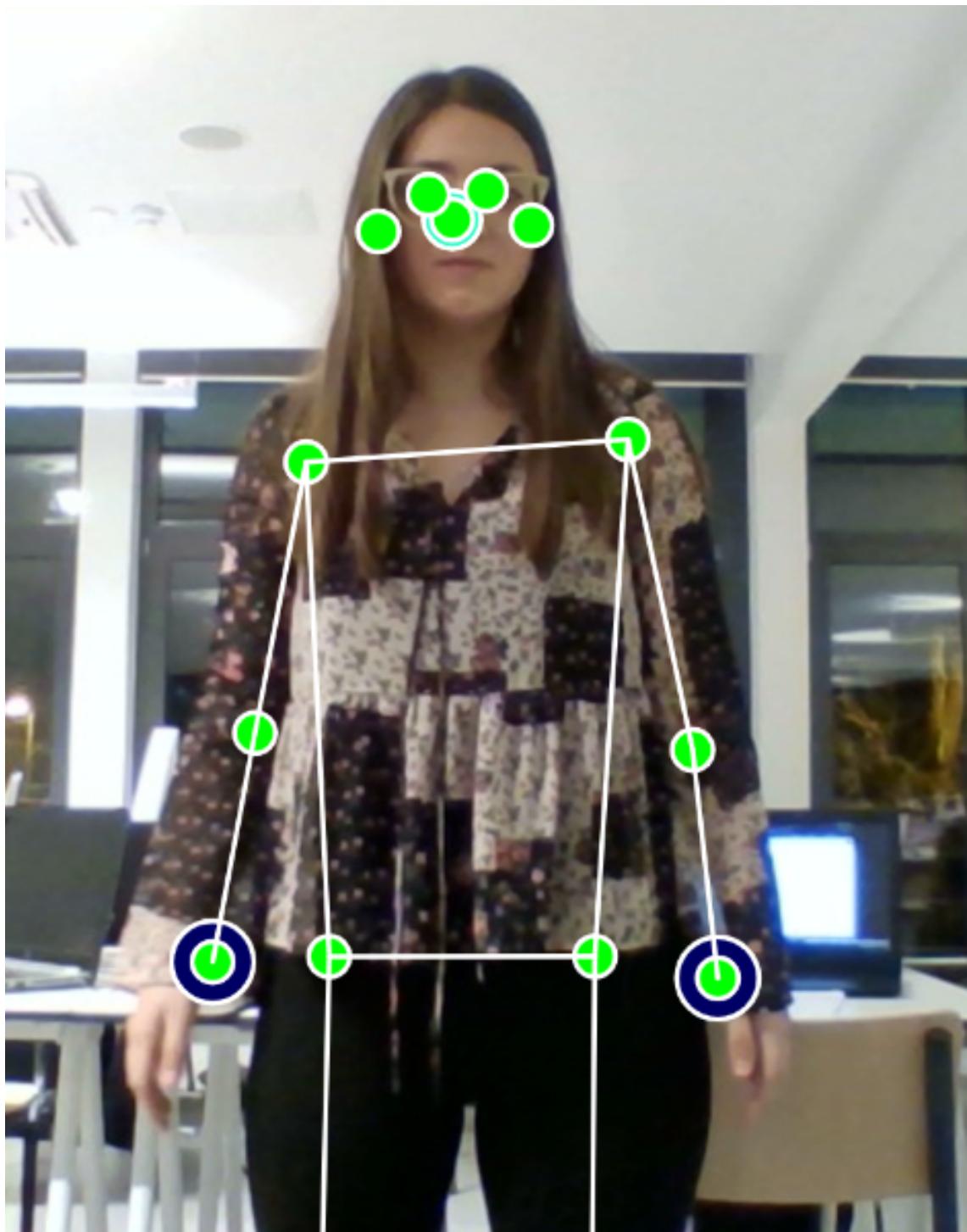
    fill(50,250,200);
    ellipse(pose.nose.x,pose.nose.y,d);
    fill(0,0,100);
    ellipse(pose.rightWrist.x,pose.rightWrist.y,32);
    ellipse(pose.leftWrist.x,pose.leftWrist.y,32);

    for(let i=0;i<pose.keypoints.length;i++) {
      let x=pose.keypoints[i].position.x;
      let y=pose.keypoints[i].position.y;
      fill(0,255,0);
      ellipse(x,y,16,16);
    }

    for(let i=0;i< skeleton.length;i++) {
      let a=skeleton[i][0];
      let b=skeleton[i][1];
      strokeWeight(2);
      stroke(255);
      line(a.position.x,a.position.y,b.position.x,b.position.y)
    }
  }
}
```

**Slika 2.11:** Crtanje krugova na mjestima ključnih točaka i spajanje istih linijama

Funkcija draw() crta pozadinu i video. Kada je u videu detektirana osoba i njezine poze, pomoću funkcije ellipse() crtamo ključne točke. Koordinate elipsi su koordinate dohvaćene iz PoseNet modela. Funkcijom line() crtamo linije među ključnim točkama tako što se, također, uzimaju koordinate PoseNet modela za argumente. Kada se pokrene program, dobiju se točke i linije kao na slici ispod :

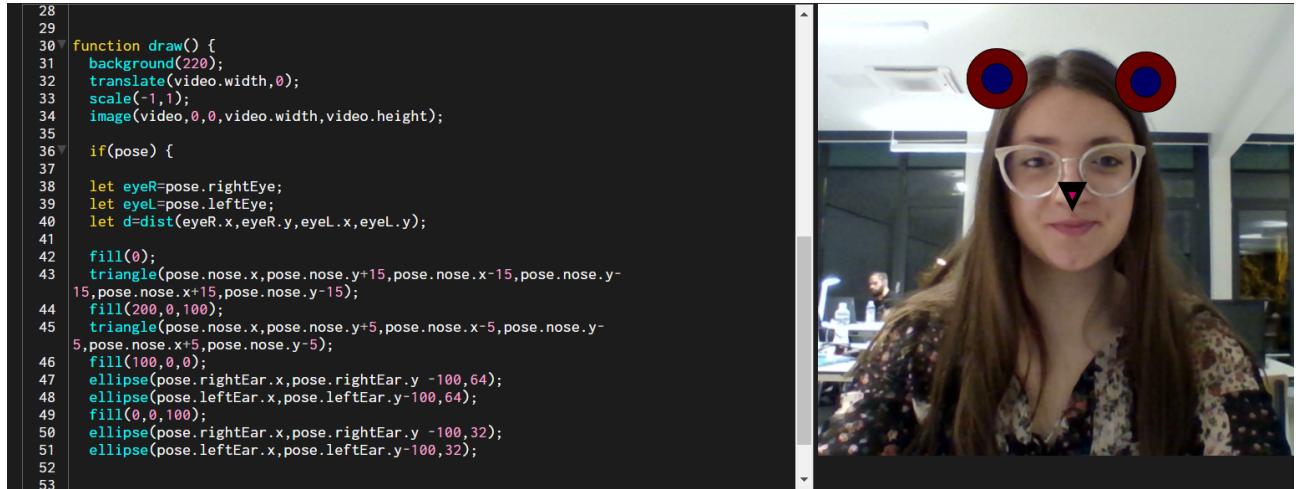


Slika 2.12: Nacrtana poza

#### 2.4.2. Filter za lice

PoseNet model prepoznaće ključne točke na mjestima gdje se nalaze oči, uši i nos. Zbog toga se može koristiti za izradu filtera za fotografije. Ovdje je napravljen demo-

filter, "efekt medvjedića". Naravno, postoje puno bolji modeli za izradu efekata za lice konkretno. Ovaj primjer pokazuje da se PoseNet model može koristiti za crtanje objekata na slici osobe dok osoba mijenja poze/ kreće se u realnom vremenu.



**Slika 2.13:** "Medvjedić" filter za lice

### 2.4.3. Igra

#### Prikupljanje poza - ulazni podaci za vlastiti model neuronske mreže

Igra koju sam zamislila treba funkcionirati na sljedeći način: - kada osoba koju kamera računala snima skoči u JumpingJack pozu, tada kuglica koja treba preskočiti zidić skače iznad zidića. Kada se osoba opet opusti, loptica pada na pod i slijedeći zidić nailazi. Tada osoba opet mora skočiti u JumpingJack pozu kako bi loptica preskočila zidić. Tako se dobije spoj sporta i e-sporta što je dosta zgodno i korisno kombinirati jer se tzv. gameri slabo gibaju dok se sportašima zagrijavanje može učiniti zanimljivijim.

Za početak, treba "uloviti" poze za prvu klasu i poze za drugu klasu. Na slici dolje je prikazan kod koji može poslužiti za to.

```

function keyPressed() {
  if (key == 's') {
    brain.saveData();
  } else {
    targetLabel = key;
    console.log(targetLabel);
    setTimeout(function() {
      console.log('collecting');
      state = 'collecting';
      setTimeout(function() {
        console.log('not collecting');
        state = 'waiting';
      }, 10000);
    }, 10000);
  }
}

function setup() {
  createCanvas(640, 480);

  video = createCapture(VIDEO);
  video.hide();
  poseNet = ml5.poseNet(video, modelLoaded);
  poseNet.on('pose', gotPoses);

  let options = {
    inputs: 34,
    outputs: 2,
    task: 'classification',
    debug: true
  }
  brain = ml5.neuralNetwork(options);
}

```

Slika 2.14: "Hvatanje" poza preko kamere

```

function gotPoses(poses) {
    // console.log(poses);
    if (poses.length > 0) {
        pose = poses[0].pose;
        skeleton = poses[0].skeleton;
        if (state == 'collecting') {
            let inputs = [];
            for (let i = 0; i < pose.keypoints.length; i++) {
                let x = pose.keypoints[i].position.x;
                let y = pose.keypoints[i].position.y;
                inputs.push(x);
                inputs.push(y);
            }
            let target = [targetLabel];
            brain.addData(inputs, target);
        }
    }
}

```

Slika 2.15: "Hvatanje" poza preko kamere

Ovaj dio je osmišljen tako da se koriste funkcije keyPressed() i gotPoses(), brain.addData i brain.saveData() te argumenti brain, inputs i target. Kada se pokrene snimanje poza pomoću keyPressed(), imamo nekoliko sekundi za namještanje i onda se poze snimaju te se pomoću gotPoses() i addData() dodaju u varijablu brain. Kada se na konzoli ispiše "not collecting" - prestaje snimanje. Sve te poze su spremljene u varijablu brain koja je mozak koji ima neuronsku mrežu. U "options" je definirano od čega se mreža sastoji, a sastoji se od 34 ulaza (17 x koordinata i 17 y koordinata za 17 ključnih točaka). Ima 2 izlaza, a to su : klasa1 (JumpingJack poza) i klasa2 (ostale poze). Zadatak je klasifikacija. Spremamo model kao JSON datoteku pomoću funkcije saveData().

### Treniranje modela

JSON datoteka u kojoj se nalaze podaci za model se učitava pomoću funkcije loadData(). U "options" su definirane karakteristike neuronske mreže. Pomoću funkcije normalizeData() iz dataReady() se normaliziraju podaci te se onda pomoću train() model mreže trenira, kako je navedeno 250 epoha. Nakon što se model utrenira, spremi se pomoću funkcije save() iz ml5 biblioteke.

The screenshot shows the TensorFlow.js Editor interface. On the left, the code editor displays `sketch.js` with the following content:

```
1 let brain;
2
3 function setup() {
4   createCanvas(640, 480);
5   let options = {
6     inputs: 34,
7     outputs: 2,
8     task: 'classification',
9     debug: true
10  }
11  brain = ml5.neuralNetwork(options);
12  brain.loadData('skokJack.json', dataReady);
13 }
14
15 function dataReady() {
16   brain.normalizeData();
17   brain.train({epochs: 250}, finished);
18 }
19
20 function finished() {
21   console.log('model trained');
22   brain.save();
23 }
```

The right side of the interface contains the Visor panel, which includes two main sections: "Training Performance" and "Model Summary".

**Training Performance**: A line chart titled "onEpochEnd" showing the loss value over 250 epochs. The y-axis is labeled "Value" and ranges from 0.00 to 0.45. The x-axis is labeled "Epoch" and ranges from 0 to 250. The "loss" series starts at approximately 0.45 and drops sharply to near zero by epoch 20, remaining low thereafter.

Layer Name	Output Shape	# Of Params	Trainable
dense_Dense1	[batch,16]	560	true
dense_Dense2	[batch,2]	34	true

**Model Summary**: A table showing the layers of the neural network. It lists the layer name, output shape, number of parameters, and whether it is trainable.

Slika 2.16: Treniranje modela

## Korištenje modela u igri

```
function setup() {
  createCanvas(640, 480);
  video = createCapture(VIDEO);
  video.hide();
  poseNet = ml5.poseNet(video, modelLoaded);
  poseNet.on('pose', gotPoses);

  player1 = new Player(140, 200);

  let options = {
    inputs: 34,
    outputs: 2,
    task: 'classification',
    debug: true
  }
  brain = ml5.neuralNetwork(options);
  const modelInfo = {
    model: 'model/model.json',
    metadata: 'model/model_meta.json',
    weights: 'model/model.weights.bin',
  };
  brain.load(modelInfo, brainLoaded);
}
```

Slika 2.17: setup() funkcija za igru

```
function brainLoaded() {
    console.log('pose classification ready!');
    classifyPose();
}

function classifyPose () {
    if(pose) {
        let inputs = [];
        for (let i = 0; i < pose.keypoints.length; i++) {
            let x = pose.keypoints[i].position.x;
            let y = pose.keypoints[i].position.y;
            inputs.push(x);
            inputs.push(y);
        }
        brain.classify(inputs,gotResult);
    } else {
        setTimeout(classifyPose,100);
    }
}

function gotPoses(poses) {
    if (poses.length > 0) {
        pose = poses[0].pose;
        skeleton = poses[0].skeleton;
    }
}
```

Slika 2.18: Klasifikacija poza pomoću modela

```

function gotResult (error, results) {
  if (results[0].confidence > 0.75) {
    poseLabel = results[0].label.toUpperCase();
  }
  console.log(results[0].confidence);
  label = results[0].label;
  SKOK();
  classifyPose();
}

function gotPoses(poses) {
  if (poses.length > 0) {
    pose = poses[0].pose;
    skeleton = poses[0].skeleton;
  }
}

function modelLoaded() {
  console.log('poseNet ready');
}

```

Slika 2.19: Funkcija gotResult()

Pomoću funkcije classifyPose() klasificiraju se poze u realnom vremenu koje se uzimaju s kamere računala. Argumenti te funkcije su, naravno, ulazi koji su trenutne poze i funkcija gotResult(). Funkcija gotResult() radi tako što, ako model daje rezultat u koji je siguran preko 75% , tada se pojavi pojavi odgovarajuća labela za rezultat. Također, funkcija spremi labelu u varijablu label i aktivira funkciju SKOK() te funkciju classifyPose();

```

function SKOK() {
    if (label == 'j')    //Skok *
        player1.jump();
}

function Player(x, y) {
    this.x = x;
    this.y = y;
    this.j = 2;

    this.display = function() {
        ellipse(this.x, this.y, 20);
        if (this.y < 200)
            this.y = this.y + 1;
        if(this.y == 200)
            this.j = 2;
    }

    this.jump = function() {
        if(this.j > 0) {
            this.y -= 90;
            this.j--;
        }
    }
}

```

**Slika 2.20:** Funkcija za skokove

SKOK() funkcija tako što - ako labela odgovara nazivu labele za klasu JumpinJack poze, onda se loptica pomiče na do odgovarajuće koordinate prema gore. Kada labela prestane odgovarati nazivu labele za skok, onda loptica pada na koordinatu y osi na kojoj je razina tla.

```
function Obstacle() {
    this.x = width;

    this.display = function() {
        rect(this.x, 210-60, 20, 60);
    }

    this.move = function() {
        this.x = this.x - 1;
    }
}
```

Slika 2.21: Funkcija za zidiće

Na slici iznad je prikazana funkcija za kretanje zidića prema loptici.

```

function draw() {
    push();
    translate(video.width, 0);
    scale(-1, 1);
    image(video, 0, 0, video.width, video.height);

    textSize(500);
    fill(255);
    //floor
    line(0, 210, width, 210);

    if (pose) {
        for (let i = 0; i < skeleton.length; i++) {
            let a = skeleton[i][0];
            let b = skeleton[i][1];
            strokeWeight(2);
            stroke(0);
        }
    }
    pop();

    player1.display();

    if(frameCount > timeWas+timer && timer != 0) {
        timeWas = frameCount;
        timer = random(100,500);
        obstacles.push(new Obstacle());
    }

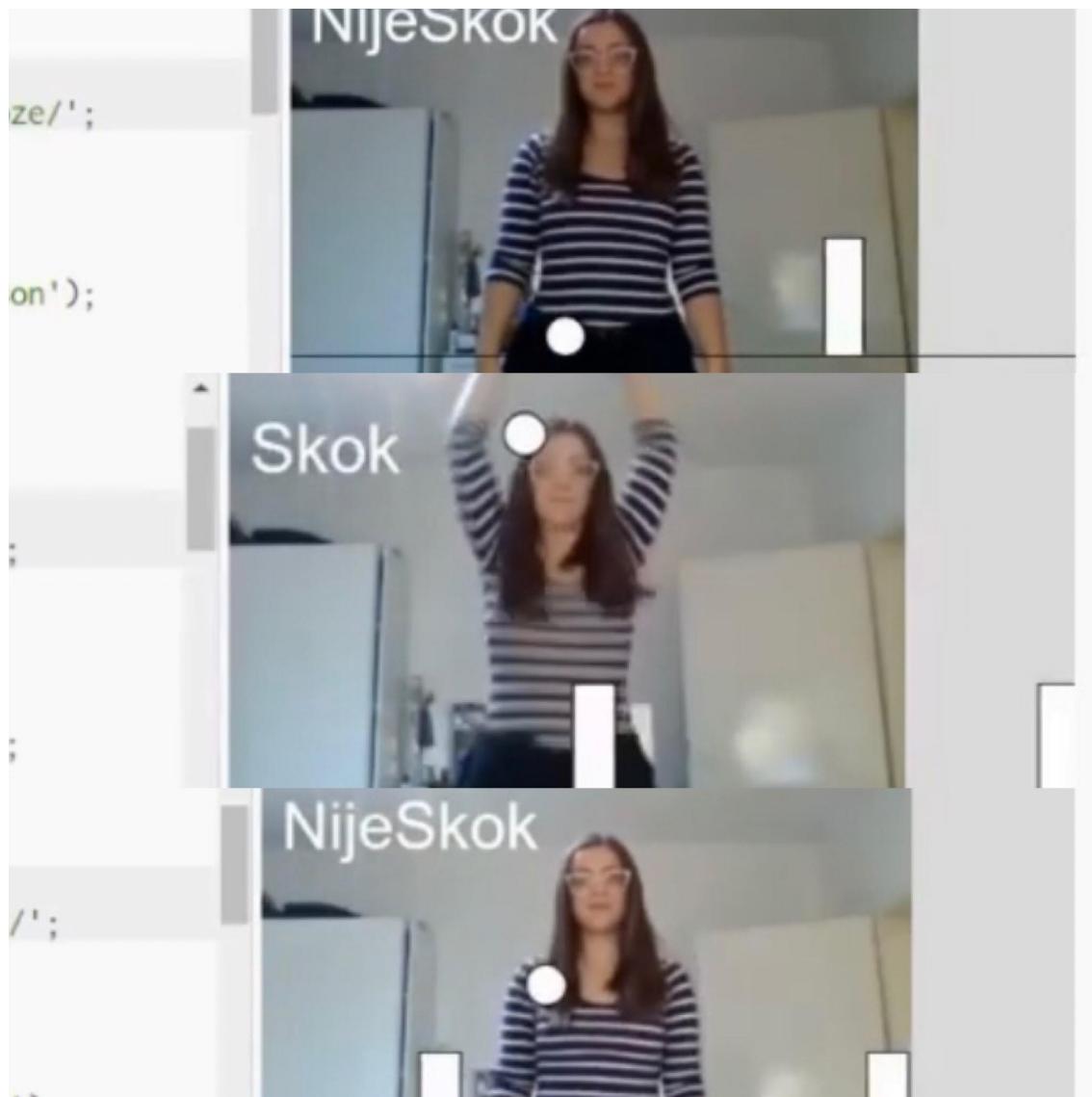
    for (var i = 0; i < obstacles.length; i++) {
        obstacles[i].display();
        obstacles[i].move();
        if (obstacles[i].x < -25)
            obstacles.splice(0, 1);
    }

    fill(255, 0, 255);
    noStroke();
    textSize(50);
    text(poseLabel,10, 50);
}

```

Slika 2.22: Funkcija za crtanje okruženja i objekata u igri

U funkciji draw() su samo pozvane sve navedene funkcije kako bi se to sve prikazalo kada se pokrene igra.



**Slika 2.23:** Igra

Na slici iznad je prikazano kako igra funkcioniра kada se pokrene.

### **3. Zaključak**

Područje računarnog vida je veoma uzbudljivo. TensorFlow je nadasve korisna biblioteka koja pruža beskrajne mogućnosti kada se kombinira s kreativnim idejama. Konkretno, PoseNet model za detekciju poze sadrži algoritme koji se daju pisati u Colabu koristeći Python što ih čini jednostavnim za rukovanje i bliskim svim onima koji su tek zaplivali vodama umjetne inteligencije i koji se tek upoznaju s divljinom koja je u interakciji s UI.

## 4. Literatura

- [1] <https://www.tensorflow.org/>. TensorFlow.
- [2] Ivan Kunyakin. <https://medium.com/@ikunyankin>. Uspoređivanje poza na slikama pomoću PoseNet-a.
- [3] Dan Oved. <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js---o-posenet-u>.
- [4] Daniel Shiffman. <https://www.youtube.com/watch?v=OIo-DIOkNVg>. The Coding train - tutorijal za korištenje PoseNet-a u projektima.

## 5. Sažetak

Strojno učenje je grana umjetne inteligencije koja ima primijenu u području računalnogvida kojemu pripada mnoštvo interesantnih tehnika, a jedna od njih je i detekcija položaja tijela osobe. Za uporabu navedene tehnike, u biblioteci TensorFlow je sadržan PoseNet model koji sadrži algoritme za procjenu poze jedne osobe odnosno procjenu poza kada je više ljudi na slici / snimci. Prva faza u procjeni poze je ulaz koji je RGB slika i prolazi kroz konvolucijsku neuronsku mrežu. Algoritam za detekciju poze pronalazi ključne točke koje su zapravo zglobovi. To radi tako što prvo stvori topinsku mapu koja daje vjerojatnost za svaku ključnu točku gdje bi se ona trebala nalaziti. Uzima se najviša vjerojatnost svake točke i onda se vektorima pomaka utvrđuju točne koordinate ključne točke. Zatim se točke povezuju linijama kako bi se dobila smislena tjelesna figura i u konačnici se crta poza osobe. Kada imamo poze, možemo ih klasificirati dalje pomoću neuronskih mreža te ih koristiti za projekte kao što je dan jednostavan primjer filtera za lice ili za neke komplikiranije poput računalnih igara, aplikacija za fitness, strojeva osjetljivih na položaj tijela osobe te za mnogobrojne druge kreativne i korisne sadržaje.