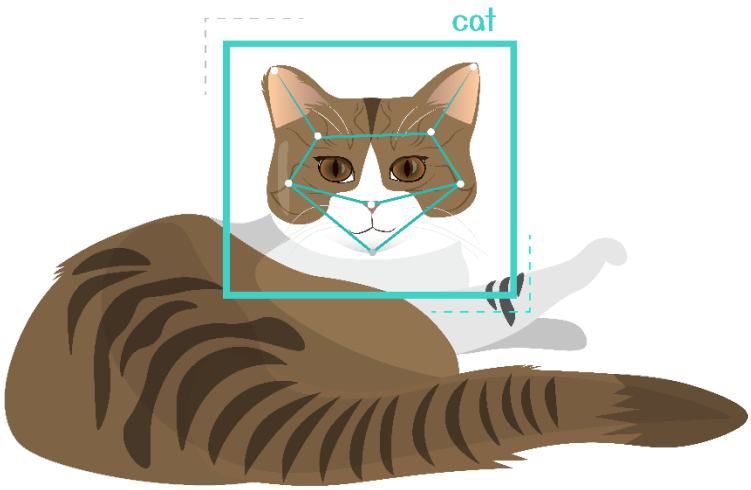
Veštačka inteligencija

Računarska vizija - prepoznavanje objekata primenom open-source biblioteka



Članovi tima ***Lispice***:

Mina Nikolić ***16786***

Mila Mirović ***16742***

1. Odeljak 1: Kratak opis problema

Problem ovog rada se odnosi na prepoznavanje objekata na slici, video snimku ili real-time prepoznavanje pomoću web kamere, a konkretno se bazira na detekciji i klasifikaciji lica mačaka (koje je u ulozi objekta). Proces koji je relativno jednostavan za ljude može biti prilično kompleksan sa strane računara tj. veštačke inteligencije koja stoji iza njega. Karakteristike tipične mačjoj vrsti čine njihovo lice tipom objekta koje se može detektovati raznim metodama računarske vizije. Kao jedan od najkorišćenijih metoda do sada za realizaciju i rešavanje ovog problema izdvaja se ***algoritam Viola-Jones***, koji potencira učenje na modelima lica koji su usmereni frontalno ka kameri, a ne u slobodnom uglu. Opisan je proces izrade modela klasifikacije i detekcije lica, kao i način i svrha korišćenja funkcija open-source biblioteke OpenCV i upotreba HAAR kaskadnih algoritama radi detekcije objekata.

2. Odeljak: Pregled i kratak opis tehnika i algoritama

Tehnike koje se koriste za detekciju objekata dele se na metode zasnovane na *mašinskom učenju* i metode zasnovane na *dubokom učenju*. Metode mašinskog učenja prvo definišu fičere, a zatim izvršavaju klasifikaciju. Tehnike dubokog učenja su sposobne da izvrše detekciju objekata bez nužne specifikacije fičera i one se tipično zasnivaju na konvolucionim neuronskim mrežama.

Tehnike koje koriste principe mašinskog učenja:

- **Viola-Jones framework za detekciju objekata**
  Algoritam za brzi pronalazak objekata na slici Haar kaskadnih klasifikatora. Prvenstvena namena je detekcija ljudskih lica, ali se može primeniti na detekciju bilo kog objekta. Zasniva se na korišćenju Haar fičera, AdaBoost algoritma i integralnih slika. U procesu učenja koriste se pozitivne i negativne oblasti, gde su pozitivne one na kojima se nalazi objekat od interesa, a negativne su one na kojima se nalazi sve ono što nije objekat od interesa. *Ovaj algoritam je detaljnije razrađen u dokumentu u prilogu.*
- **Scale-invariant feature transform (SIFT)**
  Algoritam za otkrivanje fičera slika u računarskom vidu koji je invarijantan na transformacije nad slikom. Ovaj algoritam je patentiran pa je ključni deo OpenCV-a. Sadržaj slike se prevodi u fičere koji se nalaze na određenim koordinatama i koji su invarijantni na fotometrijske transforamcije.
- **Histogram orijentisani gradijenti (HOG)**
  Pod HOG-om podrazumevamo vektor fičera koji se koristi u raznim tehnikama računarskog vida (obrada slike), a koji je sastavljen od određenog broja diskretnih histograma dobijenih obradom slike nad ćelijama u prozoru. Osnovna ideja je podeliti

sliku u više delova, nakon čega sledi izrada histograma koji govore o učestalosti smera gradijenata elemenata slike unutar svakog dela. Ova tehnika ne zahteva normalizaciju boje ili osvetljenja, jer je invarijantna na manja pomeranja i rotacije (dakle pomeraj/rotacija slike ne utiče preterano na rezultat), a pored toga poseduje normalizaciju kontrasta koja obezbeđuje invarijantnost na promene u osvetljenju.

Metodi koji se baziraju na neuronskim mrežama obuhvataju više različitih pristupa medju kojima možemo izdvojiti sledeće :

- **R-CNN** predstavlja algoritam koji se sastoji od dve faze. Prva faza bi podrazumevala identifikaciju podskupa regiona na slici koji mogu da sadrže objekat, dok je cilj druge faze klasifikacija objekata unutar svakog regiona. Njegova primena se ogleda u samovozećim automobilima, pametnim sistemima za nadzor i prepoznavanju lica. R-CNN detektor najpre generiše predloge regiona koristeći algoritam poput Edge Box-a. Predloženi regioni se zatim izdvajaju sa slike i vrši se promena njihovih dimenzija. Nakon toga konvoluciona neuronska mreza klasifikuje date regione. Okviri predloženih regiona se dalje obradjuju od strane SVM-a (support vector machine) koji se trenira fičerima konvolucione neuronske mreže.
- **Fast R-CNN** takodje koristi algoritam poput Edge Box-a za generisanje predloga regiona. Za razliku od R-CNN-a koji vrši izdvajanje i promenu dimenzija regiona, Fast R-CNN vrši procesiranje celokupne slike. Dok R-CNN mora da izvrši klasifikaciju svakog regiona, Fast R-CNN objedinjuje CNN fičere koji odgovaraju svakom predlogu regije. Smatra se efikasnijim algoritmom zato što se dele proračuni za preklapajuće regione i na taj način se vrši ušteda vremena.
- **Faster R-CNN** dodaje mrežu predloga regiona (RPN- region proposal network) kako bi se direktno vršilo generisanje predloga regiona usmesto korišćenja eksternih algoritama kao što su Edge Box algoritmi. RPN koristi Anchor Boxes za detekciju objekata.
- **YOLO (You Only Look Once)** je algoritam za detekciju objekata koji unificira komponente detekcije objekata u jednu jedinstvenu neuronsku mrežu pa na detekciju objekata gleda kao na problem regresije. Ulaznu sliku deli na pravougaone ćelije tj. na matricu ćelija, a u svakom delu matrice predviđa se određeni broj graničnih pravougaonika (bounding box-a) i dobija se procenat koji predstavlja verovatnoću da je unutar tog graničnog prvaouganonika objekat. Pravo pitanje je šta nam zaprvo ova verovatnoća govori. Ona praktično označava koliko je model siguran da neka ćelija u matrici sadrži objekat od interesa, pa je u idealom slučaju vrednost verovatnoće 0 (ili približno) ako ćelija ne sadrži objekat ili 1 (ili približno) ako ga sadrži.

3. Odeljak: Formulacija problema na način kako to odgovara izabranoj tehnici/algoritmu

Konkretan problem, koji je već opisan u prvom odeljku ovog dokumenta, rešiv je primenom algoritma Viola-Jones. Prepoznavanje lica mačaka može se primeniti u konkretnijem problemu, kao što je detekcija direktnog pogleda mačke (ili bilo koje druge životinje) u kameru. Kao algoritam sa ograničenjem da trening i test podaci moraju biti slike i to upravo frontalnog prikaza objekta/lica, u potpunosti odgovara navedenom problemu čije rešenje želimo prikazati. Implementacija ovog algoritma se može obaviti na prilično intuitivan način pomoću open-source biblioteke ***OpenCV*** i ***Python***-a kao što je opisano u sledećem odeljku.

4. Kratak opis našeg rešenja

Za praktičnu primenu principa računarske vizije je korišćen OpenCV. OpenCV je open-source biblioteka za računarsku viziju, mašinsko učenje i procesiranje slika koja igra veliku ulogu u operacijama koje zahtevaju izvršenje u realnom vremenu. Korišćenjem ovog alata moguće je procesirati slike i video materijal, identifikovati lica, rukopis, kao i veliki broj drugih objekata. Biblioteka je dostupna za akademsku i komercijalnu upotrebu, podržan je rad na operativnim sistemima Linux, Mac OS i Windows (moguć je rad i na nekim mobilnim sistemima). Omogućava rad na raznim programskim jezicima poput C++ -a, C-a, Jave i Python-a. OpenCV je instaliran zadavanjem komande:

```
pip install opencv-contrib-python
```

pri čemu je koriscena verzija Pythona 3.8.7. Editor koji je izabran za rad je Visual Studio Code.

Kod koji je implementiran se nalazi u Python skriptama, pod nazivima detekcija_macaka.py , detekcija_macaka_video.py i detekcija_macaka_web_cam.py.Unosom odgovarajuće komande u terminal editora (korišćen je VS Code) će se prikazati rezultujuća slika ili video snimak. Izlazak iz video snimka se vrši klikom na taster '"***q***".

```
> python detekcija_macaka.py
```

Za praktičnu implementaciju ovog algoritma je najpre potrebno učitati željenu sliku i konvertovati je u grayscale režim. To se može uraditi na sledeći način:

```
img = cv.imread('photos/liza.jpg')
#cv.imshow('Macka', img)

gray= cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#cv.imshow('Siva macka', gray)
```

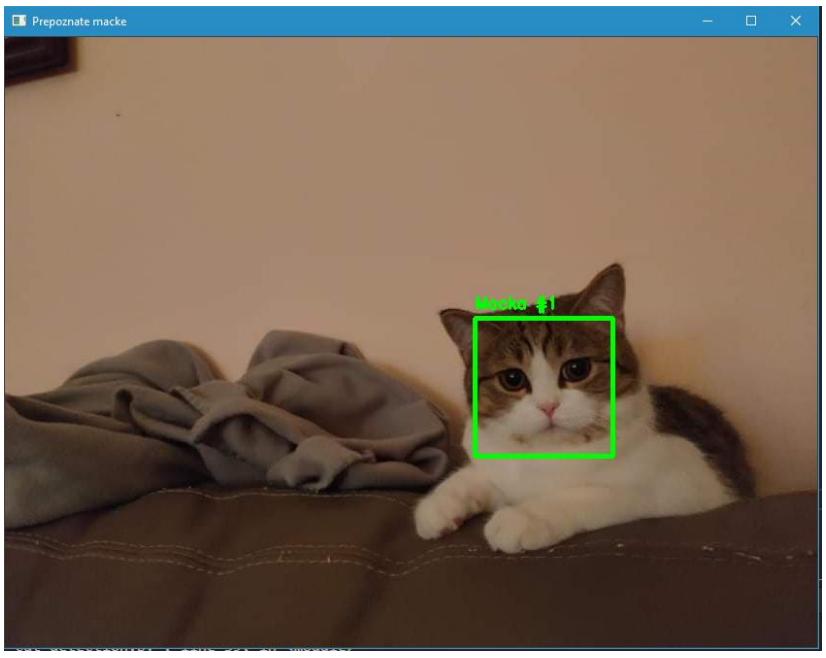
Nakon toga se izvršava funkcija kaskadnog klasifikatora :

```
haar_kaskada = cv.CascadeClassifier("cat_face_detector.xml")
```

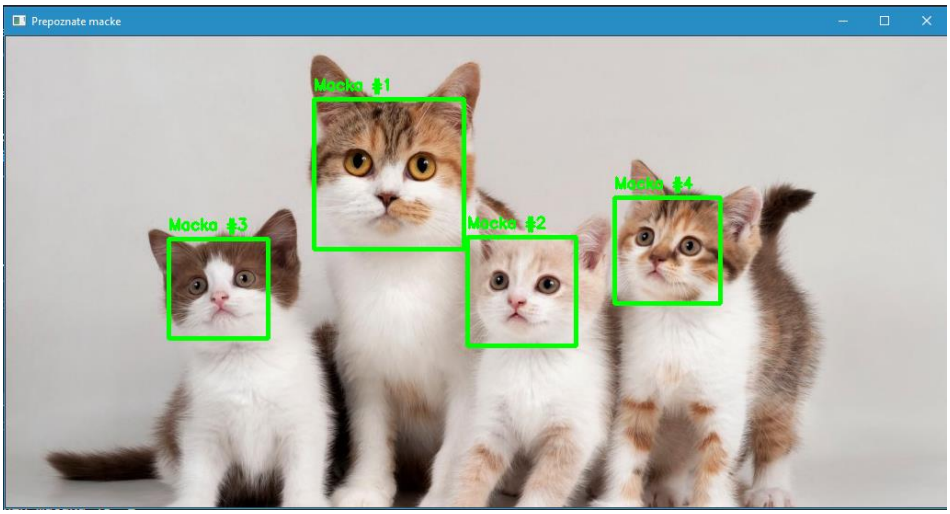
Promenom parametara funkcije ***detectMultiScale*** možemo da dobijemo različite rezultate za istu prosleđenu sliku. Parametri koji su ovde uzeti u obzir su ***scaleFactor*** i ***minNeighbors***. Prvi parametar se koristi da se kreira scale piramida. Tokom treniranja model ima fiksnu veličinu. To znači da se ta veličina lica detektuje na slici ako postoji. Ali reskaliranjem ulazne slike može se vršiti promena veličine većeg lica ka manjem čime će postati moguće da bude detektovano od strane algoritma. Uzimanjem manje vrednosi za faktor skaliranja (npr 1.05) znači da smanjujemo veličinu slike za 5% i povećavamo šansu pronalaska lica. Drugi parametar specificira koliko suseda svaki kandidat pravougaonik treba da ima da bi bio smatran pravougaonikom lica. Povećavanjem broja suseda moguće je eliminisati lažno pozitivne vrednosti, ali se isto tako mogu izgubiti i stvarne pozitivne vrednosti.

Korišćenjem sledećih vrednosti, za prosledjenu sliku dobijamo ovakav rezultat:

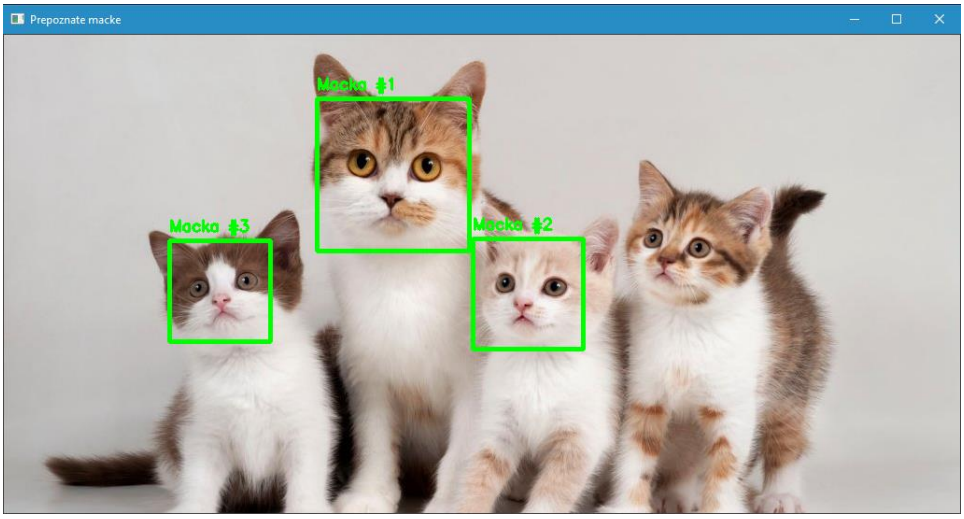
```
macka_pravougaonik = haar_kaskada.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3)
```



Sa istim vrednostima faktora skaliranja i broja suseda je moguće detektovati i mačke na sledećoj slici:



Međutim, ako bismo broj suseda povećali na 5, jedna od mačaka ne bi bila detektovana:



Ukoliko bismo povećali i faktor skaliranja na 1.4 dobili bismo prikaz gde nijedna od mačaka nije detektovana:

Na osnovu koda za detekciju slika, moguće je uraditi i detekciju mačaka na video snimcima, budući da video snimak možemo posmatrati kao niz frejmova u datom trenutku. Učitavanje video snimka se vrši na sledeći način:

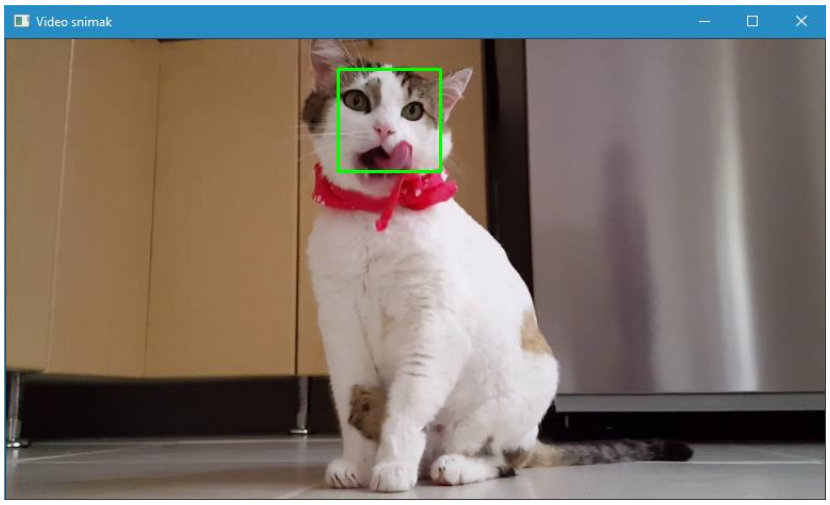
```
cap = cv.VideoCapture('videos/macka4.mp4')
```

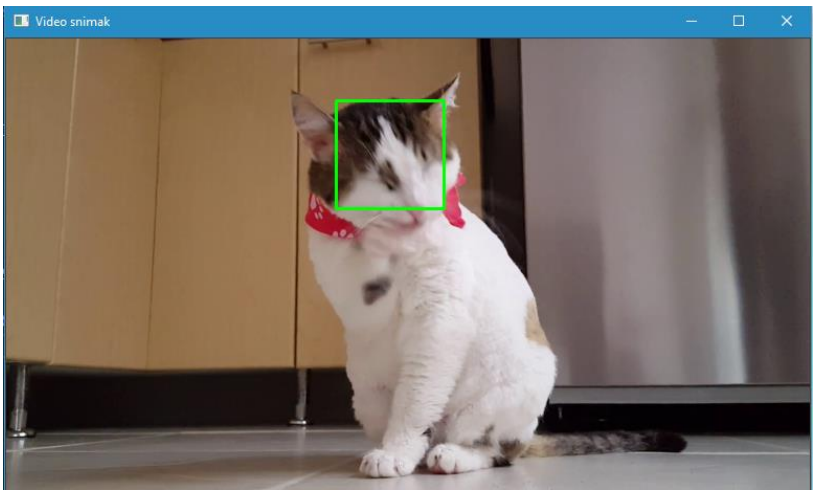
Primećeno je da bolje rezultate postižemo sa video snimcima manjih dimenzija tako da je primenjena sledeća modifikacija, odnosno reskaliranje:

```
def rescaleFrame(frame, scale =0.40):
    width = int(frame.shape[1]*scale)
    height= int(frame.shape[0]*scale)
    dimensions = (width,height)
    return cv.resize(frame,dimensions, interpolation=cv.INTER_AREA)
```

Izlazak iz video snimka se vrši klikom na taster ''***q***''.

Prikaz nakon pokretanja *python* skripte je sledeći:





Budući da je data set na kojem je treniran klasifikator namenjen detekciji frontalno orijentisanih objekata, u trenucima kada mačka pomeri glavu u stranu dešava se da dodje do prekida detekcije. Kvalitet detekcije će i u ovom slučaju zavisiti od podešavanja parametara funkcije ***detectMultiScale*** pa su vrednosti koje su izabrane za ovaj konkretan snimak date isečkom:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.1,2)
```

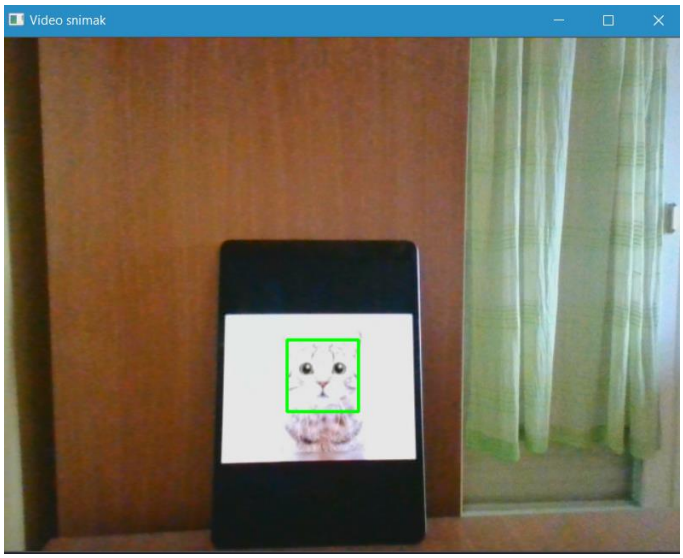
Implementirana je još jedna varijanta detekcije koja se odnosi na prikaz preko web kamere. Izmene u odnosu na kod koji je naveden za video snimak su minimalne i ogledaju se u tome da nije potrebno vršiti promenu dimenzija video snimka i umesto učitavanja sa fajl sistema vrši se detekcija na osnovu prikaza web kamere računara. Izlazak iz video snimka se vrši klikom na taster "q". Sledeća linija koda će omogućiti prikaz preko web kamere:

```
cap = cv.VideoCapture(0)
```

Parametri koji su korišćeni pri detectMultiScale funkciji su sledeći:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.06,3)
```

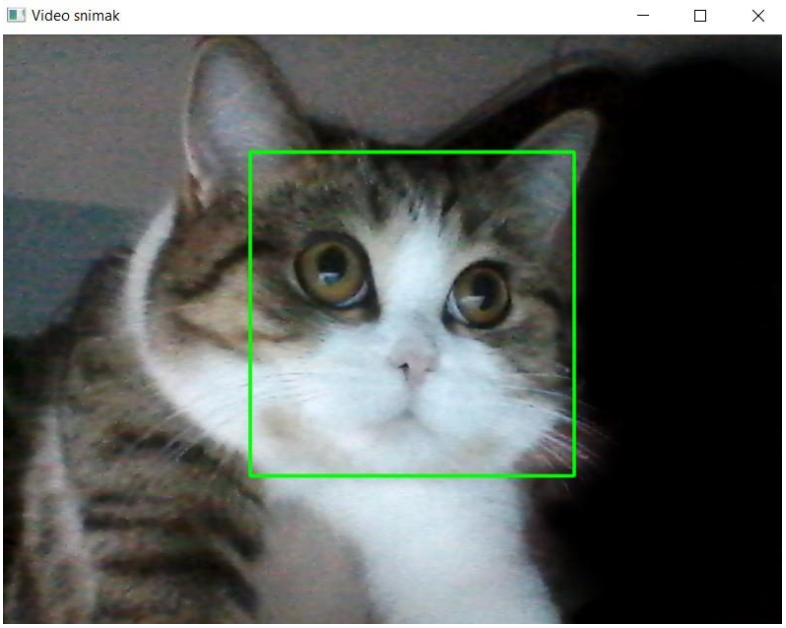
Testiranje je vršeno i na osnovu slike mačke sa tablet računara, ali i na pravoj mački. Rezultati su dati na sledećim slikama:

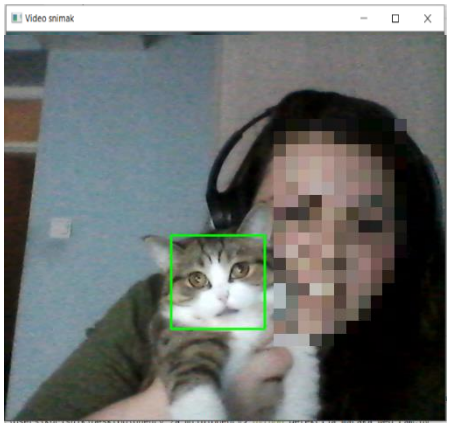


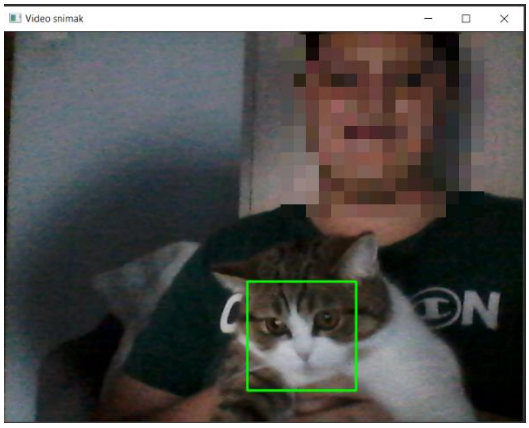
Prikaz testiranja sa pravom mačkom je dat na sledećim slikama pri čemu je izvršena sledeća promena parametara u detectMultiScale funkciji:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.08,5)
```

Promena parametara može da dovede do situacije da i ljudi budu prepoznati.







5. Zaključak

U toku proučavanja široke oblasti računarske vizije, naišle smo na teme koje su nam posebno zaokupirale pažnju i navele nas na detaljniji istraživački rad. Kao produkt našeg interesovanja i želje da se dublje posvetimo ovoj oblasti, nastao je dokument gde smo sa mnogo više detalja opisale pojam računarskog vida i nekih primena u toj oblasti, tehnike koje se najčešće upotrebljavaju, način funkcionisanja biblioteke OpenCV, kao i detaljnu implementaciju. Da bi se formalno ispoštovali zahtevi navedeni za realizaciju drugog projekta iz predmeta Veštačka inteligencija, napisana je kraća verzija u vidu izveštaja, dok se srž našeg interesovanja i istraživanja nalazi u gore pomenutom dokumentu. Sva korišćena literatura navedena je u sekciji *Reference* tog dokumenta. Od velikog značaja pri detekciji lica mačke na slici je korišćeni tutorijal koji se nalazi na sledećem linku: https://youtu.be/oXlwWbU8l2o, na osnovu koga je kasnije vršena implementacija detekcije mačke na video snimku i na prikazu sa web kamere. Tutorijal nam je pored toga pružio i osnovne informacije o korišćenju OpenCV biblioteke. U projektu se nalaze i fajlovi koji su korišćeni radi vežbanja, ali nisu namenjeni za pokretanje konkretnog projekta. Tri python skripte u kojima je realizacija projekta su navedene u delu za implementaciju i samo njih treba izvršavati. Teorijsku osnovu za razumevanje osnovnih koncepata računarske vizije pružila su nam odabrana predavanja iz predmeta *Računarski vid*, profesora Aleksandra Milosavljevića, na Elektronskom fakultetu u Nišu. Nadamo se da ćemo imati prilike da se dublje bavimo ovom oblašću, pogotovo detekcijom objekata primenom tehnika dubinskog učenja i neuronskih mreža.