Univerzitet u Beogradu
Elektrotehnički fakultet

Pronalaženje i prepoznavanje igrača tokom košarkaške utakmice

13E064SDOS: Sistemi za digitalnu obradu slike

Mentori:
Prof. dr Marko Barjaktarović
Prof. dr Ana Gavrovska

Student:
Božidar Obradović
2017/0113

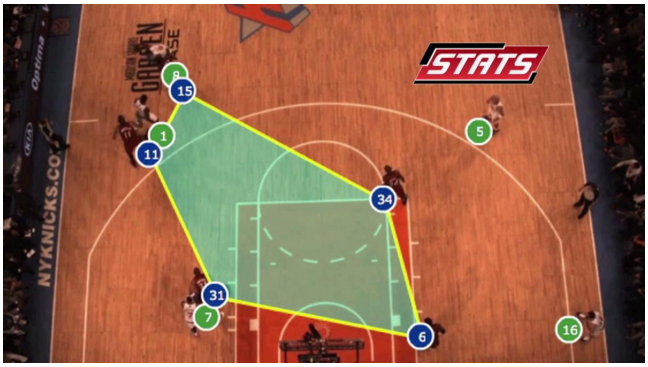
Beograd, 2021

Sadržaj

1 Uvod

Globalna sportska industrija vrednuje se na 500 milijardi dolara i predviđa se da će dostići 615 milijardi dolara do 2022. Sportski događaji okupljaju ljude raznih pozadina prevazilazeći rase, etničke pripadnosti, religije i društveno-ekonomske klase, čineći sport centralnom komponentom našeg savremenog društva. Zbog svoje važnosti i neprekidnog, brzog rasta širom sveta postoji značajno interesovanje za primenu tehnika obrade slike i kompjuterske vizije u sportu za optimizaciju i same igre i iskustva gledalaca poslednjih godina [1]. Kako se količina novca u sportu značajno povećava, timovi mnogo više ulažu u prikupljanje statističkih podataka o svojim sportistima.

Konkretno, mogućnost praćenja sportista tokom mečeva ima veliki potencijal. Vlasnik i trenersko osoblje sportskog kluba su veoma zainteresovani za praćenje performansi i zdravlja svojih sportista. Imajući u vidu sportske ugovore koji dostižu astronomske iznose i povrede koje nastavljaju da pljačkaju sve sportove od njihovih najpopularnijih i najzanimljivijih sportista, mogućnost praćenja i nadgledanja opterećenja tokom igre može da smanji rizik od povrede i samim tim da sačuva milione dolara klubovima. Sa taktičke strane, mogućnost praćenja igrača može da obezbedi informacije o akcijama, formacijama i strategijama protvnika (Slika 1). Takođe, praćenje igrača može da poboljša video emitovanje automatskim prebacivanjem na kameru sa najboljim uglom gledanja ili fokusiranjem na najbitnije igrače, čime bi se olakšao, ili u budućnosti čak i ukinuo, posao kamermana.



Slika 1: Primer sistema za praćenje igrača implementiranog u NBA ligi

U ovom radu je predstavljen algoritam koji pronalazi i prepoznaje igrače tokom košarkaške utakmice. Fokus je bio na košarci pre svega zbog afiniteta autora prema njoj, ali i zbog precizno definisanog terena, kao i manjeg broja igrača na njemu; to čini košarku idealnim sportom za razvijanje *proof-of-concept*[1] algoritma. Konačni rezultat je prikaz igrača, razdvojenih po timovima, na 2D mapi košarkaškog terena; sa kretanjem igrača na terenu tokom snimka njihove pozicije se menjaju na mapi.

Ostatak rada je predstavljen na sledeći način: poglavlje 2 opisuje sam algoritam i metode korišćene u njemu; rezultati algoritma su predstavljeni i diskutovani u poglavlju 3, a moguće primene algoritma i dalji pravci istraživanja su opisani u poglavlju 4.

[1]Izvođenje ideje ili metode u cilju potvrde njene ostvarivosti ili demonstracija koja potvrđuje njihov praktičan potencijal.

2 Metodologija

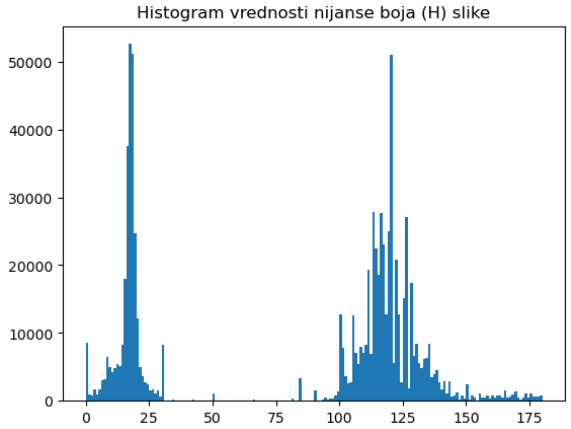
Program algoritma je napisan u *Python* programskom jeziku. Algoritam je osmišljen tako da može da se koristi i za druge sportske događaje. Ideja vodilja je da što više elemenata bude automatizovano, bez potrebe za akcijama korisnika. Snimak korišćen u ovom radu je sa utakmice univerziteta Oklahoma State i Liberty [2].

Razvijeni algoritam se sastoji od četiri koraka: **izdvajanje terena**, **detekcija igrača**, **klasifikacija po boji**, **pronalaženje koordinata** i **mapiranje u 2D ravan**.

2.1 Izdvajanje terena

Prvi korak algoritma je izdvajanje terena sa slike radi uklanjanja uticaja publike u pozadini, kao i ostalih šumova i smetnji. Ovo je u RGB sistemu boja jako teško izvesti; zbog neuniformnog osvetljenja i prisustva senki vrednosti piksela terena se znatno razlikuju. Iz tog razloga slike dobijene iz snimka su prevedene u HSV (hue, saturation, value - nijansa, zasićenost i jačina, respektivno) model boja. Nijansa boje (H) predstavlja njenu talasnu dužinu, na koju osvetljene i senke ne utiču, što je čini mnogo pouzdanijom prilikom detekcije boja.

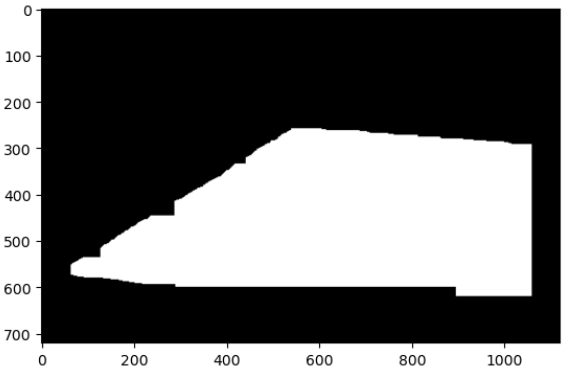
Za svaki frejm snimka određen je histogram vrednosti u H prostoru na osnovu kog je boja terena određena kao vrednost koja se najčešće pojavljuje na histogramu (Slika 2). Ova vrednost u opsegu ±7 je korišćena za binarizovanje slike (Slika 3).



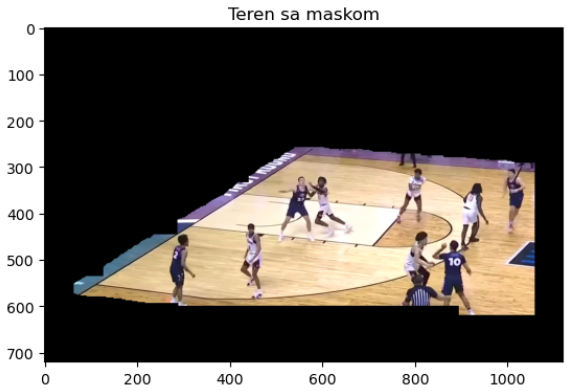
Slika 2: Histogram nijanse za jedan frejm

Slika 3: Slika binarizovana u opsegu ±7

Pošto je pozicija semafora ista na svakom frejmu on se jednostavno filtrira. Ovo je bitan korak pošto je semafor napravljen tako da odgovara bojama ekipa, što može da utiče na detekciju po bojama. Na ovako dobijenu sliku su primenjene morfološke operacije (otvaranje, zatvaranje i dilatacija, respektivno, Slika 4) za uklanjanje artifakata koji nisu vezani za teren. Konačno, ova maska je korišćena za procesiranje ostatka slike (Slika 5).



Slika 4: Morfološke operacije na maski

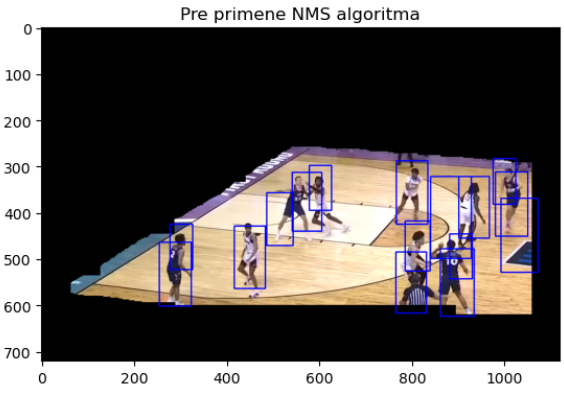


Slika 5: Teren sa primenjenom maskom

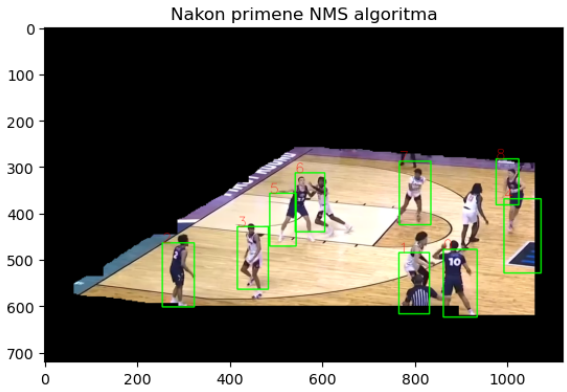
2.2 Detekcija igrača

Nakon primene maske na sliku ostaju samo igrači na terenu. Za njihovu detekciju je korišćena metoda histograma usmerenih gradijenata (HOG - *Histogram of Oriented Gradients*). Ona formira histograme smerova gradijenata u lokalizovanim delovima slike koji mogu da se koriste za prepoznavanje objekata na slici. Koristeći ovo obeležje klasifikatori na bazi mašinskog učenja, kao što su vektorske mašine za podršku (SVM - *Support Vector Machine*), mogu da se koriste da prepoznaju objekat sa slike na osnovu trening seta. HOG obeležje se pokazalo kao najpouzdanije prilikom detekcije pešaka [3].

U ovom projektu je korišćen HOG detektor iz OpenCV biblioteke, pre svega zbog činjenice da OpenCV već ima podrazumevanu bazu podataka za detekcije pešaka, kao i da su računanje HOG obeležja i SVM efikasno implementirani [4]. Konkretno, korišćena je *Daimler* baza podataka; ovaj detektor je treniran sa prozorom veličine (48, 96) piksela, samim tim, HOG detektor očekuje objekte da budu bar te veličine. Za detekciju je korišćena slika u RGB prostoru boja pošto je HOG detektor kvalitetnije detektovao u ovom prostoru. Izlaz detektora su koordinate temena pravougaonika, tj. okvir u kojem je detektovan objekat (Slika 6).



Slika 6: Izlaz detektora pre NMS



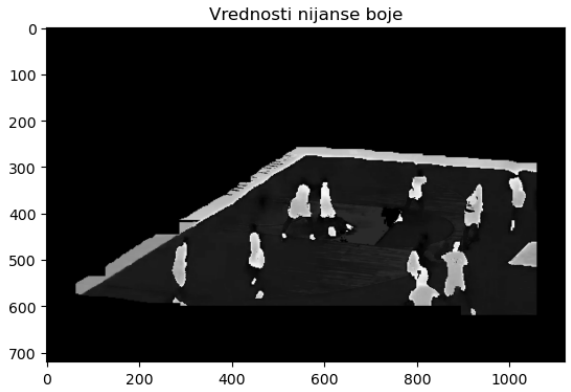
Slika 7: Izlaz detektora nakon NMS

Veoma često detektor pronađe više objekata koji se preklapaju. Ovo je rešeno primenom *non-maximum suppresion* (NMS) algoritma u implementaciji Tomasz Malisiewicz-a [5]. Ovaj algoritam uzima uzima pravougaonike koji se preklapaju u određenoj meri i spaja ih u jedan; kao rezultat, svaki okvir odgovara jednom objektu na slici (Slika 7).

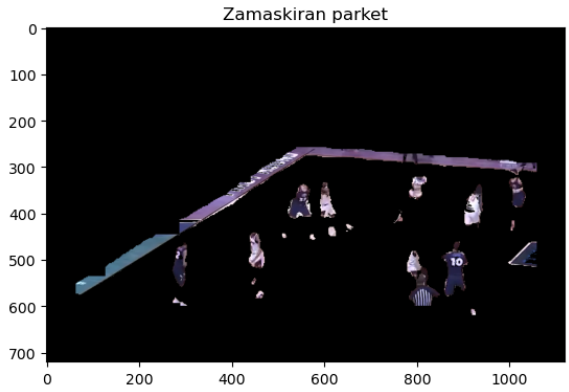
2.3 Klasifikacija po boji

U sportskim događajima, radi lakšeg prepoznavanja između igrača i bolje preglednosti za gledaoce, i uživo i preko televizora, boje dresova ekipa se razlikuju. Uglavnom je ekipa domaćin (ili koja se tako vodi, u ovom slučaju Oklahoma State) u svetloj garnituri, najčešće beloj, dok je gostujuća ekipa (Liberty) u dresovima tamnije boje; ova činjenica je iskorišćena za raspoznavanje igrača po ekipama.

Iz razloga opisanih u odeljku (2.1) dalja obrada slike je vršena u HSV prostoru boja. Igrači obe ekipe imaju velike vrednosti nijanse (H $>$ 200) u poređenju sa parketom (H $<$ 40) što otvara mogućnost ovakvog izdvajanja (Slika 8). Osim samog parketa zamaskirane su, u meri koliko je to bilo moguće, NCAA oznake (specifična jarka plava + crno blizu polovine terena) i ljubičasta pruga pored igrališta. Rezultat ovoga je zamaskiran sam teren tako da ostanu samo igrači (Slika 9).

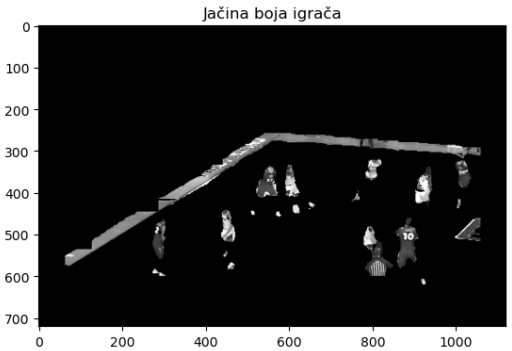


Slika 8: Nijansa boje Slike 5

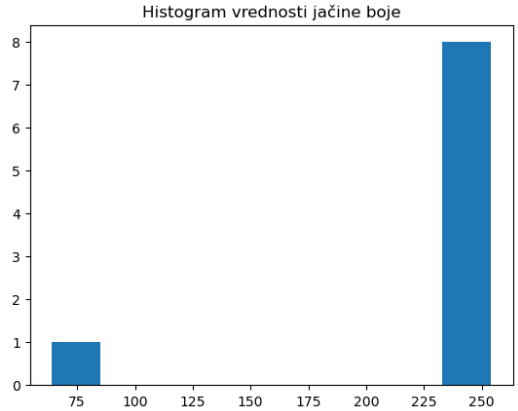


Slika 9: Slika nakon maskiranja parketa

Oklahoma State igra u dresovima bele boje, koja je u HSV prostoru boja predstavljena kao (0°, 0%, 100%), Liberty igra u dresovima mornarski plave (*navy blue*) boje, koja je u HSV prostoru predstavljena kao (216°, 87.2%, 30.6%); uočava se značajna razlika između ove dve boje u zasićenosti i jačini. Za svaki prepoznat objekat, tj. za svaki dobijeni okvir, na Slici 9 pronađene su najveće vrednosti zasićenosti i jačine i dodate u jedan niz (po komponenti). S obzirom na razlike u vrednostima očekuje se da histogram u oba slučaja bude bimodalan. Ispostavilo se da je razlika u jačinama dosta izraženija (V $>$ 200 za Oklahoma State, $<$ 100 za Liberty, Slike 10 i 11) pa je ova komponenta korišćena za razdvajanje.

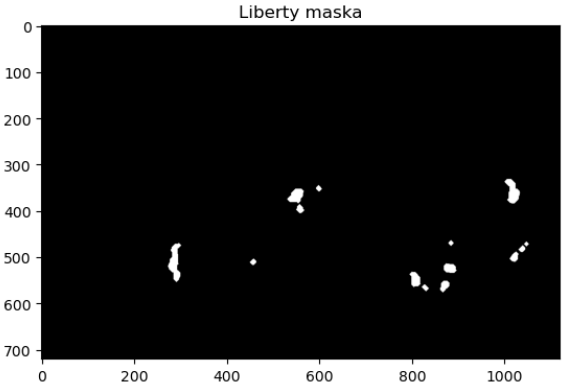


Slika 10: Komponenta jačine boje Slike 9

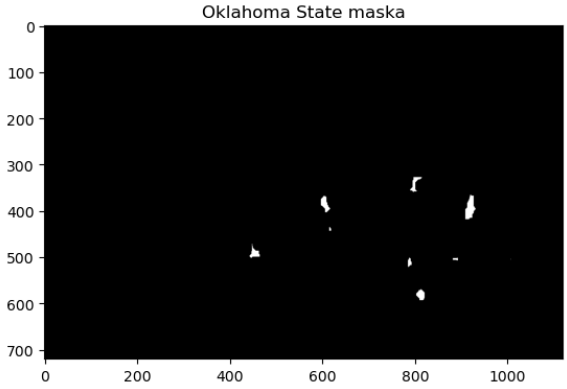


Slika 11: Histogram vrednosti jačine boje

Oba ogranka histograma su usrednjena i te vrednosti predstavljaju srednje vrednosti boja dresova za svaki tim. Dobijene vrednosti (u opsegu ± 10 za Liberty i $(-50, +10)$ za Oklahoma State[1]) su korišćene za binarizovanje slike. Morfološke operacije (zatvaranje i erozija, respektivno) su primenjene za uobličavanje kontura i uklanjanje šuma (Slike 12 i 13).



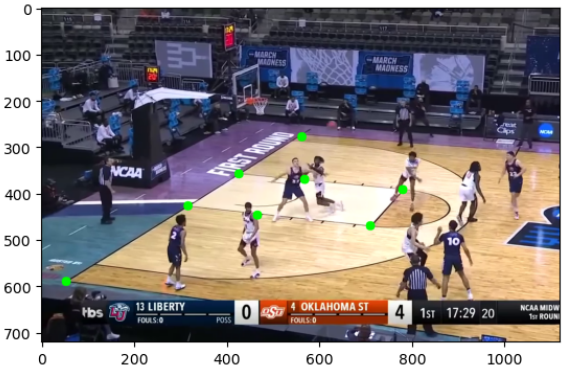
Slika 12: Maska za Liberty igrače



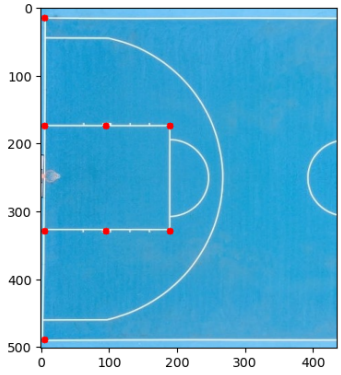
Slika 13: Maska za Oklahoma State igrače

2.4 Pronalaženje koordinata i mapiranje u 2D prostor

Na dobijenim maskama, u zavisnosti od sadržaja frejma, očekuje se oko 5 većih delova koji predstavljaju igrače. Koristeći algoritam povezanih komponenti (*connected component algorithm*) implementiran u OpenCV pronađeno je 5 najvećih regija na obe maske; centar mase svake regije predstavlja koordinate igrača. Nakon korekcije y koordinate (pomeranje malo naniže, radi adekvatnijeg predstavljanja igrača), ove koordinate su preslikane na 2D mapu terena; matrica preslikavanja je dobijena homografijom impelentiranom u OpenCV. Ulazni podaci za računanje matrice su 8 tačaka na prvom frejmu snimka i tačke na 2D projekciji terena kojima odgovaraju (Slike 14 i 15). Množenjem svakog para koordinata (tj. položaja svakog igrača) matricom preslikavanja dobijeni su njihovi položaji na 2D projekciji.



Slika 14: Odabrane tačke na frejmu

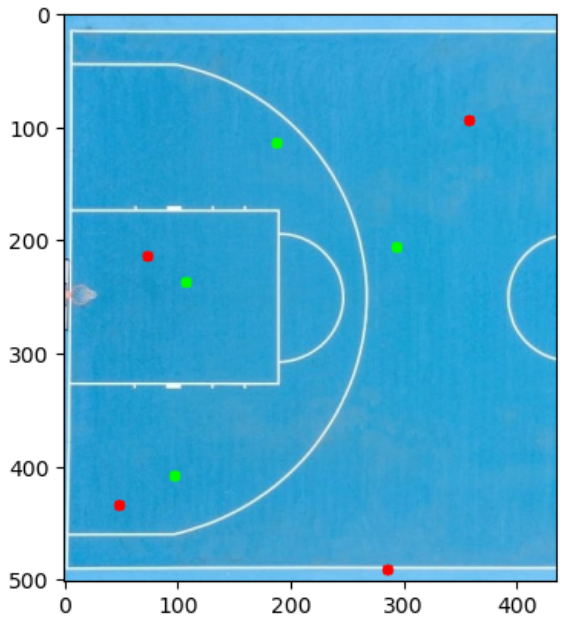


Slika 15: Odabrane tačke na projekciji

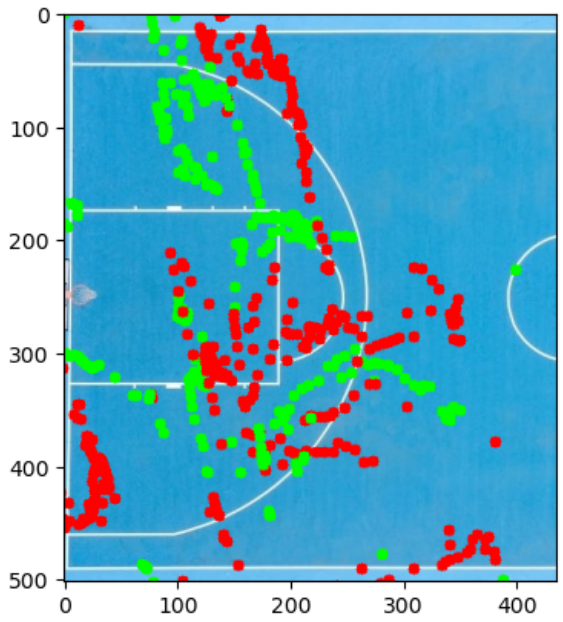
[1]Zbog poklapanja boje sa tekstom koji stoji odmah pored terena granica je znatno veća i deo teksta je manualno sklonjen

3 Rezultati

Na Slici 16 je prikazan rezultat algoritma na odabranom frejmu, Liberty igrači su označeni crvenom bojom, dok su Oklahoma State igrači označeni zelenom. Iz obe ekipe fali po jedan igrač; jedan je isečen prilikom izdvajanja terena (blizu gornje ivice), dok su ostali igrači su uspešno detektovani. Usled malih regija dobijenih segmentacijom koordinate igrača se razlikuju od pozicije na frejmu; razlika nije zanemarljiva ali boljom segmentacijom ona se može drastično smanjiti. Konačno, algoritam je primenjen na celom snimku; mapa kretanja svih igrača tokom snimka je prikazana na Slici 17.



Slika 16: Igrači mapirani u 2D



Slika 17: Mapa kretanja igrača

4 Moguće primene i unapređenja

Metode i principi korišćeni u ovom algoritmu mogu veoma jednostavno da se primene na bilo koji timski sport, bio to fudbal, rukomet, odbojka, američki fudbal, hokej ili bejzbol. Međutim, na osnovu rezultata ovako predstavljen princip nije dovoljno razrađen za generalnu upotrebu u košarci.

Za početak, detekcija samog terena ne bi smela da iseče igrače iz kadra kao što je to ovde bio slučaj. Moguće je izdvojiti teren osecanjem dela slike gde je publika, ali samo u slučajevima kada je kamera fiksna (inače ne bi moglo automatski da se iseče i da se garantuje da sam teren neće biti promenjen), što je praktično nikad; čak i u segmentima od po par sekundi, kakav je ovde analiziran, kamera se pomera i uvećava/umanjuje ekran. Još jedan problem koji potiče iz loše detekcije terena su smetnje pri detekciji boja; konkretno, natpis "First Round" bele boje koji se nalazi van terena je efektivno onemogućio uspešno prepoznavanje igrača Oklahoma State zbog poklapanja sa njihovom bojom dresova. S obzirom da je bela garnitura najčešći izbor domaćih ekipa a natpisi pored terena su česta stvar, posebno u profesionalnim ligama, ovo je problem koji se jako često pojavljuje.

Detektor korišćen za prepoznavanje igrača u velikoj većini slučajeva nije uspeo da prepozna sve igrače, takođe jako često je dolazilo do lažnih detekcija. To je i za očekivati, s obzirom da je SVM treniran na skupu pešaka, dok u sportskim događajima igrači jako retko miruju

(trčanje, saplitanje, skakanje - sve su to položaji koji se razlikuju od mirnog). Optimalno bi bilo da se SVM trenira upravo na snimcima košarkaških utakmica, tada bi preciznost detekcije verovatno bila znatno veća. Znatno preciznija detekcija igrača otvara novi način raspoznavanja po timovima. Uz poznate boje dresova pretraga bi mogla da se izvršava samo u regionima gde je detektovan igrač, koriseći logičke upite zasnovane na razlici boja dresova. U ovom slučaju to nije bilo moguće efikasno primeniti zbog loše preciznosti detektora; igrači koji nisu detektovani ne bi bili prepoznati uopšte, pa je pravljena maska po celom terenu. Detektor takođe nije umeo da se snađe kod situacija gde se igrači nalaze jedan pored drugog ili pored sudije, što je imalo kao posledicu jako male konture na konačnim maskama koje odgovaraju igračima. Za takve situacije potrebno je razviti poseban sistem, nešto slično kao [mich szcutye].

Identifikacija ključnih tačaka na terenu prilikom računanja homografije je nešto što treba da bude automatizovano. U tu svrhu se mogu koristiti razni već postojeći algoritmi (SIFT, SURF) koji sa sobom donose dalje usložnjavanje problema. Pomeranje i približavanje kamere su problemi sa kojima ovaj algoritam, trenutno napisan, ne ume da se izbori.

Generalno, problemi praćenja i detekcije igrača u košarci se danas u praksi rešavaju primenom tehnika mašinskog učenja [6] [7] [8] [9] [10]. Pristupi slični predstavljenom imaju previše mana da bi mogli uspešno da se koriste u komercijalne svrhe, njihova upotreba je ograničena na nekoliko sekundi igre, i to poželjno što jednostavnije (što više mirovanja, bez preklapanja, padanja na pod itd...). Svakako, ova tema predstavlja zanimljivu i prikladnu temu za studenski projekat, što se posebno vidi na američkim univerzitetima [11] [12] [13], i što sada autor može da potvrdi iz prve ruke.

Dodatak 1: funkcija za izdvajanje terena

```
def court_extraction(frame):

    img = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    H = img[:, :, 0]

    N, bin_edges = np.histogram(H.ravel(), bins=180, range=[0, 180])
    ind = np.argmax(N)
    thresh = bin_edges[ind]

    val1 = thresh - 7
    val2 = thresh + 7

    condition = (H > val1) & (H < val2)
    mask = np.where(condition, 1, 0)
    #scoreboard mask
    mask[620:690, 80:1120] = 0

    struct = ndi.generate_binary_structure(2, 2)
    mask = ndi.binary_opening(mask, structure = struct, iterations=10)

    struct = ndi.generate_binary_structure(2, 2)
    mask = ndi.binary_closing(mask, structure = struct, iterations=80)

    struct = np.zeros((3,3))
    struct[0, :] = 1
    mask = ndi.binary_dilation(mask, structure = struct, iterations=20)

    mask_3d = mask3(mask)
    show = (frame * mask_3d).astype(np.uint8)
    #scoreboard mask
    show[620:685, 80:1200, :] = 0

    return show
```

Dodatak 2: funkcija za prepoznavanje igrača

```
def player_detection(show):

    win_size = (48, 96)
    block_size = (16,16)
    block_stride = (8,8)
    cell_size = (8,8)
    num_bins = 9
    hog = cv2.HOGDescriptor(win_size, block_size, block_stride,
                            cell_size, num_bins)
    hog.setSVMDetector(cv2.HOGDescriptor_getDaimlerPeopleDetector())


    (rects, weights) = hog.detectMultiScale(show, winStride=(4, 4),
                    padding=(8, 8), scale=1.01)

    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    pick = non_max_suppression_fast(rects, overlapThresh=0.3)

    del_box = []
    j = 0
    for(xA, yA, xB, yB) in pick:
        if yB - yA > 160:
            del_box.append(j)
        j = j + 1
    pick_red = np.delete(pick, del_box, 0)

    return pick_red
```

Dodatak 3: funkcija za klasifikaciju po boji

```
def colour_detection(show, pick_red):

    test = cv2.cvtColor(show, cv2.COLOR_BGR2HSV)
    #court has low hue value
    cond_court = test[:, :, 0] > 40
    #removing NCAA logo colour, black and purple
    cond_black = test[:, :, 2] > 0
    cond_ncaa = (test[:, :, 1] < 160) & (test[:, :, 0] > 0)

    mask_arena1 = np.where(cond_court * cond_black * cond_ncaa, 1, 0)
    mask_arena = mask3(mask_arena1)

    show = (show * mask_arena).astype(np.uint8)
    img = cv2.cvtColor(show, cv2.COLOR_BGR2HSV)

    histV = np.zeros(np.max(np.shape(pick_red)),)
    c = 0
    for(xA, yA, xB, yB) in pick_red:
        roi = img[yA:yB, xA:xB, :]
        V = roi[:, :, 2]
        N, bin_edges = np.histogram(V.ravel(), bins=256)
        peakV = np.argmax(N[1:])
        histV[c] = peakV
        c = c + 1

    lib_c = np.mean(histV[histV < 100])
    ok_c = np.mean(histV[histV > 200])

    cond_lib = (img[:,:, 2] > lib_c - 7) & (img[:,:, 2] < lib_c + 7)
    cond_ok = (img[:,:, 2] > ok_c - 50) & (img[:,:, 2] < ok_c + 7)

    mask_lib = np.where(cond_lib, mask_arena1, 0)
    mask_ok = np.where(cond_ok, mask_arena1, 0)

    struct = ndi.generate_binary_structure(2, 2)
    mask = ndi.binary_closing(mask_lib, structure = struct, iterations=5)
    struct = ndi.generate_binary_structure(2, 1)
    mask = ndi.binary_opening(mask, structure = struct, iterations=5)
    LIB = players(mask, 4)

    struct = ndi.generate_binary_structure(2, 1)
    mask = ndi.binary_opening(mask_ok, structure = struct, iterations=3)
    struct = ndi.generate_binary_structure(2, 2)
    mask = ndi.binary_closing(mask, structure = struct, iterations=10)
    OK = players(mask, 4)

    return LIB, OK
```

Dodatak 4: funkcija za mapiranje igrača

```
def map_players(H, LIB, OK):

    # H, _= cv2.findHomography(data1, data2, cv2.RANSAC, 5.0)

    liblib = np.zeros((3,4))
    c = 0
    for (x,y,k) in LIB:
        vals = np.array([[x],[y],[k]])
        arr = np.matmul(H, vals)
        liblib[:, c] = arr.ravel() / arr[2]
        c = c + 1
    lib_lib = np.transpose(liblib[0:2, :])

    okok = np.zeros((3,5))
    c = 0
    for (x,y,k) in OK:
        vals = np.array([[x],[y],[k]])
        arr = np.matmul(H, vals)
        okok[:, c] = arr.ravel() / arr[2]
        c = c + 1

    ok_ok = np.transpose(okok[0:2, :])

    return lib_lib, ok_ok
```

Dodatak 4: *main* funkcija

```
import numpy as np
import matplotlib.pyplot as plt

import cv2
import scipy.ndimage as ndi

plt.close('all')
cv2.destroyAllWindows()


cap = cv2.VideoCapture('v98.mp4')

teren = cv2.imread('teren.jpg')
data1 = np.array([[427,358],[317,427],[569,370],[467,447],\
                  [781,392],[711,470], [563,278],[54,590]])

data2 = np.array([[6,175],[6,330],[96,175],[96,330],\
                  [190,175],[190,330],[6,16],[6,490]])
H, _= cv2.findHomography(data1, data2, cv2.RANSAC, 5.0)

im_vid = []
ret, frame = cap.read()
while ret:
    frame = frame[:, 0:1120, :]

    show = court_extraction(frame)
    pick_red = player_detection(show)
    LIB, OK = colour_detection(show, pick_red)
    lib, ok = map_players(H, LIB, OK)

    for (x,y) in lib:
        cv2.circle(teren, (int(x),int(y)), 5, (255,0,0), -1)
    for (x,y) in ok:
        cv2.circle(teren, (int(x),int(y)), 5, (0,255,0), -1)

    im_vid.append(teren)
    ret, frame = cap.read()

(y,x,_) = im_vid[0].shape
vid = cv2.VideoWriter('vid.avi', cv2.VideoWriter_fourcc(*'DIVX'), 15, (x,y))

for i in range(len(im_vid)):
    vid.write(im_vid[i])

vid.release()
cap.release()
```

Literatura

[1] Li Guangjing and Cuiping Zhang. "Automatic Detection Technology of Sports Athletes Based on Image Recognition Technology". In: *J Image Video Proc.* 15 (2019). DOI: https://doi.org/10.1186/s13640-019-0415-x.

[2] March Madness. *Oklahoma State vs. Liberty - First Round NCAA tournament extended highlights.* URL: https://www.youtube.com/watch?v=a5cldup-Qek. (accessed: 25.08.2021).

[3] et al. Dollar Piotr. "Pedestrian detection: An evaluation of the state of the art". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions* (2012), pp. 743–761.

[4] Adrian Rosebrock. *Pedestrian Detection OpenCV.* URL: www.pyimagesearch.com/2015/11/09/pedestriandetection-opencv/. (accessed: 25.08.2021).

[5] Tomasz Malisiewicz. *Blazing fast non-maximum suppresion.* URL: https://gist.github.com/quantombone/1144423. (accessed: 25.08.2021).

[6] Jiangrong Shen et al. "Recognizing Scoring in Basketball Game from AER Sequence by Spiking Neural Networks". In: July 2020. DOI: 10.1109/IJCNN48605.2020.9207568.

[7] Tsung-Yu Tsai et al. "Recognizing offensive tactics in broadcast basketball videos via key player detection". In: *2017 IEEE International Conference on Image Processing (ICIP).* 2017, pp. 880–884. DOI: 10.1109/ICIP.2017.8296407.

[8] Wei-Lwun Lu et al. "Learning to Track and Identify Players from Broadcast Sports Videos". In: *IEEE transactions on pattern analysis and machine intelligence* 35 (July 2013), pp. 1704–16. DOI: 10.1109/TPAMI.2012.242.

[9] Sara Battelini. "Computer vision for detecting and tracking players in basketball videos". MA thesis. Politecnico Di Torino, 2020.

[10] Chen Huang et al. "Recognizing tactic patterns in broadcast basketball video using player trajectory". In: *Journal of Visual Communication and Image Representation* 23 (Aug. 2012), pp. 932–947. DOI: 10.1016/j.jvcir.2012.06.003.

[11] Scott Parsons and Jason Rogers. "Basketball Player Tracking and Automated Analysis". EE368 final project. University of Stanford, 2014.

[12] Evan Cheshire, Cibele Halasz, and Jose Krause Perin. "Player Tracking and Analysis of Basketball Plays". EE368 final project. University of Stanford, 2015.

[13] Simon Xie, Clive Unger, and Kevan Patel. "Basketball player tracking". Semester project. University of Texas.