

SVUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1900

**Detekcija igrača u snimkama nogometnih
utakmica temeljena na dubokom učenju**

Ivan Matak

Zagreb, lipanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 8. ožujka 2019.

DIPLOMSKI ZADATAK br. 1900

Pristupnik: **Ivan Matak (0036487073)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Detekcija igrača u snimkama nogometnih utakmica temeljena na dubokom učenju**

Opis zadatka:

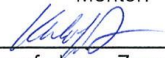
Analiza kretanja igrača tijekom utakmice pruža dragocjene informacije treneru, a može se olakšati i poboljšati primjenom tehnika računalnog vida. Moderni postupci računalnog vida temeljeni na dubokim modelima omogućavaju oblikovanje sustava za detekciju objekata učenjem s kraja na kraj.

U okviru diplomskog rada treba proučiti pristupe za detekciju objekata opisane u literaturi te ispitati prikladnost metoda dubokog učenja. Odabrati prikladan pristup te programski ostvariti sustav za detekciju igrača u video-snimkama nogometnih utakmica. Pripremiti skup uzoraka za učenje i testiranje oblikovanog sustava. Analizirati dobivene rezultate u pogledu točnosti detekcije igrača te računske zahtjevnosti.


Radu priložiti izvorni i izvršni kôd razvijenih postupaka, ispitne video-sekvence i rezultate, uz potrebna objašnjenja i dokumentaciju.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 28. lipnja 2019.

Mentor:


Izv. prof. dr. sc. Zoran Kalafatić

Djelovođa:


Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za
diplomski rad profila:


Doc. dr. sc. Marko Čupić

Želio bih se zahvaliti svome mentoru izv. prof. dr. sc. Zoranu Kalafatiću na korisnim i kvalitetnim savjetima, koji su uvelike olakšali izradu nekih radova, spremnosti na pomoć i uloženom vremenu tijekom sve tri godine mentorstva.

Sadržaj

1. Uvod.....	1
2. Detekcija i praćenje objekata	2
2.1. Detekcija objekata.....	2
2.1.1. Podatkovni zahtjevi.....	3
2.1.2. Općeniti radni okvir za sustav za detekciju objekata.....	3
2.2. Praćenje objekata	4
2.2.1. Reprezentacija objekata	4
2.2.2. Radni okvir za praćenje	5
3. Detekcija i praćenje objekata na nogometnim utakmicama.....	6
3.1. Definicija problema	6
3.2. Sustav za detekciju igrača	7
3.2.1. Detekcija na tri različite veličine mape značajki.....	8
3.2.2. Predikcija omeđujućeg pravokutnika	9
3.2.3. Funkcija gubitka	10
3.2.4. Višeklasna klasifikacija.....	11
3.3. Sustav za praćenje igrača.....	12
3.3.1. Definicija problema.....	12
3.3.2. Prikaz stanja sustava u trenutku t	12
3.3.3. Predikcija stanja	12
3.3.4. Ažuriranje stanja.....	14
3.3.5. Vjerojatnosni modeli	14
3.3.6. Podjela detekcija među trajektorijama.....	17
3.3.7. Rješavanje problema zakrivljanja igrača	18
3.3.8. Klasifikacija timova	18
3.3.9. Algoritam sustava za praćenje	19

4. Eksperimenti i rezultati	24
4.1. Odabir parametra k za algoritam kNN.....	24
4.2. Analiza sustava za praćenje	25
4.2.1. Praćenje dobro definiranih objekata	25
4.2.2. Zakrivanje.....	26
4.3. YOLOv3 detekcija.....	28
4.3.1. Treniranje YOLOv3 arhitekture	28
4.3.2. Provedba testova za YOLOv3 arhitekturu	29
4.4. Prosječna brzina sustava za detekciju i sustava za praćenje	37
5. Zaključak	38
6. Literatura	39

Tablica slika

Slika 1. Detekcija objekata [1]	3
Slika 2. Načini reprezentacije objekata koji se prate [5]	5
Slika 3. Prikaz okvira utakmice.....	6
Slika 4. Odnos veličine igrača naspram veličine videa	7
Slika 5. YOLOv3 arhitektura [6]	8
Slika 6. Predikcija omeđujućeg pravokutnika [9]	10
Slika 7. Računanje histograma za sliku	16
Slika 8. Označavanje igrača za učenje klasifikatora	19
Slika 9. Graf testiranja parametra k	25
Slika 10. Uspješno praćenje igrača	25
Slika 11. Neuspješno praćenje igrača	26
Slika 12. Uspješan oporavak od zakrivljanja	27
Slika 13. Neuspješan oporavak od zakrivljanja	27
Slika 14. Detekcija pomoću YOLOv3 prije treniranja na utakmici.....	29
Slika 15. Detekcija pomoću YOLOv3 nakon treniranja.....	29
Slika 16. Izračun IoU vrijednosti [13]	30
Slika 17. Izračun krivulje preciznosti i odziva i interpolirane krivulje vrijednosti i odziva [13]	35
Slika 18. Testiranje YOLOv3 tijekom treniranja	36

1. Uvod

Računalni vid je područje umjetne inteligencije koje za cilj ima razviti tehnike koje će računalima omogućiti da razumiju i interpretiraju digitalni sadržaj poput slika i videa. Međutim, iako je za ljude taj problem relativno jednostavan, jer ljudi bez većih napora mogu opisati što je na slici i što se u videu događa, za računalo je taj problem znatno teži [8]. Neki od razloga za to, a koji računalu predstavljaju dosta velik problem, su promjena osvjetljenja objekta, promjena orijentacije objekta (što često povlači i promjenu slike objekta), zakrivljanje jednog objekta drugim itd. Neki od problema kojima se bavi područje računalnog vida su detekcija objekata, praćenje objekata, segmentacija objekata, klasifikacija objekata itd., a naglasak u ovom radu je na detekciji i praćenju objekata. Dok je detekcija objekata u slici problem koji je dosta dobro riješen, pogotovo stupanjem dubokog učenja na scenu i korištenjem konvolucijskih mreža, praćenje objekata u slijedu slika je još uvijek zahtjevan i težak zadatak za računala prvenstveno zbog gore navedenih problema računalnog vida koji svi pogađaju praćenje objekata. Dodatno, još neki od razloga koji utječu na težinu razvoja sustava praćenja su nedostatak kvalitetnih skupova za učenje samih sustava, ali i izostanak kvalitetnih smjernica za izradu samog sustava kao kod detekcije. U radu se rješava specifičan problem detekcije i praćenja igrača na nogometnoj utakmici i dok se za detekciju koristi moderan pristup temeljen na dubokom učenju i detekcija se obavlja pomoću YOLOv3 arhitekture, za praćenje se koristi heuristički pristup te se pomoću preddefiniranih pravila pokušava otkriti identitet igrača kako bi se povezao s postojećim trajektorijama. Skup podataka korišten za rješavanje navedenog problema je video utakmice sniman nepomičnom kamerom između GNK Dinamo Zagreba i RNK Splita.

2. Detekcija i praćenje objekata

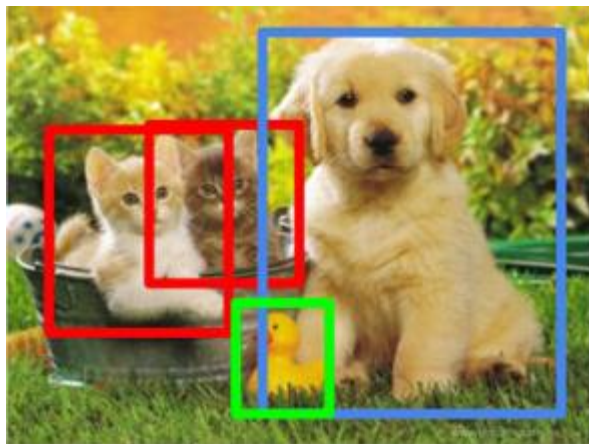
Ljudi bez većih problema i bez većih napora mogu detektirati i pratiti objekte u slici ili slijedu slika. To je moguće jer je ljudski vizualni sustav brz i precizan te može vrlo lako odrađivati kompleksne zadatke poput pronalaska objekata na slici ili praćenja istih u slijedu slika. Međutim, iako je nekad takav tip zadatka za računala bio prezahtjevan, danas uz prisutnost sve moćnijih grafičkih jedinica (eng. Graphics Processing Unit (*GPU*)), sve boljih algoritama kao i sve veće količine podataka moguće je napraviti kvalitetne sustave za praćenje i detekciju objekata koji postižu vrlo dobre rezultate [2].

2.1. Detekcija objekata

S nedavnim razvojem modela računalnog vida zasnovanih na dubokom učenju olakšana je primjena algoritama detekcije objekata. Osim znatnog poboljšanja performansi, modeli iskorištavaju i velike skupove podataka u svrhu bolje generalizacije kako bi se prilagodili novim situacijama u kojima je potrebna njihova primjena. Također današnji modeli su uglavnom tzv. modeli s kraja na kraj (cjeloviti modeli, eng. *end-to-end*) što znatno ubrzava vrijeme obrade podataka te u nekim slučajevima omogućuje obradu podataka u realnom vremenu [1].

Cilj same detekcije objekata je što kvalitetnija lokalizacija objekata na slici tako da ih označimo najčešće sa omeđujućim pravokutnikom (eng. *bounding box*) te odredimo klasu samog objekta. Pri određivanju klase najčešće se koriste specifične značajke za određenu klasu koje se nauče treniranjem samog modela detektora (npr. specifičnost kvadrata je da su mu stranice jednake dužine i da mu se stranice spajaju pod kutem od 90 stupnjeva. Također kod detekcije lica imamo specifične značajke kao što su oči, usta, nos, razmak između očiju...).

Danas je primjena detekcije objekata jako raširena i možemo je pronaći kod praćenja objekata, video nadzora, detekcije pješaka, detekcije lica, u samovozećim autima, algoritama za brojanje ljudi na nekom događaju itd.



MAČKA, PATKA, PAS

Slika 1. Detekcija objekata [1]

2.1.1. Podatkovni zahtjevi

Kako bi se istrenirao neki sustav za detekciju objekata potrebni su označeni podaci. U kontekstu detekcije objekata to su slike koje imaju označene objekte s omeđujućim pravokutnicima (eng. bounding box) i oznakom klase za svaki pojedini objekt na slici. Također za uspješno treniranje bitna je reprezentativnost podataka tj., da slike odgovaraju scenarijima u kojima će se sustav upotrebljavati. Npr., ako izgrađujemo sustav za detekciju prometnih znakova onda moramo uzeti u obzir različito osvjetljenje i vremenske uvjete te izgraditi sustav koji će biti jednako kvalitetan u svim mogućim različitim situacijama [1].

2.1.2. Općeniti radni okvir za sustav za detekciju objekata

Uobičajeno, najčešće postoje tri koraka u radnom okviru (eng. *framework*) za sustav za detekciju objekata [1]:

1. Algoritam ili model na temelju značajki, koje određuju različite klase objekata, određuje područja gdje bi se mogli nalaziti objekti
2. Algoritam pomoću značajki pronađenih područja određuje nalazi li se stvarno objekt na tom području i kojoj klasi pripada

3. Na kraju, omeđujući pravokutnici koji se preklapaju, se kombiniraju u jedan veći ili se oni koji su nepotrebni brišu (npr. ako jedan pravokutnik u potpunosti leži u drugom)

2.2. Praćenje objekata

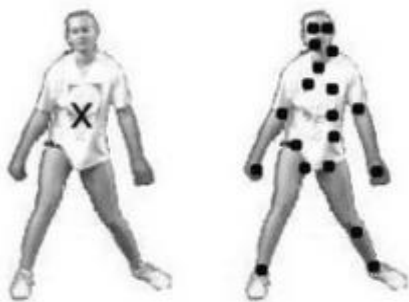
Praćenje objekata je dosta važno i dobro proučeno područje računalnog vida. Iako je praćenje objekata još uvijek kompleksan zadatak, razvojem i daljnjim istraživanjem i primjenom različitih modela znatno se poboljšala mogućnost praćenja [3]. Sam proces uključuje praćenje nekog objekta kroz slijed slika (najčešće u videu) usprkos brojnim problemima koji se mogu pojaviti tijekom praćenja (npr. zakrivljanje objekta, nestanak iz kadra na kratak period vremena, promjena osvjetljenja itd.).

Danas praćenje objekata ima široku primjenu te se upotrebljava u video nadzoru, navigaciji robota, nadzoru prometa itd.

2.2.1. Reprezentacija objekata

Ovisno o željenoj primjeni i cilju koji se želi postići praćenjem objekti se mogu prikazati na razne načine [5]:

1. **Pomoću točke/točaka** – objekt se može prikazati pomoću točke koja je centroid ili skupa točaka. Koristi se za praćenje objekata koji zauzimaju mali prostor na području slike
2. **Osnovni geometrijski oblici** – objekt se prikazuje pomoću pravokutnika, elipse itd. Koristi se za prikaz jednostavnih objekata
3. **Silueta ili kontura objekta** – prikazuje se granica objekta. Područje unutar kontura je silueta objekta. Koristi se za praćenje kompleksnih objekata
4. **Artikulirani oblici** – objekti se sastoje od dijelova tijela koji su spojeni zglobovima



(a) prikaz pomoću točke/točaka



(b) prikaz pomoću primitivnih oblika



(c) prikaz pomoću artikuliranih oblika



(d) prikaz pomoću siluete

Slika 2. Načini reprezentacije objekata koji se prate [5]

2.2.2. Radni okvir za praćenje

Da bi se praćenje provelo i kvalitetno dodijelio identifikator svakom objektu u slici potrebno je provesti nekoliko koraka:

1. Detekcija objekata – potrebno je obaviti kvalitetnu detekciju objekata pomoću modela kako bi se utvrdila područja od interesa, tj. objekti koji se prate
2. Korištenje algoritma praćenja kako bi se odredila sličnost detektiranih objekata u trenutnoj slici s objektima u prethodnoj (ili n prethodnih slika)
3. Dodjela identifikatora na temelju prethodno određene sličnosti objekata s prethodnim slikama

Postupak se ponavlja iterativno sve dok ima novih slika (okvira) u videu.

3. Detekcija i praćenje objekata na nogometnim utakmicama

3.1. Definicija problema

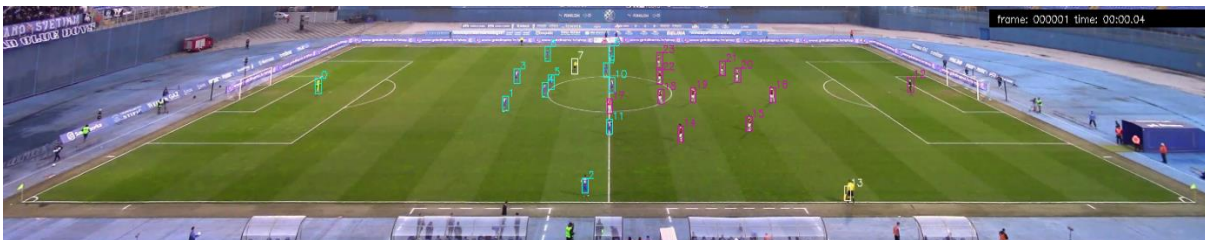
Problem koji se rješava u radu je detekcija igrača na nogometnim utakmicama korištenjem YOLOv3 arhitekture te praćenje detektiranih igrača korištenjem heuristike. Detekcija i praćenje u radu se izvodi nad utakmicom GNK Dinamo – RNK Split koja je snimana nepomičnom kamerom. Video traje 5 minuta, razlučivosti je 3260x570 te je brzine 25 *FPS*.



Slika 3. Prikaz okvira utakmice

Tijekom izvođenja videa javljaju se problemi koji otežavaju detekciju i praćenje samih objekata što ponekad dovodi algoritme u zabunu, tj. dolazi do grešaka tijekom detekcije i praćenja.

Prvi i najveći problem je veličina slike igrača u odnosu na kompletan okvir videa [12]. Igrači su jako mali te dolazi do problema detekcije tako malih objekata u okviru videa što ponekad uzrokuje gubitak detekcije nekih igrača. Uz to, veličina igrača varira ovisno o udaljenosti od kamere što ponekad dovodi do stapanja igrača s pozadinom što također dovodi do gubitka detekcije, a posljedično tome i do problema u praćenju.



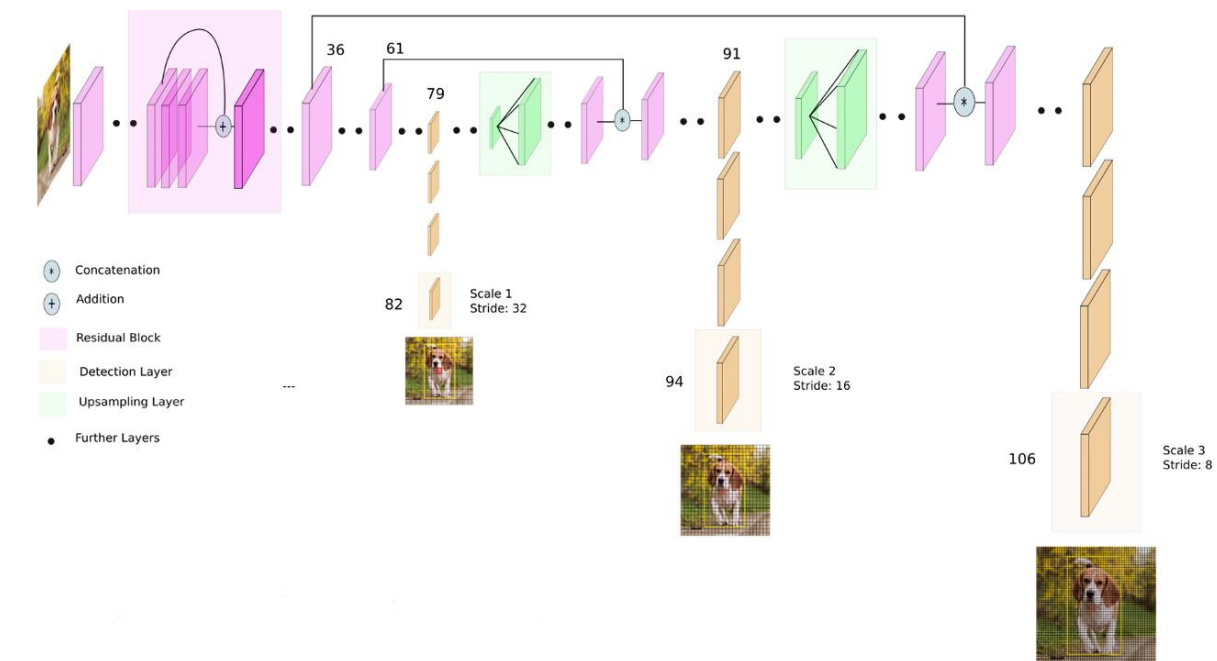
Slika 4. Odnos veličine igrača naspram veličine videa

Sljedeći problem koji otežava praćenje je brza promjena smjera kretanja, ali i same brzine kretanja igrača što znatno otežava predikciju položaja igrača na utakmici, a time i dodjelu identifikatora igraču.

Još jedan problem koji se javlja tijekom praćenja i detekcije je zakrivljanje igrača što dovodi do gubitka informacije o igraču, pa ga sustav ne može detektirati jer ga ne vidi. Taj problem je dosta zahtjevan jer ako igrači trče za ničijom loptom na duže relacije kroz više desetaka okvira videa se može dogoditi da se pojedini igrač uopće ne vidi što u potpunosti onemogućuje praćenje igrača.

3.2. Sustav za detekciju igrača

Za detekciju igrača koristi se YOLOv3 arhitektura. To je 106-slojna konvolucijska mreža te upravo zbog svoje veličine predstavlja usporavajući faktor same sustava za detekciju. Jedan od značajnijih elemenata YOLOv3 arhitekture, koji pomaže u detekciji objekata na slikama, su rezidualne mreže i detekcija na više različitih skala [6].



Slika 5. YOLOv3 arhitektura [6]

3.2.1. Detekcija na tri različite veličine mape značajki

U YOLOv3 detekcija se odvija na tri različite veličine mape značajki korištenjem 1×1 detekcijske jezgre na tri različita mjesta u arhitekturi [6]. Veličina mape značajki na kojima se odvija detekcija su za faktor 32, 16 i 8 manje od početne veličine slike (npr. ako je slika veličine 416×416 onda su mape veličine 13×13 , 26×26 i 52×52). Veličina detekcijske jezgre je $1 \times 1 \times (B \times (C \times 5))$. B je broj omeđujućih pravokutnika (eng. *bounding box*) za koje se radi predikcija na svakoj pojedinoj ćeliji na mapi značajki. "5" predstavlja 4 atributa omeđujućeg pravokutnika (t_x , t_y , t_w , t_h) i vjerojatnost da se u omeđujućem pravokutniku nalazi objekt. C predstavlja broj razreda koji se predviđaju. S obzirom na to da je YOLOv3 predtreniran na skupu podataka COCO, broj razreda je 80. Uz 3 generirana omeđujuća pravokutnika (eng. *bounding box*) na svakoj od različitih veličina mape značajki detekcijska jezgra je dimenzije $1 \times 1 \times 255$. Izlazna mape značajki po kojoj prođe detekcijska jezgra ima jednaku visinu i širinu kao ulazna mape značajki, a po dimenziji dubine ima detekcijske attribute.

Provedba detekcije na više razina omogućuje detekciju objekata različitih veličina. Tako je najmanja mapa značajki na kojoj se odvija detekcija zadužena za detekciju najvećih objekata, srednja za nešto manje objekte, a najveća za najmanje.

Broj omeđujućih pravokutnika (eng. *bounding box*) (N) koje na kraju prolaska kroz mrežu izgenerira YOLOv3 arhitektura jednak je:

$$N = 3 * (visina_najmanje_mape * širina_najmanje_mape) + \\ 3 * (visina_srednje_mape * širina_srednje_mape) + \\ 3 * (visina_najveće_mape * širina_najveće_mape)$$

3.2.2. Predikcija omeđujućeg pravokutnika

Za svaki omeđujući pravokutnik (eng. *bounding box*) se predviđa pet atributa, a to su t_x , t_y , t_w , t_h i t_o [9]. (t_x, t_y) su koordinate omeđujućeg pravokutnika koje se određuju relativno u odnosu na poziciju ćelije na mapi značajki. t_w i t_h su predikcija visine i širine omeđujućeg pravokutnika dok je t_o predikcija vjerojatnosti objekta u omeđujućem pravokutniku. Za pretvorbu relativnih vrijednosti u prave vrijednosti pravokutnika koje će se iscrtati na slici koriste se sljedeće formule:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

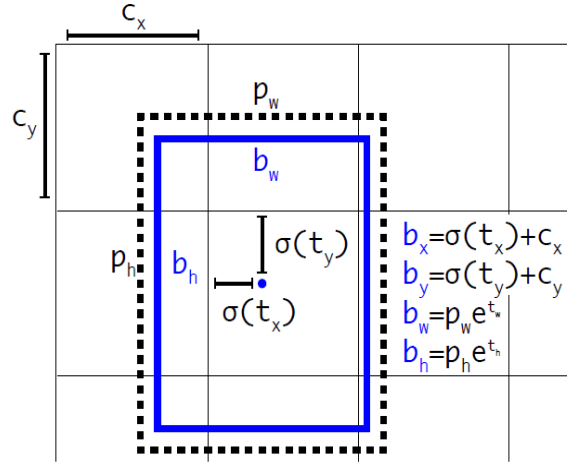
$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$Pr(object) * IOU = \sigma(t_o)$$

gdje su c_x i c_y koordinate gornjeg lijevog kuta ćelije, p_w i p_h visina i širina osnovnog omeđujućeg pravokutnika (eng. *anchor box*). σ predstavlja logističku (sigmoidalnu) funkciju koja djeluje nad t_x , t_y i t_o što ograničava vrijednosti na interval $[0,1]$ te

omogućuje stabilnije učenje sustava za detekciju dok se eksponencijalna funkcija u formuli za izračun širine (b_w) i visine (b_h) omeđujućeg pravokutnika koristi za skaliranje osnovnog pravokutnika na temelju vrijednosti izlaza predikcije za širinu (t_w) i visinu (t_h).



Slika 6. Predikcija omeđujućeg pravokutnika [9]. Plavo je označen pravokutnik koji je rezultat predikcije, a isprekidano je označen osnovni pravokutnik (eng. anchor box)

3.2.3. Funkcija gubitka

$$\begin{aligned}
 gubitak &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\
 &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
 &+ \sum_{i=0}^{S^2} \sum_{j=0}^B C_i \log_2 \hat{C}_i + \lambda_{noobj} (1 - C_i) \log_2 (1 - \hat{C}_i) \\
 &+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} p(c_i) \log_2 p(\hat{c}_i) + (1 - p(c_i)) \log_2 (1 - p(\hat{c}_i))
 \end{aligned}$$

YOLOv3 funkcija gubitka izvedena je na temelju opisa u [11], te uvodi neke novosti (odnosi se na (3) i na (4) član funkcije gubitka) u odnosu na funkciju gubitka u YOLOv1 i YOLOv2, a sastoji se od 4 dijela:

(1) – odnosi se na gubitak predikcije centra omeđujućeg pravokutnika (x,y) . Suma se provodi za sve pravokutnike $(j = 0...B)$ u ćeliji za svaku ćeliju $(i = 0...S^2)$. Ako se objekt nalazi u ćeliji i , a j -ti omeđujući pravokutnik je odgovoran za objekt (znači da centar od referentnog pravokutnika (eng. *ground truth*) upada u tu ćeliju) onda je 1_{ij}^{obj} jednak jedan, inače je jednak nula i nema gubitka [7]

(2) – odnosi se na gubitak predikcije širine i visine pravokutnika. S obzirom na to da bi malo odstupanje za velike objekte trebalo puno manje utjecati na gubitak nego malo odstupanje za male objekte onda se koristi korijen iz tih vrijednosti [7]

(3) – odnosi se na gubitak za predviđanje vjerojatnosti za prisutnost objekta u omeđujućem pravokutniku. C_i je jedan ako je objekt u pravokutniku ili nula ako nije, a \hat{C}_i je vrijednost predikcije da je objekt u pravokutniku te je jednaka $\sigma(t_o)$.

(4) – odnosi se na gubitak klasifikacije. Ako nema objekta u ćeliji onda je 1_i^{obj} jednako 0 i nema doprinosa funkcije gubitka za tu ćeliju.

Lambde u funkciji gubitka služe kao regularizacijski faktori i doprinose stabilnosti učenja samog sustava.

3.2.4. Višeklasna klasifikacija

YOLOv3 koristi višeklasnu klasifikaciju umjesto softmax funkcije jer za razliku od softmax funkcije, koja pretpostavlja nezavisnost različitih klasa, višeklasna klasifikacija uzima u obzir preklapanja između pojedinih klasa u detekciji (npr. ako u skupu podataka imamo razred pas i razred zlatni retriver). Umjesto softmax funkcije za svaku klasu se koristi logistička regresija i određeni prag (eng. *threshold*) za predikciju više klasa za detektirani objekt.

3.3. Sustav za praćenje igrača

3.3.1. Definicija problema

Nogometna utakmica može se predstaviti kao skup uzastopnih stanja i njihovih prijelaza. Stanje utakmice, u bilo kojem trenutku, se može opisati kao skup značajki koje opisuju pozicije igrača, njihov izgled, model pokreta itd. Cilj samog zadatka praćenja je odrediti stanje u trenutku t ako imamo saznanja o prijašnjim stanjima u utakmici. Cilj je efikasno procijeniti sljedeće stanje stohastičnog procesa u kojem se svaki igrač na utakmici prati odvojeno. Interakcije i dinamika same utakmice je obuhvaćena detektorom igrača (YOLOv3) koji daje predikcije samih igrača te se one koriste za napredovanje procesa određivanja novih pozicija igrača i sudjeluju u ažuriranju pojedinačnog sustava za praćenje svakog igrača. Za raspoređivanje pozicija među igračima u trenutku t zadužen je model koji računa izgled igrača (eng. appearance model) i model pokreta (eng. motion model).

3.3.2. Prikaz stanja sustava u trenutku t

Stanje sustava u trenutku t može se prikazati kao skup pojedinačnih stanja svakog od igrača $S_t = \{s_t^1, s_t^2, \dots, s_t^N\}$, gdje je N broj igrača (objekata) koji se prate. Svako stanje pojedinog igrača definirano je kao vektor [10]:

$$s_t^n = [p_t^n v_t^n b_t^n]$$

p_t^n predstavlja poziciju igrača, tj. (y, x) koordinatu njegove pozicije, v_t^n predstavlja komponentu brzine po y -osi i komponentu brzine po x -osi (v_y, v_x), a b_t^n je prikaz referentnog vektora koji opisuje izgled igrača kojeg se prati.

3.3.3. Predikcija stanja

Izuzmemo li vektor b^n iz stanja pojedinog igrača, za predikciju sljedeće pozicije igrača koristi se formula [10]:

$$p(s_t^n | s_{t-1}^n) = F s_{t-1}^n + \omega_t$$

gdje je F tranzicijska matrica koja omogućuje prijelaz u sljedeće stanje, a ω_t predstavlja šum koji predstavlja akceleraciju te je iz normalne razdiobe $N \sim (0, Q)$.

F simulira jednoliko pravocrtno gibanje prema formulama:

$$y_t^n = y_{t-1}^n + v_{y(t-1)}^n \Delta t - y \text{ koordinata}$$

$$x_t^n = x_{t-1}^n + v_{x(t-1)}^n \Delta t - x \text{ koordinata}$$

$$v_{y(t)}^n = v_{y(t-1)}^n - \text{komponenta y brzine}$$

$$v_{x(t)}^n = v_{x(t-1)}^n - \text{komponenta x brzine}$$

iz čega dobijemo:

$$s_t^n = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{t-1}^n \\ x_{t-1}^n \\ v_{y(t-1)}^n \\ v_{x(t-1)}^n \end{bmatrix}$$

$$s_t^n = F s_{t-1}^n$$

Uz jednoliko pravocrtno gibanje igrača, zbog stohastičnog gibanja igrača, dodajemo šum ω_t koji je iz normalne razdiobe $N \sim (0, Q)$ gdje je Q 4x4 matrica oblika [10]:

$$Q = \sigma_{acc} \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{3} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{3} \\ \frac{\Delta t^3}{3} & \frac{\Delta t^3}{3} & \Delta t^2 & 0 \\ \frac{\Delta t^3}{3} & \frac{\Delta t^3}{3} & 0 & \Delta t^2 \end{bmatrix}$$

gdje je σ_{acc} predstavlja standardnu devijaciju akceleracije i iznosi 0.5. Δt u matricama predstavlja vremenski interval između dva stanja i jednak je $1/FPS$ (u radu se koristi video kojemu je FPS 25).

3.3.4. Ažuriranje stanja

Za ažuriranje stanja potrebne su detekcije igrača u okviru videa. Svaka detekcija je predstavljena svojim pravokutnikom (eng. *bounding box*) te se na temelju tog pravokutnika izračuna njegov centar kao (x,y) koordinata. Također, uz (x,y) koordinatu računa se i model izgleda tog pravokutnika za usporedbu s postojećim trajektorijama do trenutka t te svaka detekcija ima svoju vjerojatnost da se igrač nalazi u tom pravokutniku (eng. *confidence score*).

U svakom vremenskom trenutku t dobivene detekcije se raspodjeljuju između trajektorija s^n na temelju umnoška vjerojatnosti modela izgleda i modela pokreta. Na kraju, konačna vrijednost nove pozicije igrača se dobije tako da uzmemo (x,y) koordinatu centra detekcije koja je dala najveću vrijednost umnoška obiju vjerojatnosti za danu trajektoriju s^n , a za vektor brzine (v_y, v_x) se uzme razlika između pozicije u prijašnjem okviru i novopridružene pozicije, tj. centra pridružene detekcije.

3.3.5. Vjerojatnosni modeli

U nogometnoj utakmici suigrači su uglavnom na međusobno većim udaljenostima dok su znatno bliže protivničkim igračima kako bi im pokušali oduzeti loptu i preuzeti posjed. Zbog toga, boja je dosta bitan indikator u određivanju identiteta igrača i njegovog razdvajanja od protivnika koji nosi drugačiji dres. Međutim, mana korištenja samo modela zasnovanog na izgledu (boji) može dovesti do preuzimanja identiteta među igračima iste ekipe ako se nalaze jedan blizu drugoga pa se zbog toga kao korektivna mjera s modelom zasnovanom na izgledu uvodi i model pokreta [10].

Vjerojatnost da je trajektorija s^n na detekciji d^m u trenutku t se evaluira zasebno za model zasnovan na izgledu i za model pokreta te se te dvije vjerojatnosti pomnože. Nakon toga detekcije se pridruže različitim trajektorijama ovisno o izračunatoj zajedničkoj vjerojatnosti modela, tj. tako da se detekcija pridruži onoj trajektoriji s kojom ima najveću združenu vjerojatnost izgleda i pokreta.

3.3.5.1. Model zasnovan na izgledu

Model zasnovan na izgledu (eng. appearance model) kreira se tako da svaku pojedinu detekciju d^m podijelimo po y -osi na gornji i donji dio slike (Slika 7.) i izračunamo HSV histogram za svaki od tih dijelova. Razlog takvog dijeljenja je što igrači najčešće imaju različitu boju majice od boje hlača što razdvajanjem slike na gornji i donji dio omogućuje kvalitetnije razlikovanje jednog tima od drugog (npr. ako oba tima imaju jako slične majice, a različite hlače prilikom usporedbe histograma koji nisu podijeljeni u dva dijela doprinos tih različitih hlača u razlikovanju igrača bio bi manji nego prilikom podjele slike igrača na gornji i donji dio). Svaki od histograma (za gornji i donji dio slike) se sastoji od spoja HS i V histograma s ukupno $C = C_h C_s + C_v$ pretinaca. U formulama $a[c]$ označava broj piksela u c -tom pretincu, dok c ide od 0 do $C-1$. Svaki histogram a je normaliziran tako da pikseli predstavljaju vjerojatnosnu razdiobu i da vrijedi $\sum_{c=0}^{C-1} a[c] = 1$. Na isti način se računa referentni histogram b^n koji se dobije tako da uzmemo detekciju igrača u prvom okviru videa, a ažurira se tako da ga mijenjamo s histogramom u svakom sljedećem okviru ako igrač nije zakriven s drugim igračem [10].

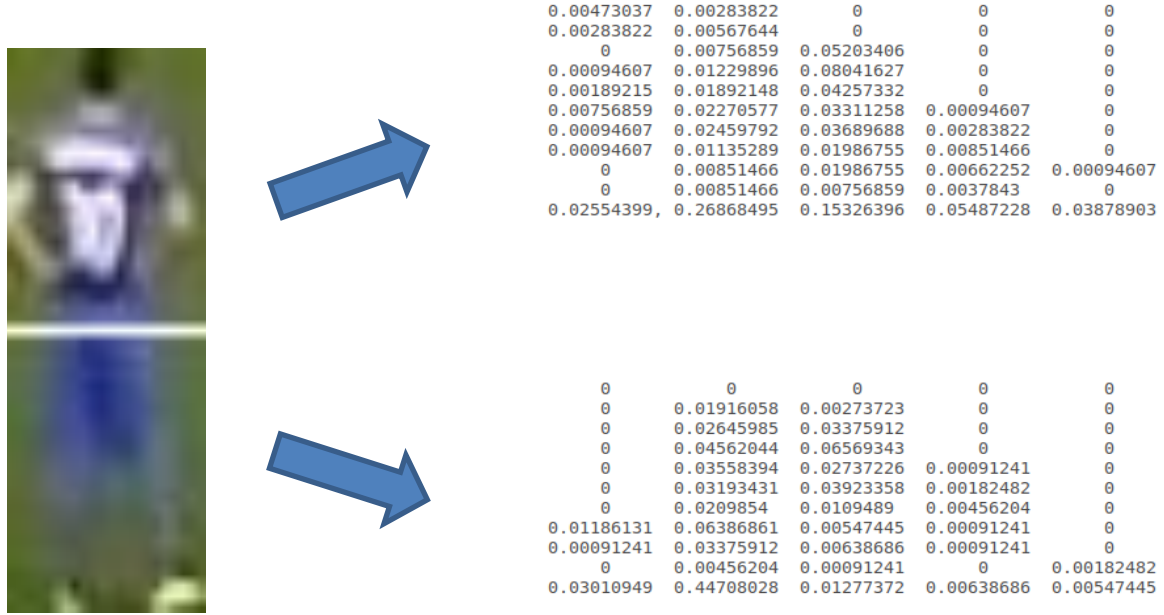
Da bi se izračunala vjerojatnost boje uspoređuju se referentni histogrami (histogram gornjeg dijela igrača i histogram donjeg dijela igrača) trajektorije s^n s histogramom detekcije (također gornjeg i donjeg dijela igrača) korištenjem Bhattacharyya koeficijenta sličnosti i to samo za detekcije koje se nalaze dovoljno blizu trajektoriji (to će detaljnije biti objašnjeno u sljedećem potpoglavlju). Udaljenost između dva histograma boje je definirana kao [10]:

$$d_{boja}(b^n, a^m) = (1 - \sum_{c=0}^{C-1} \sqrt{b^n[c]a^m[c]})$$

a vjerojatnost na temelju histograma je definirana kao:

$$p_{boja}(d^m | s^n) = \exp(-\lambda \frac{1}{J} \sum_{j=1}^J d_{boja}(b_j^n, a_j^m))$$

gdje je $J = 2$ zbog podjele slike na gornji i donji dio, shodno tome imamo za svaki od tih dijelova izračunat i histogram (slika 7.), a b_j^n i a_j^m predstavljaju redom dijelove referentnih histograma i histograma detekcije. Vrijednost C_h je postavljena na 10, a C_s i C_v na 5 i $\lambda=20$ [10].



Slika 7. Računanje histograma za sliku

3.3.5.2. Model pokreta

Model pokreta koristi se predikcijama pozicije pomoću formule $p(s_t^n | s_{t-1}^n) = F s_{t-1}^n + \omega_t$ na temelju pozicije iz prijašnjeg okvira. Na temelju predikcije pozicije računa se udaljenost između rezultata koje daje predikcija pozicije i centar svake od detekcija prema formuli [10]:

$$d_{pokreta}(p^n, d^m) = ||p^n - q^m||$$

gdje je p^n predikcija, a q^m pozicija centra detekcije d^m , tj. (x,y) koordinata centra detekcije. Vjerojatnost pokreta je obrnuto proporcionalna vrijednosti udaljenosti $d_{pokreta}$ te je veća što su predikcija i detekcija bliže, a manja što su detekcija i

predikcija pozicije dalje. Kako bi se na kraju odredila vjerojatnost pokreta koristi se funkcija φ [10]:

$$\varphi(d) = \frac{1}{\sigma_{pokreta}\sqrt{\pi}} \exp\left(-\frac{d^2}{\sigma_{pokreta}^2}\right)$$

$$p_{pokreta}(d^m|s^n) = \varphi(d_{pokreta}(p^n, d^m))$$

gdje je $\sigma_{pokreta}$ devijacija normalne distribucije koja određuje interval mogućih vrijednosti pokreta.

3.3.6. Podjela detekcija među trajektorijama

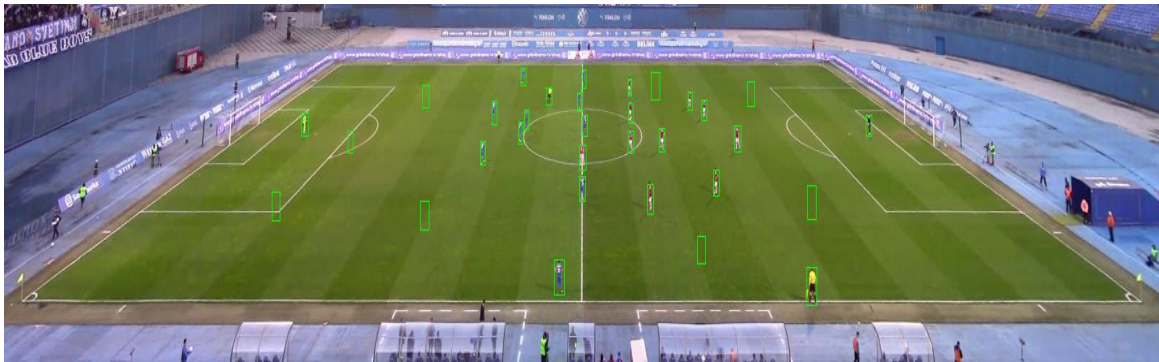
U svakom vremenskom trenutku t , pozicija trajektorije čija je predikcija od detekcije udaljena manje od r_{max} , tj. ako vrijedi $||p^n - q^m|| \leq r_{max}$ tada se detekcija pridružuje skupu detekcija koje su kandidati za dodjelu trajektoriji. r_{max} predstavlja radijus koji igrač može prijeći u trenutku Δt . Na kraju se trajektoriji pridružuje detekcija koja ima najveću vrijednost umnoška vjerojatnosti koje su dali model zasnovan na izgledu (eng. appearance model) i model pokreta. S obzirom na to da predikcija često zna biti neprecizna što za posljedicu ima veliku udaljenost predikcije od detekcije kojoj bi se trebala pridružiti trajektorija uvodi se nova varijabla $max_distance$ koja u većem radijusu od r_{max} pretražuje prostor te provjerava nalazi li se ijedna detekcija na udaljenosti manjoj od $max_distance$ od predikcije pozicije trajektorije te ako se nalazi onda se pridružuje skupu detekcija koje su kandidati za dodjelu trajektoriji te se kao i kod detekcija koje su udaljene manje od r_{max} , trajektoriji pridružuje ona detekcija koja ima najveći umnožak vjerojatnosti modela izgleda i modela pokreta.

3.3.7. Rješavanje problema zakrivanja igrača

Budući da će igrač koji je zakriven s drugim, najčešće protivničkim igračem, imati vrlo mali izlaz za umnožak vjerojatnosti modela zasnovanog na izgledu i modela pokreta gotovo je sigurno da će doći do gubitka detekcije, a time i trajektorije koja prati tog igrača. Kako bi se taj problem riješio, u slučaju da se niti jedna detekcija iz ekipe koja pripada igraču (za klasifikaciju timova se koristi algoritam *kNN* koji je objašnjen u idućem potpoglavlju) ne može pridružiti zadanoj trajektoriji, onda se trajektoriji pridružuje detekcija koja pripada nekom protivničkom igraču, ako ima manju udaljenost od r_{max} od trajektorije. U slučaju da se nijedna detekcija igrača iz protivničke ekipe ne nalazi u radijusu manjem od r_{max} onda se provjerava radijus $max_distance$, zbog već navedene nepreciznosti predikcije. Razlog zbog kojeg detekciju igrača pridružujemo detekciji koja pripada protivničkom igraču je pretpostavka da se igrač nalazi iza tog protivničkog igrača. Ta sumnja je opravdana zato što ako smo u prethodnom okviru imali detekciju tog igrača i uspješno smo ga pratili, a u trenutnom okviru je odjednom nestao, s obzirom na to da nije mogao prijeći veliku udaljenost u jednom vremenskom trenutku, vrlo vjerojatno je iza protivničkog igrača koji mu je u tom trenu bio blizu. Tako se omogućuje djelomično rješavanje problema zakrivanja.

3.3.8. Klasifikacija timova

Na nogometnoj utakmici prisutni su domaći, gostujući igrači i suci. S obzirom na specifičnost golmana u obje ekipe (imaju drukčije dresove od ostatka svoje momčadi), njih je potrebno klasificirati u zasebne kategorije. Stoga imamo kategorije: domaći i gostujući golman, domaći i gostujući igrači, suci i pozadina. Kategorije se dodjeljuju ručno na početku utakmice označavanjem igrača na terenu [10]. Za klasifikaciju se koristi *kNN* algoritam, a uči se na histogramima označenih igrača, sudaca i pozadine. Kako bi se prevladala promjena izgleda igrača zbog iznenadne promjene osvjetljenja ili poze, algoritam se uči svaku sekundu s novododanim histogramima (koji se dodaju u postojeći skup histograma tako da skup za učenje raste) igrača koji nisu prekriveni s niti jednim drugim igračem.



Slika 8. Označavanje igrača za učenje klasifikatora. Od korisnika se očekuje da prilikom pokretanja sustava za praćenje na prvom okviru videa ručno označi igrače oba tima, suče te pozadinu (travu na terenu) kako bi algoritam mogao ispravno razlikovati navedene sudionike na utakmici

3.3.9. Algoritam sustava za praćenje

U nastavku je prikazana iteracija algoritma za jedan okvir videa:

$f_1(s^n)$ – lista u koju se dodaju detekcije koje pripadaju istoj grupi igrača kao i trajektorija i udaljene su manje od r_{max} od predviđene pozicije trajektorije

$g_1(d^m)$ – lista u koju se dodaju trajektorije koje pripadaju istoj grupi i udaljene su manje od r_{max} od detekcije

$f_2(s^n)$ – lista u koju se dodaju detekcije koje pripadaju različitoj grupi igrača od trajektorije i udaljene su manje od r_{max} od predviđene pozicije trajektorije

$g_2(d^m)$ – lista u koju se dodaju trajektorije koje pripadaju različitoj grupi i udaljene su manje od r_{max} od detekcije

$g_3(d^m)$ – lista u koju se dodaju trajektorije koje pripadaju istoj grupi i udaljene su više od r_{max} od detekcije, a manje od $max_distance$

$g_4(d^m)$ – lista u koju se dodaju trajektorije koje pripadaju različitoj grupi i udaljene su više od r_{max} od detekcije, a manje od $max_distance$

$h_1(s^n)$ – rječnik u kojeg se dodaju udaljenosti detekcija od trajektorije koja pripada istoj grupi, a udaljena je više od r_{max} od detekcije, a manje od $max_distance$

$h_2(s^n)$ – rječnik u kojeg se dodaju udaljenosti detekcija od trajektorije koja pripada različitoj grupi, a udaljena je više od r_{max} od detekcije, a manje od $max_distance$

$i_1(s^n)$ – lista u koju se sprema umnožak vjerojatnosti pokreta i boje za istu grupu i udaljenost manju od r_{max}

$i_2(s^n)$ – lista u koju se sprema umnožak vjerojatnosti pokreta i boje za različitu grupu i udaljenost manju od r_{max}

$k(s^n)$ – lista u koju se zapisuju udaljenosti trajektorije od detekcija koje su udaljene više od r_{max} od detekcije, a manje od $max_distance$

$$\text{vektor_pokreta} - \begin{bmatrix} p_t \\ v_t \end{bmatrix} = \begin{bmatrix} y_t \\ x_t \\ v_{y(t)} \\ v_{x(t)} \end{bmatrix}$$

$$s^n = (\text{vektor_pokreta}, b^n)$$

* - predstavlja matrično množenje

Pseudokod:

učitaj detekcije za okvir videa:

za svaku putanju $s^n \in S_t$:

$$\text{vektor_pokreta}, b^n = s^n$$

$$p^n = F * \text{vektor_pokreta}$$

za svaku detekciju $d^m \in D$:

$$q^m, \text{histogram_det} = \text{izračunaj_centar_detekcije_i_histogram}(d^m)$$

grupa_det = kNN.odredi_klasu(histogram_det)

ako $\|p^n - q^m\| \leq r_max$ i (grupa_putanje == grupa_det):

$$i_1(s^n) \leftarrow p_{boja \times pokret}(d^m|s^n) = p_{boja}(d^m|s^n) * p_{pokret}(d^m|s^n)$$

$$f_1(s^n) \leftarrow d^m$$

$$g_1(d^m) \leftarrow s^n$$

continue

ako $\|p^n - q^m\| > r_max$ i $\|p^n - q^m\| \leq max_distance$ i (grupa_putanje == grupa_det):

$$g_3(d^m) \leftarrow s^n$$

$$h_1(d^m) \leftarrow \text{udaljenost}(q^m, p^n)$$

continue

ako $\|p^n - q^m\| \leq r_max$ i (grupa_putanje != grupa_det):

$$i_2(s^n) \leftarrow p_{boja \times pokret}(d^m|s^n) = p_{boja}(d^m|s^n) * p_{pokret}(d^m|s^n)$$

$$f_2(s^n) \leftarrow d^m$$

$$g_2(d^m) \leftarrow s^n$$

continue

ako $\|p^n - q^m\| > r_max$ i $\|p^n - q^m\| \leq max_distance$ i (grupa_putanje != grupa_det):

$$g_4(d^m) \leftarrow s^n$$

$$h_2(d^m) \leftarrow \text{udaljenost}(q^m, p^n)$$

continue

ako $\text{duljina}(f_1(s^n)) == 0$ i $\text{duljina}(f_2(s^n)) > 0$ i $\text{duljina}(h_1(d^m)) == 0$:

$$f_1(s^n) \leftarrow f_2(s^n)$$

$$g_1(s^n) \leftarrow g_2(s^n)$$

$$i_1(s^n) \leftarrow i_2(s^n)$$

ako $\text{duljina}(h_1(d^m))=0$:

$$k(s^n) \leftarrow h_2(d^m)$$

inače:

$$k(s^n) \leftarrow h_1(d^m)$$

za svaku detekciju $d^m \in D$:

za svaku putanju $s^n \in g_1(d^m)$:

$$p_{boja \ x \ pokret} = i_1(s^n)$$

ako $p_{boja \ x \ pokret}(d^m) < \max(p_{boja \ x \ pokret})$:

$$f_1(s^n) = f_1(s^n) - d^m$$

za svaki $s^n \in S_t$:

ako $\text{duljina}(f_1(s^n))=0$:

ako $\text{duljina}(h_1(s^n)) \neq 0$

za svaki ključ, vrijednost iz $h_1(s^n)$:

$$d^m = D[\text{ključ}]$$

izračunaj $p_{boja \times pokret}(d^m)$

u $f_1(s^n)$ dodaj onu detekciju koja ima maksimalnu vrijednost
 $p_{boja \times pokret}$

inače:

pretraži_određeni_broj_okolnih_pozicija

ako neka(ili više njih) od pozicija ima istu grupu kao i putanja:

u $f_1(s^n)$ dodaj onu detekciju koja ima maksimalnu vrijednost
 $p_{boja \times pokret}$

za svaki $s^n \in S_t$:

nova_pozicija = centar od $d^m \in f_1(s^n)$

novo_stanje_putanje = nova_pozicija

za svaku $d^m \in D$:

ako su sve liste s trajektorijama koje pripadaju detekciji d^m prazne:

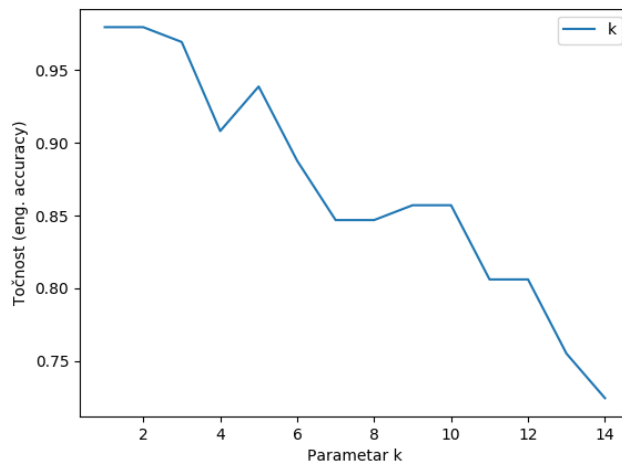
inicijaliziraj novu putanju

zapiši trajektorije u datoteku

4. Eksperimenti i rezultati

4.1. Odabir parametra k za algoritam kNN

Razlikovanje samih timova te sudaca i pozadine omogućuje kvalitetniju dodjelu identifikatora i bolje praćenje igrača. Kako bi to bilo moguće, koristi se algoritam kNN koji se uči na oznakama koje su ručno dodijeljene svakom objektu od interesa na terenu. Prije samog korištenja algoritma kNN koristi se unakrsna provjera za parametar k . Provjerava se vrijednost $k \in [1, 14]$ korištenjem unakrsne provjere “izostavi jednog” (eng. *Leave One Out*). Rezultati su prikazani na grafu ispod. Iako je algoritam tijekom unakrsne provjere dao najbolje rezultate za $k=1$, s obzirom na to da to predstavlja pohlepni pristup odabran je $k=3$ (također je $k=3$ tijekom izvođenja algoritma sustava za praćenje pokazao najbolje rezultate). Razlog tome je taj što postoji mogućnost da je najbliži susjed detekcije za koju kNN treba dati predikciju možda krivo klasificiran što će uzrokovati kontinuiranu krivu klasifikaciju objekta što može utjecati na samo praćenje. Početno, algoritam kNN se uči na skupu podataka označenom na prvom okviru videa “t7.mp4” što predstavlja oko 100 primjera za učenje (u njima su domaći tim, gostujući tim, golmani, suci i pozadina). Mali broj primjera za učenje algoritma kNN je jedan od nedostataka koji ponekad dovodi do krive klasifikacije tima kojem pripada igrač, posebno u početku izvođenja algoritma. Kasnije, kako se algoritam provodi na videu, nakon svake sekunde, dodaju se nove detekcije u postojeći skup primjera (skup raste) te se kNN uči ponovno kako bi se poboljšala sposobnost klasifikacije.

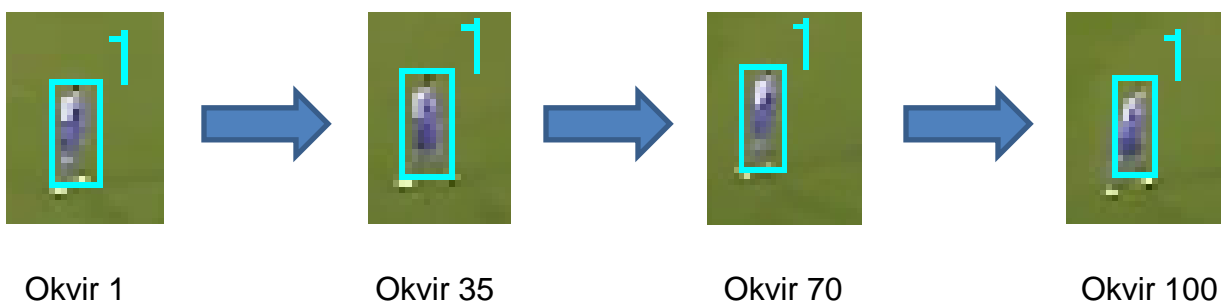


Slika 9. Graf testiranja parametra k

4.2. Analiza sustava za praćenje

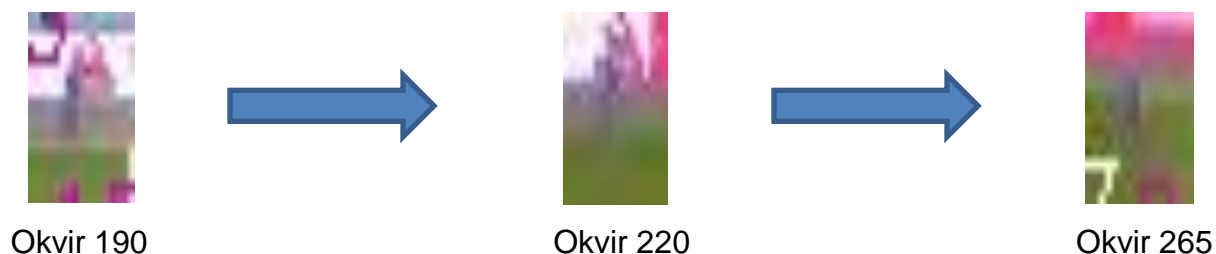
4.2.1. Praćenje dobro definiranih objekata

Sustav za praćenje objekata nema problema s praćenjem objekata koji su dobro definirani u svojoj okolini, tj. ako nema nikakvih prekrivanja. To je vidljivo na sljedećoj slici gdje se objekt prati 100 sličica i nema nikakvih problema jer je dobro definiran u okolini.



Slika 10. Uspješno praćenje igrača

S obzirom na to da se sustav za praćenje jako oslanja na sustav za detekciju ponekad se zna dogoditi da nekoliko (ponekad i nekoliko desetaka) uzastopnih okvira igrač uopće nije detektiran iako je dobro definiran u okolini. Razlog tome je što je premali te ga je s obzirom na veličinu slike, ali i moguće stapanje s pozadinom, detektoru jako teško uočiti.



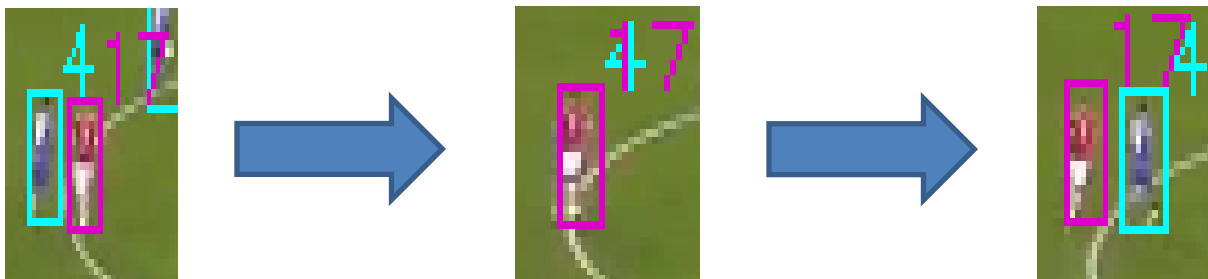
Slika 11. Neuspješno praćenje igrača

4.2.2. Zakrivljanje

Jedan od najvećih problema praćenja je pojava zakrivljanja među igračima. Tu dolazi do potpunog gubitka informacije o zakrivenom igraču jer se gubi sama detekcija tog igrača. Problem je utoliko teži jer ne znamo koliko će to zakrivljanje trajati, kada će se igrači razdvojiti i u kojem smjeru će se nastaviti kretati zakriveni igrač.

4.2.2.1. Uspješno riješeno zakrivljanje

Na sljedećoj slici je prikazan primjer uspješnog razrješavanja zakrivljanja gdje je iskorištena detekcija protivničkog igrača (crvenog) kako se ne bi izgubila putanja plavog igrača. Također olakotna okolnost je ta što je plavi igrač nastavio smjer kretanja u kojem je bio prije početka zakrivljanja pa je model pokreta relativno dobro predvidio buduću poziciju igrača.

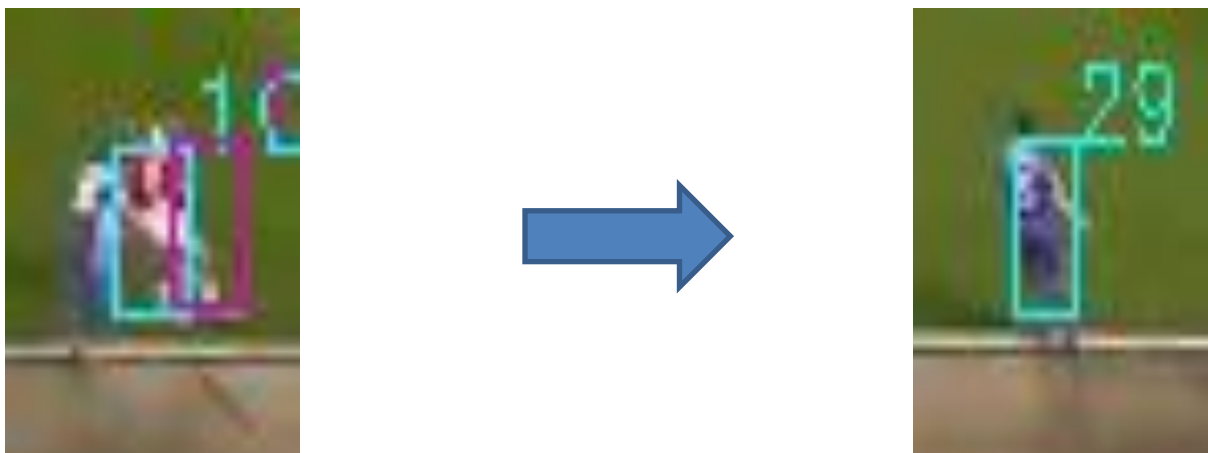


Slika 12. Uspješan oporavak od zakrivanja

Na Slici 12. u prvoj slici lijevo crveni igrač s identifikatorom 17 se približava plavom igraču s identifikatorom 4. U sljedećoj slici igrač s identifikatorom 4 je potpuno zakriven igračem s identifikatorom 17. U trećoj slici igrač više nije zakriven i problem zakrivanja je uspješno riješen bez gubitka identifikatora.

4.2.2.2. Neuspješno riješeno zakrivanje

Na sljedećoj slici je prikazano neuspješno razrješavanje zakrivanja te dodjela novog identifikatora. Jedan od razloga neuspješnog razrješavanja je nagla promjena smjera kretanja igrača koji je bio zakriven pa je model pokreta krivo procijenio poziciju u kojoj će igrač biti.



Slika 13. Neuspješan oporavak od zakrivanja

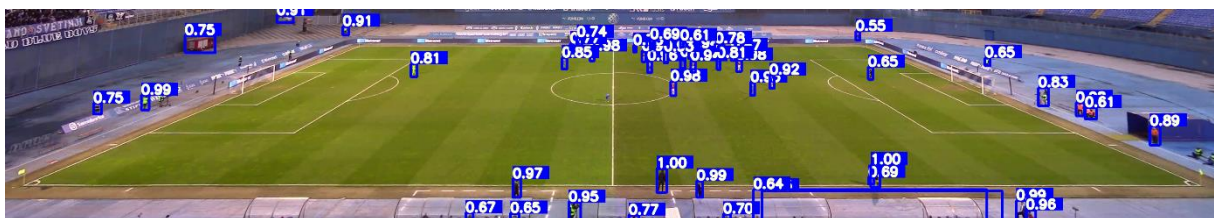
4.3. YOLOv3 detekcija

4.3.1. Treniranje YOLOv3 arhitekture

Kao sustav za detekciju koristi se već predtrenirani model YOLOv3 na skupu podataka COCO u kojem je prisutno 80 različitih razreda objekata. Budući da YOLOv3 arhitektura predtrenirana na skupu podataka COCO ne uspijeva napraviti dovoljno dobru detekciju igrača radilo se fino podešavanje (eng. *fine tuning*) cjelokupne mreže YOLOv3 arhitekture na početnim okvirima videa nogometne utakmice "t7.mp4". S obzirom na to da je rezolucija videa prevelika da bi se u sustav za treniranje YOLOv3 mogao slati okvir po okvir (zbog memorijskog ograničenja GPU), treniranje se odvija tako da svaki okvir podijelimo na 6 manjih dijelova. Svaki okvir se dijeli tako da uzmemo cijelu visinu slike, ali za širinu uzimamo intervale [0, 727], [527, 1254], [1054, 1781], [1581, 2308], [2108, 2835], [2635, 3260]. Tako dobijemo šest slika od kojih je svaka dimenzije 727x570 (osim zadnje koja je dimenzije 625x570), te svaka slika sa svojom prethodnom se preklapa u 200 piksela po x-osi. Kao referentne oznake igrača (eng. *ground truth*) koriste se izlazi iz jednog drugog sustava za detekciju i praćenje igrača u kojima je prisutan šum, tj. igrači nisu uvijek dobro označeni. Razlog tome je što sustav za detekciju i praćenje nije savršen što uzrokuje da se ponekad izgubi detekcija igrača. Kako bi se to kompenziralo angažiran je ljudski operater koji je spajao trajektorije igrača čija bi se detekcija izgubila pa nakon nekog vremena ponovno uspostavila. Za spajanje između posljednje lokacije na kojoj je igrač još uvijek detektiran (neposredno prije gubitka detekcije) i njegove novouspostavljene detekcije koristila se interpolacija. Kao posljedica takvog pristupa, u okvirima u kojima je detekcija igrača i njegova trajektorija bila izgubljena, ponekad dolazi do toga da je umjesto igrača označena pozadina, tj. zeleni travnjak. S obzirom na to da se YOLOv3 učio na tim oznakama igrača ponekad se dogodi i da YOLOv3 označi pozadinu kao igrača. Arhitektura je trenirana 45 epoha na prvih 16 sekundi videa (400 okvira), a skup podataka za treniranje je podijeljen na dijelove (eng. batch) od po 4 manja dijela jednog okvira. Na navedenim slikama su se provele nasumične transformacije kako bi se spriječila prenaučenosť same arhitekture na skup podataka za treniranje.

Nakon učenja same mreže, u fazi predikcije, moguće je obrađivati kompletan okvir videa (dimenzija 3260x570) jer nije potrebno propagirati unatrag gradijente za učenje mreže (kao za vrijeme treniranja) pa nema problema s memorijskim ograničenjem GPU-a.

Na sljedećim slikama prikazana je detekcija YOLOv3 prije i nakon učenja.



Slika 14. Detekcija pomoću YOLOv3 prije treniranja na utakmici

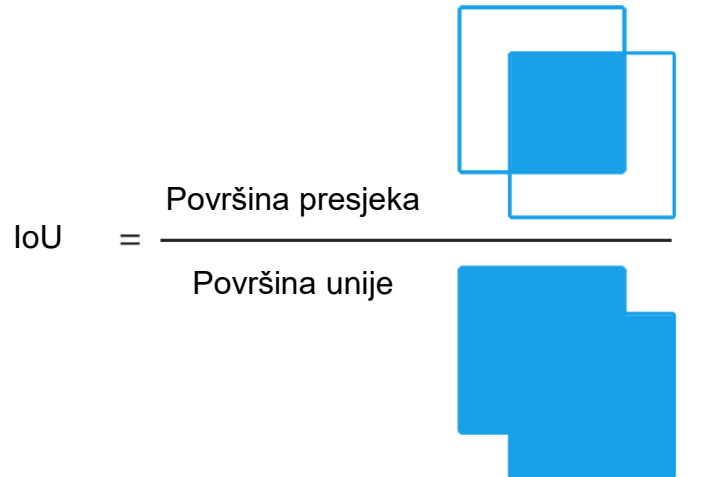


Slika 15. Detekcija pomoću YOLOv3 nakon treniranja

4.3.2. Provedba testova za YOLOv3 arhitekturu

4.3.2.1. *IoU* mjera, klasifikacija objekata i vjerojatnost da je objekt u omeđujućem pravokutniku

IoU (eng. *Intersection over Union*) je mjera koja računa omjer površine presjeka i površine unije referentne oznake (eng. *ground truth*) i predikcije koju je dao model za detekciju. Svrha *IoU* mjere je da nagradi one predikcije omeđujućih pravokutnika koje su dobro poravnate s referentnim oznakama.



Slika 16. Izračun *IoU* vrijednosti [13]

Kako bi se opisao izračun *IoU* mjere za korespondentni par referentne oznake i predikcije u YOLOv3 potrebno je objasniti izgled mape značajki nad kojom se radi predikcija kao i povezanost osnovnih pravokutnika (eng. *anchor box*), pravokutnika koji su rezultat predikcije i referentnih oznaka. Mapa značajki se može prikazati kao mreža s (x,y) koordinatama (npr. mreža 13x13). U svakoj točki te mreže nalaze se tri osnovna pravokutnika (eng. *anchor box*) (njihovi su indeksi 0,1,2), tako da se mreža s osnovnim pravokutnicima može prikazati s indeksom $[ind_b, x, y]$, gdje je *ind_b* indeks osnovnog pravokutnika u rasponu {0,1,2} i predstavlja jedan od tri različita osnovna pravokutnika, *x* je koordinata širine mreže, a *y* koordinata visine mreže na mapi značajki. Izlaz iz sustava za detekciju je mapa značajki s predikcijama (to su zapravo modificirani osnovni pravokutnici) te se i ona može prikazati kao i mreža s osnovnim pravokutnicima pomoću $[ind_b, x, y]$ jer mreža s osnovnim pravokutnicima je korespondentna s mrežom koja je izašla iz sustava za detekciju (npr. predikcija na lokaciji [2,1,1] je korespondentna i sličnog je oblika kao osnovni pravokutnik na lokaciji [2,1,1]). Referentne oznake se također nalaze u mreži mape značajki (npr. 13x13) samo što za razliku od mreže osnovnih pravokutnika i mreže predikcija nema svoje referentne oznake (pravokutnike) u svakoj od (x,y) koordinata mreže već samo tamo gdje se nalaze objekti u mapi značajki.

Kod predikcije nad nekom od tri mape značajki u YOLOv3 prvo se odabire jedan od tri osnovna pravokutnika koji oblikom najbolje odgovara pojedinoj referentnoj oznaci (indeksi osnovnih pravokutnika su $\{0,1,2\}$). Za to se koristi *IoU* mjera, ali ne koriste se stvarne koordinate na slici već se samo uspoređuju oblici referentne oznake i osnovnog pravokutnika. Nakon što su odabrani odgovarajući osnovni pravokutnici sortiramo ih po *IoU* vrijednosti tako da dobijemo indekse *IoU* mjera iz liste u padajućem nizu (od najveće vrijednosti *IoU* do najmanje, koristi se funkcija *argsort* iz Pythonove biblioteke Numpy). Iza toga mičemo one indekse osnovnih pravokutnika, referentnih oznaka i indekse onih referentnih oznaka u mreži koje imaju manju vrijednost ranije izračunate *IoU* vrijednosti od nekog preddefiniranog praga (u radu je to 0.1). Razlog tome je što u referentnim oznakama imamo objekte svih veličina (od malih do velikih), a YOLOv3 ovisno o veličini mape značajki detektira male, srednje ili velike objekte pa nema svrhe da s malim osnovnim pravokutnikom radimo predikciju za veliki objekt. Sada, budući da mreža osnovnih pravokutnika, ali i mreža u kojoj se nalaze referentne oznake odgovara mreži predikcije koristimo profiltrirane indekse osnovnih pravokutnika i mreže mape značajki da odaberemo detekcije u mreži koju je dao izlaz sustava za detekciju kako bi izračunali *IoU* mjere s referentnim oznakama. Budući da su referentne oznake filtrirane na isti način kao i indeksi osnovnih pravokutnika i mreže u kojoj se nalaze, onda prva referentna oznaka odgovara prvoj kombinaciji indeksa osnovnog pravokutnika i profiltriranih indeksa mreže, druga oznaka drugoj kombinaciji indeksa osnovnog pravokutnika i profiltriranih indeksa mreže itd. Tako dobijemo *IoU* referentne oznake i korespondentne predikcije. Također uparene referentne oznake i detekcije koristimo i za provjeru ispravnosti klasifikacije.

Kod klasifikacije objekta u omeđujućem pravokutniku se određuje odgovara li razred objekta u referentnoj oznaci razredu koji je dao izlaz iz sustava za predikciju. Također sustav za predikciju daje još i vjerojatnost da se objekt nalazi u omeđujućem pravokutniku što je također bitno kod određivanja hoće li objekt biti proglašen *TP* (ispravnom detekcijom, eng. *true positive*), *FP* (krivom detekcijom, eng. *false positive*) ili *FN* (propuštenom detekcijom, eng. *false negative*). Ispravne, krive i propuštene detekcije će biti detaljnije objašnjene u sljedećem potpoglavlju.

4.3.2.1. Objašnjenje pojmova *TP*, *FP* i *FN*

Klasifikacija, *IoU* i vjerojatnost je li objekt u omeđujućem pravokutniku utječu na to hoće li objekt u slici biti proglašen kao *TP*, *FP* ili *FN*. Ako je vrijednost *IoU* veća od nekog preddefiniranog praga (u radu je to 0.5) i klasa objekta odgovara stvarnoj klasi objekta u referentnoj oznaci i vjerojatnost objekta u omeđujućem pravokutniku je veća od preddefiniranog praga (u radu je to 0.5) onda je to *TP*. Ako je vrijednost *IoU* manja od preddefiniranog praga, vjerojatnost objekta u pravokutniku veća od praga, a predikcija klase odgovara stvarnoj vrijednosti klase u referentnoj oznaci onda je to *FP*. Također, detekciju označavamo kao *FP* ako je *IoU* veći od praga i vjerojatnost objekta u pravokutniku je veća od praga, ali je detekcija krivo klasificirana u odnosu na referentnu oznaku (npr. razred referentne oznake je pas, a sustav za detekciju je klasificirao objekt kao čovjeka). Zadnji slučaj je kada je vjerojatnost objekta u omeđujućem pravokutniku manja od preddefiniranog praga. Tada je to *FN* [13].

4.3.2.1. Preciznost

Preciznost (eng. *precision*) je dosta korištena metrika kojom se mjeri uspješnost nekog modela. Preciznost za detekciju neke klase objekata se može definirati kao broj ispravno detektiranih objekata te klase u odnosu na ukupan broj detektiranih objekata koji su klasificirani u tu klasu (to uključuje objekte koji su ispravno klasificirani u tu klasu (eng. *true positive(TP)*) i objekte koji su neispravno klasificirani u tu klasu (eng. *false positive(FP)*). Preciznost se može prikazati formulom:

$$preciznost = \frac{TP}{TP + FP}$$

Vjerojatnost da detektirani objekt koji je klasificiran u neku klasu pripada stvarno toj klasi je to veća što je vrijednost preciznosti te klase bliža jedan. Ako model koji radi detekciju radi s više klasa onda kao vrijednost preciznosti modela uzimamo srednju vrijednost izračunatih preciznosti za svaku klasu.

4.3.2.2. Odziv

Kao i preciznost, i odziv (eng. *recall*) se dosta često koristi u proračunu uspješnosti nekog modela. Odziv za neku klasu se može definirati kao broj ispravno detektiranih objekata neke klase (eng. *true positive*(*TP*)) u odnosu na zbroj ispravno detektiranih objekata te klase i broj propuštenih detekcija objekata za tu klasu (eng. *false negative*(*FN*)). Odziv se može prikazati formulom:

$$odziv = \frac{TP}{TP + FN}$$

Ako model koji radi detekciju radi s više klasa onda kao vrijednost odziva modela uzimamo srednju vrijednost izračunatih odziva za svaku klasu.

4.3.2.3. Srednja prosječna preciznost

Srednja prosječna preciznost (eng. *mean average precision* (*mAP*)) je mjera koja se koristi kod modela koji se koriste za detekciju objekata jer uspješno može odrediti kvalitetu modela uzimajući u obzir ispravnost klasifikacije (predikcija klase kao izlaz iz sustava za detekciju se uspoređuje s klasom referentne oznake), vjerojatnost da je objekt u omeđujućem pravokutniku (najčešće mora biti veća od nekog predefiniranog praga, isto izlaz iz sustava za detekciju) i kvalitetu same detekcije (za kvalitetu detekcije se koristi *IoU* mjera) [13].

Kada izračunamo *TP*, *FP* i *FN* na temelju predikcija koje je dao sustav za detekciju i njihove usporedbe s referentnim oznakama, možemo ih iskoristiti za proračun preciznosti i odziva koji će se kasnije koristiti za izračun krivulje preciznosti i odziva (eng. *PR curve*) koja se u zadnjem koraku koristi za proračun prosječne preciznosti (eng. *average precision*). Krivulja preciznosti i odziva je krivulja koja pokazuje kompromis između preciznosti i odziva, tj. kako rastom odziva pada vrijednost preciznosti, ali ne monotono (preciznost nemonotono pada sve dok u skupu detekcija ima ispravnih detekcija koje još nisu prihvaćene) pa je za proračun prosječne preciznosti potrebno interpolirati vrijednosti krivulje. Porast odziva i pad preciznosti je zapravo posljedica smanjivanja praga za prihvaćanje detekcija (to može biti

smanjivanje praga za vjerojatnost objekta u omeđujućem pravokutniku ili praga *IoU* mjere, ili oboje) što dovodi do prihvatanja puno više krivih detekcija (*FP*) u odnosu na ispravne detekcije (*TP*) što smanjuje preciznost, a odziv ostavlja na istoj vrijednosti ili ga povećava (odziv kao i preciznost raste ako raste broj ispravnih detekcija, ali na njega ne utječu krive detekcije pa on ne pada kad i preciznost). S obzirom na to da je za potrebe proračuna prosječne preciznosti potrebno interpolirati krivulju odziva i preciznosti uvodi se nova mjera, a to je interpolirana preciznost. Interpolirana preciznost je definirana formulom:

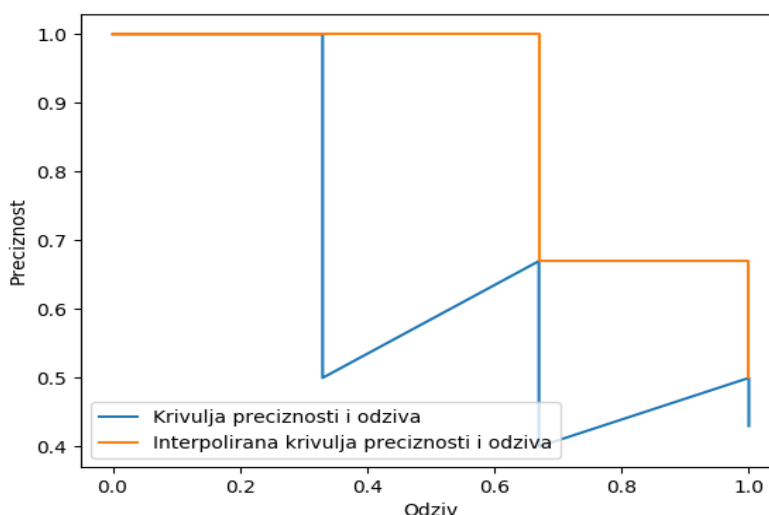
$$P_{interpolirana}(r) = \max_{\bar{r}, \bar{r} \geq r} P(\bar{r})$$

Vidljivo je da je to mjera koja za svaku vrijednost odziva \bar{r} koja prelazi ili je jednaka vrijednosti r uzima maksimalnu vrijednost preciznosti.

U Tablici 1. prikazan je primjer detekcije igrača. Imamo 7 detekcija od kojih su tri ispravne detekcije (*TP*), a 4 neispravne detekcije (*FP*). Primjeri su posloženi silazno po vjerojatnosti da se igrač nalazi u pravokutniku (eng. *confidence score*) tako da je ona detekcija s najvećom vjerojatnosti na vrhu tablice, a ona s najmanjom vjerojatnosti na dnu. Iz tablice možemo vidjeti kako kada je prag za prihvatanje detekcije dosta visok (npr. 0.85) imamo veliku preciznost, ali mali odziv zato što zbog visokog praga dosta ispravnih detekcija ne prihvaćamo. Smanjivanjem praga na 0.8 dolazi do smanjenja preciznosti zato što smo dopustili prihvatanje neispravnih detekcija. Odziv ostaje isti jer na njega ne utječu neispravne detekcije. Daljnjim smanjivanjem praga (sve do 0.45) preciznost se nemonotono mijenja (npr. kod praga 0.8 preciznost je 0.5, a kod praga 0.75 raste na 0.67 jer smo prihvatili ispravnu detekciju, daljnjim smanjivanjem praga opet pada na 0.5 itd.), dok odziv ostaje nepromijenjen ili raste. U zadnjem stupcu je prikazana interpolirana preciznost koja je izračunata prema gore navedenoj formuli.

	TP/FP	Vjerojatnost igrača (eng. confidence score)	Preciznost	Odziv	Interpolirana preciznost
1.	TP	0.90	$1/1 = 1$	$1/3 = 0.33$	1
2.	FP	0.82	$1/2 = 0.5$	$1/3 = 0.33$	1
3.	TP	0.76	$2/3 = 0.67$	$2/3 = 0.67$	0.67
4.	FP	0.71	$2/4 = 0.5$	$2/3 = 0.67$	0.67
5.	FP	0.64	$2/5 = 0.4$	$2/3 = 0.67$	0.67
6.	TP	0.58	$3/6 = 0.5$	$3/3 = 1$	0.5
7.	FP	0.49	$3/7 = 0.43$	$3/3 = 1$	0.5

Tablica 1. Prikaz izračuna preciznosti, odziva i interpolirane preciznosti za detekciju igrača na utakmici (primjer je ilustrativne prirode) [13]



Slika 17. Izračun krivulje preciznosti i odziva i interpolirane krivulje vrijednosti i odziva [13]

Prosječna preciznost se računa tako da se uzme vrijednost ispod interpolirane vrijednosti krivulje i odziva u jedanaest točaka odziva, a to su $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Formula za izračun prosječne preciznosti je:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\dots,1\}} P_{interpolirana}(r)$$

Za primjer iz tablice 2 ona iznosi:

$$AP = \frac{1}{11} (1 + 1 + 1 + 1 + 1 + 1 + 1 + 0.67 + 0.67 + 0.67 + 0.5) \approx 0.865$$

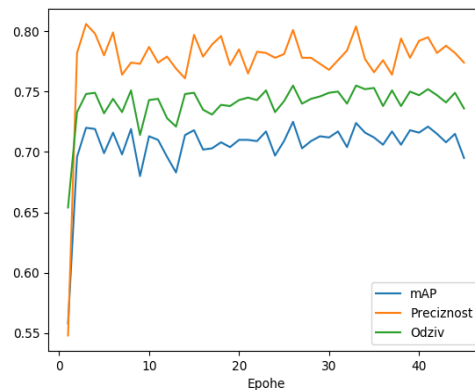
Navedena srednja preciznost izračunata je za samo jednu klasu. Ako imamo više klasa onda računamo prosječnu srednju preciznost (eng. *mean average precision*) tako da uzmemo srednju preciznost za svaku klasu i uzmemo prosjek prema formuli:

$$mAP = \frac{1}{broj\ klasa} \sum_{c \in klase\ objekata} AP(c)$$

Također uz oznaku mAP pridružuje se i oznaka @ k (npr. mAP@0.5), gdje k predstavlja *IoU* prag za određivanje je li detekcija *TP*, *FP* ili *FN*.

4.3.2.4 Rezultati testiranja

YOLOv3 je testiran na istom videu od 2000. do 2400. okvira koji su isto podijeljeni na šest manjih slika kao i skup za treniranje. Provedeno je testiranje YOLOv3 arhitekture za mAP@0.5, preciznost i odziv nakon svake epohe treniranja kako bi vidjeli napredovanje učenja arhitekture, a rezultati su prikazani na grafu ispod.



Slika 18. Testiranje YOLOv3 tijekom treniranja

4.4. Prosječna brzina sustava za detekciju i sustava za praćenje

S obzirom na to da je YOLOv3 implementiran korištenjem radnog okvira PyTorch pokušalo se maksimizirati korištenje grafičke kartice kako bi se skratilo vrijeme izvođenja. Test se izvodi na grafičkoj kartici Nvidia Tesla K80 memorijskog kapaciteta 11GB koju pruža usluga Google Colaboratory. Sustav za praćenje se ne izvodi paralelno stoga je on testiran na računalnom procesoru Intel i5-4210U. Oba sustava su testirana na prvih 1000 okvira videa i uzeto je prosječno vrijeme izvođenja po okviru.

Vrijeme izvođenja (s)	
Sustav za detekciju	0.3435
Sustav za praćenje	0.3275

Tablica 2. Prosječna brzina izvođenja za oba sustava

Možemo vidjeti da je sustav za detekciju nešto sporiji. Razlog tome je što je mreža koju koristi YOLOv3 arhitektura jako velika te unatoč paralelizaciji na grafičkoj kartici broj operacija nad slikom je još uvijek velik.

5. Zaključak

U radu je prikazana detekcija igrača na nogometnoj utakmici korištenjem YOLOv3 detektora te je na rezultatu detekcija proveden algoritam za praćenje igrača u videu. Vidljivo je da su detekcija, a prvenstveno praćenje dosta kompleksni problemi s brojnim otegotnim okolnostima koje otežavaju njihovu primjenu i pogoršavaju samu kvalitetu rezultata. U radu je dio tih problema riješen, dok su neki i dalje prisutni te im u budućem radu treba posvetiti više pažnje. To se prvenstveno odnosi na zakrivljanje igrača tijekom kojega zakriveni igrač značajno mijenja smjer kretanja što zbunjuje sustav za praćenje. Taj problem se djelomično rješava uvođenjem radijusa pretraživanja *max_distance* koji pretražuje veći prostor od *r_max*.

Sustav za detekciju za navedeni problem detekcije igrača radi jako kvalitetno osim situacija kad se igrač jako udalji od kamere i praktički stopi s pozadinom što sustavu za detekciju onemogućuje kvalitetno izvlačenje značajki igrača što završava s nedetektiranim igračem. S obzirom na to da se sustav za praćenje naslanja na sustav za detekciju, ako dođe do toga da sustav za detekciju ne uspije detektirati nekog igrača vrlo često dolazi do gubitka tog igrača, a samim time i njegove trajektorije, tj. sustav za praćenje zakaže u slučaju nedetektiranog igrača. Navedeni problem bi bio umanjen uz bolju kvalitetu videa što bi detektoru omogućilo mnogo kvalitetnije izvlačenje značajki i raspoznavanje igrača.

Još jedan od problema navedenih sustava je taj što nisu dovoljno brzi za izvođenje u realnom vremenu. Problem brzine bi se mogao riješiti daljnjim pokušajem optimizacije koda, uvođenjem višedretvenosti i korištenjem snažnije GPU.

6. Literatura

1. Lars Hulstaert, A Beginner's Guide to Object Detection, 19.4.2019.,
<https://www.datacamp.com/community/tutorials/object-detection-guide>,
12.6.2019.
2. Introduction to Object Detection, 7.8.2019.,
<https://www.hackerearth.com/blog/developers/introduction-to-object-detection/>,
12.6.2019.
3. Agren, S., Object tracking methods and their areas of application: A meta-analysis, magistrski rad, Umea Universitet, 2017.
4. Priya Dwivedi, People Tracking using Deep Learning, 7.2.2019.,
<https://towardsdatascience.com/people-tracking-using-deep-learning-5c90d43774be>, 13.6.2019.
5. Sri Vidhya K., Object tracking, 23.9.2013.,
<https://www.slideshare.net/srividhya3950/object-tracking-26460611>, 13.6.2019.
6. Ayoosh Kathuria, What's new in YOLO v3?, 23.4.2018.,
<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>,
15.6.2019.
7. Nikolas Laskaris, New State-of-the-art in Logo Detection Using YOLOv3 and Darknet, 26.3.2019., <https://platform.ai/blog/page/7/new-state-of-the-art-in-logo-detection-using-yolov3-and-darknet/>, 15.6.2019.
8. Jason Brownlee, A Gentle Introduction to Computer Vision, 19.3.2019.,
<https://machinelearningmastery.com/what-is-computer-vision/>, 19.6.2019.
9. Redmon J., Farhadi A., YOLOv3: An Incremental Improvement, Computing Research Repository, 8.4.2018.
10. Sermetcan Baysal, Model Field Particles with Positional Appearance Learning for Sports Player Tracking, Doktorski rad, The graduate school of engineering and science of Bilkent University, 2016.
11. YOLOv3: <https://github.com/ultralytics/yolov3>
12. Ivan Fabijanić, Duboki konvolucijski modeli za praćenje objekata, Diplomski rad, Fakultet elektrotehnike i računarstva, 2017.
13. Rae Jie Tan, Breaking Down Mean Average Precision (mAP), 24.3.2019.,

<https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>, 26.6.2019.

Detekcija igrača u snimkama nogometnih utakmica temeljena na dubokom učenju

Sažetak

Rad se bavi problemom detekcije i praćenja nogometnih igrača na utakmici. Za detekciju se koristi YOLOv3 arhitektura te je pobliže objašnjena njena funkcija gubitka, način detekcije objekata, predikcija omeđujućeg pravokutnika (eng. bounding box) te način klasifikacije objekata. Za praćenje se koristi heuristika koja uz pomoć modela zasnovanog na izgledu (eng. appearance model), modela pokreta i algoritma za klasifikaciju timova dodjeljuje identifikatore igračima te ažurira pozicije igrača tijekom utakmice. Opisani su problemi praćenja te način na koji su riješeni. Uz to, opisan je način učenja klasifikatora za timove te način treniranja i testiranja sustava za detekciju. Skup podataka koji je korišten za razvoj oba sustava je nogometna utakmica snimana nepomičnom kamerom između GNK Dinamo Zagreb i RNK Split.

Ključne riječi: praćenje objekata; detekcija objekata; YOLOv3 arhitektura; heuristike; računalni vid; duboko učenje; strojno učenje; PyTorch

Player Detection in Football Game Video Based on Deep Learning

Summary

This thesis deals with the problem of detection and tracking football players in the game video. For detection, YOLOv3 architecture is used. Its loss function, way of detecting objects, its prediction of bounding boxes and way of classifying objects are explained in more detail in the thesis. The tracking uses a heuristics that assigns identifiers to players and updates their positions during the match with the help of appearance model, motion model and classification algorithm (kNN). Also, the problems that tracking system encounters during its execution are described along with the way how that problems were solved. In addition, it is described how was kNN classifier trained and how was the object detection system trained and tested. The dataset used to develop both systems was a football game video recorded by a stationary camera between GNK Dinamo Zagreb and RNK Split.

Keywords: object tracking; object detection; YOLOv3 architecture; heuristics; computer vision; deep learning; machine learning; PyTorch