Univerza v Ljubljani
Fakulteta za elektrotehniko

Matej Kristan

Sledenje objektov v robotskem nogometu

Diplomsko delo

Mentor: prof. dr. Stanislav Kovačič
Somentor: prof. dr. Drago Matko

Ljubljana, september 2003

Poglavje 1

Uvod

Robotski nogomet je visoko tehnološki šport, ki ga je leta 1995 na korejskem tehnološkem inštitutu razvil profesor Jong-Hwan Kim kot večnamensko okolje za učenje in testiranje aplikacij analize slik, umetne inteligence, senzorjev, komunikacij itd. V zadnjih osmih letih je robotski nogomet doživel velik razmah tako v sklopu zabavne elektronike kot v sklopu testiranja in razvoja novih tehnologij. Danes obstajata dve mednarodni zvezi robotskega nogometa in sicer Robocup in FIRA (Federation of International Robot Association). Vsaka izmed obeh zvez organizira ločena tekmovanja v različnih kategorijah, kategorija pa določa lastnosti izvedbe tekme in sicer od čistih simulacij na računalniku preko mikrorobotov do humanoidnih robotov. Na Fakulteti za Elektrotehniko v Ljubljani so se z robotskim nogometom kategorije MiroSot pričeli ukvarjati leta 2000, in ga poimenovali Robobrc. Robobrc deluje v dveh različicah kategorije MiroSot, ki se razlikujeta le v številu igralcev in dimenzijah igrišča. V prvi različici vključuje vsak tim po tri igralce (igra treh igralcev ali mala liga), v drugi različici pa po pet (igra petih igralcev ali srednja liga). S prehodom iz igre treh na igro petih igralcev se je pojavila potreba po učinkovitem sledilniku, ki bi ločil večje število barv in sledil desetim robotkom in žogici v realnem času.

V diplomski nalogi je obravnavana aplikacija sledenja robotkov pri tekmah Robobrca, zato bomo najprej v uvodu predstavili le tista pravila obeh različic igre, ki so pomembna za

nadzorni sistem računalniškega vida. V nadaljevanju bomo predstavili še področje računalniškega vida in sledenja objektov, kjer bomo podali kratek pregled literature o sledenju v športu in robotskem nogometu.

1.1 Pravila igre v kategoriji MiroSot

Pravila igre v kategoriji Mirosot najdemo v [1], zato bomo v tem podpoglavju predstavili le tista, ki se tako ali drugače nanašajo na temo diplome.

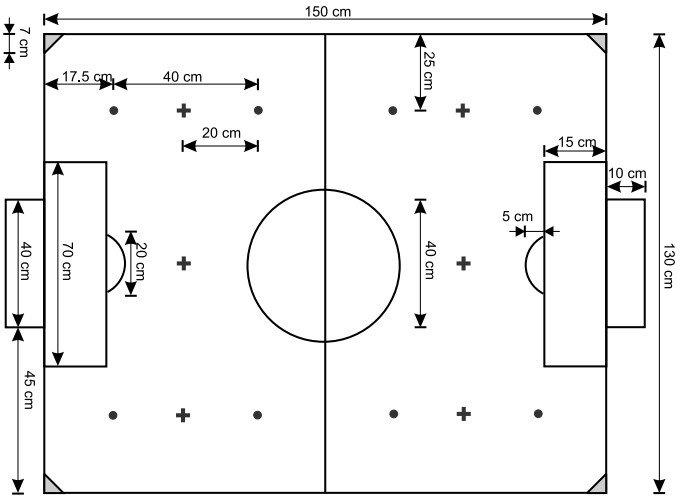
1.1.1 Igrišče

V obeh različicah je igrišče leseno in je pravokotne oblike, njegova površina pa mora biti črne ali temno zelene ne-reflektivne barve. Igrišče obdaja 5cm visok in 2.5cm širok rob, katerega zgornja površina je enaka barvi igrišča. Ploskev, ki gleda v notranjost igrišča, pa je bele barve. Da se žogica ne zagozdi v kot, so v vse štiri kote igrišča vstavljeni enakokraki trikotniki. Tekstura površine igrišča je enaka kot pri ping-pong mizi. Igrišče je vedno locirano v zaprtem prostoru, kjer mora biti osvetljitev okoli 1000 *lux*. Slika 1.1 in slika 1.2 prikazujeta dimenzije in oznake igrišča pri igri treh in petih igralcev.

1.1.2 Žogica in igralci

Za igralno žogico se uporablja oranžna golf žogica s premerom 42.7 *mm*. Igralci so kocke na kolesih s stranico 7.5 *cm*, pri čemer višina antene za RF komunikacijo ni omejena. Na vrhu robotka se nahajata dve barvni oznaki, ki predstavljata timsko in identifikacijsko barvo robotka. Praviloma se barva žogice ne sme nahajati na zgornji strani robotka. Mera različnosti barve žogice od ostalih barv ni podana. Oznaka modre ali rumene barve (ki jo dodelijo organizatorji) predstavlja timsko barvo robotka. Barvna oznaka mora biti velikosti

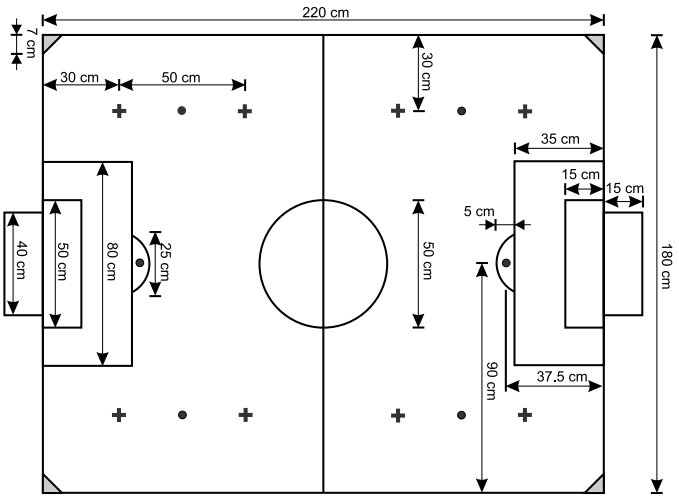
vsaj 3.5 *cm* x 3.5 *cm* in je homogene barve. Nekatere ali vse identifikacijske barve so lahko enake pri obeh timih. Da je omogočeno infra-rdeče zaznavanje, morajo biti stranske ploskve robotkov obarvane svetle barve. Slika 1.3 prikazuje sliko robotka in žogice.



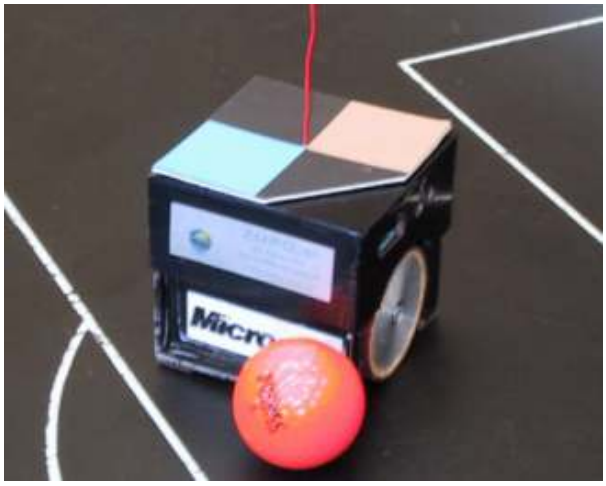
Slika 1.1: *Skica igrišča za igro treh igralcev*

1.1.3 Sistem umetnega vida

Za identifikacijo robotkov in žogice se lahko uporablja sistem umetnega vida. Timska kamera ali senzorski sistem sta lahko nameščena le preko in nad polovico igrišča (vključno s sredinsko črto), ki pripada dotičnemu timu. Če želita oba tima imeti kameri nameščeni nad centrom igrišča, se kameri postavi drugo ob drugi ekvidistančno od sredinske linije in čim bližje. Senzorski sistem in kamere morajo biti nameščeni vsaj 2 *m* nad igriščem.



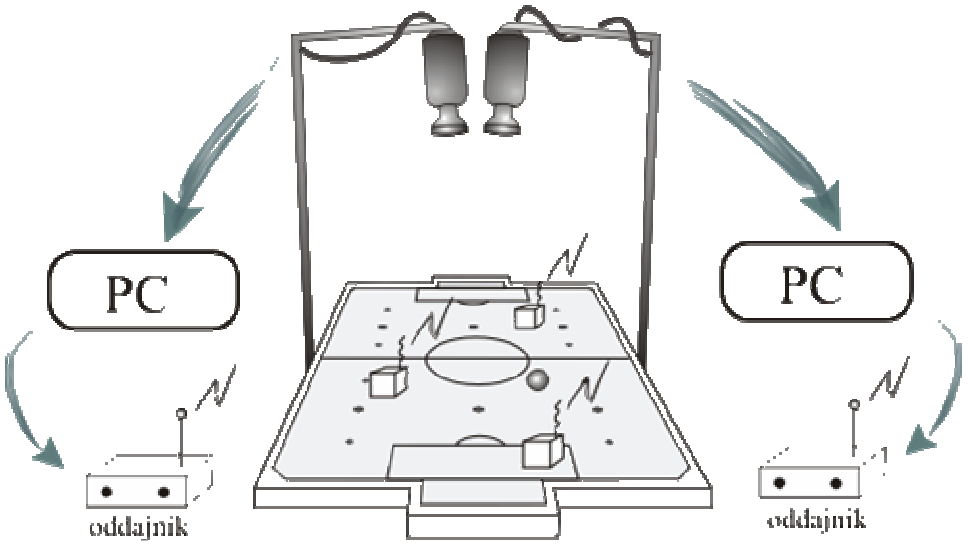
Slika 1.2: *Skica igrišča za igro petih igralcev*



Slika 1.3: *Slika robotka in žogice*

1.2 Celotni sistem v kategoriji MiroSot

Tekmo igrata dva tima s tremi ali petimi robotki (odvisno od igre), od katerih je eden lahko golman. Vsakemu timu je dovoljen le en računalnik, ki je v glavnem namenjen za procesiranje informacij umetnega vida. Skico celotnega sistema prikazuje slika 1.4.



Slika 1.4: *Skica celotnega sistema*

1.3 Računalniški vid in sledenje objektov

Računalniški vid je ena od najmlajših panog računalništva, ki temelji na pridobivanju informacije o sestavnih delih prizora z obdelavo vizualnih podatkov. Kot močan pripomoček se je uveljavil na mnogih področjih, npr. v strojni industriji, farmacevtski industriji, medicini, biologiji, fotogrametriji, astronomiji, nadzorovanju in športu. Objekte na sliki se razpoznava s tako imenovano razgradnjo slike (*angl. segmentation*). Le-ta upošteva lokalne značilnosti slike, ki jih z globalnimi kriteriji skušamo povezati v smiselne enote.

Sledenje (*angl. tracking*) in analiza gibanja (*angl. motion analysis*) sta tista dela računalniškega vida, katerih naloga je pridobiti informacije o gibanju določenih elementov v prizoru. Na podlagi teh informacij lahko dalje izpeljemo nove informacije npr. hitrost,

pospešek, območje gibanja itd. Sledenje se je v zadnjih letih precej uveljavilo na področju športa in nadzorovanja.

S. S. Intille v [2] sledi objektom v igri ameriškega nogometa, posneto z eno gibljivo kamero, na podlagi primerjave s prilagodljivo predlogo (*angl. adaptive template matching*). Perš in Kovačič [3] sledita igralcem v igri rokometa, posneti z dvema statičnima kamerama, na podlagi odštevanja trenutne slike od referenčne in na podlagi barve dresa v RGB prostoru. Podobno aplikacijo sledenja na enakih posnetkih je predstavil Grešak [4], vendar uporablja HSI barvni prostor in vsi igralci ne nosijo drese različnih barv. Pregled sledenja igralcev z dinamično kamero v športnih igrah podaja Lesjak [5]. Kot aplikacije nadzorovanja najdemo sledenje v S. S. Intille *et. al.* [6], kjer s statično kamero sledijo otrokom med igro v sobi. Za sledenje eni osebi na podzemni železnici v realnem času D. Comaniciu *et. al.* v [7] uporabljajo Mean Shift algoritem. S. J. McKenna *et. al.* v [8] uporablja kot model funkcije porazdelitvene gostote naučene barve Gaussov prilagodljiv model barve za sledenje objekta v realnem času v HSI prostoru, pri čemer zanemari komponento intenzitete. G. Klančar et al. [9] z barvnim upragovanjem s paralelepipedi v RGB barvnem prostoru dosega sledenje šestih robotkov in žogice v realnem času (30Hz) na slikah dimenzije 640X480 na računalniku pentium s 1.7 GHz procesorjem. V [10] so P. Borensztejn et.al. predstavili uspešno segmentacijo slike v igrah kategorije Mirosot na podlagi Bayesovega naivnega razvrščevalnika v RGB barvnem prostoru, vendar ni podatkov o velikosti slik in frekvenci sledenja. Uporaba nevronskih mrež za razpoznavanje objektov v RoboCup-99 na HSI barvnem prostoru so predstavili C. Amoroso *et. al.* v [11]. L. Iocci in D. Nardi [12] uporabljata Houghovo transformacijo za samo lokalizacijo robotkov v robotskem nogometu, kjer globalni senzorski sistem ni dovoljen, in so kamere locirane na vsakem robotku.

1.4 Sestava diplomske naloge

Cilj diplomske naloge je bil razviti sistem za uspešno sledenje robotkov v tekmah Robobrca. Tako bomo v drugem poglavju navedli nekatere splošne metode računalniškega vida, ki jih uporabljata proceduri za sledenje in kalibracjo kamere. V tretjem poglavju bomo opisali nastanek slike preko kamere, izpostavili glavne probleme in podali poenostavljen model kamere. Nekoliko širše bomo v četrtem poglavju podali sestavo sledilnika, v petem poglavju bo opisana izvedba kalibracije kamere in sledilnika, z rezultati. V zadnjem- šestem poglavju bomo povzeli ugotovitve in predlagali izboljšave. V dodatku bosta opisana programa za sledenje in kalibracijo kamere, prikazan pa bo tudi tipičen primer inicializacije sledilnika.

Poglavje 2

Osnovni pojmi in nekateri postopki obdelave slik

V nadaljnih poglavjih bomo omenjali nekatere postopke obdelave slik, ki jih bomo zaradi njihove splošnosti navedli na tem mestu. Postopek entropijskega upragovanja slike (podpoglavje 2.1), morfološkega tanjšanja (podpoglavje 2.2), Houghove transformacije (podpoglavje 2.3) in prileganje premic na podatke (podpoglavje 2.4) so uporabljeni v proceduri za kalibracijo kamere, medtem ko je postopek označevanja komponent (podpoglavje 2.4) uporabljen v sledilniku.

2.1 Entropijsko upragovljanje slike z enim pragom

Digitalna slika, ki jo obdelujemo v aplikacijah računalniškega vida, je skupina diskretnih vrednosti svetlosti. Postopkov za razčlenjevanje tako zapisane slike je izredno veliko, eden od najpreprostejših pa je upragovanje slike. Vse metode upragovanja temeljijo na takšni ali drugačni interpretaciji histograma slike, iz katerega nato ocenimo prag nad katerim naj so vrednosti 1 in pod katerim naj so vrednosti 0. Tako upragovanje prevede sivinsko večnivojsko sliko v dvonivojsko ali binarno. Funkcijo predstavljajo naslednje enačbe:

(2.1)

(2.2)

$$U(x,y) = fp(P(x,y),t),$$
$$fp(I,t) = \begin{cases} 1; & I >= t \\ 0; & \text{sicer} \end{cases}$$

Kjer je *P(x,y)* sivinska slika, *t* pa je pragovna vrednost upragovanja. Eden od postopkov za avtomatsko upragovljanje slike je postopek upragovljanja z maksimizacijo informacije. Pri tem postopku gre v bistvu za določanje tiste pragovne vrednosti *t*, ki maksimizira informacijo v histogramu levo in desno od pragovne vrednosti.

Če sliko upragovljamo s pragom t, delimo množico sivih nivojev $G_L = \{0, 1, ..., L-1\}$, ki ji pripada porazdelitev sivih nivojev $P = \{P_0, P_1,...,P_{L-1}\}$, v dve tuji podmnožici:

$$\underline{G_0} = \{0, 1, ...,t\} \tag{2.3}$$

in

$$\underline{G_1} = \{t+1, t+2,..., L-1\}, \tag{2.4}$$

katerima pripadata porazdelitvi:

$$\begin{aligned} \underline{P_0} &= \left(\frac{P_0}{P^*(t)}, \frac{P_1}{P^*(t)}, ..., \frac{P_t}{P^*(t)} \right), \\ \underline{P_1} &= \left(\frac{P_{t+1}}{1-P^*(t)}, \frac{P_{t+2}}{1-P^*(t)}, ..., \frac{P_{L-1}}{1-P^*(t)} \right), \end{aligned} \tag{2.5}$$

kjer je

$$P^*(t) = \sum_{i=0}^{t} P_i \ . \tag{2.6}$$

Vzemimo, da sta para *($\underline{G_0}$,$\underline{P_0}$)* in *($\underline{G_1}$,$\underline{P_1}$)* verjetnostni shemi dveh diskretnih naključnih spremenljivk z informacijo:

$$H_0(t) = -\sum_{i=0}^{t} \frac{P_i}{P^*(t)} \log \frac{P_i}{P^*(t)},$$
$$H_1(t) = -\sum_{i=t+1}^{L-1} \frac{P_i}{1-P^*(t)} \log \frac{P_i}{1-P^*(t)} \quad . \tag{2.7}$$

Na sliki, ki je upragovljena s pragom *t*, predstavljata $H_0(t)$ in $H_1(t)$ informaciji predmeta in podlage. Optimalni prag *t** je tisti ki maksimizira vsoto informacije:

$$t^* = \arg \underset{t \in G_L}{Max} \{H_0(t) + H_1(t)\}. \tag{2.8}$$

2.2 Morfološko tanjšanje

Rezultati nekaterih postopkov v obdelavi slik (npr. upragovanje) so binarne slike, ki predstavljajo elemente zanimanja (različne obrise, črte, itd.). Pogosto potrebujemo za dobro nadaljno obdelavo vse elemente debeline enega slikovnega elementa (*angl. pixel*). Tanjšanje je operacija, ki oblike na binearnih slikah stanjša na debelino enega slikovnega elementa, torej zmanjša komponente slike na njihovo bistveno informacijo. To informacijo včasih imenujemo tudi skelet. Čeprav lahko tanjšanje apliciramo na območja kakršnekoli oblike, je primarno uporabnejša za »razvlečene« oblike.

Tanjšanje mora zadostiti sledečim pogojem:

1. Povezana področja na sliki se stanjšajo v povezane strukture linij.
2. Rezultat mora biti osem smerna povezanost.
3. Približne pozicije koncev linij morajo biti ohranjene.
4. Postopek tanjšanja mora rezultirati v približku sredinskih linij.
5. Prisotnost odvečnih izrastkov ki so posledica tanjšanja mora biti minimizirana.

Četrte zahteve ne moremo vedno izpolniti. Za primer vzemimo vertikalno linijo debeline dveh slikovnih elementov. Prava sredinska linija bi morala potekati vertikalno med obema slikovnima elementoma. Ker tega ne moremo predstaviti v digitalnih slikah, bo rezultat linija, ki bo potekala ob eni strani originalne. Kar se tiče zahteve pet, je včasih težko opredeliti kaj je šum in kaj ni. Izrastkov, ki so posledica majhnih izboklin, se želimo znebiti, tistih, ki nastopijo zaradi velikih izboklin, pa včasih ne. Čeprav imajo nekatere metode tanjšanja že vgrajene parametre za izločanje izrastkov, pa mnogi avtorji predlagajo ločeno tanjšaje in filtriranje šuma in izrastkov.

Bistvo morfološkega postopka tanjšanja je pregledovanje sosedov vsakega slikovnega elementa na sliki in primerjanje z vnaprej določenimi maskami velikosti 3x3 slikovnih elementov. Tak postopek je iterativni in z vsako iteracijo »lupi« področje sloj za slojem, dokler ni stanjšano na debelino enega slikovnega elementa.

Slika prikazuje dve izmed osmih mask, preostalih šest pa dobimo tako, da maski W1 in W2 zavrtimo po trikrat za 90°. Zvezdice v maskah pomenijo, da sovpadanje na teh mestih ni zahtevano.

0	0	0
*	1	*
1	1	1

(W1)

*	0	0
1	1	0
*	1	*

(W2)

Slika 2.1: *Dve izmed osmih mask, ki jih uporabimo pri tanjšanju*

Naj bo B(x,y) binearna slika in Wi (i=1...8) maska. Postopek tanjšanja sledi algoritmu 2.1.

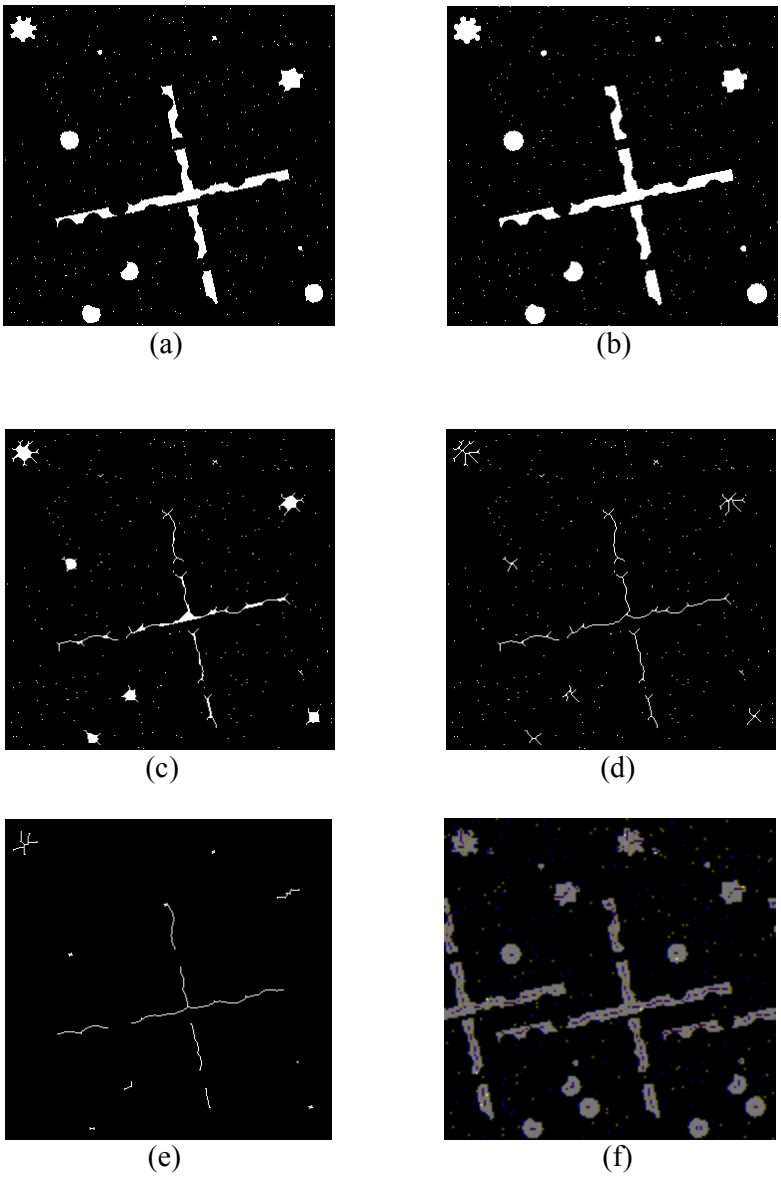
Algoritem 2.1: *Tanjšanje binarne slike*

```
dokler »prihaja do sprememb« naredi
 za »vsako masko Wi« naredi
  za »vsak (x,y), za katerega B(x,y)==1« naredi
   »primerjaj vrednosti 8-ih sosedov slikovnega elementa
    (x,y) z vrednostmi polj Wi, ki so položeni nanje«;
   če »vse vrednosti polj na maski Wi in istoležnih
      slikovnih elementov sovpadajo« potem
    B(x,y)=0;
   konec-če
  konec-za
 konec-za
konec-dokler
```

Po končanem tanjšanju je potrebno še odstraniti šum in odvečne izrastke to storimo po sledečem postopku:

Pregledujemo celo sliko. Če naletimo na slikovni element z le enim sosedom, smo na začetku možne nezaključene krivulje. Premaknemo se na soseda in dalje ponavljamo toliko časa, dokler ima element le dva soseda. Tako izmerimo dolžine vseh odprtih krivulj, in tiste ki so manjše od določene prednastavljene pragovne vrednosti pobrišemo.

Primer tanjšanja in filtriranja prikazujejo slike 2.2 (a-e).



Slika 2.2: *binarna slika (a), binarna slika po prvi iteraciji (b), binarna slika po peti iteraciji (c), binarna slika po zadnji (petnajsti) iteraciji (d), stanajšana slika po odstranjevanju izrastkov in šuma (e) in binarna slika z njenim skeletom (f)*

2.3 Houghov transform za iskanje ravnih črt

Houghova transformacija (HT) je način detektiranja v naprej znanih neprekinjenih in prekinjenih oblik kot so črte, krogi ali krivulje v digitalni sliki.

HT za detekcijo ravnih črt deluje na preslikavi vsakega ugodnega slikovnega elementa v parametrični prostor (PP) preko transformacijske enačbe:

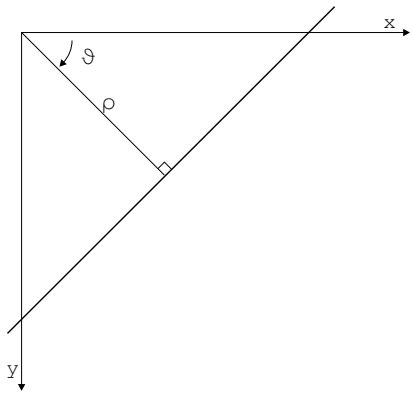
$$y = k \cdot x + n \tag{2.9}$$

ali pa

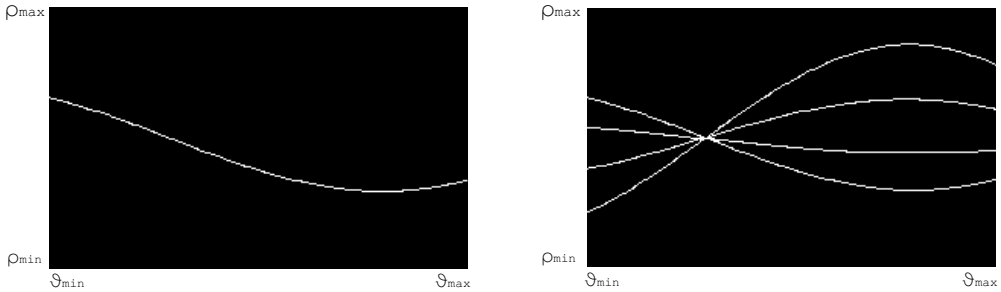
$$\rho = x \cdot \cos(\vartheta) + y \cdot \sin(\vartheta) \tag{2.10}$$

Druga enačba je pogosteje uporabljana, saj se s prvo pojavljajo težave pri opisu premic, ki so vzporedne z y osjo. Enačba 2.10 predstavlja enačbo premice v polarnih koordinatah, ki je enolično definirana s parametroma υ in ρ. Kot je prikazano na sliki 2.3, predstavlja υ kot med x-osjo in pravokotno razdaljo med koordinatnim izhodiščem in premico, ρ pa je pravokotna razdalja med premico in izhodiščem.

Potek transformacije se prične s preiskovanjem celotne slike. Vsak ugoden slikovni element na katerega naletimo je preslikan preko enačbe 2 v parametrični prostor (PP). Parametrični prostor za premice je matrika dimenzij *th_dim* krat *ro_dim*. Vsaka celica ima pred začetkom transforma vrednost nič. Preslikava nekega slikovnega elementa s koordinatami (x_1,y_1) v PP poteka tako, da za vsak υ_i od υ_{min} do υ_{max} pri stalnih x_1 in y_1 izračunamo ρ_i po enačbi 2.10 in celici (ρ_i, υ_i) povečamo vsebino za ena. Transformacija enega slikovnega elementa po enačbi 2.9 v PP je premica, po enačbi 2.10 pa sinusoida (slika 2.4). Če preslikamo po enačbi 2.10 kolinearne točke v PP, dobimo več sinusoid, ki se praviloma sekajo v eni točki. Slika 2.4b predstavlja preslikavo štirih kolinearnih točk v PP. Velja, da premici iz slikovnega prostora, ustreza točka v parametričnem prostoru.



Slika 2.3: *Predstavitev premice z ρ in υ*



Slika 2.4: *PP pri transformaciji enega slikovnega elementa (levo) in PP s preslikavo štirih (desno)*

2.3.1 Parametrični prostor in interpretacija vsebine

Po preslikavi celotne slike v PP je potrebno poiskati celice z največjo vsebino, saj njihove koordinate (ρ,υ) predstavljajo najdaljše (ne nujno neprekinjene) črte v sliki.

Učinkovito določanje maksimumov je močno odvisno od razdelitve PP na zbiralne celice, kar izhaja iz dejstva, da uporabljamo diskretni prostor. Če je razdelitev pregroba, detektirane premice ne bodo točno ležale na črtah na sliki, če pa je razdelitev preveč fina, se pojavi problem pri detektiranju maksimumov v PP, saj bodo slabo izraženi.

Problem se prav tako pojavi, ker vsaka točka iz slike robov voli za več kot eno točko v PP. Že v primeru, ko imamo sliko z le eno ravno črto, se v PP pojavi en velik vrh, in več manjših, katerih višina pada z oddaljenostjo od glavnega vrha. Kombiran efekt teh stranskih vrhov lahko v primeru, ko je prisotnih več črt, povzroči nastop napačnega vrha v PP.

V primeru, ko imamo poleg slike še informacijo o smeri linije kateri nek slikovni element pripada (npr. če smo sliko obdelali s Cannyjevim ali Sobelovim operatorjem), lahko temu primerno zmanjšamo število celic za katere voli nek slikovni element. Če take informacije nimamo, se lahko poslužimo MMHT (Mathematical Morfology Hough Transform) [13].

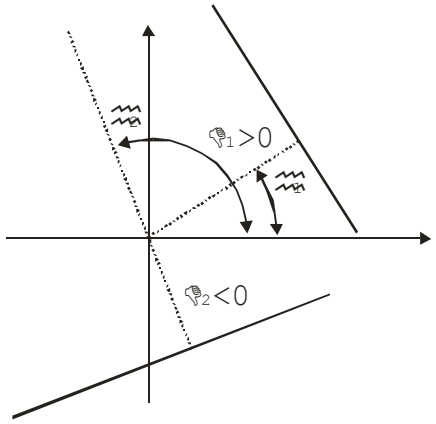
Mnogo avtorjev je predlagalo različne načine detekcije vrhov (maksimumov) v PP. Najpreprostejši način je denimo najti celice, ki so lokalni maksimum v neki v naprej določeni okolici. Zaradi zgoraj omenjenih problemov nekateri avtorji predlagajo glajenje tudi PP z na primer Gaussovim filtrom.

Možno je tudi detektirati največji maksimum, in upražiti PP z nekim procentom tega maksimuma. V takem primeru dobimo nekakšne otočke celic v PP. Dalje se aplicira dilatacija, povezane celice označimo, in med tistimi z isto oznako poiščemo po en lokalni maksimum. To metodo je možno še izboljšati, če za vsako označeno celico (ρ_i, υ_i) v PP ponovno izračunamo glasove, tako da preštejemo koliko slikovnih elementov leži na premici (ρ_i, υ_i), in te slikovne elemente sproti brišemo iz slike (inverzni HT).

Pri uporabi (ρ, υ) PP se pojavi vprašanje kako nastaviti υ_{max}, υ_{min}, ρ_{max} in ρ_{min}. Navadno je izhodišče koordinatnega sistema slike v njenem centru. Omejitve so tako:

$$\begin{aligned}\rho_{\min} &= -\frac{R_{\max}}{2} \\ \rho_{\max} &= \frac{R_{\max}}{2} \\ \vartheta_{\min} &= 0 \\ \vartheta_{\max} &= \pi\end{aligned} \tag{2.11}$$

V takem primeru se srečamo s pojavom negativnega ρ. Slednji nastopi zaradi črt pri katerih je kot υ v tretjem in četrtem kvadrantu. Primer prikazuje slika 2.5.



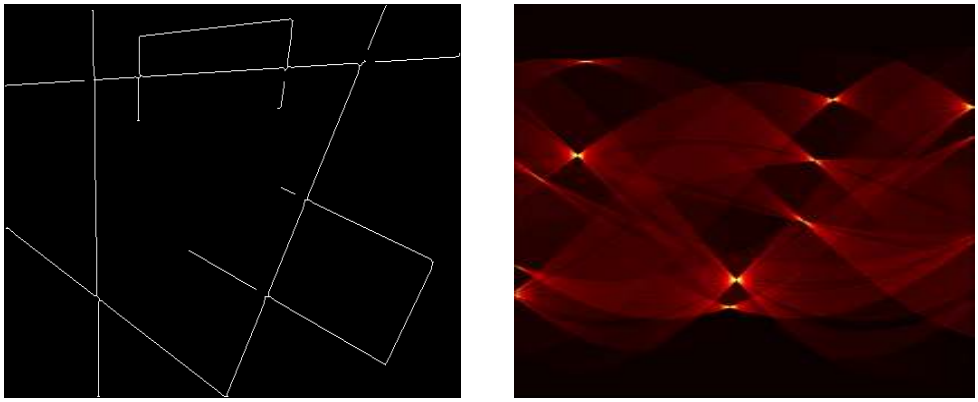
Slika 2.5: Ker υ teče le od 0 do π, moramo v primeru kota večjega od π uporabiti negativen ρ.

Pri vrednostih v PP kjer je υ *0* in π, lahko pride do pojava napačnega lokalnega maksimuma. Denimo, da pri koordinatah *(ρ,0)* v sliki obstaja premica. Za to premico dobimo torej dva približno enako velika maksimuma v PP in sicer pri *(ρ,0)* in *(-ρ,π)*.

V našem primeru smo iskali po PP vse lokalne maksimume z v naprej določeno okolico ($\Delta\rho=10$ slikovnih elementov, $\Delta\upsilon=5°$). Razločljivost PP se je nastavila po sledečih enačbah:

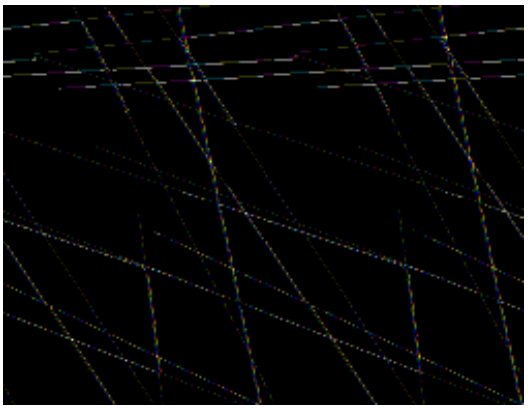
$$\vartheta\_korak = \arctan(\frac{1}{\rho_{\max}})$$
$$\rho\_korak = \rho_{\max} \cdot \sin(\frac{\vartheta\_korak}{2}) \tag{2.12}$$

kjer je ρ_{max} dolžina diagonale slike.



Slika 2.6: *Črte v slikovnem prostoru (levo) in PP po preslikavi (desno)*

Ker lahko v neposredni bližini obstajata dva enaka maksimuma, smo definirali lokalne maksimume kot vrednost v okolici katere ne obstaja nobena večja. Po končanem pregledu PP imamo zbirko lokalnih maksimumov, ki pa so lahko podvojeni. Izločevanje postane lahko težavno, saj imamo zapisane koordinate maksimumov tudi z negativnimi vrednostmi ρ. V ta namen pretvorimo vse koordinate dobljenih maksimumov tako, da so vsi ρ pozitivni, υ pa teče od $(0 \text{ do } 2\pi)$. Ponovno preiščemo dobljeno zbirko in združimo maksimume, če so v prej omenjeni okolici (če v neki okolici obstaja več maksimumov, vzamemo največjega). Posebna težava se nahaja pri koordinatah $(0,0)$. Če poteka neka premica skozi to izhodišče in je nekoliko zašumljena, lahko detektor pokaže dve premici in sicer (ρ,υ) in $(\rho, \upsilon+\pi)$, pri čemer ima ρ vrednost nič ali le malo večjo on nič. V ta namen je potrebno za vsako premico, ki ima dovolj majhen ρ pregledati, če obstaja še kakšna z majhnim ρ in se njuni υ razlikujeta za približno π. Med takimi dvemi premicami je na primer spet potrebno vzeti tisto, ki gre skozi več točk.



Slika 2.7: *Rezultat detektiranja črt po zgoraj omenjenem primeru: najdeno je 10 lokalnih maksimumov*

2.4 Prileganje premic na podatke

Včasih so premice, ki jih dobimo preko HT sicer dobre, vendar bi jih želeli še nekoliko natančneje »napeti« na podatke v njihovi okolici. V ta namen se uporablja metoda regresije premic na podatke. Tu bomo predstavili dve, ki sta si sicer zelo sorodni, vendar različni kar se tiče robustnosti glede na šum v podatkih.

2.4.1 Metoda najmanjših kvadratov

Enačbo premice v polarnih koordinatah predstavlja enačba:

$$\rho = x\cos\theta + y\sin\theta \tag{2.13}$$

Minimizirati je potrebno kvadratno pravokotno razdaljo točk (x_i,y_i) od premice (3).

$$\chi^2 = \sum_i (x_i \cos\theta + y_i \sin\theta + \rho)^2 \tag{2.14}$$

Rešitev regresije je

$$\rho = \bar{x}\cos\theta + \bar{y}\sin\theta \tag{2.15}$$

kjer sta

$$\begin{aligned} \bar{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \bar{y} &= \frac{1}{n}\sum_{i=1}^{n} y \end{aligned} \tag{2.16}$$

Parameter θ je podan po enačbi (7)

$$\tan\theta = \frac{a}{b+c} - \frac{\pi}{2} \tag{2.17}$$

in velja

$$a = 2\sum_{i=1}^{n} x_i' y_i' \tag{2.18}$$

$$b = \sum_{i=1}^{n} x_i' - \sum_{i=1}^{n} y_i' \tag{2.19}$$

$$\mathrm{c} = \sqrt{\mathrm{a}^2 + b^2} \tag{2.20}$$

pri čemer je

$$x_i' = x_i - \bar{x} \tag{2.21}$$

$$y_i' = y_i - \bar{y}\ . \tag{2.22}$$

Popolna regresija uporablja za normo najmanjšo kvadratno napako. Le-ta je optimalna, če je napaka normalno porazdeljena, ni pa primerna če vhodni podatki vsebujejo točke, ki ne ležijo na iskani premici (*neskladne točke*, *angl. outliers*). Neskladne točke so lahko

posledica šuma ali pa napake pri razvrščanju točk v razrede premic. Praktičen primer prikazuje slika 2.8.

Celo ena neskladna točka je včasih dovolj za premik regresijske premice daleč stran od njene pravilne lege. V ta namen se uporabljajo metode robustne regresije, katerih predstavnica je regresija po metodi mediane najmanjših kvadratov.

2.4.2 Prileganje premic po metodi mediane najmanjših kvadratov

Ta metoda je zelo preprosta za implementacijo, in se je pokazala za zelo močno orodje pri regresijskih problemih, kjer je odstotek neskladnih točk zelo visok (do 50%). To pomeni, da je lahko do pol podatkov poljubnih ne da bi pri tem prizadeli uspešnost regresije [14].

Približki parametrov premice so dani preko minimiziranja mediane kvadratov pravokotnih razdalj :

$$\min(med(\chi^2)) \tag{2.23}$$

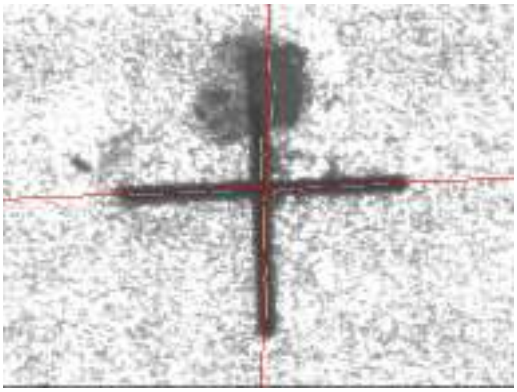
Rezultat regresije po metodi mediane najmanjših kvadratov prikazuje slika 2.9.

2.5 Označevanje povezanih področij slike

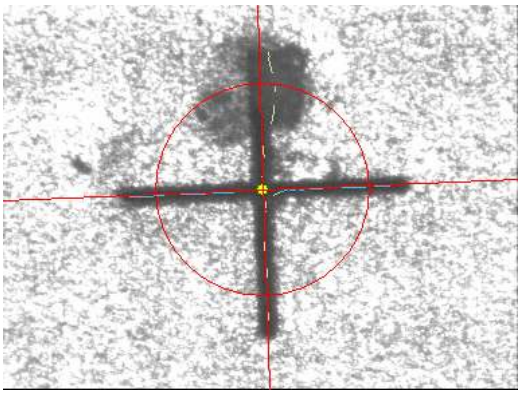
Naj obstaja neka množica slikovnih elementov $\boldsymbol{B}$. Če je možno od vsakega $p_i \in \boldsymbol{B}$ priti do vsakega $p_j \in \boldsymbol{B}$ preko nekaterih elementov iz množice $\boldsymbol{B}$, potem je taka množica povezano področje.

Eden od najpogosteje uporabljanih postopkov v računalniškem vidu je označevanje komponent (*angl.: component labelling*). Postopek temelji na iskanju zgoraj omenjenih množic v sliki katerim je funkcija podobnosti predpisana (denimo sivinski nivo večji od

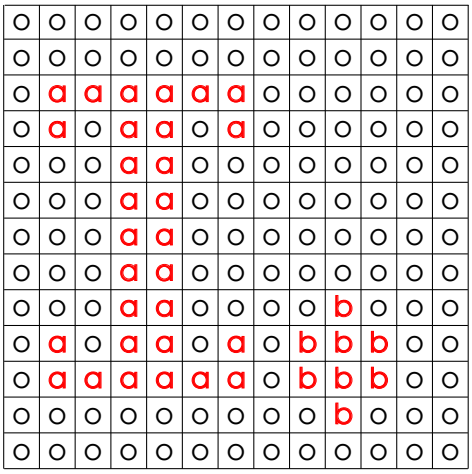
neke pragovne vrednosti, zahteva po enakosti barve ali le enega kanala itd...). Označevanje komponent je zelo ugoden postopek v primerih, ko imamo opravka z večimi segmenti na sliki in so objekt zanimanja vsi ali le eden. V splošnem obstajata dva postopka označevanja komponent: rekurzivni in iterativni. Iterativni je primeren za sekevenčne procesorje, vendar pa je rekurzivni bolj razširjen.



Slika 2.8: *Prva množica točk za regresijo je označena z modro, druga pa z zeleno. Z rdečo so izrisane detektirane premice. Viden je uspeh totalne regresije za prvi set in neuspešna regresija za drugi set, saj slednji vsebuje napako, ki je posledica packe preko navpične črte plusa.*



Slika 2.9: *Uspešna regresija po metodi mediane najmanjših kvadratov na podatke označene z zeleno barvo. Kljub velikem številu neskladnih točk je prileganje premice uspešno.*



Slika 2.10: *primer slike s tremi povezanimi komponentami A, B in O (ozadje)*

Princip iterativnega algoritma je pregledovanje dveh vrstic hkrati in sicer preverjanje podobnosti med trenutnim slikovnim elementom, tistim ki leži levo od opazovanega, in slikovnim elementom, ki leži eno vrstico nad opazovanim. Hkrati mora obstajati tabela oznak, kjer je označeno kateremu področju pripada kateri slikovni element. V tabeli se oznake tekom pregleda slike spreminjajo glede na povezanosti elementov. Tak postopek zahteva le en obhod po sliki in en po tabeli oznak.

Poglavje 3

Kalibracija

Vedno, kadar imamo opravka s pridobivanjem informacije o položajih objektov v slikah, je potrebno poznati transformacijo med koordinatnim sistemom slike in sveta, ki ga le-ta predstavlja. Postopek določevanja te transformacije se imenuje kalibracija in v našem primeru pomeni ugotovljanje parametrov preslikave med 3D svetovnim prostorom in 2D prostorom slike.

Kalibracija kamere se prevede na določevanje njenih internih (notranjih) in eksternih (zunanjih) parametrov. Notranji (*angl. intrinsic)* parametri se nanašajo na geometrijske lastnosti kamere, zunanji (*angl. extrinsic*) pa na orientacijo in pozicijo kamere v prostoru. Postopek kalibracije torej lahko podamo kot iskanje parametrov dveh preslikav: preslikavo 3D svetovnih koordinat v 3D koordinate kamere in preslikavo 3D koordinat kamere v 2D koordinate slike.

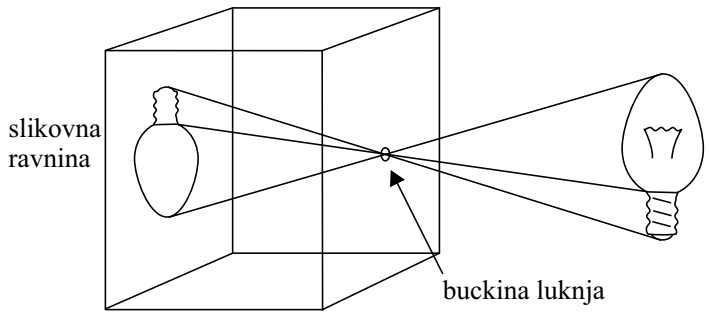
V strokovni literaturi najdemo mnogo člankov ki se tičejo pristopov h kalibraciji in modelov kamere. Najbolj razširjen postopek je ob koncu osemdesetih let predstavil R. Y. Tsay [15], vendar naša aplikacija ne potrebuje tako natančnega modela kamere. V ta namen bomo v poglavju 3.1 najprej predstavili nekoliko bolj natančen model kamere, ki ga bomo v poglavju 3.2 poenostavili.

3.1 Model kamere

V sledečih podpoglavjih bomo predstavili nastanek slike preko kamere, ter glavne dejavnike popačitev slike.

3.1.1 Centralno projekcijski model kamere

Predstavljajmo si škatlo s prosojnim zadnjim delom. Če z buciko v centru sprednjega dela škatle izvrtamo luknjo in jo v zatemnjeni sobi obrnemo proti osvetljenem objektu, se na njenem zadnjem delu pojavi obrnjena slika opazovanega objekta. To sliko generirajo žarki, ki prihajajo skozi luknjo v sprednji strani. Če bi (kolikor je fizikalno mogoče) zmanjšali luknjo, bi natanko en žarek potekal skozi vsako točko v slikovni ravnini na zadnji strani škatle (slika 3.1). Takemu modelu kamere pravimo centralno projekcijski (*angl. pin-hole*) model. V resnici je luknja končne velikosti in skozi vsako točko v slikovni ravnini poteka stožec žarkov. Večja ko je luknja, širši so stožci žarkov ki potekaljo skozi vsako točko na slikovni ravnini, in bolj zamegljena je slika. To je glavni vzrok zaradi katerega se v dejanskih primerih v odprtino vstavi leča. S tem je odprtina lahko manjša, saj leča zbere več svetlobe.



Slika 3.1: *Generiranje slike po centralno projekcijskem modelu kamere.*

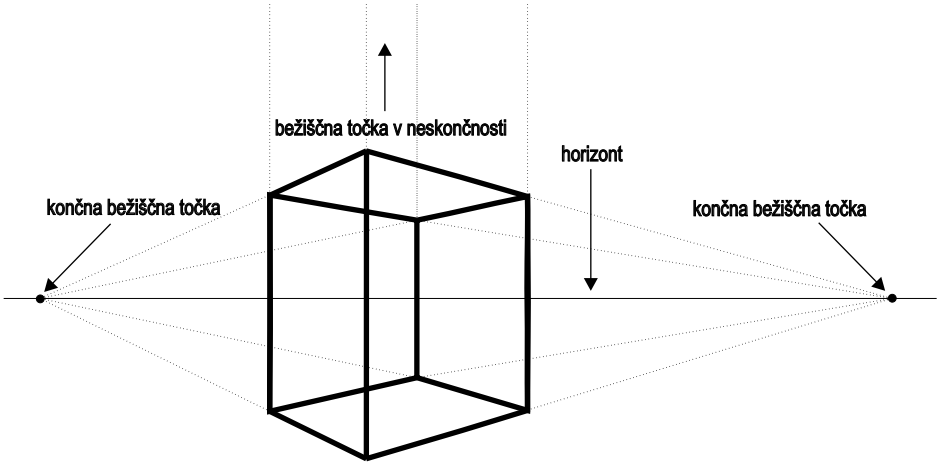
3.1.2 Perspektivna projekcija in bežiščne točke

Projekcija na ravnino slike ali perspektivna projekcija, je projeciranje iz prostora kamere na nepopačen slikovni prostor (ravnino). Naj bo ${}^{K}P$ točka v koordinatnem sistemu kamere, ${}^{c'}p$ pa njena projekcija na slikovno ravnino v koordinatah k.s. ***C'*** (slika 3.5). Povezavo med koordinatami pri predpostavki centralno projekcijskega modela kamere predstavlja matrična enačba 3.1.

$$\begin{bmatrix} x_{c'} \\ y_{c'} \\ z_{c'} \\ w \end{bmatrix} = {}^{c'}_{k}T \cdot \begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} \; ; \; {}^{c'}_{k}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f_k} & 0 \end{bmatrix} \tag{3.1}$$

Pri tem je f_k efektivna goriščna razdalja kamere ali razdalja med odprtino in točko kjer optična os prebada slikovno ravnino (slika 3.5).

Pri predpostavki o idealni projekciji, recimo pri centralno projekcijskem modelu kamere, se skupina vzporednih linij v sceni preslika v šop linij v sliki, ki se sekajo v skupni točki. To presečišče, ki je lahko tudi v neskončnosti, se imenuje bežiščna točka. Tiste bežiščne točke, ki ležijo na isti ravnini v sceni, tvorijo v sliki bežiščno linijo (obzorje). Slika 3.2 prikazuje tri bežiščne točke in bežiščne linije kocke. Tri bežiščne točke, ki so slika treh ortogonalnih smeri v prostoru tvorijo v slikovni ravnini trikotnik. Presečišče višin tega trikotnika imenujemo središče slike (*angl: principal point*), ki je hkrati prebodišče slikovne ravnine z optično osjo v koordinatah slike. Kljub temu, da je središče blizu centra slike, ta dva pojma ne smemo enačiti. Če je višina slike *H*, širina pa *W*, so koordinate centra slike *(W/2,H/2)*, središče pa se nahaja v neki bližnji okolici.



Slika 3.2: *Tri bežiščne točke in bežiščne linije kocke*

3.1.3 Kamere z lečami

Kot smo že omenili v podpoglavju 3.1.1, rešujemo problem velikosti odprtine kamere z uporaba leč, ki pa dodatno prispevajo k popačenjem in zamegljenosti slike. Srečujemo se z dvema vrstama popačenja in sicer: radialno in tangencialno. V praksi se navadno popravlja le radialno, nekateri natančenjši sistemi pa vsebujejo tudi korekcijo tangencialnega. Popačenje izgleda kot projekcija slike na sfero in je enako za vse točke, ki so enako oddaljene od središča slike. Učinke take distorzije je moč zapisati matematično z neskončnim polinomom sodih potenc. Dovolj dober približek dobimo že z uporabo le polinoma tretjega ali nižjega reda [16]. Model predstavlja enačba (3.2),

$$x_p = \frac{x_n}{1 + k_1 r^2 + k_2 r^4}$$
$$y_p = \frac{y_n}{1 + k_1 r^2 + k_2 r^4} \tag{3.2}$$

kjer sta (x_p,y_p) koordinati točke popačene z radialno distorzijo, (x_n,y_n) nepopačeni koordinati iste točke, r pa razdalja med središčem slike in točko (x_p,y_p).

Distorzijo lahko opišemo tudi z modelom, ki so ga predstavili J. Perš *et. al.* [17], čigar prednost je potreba po oceni le enega parametra (enačba 3.3).

$$f(r_i) = R_i = \frac{H}{2} \frac{(e^{-\frac{2r_i}{H}} - 1)}{e^{-\frac{r_i}{H}}}. \tag{3.3}$$

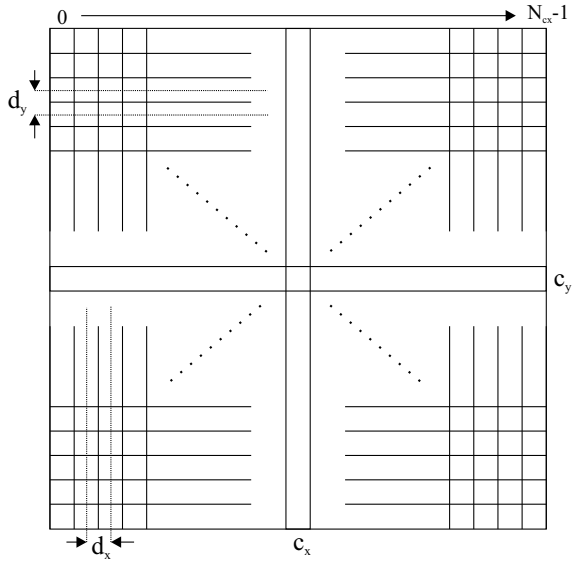
Pri čemer je R_i nepopačena razdalja med neko točko in središčem slike, r_i je razdalja med isto popačeno točko in središčem slike, H pa je parameter fokalne razdalje leče.

3.1.4 CCD kamere

V večini današjih aplikacij, od kam-rekorderjev do posebnih kamer za mikroskopijo ali astronomijo, najdemo CCD (*angl: charge-coupled-device*) kamere. CCD senzor je pravokotna matrika zbiralnikov fotonov položenih na tanko plast silicija, ki meri količino do vsakega zbiralnika dostopajoče svetlobe. Po nekem času osvetlitve nastane na senzorju slika, ki se paralelno vrstico za vrstico prekopira v serijski register. Ko je register poln (po prekopiranju ene vrstice), se prenesejo naboji v ojačevalnik, ki jih ojači in pošlje dalje. Ta postopek se ponavlja dokler ni celotna slika prebrana. Hitrost prenosa cele slike variira od 30 slik na sekundo (televizijska hitrost) za video aplikacije pa do precej počasnejših (tudi reda ur) za astronomske slike pri majhni količini svetlobe.

Barvne CCD kamere za širšo uporabo uporabljajo v osnovi enake senzorje kot črno bele, le da so zaporedne vrstice ali stolpci senzorjev občutljivi na rdečo, zeleno ali modro svetlobo. V rabi so sicer tudi drugi vzorci razporeditve barvnih senzorjev v matriki. Kamere višje kvalitete vsebujejo razdelilec žarkov, ki žarek razdeli v tri in vsakega pošlje svojemu CCD senzoju preko barvnih filtrov. Posamezni barvni kanali so potem diskretizirani posebej

(RGB izhod), lahko so sestavljeni v kompozitni video signal (NTSC izhod) ali pa kodirani v komponentni video format z ločeno informacijo o barvi in svetlosti.

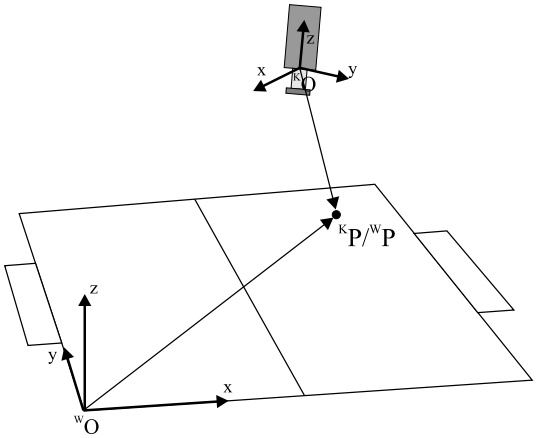


Slika 3.3: *Skica CCD senzorja.*

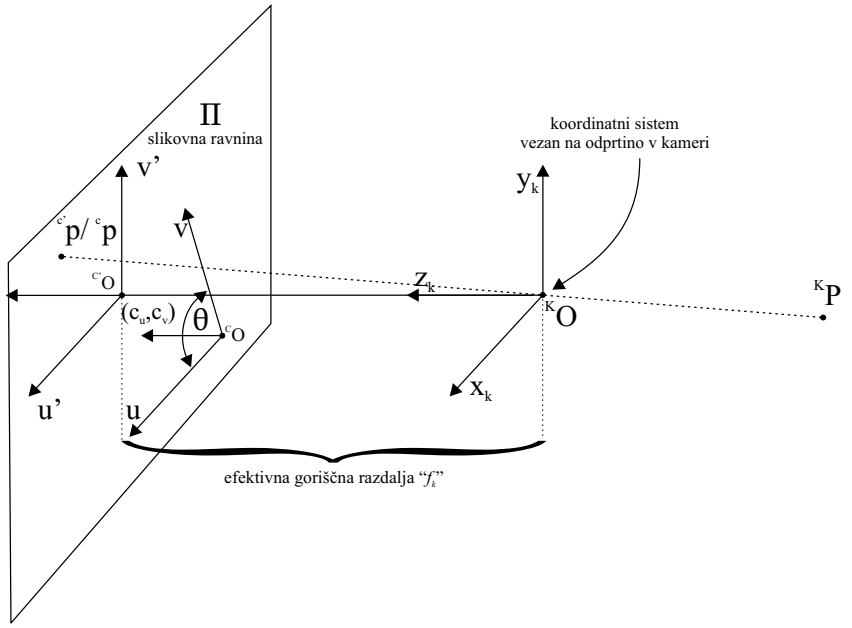
3.1.5 Nastanek slike preko kamere

Naj bo na kamero pripet koordinatni sistem ***K***, in ${}^{K}P({}^{K}x_p, {}^{K}y_p, {}^{K}z_p)$ točka v tem koordinatnem sistemu (slika 3.4 in 3.5). Ravnina Π naj ima koordinatni sistem slikovne ravnine ***C'*** in koordinatni sistem CCD matrike ***C*** (slika 3.5). Normala ravnine Π naj bo vzporedna z_k*-osi*. Če predpostavimo idealne leče brez distorzije, se bo točka ${}^{K}P$ preslikala na ravnino Π po enačbi perspektivne projekcije:

$$
\begin{aligned}
{}^{C'}u_p &= \frac{f_k . {}^{K}x_p}{{}^{K}z_p} \\
{}^{C'}v_p &= \frac{f_k . {}^{K}y_p}{{}^{K}z_p}
\end{aligned}
\qquad (3.4)
$$



Slika 3.4: *Pozicije kamere in koordinatnih sistemov*



Slika 3.5: *Koordinatni sistemi **K**, **C'** , **C*** in *ravnina Π, ki predstavlja slikovno ravnino v kameri.*

Ker imamo opravka z realnimi lečami, ki vnašajo popačenja, pa se ${}^{K}P({}^{K}x_p, {}^{K}y_p, {}^{K}z_p)$ preslikajo v ${}^{C'}P({}^{C'}u_p, {}^{C'}v_p)$ preko enačb (3.5) do (3.11).

$$u''_p = \frac{f_k \cdot {}^K x_p}{{}^K z_p} \tag{3.5}$$

$$v''_p = \frac{f_k \cdot {}^K y_p}{{}^K z_p} \tag{3.6}$$

$$\mathrm{r} = \mathrm{H} \cdot \ln\left(\frac{\mathrm{R}}{\mathrm{H}} + \sqrt{1 + \frac{R^2}{H^2}}\right), \tag{3.7}$$

$$\mathrm{R} = \sqrt{\mathrm{u''_p}^2 + v''^2_p}, \tag{3.8}$$

$$\varphi = arctg\left(\frac{v''_p}{\mathrm{u''_p}}\right), \tag{3.9}$$

$${}^{C'}u_p = r \cdot \cos(\varphi), \tag{3.10}$$

$${}^{C'}v_p = r \cdot \sin(\varphi). \tag{3.11}$$

Zaradi napak v proizvodnji kamere, pride še do ene popačitve slike. Zatorej slika izgleda, kot bi nanjo delovale strižne sile, ali drugače rečeno, kvadrat bi izgledal po tej transformaciji kot romb. To popačenje v [18] pripisujejo nepravokotni obliki senzorja, v drugi literaturi pa te distorzije ni zapaziti. Predpostavimo torej, da je to popačenje posledica nepravokotnosti matrike senzorja, in kot θ med osema u in v ni 90° (slika 3.5). Transformacijo točk iz pravokotnega koordinatnega sistema ***C'*** v točke nepravokotnega ***C*** opisuje transformacijska matrika (3.12),

$${}^{C}_{C'}T = \begin{bmatrix} 1 & 0 & 0 & c_u \\ 1 & \tan(\theta) & 0 & c_v \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

kjer sta c_u in c_v koordinati središča slike v koordinatah k.s. ***C***. Zaradi neenakosti v širini in višini zbiralnikov v CCD matriki, in neujemanja števila elementov po širini in višini senzorja z dejansko sliko, pride do popačitev. Transformacijska matrika (3.13) opisuje transform zaradi teh dejstev.

$$ {}^{slika}_{C}T = \begin{bmatrix} \frac{s_x}{d'_x} & 0 & 0 & 0 \\ 0 & \frac{1}{d_y} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13} $$

$$ d'_x = d_x \frac{N_{cx}}{N_{fx}} \tag{3.14} $$

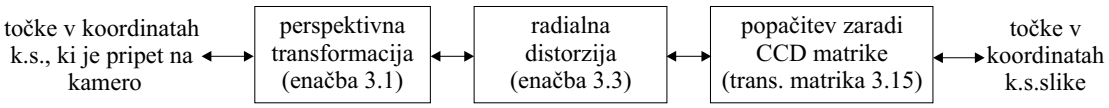
Pri čemer je d_x horizontalni razmik med centroma dveh zaporednih celic v senzorski matriki, d_y pa je vertikalni razmik med centroma dveh sosednjih celic v senzorski matriki. N_{cx} je število elementov v eni vrstici matrike senzorja, N_{fx} je število slikovnih elementov v eni vrstici matrike slike, s_x pa je horizontalni faktor skaliranja.

Preslikavo iz $\boldsymbol{C'}$ v koordinate slike dobimo, če združimo transformacijski matriki v enačbah 3.12 in 3.13 v skupno transformacijsko matriko ${}^{\text{slika}}_{C'}T$.

$$ {}^{\text{slika}}_{C'}T = {}^{slika}_{C}T \cdot {}^{C}_{C'}T = \begin{bmatrix} \frac{s_x}{d'_x} & 0 & 0 & c_u \frac{s_x}{d'_x} \\ \frac{1}{d_y} & \tan(\theta)\frac{1}{d_y} & 0 & c_v \frac{1}{d_y} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15} $$

Kjer sta sedaj $(c_u \frac{s_x}{d'_x}, c_v \frac{1}{d_y})$ koordinati središča slike v koordinatah slike.

Blokovno shemo transformacij od točke v na kamero fiksiranem koordinatnem sistemu do preslikane točke v koordinatni sistem slike, ki ima enako orientiran koordinatni sistem kot CCD matrika, prikazuje slika 3.6.



Slika 3.6: *Transformacije točk iz k.s. kamere v k.s. slike in obratno.*

3.1.6 Notranji parametri

Iskanje parametrov transformacij v prejšnjem podpoglavju, pomeni določiti notranje parametre. Med slednje spadajo:

- *Konstanta kamere*:

  efektivna goriščna razdalja, ali oddaljenost slikovne ravnine od optičnega središča

- *Središče slike*:

  izhodišče koordinatnega sistema slikovne ravnine (točka kjer optična os prebada slikovno ravnino) (*angl. principal point*)

- *Koeficienti distorzije leče*:

  parametri, ki vplivajo na spremembe koordinat v slikovni ravnini zaradi nepopolnih leč

- Skalirni faktor:

  parametri razdalj med stolpci in vrstami matrike CCD senzorja in relacija na dimenzije slike

Konstanto kamere lahko aproksimiramo z goriščno razdaljo, središče slike pa s centrom slike. Kljub temu, da je konstanta kamere blizu goriščne razdalje leče in je središče slike blizu centra slike, pa taki približki niso ustrezni v mnogih aplikacijah [19].

3.1.4 Zunanji parametri

Zunanji parametri se nanašajo na transformacijo točk iz svetovnih koordinat v koordinate k.s. kamere. Naj bo ${}^{W}P$ točka v svetovnih koordinatah in ${}^{K}P$ ista točka v koordinatah kamere (slika 3.4). Transformacija iz ${}^{W}P$ v ${}^{K}P$ sledi enačbi 3.14:

$$ {}^{K}P = {}^{K}_{W}T \cdot {}^{W}P \tag{3.16} $$

$$ {}^{K}_{W}T = (R \;\; t) \tag{3.17} $$

kjer je ${}^{K}_{W}T$ transformacijska matrika sestavljena iz rotacije $\boldsymbol{R}$ in translacije $\boldsymbol{t}$. Vektor $\boldsymbol{t}$ predstavlja koordinate izhodišča svetovnega k.s. v koordinatah k.s. kamere, $\boldsymbol{R}$ pa je rotacijska matrika iz svetovnega k.s. v k.s. kamere.

Rotacijska matrika $\boldsymbol{R}$ je sestavljena iz rotacij okoli posamičnih koordinatnih osi kamere.

$$ R = R_x \cdot R_y \cdot R_z, \tag{3.18} $$

$$ R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) & 0 \\ 0 & -\sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.19} $$

$$ R_y = \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.20} $$

$$R_z = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.21}$$

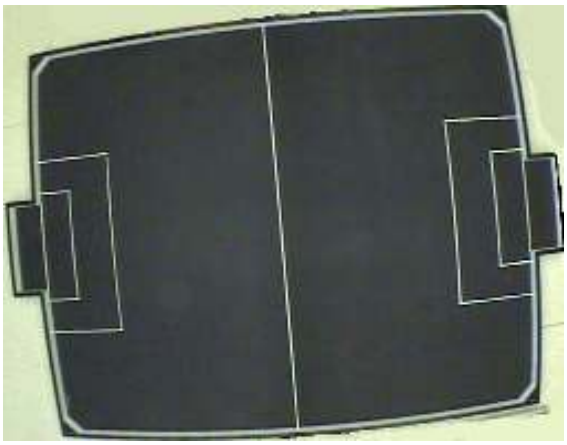
Kalibriranje zunanjih parametrov je torej iskanje toge preslikave, ki preslika svetovni koordinatni sistem v koordinatni sistem kamere.

3.2 Poenostavljen model kamere

Predstaavljeni model, je poenostavljen model iz poglavja 3.1, ki predpostavlja ekvivalentnost centra in središča slike in ne upošteva popačitev zaradi CCD senzorja.

Kamera, ki je iz sredine nekoliko izmaknjena, je nameščena približno 2 metra nad igralnim poljem. Ker želimo posneti sliko celotnega igrišča, je potrebno uporabiti širokokotno lečo, prav tako pa je potrebno kamero nekoliko nagniti, saj težimo k temu, da igrišče zajema kar največji delež slike. Tipično tako pridobljeno sliko prikazuje slika 3.7.

Model radialne distorzije je povzet po [17] in ga opisuje enačba (3.3), pri čemer smo središče slike aproksimirali z njenim centrom.



Slika 3.7: *Slika posnetega igrišča*

Ker je razdalja med kamero in igriščem velika v primerjavi z razdaljo med vrhom robotkov (ali žogice) in igriščem, lahko privzamemo predpostavko, da vsa scena leži v isti ravnini. Torej lahko uporabimo planarni model perspektivne projekcije upoštevajoč le dve bežiščni točki. Model perspektivne projekcije, ki smo ga uporabili, je povzet po [20]. Le-ta upošteva dve bežiščni točki in sledi enačbam (3.22, 3.23, 3.24 in 3.25) ,

$$x_p = \frac{x_n}{1 - q \cdot y_n - p \cdot x_n} \tag{3.22}$$

$$y_p = \frac{y_n}{1 - q \cdot y_n - p \cdot x_n} \tag{3.23}$$

$$p = \frac{1}{X_v} \tag{3.24}$$

$$q = \frac{1}{Y_v} \tag{3.25}$$

kjer sta (x_n,y_n) koordinati točke na posneti sliki, (x_p,y_p) koordinati iste točke na sliki, ki ima obe bežiščni točki v neskončnosti, X_v je razdalja od centra slike do bežiščne točke, ki leži na x osi, Y_p pa je razdalja od centra slike do bežiščne točke, ki leži na y osi. Ker zgornji model zahteva , da je vektor od centra slike do bežiščne točke v x smeri vzporeden z x osjo, vektor od centra slike do bežiščne točke v y smeri pa vzporeden z y osjo, je potrebno sliko

pred perspektivno transformacijo še zavrteti za kot θ (poravnamo k.s. igrišča in slike), da zadostimo temu pogoju.

Izhodišče koordinatnega sistema igrišča lahko določimo iz slike in je definirano v levem spodnjem kotu igrišča.

Na koncu je potrebno poznati še faktor razmerja med dimenzijami v sliki in dimenzijami v sceni. Privzamemo, da je faktor skaliranja Ω enak v x in y smeri.

Naj bo sedaj S slika širine W in višine H, faktor skaliranja naj bo Ω, ${}^{S}C({}^{S}c_x, {}^{S}c_y)$ center slike v koordinatah slike, ${}^{S}O({}^{S}o_x, {}^{S}o_y)$ izhodišče k.s. igrišča v koordinatah slike, točka ${}^{S}P({}^{S}x, {}^{S}y)$ naj bo neka točka v posneti sliki, ${}^{I}P({}^{I}x, {}^{I}y)$ ista točka v koordinatnem sistemu igrišča, θ pa kot zavrtenosti igrišča v sliki. Transformacija točke ${}^{S}P$ v ${}^{I}P$ sledi enačbam (3.26) do (3.39).

$$ {}^{S}c_x = \frac{W}{2} \tag{3.26} $$

$$ {}^{S}c_y = \frac{H}{2} \tag{3.27} $$

$$ {}^{S'}P = {}^{S}P - {}^{S}C \tag{3.28} $$

$$ r = \sqrt{{}^{S'}x^2 + {}^{S'}y^2} \tag{3.29} $$

$$ \varphi = \arctan\left(\frac{{}^{S'}y}{{}^{S'}x}\right) \tag{3.30} $$

$$ R = \frac{H}{2}\left(\frac{e^{\frac{2r}{H}} - 1}{e^{\frac{r}{H}}}\right) \tag{3.31} $$

$$ {}^{R}P \equiv ({}^{R}x, {}^{R}y) \tag{3.32} $$

$$ {}^{R}x = R \cdot \cos(\varphi) \tag{3.33} $$

$$ {}^{R}y = R \cdot \sin(\varphi) \tag{3.34} $$

$$ {}^{R'}P \equiv ({}^{R'}x, {}^{R'}y) \tag{3.35} $$

$$ {}^{R'}x = {}^{R}x \cdot \cos(\theta) - {}^{R}y \cdot \sin(\theta) \tag{3.36} $$

$$ {}^{R'}y = {}^{R}x \cdot \sin(\theta) + {}^{R}y \cdot \cos(\theta) \tag{3.37} $$

$$ {}^{I}x = \left(\frac{{}^{R'}x}{1 - {}^{R'}x \cdot {}^{R'}p - {}^{R'}y \cdot {}^{R'}q} + {}^{S}c_x - {}^{R'}o_x\right) \cdot \Omega \tag{3.38} $$

$$ {}^{I}y = \left(-\left(\frac{{}^{R'}y}{1 - {}^{R'}x \cdot {}^{R'}p - {}^{R'}y \cdot {}^{R'}q} + {}^{S}c_y\right) + {}^{R'}o_y\right) \cdot \Omega \tag{3.39} $$

${}^{R}P$ je točka v sliki z odpravljeno radialno distorzijo, ${}^{R'}P$ pa ista točka v sliki s poravnanim igriščem (slika 3.4 in slika 3.7).

Poglavje 4

Sledenje

S hitrim porastom zmogljivosti osebnih računalnikov so aplikacije sledenja postale dostopne širšemu krogu uporabnikov. Prav tako pa se je pojavilo veliko prispevkov v strokovni literaturi, ki se nanašajo na sledenje in analizo v športnih igrah ([4],[3],[5],[2]), nadzorovanje ([6],[7],[8]) in analizo gibanja.

V tem poglavju bomo podali metode obdelave in analize slik, ki jih uporablja naš sledilnik. Metodi odštevanja slik in klasifikacije barv bomo najprej predstavili nekoliko širše, nato pa se bomo odločili za specifične variante in razložili zakaj. Sledila bo predtavitev metode identifikacije robotkov z uporabo modela in nazadnje izračun položaja robotka.

4.1 Odštevanje slik

Iskanje področji na sliki kjer so nastopile spremembe je zelo priročno in pogosto uporabljeno orodje v aplikacijah sledenja [4], [5], [21], [22]. Najenostavnejši metodi sta odštevanje dveh zaporednih slik in odštevanje slike od neke referenčne slike. Kljub temu, da so metode odštevanja slik v osnovi zelo preproste, je njihova uporaba v kombinaciji z drugimi postopki še vedno zelo razširjena tudi v komercialnih sistemih [21]. Dejstvo pa je, da naj bi bile vse operacije nizko-nivojskega računalniškega vida podprte z neodvisnim visokonivojskim [23]. Skupni problem metodam odštevanja slik je visokofrekvenčno

šumenje, ki ga je možno odpraviti z ugodno nastavitvijo pragovne vrednosti razlike ali/in z dodatnim filtriranjem (npr.: filter velikosti povezanih področji [24], medianin filter [21], salt and pepper [4] itd.).

Slabi lastnosti prve metode (poleg zgoraj omenjenih) sta še, da v primeru počasi premikajočega objekta dobimo njegov delni obris, ki je lahko podvojen (odvisno od načina odštevanja) in da nekega objekta ne vidimo, če se ni premaknil v dveh zaporednih slikah, ki jih odštevamo.

Problem druge metode (poleg tistih, ki so skupne obema) je ta, da del objekta lahko izgine, če se le-ta pojavi na mestu, kjer na referenčni sliki obstaja njemu podobna barva. Prav tako je pomembna odločitev kako določiti referenčno sliko, kar pa je tudi odvisno od aplikacije. V sledilnikih igralcev v športnih igrah, kjer je kamera statična, se referenčna slika dobi preko časovne mediane slik celotne ali dela igre [21],[4]. Enak postopek pridobivanja referenčne slike je možen pri uporabi dinamične kamere, če predhodno mozaičimo sliko [5]. V naših razmerah je osvetlitev konstantna in imamo možnost pridobiti referenčno sliko, tako da pred tekmo, ko je osvetljava igrišča že nastavljena, posnamemo sliko igrišča.

Enačbo binarne izhodne slike, ki nastane preko odštevanja trenutne od referenčne, predstavlja enačba (4.1),

$$D(x,y,t) = f(S(x,y,t), R(x,y,t_0))$$
$$f(a,b) = \begin{cases} 1 & ; d(a,b) > p \\ 0 & ; \text{sicer} \end{cases} \qquad (4.1)$$

kjer je *D(x,y,t)* slikovni element odštete slike, *S(x,y,t)* slikovni element trenutne slike, *R(x,y,t)* slikovni element referenčne slike, *d(a,b)* funkcija razlike in *p* pragovna vrednost.

Glede na definicijo funkcije razlike dobimo različne rezultate odštevanja. Ker želimo dobiti kar se da hitro odštevanje in čim boljšo kvaliteto odštete slike, smo se odločili za

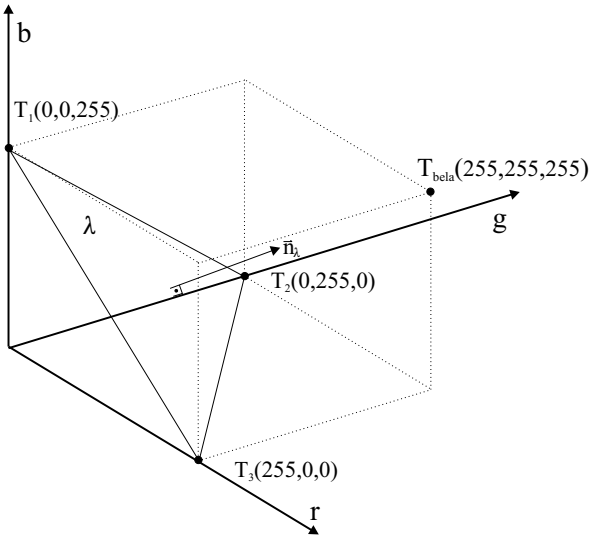
evklidsko razdaljo. Ker lahko v inicializaciji poljubno obdelamo referenčno sliko, jo lahko spremenimo v sivinsko kot uteženo povprečje vseh treh kanalov. Če vsako sliko sproti pretvorimo v sivinsko in tako računamo evklidsko razdaljo do referenčne le po eni dimenziji, bi to moralo rezultirati v večji hitrosti. Rezultati pokažejo slabšo kvaliteto odštete slike, kar bomo razložili v naslednjem podpoglavju.

4.1.1. Geometrijska interpretacija transformacije v sivinsko sliko

Sivinski nivo sivega slikovnega elementa izračunamo iz uteženega povprečja vseh treh barvnih kanalov po enačbi (4.2),

$$d_S = \mathrm{r}\cdot \mathrm{n}_\mathrm{r} + \mathrm{g}\cdot \mathrm{n}_\mathrm{g} + \mathrm{b}\cdot \mathrm{n}_\mathrm{b} = [r,g,b]\cdot \lfloor \mathrm{n}_\mathrm{r}, \mathrm{n}_\mathrm{g}, \mathrm{n}_\mathrm{b} \rfloor = \vec{t}_{RGB}\cdot \vec{n} \tag{4.2}$$

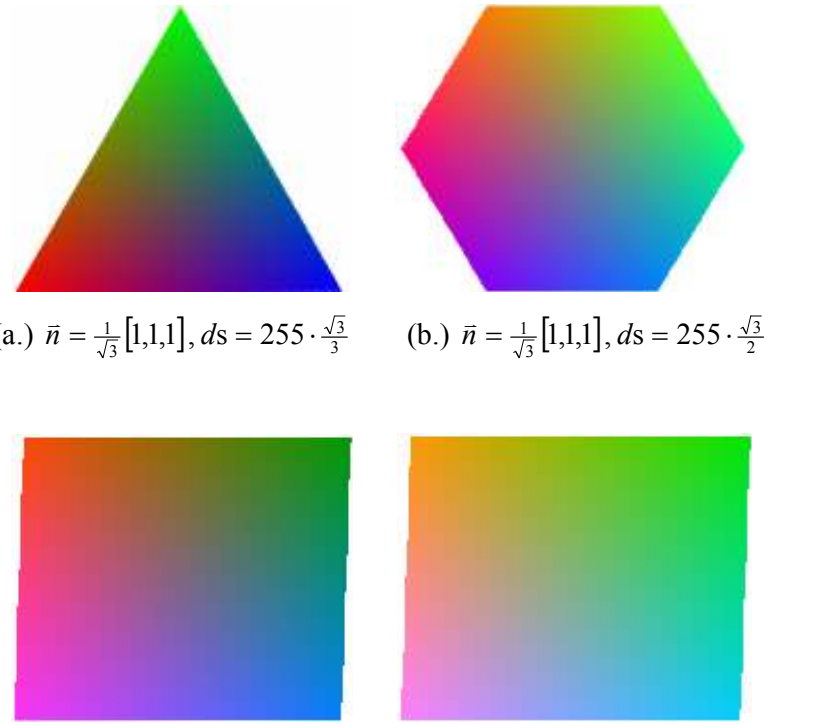
kjer je $\boldsymbol{n}$ enotski vektor uteži (njegova dolžina je 1), $\boldsymbol{t}_{RGB}$ koordinate barve v RGB kocki, d_S pa je izračunan sivinski nivo. Enačba 4.2 je ekvivalentna zapisu ravnine v RGB prostoru z normalo $\boldsymbol{n}$ in pravokotno oddaljenost ravnine od izhodišča (d_S). Torej velja, da se vse točke na isti ravnini s sivinsko preslikavo preslikajo v isto vrednost d_S. Naj obstajata ravnina $\boldsymbol{\varepsilon}$ s točko T_ε in ravnini $\boldsymbol{\varepsilon}$ vzporedna ravnina $\boldsymbol{\lambda}$ s točko T_λ. Točka T_λ se po sivinski transformaciji preslika v oddaljenost ravnine $\boldsymbol{\lambda}$ od izhodišča ($d_{S\lambda}$), medtem ko se točka T_ε po sivinski transformaciji preslika v oddaljenost ravnine $\boldsymbol{\varepsilon}$ od izhodišča ($d_{S\varepsilon}$). Evklidska razdalja med tako preslikanima točkama je razdalja med ravnino $\boldsymbol{\lambda}$ in $\boldsymbol{\varepsilon}$. Če torej točki ležita na isti ravnini ali na ravninah, ki sta si blizu, bo izračunana razdalja majhna, čeprav sta sicer v prostoru lahko precej oddaljeni.



Slika 4.1: *Kocka v RGB prostoru, z maksimalno vrednostjo 255 in minimalno 0.*

Sivinska preslikava je geometrijsko gledano preslikava iz 3D RGB prostora na 1D sivinsko premico v tem prostoru.

Zaradi zgornjih omejitev smo se odločili, pri odštevanju trenutne slike od referenčne, vzeti za funkcijo razdalje evklidsko razdaljo v tro razsežnem RGB prostoru.



Slika 4.2: *Barve na preseku RGB kocke z ravnino. Poleg vsake slike je napisana normala in oddaljenost ravnine od izhodišča. Vse barve, ki jih vidimo na slikah, se preslikajo v isto barvo.*

4.2 Razvrščanje barv

Razvrščanje ali klasifikacija pomeni razvrstiti objekte v naprej določene razrede na podlagi *m* spremenljivk. Vsak objekt je predstavljen z *m* meritvami za *m* spremenljivk in je ponavadi podan z meritvenim vektorjem

$$x_i = [x_{i1}, x_{i2}, ..., x_{im}]^T . \tag{4.3}$$

Če vsaka spremenljivka odgovarja neki osi v metričnem prostoru, potem meritveni vektorji x_i predstavljajo točke v tem *m*-dimenzionalnem prostoru ali prostoru značilk *(angl. feature space)*. V idealnem primeru različne populacije razredov zasedajo med seboj ločena področja v prostoru značilk, kar dovoljuje dobro razvrščevanje objektov glede na njihovo lego v prostoru. V praksi ponavadi pride do delnega prekrivanja dveh ali več razredov in zato 100% pravilna klasifikacija ni mogoča. Prav tako ponavadi porazdelitve populacij razredov niso znane in jih je potrebno oceniti iz *učnih vzorcev* (angl.: *training samples*). Celotno množico učnih vzorcev posameznega razreda imenujemo učna množica razreda. Glede na izrabo učnih vzorcev v smislu razvrščanja ločimo parametrične in neparametrične razvrščevalnike. Parametrične metode modelirajo razrede na predpostavkah porazdelitev dotičnega razreda, ki jih ocenimo iz učnih vzorcev (npr.: QDA, LDA). Neparametrične metode pa v nasprotju s parametričnimi razvrstijo nek objekt v določen razred na podlagi prisotnosti učnih vzorcev razredov v okolici objekta v prostoru značilk (npr.: k-najbližji sosedi, mean shift analysis [23]). Število spremenljivk v učnih vzorcih imenujemo dimenzionalnost *(angl.: dimensionality)* problema razvrščanja. Uspešnost razvrščanja je s povečevanjem dimenzionalnosti navzogor omejena. Prav tako je pomembno razmerje med dimenzionalnostjo in številom učnih vzorcev. Foley (D.Foley, 1972) in Kanal (L. Kanal, 1974) sta ugotovila, da mora za uspešno učenje razvrščevalnika, vsak razred vsebovati v lastni učni množici vsaj 3 do 5 krat več vzorcev kot je najmanjše število značilk v opisu te-teh [25].

4.2.1. Razvrščevalniki na podlagi najmanjše razdalje

Razvrščevalnik na podlagi najmanjše razdalje (*angl. minimum distance classifiers*) ali MD razvrščevalnik razvrsti meritve v razrede, tako da minimizira razdaljo med meritvami in razredi v prostoru značilk. Razdalja je definirana kot faktor podobnosti tako, da je minimalna razdalja ekvivalentna maksimalni podobnosti. Pri takih razvrščevalnikih so pogosto v rabi sledeče razdalje.

1. *Evklidska razdalja:*

$$d_k^2 = (x_i - \mu_k)^T (x_i - \mu_k) \tag{4.4}$$

kjer je x_i *i*-ti vzorec, μ_k srednja vrednost razreda *k*, d_k pa razdalja med *i*-tim vzorcem in *k*-tim razredom.

2. *Normirana Evklidska razdalja:*

$$d_k^2 = (x_i - \mu_k)^T \sigma_k^{-1} (x_i - \mu_k) \tag{4.5}$$

kjer je x_i *i*-ti vzorec, μ_k srednja vrednost razreda *k*, d_k razdalja med *i*-tim vzorcem in *k*-tim razredom, σ_k variančna matrika razreda *k*:

$$\sigma_k = \begin{bmatrix} \sigma_{11} & 0 & \dots & 0 \\ 0 & \sigma_{22} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{mm} \end{bmatrix}, \tag{4.6}$$

σ_{ii} pa so kvadrati standardnih devijacij (variance) za posamezno značilko razreda *k*.

3. *Mahalanobisova razdalja:*

$$d_k^2 = (x_i - \mu_k)^T S_k^{-1} (x_i - \mu_k) \tag{4.7}$$

kjer je x_i *i*-ti vzorec, μ_k srednja vrednost razreda *k*, d_k razdalja med *i*-tim vzorcem in *k*-tim razredom, S_k kovariančna matrika razreda *k*:

$$S_k = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1m} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{2m} \\ . & . & . & & . \\ . & . & & . & . \\ \sigma_{m1} & \sigma_{m2} & . & . & \sigma_{mm} \end{bmatrix}, \tag{4.8}$$

σ_{ij} pa so mešani produkti standardnih devijacij (kovariance) posameznih značilk razreda k.

4.2.2. Razvrščevalniki na podlagi največje podobnosti

Razvrščevalniki na podlagi največje podobnosti (*angl. maximum likehood classifiers*) ali ML razvrščevalniki so ena od najpopularnejših metod razvrščanja v daljinskem zaznavanju. Pri takem razvrščanju se vzorec razvrsti v razred k na podlagi posteriorne verjetnosti, da vzorec pripada določenemu razredu, oziroma, da je podobnost $L_k(x_i)$ največja:

$$k = \arg Max\{L_k(x_i)\} = \arg Max\{P(k \mid xi)\} \tag{4.9}$$

Definirajmo $L_k(x_i)$ kot posteriorna verjetnost za razred k pri danem vzorcu $X = x_i$:

$$L_k(x_i) = P(k \mid X = x_i) = \frac{P(k)p(x_i \mid k)}{p(x_i)}, \tag{4.10}$$

$$p(x_i) = \sum_{k=1}^{K} p(x_i \mid k) \tag{4.11}$$

kjer je:

$P(k)$ ... apriori verjetnost nastopa razreda k (*angl.: prior*)
$p(x_i \mid k)$ ... pogojna verjetnost da je x_i zares iz razreda k (*angl.: posterior*)
$p(x_i)$ ... verjetnost nastopa zorca x_i

Če privzamemo, da se vzorci razreda *k* porazdeljujejo normalno, potem lahko pišemo:

$$p(x_i \mid k) = \frac{1}{\sqrt{(2\pi)^m \cdot |S_k|}} e^{\left[-\frac{1}{2}Dm^2\right]}, \tag{4.12}$$

kjer je:

Dm ... Mahalanobisova razdalja
$|S_k|$ ... determinanta variančno-kovariančne matrike
m ... dimenzija prostora značilk

Če izraz 4.12 vstavimo v enačbo 4.10, in obe strani logaritmiramo z naravnim logaritmom, dobimo:

$$lL_k(x_i) = \ln(P(k)) - \tfrac{1}{2}\left|Dm^2\right| - \tfrac{m}{2}\ln(2\pi) - \tfrac{1}{2}\ln(|S_k|) - \ln(p(x_i)) \tag{4.13}$$

Tretji člen v izrazu 4.13 lahko zanemarimo, ker je vedno konstanten in ne vpliva na relativno bližino razredu *k*. Ker je peti člen konstanten za vse razrede pri danem vzorcu, prav tako ne vpliva na določitev *k* in ga tudi zanemarimo. Enačbo 4.13 lahko poljubno množimo z neko pozitivno konstanto in s tem ne vplivamo na *k*. Zaradi lepšega zapisa jo množimo z 2 in dobimo:

$$lL'_k(x_i) = 2\ln(P(k)) - \left|Dm^2\right| - \ln(|S_k|) \tag{4.14}$$

Prvi člen v enačbi 4.14 vsebuje priori verjetnost za nastop razreda P(k). Če predpostavimo enako verjetnost nastopa katerega koli razreda, lahko prvi člen tudi zanemarimo. Drugi člen v enačbi 4.14 vsebuje kvadrat Mahalanobisove razdalje, tretji pa determinanto

variančne-kovariančne matrike. Z večanjem raznolikosti znotraj nekega razreda, se torej manjša zaupanje s katerim lahko slednjemu dodeljujemo objekt.

Če v izrazu v enačbi 4.14 pomnožimo levo in desno stran z -1, dobimo obliko ki jo uporablja metoda QDA (*Quadratic Discriminant Analysis*):

$$c_k(x_i) = (x_i - \mu_k)^T S_k^{-1}(x_i - \mu_k) + \ln(|S_k|) - 2\ln(P(k)) \qquad (4.15)$$

Sedaj dodelimo vzorcu x_i tisti razred, ki minimizira vrednost $c_k(x_i)$. Problem takega razvrščevalnika je v tem, da je treba oceniti kovariančno matriko in vektor srednjih vrednosti. Oceni sta zelo slabi, kadar imamo opravka z malo vzorci. V takih primerih ocenimo vektor srednjih vrednosti za vsak razred posebej. Namesto lastnih kovariančnih matrik za vsak razred pa uporabimo skupno kovariančno matriko, ki je povprečna matrika vseh ostalih, kar včasih pripelje do boljše ocene, ker s tem zmanjšamo število parametrov, ki jih je potrebno oceniti [26]. To velja celo kadar so si prave kovariančne matrike razredov precej različne [27].

Če upoštevamo skupno kovariančno matriko in jo vstavimo v enačbo 4.15, dobimo enačbo, ki jo uporablja metoda LDA (Linear Discriminant Analysis). Ime izhaja iz posledice, da so sedaj meje med razredi linearne [28].

Metoda RDA (Regularized Discriminant Analysis) temelji na izboljšanju kovariančnih matrik, tako da se za vsak razred določi nova kovariančna matrika, ki je linearna kombinacija skupne in tiste, ki jo izračunamo za posamezen razred. V [26] so predlagali za oceno kovariančno-variančnih matrik linearne kombinacije lastne, skupne in diagonalni obeh.

Nekateri avtorji predlagajo izboljšanje razvrščevalnika na podlagi odločitev večih razvrščevalnikov [29]. Pri tem pristopu pa se pojavi vprašanje, ali je možno odločitve večih razvrščevalnikov kombinirati na tak način, da bo rezultat boljši kot pri le enem.

4.2.3. Izbira razvrščevalnikov

Razvrščevalnik mora biti robusten da dovolj dobro razloči okoli 15 barv tudi pri nekoliko neenakomerni osvetlitvi igrišča. Na podlagi prejšnjih podpoglavji smo se odločili za testiranje dveh razvrščevalnikov. Prvi je MD razvrščevalnik na podlagi evklidske razdalje, drugi pa je ML razvrščevalnik na podlagi metode LDA.

LDA razvrščevalnik se še dalje poenostavi, če predpostavimo enako verjetnost nastopa vsakega razreda, in lahko tretji člen v enačbi 4.15 zanemarimo. Ker je vsem razredom skupna kovariančna matrika, zanemarimo tudi drugi člen v tej enačbi in tako nam ostane le prvi člen, ki je kvadrat Mahalanobisove razdalje. Torej je naš drugi razvrščevalnik v bistvu MD razvrščevalnik na podlagi Mahalanobisove razdalje.

4.2.4. Vpeljava prevajalne tabele za razvrščevalnik

Za potrebe obdelave slike v realnem času, je čas obdelave slike z izbranima razvrščevalnikoma na trenutnih računalnikih nesprejemljiv. V ta namen se ob koncu določevanja barvnih razredov generira prevajalna tabela za razvrščanje barv. Ker je vhodna slika iz kamere kodirana v RGB prostoru, so dimenzije tabele 255x255x255, tako da vsaka točka v tabeli predstavlja barvo v RGB prostoru. Generiranje tabele poteka tako, da se za vsako točko (barvo) v tabeli določi razred kateremu pripada, in se na to mesto vpiše njegova zaporedna številka. Uporaba tabele je preprosta, saj za vsako barvo lahko takoj odčitamo kateremu razredu pripada. Poleg tega se lahko uporabi tudi računsko bolj

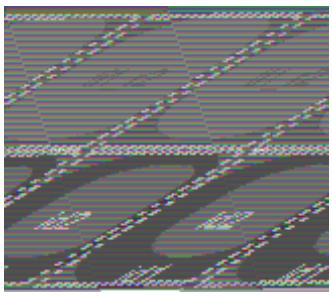
potraten in morda bolj natančen razvrščevalnik za generiranje tabele, ki temelji na poljubnem barvnem prostoru, sama hitrost razvrščanja pa pri tem ne bo spremenjena.

4.3 Identifikacija robotkov

V fazi identifikacije je potrebno vse detektirane packe oznak na sliki povezati ali zavreči, tako da dobimo robotke. Ker poznamo način označevanja igralcev našega tima, moramo tu zagotoviti kar se da robustno identifikacijo vsaj naših robotkov.

Identifikacija robotkov je precej otežena zaradi različnih vplivov kot so:

1. Podvojevanje in deformacija objektov zaradi velike hitrosti premikanja.
2. Slikovni elementi na robu oznak robotkov so podvrženi vplivu barve na katero mejijo (npr.: barva ozadja, barva pokrova robotka).
3. Na oznaki robotka se lahko pojavita dve barvi, ki sta si podobni (slika 4.4).
4. Če pride do stika dveh ali več robotkov z enakima barvama, se ti dve področji lahko zlijeta v eno.
5. Identifikacijska barva nekega robotka iz prvega tima, je lahko enaka identifikacijski barvi robotka v drugem timu.

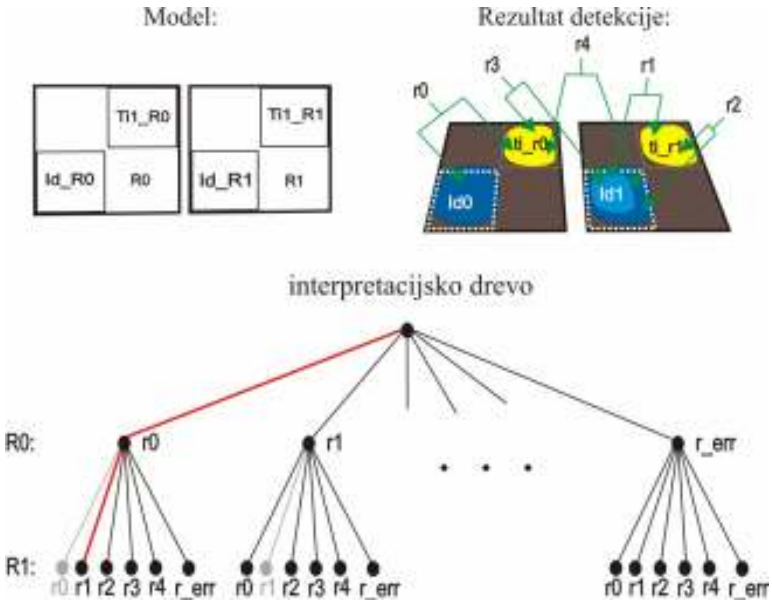


Slika 4.4: *Primer, ko detektiramo dve različni identifikacijski barvi namesto ene.*

Vhod v fazo identifikacije je množica detektiranih pack različnih velikosti, med katerimi so le nekatere packe dejansko deli robotkov, ostale pa predstavljajo šum zaradi zgoraj omenjenih vplivov 2, 3 in 4. Potrebno je uporabiti metodo, ki upošteva neko znanje o objektih zanimanja, in na podlagi tega poišče ustrezno kombinacijo med detektiranimi packami.

Popularna metoda simbolnega prilagajanja modelu v računalniškem vidu je metoda interpretacijskega drevesa (*angl. Interpretation Tree*), kjer vsak list predstavlja eno možno rešitev. Slaba lastnost algoritma klasičnega interpretacijskega drevesa je velika računska kompleksnost, zaradi česar je bilo razvitih mongo bolj ali manj uspešnih različic. Dober pregled enajstih izvedb podaja [30], kjer kot najboljšo med vsemi predlagajo tako imenovan »prvi-najboljši« (*angl. best-first*) algoritem.

Glavni prispevki h kompleksnosti interpretacijskih dreves so poti, ki vsebujejo pretežno jokerje (*angl. wild cards*) z eno ali dvemi napačnimi interpretacijami in preiskovanje že pregledanih poddreves. Že prej omenjeni prvi-najboljši algoritem v veliki meri reducira to odvečno preiskovanje, vendar smo se zaradi preprostosti izvedbe odločili za različico klasičnega interpretacijskega drevesa. Naša različica izbere prvo pot preko drevesa pri čemer upošteva možnost izgube elementov modela ali tako imenovane jokerje, toda le za tiste elemente modela katerim ne najde ustreznega para. Da je napaka interpretacije kar se da majhna, se podatki v listi možnih robotkov uredijo tako, da so najbolj verjetni robotki na začetku liste. Primer takega drevesa in interpretacije predstavlja slika 4.5, kjer model sestavljata *R0* in *R1*, listo možnih robotkov pa tvorijo *r0*, *r1*, *r2, r3* in *r4*.



Slika 4.5: *Primer izgradnje interpretacijskega drevesa. Model sta robotka R0 in R1, možni robotki pa so r0,r1,r2,r3 in r4. Zaradi večje preglednosti nismo izrisali celega drevesa, pač pa le dele, ki so bistveni za predstavo. S sivo so označeni deli poti, ki jih ne pregledujemo, z rdečo pa je označena pravilna pot skozi drevo.*

4.4 Določanje usmerjenosti robotkov

Oznake robotkov našega tima so postavljene kot prikazuje slika 4.7. Temna puščica na tej sliki prikazuje smerni vektor za vsakega robotka. Smerni vektor vedno poteka od identifikacijske barve proti timski. Dejansko orientacijo robotka dobimo, če kotu smernega vektorja prištejemo korekcijski kot φ_{corr}, ki je pri takih robotkih navadno $\pi/4$, v splošnem pa je poljuben. Položaj na igrišču opisujejo enačbe (4.16) do (4.19),

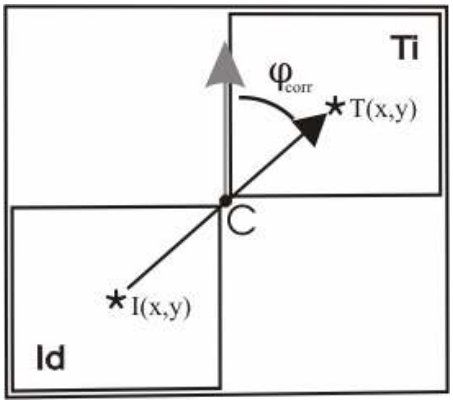
$$\theta = arctg(\Delta Y, \Delta X) + \varphi_{corr} \tag{4.16}$$

$$\Delta Y = T_y - I_y \tag{4.17}$$

$$\Delta X = T_x - I_x \tag{4.18}$$

$$C = \frac{(T + I)}{2} \tag{4.19}$$

kjer je θ usmerjenost robotka, T je težišče njegove timske packe, I težišče njegove identifikacijske packe, C pa pozicija robotka na igrišču.



Slika 4.7: *Usmerjenost robotka.*

Poglavje 5

Izvedba in rezultati

V tem poglavju bomo najprej predstavili in opisali izvedbo avtomatske kalibracije kamere z modelom v podpoglavju 3.2 in predstavili rezultate. V podpoglavju 5.2 bomo opisali princip sledilnika z uporabo metod opisanih v podpoglavju 4, predstavili njegovo izvedbo in opozorili na nekatere težave. V zadnjem podpoglavju 5.3, bomo predstavili rezultate segmentacije in sledenja.

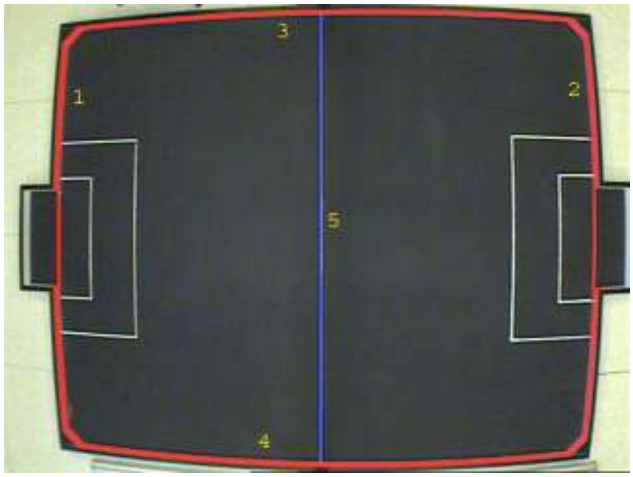
5.1 Izvedba kalibracije

Glede na model v podpoglavju 3.2 lahko razdelimo proces kalibracije v štiri glavne dele:

- Predobdelava slike
- Korekcija radialne distorzije (optimizacija parametra H v enačbi (3.31))
- Perspektivna transformacija (iskanje parametrov ${}^{R'}p$ in ${}^{R'}q$ v enačbi (3.38))
- Transformacija točk v koordinatni sistem igrišča

V literaturi je predstavljenih precej metod avtomatske kalibracije, pri čemer nekatere temeljijo na uporabi specialnih objektov s kontrolnimi točkami [15], [31], nekatere pa se nanašajo na krive (ali ravne) oblike v sliki [33], [34],[35],[36].

Za igrišče je (neodvisno od tipa) značilno pet glavnih črt (slika 5.1). Ker so te črte vedno bolj ali manj vidne, smo se odločili osnovati avtomatsko kalibracijo, ki temelji na le-teh. Prav tako je zelo pomembno dejstvo, da so si v pravokotnem ali vzporednem odnosu.



Slika 5.1: *Slika igrišča z označenimi najdaljšimi črtami. Rdeče so označene robne črte, z modro pa je označena sredinska*

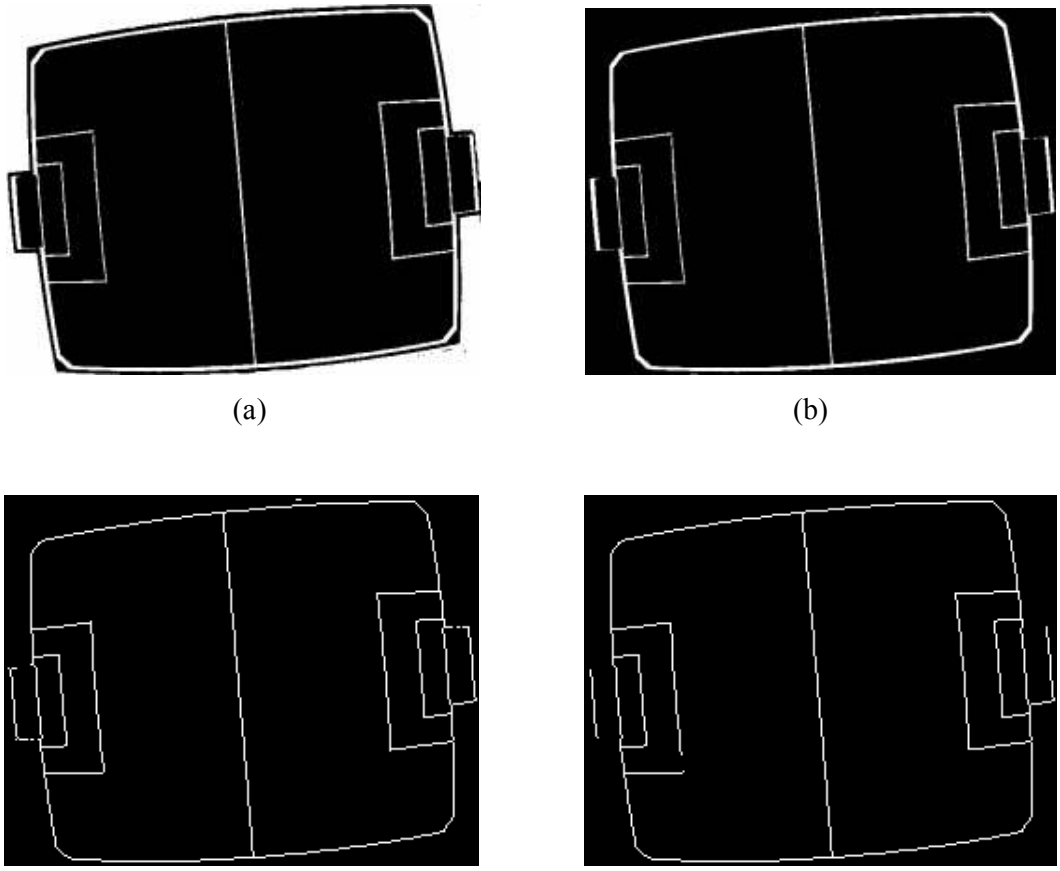
5.1.1 Predobdelava slike

Sliko igrišča je potrebno najprej upragovati, da dobimo dvonivojsko sliko. Pragovna vrednost se avtomatsko nastavi z entropijskim upragovanjem (podpoglavje 2.1).

Ker včasih na sliki niso vidne le črte igrišča, pač pa tudi okolica, ki je lahko svetla, je potrebno le-te detektirati in odstraniti. To izvedemo s preletom slike z oknom fiksne dimenzije (v našem primeru 5x5 slikovnih elementov). Na tistih mestih kjer so vsi elementi dvobitne slike znotraj okna »beli«, prožimo proceduro »flood fill« barvanja na »črno«.

V nadaljni obdelavi je potrebno sliko stanjšati na debelino enega slikovnega elementa in v ta namen uporabimo morfološko tanjšanje (podpoglavje 2.2). Ker imajo stanjšane krivulje

nekakšne izrastke in slika vsebuje še vedno nekaj šuma, zbrišemo vse nezaključene krivulje, katerih dolžina je manj kot neka v naprej določena vrednost. V našem primeru je ta vrednost 10 slikovnih elementov. Slika 5.2 prikazuje posamezne faze obdelave.



(a) (b)

(c) (d)

Slike 5.2: *Prikaz različnih faz predobdelave slike. Faze si sledijo: entropijsko upragovanje (a), izločanje ozadja (b), tanjšanje (c), filtriranje tanjšane slike (d)*

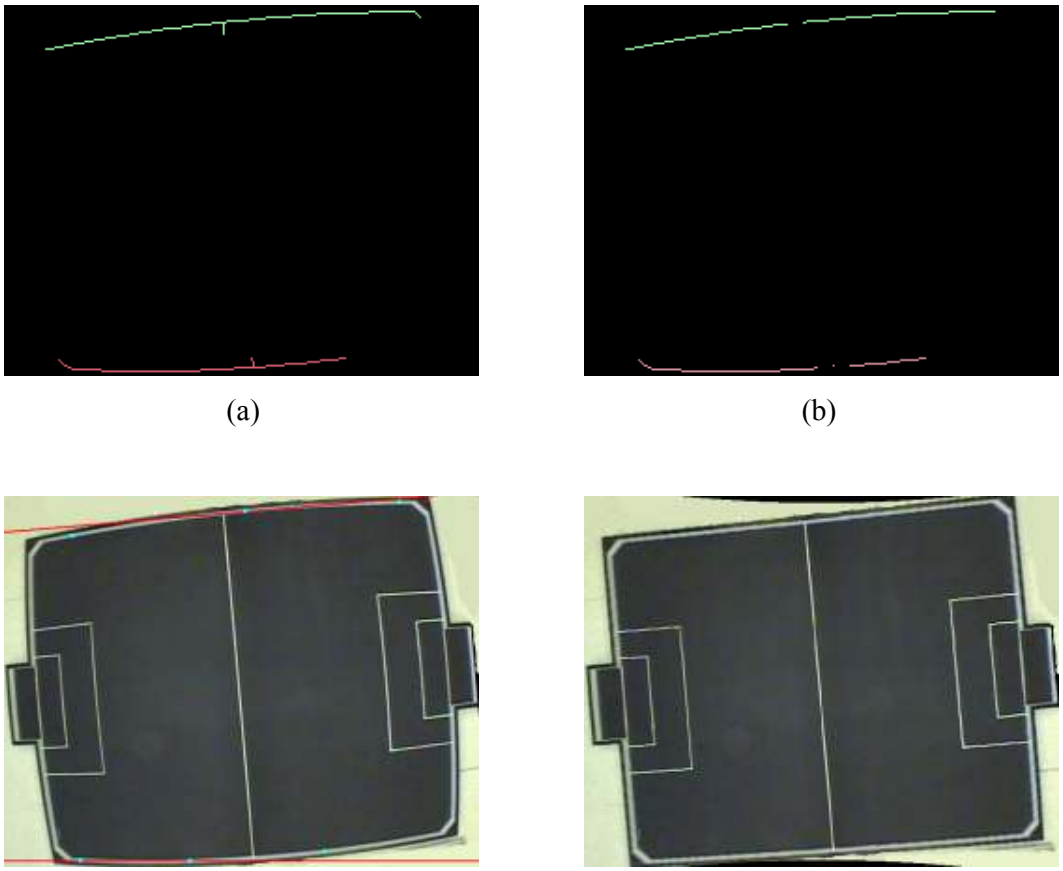
5.1.2 Avtomatska korekcija radialne distorzije

Slika, obdelana po postopku v prejšnjem poglavju, sedaj vsebuje črte debeline enega slikovnega elementa, ki so ukrivljene zaradi radialne distorzije. Na tej sliki izvedemo

Houghov transform (podpoglavje 2.3). Ker je večina črt ukrivljena, je potrebno za vsak dovolj velik maksimum v parametričnem prostoru izvesti inverzni HT z dovolj velikim oknom. V tako popravljenem parametričnem prostoru dva največja maksimuma pripadata črtama 3 in 4 (Slika 5.1). Z oknom predpisane velikosti se posnamejo vse točke stanjšane slike v okolici obeh detektiranih črt. Taki dve množici lahko še nekoliko filtriramo (zbrišemo točke v predpisani okolici presečišč detektiranih premic skozi črte 1,2,3 in 4). Postopek prikazujejo slike 5.3a-c. Iz obeh množic se izberejo po tri točke, ena na sredini množice in po dve, ki sta od krajišč oddaljeni za 10 procentov dolžine kruvulje v množici. Izvede se optimizacija parametra ukrivljenosti tako, da sta oba trojčka kolinearna. Slike 5.3a-c prikazujejo izločevanje množic točk za kalibracijo, slika 5.3d pa rezultat korekcije z modelom po enačbi 3.31.

5.1.3 Avtomatska detekcija perspektive

Vsi algoritmi detekcije bežiščnih točk v strokovni literaturi (ki ne temeljijo na posebnih kontrolnih točkah) se vedno opirajo na predpostavke, da je opazovana scena sestavljena iz pravokotnih in vzporednih oblik. Tako gre pri teh postopkih v glavnem za izločanje segmentov, ki so možni robovi takih struktur in določanje bežiščnih točk na podlagi usmerjenosti in dolžin teh segmentov (npr. [34]). Zelo zanimiva metoda detekcije bežičnih točk in horizontov je kaskadna Houghova transformacija [35]. Pri HT preslikavi slike v parametrični prostor, se vsaka črta preslika v točko. Če uporabimo zapis premic po enačbi (2.9), se dve točki v slikovnem prostoru preslikata v eno točko v parametričnem, prav tako pa se dve točki v parametričnem prostoru preslikata v točko v slikovnem prostoru. Z zaporednimi HT tako preklapljamo med *(x,y)* in *(k,n)* prostorom. Tako metodo lahko uporabimo za detekcijo bežiščnih točk, horizontov ,osi simetrij idt. [35]. Glavna težava kaskadnega HT se nanaša na zapis neomejenega *(k,n)* parametričnega prostora. Slednji problem je sicer lahko rešljiv na več načinov npr. z zapisom neomejenega prostora s tremi omejenimi podprostori [35].



(a) (b)

(c) (d)

Slike 5.3: *Prikaz določevanja kontrolnih točk(a-c) in končni rezultat optimizacije (d). Faze si sledijo: HT in izbira krivulj 3 in 4 (a), filtriranje krivulj (b), izbor treh točk iz vsake množice (c), in končni rezultat optimizacije (d).*

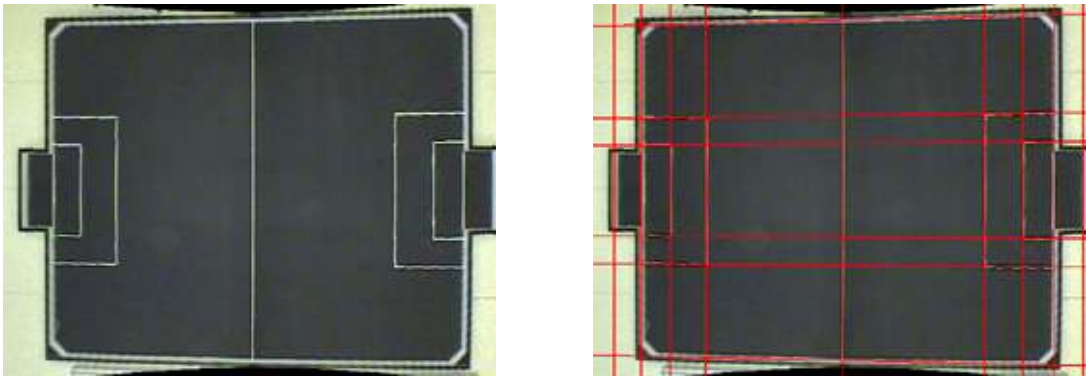
Ker vemo, da imamo opravka s sliko igrišča, se spet opremo na pet najdaljših črt na igrišču ki so v vzporednem ali pravokotnem odnosu (slika 5.1). Detekcija teh črt je dovolj za dobro oceno lege bežiščnih točk in zavrtenosti igrišča v sliki.

Detekcija bežiščnih točk tako razpade na:

- Problem iskanja petih največjih črt v sliki z odpravljeno radialno distorzijo.
- Iskanje presečišč in zapis slikovnega prostora za bežiščne točke.

- Izbira pravih bežiščnih točk med vsemi detektiranimi.

Iskanje največjih črt je enako kot v poglavju 5.1.1, z uporabo HT (slika 5.4b). Da detektirane premice še nekoliko izboljšamo, jih po metodi mediane najmanjših kvadratov (podpoglavje 2.4.2) prilegamo na točke v njihovi okolici, in šele nato iščemo potencialne bežiščne točke zgolj med njihovimi presečišči. Privzame se dejstvo, da bežiščna točka v našem primeru nikoli ne leži znotraj meja slike.



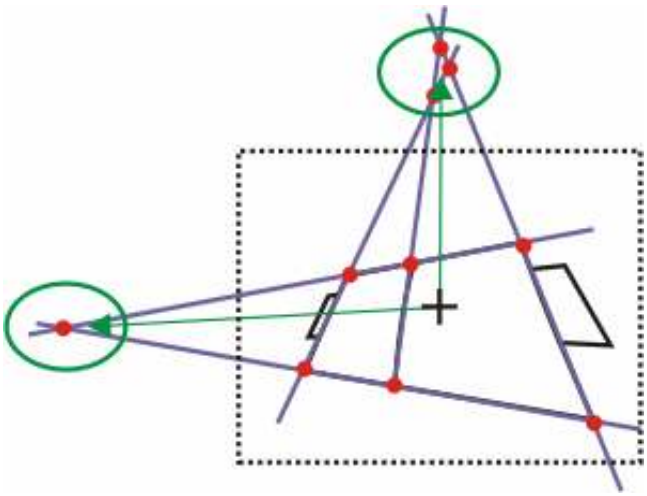
(a) (b)

Slika 5.4: *Slika igrišča z odpravljeno radialno distorzij (a) in rezultat detekcije bežiščnih premic, ki jih določajo segmenti igrišča (b).*

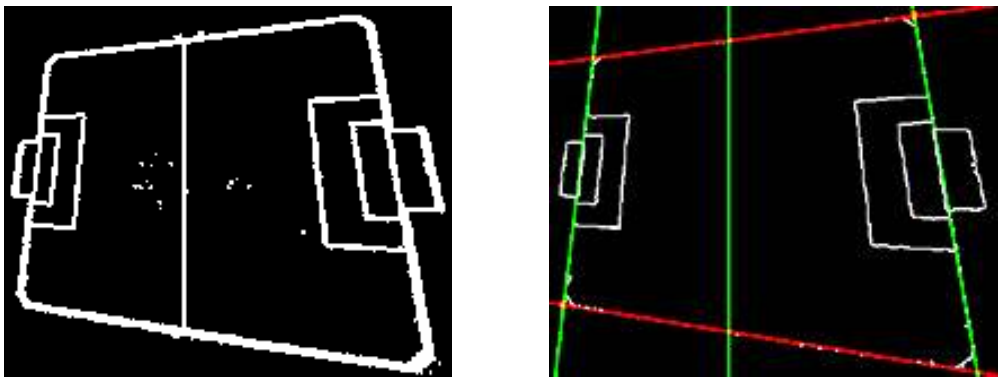
Za zapis prostora bežiščnih točk smo izbrali zapis v polarnih koordinatah, ki je tudi najbolj primeren za iskanje zavrtenosti igrišča. Vsaka možna bežiščna točka se zapiše s $\theta \in [0,2\pi]$ in $\rho \in [0,\inf]$. Neskončnost je definirana kot minimalni kot med dvema premicama, ki tvorita možno bežiščno točko.

Iskanje para bežiščnih točk je rešeno na zelo preprost način. Ker vemo, da bosta črti 3 in 4 tvorili eno bežiščno točko, imamo prvo že določeno. Vse kar preostane, je prvi bežiščni točki poiskati par s katerim tvori preko centra slike »najboljši pravi kot«. Povprečni kot med krajevnimi vektorji bežiščnih točk in koordinatnimi osmi koordinatnega sistema slike, z izhodiščem v njenem centru, da rotacijo igrišča v sliki. Slika 5.6 prikazuje primer

detekcije bežiščnih črt v precej pretiranih razmerah, slika 5.7b pa rezultat celotne avtomatske korekcije, vključno s korigiranjem radialne distorzije.



Slika 5.5: *Primer presečišč med petimi detektiranimi premicami. Možne bežiščne točke so označene z zelenima elipsama.*



(a) (b)

Slika 5.6: *Detekcija bežiščnih črt. Rdeči sta skupni eni, zelene pa drugi bežiščni točki.*

5.1.4 Transformacija točk v koordinatni sistem igrišča

Detektirati je potrebno spodnji levi in zgornji desni vogal igrišča. Ker smo na sliki že izvršili HT, lahko uporabimo presečišča detektiranih premic za oceno lege vogalov igrišča v sliki. Srednjo premico (glej sliko 5.6b) izločimo tako, da izmed petih detektiranih izločimo tisto, ki je najbližje centru slike, presečišča ostalih štiri pa tvorijo vogale igrišča. Ker smo kalibracijo izvedli na črtah, ki potekajo po notranjem robu igrišča, je potrebno detektirano igrišče razširiti za približno 1%, da dobimo zunanje meje igrišča. Z levo spodnjo in desno zgornjo točko po korekciji perspektive in razširitvi za 1% določimo še faktor pretvorbe v metre Ω. Levi spodnji kot igrišča po transformaciji predstavlja izhodišče koordinatnega sistema. Znani so notranji in zunanji parametri našega modela kamere.

5.1.5 Vrednotenje rezultatov kalibracije

Kalibracijski postopek upošteva zgolj radialno distorzijo in perspektivno projekcijo z le dvema bežiščnima točkama, središče slike pa aproksimira z njenim centrom. Za oceno kalibracije smo na igrišču ročno izbrali enaindvajset kontrolnih točk (slika 5.7) in izmerili odstopanja od njihovih predpisanih koordinat. Srednja kvadratna napaka znaša 1.9 *cm* (tabela 5.1), pri čemer je bila razdalja med dvemi zaporednimi slikovnimi elementi v levem zgornjem kotu igrišča na sliki ekvivalentna razdalji 1.5 *cm* na igrišču.

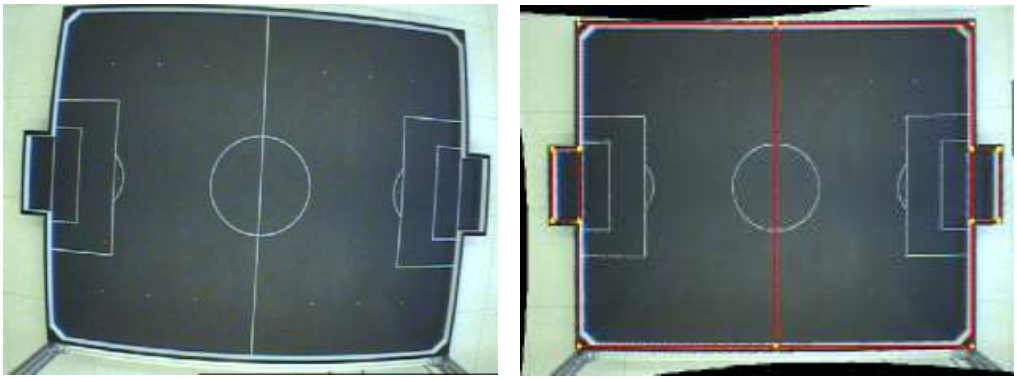
Kljub preprostosti model zadosti potrebam sledilnika in enote za vodenje. Slika 5.8a prikazuje vhodno sliko v program za kalibracijo, slika 5.8b pa popravljeno sliko z izrisanim modelom igrišča (rdeče). V primerih, ko igrišče ni dobro vidno, (če so npr. črte zakrite z reklamnimi napisi) lahko uporabnik kontrolne točke, ki se sicer detektirajo avtomatsko, določi ročno.

	predpisane koordinate		*izmerjene koordinate*		
	x [m]	*y [m]*	*x [m]*	*y [m]*	*napaka [m]*
t_1	0.35	1.3	0.354	1.285	0.015
t_2	0.35	0.5	0.359	0.503	0.010
t_3	0.15	1.15	0.165	1.142	0.017
t_4	0.15	0.65	0.162	0.656	0.014
t_5	1.1	1.15	1.094	1.136	0.014
t_6	1.1	0.65	1.097	0.652	0.004
t_7	0.375	0.9	0.384	0.893	0.012
t_8	2.05	1.15	2.028	1.149	0.021
t_9	2.05	0.65	2.019	0.662	0.033
t_{10}	1.85	1.3	1.825	1.291	0.026
t_{11}	1.85	0.5	1.825	0.512	0.028
t_{12}	1.825	0.9	1.803	0.900	0.022
t_{13}	0.3	1.5	0.307	1.483	0.019
t_{14}	0.55	1.5	0.553	1.479	0.021
t_{15}	0.8	1.5	0.799	1.477	0.023
t_{16}	0.3	0.3	0.312	0.311	0.017
t_{17}	0.55	0.3	0.558	0.306	0.011
t_{18}	0.8	0.3	0.805	0.305	0.008
t_{19}	1.4	1.5	1.388	1.482	0.021
t_{20}	1.65	1.5	1.634	1.482	0.024
t_{21}	1.85	1.5	1.879	1.489	0.031
t_{22}	1.4	0.3	1.390	0.311	0.015
t_{23}	1.65	0.3	1.631	0.313	0.023
t_{24}	1.85	0.3	1.874	0.314	0.029
			srednji kvadratni pogrešek		0.019

Tabela 5.1: *Napaka kalibracije.*

(a)

Slika 5.7: Izbira *kontrolnih točk za vrednotenje kalibracije.*



(a) (b)

Slika 5.8: *Slika nekalibrirane kamere (a) in ista slika po kalibraciji z napetim modelom igrišča (b).*

5.2 Sledilnik

V tem poglavju bomo predstavili izvedbo sledilnika, ki ga sestavljajo trije glavni deli: iskanje oznak robotkov, kombiniranje oznak v robotke in nazadnje izračun položaja robotkov na igrišču. Predstavili bomo tudi glavne težave in uspešnost sledilnika v praksi.

5.2.1 Izvedba segmentacije slike

Segmentacija slike je izvedena z razvrščevalnikom barv v kombinaciji z metodo odštevanja trenutne slike od slike ozadja (podpoglavje 4.1) na osnovi evklidske razdalje. Odštevanje trenutne slike od slike ozadja je zelo pomembno, saj pospeši segmentacijo in jo naredi bolj robustno in točno. Za razvrščanje barv smo testirali v podpoglavju 4.2.3 omenjena razvrščevalnika na principu Mahalanobisove in Evklidske razdalje.

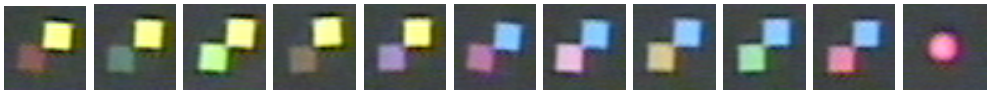
Za uspešnost razvrščanja barv z razvrščevalnikom na podlagi Mahalanobisove razdalje je kritičnega pomena ocena kovariančnih matrik. Izkazalo se je, da je segmentacija zelo uspešna, če uporabimo enako kovariančno matriko za vse razrede barv. Skupno kovariančno matriko smo najprej določili kot uteženo vsoto kovariančnih matrik vseh razredov, kar je rezultiralo v uspešni segmentaciji v večini primerov. Še boljše rezultate smo dobili, ko smo skupno kovariančno matriko izračunali iz barve igrišča, saj smo lahko izbrali zelo veliko število slikovnih elementov za tvorbo le-te.

Oba razvrščevalnika smo testirali na treh barvnih prostorih: RGB, HSV in Lab. Tabela 5.2 prikazuje po uspešnosti razvrščene razvrščevalnike. Najuspešnejša sta bila razvrščevalnik na osnovi Mahalanobisove razdalje v Lab prostoru (lab_M razvrščevalnik) in na osnovi Evklidske razdalje v Lab prostoru (lab_E razvrščevalnik). Oba razvrščevalnika sta približno enako dobro klasificirala barve, vendar je lab_E razvrščevanje rezultiralo v večjih barvnih zaplatah, kar včasih popači robotka. Glede na pravilnost razvrščanja barv, se je nekoliko slabše odrezal razvrščevalnik na osnovi Mahalanobisove razdalje v RGB prostoru

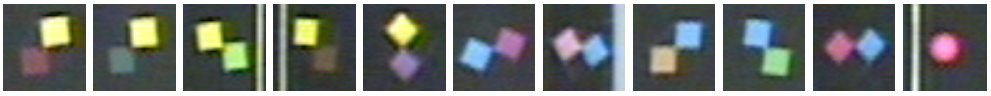
(rgb_M razvrščevalnik). Razvrščevalnika na principu Mahalanobisove in Evklidske razdalje sta se najslabše obnesla v HSV prostoru. Poudariti je treba, da smo testirali razvrščevalnike v precej zahtevnih pogojih, saj je bilo veliko barv zelo podobnih (Slika 5.10 a). Tako je razvrščevalnik v HSV prostoru na osnovi Evklidske razdalje deloval precej uspešno, vendar je izgubljal žogico, ker je bila zelo podobne barve kot ena od oznak robotkov.

najboljši po vrsti	način razvščanja	barvni prostor
1	*Mahalanobisova razdalja*	*Lab*
2	*Evklidska razdalja*	*Lab*
3	*Mahalanobisova razdalja*	*RGB*
4	*Evklidska razdalja*	*RGB*
5	*Evklidska razdalja*	*HSV*
6	*Mahalanobisova razdalja*	*HSV*

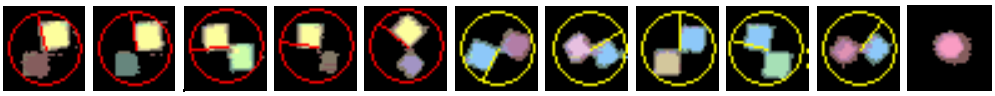
Tabela 5.2: *Uspešnost razvrščevalnikov v različnih barvnih prostorih. Najuspešnejši je razvrščevalnik na osnovi Mahalanobisove razdalje v Lab barvnem prostoru.*



(a)

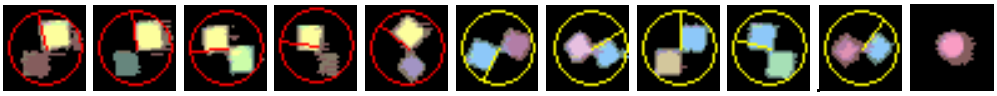


(b)



(c)

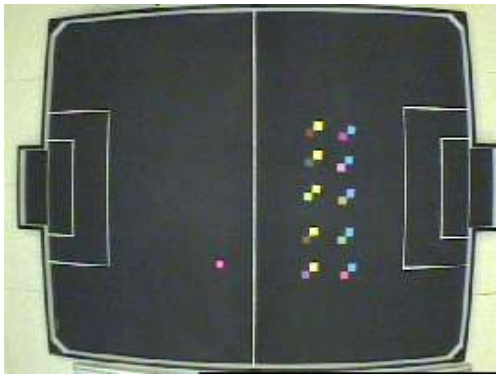
(d)



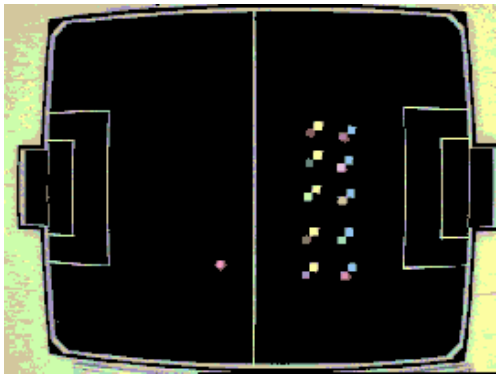
(e)

Slika 5.10: *Primerjava rezultatov treh najboljših razveščevalnikov: rgb_m (c), lab_e (d) in lab_m (e) . Vsi trije razvrščevalniki so bili inicializirani z enakimi podatki iz slike (a) in preizkušeni na sliki (b).*

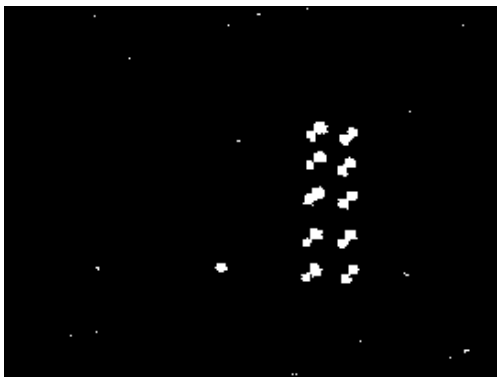
Slike 5.10 do 5.16 prikazujejo uspešnost razvrščanja barv po prvih treh najboljših razvrščevalnikih v tabeli 5.1. Nanašajoč se na zgornje ugotovitve in slike 5.10 do 5.16, smo kot najboljšo metodo razvrščanja barv glede na popačitev oznak izbrali lab_M razvrščevalnik, kar se tiče na velikosti zaplate pravilno razvrščene barve, pa lab_E razvrščevalnik .



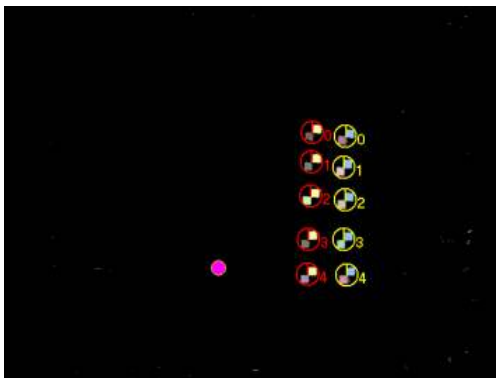
(a)



(b)



(c)

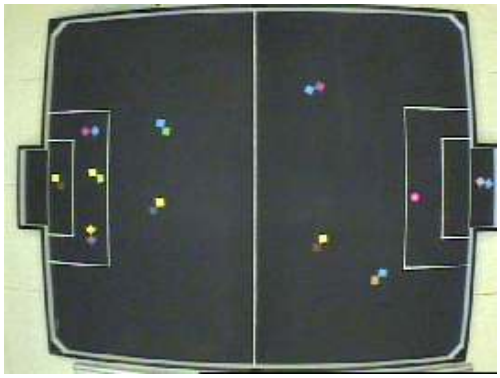


(d)

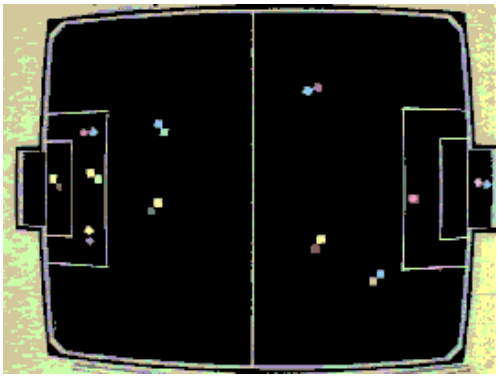
	robot st.0	robot st.1	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

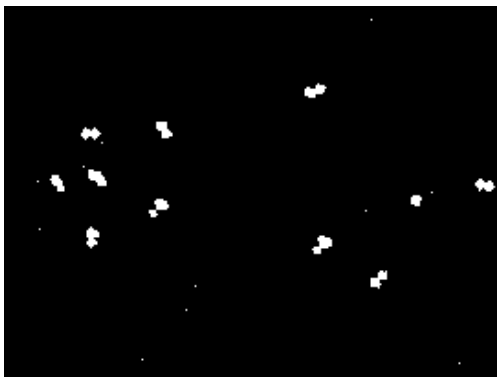
Slika 5.11: *Slika igrišča z robotki in žogico (a), segmentacija slike z rgb_M razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (d) z označenimi detektiranimi robotki in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki (a).*



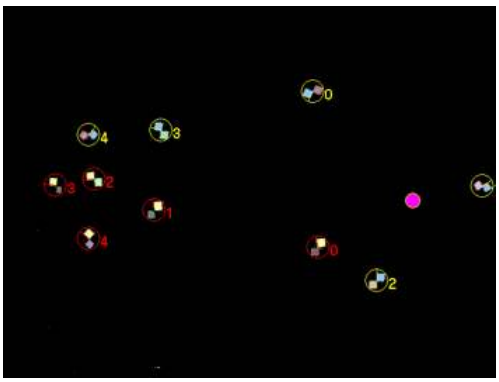
(a)



(b)



(c)

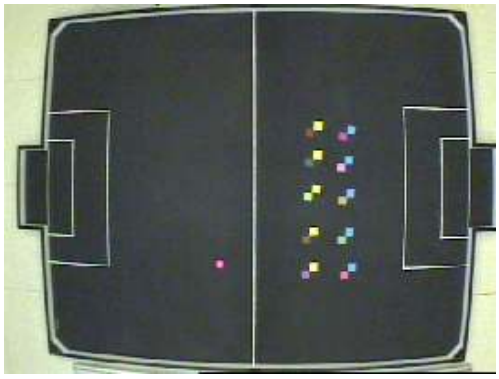


(d)

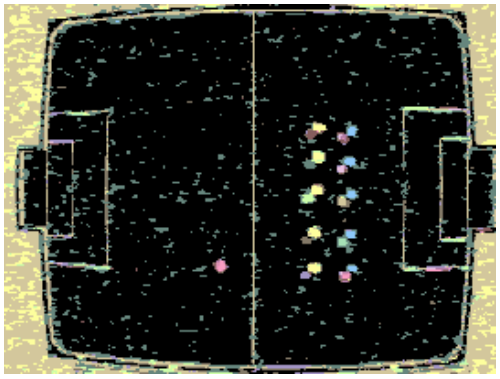
	robot st.0	robot st.1	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

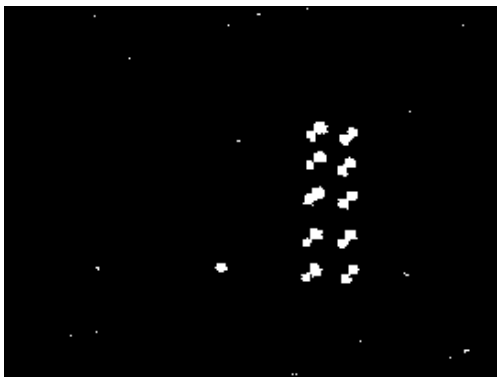
Slika 5.12: *Slika igrišča z robotki in žogico (a), segmentacija slike z rgb_M razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (d) z označenimi detektiranimi roboti in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki 5.11a.*



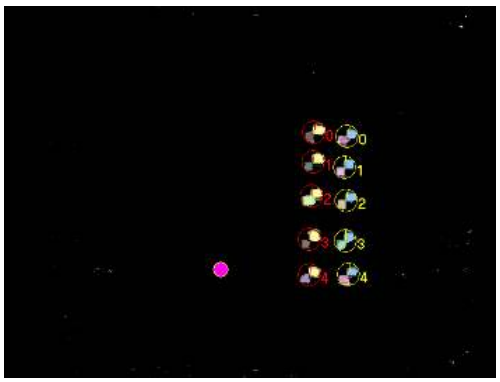
(a)



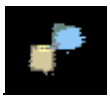
(b)



(c)

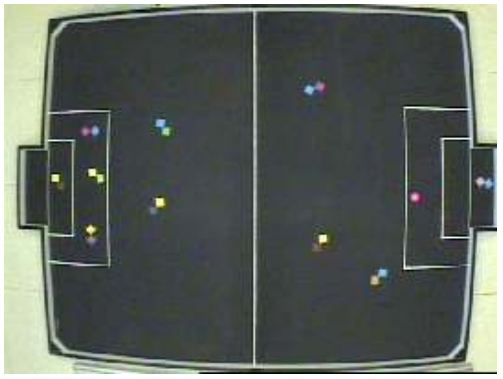


(d)

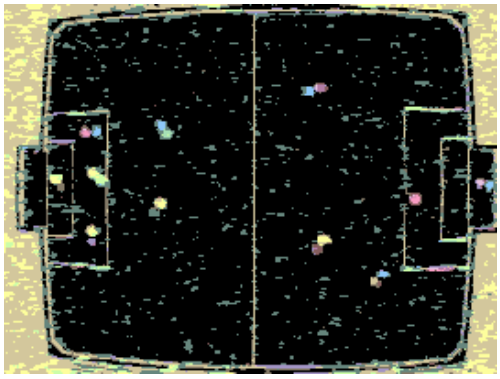
	robot st.0	robot st.1	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

Slika 5.13: *Slika igrišča z robotki in žogico (a), segmentacija slike z lab_E razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (d) z označenimi detektiranimi roboti in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki (a).*

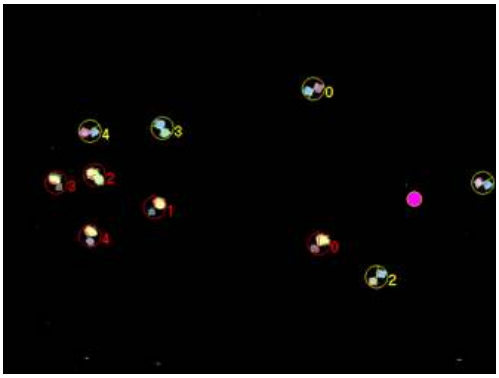


(a)



(b)

(c)

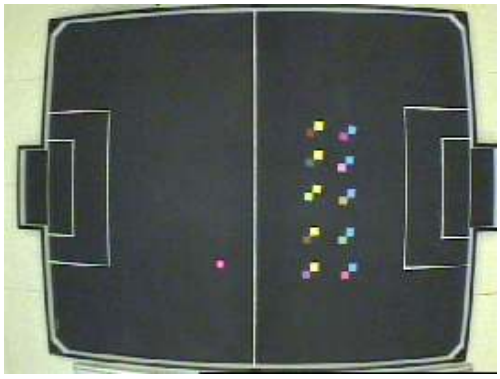


(d)

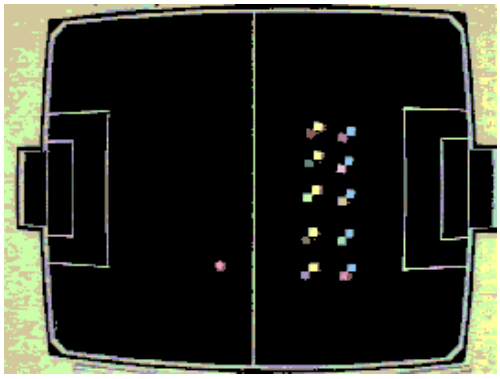
	robot st.0	robot st.1	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

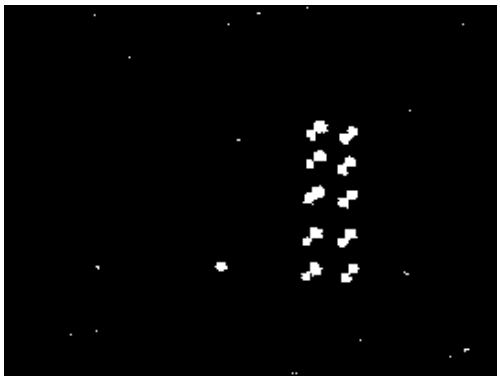
Slika 5.14: *Slika igrišča z robotki in žogico (a), segmentacija slike z lab_E razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (d) z označenimi detektiranimi roboti in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki 5.14a.*



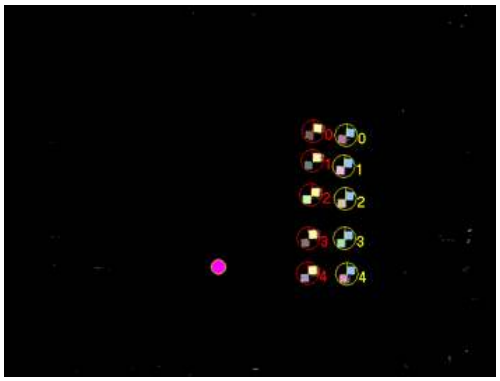
(a)



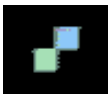
(b)



(c)

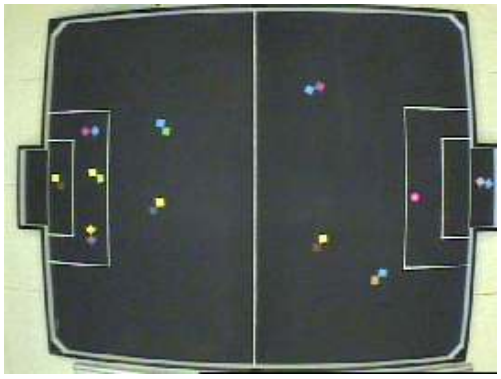


(d)

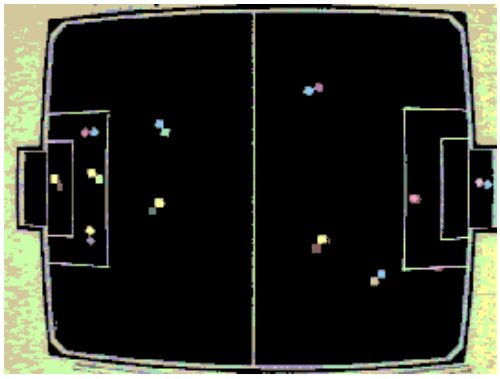
	robot st.0	robot st.0	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

Slika 5.15: *Slika igrišča z robotki in žogico (a), segmentacija slike z lab_M razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (d) z označenimi detektiranimi roboti in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki (a).*

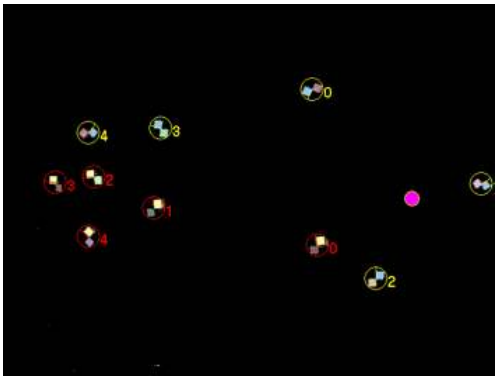


(a)



(b)

(c)



(d)

	robot st.0	robot st.1	robot st.2	robot st.3	robot st.4	zogica
tim 1						
tim 2						

(e)

Slika 5.16: *Slika igrišča z robotki in žogico (a), segmentacija slike z lab_M razvrščevalnikom (b), upragovljena slika razlike med trenutno in odšteto sliko (c), presek slik (c) in (b) z označenimi detektiranimi roboti in žogico (d), segmentirani roboti in žogica (e). Razvrščevalnik je bil inicializiran na sliki 5.16a.*

5.2.2 Izvedba iskanja oznak robotkov in žogice

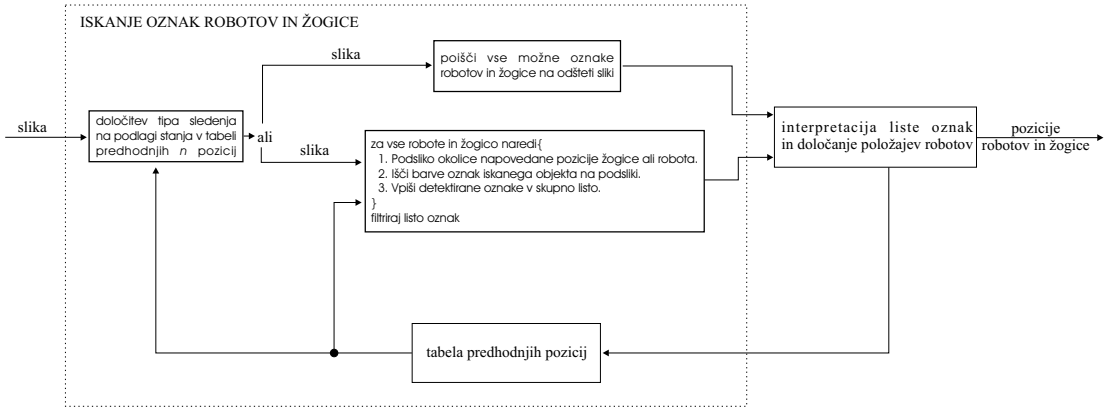
Sliko segmentiramo po postopku omenjenem v prejšnjem podpoglavju, in lista ustrezno velikih detektiranih možnih oznak gre dalje v proceduro za sestavljanje robotkov z interpretacijskim drevesom.

Da bi postopek detekcije robotkov pohitrili, smo implementirali podobno izvedbo kot pri [6], upoštevajoč predhodne pozicije sledenih objektov.

Za vsakega robotka in žogico je potrebno generirati tabelo predhodnih pozicij in na vsaki sliki iskati oznake v neki okolici točke, ki jo izračunamo na podlagi prejšnjih *n* pozicij. Okolica je definirana kot kvadrat znotraj katerega se v dveh zaporednih slikah robotek sigurno še nahaja. Stranico kvadrata ocenimo preko največje možne hitrosti robotka, ki je v našem primeru pri igri petih igralcev od 20 do 40 slikovnih elementov, kar sovpada z njegovo dvakratno in štirikratno velikostjo.

Recimo, da sledimo desetim robotkom in eni žogici. Celotna vhodna slika se tako razdeli v enajst podslik, ki se lahko tudi prekrivajo. Dimezije vsake podslike so odvisne od dimenzij objekta in uspešnosti njegove detekcije v predhodnji sliki. Na vsaki podsliki iščemo natanko dve barvi, če iščemo robotka, in le eno barvo, če iščemo žogico. Vse detektirane packe se vpišejo v skupno listo oznak. Ker se področja podslik lahko prekrivajo, je listo potrebno še nekoliko filtrirati, da izločimo podvojevanje detekcije iste oznake. Tako obdelana lista oznak se dalje prenese proceduri za sestavljanje robotkov.

V kolikor v prvi iteracji izgubimo iskani objekt, ga ponovno iščemo na istem mestu z dvakratno velikostjo okna. Če detekcija še vedno ni uspešna preklopimo na iskanje po celi sliki. Shemo iskanja oznak prikazuje slika 5.9.

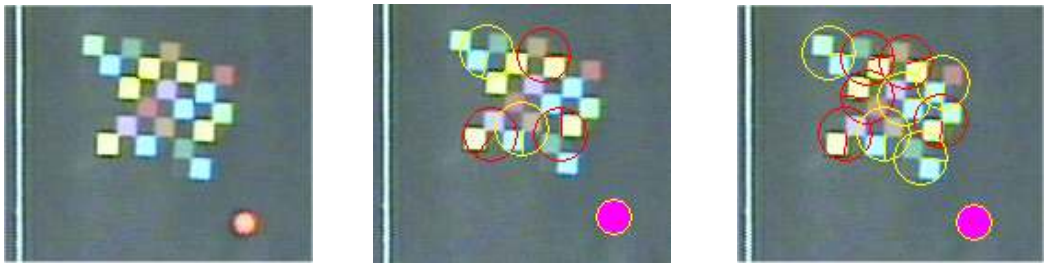


Slika 5.17: *Shema iskanja oznak objektov vsebuje dve rutini. Izbira rutine je odvisna od stanja v tabeli predhodnjih pozicij objektov.*

5.2.3 Uspešnost identifikacije robotkov v praksi

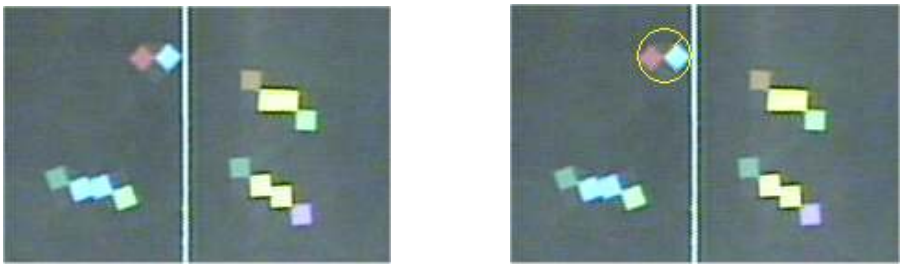
Med vsemi detektiranimi packami možnih oznak se generirajo možni robotki preko modela, ki upošteva timsko in identifikacijsko barvo in njuno medsebojno razdaljo. V množici možnih robotkov je potrebno poiskati tisto množico, ki dejansko predstavlja robotke. Kot metodo iskanja najboljše kombinacije detektiranih oznak, smo implementirali interpretacijsko drevo, ki popolnoma sledi opisu v podpoglavju 4.3. Tak način sestavljanja oznak v robotke se je izkazal za zelo uspešnega in zanesljivega, ko je razdalja med robotki dovolj velika. Zaradi svoje preprostosti je algoritem dovolj hiter in uspešen v veliki večini situacij, ki nastopijo v praksi. Če pa robotke zložimo v mozaik, kot na sliki 5.18a, algoritem odpove (slika 5.18b).

Če s klasičnim algoritmom interpretacijskega drevesa iščemo tisto pot, ki poleg največjega ujemanja z modelom poišče tudi največjo površino detektiranih robotkov, je metoda zelo uspešna, vendar hitro postane počasna za uporabo v realnem času (slika 5.18c).



(a) (b) (c)

Slika 5.18 : *Mozaik oznak robotkov dveh timov z enakimi identifikacijskimi barvami.*



(a) (b)

Slika 5.19 : *Primer zlitja oznak robotkov (a) in rezultat identifikacije (b)*

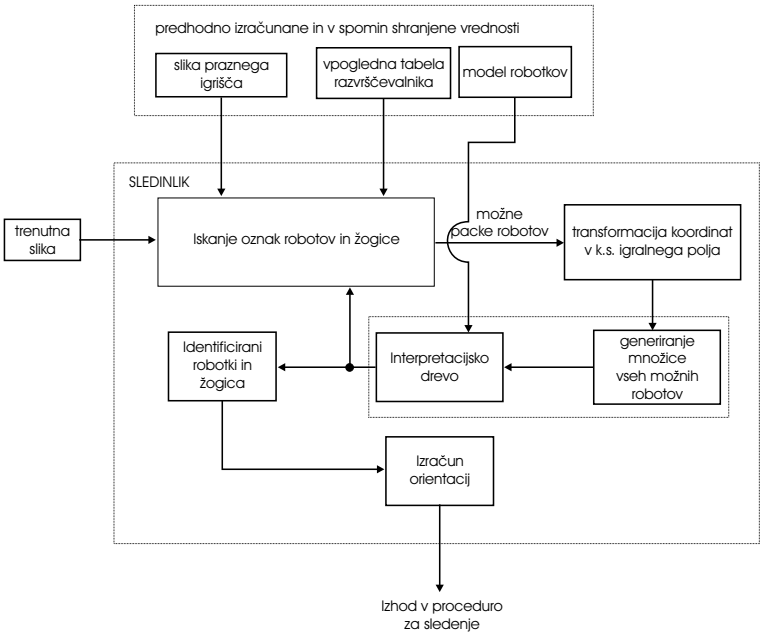
Kot smo omenili že v poglavju 4.3, nastopi problem detekcije v primerih, ki jih prikazuje slika 5.19. Če se dva ali več fizično ločenih oznak enakih barv stakne, se zlijejo v eno veliko področje in v takih primerih identifikacija odpove. Ker v resnici trki navadno trajajo zelo malo časa, poleg tega pa do situacij na sliki 5.12 ne prihaja pogosto, se nismo posvetili reševanju tega problema.

5.2.4 Izvedba sledilnika

V prejšnjih podpoglavjih smo posebej opisali postopke, ki jih uporablja naš sledilnik. Celotno zaporedje metod v sledilniku prikazuje diagram poteka na sliki 5.20. Slika praznega igrišča mora biti shranjena v inicializaciji, prav tako pa morajo biti določena

»imena« robotkov, žogice (vpisno mesto v pomnilniku za vsakega robotka in žogico posebej), modeli robotkov in model timov.

Trenutna slika, slika praznega igrišča in tabela predhodnjih *n* pozicij objektov so vhodi v sledilnik. Na podlagi stanj v tabeli preteklih pozicij oznak se določi metoda iskanja le-teh (podpoglavje 5.2.1). Z metodo označevanja komponent in barvne segmentacije z lab_M razvrščevalnikom, se določijo vse potencialne packe robotkov na sliki, med katerimi se izberejo le tiste, ki po velikosti ustrezajo dimenzijam oznak. Tako detektirane packe vstopijo funkcijo, ki transformira njihove koordinate v koordinatni sistem igrišča. Preko znanja o razdaljah med oznakami na enem robotku, se generira množica vseh možnih robotkov. Iz te množice se preko interpretacijskega drevesa izbere ustrezna množica robotkov. Za žogico določimo tisto packo, ki je največja packa z barvo žogice in je od centrov vseh detektiranih robotkov dovolj oddaljena. Nazadnje se izračunajo še orientacije vsakega robotka. Podatki o poziciji in orientaciji se vpišejo v tabelo preteklih pozicij objektov in podajo proceduri za vodenje robotkov.



Slika 5.20: *Potek detekcije in identifikacije robotkov*

5.3 Rezultati sledenja

Sledilnik je bil napisan v programskem jeziku C++ Borland Builder 5.0 in testiran na osebnem računalniku (1.7 *GHz*, 255 *Mb* dinamičnega pomnilnika, zajemalnik slike Matrox II s frekvenco vzorčenja 30 *Hz*) z operacijskim sistemom Windows NT/2000.

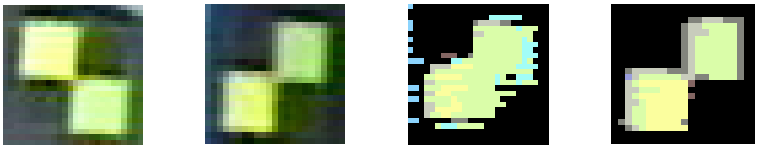
Testi so bili izvršeni za igro treh in petih robotkov, kjer je bila velikost vhodnih slik 640 x 480 slikovnih elementov. Za razvrščanje barv smo preizkusili prva dva najboljša razvrščevalnika iz podpoglavja 5.2.1 (lab_M in lab_E), pri tem smo kovariančno matriko lab_M razvrščevalnika ocenili iz barve igrišča. Sestavljanje detektiranih oznak v robotke je bilo izvedeno s preprostim interpretacijskim drevesom iz poglavja 4.3.

5.3.1 Barve oznak robotkov in žogice

Barve oznak morajo biti med seboj dovolj različne, da jih sledilnik razloči, tudi ko robotek pelje skozi različno osvetljene dele igrišča. Če dovolj različnih barv ni mogoče najti, imata lahko oba tima nekatere ali vse identifikacijske barve enake, kar naredi detekcijo bolj robustno, vendar se poveča verjetnost nastopa problemov iz podpoglavja 5.2.3. Predvsem je pomembna izbira takih parov oznak, da timska barva ni preveč podobna identifikacijski (slika 5.21a), prav tako pa si dve različni identifikacijski barvi ne smeta biti preveč podobni (slika 5.22). Slika 5.21c prikazuje slabo razvrščanje razvrščevalnika lab_E, ker sta si identifikacijska in timska barva preveč podobni, medtem ko na sliki 5.21d vidimo uspešno razvrščanje z uporabo lab_M razvrščevalnika.

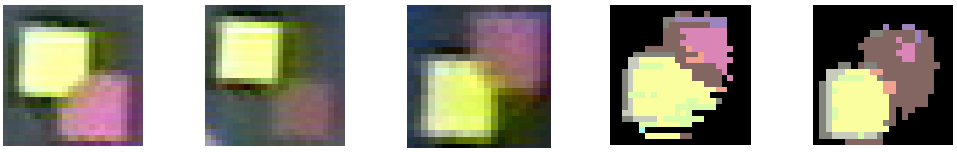
Za lažjo izbiro med lab_E in lab_M razvrščevalnikom, smo oba testirali na enakih učnih podatkih. Kot testno okolje smo izbrali veliko igrišče, barve oznak pa smo izbrali nekoliko kritične (slika 5.23). Razvrščevalnik smo naučili na sredini igrišča, nato pa postavili robotke še v šest ostalih pozicij (slika 5.24) in segmentirali slike.

Rezultati razvrščanja (slika 5.25 in 5.26) so pokazali, da je kljub večjemu šumenju izbira lab_E razvrščevalnika ustreznejša, kadar segmentiramo večje število podobnih barv na velikem igrišču. Kadar segmentiramo barve na malem igrišču pa je izbira lab_M razvrščevalnika bolj ustrezna, saj so barvne zaplate oznak na sliki dovolj velike, poleg tega pa je popačitev objektov zaradi lab_M razvrščevalnika manjša kot pri lab_E.



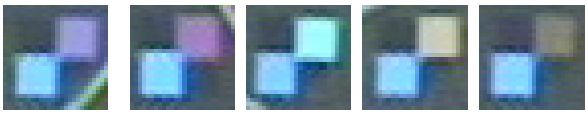
(a) (b) (c) (d)

Slika 5.21: *Slaba kombinacija timske in identifikacijske barve. Na slikah vidimo robotek v učni poziciji na sredini igrišča (a), isti robotek v kotu igrišča (b), segmentiran robotek v kotu z lab_E razvrščevalnikom (c) in segmentriran isti robotek z lab_M razvrščevalnikom.*

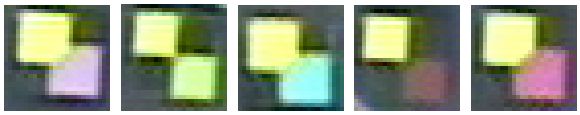


(a) (b) (c) (d) (e)

Slika 5.22: *Primer pomešanja dveh barv. Na slikah vidimo vijolično identifikacijsko barvo v učni poziciji na sredini igrišča (a), rdečo identifikacijsko barvo v učni poziciji na sredini igrišča (b), vijolično identifikacijsko barvo v kotu igrišča (c), pravilno segmentirano vijolično barvo z lab_E razvrščevalnikom (d), in neustrezno segmentacijo z lab_M razvrščevalnikom (e). Vijolična barva robotka na sliki (c) je dejansko precej podobna barvi v sliki (b).*

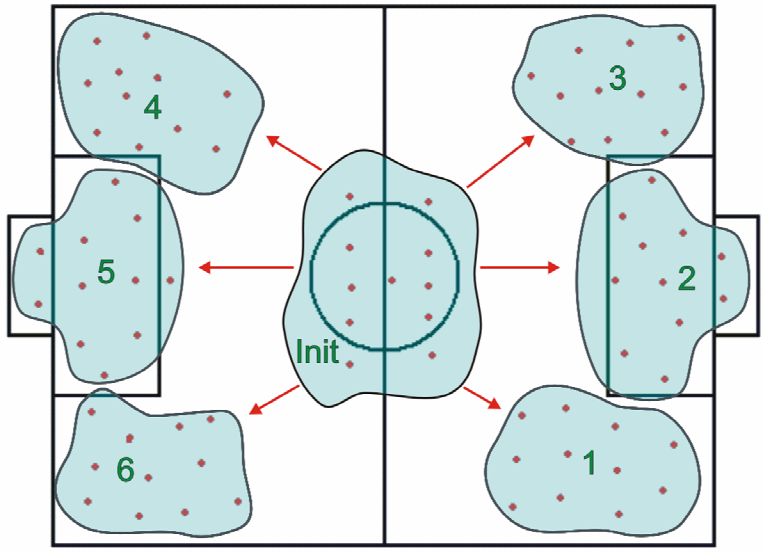


(a)



(b)

Slika 5.23: *Oznake robotov za testiranje razvrščevalnikov. Iz slike (b) vidimo, da se timske oznake razlikujejo v obliki in velikosti.*



Slika 5.24: *Postavitve robotov na igrišču v inicializaciji in območjih od 1 do 6.*

5.3.2 Natančnost ocenjevanja položaja igralcev in žoge

Če na igrišču snemamo statičnega robotka, bo zaradi šumenja v barvi vsaka zaporedna slika nekoliko drugačna. Po segmentaciji teh slik bodo barvne zaplate na robotku različne po velikosti in obliki za vsako sliko posebej. Šumenje v identifikaciji barve je precej odvisno od izbire barv oznak, enakomernosti osvetljave igrišča, položaja robotkov na igrišču in razvrščevalnika barv. Naš sledilnik detektira po dve barvi na robotku, iz katerih oceni njegov položaj na igrišču. Določanje natančnosti ocene položaja smo izvedli ločeno za veliko in malo igrišče. Robotke smo inicializirali na sredini igrišča, nato pa jih postavili v sedem položajev, kot prikazuje slika 5.24. V vsakem položaju smo jih snemali približno pol minute in podatke zabeležili. Za vsako področje posebej se je izračunal srednji kvadratni pogrešek položaja, povprečni kvadratni pogrešek pa kot povprečje vseh sedmih. Tabela 5.2 prikazuje pogreške pri oceni centra in kota robotkov na velikem in malem igrišču.

	malo igrišče	*veliko igrišče*
povprečni srednji kvadratni pogrešek centra robota [pixel]	0.256	0.313
povprečni srednji kvadratni pogrešek kota robota [deg]	2.086	1.942
povprečni srednji kvadratni pogrešek centra žogice [pixel]	0.417	0.378

Tabela 5.3: *Napaka pri ocenjevanju položaja robotkov v sliki.*

5.3.3 Hitrost obdelave podatkov

Hitrost segmentacije slike in identifikacije robotkov in žogice je ključnega pomena za dobro vodenje. Trenutna frekvenca nastopa vhodnih podatkov v proceduro za vodenje je 30 Hz. To pomeni, da mora sledilnik od nastopa vhodne slike končati obdelavo in določiti

podatke o položajih v približno tretjini sekunde. Delovati mora torej z višjo frekvenco kot 30Hz. Kot smo že omenili v poglavju 5.2.2 lahko sledilnik išče objekte preko celotne slike (globalno iskanje) in preko manjših področji (lokalno iskanje). Tabela 5.4 prikazuje rezultat hitrosti obdelave slike v globalnem in segmentnem načinu pri igri petih igralcev, kjer sledimo desetim robotkom in žogici, ter pri igri treh robotkov, kjer sledimo šestim robotkom in žogici. Prav tako je iz tabele razvidno, da celo globalno iskanje pri igri petih robotkov dosega hitrost obdelave v realnem času, vendar je hitrost zelo blizu mejni.

	globalno iskanje	*segmentno iskanje okno: 20 pixel*	*segmentno iskanje okno: 40 pixel*
igra treh	23.5 ms (~42 Hz)	premajhno okno	8.1 ms (~123 Hz)
igra petih	28.1 ms (~35 Hz)	4.2 ms (~241 Hz)	16 ms (~42 Hz)

Tabela 5.4: *Hitrost sledenja.*

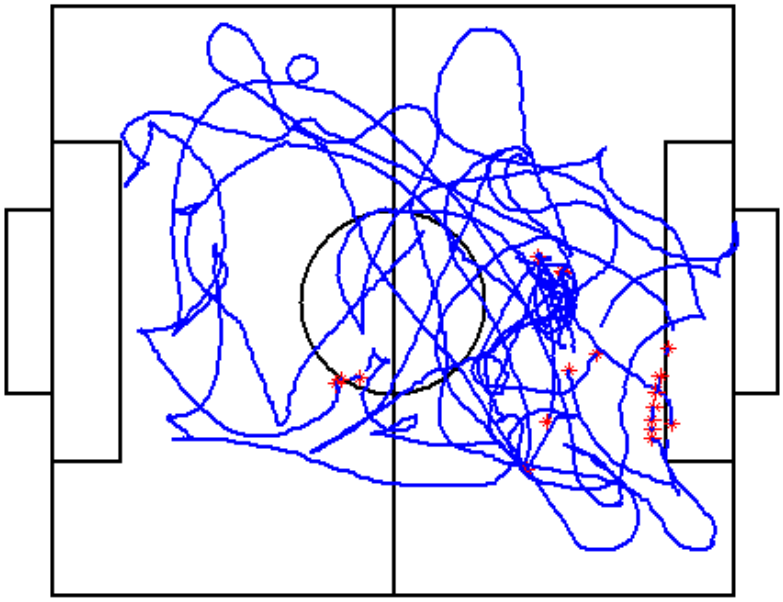
5.3.4 Primer sledenja

Za primer sledenja smo izvedli trening-tekmo treh robotkov, kjer so vodeni le robotki enega tima drugi tim pa ni voden. Barve robotkov in žogice prikazuje slika 5.25. Sledilnik je uspešno sledil šestim robotkom in žogici, kar tudi potrjuje tabela 5.5. Med treningom je prihajalo do trkov med robotki, kar je v nekaterih primerih rezultiralo v izgubi igralca (npr. stik dveh oznak enake barve), vendar je odstotek takih dogodkov zanemarljiv.

Slika 5.24: *Oznake robotkov za trening-tekmo. Dve identifikacijski barvi sta enaki za oba tima.*

Trajanje trening-tekme	1.5 min
Število izgubljenih objektov	98
Odstotek izgubljenih objektov	0.45 %
Povprečni čas procesiranja ene slike	3.05 ms
Povprečna frekvenca obdelave ene slike	327 Hz

Tabela 5.5: *Analiza sledenja v trening-tekmi.*



Slika 5.25: *Izrisana pot gibanja robotka št. 3 na igrišču. Rdeče zvezdice označujejo mesto, kjer je prišlo do izgube robotka.*

Poglavje 6

Zaključek

V diplomskem delu smo predstavili sistem za sledenje robotkov in žogice v igrah robobrca. Sistem je sestavljen iz programa za kalibracijo kamere in programa za sledenje. Predstavljen in uporabljen je poenostavljen model kamere, ki upošteva radialno distorzijo in planarni model perspektivne projekcije z dvema bežiščnima točkama, ter predpostavlja ekvivalentnost centra in središča slike.

Kalibracija kamere se lahko izvede avtomatsko, ali pa z ročno izbiro devetih referenčnih točk. Avtomatski del je sicer uspešen, vendar v nekaterih primerih slabo popravi radialno distorzijo. Vzrok za odpoved je v tem, da referenčne točke na radialno popačeni sliki iščemo s Houghovim transformom za ravne črte, iz vsake črte pa se vzame le po tri referenčne točke. V nadaljnem razvoju bi lahko avtomatsko kalibracijo naredili bolj robustno in uspešno, če bi črte igrišča iskali s Houghovim transformom za elipse. Tako bi lahko vse točke, ki dovolj dobro ustrezajo posamezni elipsi, uporabili kot referenčne točke kalibracije.

Program za sledenje je sestavljen iz dveh glavnih delov: iskanje barvnih oznak in sestavljanje oznak v robotke. Iskanje barvnih oznak je omejeno na področja, ki jih določimo preko odštevanja referenčne slike od trenutne. Razvrščanje barv je lahko izvedeno z razvrščevalnikom na osnovi evklidske ali Mahalanobisove razdalje. Operacije razvrščanja vedno potekajo v Lab barvnem prostoru in so časovno požrešne. Zaradi potrebe po obdelavi v realnem času, se po označevanju barvnih oznak v inicializaciji generira prevajalna tabela z izbranim razvščevalnikom. Vsaki barvi v RGB prostoru v tabeli ustreza določen razred, ki smo ga izračunali z uporabo razvrščevalnika v Lab

prostoru. Pri razvrščevalniku z Mahalanobisovo razdaljo za vse razrede uporabimo enako kovariančno matriko, ki jo ocenimo iz homogenih delov barve igrišča. Rezultat uporabe evklidske razdalje v nasprotju z Mahalanobisovo so nekoliko večje, vendar bolj popačene oblike oznak, zato je izbira odvisna od situacije. Kadar imamo opravka z majhnimi vidnimi področji oznak robotkov (igra petih igralcev) je najprimernejša razdalja za razvrščanje barv evklidska, kadar pa so oznake dobro vidne (igra treh igralcev) je Mahalanobisova boljša. Iskanje oznak robotkov in žogice lahko poteka na celi sliki (globalno iskanje) ali pa v okolici njihovih predhodnih pozicij (lokalno iskanje). Program avtomatsko preklaplja med obema načinoma iskanja. Drugi zelo pomemben del sledilnika je iskanje pravilne kombinacije detektiranih oznak, ki ustreza iskanim robotkom in žogici. Uporabljena je preprosta in hitra različica interpretacijskega drevesa, ki v praksi navadno poišče ustrezno kombinacijo, vendar v nekaterih situacijah odpove. V nadaljnem razvoju predlagamo za določanje najboljše kombinacije uporabo metode interpretacijskega drevesa »prvi-najboljši«, saj naj bi bil v primerni uporabi kar deset do dvajsetkrat hitrejši od klasične variante [30].

Zahteve po obdelavi v realnem času so bile dosežene pri sledenju objektov v igri treh in igri petih igralcev. Čas obdelave slike pri igri petih igralcev je v globalnem načinu iskanja okoli 28 *ms* (35 *Hz*), v lokalnem pa 4 – 16 *ms* (241 – 62 *Hz*). V igri treh robotkov je globalno iskanje porabilo okoli 23 *ms* na sliko (42 *Hz*), lokalno pa približno 8 *ms* (123 *Hz*). Kadar so si barve oznak podobne, nastopa pri igri na velikem igrišču do izgubljanja igralcev, zato predlagamo bolj kompleksne oznake z manj barvami.

S časom se bodo pojavile potrebe po igri z večjim številom igralcev, slednje pa bo rezultiralo v problemu izbire barv identifikacijskih oznak robotkov. Predlagamo izbiro čim manjšega števila različnih barv, kar kompenziramo z večjim številom oznak na robotku.Slednje bo dovoljevalo izbiro med seboj bolj različnih barv. Princip sledenja bo ostal popolnoma enak, spremeni pa se bo zgolj model v interpretacijskem drevesu.

Potrebno bi bilo zgraditi še sistem, ki bi ocenil, kako različne (glede na razvrščevalnik) so si barve na igrišču. Tako bi pred učenjem, ko so barve nasprotnega tima že izbrane, postavili na igrišče nasprotnikove robote, in vse možne lastne barve oznak. Uporabnik bi označil vse barve, program pa bi med njimi izbral najbolj različno kombinacijo barv za oznake.

Predvidevamo, da bo globalni način iskanja večjega števila robotkov, na trenutnih računalnikih dosegel ali nekoliko presegel mejo 33 ms na sliko, lokalni način pa ne. Upoštevajoč trend rasti hitrosti osebnih računalnikov lahko sklepamo, da bo obstoječi sistem kmalu dosegal uspešno procesiranje v realnem času na novih, cenovno dostopnih računalnikih celo z globalnim iskanjem.

Literatura

[1] Spletna stran: www.fira.net/soccer/mirosot, zadnji obisk: julij, 2003

[2] S. S. Intille, A.F. Bobick, »Tracking using a local closed-world assumption: Tracking in the football domain«, *M.I.T., Media Laboratory Perceptual Group Technical Report, No. 296, 1994.*

[3] J. Perą, S. Kovačič, »Tracking People in Sport : Making Use of Partially Controlled Environment«, *SKARBEK, Wĺadysĺaw (Ed.). Computer analysis of images and patterns: 9th international conference, CAIP 2001, Warsaw, Poland, September 5-7, 2001: proceedings, (Lecture notes in computer science, 2124). Springer, 2001, str. 374-382.*

[4] K. Grešak: »Analiza športnih iger z računalniškim vidom«, *diplomska naloga , Fakulteta za računalništvo in informatiko, Ljubljana , Slovenija XXX.*

[5] I. Lesjak: »Sledenje in analiza vizalne informacije na posnetkih športnih dogodkov«, *magistrsko delo, Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija 2001.*

[6] S. S. Intille, J. W. Davis, A.F. Bobick: »Real-Time Closed-World Tracking«*, M.I.T. Media Laboratory Perceptual Computing Section Technical Report, No. 403, 1996.*

[7] D. Comaniciu, V. Ramesh, P. Meer: »Real-Time Tracking of Non-Rigid Objects using Mean Shift*«, Proc. IEE Conf. on Computer Vision and Pattern Recognition, Hilton Head, SC, volume II, Junij 2000, str. 142-149.*

[8] S. J. McKenna, Y. Raja, S. Gong: »Object Tracking using Adaptive Colour Mixture Models«, *Asian Conference on Computer Vision, 1998.*

[9] G. Klančar, O. Orqueda, D. Matko, R. Karba: »Robust and Efficient Vision System for mobile robots control-application to soccer robot« , *Elektrotehniški vestnik 68(5), str. 247-253, 2001.*

[10] P. Borensztejn, J. Jacobo, M. Mejail: »Design of the Vision System for the UBA-Sot team«, *Universidadde Buenos Aires, Argentina 2001.*

[11] C. Amoroso, A. Chella, V. Morreale, P. Storniolo: »A Segmentation System for soccer robot based on neural networks«, *Lecture Notes in Artificial Intelligence, 2000, Vol 1856, str.136-147.*

[12] L. Iocci, D. Nardi: »Self-Localization in the RoboCup Enviroment«, *Proceedings of 3rd RoboCup Workshop 1999.*

[13] Shintaro Okuda: »Range Finding for Robot Vision based on Mathemathical Morphology and Hough Transform (implemented on PIPADS Image Processing Machine)«, *master degree, The University of Electro-Communications, Tokyo, Japan, 1995.*

[14] R. Jain, R. Kasturi, B. G. Schnuck: »Machine Vision«, *McGraw-Hill inc.: str. 214-219, 1995.*

[15] Y. R. Tsai: »A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses«, *IEEE Journal of Robotics and Automation, RA-3(4): 323-344, 1987.*

[16] D. A. Forsyth, J. Ponce: »Computer Vision: A Modern Approach«, *Prentice Hall:str. 47-48, 2003.*

[17] J. Perš, S. Kovačič: Nonparametric, Model-Based Radial Distortion Correction Using Tilted Camera Assumption

[18] D. A. Forsyth, J. Ponce: »Computer Vision: A Modern Approach«, *Prentice Hall:str. 29, 2003.*

[19] R. Jain, R. Kasturi, B. G. Schnuck: »Machine Vision«, *McGraw-Hill inc.: str. 341, 1995.*

[20] G. Fangi, G. Gagliardini, E. S. Malinverni: »Photointerpretation and small scale stereoplotting with digitally rectified photographs with geometrical constraints«, *CIPA Symposium 2001.*

[21] J. Perš: »Sledenje ljudi z metodami računalniškega vida*«, magistrsko delo, Fakulteta za elektrotehniko, Ljubljana, april 2001.*

[22] R. Jain, R. Kasturi, B. G. Schnuck: »Machine Vision«, *McGraw-Hill inc., 1995, str. 408.*

[23] D. Comaniciu, P. Meer: »Mean Shift Analysis and Applications«, *IEEE International Conference of Computer Vision, Kerkyra, Greece,1197-1203,1999.*

[24] R. Jain, R. Kasturi, B. G. Schnuck: »Machine Vision«, *McGraw-Hill inc., 1995, str. 409-410.*

[25] N. Pavešič: »Razpoznavanje vzorcev: Uvod v analizo in razumevanje vidnih in slušnih signalov«, *Fakulteta za elektrotehniko, Ljubljana, 2000, str. 206.*

[26] J. P. Hoffbeck, D. A. Landgebe: »Covariance Matrix Estimation and Classification with Limited Trainig Data«, *IEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 7, str. 763-767, julij 1996.*

[27] J. H. Friedman: »Regularized Discriminant Analysis, Journal of American Statistical Association«, *Vol. 84, str. 165-175, Marec 1989.*

[28] S. Aeberhard, D Coomans, O. de Vel: »The Performance of Statistical Pattern Recognition Methods in High Dimensional Settings«, *School of Information Technology , Australia, 1993, Technical Report 93/4.*

[29] T. K. Ho, J. J. Hull, S. N. Srihari: »Combination of Decisions by Multiple Classifiers«, Structured Document Image Analysis, *Springer-Verlag, 1992, 188-202.*

[30] R. B. Fisher: »Best-First and Ten Other Variations of the Interpretation-Tree Model Matching Algorithm«, *Dept. of Artificial Intelligence, The University of Edinburgh, Research Paper, 1994.*

[31] L. Robert: »Camera Calibration without Feature Extraction«, *Institut National de Recherche en Informatique et Automatique (INRIA), Programme 4, Robotique, image et vision, rapport de recherche No. 2204, Februar 1994.*

[32] R. Cipolla, T. Drummond, D. Robertson: »Camera Calibration from vanishing points in images of architectural scenes«, *Proc. British Machine Vision Conference, Nottingham, volume 2, str. 382-391, september 1999.*

[33] F. Devernay, O. Faugeras: »Automatic calibration and removal of distortion from scenes of structured enviroments«, *Proceedings of the SPIE Conference on Investigate and Trial Image Processing, 2567, SPIE, San Diego, CA, julij 1995.*

[34] C. Rother: »A new Approach for Vanishing Point Detection in Architectural Environments«, *Proceedings of the 11th British Machine Vision Conference (BMVC'00), 2000, Bristol, UK, pp. 382-391.*

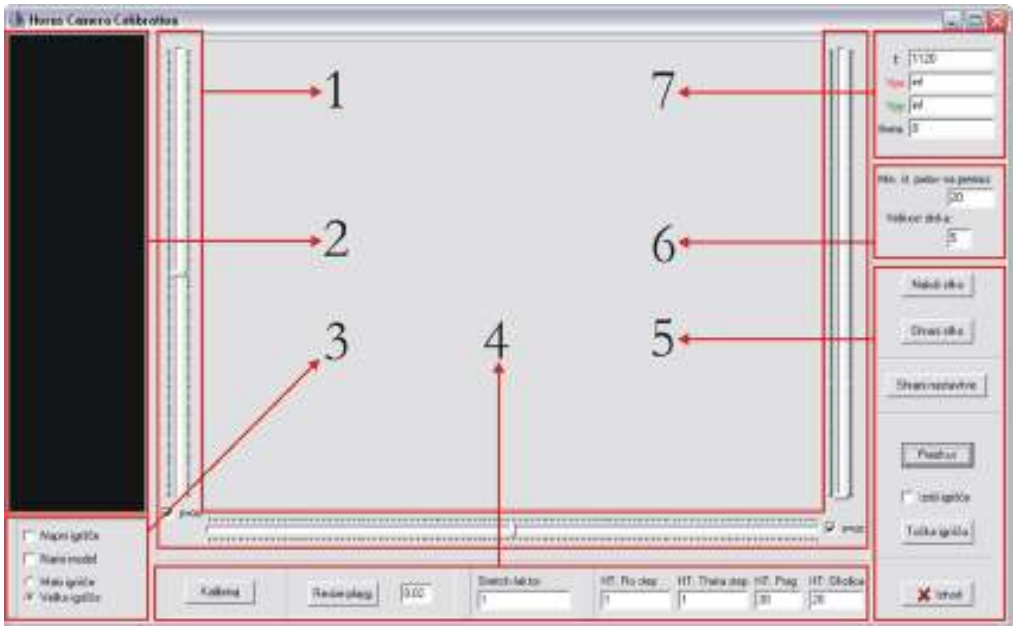
[35] T. Tuytelaars, L. Van Gool, M. Proesmans, T Moons: »The Cascaded Hough Transform«, *International Conference on Image Processing, 1997: Proceedings.*

[36] F. Szenberg, P. C. P Carvalho, M. Gattass: »Automatic Camera Calibration for Image Sequences of a Football Match«, *Advances in Pattern Recognition, Icapr 2001, Second International Conference, March 2001: Proceedings, str. 301-311.*

Dodatek A

Priročnik za uporabo programa »Horus Camera Calibration«

Horus Camera Calibration (HCC) je samostojen program za kalibracijo kamere v tekmah robobrca. Program uporablja informacijo posnetega igrišča, in sicer njegove vidne notranje robove. Notranji parametri kamere se določijo le na podlagi njegovih mejnih črt, neodvisno od njihove dolžine. HCC predvideva, da so vidne robne črte igrišča in tvorijo v ravnini zarotiran pravokotnik. Model kamere, ki ga uporablja, je opisan v poglavju 3.2 diplomskega dela.



Slika A.1: *Kalibracijska konzola.*

Uporabniški vmesnik je sestavljen iz področij, katera prikazuje slika A.1, in si sledijo:

- **Področje 1:** Drsniki za nastavljanje bežiščne točke po x osi *Vpx*, bežiščne točke po y osi *Vpy* in parametra fokalne razdalje *f* (v poglavju 3.2 je označen kot *H*).

  S spreminjanjem levega navpičnega drsnika, se spreminja vrednost *Vpy*. Pod drsnikom se nahaja stikalo »y=4«. Če je to stikalo odkljukano, pomeni, da je bežiščna točka *Vpy* v neskončnosti, torej v kalibraciji ni upoštevana, in je levi navpični drsnik neaktiven.

  S spreminjanjem spodnjega vodoravnega drsnika, se spreminja vrednost *Vpx*. Pod drsnikom se nahaja stikalo »x=4«. Če je to stikalo odkljukano, pomeni, da je bežiščna točka *Vpx* v neskončnosti, torej v kalibraciji ni upoštevana, in je spodnji vodoravni drsnik neaktiven.

  S spreminjanjem desnega navpičnega drsnika, se spreminja vrednost *f*. Najmanjša vrednost drsnika je nič, saj vrednost *f* nikoli ni negativna. Če je drsnik postavljen na vrednost »0«, se parametra *f* ne upošteva v modelu.

- **Področje 2:** Okno za izpisovanje poteka avtomatske kalibracije

- **Področje 3:**

  Napni igrišče: Aktiviraj ročno napenjanje igrišča. Ročno določimo izhodišče koordinatnega sistema igrišča, in parameter Ω (razmerje m/pixel). (Parametri f, Vpx in Vpy morajo biti določeni!)

  Nariši model: Aktiviraj ročno izbiro kontrolnik točk za kalibracijo. Nariše model kontrolnik točk za kalibracijo.

  Malo igrišče: Določi, da se na sliki nahaja malo igrišče. To je pomembno, saj so razmerja različna za malo in veliko igrišče. Določanje parametrov *f*, *Vpx* in Vpy, in izhodišče koordinatnega sistema igrišča je neodvisno od te izbire. Pomembno je za določitev faktorja Ω in pravilnost mej igrišča.

  Veliko igrišče: Določi, da se na sliki nahaja veliko igrišče. To je pomembno, saj so razmerja različna za malo in veliko igrišče. Določanje parametrov *f*, *Vpx* in

Vpy, in izhodišče koordinatnega sistema igrišča je neodvisno od te izbire. Pomembno je za določitev faktorja Ω in pravilnost mej igrišča.

- **Področje 4:**

  Kalibriraj: Izvajanje kalibracije. Če je aktivirano stikalo »Narisi model«, bo kalibracija potekala po ročno izbranih točkah, sicer pa avtomatsko.

  Resize playg.: Spremeni velikost označenega igrišča za določen odstotek *p*. Vrednost parametra *p* se določi v prvem oknu desno od gumba, vrednosti pa so podane v »procent/100«. Raztegnitev igrišča poteka iz sredine navzven. Igerišče mora biti že določeno.

  Stretch faktor: Vrednost, ki se sama nastavi preko kalibracije, in je pomembna po določevanju paramera *f*. Možno jo je ročno nastavljati.

  HT: Ro step: Razločljivost parametričnega prostora za HT po polmeru. Vrednost se avtomatsko nastavi v kalibraciji in je ni moč spreminjati.

  HT: Theta step: Razločljivost parametričnega prostora za HT po kotu. Vrednost se avtomatsko nastavi v kalibraciji in je ni moč spreminjati.

  HT: Prag: Pragovna vrednost pri iskanju maksimumov v parametričnem prostoru. Le celice parametričnega prostora z vsebino višjo od te vrednosti so upoštevane.

  HT: Okolica: Okolica, ki določi maksimum v HT. Neka celica je maksimum, če v njeni okolici ni nobene celice z višjo vsebino.

- **Področje 5:**

  Naloži sliko: Naloži iz diska sliko igrišča, ki ga uporabimo za kalibracijo.

  Shrani sliko: Shrani na disk trenutno sliko.

  Shrani nastavitve: Shrani parametre kalibracije. Te parametre se potem naloži v programu za sledenje

  Preizkus: Izvede korekcijo slike s predhodno določenimi parametri.

  Izriši igrišče: S klikom na »Preizkus« izriše še igrišče.

  Točke igrišča: Nariše nepopravljeno sliko, in na njej označi glavne točke igrišča.

- **Področje 6:**

  Št. pix na premici: Najmanjše število slikovnih elementov na premici.

  Velikost diska: Velikost diska pri odstranjevanju ozadja.

- **Področje 7:**

  f: Vrednost parametra *H* v modelu iz poglavja 3.2.

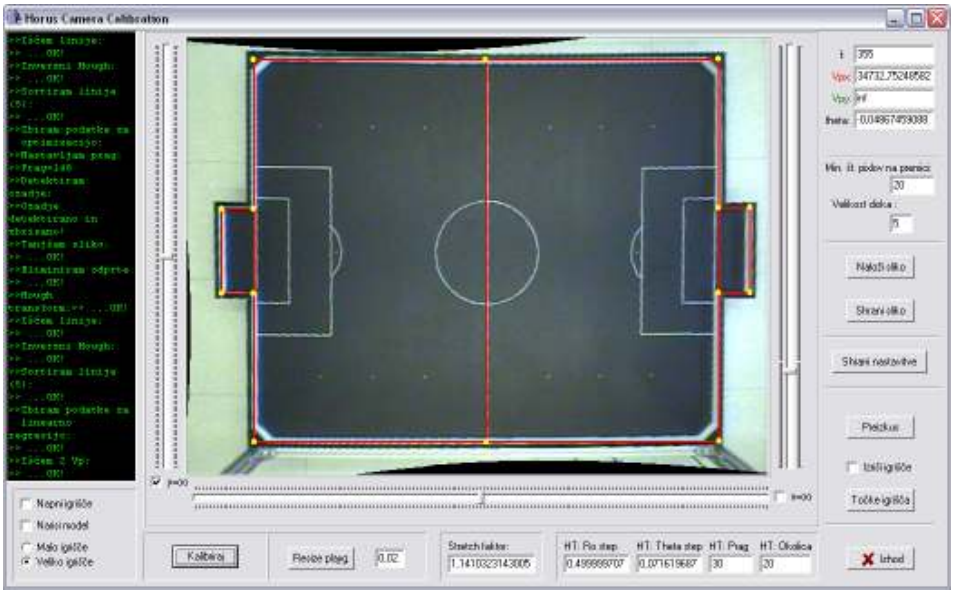
  *Vpx***:** Vrednost parametra *p* v modelu iz poglavja 3.2.

  *Vpy***:** Vrednost parametra *q* v modelu iz poglavja 3.2.

  *theta***:** Vrednost parametra θ v modelu iz poglavja 3.2.

A.1 Tipičen primer avtomatske kalibracije

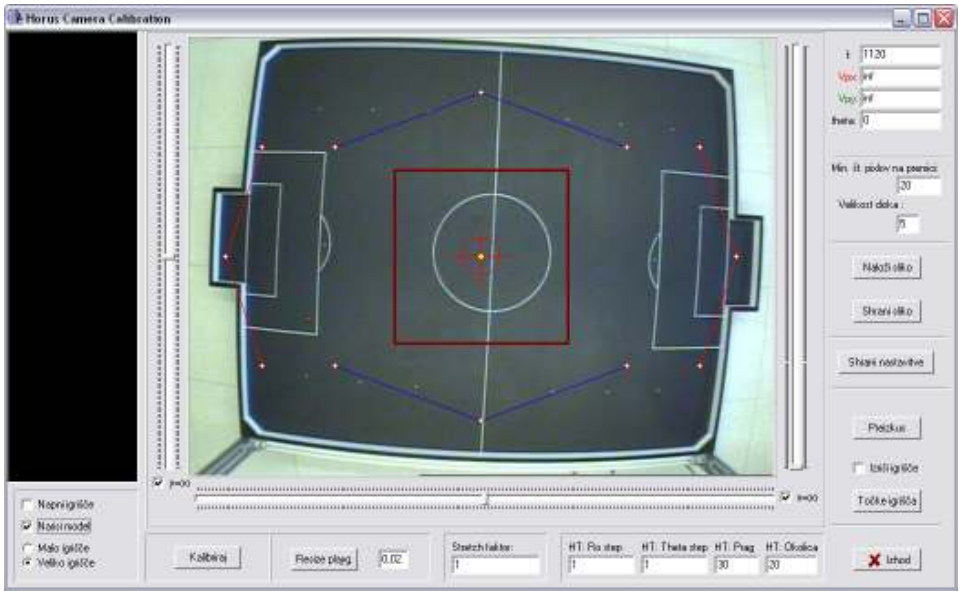
Naložimo sliko s klikom na gumb »Naloži sliko« (področje 5). Če nalagamo sliko velikega igrišča, naj bo v področju 3 označeno »Veliko igrišče«, sicer pa »Malo igrišče«. Če je slika premajhne ali prevelike resolucije, jo lahko avtomatsko prilagodimo, če se postavimo na sliko in kliknemo desni gumb na miški. S klikom »Kalibriraj« v področju 4 sprožimo proces kalibracije. Obdelava slike se bo prikazovala po vsakem koraku, v področju 1 pa se bo izpisoval potek procesiranja. Končni rezultat bo popravljena slika, z izrisanim igriščem (slika A.2).



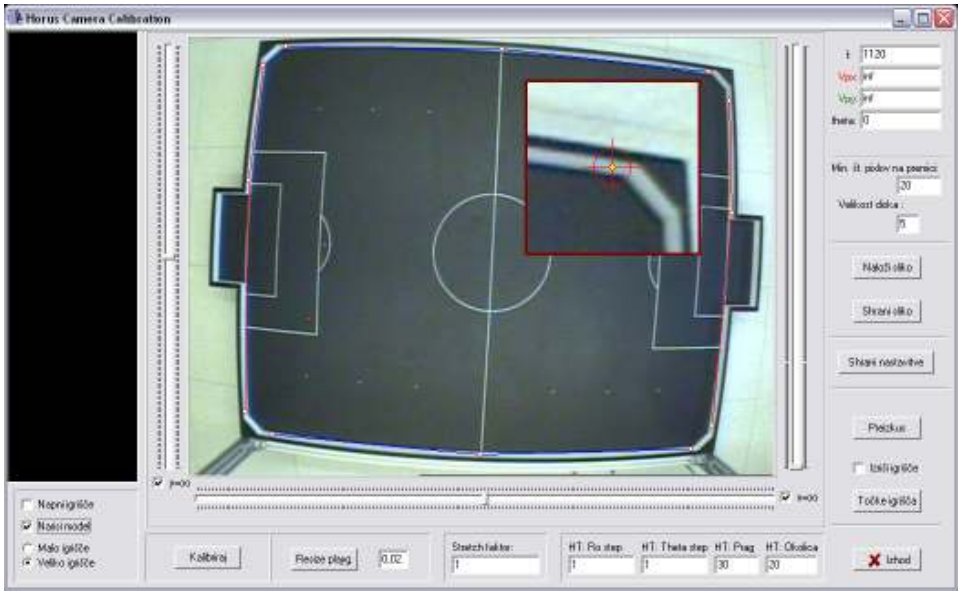
Slika A.2: *Uspešna kalibracija z izrisanim igriščem.*

A.2 Tipičen primer ročne kalibracije

Naložimo sliko s klikom na gumb »Naloži sliko« (področje 5). Če nalagamo sliko velikega igrišča, naj bo v področju 3 označeno »Veliko igrišče«, sicer pa »Malo igrišče«. Če je slika premajhne ali prevelike resolucije, jo lahko avtomatsko prilagodimo, če se postavimo na sliko in kliknemo desni gumb na miški. V področju 3 izberemo »Nariši model«. Prikaže se model kot na sliki A.3. Točke napnemo na robne črte igrišča (slika A.4). Pri premikanju točk, se zraven prikazuje okno s povečano okolico dotične točke. Tako lahko dovolj natančno določimo točko, če ob premikanju ne gledamo na sliko, pač pa le v okno. S klikom »Kalibriraj« v področju 4 sprožimo proces kalibracije. Obdelava slike se bo prikazovala po vsakem koraku, v področju 1 pa se bo izpisoval potek procesiranja. Končni rezultat bo popravljena slika, z izrisanim igriščem kot v sliki A.2. Po končani kalibraciji stisnemo gumb »Resize playg.«, ki za določen procent raztegne (ali skrči če je predznak negativen) igrišče.



Slika A.3: *Izris kontrolnih točk. Ustrezne točke so v trojčke povezani z daljicami.*



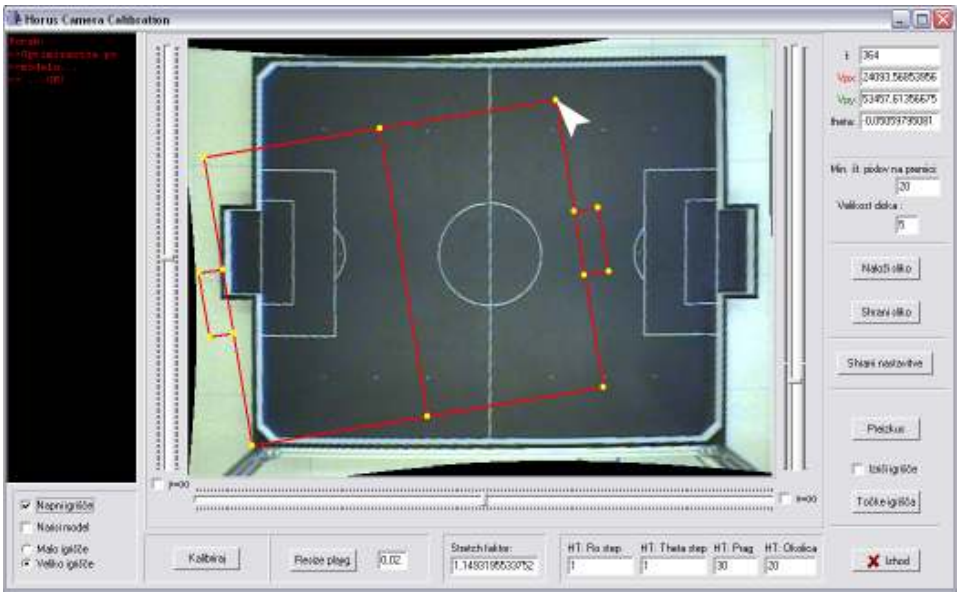
Slika A.4: *Kontrolne točke napete na črte igrišča.*

A.3 Korekcija igrišča

Včasi se zgodi, da po končanem postopku kalibracije igrišče ni pravilno izisano. Takrat lahko igrišče ponovno napnemo. V področju 3 izberemo možnost »Napni igrišče«. Z levim gumbom na miški kliknemo na levi spodnji rob igrišča, in še enkrat na desni zgornji. Izvedla se bo korekcija parametrov kamere, kar se tiče dimenzij in koordinatnega sistema. Igrišče precej točno lahko določimo, saj se ves čas, do drugega klika z miško izrisuje na zaslon. Sliko, ko določamo drugo točko, prikazuje slike A.5.

POZOR:

Igrišče določa tudi orientacijo v sliki, zato bo popravljen tudi parameter θ.



Slika A.5: *Napenjanje igrišča.*

Dodatek B

Priročnik za uporabo programa »Horus Robot-Soccer Tracker«

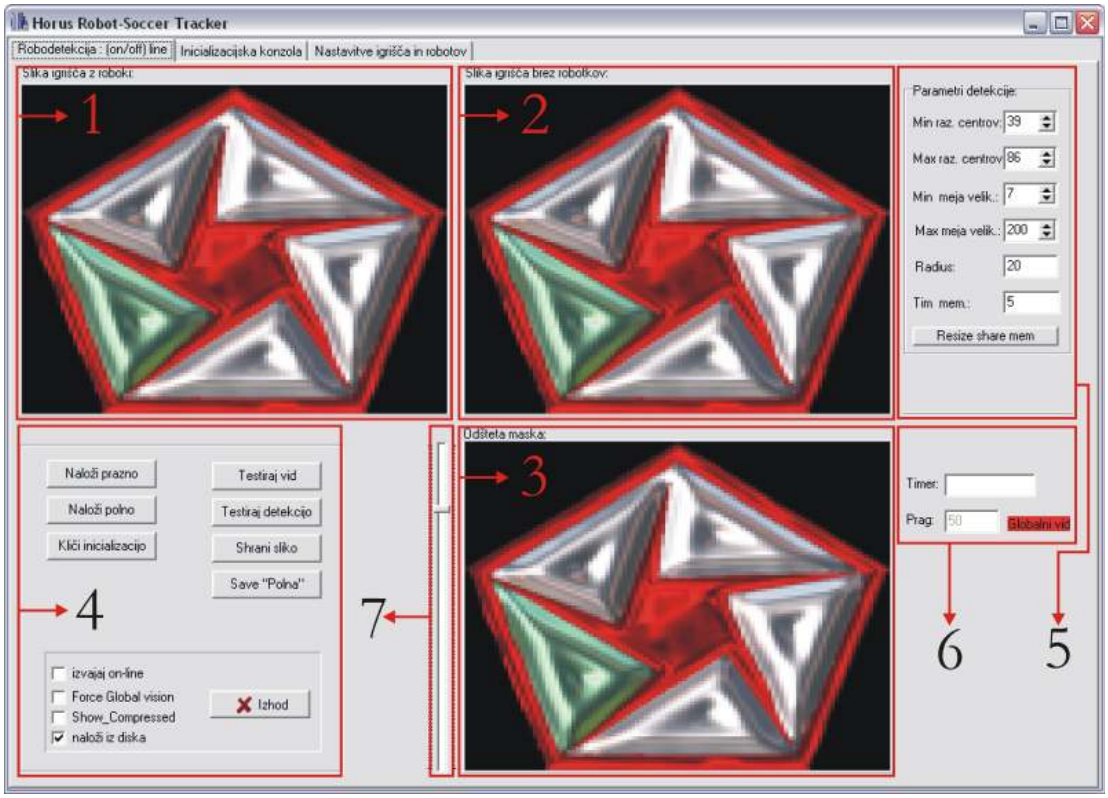
Horus Robot-Soccer Tracker (HRST) je samostojen program za sledenje robotov v igrah robobrca. Sestavljen je iz treh glavnih vmesnikov:

1. Robodetekcija (on/off) line
2. Inicializacijska konzola
3. Nastavitve igrišča in robotov

V nadaljnih podpoglavjih bomo opisali vsako konzolo posebej, in zaključili s tipičnim primerom inicializacije sledilnika.

B.1 Robodetekcija (on/off) line

To je glavni vmesnik za sledenje in prikazovanje detekcije, ki ga sestavlja sedem področji (slika B.1) .



Slika B.1: *Vmesnik Robodetekcija (on/off) line z označenimi področji.*

- **Področje1:**

  Slika igrišča z robotki: Slika, iz katere sledilnik naučimo barve in razmerja.

- **Področje2:**

  Slika praznega igrišča: Slika posnetega igrišča brez robotkov. (Služi za odštevanje ozadja od slike).

- **Področje3:**

  Odšteta maska: Prikazuje različne rezultate

- **Področje4:**

Naloži prazno: Naloži sliko v okno 2.

Naloži polno: Naloži sliko v okno 1.

Kliči inicializacijo: Preklopi na vmesnik »inicializacijska konzola«.

Testiaj vid: Segmentira sliko z uporabo prevajalne tabele, in prikaže rezultat v področje 3.

Testiraj detekcijo: Detektira robote in žogico na sliki v področju 1 in izriše rezultat v sliko iz področja 3.

Shrani sliko: Shrani sliko v področju 2. (Za shranjevanje slike praznega igrišča)

Save »Polna«: Shrani sliko iz področja 1.

Izhod: Izhod iz programa.

Izvajaj online: Prične z detekcijo na slikah iz kamere, in izrisuje rezultat v področje 2, dokler je forma aktivna.

Force Global vision: Vsili globalno detekcijo brez preklapljanja na lokalno.

Show_Compressed: Prikazuje segmentirano sliko iz kamere.

Naloži iz diska: Določi izvor slike. Če je okno odkljukano naloži iz diska, sicer pa iz kamere.

- **Področje5:**

  Min raz. centrov: Minimalna razdalja med dvema centroma oznak na enem robotu v milimetrih.

  Max raz. centrov: Maximalna razdalja med dvema centroma oznak na enem robotu v milimetrih.

  Min meja velik: Spodnja meja velikosti možne oznake.

  Max meja velik: Zgornja meja velikosti oznake.

  Radius: Radij okolice pri lokalnem sledenju.

  Tim mem: Število robotov v programu za vodenje. To je zelo pomembno, saj obstajajo različni programi za vodenje. Tisti, ki vodi igro treh robotov ima drugačno strukturp vhoda kot program, ki vodi igro petih. Če spremenimo to vrednost, program avtomatsko spremeni velikost vhodne strukture za program vodenja robotov.

Resize share mem: Spremeni velikost izhodne strukture glede na vrednost »Tim mem«.

- **Področje 6:**

Timer: Meri čas obdelave ene slike. (V namen testiranja)

Prag: Pragovna vrednost za odštevanje slike.

- **Področje 7:**

Drsnik: Določa pragovno vrednost za odštevanje slike, in jo vpiše v področje 6, in sicer v *Prag*.

B.2 Inicializacijska konzola

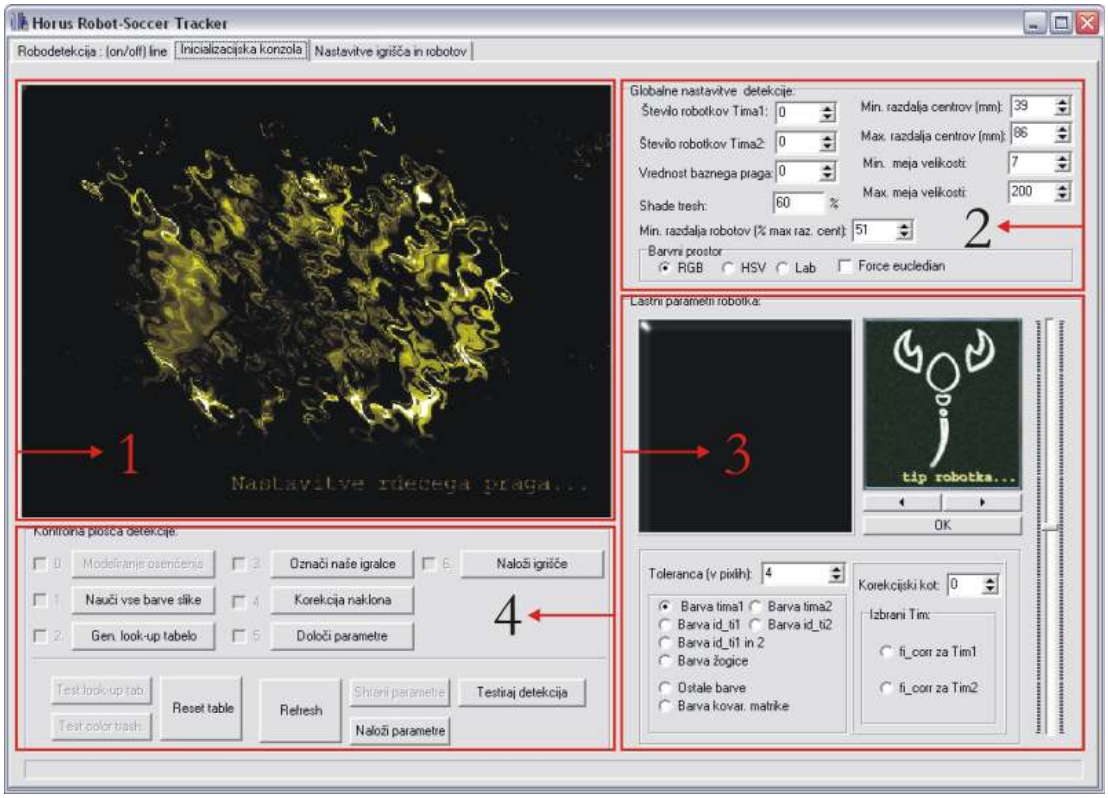
To je glavni vmesnik za sledenje in prikazovanje detekcije, ki ga sestavlja sedem področji (slika B.1) .

- **Področje 1:**

Slika: Inicializacijska slika. Kadar učimo barve (»Nauči vse barve«), se s stiskom levega gumba na miški izrisuje povečana slika v področje 3. Če je pritisnjen desni gumb na miški in se premika kurzor navzgor, se povečava v področju 3 veča. sicer pa manjša.

- **Področje 2:**

Število robotkov Tima1: Število robotko prvega (našega) tima. Vrednost se avtomatsko nastavlja z učenjem barv, lahko pa se tudi ročno, vendar če je učenje pravilno izvedeno to ni potrebno.



Slika B.1: *Izgled v štiri področja razdeljene inicializacijske konzole.*

Število robotkov Tima2: Število robotko drugega (nasprotnega) tima. Vrednost se avtomatsko nastavlja z učenjem barv, lahko pa se tudi ročno, vendar če je učenje pravilno izvedeno to ni potrebno.

Vrednost baznega praga: Vrednost pragovne vrednosti pri odštevanju slike.

Min. razdalja centrov (mm): Minimalna razdalja med težišči oznak robotka.

Max. razdalja centron (mm): Maximalna razdalja med težišči oznak robotka.

Min. meja velikosti: Najmanjša velikost vidne barvne zaplate oznake robotka.

Max. meja velikosti: Največja velikost vidne barvne zaplate oznake robotka.

Shade tresh: Neveljavna opcija v tej verziji.

Min. razdalja robotov: Najmanjša razdalja med centri dveh robotkov.

Barvni prostor:

RGB: razvrščanje v RGB barvnem prostoru.

HSV: razvrščanje v HSV barvnem prostoru.

Lab: razvrščanje v Lab barvnem prostoru.

Force eucledian: Če je označen, bo razvrščevalnik razvrščal z evklidsko razdaljo, *sicer pa z Mahalanobisovo.*

- **Področje 3**

Toleranca v pixlih: V slikovnih elementih podana velikost okna za zajem barve.

Barva tima1: Učimo razred timske barve prvega (našega) tima.

Barva tima2: Učimo razred timske barve drugega (nasprotnega) tima.

Barva Id_ti1: Učimo identifikacijsko barvo prvega (našega) tima.

Barva Id_ti2: Učimo identifikacijsko barvo drugega (nasprotnega) tima.

Barva Id_ti1 in ti2: Učimo razred identifikacijske barve, ki je skupna prvemu in drugemu timu.

Barva žogice: Učimo razred barve žogice.

Ostale barve: Učimo razred barve, ki ne pripada oznakam ali žogici.

Barva kovar. matrike: Označimo področje na sliki, ki določi ovariančno matriko.

Korekcijski kot: Korekcijski kot robotkov.

fi_corr za Tim1: Indikator, da želimo popravljati korekcijski kot prvega (našega) tima.

fi_corr ta Tim2: Indikator, da želimo popravljati korekcijski kot drugega (nasprotnega) tima.

Drsnik: Spreminja korekcijski kot izbranega tima.

- **Področje 4:**

Nauči vse barve slike: Resetira prevajalno tabelo in pobriše vse barvne razrede. Ta gumb sproži učenje barv in na sliki v področju 3 se izpišejo navodila.

Gen. look-up tabelo: Ko so vsi razredi označeni, generira prevajalno tabelo za razvrščanje barv.

Označi naše igralce: Gumb za dodeljevanje imen robotkov (vpisni naslov v memoriji).

Korekcija naklona: Sproži spreminjanje korekcijskega kota na oznakah.

Določi parametre: Gumb ni veljaven v tej verziji.

Naloži igrišče: Naloži parametre kamere.

Test look-up tab: Testiraj segmentacijo s prevajalno tabelo.

Test color trash: Testiraj segmentacijo le z naučenimi razredi (brez uporabe prevajalne tabele).

Reset table: Resetiraj prevajalno tabelo.

Refresh: V področje 1 nariši sliko z robotki.

Shrani parametre: Shrani parametre inicializiranega sledilnika.

Naloži parametre: Naloži parametre inicializiranega sledilnika.

Testiraj detekcija: Testiraj uspešnost globalne detekcije.

B.3 Nastavitve igrišča in robotov

Ta vmesnik se uporablja za testiranje parametrov kamere, in določanje dimenzij oznak in njihovih medsebojnih razdalj. Sestavljajo ga tri področja:

- **Področje 1:**

  Slika: Prostor za prikazovanje rezultatov.

- **Področje 2:**

  Slika: Prostor za prikazovanje rezultatov.

  Določi faktor (m/pixel) in izhodišče k.s.: Neveljavna opcija v tej verziji.

  Izmeri velikosti v pixlih: Za merjenje širine in višine oznake.

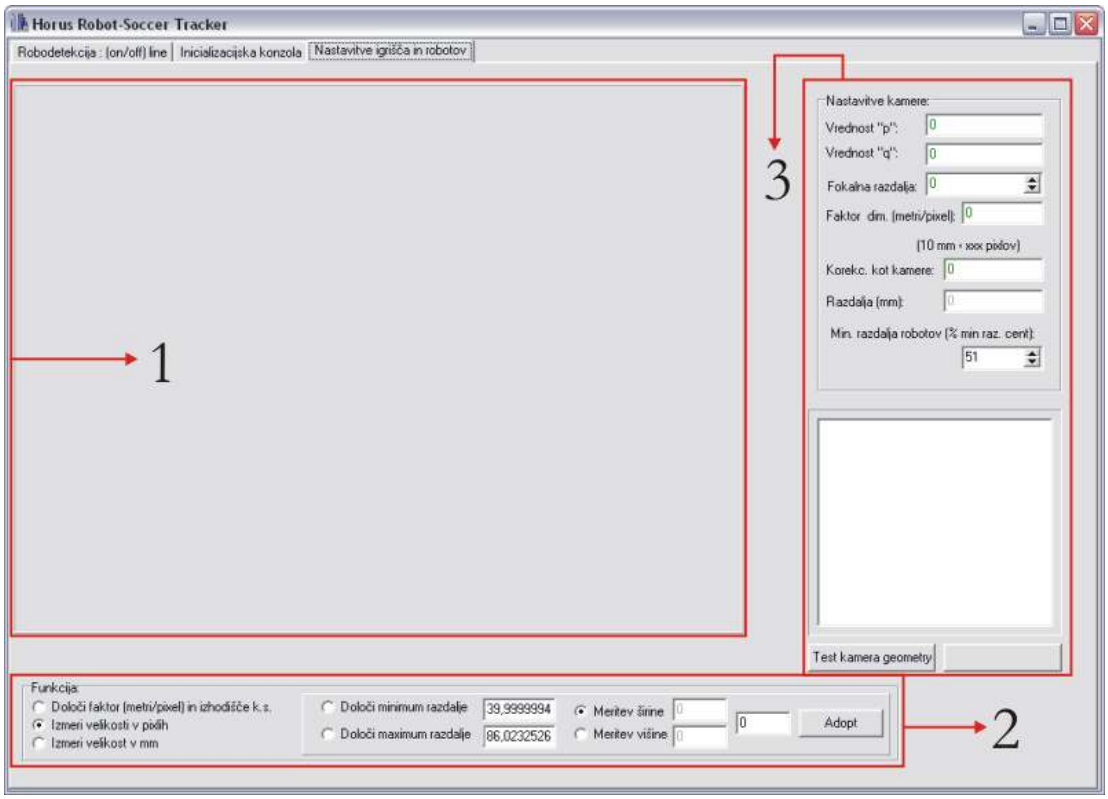
  Imeri velikost v mm: Za merjenje razdalj med oznakami.

  Določi minimum razdaje: Določi minimalno razdaljo med težišči oznak enega robotka.

Določi maksimum razdaje: Določi maksimalno razdaljo med težišči oznak enega robotka.

Meritev višine: Določi višino oznake.

Meritev širine: Določi širino oznake.



Slika B.3: *Nastavitve igrišča in robotov.*

Adopt: Osveži meje velikosti oznak.

- **Področje 3:**

  Vrednost »p«: Parameter »p« v modelu kamere.

  Vrednost »q«: Parameter »q« v modelu kamere.

  Fokalna razdalja: Parameter »f« v modelu kamere.

Faktor dim. (metri/pixel): Parameter »Ω« v modelu kamere.

Korekc. kot kamere: Parameter »θ« v modelu kamere.

Razdalja (mm): Neveljavna opcija v tej verziji.

Min. razdalja robotov (% min cent raz): Minimalna razdalja med roboti v procentih minimalne razdalje med dvema oznakama na robotu.

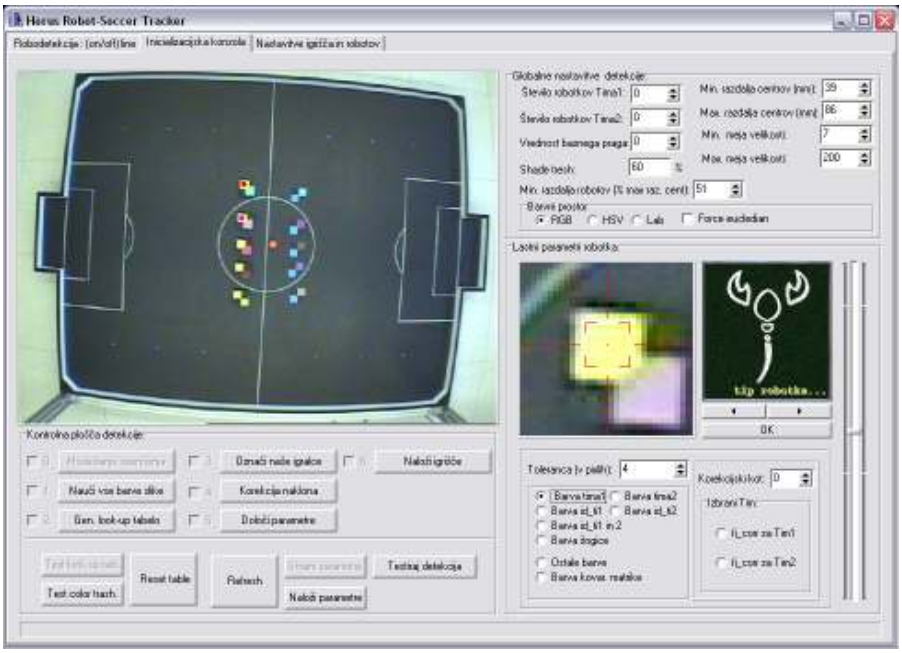
Test kamera geometry: Popravi sliko na osnovi parametrov kamere.

B.4 Tipičen primer inicializacije sledilnika

Zaženemo program sledilnika. V področju 5 na prvem vmesniku nastavimo število robotov vsakega tima. To storimo z vpisom števila v »Tim mem.« in pritisnemo »Resize share mem«. Ker nalagamo sliko iz kamere, ne sme biti »naloži iz diska« (področje 4 v prvem vmesnku) odkljukan. Igrišče v tem trenutku še ne sme vsebovati robotov ali žogice. Pritisnemo »Naloži prazno sliko« in prikaže se slika praznega igrišča. Sliko shranimo s pritiskom na »Shrani sliko«. Zaženemo program za kalibracijo kamere in izvedemo kalibracijo na shranjeni sliki, nakar shranimo parametre kalibracije (poglavje A.1).

Na igrišče postavimo vse robote in žogico. Na prvem vmesniku sledilnika pritisnemo »Naloži polno«. Z drsnikom v področju 7 nastavimo pragovno vrednost za odštevanje slik in pritisnemo »Kliči inicializacijo«. Pojavi se drugi vmesnik. Pritisnemo »Nauči vse barve slike«. V področju 3 izberemo »Barva tima1« in kliknemo na timsko barvo prvega tima v področje 1. V področju 3 se prikaže povečana slika robota. V področju 1 držimo pritisnjen desni gumb na miški in se premikamo. Vidimo, da tako nastavljamo povečavo v področju 3. Če se sedaj postavimo z miško na sliko v področju 3, opazimo okno, s katerim označujemo barvo. Velikost tega okna se nastavlja s »Toleranca (v pixlih)«. Ko je vsa barva, ki nas zanima znotraj okna, pritisnemo levi gumb na miški. Naučili smo razred barve našega tima. Da je oznaka označena vidimo na sliki v področju 1, saj se je na tem mestu pojavil rdeč krogec. Če želimo pravkar naučeni razred še nekoliko bolj naučiti,

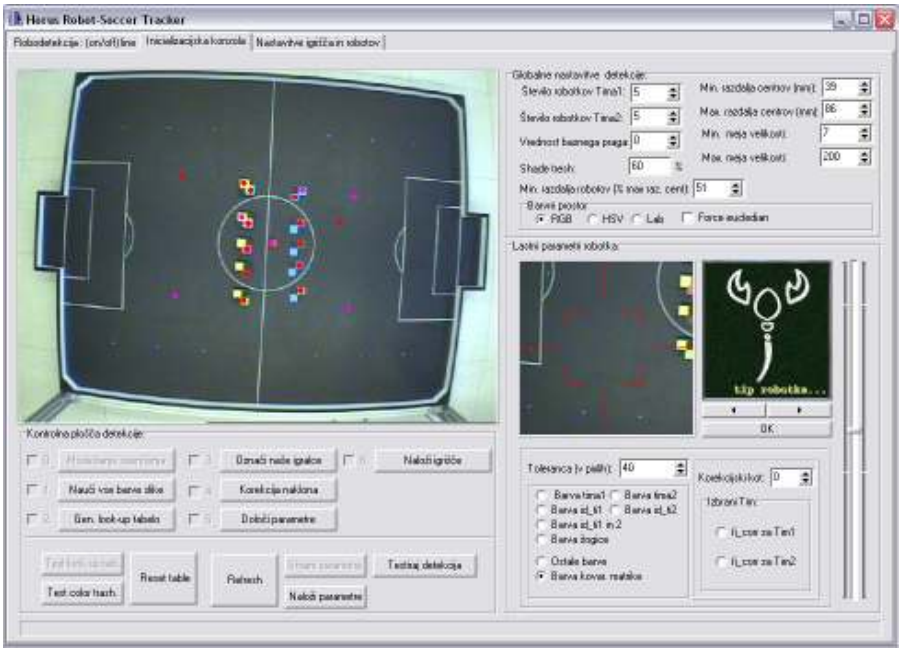
lahko s pritisnjeno tipko »shift« označimo nova področja in razred se bo osvežil z novimi podatki. Vsa področja označena s tipko »shift«, so označena z roza krogom v sliki področja 1 (slika B.4).



Slika B.4: *Označevanje timske barve našega tima.*

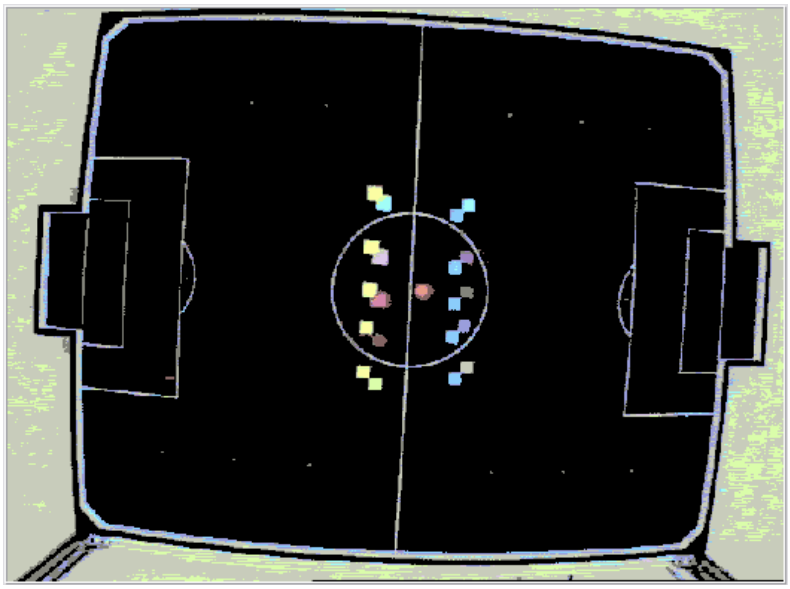
Sedaj želimo generirati razred za timsko barvo nasprotnega tima. Izberemo »Barva tim2« v področju 3 in postopamo enako, kot prej. Ko želimo ustvariti razrede za identifikacijske barve prvega (našega) tima, označimo »Barva id_ti1«, če želimo ustvariti razrede za identifikacijske barve drugega (nasprotnega) tima, označimo »Barva id_ti2«, če sta dve barvi enaki za oba tima pa označimo »Barva id_ti1 in t2«. Z vsakim na novo ustvarjenim razredom identifikacijske barve, se poveča število robotov ustreznega tima v področju 2. Ko učimo razred žogice, mora biti označen »Barva žogice«, ko učimo razrede ozadja ali recimo šumov, pa »Ostale barve«. Za oceno kovariančne matrike mora biti označeno »Barve kovar. matrike«. Tu lahko označimo dele igrišča, ki so enake barve (uporaba »shift« tipke), ali pa barvna področja, ki so si različne barve. V slednjem primeru se bodo

izračunale kovariančne matrike za vsa področja in nato iz utežene vsote povprečna kovariančna matrika. Uteži v vsoti so proporcionalne številu vzorcev za oceno posamezne matrike. V sliki B.5 vidimo primer ocenjevanja kovariančne matrike.



Slika B.5: *Ocenjevanje kovariančne matrike.*

Sedaj testiramo razvrščevalnike. V področju 2 izberemo barvni prostor in način razvrščanja. S pritiskom na »Test color trash« v področju 4 se izvede segmentacija slike (to lahko preizkusimo za vseh šest razvrščevalnikov). Če s segmentacijo nismo povsem zadovoljni, pritisnemo »Refresh« in učimo dalje še več razredov, ali pa pritisnemo »Nauči vse barve slike« in učimo razrede od začetka. Če smo zadovoljni s segmentacijo pritisnemo »Generiraj look-up tabelo« v področju 4. Indikator v spodnjem delu konzole bo prikazoval napredovanje klasificiranja barvnega prostora. Ko je tabela generirana, lahko s pritiskom na »Test look-up tab« preizkusimo segmentacijo slike preko tabele.



Slika B.6: *Segmentacija slike.*

S pritiskom na »Naloži igrišče«, naložimo parameter kalibracije. Preklopimo tretji vmesni z imenom »Nastavitve igrišča in robotov«. Označimo »Določi minimum razdalje« in z dvemi pritiski na sliko določimo to vrednost. Enako storimo za »Določi maximum razdalje«, »Meritev širine« in »Meritev višine«. S klikom na testovno polje levo od gumba »Adopt«, se izračuna ploščina oznak. S pritiskom na »Adopt«, pa se ta ploščina preračuna v nastavitve v »Min meja veliosti« in »Max meja velikost« na prvem in drugem vmesniku. Vrnemo se na drugi vmesnik z imenom »Inicializacijska konzola«.

Na vmesniku pritisnemo »Označi naše igralce«, in prikaže se slika označenih igralcev. S klikom na vsakega lahko določimo njegovo ime (prvo ime je 0!). V kolikor izberemo že uporabljeno ime, nam to program prikaže s številko »-1« poleg robota. Imena nasprotikovega tima se avtomatsko preuredijo glede na izbiro imen lastnega tima.

Kliknemo na »Korekcija naklona« in v področju 3 se prikaže robot tima, ki ga izberemo »fi_corr ta Tim1« ali »fi_corr ta Tim2«. Z drsnikom v področju 3 nastavimo željen korekcijski kot.

Detekcijo preizkusimo s pritiskom na testiraj detekcija. Vse parametre (razen barv) lahko spreminjamo, dokler ne dobimo ustreznih rezultatov.

S pritisko na »Shrani parametre«, shranimo nastavitve inicializacije, in jih kasneje naložimo preko »Naloži parametre«.