Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Siniša Šegvić

Višeagentsko praćenje objekata aktivnim računarskim vidom

Disertacija

Zagreb, 2004.

Disertacija je izrađena na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva u Zagrebu.

Mentor: prof.dr.sc. Slobodan Ribarić

Disertacija ima xxx stranica.

Disertacija br: yyyyyy.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Praćenje kretanja objekata u sceni na temelju slijeda slikovnih okvira je jedno od važnih područja računarskog vida. Velik broj primjena na tom području razmatra složene scene s izraženom trodimenzionalnom strukturom, u kojima se pristupi temeljeni na jedinstvenoj promatračkoj točki suočavaju s brojnim ograničenjima. Ti problemi uključuju ograničeno "pokrivanje" prostranih i razvedenih scena, međusobno prekrivanje objekata, spekulativno i neprecizno određivanje 3D položaja objekata, te osjetljivost na kvarove kamera i razne ostale sklopovske pogreške. Neka od spomenutih ograničenja mogu biti ublažena korištenjem aktivnog vida i panoramskih kamera, ali najelegantnije i najrobustnije rješenje se postiže porazdijeljenim praćenjem, kombinacijom rezultata dobivenih većim brojem strateški raspoređenih promatrača.

Sasvim općenito, sustav za porazdijeljeno praćenje višestrukim promatračima mora sadržavati postupke za klasično praćenje u jednom slijedu slika, te postupke za povezivanje pojedinačnih rezultata u koherentni prikaz. Klasično praćenje je samo za sebe jako složeno područje pa će ovdje biti dani samo tehnički detalji vezani za odabranu minimalističku izvedbu, a uglavnom će se razmatrati različiti aspekti povezivanja promatrača u porazdijeljeni sustav. Zanimljivi problemi tako uključuju integraciju pojedinačnih rezultata, interpretaciju ukupnih rezultata, te njihovu upotrebu u povratnoj petlji u cilju koordinacije promatrača ili bolje predikcije budućih položaja objekata u pojedinim slijedovima slika.

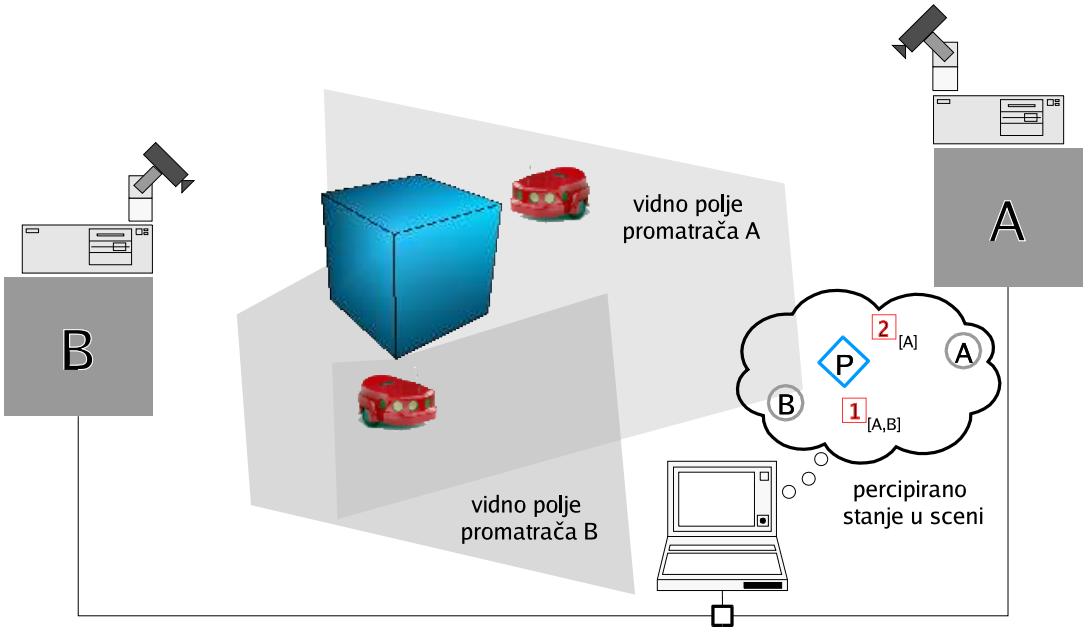
Središnja tema ovog rada su arhitekture za porazdijeljeno praćenje *aktivnim* vidom, gdje se podrazumijevaju promatrači opremljeni upravljivim kamerama. Kod takvih sustava, javlja se potreba za koordiniranim podešavanjem smjerova gledanja promatrača, u skladu s geometrijom scene, gibanjem objekata i sklopovskim ograničenjima kamera. S obzirom da se praćenje aktivnim kamerama uvijek odvija u stvarnom vremenu, u većini primjena promatrači moraju imati barem djelomičnu autonomiju pri odabiru smjera gledanja. Interakcija među autonomnim procesnim entitetima je stoga ključan element takvih arhitektura, pa ih je pogodno razmatrati u okviru teorije višeagentskih sustava.

Teorijske postavke će biti razmatrane i ilustrirane u okviru eksperimentalnog laboratorijskog sustava čiji je zadatak praćenje jednostavnih pokretnih robota u granicama laboratorija te održavanje ukupnog prikaza scene u stvarnom vremenu. Motivacija za izgradnju takvog sustava je ostvarivanje infrastrukture globalnog vida, koja bi omogućila samostalnu navigaciju siromašnije opremljenih robota upravljanih radio vezom s udaljenog računala.

Željena svojstva eksperimentalnog sustava detaljnije će se opisati u nastavku Uvoda, zajedno s pretpostavkama koje vrijede u razmatranoj problemskoj domeni, skicom odabranog pristupa rješenju, mogućim primjenama i strukturom ostatka rada.

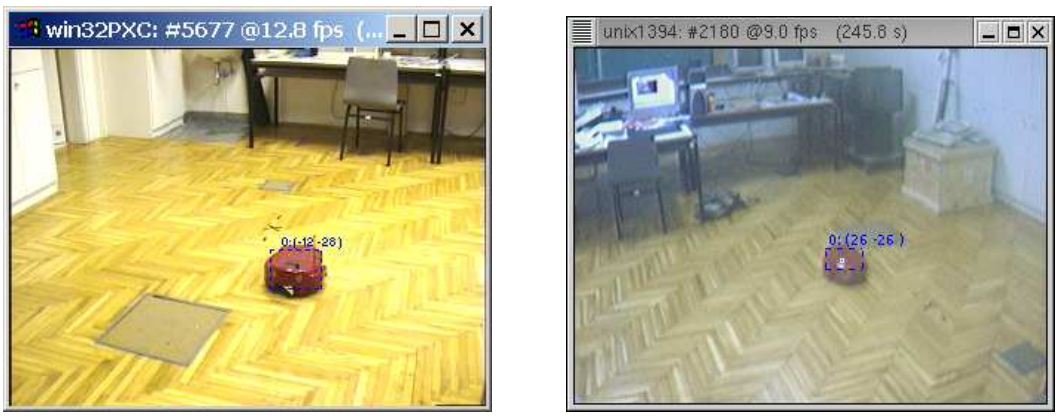
1.1 Opis problema

Razmatra se porazdijeljeni sustav za praćenje ravninskog kretanja jednostavnih robota čiji je krajnji cilj održavanje ukupnog prikaza položaja objekata u sceni. Pogodno je imati više promatračkih točaka, jer zbog razvedenosti scene ne postoji točka gledanja iz koje se može dobiti zadovoljavajući uvid u cijelu scenu. Promatrače je prikladno opremiti pokretnim kamerama jer, zbog ograničenog broja promatrača, niti sa svim kamerama nije moguće napraviti raspored kojim bi se pokrili svi zanimljivi dijelovi scene. Zbog širine primjene, pretpostavlja se da broj promatrača ne mora biti velik pa je zalihost promatrača malena (ne radi se o masovnoj senzorskoj mreži [zhao03]). Tipičan takav sustav je ilustriran na sl. 1.1, gdje su zbog jednostavnosti prikaza ucrtana samo dva promatrača.



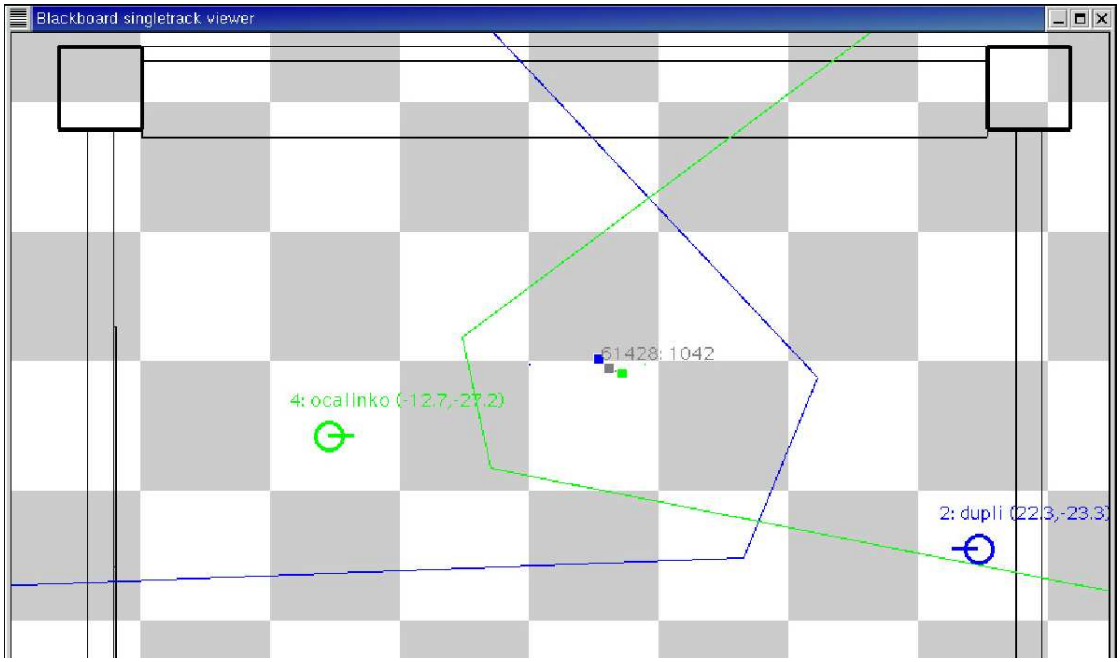
Slika 1.1: Porazdijeljeni sustav za lokalizaciju robota globalnim vidom.

Za zorniju ilustraciju zadataka sustava, sl. 1.2 prikazuje poglede dvaju promatrača koji prate isti objekt. U slikama je ucrtano slikovno okno detektiranog objekta, te tekst oblika `n:(`ϕ θ`)`, gdje `n` označava indeks objekta, dok ϕ i θ označavaju zakret i nagib kutnog položaja objekta u stupnjevima u odnosu na referentni smjer gledanja promatrača.



Slika 1.2: Pogledi dvaju promatrača na isti objekt.

Percipirano stanje u sceni dobiva se u stvarnom vremenu udruživanjem mjerenja pojedinih promatrača. To je prikazano na sl. 1.3, gdje lijevom promatraču odgovara lijevi pogled na sl. 1.2. U pozadini slike su ucrtani tlocrt laboratorija i referentna kvadratna mreža dimenzija 1×1 metar. Preko toga, za svakog promatrača je posebnom bojom ucrtan položaj, poligon ravnine gibanja koji se nalazi unutar vidnog polja, te tekst oblika `m:ime(`γ_z γ_n`)`. Pri tome `m` označava identifikator promatrača, `ime` mrežno ime računala na kojem se promatrač izvršava, dok γ_z i γ_n označavaju zakret i nagib smjera gledanja promatrača u stupnjevima. Promatrači interpretiraju detektirane položaje objekta u referentnom koordinatnom sustavu scene na temelju znanja o vlastitom položaju, a dobiveni položaji su u slici ucrtani kvadratima. Dva mjerenja su na temelju načela prostorne i vremenske bliskosti udružena u jedinstveni percipirani objekt, čiji položaj je ucrtan između dva pojedinačna rezultata. Percipirani objekt je dodatno označen vremenom zadnjeg opažanja, te ocijenjenom površinom u cm^2.



Slika 1.3: Ukupni prikaz stanja u sceni.

Kao što se iz ilustracije da naslutiti, najvažnije sposobnosti traženog sustava su:

1. praćenje projekcija objekata pojedinačnim promatračima;
2. udruživanje pojedinačnih pogleda u ukupni prikaz scene, uz sljedeće pretpostavke:
   - kamere su kalibrirane u odnosu na zajednički koordinatni sustav scene,
   - slijedovi slika pribavljeni od strane promatrača međusobno nisu sinkronizirani,
   - procesne performanse pojedinih promatrača ne moraju biti jednake,
   - promatrači imaju nezavisne satove koje je potrebno uskladiti;
3. koordinirana prilagodba smjerova gledanja promatrača u cilju "optimalnog" praćenja stanja scene, što obuhvaća barem sljedeća dva međusobno proturječna cilja:
   - precizno određivanje položaja praćenih objekata,
   - pokrivanje što većeg prostora scene;

4. robustnost u odnosu na kvarove postojećih i dodavanje novih promatrača;

5. rad u stvarnom vremenu, pri čemu se teži čim većim prosječnim učestalostima obrade (soft real time).

Iz navedenih svojstava se vidi da se razmatrani sustav nalazi na presjecištu domena aktivnog računarskog vida, raspoznavanja uzoraka te porazdijeljene umjetne inteligencije odnosno višeagentskih sustava. Promatrači moraju međusobno komunicirati pri obavljanju većine ključnih zadataka, pa je njihova međusobna interakcija vrlo važan detalj arhitekture sustava. Kako bi se omogućile brze reakcije sustava, povoljno je da upravljanje smjerom gledanja promatrača bude barem djelomično autonomno, pa se rješenje problema može izraziti višeagentskim sustavom u kojem agente predstavljaju promatrači. Svaki takav agent je instanca računalnog programa koji se izvodi na dodijeljenom umreženom heterogenom računalu, te upravlja kamerom i ostalim resursima računala. Agenti promatrači moraju međusobno surađivati jer je koordinacija smjerova gledanja nužna pri zadovoljenju ciljeva sustava. Stoga predloženi problem može poslužiti kao model za istraživanje kako teoretskih tako i praktičnih aspekata pristupa ostvarivanju kooperativnog ponašanja.

1.2 Temeljne pretpostavke

Pri projektiranju višeagentskog sustava za porazdijeljeno praćenje, javlja se više pitanja na koja nije moguće dati jednoznačan odgovor koji ne bi ovisio o konačnoj primjeni. Prva takva dilema odnosi se na detalje određivanja 3D položaja praćenih objekata. Zbog fizikalnih svojstava procesa stvaranja slike, u općenitom slučaju nije moguće odrediti 3D položaj objekta na temelju samo jedne njegove slike. Svaki slikovni element definira svjetlosnu zraku koja spaja žarište kamere i odgovarajući dio objekta, pa je za određivanje udaljenosti i stvarne veličine objekta potrebno imati najmanje dvije slike, pribavljene iz različitih točaka prostora. Ukoliko je broj točaka promatranja jednak dva, a njihova međusobna udaljenost relativno malena u odnosu na udaljenosti točaka do samog objekta, radi se o klasičnom problemu stereo vida. Kod tog problema se pretpostavlja da udaljeni objekti izgledaju vrlo slično iz pojedinih točaka promatranja, pa se 3D informacije obično dobivaju uparivanjem lokalnih značajki niske razine (malena područja, kutevi). U ovom radu, međutim, interesantnije su konfiguracije kod kojih ima više točaka promatranja koje su međusobno vrlo udaljene, pa izgled istog objekta u različitim slikama može biti vrlo različit [oswald01] (npr, ljudsko lice sprijeda i u profilu). Lokalne značajke u takvim konfiguracijama nisu od koristi, pa je uparivanje moguće obaviti samo na značajkama više razine (dijelovi objekata, pokretni objekti).

U kontekstu sustava za porazdijeljeno praćenje, 3D položaj objekta je moguće odrediti na temelju komunikacije dvaju promatrača, kao presjecište zraka koje definiraju položaji istog objekta u pojedinim slikama [dockstader01, gueziec02, matsuyama02]. Međutim, u slučaju više promatrača i više objekata, takav pristup se suočava s vrlo složenim algoritmima za pronalaženje korespondencije između slika objekata koje prijavljuju pojedini promatrači [dockstader01], koji bi mogli onemogućiti izvedbu u stvarnom vremenu. Mnogo jednostavniji posao bi se dobio kad bi se na temelju nekog apriornog znanja o sceni mogao barem ocijeniti 3D položaj objekata samo jednim promatračem. S obzirom da se razmatra praćenje robota, prirodno je u tu svrhu iskoristiti ravninsko svojstvo njihovog gibanja, tim prije što se ono javlja u velikom broju primjena. Stoga se u ovom radu pretpostavlja da je kamere moguće postaviti koso iznad ravnine kretanja, tako da objekti budu relativno mali u slikama pribavljenim iz svake točke promatranja. U tom slučaju, svaki promatrač opremljen kalibriranom

kamerom može samostalno dati prvu ocjenu položaja detektiranih objekata. Te prve ocjene mogu poslužiti za određivanje korespondencije, dok se točniji položaj objekata percipiranih korespondencijom mogu odrediti kao presjecišta 3D pravaca definiranih geometrijama odgovarajućih promatrača.

Predmet još jedne dileme jest korištenje ukupnog prikaza pri koordinaciji smjerova gledanja promatrača. Ukoliko se promatrači koordiniraju samo na temelju vlastitih pogleda na scenu i bilateralnih sporazuma [matsuyama02], ponašanje sustava je ograničeno na lokalne sheme koordinacije. Prednosti takve arhitekture uključuju jednostavnost i fleksibilnost u smislu broja promatračkih agenata i mogućnosti rada u stvarnom vremenu. Ipak, dobiveno inteligentno ponašanje skupine je dosta ograničeno po pitanju donošenja globalnih odluka, pa se u ovom radu predlaže pristup u kojem bi se koordinacija promatrača odvijala na dvije razine. Na nižoj razini, promatrači bi smjerom gledanja upravljali u reaktivnom stilu, autonomno, u skladu s pridruženim odgovornostima koje se mogu odnositi ili na praćenje pojedinih objekata, ili na motrenje zadanog dijela scene. Spomenute odgovornosti bi se stvarale odnosno prilagođavale stanju scene na hijerarhijski višoj razini, uz korištenje ukupnog prikaza stanja u sceni. Dobro svojstvo takve hijerarhijske podjele upravljanja jest da se variranjem učestalosti upravljačkih intervencija mogu postići različita ponašanja sustava, od potpuno reaktivnog do sasvim planskog odnosno kognitivnog.

Poseban problem pri svim oblicima praćenja predstavlja prekrivanje objekata i strukture scene. Jedno od bitnih pitanja u tom kontekstu jest, da li promatrači mogu automatski odrediti "slijepe" zakutke, u kojima objekti ne mogu biti detektirani. Takve mogućnosti su posebno interesantne kada su prekrivanja objekata i strukture scene česta, jer omogućuju razvoj sofisticiranih shema za optimalnu raspodjelu odgovornosti među promatračima. Kod objekata čije je gibanje ograničeno ravninom, rješenje tog problema se svodi na razvoj postupaka za detekciju onih područja ravnine gibanja koja nisu zaklonjena strukturom scene, te korištenje dobivenih informacija na obje razine koordinacije promatrača. U konkretnom slučaju, detekcija vidljivih područja scene u okruženju eksperimentalnog sustava mogla bi se obaviti analizom teksture i boje podloge laboratorija, a tu mogućnost potvrđuju i preliminarni eksperimenti.

1.3 Moguće primjene

Većina postojećih sustava za porazdijeljeno praćenje namijenjena je za praćenje ljudi u zatvorenim prostorima. Ta istraživanja su bila motivirana sljedećim područjima primjene:

- interakcija između čovjeka i računala [nakazawa98, krumm00, karuppiah01].
- nadgledanje prostora i detekcija posjetioca [cai99, collins01, yang03, zhao03],
- analiza ljudskog kretanja [dockstader01, mittal03],

Možda najveća komercijalna potreba za automatiziranim porazdijeljenim praćenjem javlja se na području nadgledanja u trgovinskim, policijskim i vojnim primjenama. Navodi se [collins01] kako cijena ljudskih resursa u ukupnim troškovima sustava video nadzora iznosi puno više od cijene računalne opreme i infrastrukture, što predstavlja vrlo plodno tlo za inovacije u cilju automatizacije postupka. Pored toga, iako se takvi sustavi već sad koriste u bankama, trgovinskim centrima i parkiralištima, obrada snimljenog materijala se u pravilu obavlja tek nakon prijave nezakonitih radnji. Omogućavanjem ranijeg alarmiranja nadležnih ustanova, automatizirane primjene bi stoga mogle kvalitativno poboljšati iskoristivost postojeće opreme za nadgledanje.

Drugo važno područje primjene porazdijeljenog praćenja su izravni TV prijenosi različitih sportskih događaja. Podatci o stanju natjecanja se mogu koristiti za poboljšanje televizijskog prijenosa putem posebne sličice u kutu ekrana koja bi, na primjer, mogla prikazivati položaje igrača i lopte [gueziec02], koji najčešće nisu svi vidljivi iz tekućeg pogleda. U ambicioznijem duhu, dobiveni podatci bi mogli biti upotrijebljeni za poluautomatsku režiju prijenosa. Takva primjena bi podrazumijevala automatizirani upravljački sustav koji bi obavljao selekciju tekućeg pogleda te podešavanje smjerova gledanja dostupnih kamera u cilju postizanja prihvatljivog prikaza susreta.

Predloženi eksperimentalni problem je izravno inspiriran od strane još jedne važne primjene porazdijeljenog praćenja, a to je pružanje lokalizacijskih usluga grupi skromno opremljenih pokretnih autonomnih robota (vidjeti sl. 1.1). Takav pristup se u literaturi naziva globalnim vidom [kay93, veloso98], porazdijeljenim vidom [sogo99], odnosno senzorskom mrežom za pokretnu robotiku [hoover00]. Globalni vid, kao metoda određivanja položaja pokretnog robota, je svrstan u grupu s lokalizacijom uz pomoć referentnih objekata [borenstein96], jer zahtjeva posebne intervencije u okolišu u kojem se odvija navigacija. Pristup je posebno pogodan za primjene koje zahtijevaju velik broj autonomnih vozila (npr, automatizirano skladište), jer omogućava uštedu na varijabilnim troškovima naprednih senzora na račun fiksnih troškova infrastrukture globalnog vida [kay93]. U posljednje vrijeme, globalni vid se često koristi u utakmicama lakših kategorija robotskog nogometa, kao metoda za određivanje položaja pojedinačnih "igrača" (pogledati npr. [veloso98]).

1.4 Struktura rada

Opisuje se arhitektura za porazdijeljeno praćenje višestrukim promatračima koja bi mogla omogućiti lokalizaciju grupe pokretnih robota metodom globalnog vida. Arhitektura podrazumijeva eksplicitni ukupni prikaz scene, te pretpostavlja ravninski model gibanja praćenih objekata. Razmatraju se teoretski aspekti ostvarivanja kooperativnog inteligentnog ponašanja, kao i praktični problemi pri izgradnji sustava s takvim svojstvima.

U poglavlju 2, prikazani su do sada postignuti rezultati u srodnim istraživačkim područjima: višeagentskim sustavima, porazdijeljenom praćenju te lokalizaciji globalnim vidom. Teorijski temelji trodimenzionalnog računarskog vida koji su neophodni pri kalibraciji opreme i interpretaciji rezultata, su dani u poglavlju 4. Temeljna višeagentska arhitektura sustava te njena proširenja u smislu višerazinske organizacije i mogućnosti uvođenja pokretnih promatrača kao proširenja temeljne teorije su izloženi u poglavlju 3, dok poglavlje 5 opisuje implementacijske detalje eksperimentalne izvedbe pojedinih komponenti sustava. Poglavlje 6 sadrži prikaz postignutih rezultata na pojedinim razinama sustava, dok se u poglavlju 7 nalazi kratka rasprava o postignutim rezultatima i mogućim smjerovima daljnjeg istraživanja.

Poglavlje 2

Pregled srodnih istraživačkih područja

U kontekstu ovog rada, razmatraju se različiti aspekti suradnje u sustavu za porazdijeljeno praćenje pokretnih objekata višestrukim promatračima. Posebno je zanimljivo ostvarivanje suradnje među tzv. aktivnim promatračima koji su opremljeni upravljivim kamerama, čiji je smjer gledanja moguće podešavati u skladu s razvojem događaja u sceni. Pokazuje se da takvi promatrači moraju imati stanoviti stupanj autonomije pa će arhitekturu takvog sustava biti prirodno izraziti u okviru višeagentske teorije. Konačno, pri oblikovanju eksperimentalnog sustava, koristit će se pretpostavke koje diktira područje primjene, pa će biti zanimljive i osnovne postavke metode lokalizacije jednostavnih robota temeljene na zamisli globalnog vida.

Glavni dijelovi ovog poglavlja stoga su pregledi istraživanja na područjima višeagentskih sustava i porazdijeljenog praćenja višestrukim promatračima. Odjeljak o porazdijeljenom praćenju zaključen je kraćim pregledom osnovnih tipova arhitektura za porazdijeljeno praćenje, te mogućnostima za njihovu primjenu u željenom eksperimentalnom sustavu. Pored toga, dani su i kraći pregledi zanimljivijih ostvarenja na područjima ostalih primjena višeagentskog računarskog vida i lokalizacije globalnim vidom.

2.1 Višeagentski sustavi

Višeagentska organizacija [wooldridge99] je u posljednje vrijeme postala često citirana arhitektonska paradigma na području razvoja programske podrške. Kod višeagentskog pristupa, značajan dio funkcionalnosti sustava se postiže interakcijom autonomnih antropomorfnih procesnih entiteta ili agenata. Pristup je posebno prikladan kod problema koji su ili inherentno porazdijeljeni, pa se autonomni čimbenici ističu već u specifikaciji, ili iznimno složeni te ih je pogodno izraziti kroz što autonomnije specijalizirane podsustave koji se mogu neovisno razvijati i ispitivati. Tako se kao primjeri upotrebe višeagentskih sustava u industrijskim sredinama [chaib95] navode suradnja više ekspertnih sustava u stvarnom vremenu, dijagnostički sustav složenog postrojenja te dodjeljivanje resursa u automatiziranoj proizvodnji.

2.1.1 Motivacija

Wegner [wegner95, wegner97] kritizira tradicionalni proceduralni pristup programiranju, tzv algoritamsku programsku paradigmu, u smislu nemogućnosti izražavanja zadataka koji tijekom izvođenja inherentno ovise o *drugim čimbenicima* u okolini. Kao primjer, navodi se zadatak "odvesti se s posla kući", kojeg je vrlo teško (po autorima nemoguće) izraziti

algoritmom, iako je izrečen u proceduralnom stilu. Stoga se uvode *objekti*, kao elementi bolje, *interaktivne* paradigme programiranja. Odnos između objekata i algoritama se ilustrira primjerom, pa se navodi da analogan odnos postoji između "bračnog ugovora" koji podrazumijeva dugotrajnu interakciju s ostalim ugovornim stranama, i "kupoprodajnog ugovora" kod kojeg se interakcija svodi na diskretan trenutak samog potpisivanja ugovora. Objektni pristup je bolji jer omogućava izražavanje ugovora prema kojim se zainteresirane strane dinamički prilagođavaju stanju ostalih prilika u okolišu. Huhns i Singh [huhns99a] navode kako se analogan "prijelaz paradigme" događa i na (hijerarhijski višem) području arhitekture inteligentnih sustava, od tradicionalne monolitne arhitekture prema višeagentskoj organizaciji. Interakcija među ravnopravnim programskim entitetima prema njima je ključno svojstvo višeagentskog sustava pa se, u duhu Turingovog testa inteligencije[1], višeagentska paradigma opisuje agentskim testom [huhns97, huhns99a]:

> Sustav koji sadrži jednog ili više priznatih agenata bi se trebao značajno promijeniti, ukoliko se još jedan priznati agent uvede u sustav.

Drugim riječima, da bismo sustav mogli nazvati višeagentskim, njegove komponente (agenti) bi se trebale ponašati kao da su svjesne međusobnog prisustva. Motivacija za takvu organizaciju je upravo mogućnost modeliranja interaktivnih i dinamičkih sustava, koji bi mogli obavljati složene zadatke koji u sadašnjosti ljudi obavljaju mnogo bolje od računala. Naravno, višeagentska paradigma neće napraviti takve zadatke jednostavnima, nego će možda pružiti pretpostavke za njihovo jednostavnije formalno izražavanje.

2.1.2 Terminologija

Jedan od problema s kojima se susreće početnik na području višeagentske teorije je učestalost upotrebljavanja riječi agent u različitim kontekstima. Nesporazum je uvjetovan višeznačnošću riječi agent koja i u britanskoj [cambridge:www] i u američkoj [webster:www] inačici engleskog jezika ima dva značenja:

1. "onaj kojemu je dozvoljeno da djeluje umjesto nekog drugog (predstavnik, zastupnik)".
2. "onaj koji djeluje, primjenjuje silu (činioc)",

Agent–činioc preciznije slijedi latinski korijen te vrlo dobro odgovara značenju koje se koristi i u teoriji višeagentskih sustava. Međutim, agent–zastupnik se kolokvijalno češće koristi, u kontekstu kupoprodajnih, turističkih, tajnih i ostalih agenata. To potvrđuje i rječnik računarskih pojmova [howe:www] u kojem se oba ponuđena značenja odnose na program koji obavlja neki zadatak u ime korisnika ili operacijskog sustava, npr. "mail agent", "news–reader agent", "SNMP agent" i sl. Da bi zbrka bila veća, agenti–zastupnici su došli i do stručne i znanstvene literature: tako se navodi [karnik98] da je agent bilo koji program koji djeluje u ime korisnika.

Međutim, čak ni u okviru višeagentske teorije ne postoji konsenzus oko precizne definicije agenta osim što se intuitivno podrazumijevaju autonomija i sposobnost komuniciranja. Tako Shoham [shoham93] navodi da je agent entitet koji ima svoja vjerovanja, mogućnosti, izbore i obaveze. Ovakva apstraktna definicija obuhvaća najrazličitije naprave, uključujući i najobičniji prekidač, ali to po autoru nije problem jer takav opis ne bi donio korisna nova

[1] Navedeni test je sličan Turingovom testu utoliko što se jedno vrlo složeno svojstvo definira preko manifestacije u zamišljenom eksperimentu, koji se odvija u intuitivno pristupačnom okruženju. Manifestacija se pri tome implicitno promovira kao glavna odlika, srž razmatranog svojstva.

svojstva uobičajenom modelu prekidača. Ferber [ferber99] pokušava biti određeniji pa donosi dugu listu svojstava koja bi priznati agent morao zadovoljavati. Međutim, neka od tih svojstava su međusobno redundantna, dok su druga specifična samo za neke primjene, kao na primjer mogućnost reprodukcije. Genesereth i Ketchpel [genesereth97] prebacuju naglasak sa samog agenta na višeagentski komunikacijski protokol koji je propisan agentskim komunikacijskim jezikom (ACL, agent communication language). Tako se kaže da je neki entitet agent, ako i samo ako ispravno komunicira u skladu s važećim ACL-om. To je zgodan način da se u višeagentskom kontekstu naglasi važnost komunikacije, ali nije sasvim jasno kako bi se u takvu definiciju uklopili reaktivni agenti kod kojih eksplicitna komunikacija ne postoji. Huhns i Singh [huhns97] uvažavaju činjenicu da je područje još nezrelo za precizne definicije, pa pojam agenta opisuju spomenutim agentskim testom.

Najupotrebljiviju definiciju agenta nudi [wooldridge99], gdje se navodi da je agent računalni sustav koji poduzima autonomne akcije u svom okolišu kako bi postigao svoje ciljeve. Ključne sposobnosti inteligentnog agenta tako su:

- **djelovanje usmjereno prema cilju**: agent je sposoban preuzeti inicijativu u smislu zadovoljenja svojih ciljeva;
- **reaktivno djelovanje**: agent opaža događaje u okolini, i po potrebi pravovremeno reagira u smislu zadovoljenja svojih ciljeva;
- **sposobnost interakcije**: agent je po potrebi sposoban komunicirati s ostalim agentima u smislu zadovoljenja svojih ciljeva.

Pokazuje se da oblikovanje čisto reaktivnih i sasvim usmjerenih sustava (npr. termostat odnosno prevodioc) nije toliko teško, odnosno da se može postići postojećim metodologijama bez previše poteškoća. Međutim, problemi se javljaju kod primjena gdje je potrebno pažljivo odvagnuti odnosno naći mjeru između reaktivnog i usmjerenog djelovanja. Upravo takve primjene razmatra višeagentska teorija, a obično su povezane s iznimno složenim (realističnim!) okolinama koje su presložene i prebrzo se mijenjaju da bi se unaprijed modelirale, a tražena funkcionalnost je prezahtjevna da bi se modelirala reaktivnim ponašanjima.

2.1.3 Glavni problemi i ograničenja

Pored problema s terminologijom, primjena višeagentske paradigme u praksi se susreće s brojnim drugim problemima. Bradshaw et al. [bradshaw99] tako konstatiraju da se malobrojni stvarno izgrađeni višeagentski sustavi ne ponašaju u skladu s vizijama iz istraživačkih projekata: nisu robustni na promjene u okolišu te surađuju po trivijalnim i predodređenim komunikacijskim obrascima. Ocjenjuje se da su za omogućavanje bržeg prodora u svijet industrijskih primjena potrebna dodatna poboljšanja na područjima višeagentske teorije, programske infrastrukture te korisničkih alata.

Problem s višeagentskom paradigmom je u tome što nudi obrasce za rješenje problema kroz interakciju jednostavnijih programskih jedinica, dakle jednu nadgradnju nad postojeću funkcionalnost. Međutim da bi se fokus razvoja prenio s funkcionalnosti agenata na funkcionalnost sustava, postojeće funkcionalnosti moraju biti dobro razrađene, drugim riječima, područje primjene mora biti *zrelo* za proširenje. Na nesreću, upravo to područje, analiza domene budućeg računalnog sustava [czarnecki00] je iznimno teško, pogotovo za interaktivne okoline, pa je najčešće potrebno više razvojnih iteracija da bi se došlo do upotrebljive baze za višeagentsku nadgradnju.

Područje primjene višeagentske teorije je ograničeno na one probleme kod kojih se značajan dio rješenja da izraziti kroz agentsku interakciju. Ti problemi obuhvaćaju situacije

u kojima se nezavisni entiteti natječu za ograničene resurse (npr. vrijeme specijalnog stroja u sustavu za upravljanje automatiziranom proizvodnjom [chaib95]), zajednički upotpunjuju rješenje asinkronim doprinosima na hijerarhijski različitim razinama rješenja (npr. prepoznavanje glasova, slogova, riječi u sustavu za raspoznavanje govora [huhns99a]), ili međusobno potpomaganje u preciznijem određivanju položaja praćenih objekata [lesser83].

2.1.4 Elementi za izgradnju višeagentskih sustava

Ključni elementi pri oblikovanju višeagentskog sustava su odabir arhitekture agenata te modela njihove međusobne interakcije i odgovarajućeg komunikacijskog protokola, a ovise izravno o području primjene. Međutim, da bi se različite arhitekture i protokoli mogli razmatrati u općenitom kontekstu, potrebno je prvo definirati zajednički okvir, u obliku apstraktnog opisa agenta.

Apstraktan opis agenta

Sasvim općenito, agent je entitet koji obavlja sljedeće operacije: opažanje okoline, obnavljanje unutrašnjeg stanja i djelovanje [wooldridge99]. Definirajmo sljedeće apstraktne skupove:

S: skup svih stanja okoline;

P: skup svih mogućih percepcija okoline;

I: skup svih unutrašnjih stanja agenta;

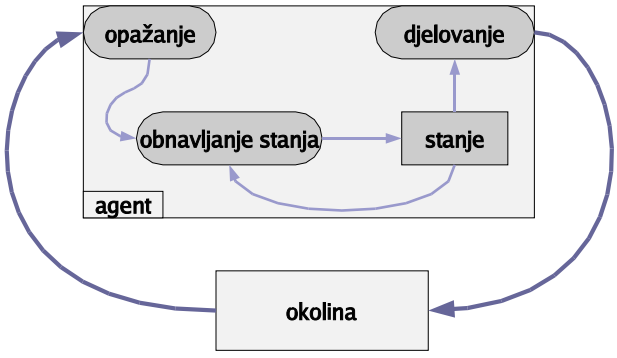
A: skup svih akcija agenta.

Tada se funkcionalnost agenta može definirati preko spomenutih operacija, kao što je prikazano u tablici 2.1.

operacija	opis	definicija
gledaj	opažanje	$S \rightarrow P$
razmisli	obnavljanje stanja	$I \times P \rightarrow I$
djeluj	odabir akcije	$I \rightarrow A$

Tablica 2.1: Operacije apstraktnog agenta.

Apstraktna funkcionalnost agenta se može opisati ciklusom *gledaj* - *razmisli* - *djeluj*, kao što je zornije ilustrirano na sl. 2.1. Spomenuti ciklus je prije svega konceptualan, jer se u izvedbi



Slika 2.1: Shema apstraktnog agenta.

može pokazati korisno da se tri procesa usporedno izvršavaju [matsuyama02]. Takve situacije se mogu dogoditi kada se percepcija izvodi u diskretnim epizodama, kao što je tipično slučaj kod računarskog vida, dok je priroda primjene takva da je akcije potrebno poduzimati češće. Akcije se mogu poduzimati i onda kada novi rezultati percepcije nisu dostupni jer je ocjenu trenutnog stanja scene moguće ekstrapolirati na temelju dotadašnje dinamike i podatka o stvarnom vremenu.

Često se u literaturi [matsuyama98, huhns99] može naći podjela agenata na utjelovljene (*engl.* embodied agent), i beztjelesne (*engl.* vacuous agent) ili programske agente (*engl.* software agent). Navodi se razlika da programski agenti percipiraju okolinu isključivo putem primljenih poruka (ne očitanjem stvarnih senzora), te da djeluju na okolinu slanjem poruka drugih agentima (ne upravljanjem efektorima). Ta podjela je uglavnom beskorisna jer se bez smanjenja općenitosti svaki utjelovljeni agent da opisati programskim agentom koji šalje poruke upravljačkim programima efektora, a prima poruke od strane upravljačkog programa senzora. Valja primijetiti da tako biva i u stvarnosti, jer je kôd koji upravlja fizičkim ulazno–izlaznim vratima obično više programskih slojeva udaljen od kôda za kojeg se smatra da definira agenta.

Arhitekture agenata

Glavni čimbenici pri izgradnji agenata za neki konkretan zadatak su okolina u kojoj se agenti nalaze (skup S), i sam zadatak agenta (skup A). Na temelju tih skupova, određuje se postupak opažanja agenta (operacija *gledaj*) te očekivani skup opažaja P. Sam postupak opažanja se u ovom kontekstu obično ne raščlanjuje dalje, iako se radi o vrlo teškom problemu. Konačno, oblikuje se prikladna struktura unutrašnjeg stanja agenta (elementi skupa I) te postupci njegovog obnavljanja (operacija *razmisli*) i donošenja odluke o akciji (operacija *djeluj*).

Za arhitekturu agenta, ključan je odabir formalizma za prikaz unutrašnjeg stanja, koji mora biti dovoljno sofisticiran da omogući željeno ponašanje agenta. Različiti prikazi stanja će uvjetovati i različite arhitekture, pa se tako u literaturi mogu naći i sljedeći tipovi agentskih arhitektura [wooldridge99]:

- arhitekture temeljene na logici:
  - trenutno stanje agenta ($x \in I$) se modelira kao skup klauzula predikatne logike;
  - operacija *djeluj* se svodi na poznate metode zaključivanja u predikatnoj logici;
  - prednosti: teoretska utemeljenost
  - nedostatci: računska složenost postupaka zaključivanja, poluodlučljivost predikatne logike.
- reaktivne arhitekture:
  - agenti bez unutrašnjeg stanja ($I = \emptyset\varnothing$);
  - odluka o akciji se donosi samo na temelju trenutno opaženog stanja okoline: *djeluj*:$P \to A$;
  - postupak odluke se da izraziti kao skup "ako–onda" pravila, a svako od njih definira po jednu komponentu ukupnog ponašanja agenta;
  - prednosti: jednostavnost, robustnost, elegancija;
  - nedostatci: ograničeno područje primjene.

- arhitekture BDI (*engl.* Belief–Desire–Intention):
  - stanje agenta se sastoji od dva elementa $s = (s_B, s_I) \in I$:

    s_B: vjerovanje o stanju okoline,
    $s_B \in \wp(\text{Bel})$, Bel je skup svih vjerovanja[2];

    s_I: skup usvojenih namjera,
    $s_I \in \wp(\text{Int})$, Int je skup svih namjera;

    Vrijedi: $I = \wp(\text{Bel}) \times \wp(\text{Int})$
  - operacija *razmisli* obnavlja vrijednosti svih elemenata stanja funkcijama:
    - revizija vjerovanja o stanju okoline
      BRF: $\wp(\text{Bel}) \times P \to \wp(\text{Bel})$,
    - generiranje želja na temelju vjerovanja i namjera
      OPTIONS: $\wp(\text{Bel}) \times \wp(\text{Int}) \to \wp(\text{Des})$, Des je skup svih mogućih želja (one su konkretnije od tekućih namjera),
    - promoviranje odabranih želja u namjere, odbacivanje nerealnih namjera
      FILTER: $\wp(\text{Bel}) \times \wp(\text{Des}) \times \wp(\text{Int}) \to \wp(\text{Int})$;
  - ključna je funkcija OPTIONS jer modelira usmjereno rasuđivanje: izražavanje apstraknih namjera konkretnim željama (odabrane želje mogu i same postati namjere i u sljedećem ciklusu biti razložena na još konkretnije želje);
  - prednosti: intuitivnost, jasna funkcijska podjela;
  - nedostatci: ne specificira se način implementacije.
- višeslojne arhitekture:
  - funkcionalnost agenta se postiže suradnjom nezavisnih eksperata koji obavljaju operaciju *razmisli*, a organizirani su u više slojeva;
  - više izvedbenih inačica:
    - *horizontalna organizacija*: agenti paralelno analiziraju percipirani okoliš i odabiru potrebne akcije, konflikti se razrješavaju ili centraliziranim arbitrom ili čvrstim prioritetima,
    - *vertikalna organizacija*, agenti su organizirani u više slojeva uz porast apstrakcije prikaza unutrašnjeg stanja, najdonja razina analizira stanje agenta, te ukoliko se ne može odlučiti za odabir akcije, prepušta odluku agentima na višim razinama;
  - prednosti: konfigurabilna funkcijska podjela;
  - nedostatci: problem arbitraže.

Detaljnija razrada agentskih arhitektura iz pojedinih grupa je izvan dosega ovog rada, a više detalja se može naći u literaturi [wooldridge99].

Elementi komunikacijskog protokola

Većina autora se slaže kako je društvena aktivnost ključno svojstvo agenta. Stoga su u kontekstu višeagentske teorije, tehnike za tvorbu poruka iznimno bitne. Najčešće je prikladno

[2] $\wp(X)$ označava skup svih podskupova ili partitivni skup od X.

temeljiti komunikaciju na mrežnom protokolu iz porodice TCP/IP, jer se time dobiva transparentnost korisničkog kôda o tehnologiji fizičkog sloja (ožičeni spoj ili radio veza). Nadalje, pogodno je koristiti neki od standardnih agentskih komunikacijskih jezika [chaib02]) (*engl.* ACL – Agent communication language), koji propisuju *vanjske* formate poruka, uz mogućnost korištenja različitih semantičkih formalizama. Najčešće korišteni ACL standard još uvijek je KQML [huhns99], dok se u novije vrijeme sve češće koriste specifikacije Zaklade za inteligentne fizičke agente FIPA [steiner98]. Sama semantika poruke također može biti predstavljena standardnim formalizmom, pri čemu se najčešće koriste predikatna logika ili njena proširenja npr. KIF [huhns99]. Korištenje standardnih formata poruka je posebno prikladno za zrele sustave u fazi produkcije jer standardi tada garantiraju otvorenost sustava, odnosno mogućnost njegovog proširenja s agentima koji će biti tek naknadno zamišljeni i isporučeni.

Značenje poruka može biti klasificirano s obzirom na tri osnovna tipa: izjava, pitanje i zahtjev. Robustni komunikacijski protokoli će uglavnom koristiti izjave i upite, na način koji ne obavezuje primaoca poruke (npr: predloži akciju, prihvati prijedlog, odbij prijedlog). Agente možemo klasificirati prema vrstama poruka koje mogu primati odnosno slati, kao što je navedeno u tablici 2.2. Slijedi popis češće korištenih poruka:

- izjave: odgovor, natječaj, prihvaćanje, odbijanje, objašnjenje
- pitanja: upit, ponuda
- zahtjevi: zahtjev, naredba

svojstvo	jednostavni agent	pasivni agent	aktivni agent	ravnopravni sudionik
prima izjave	•	•	•	•
prima upite		•		•
šalje izjave		•	•	•
šalje upite			•	•

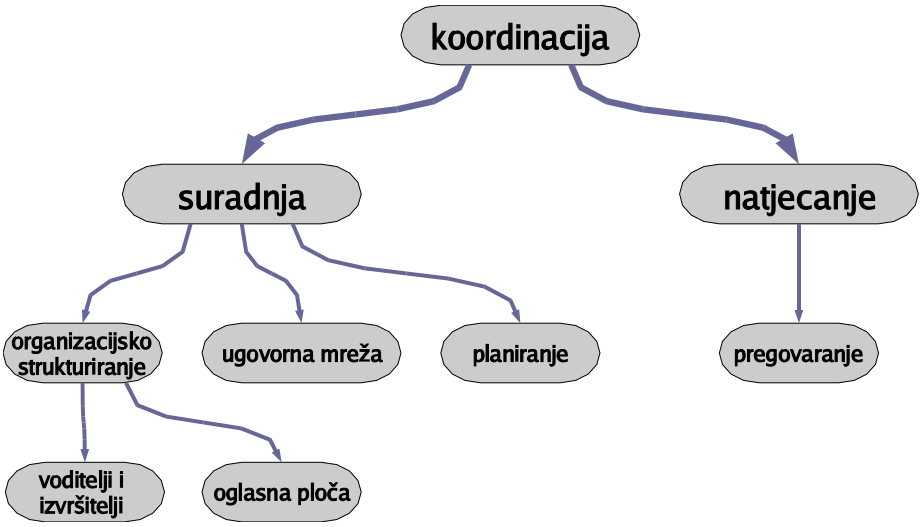
Tablica 2.2: Podjela agenata prema ulozi u komunikacijskom protokolu.

Oblici višeagentske koordinacije

U višeagentskom sustavu, agenti komuniciraju u cilju postizanja svojih osobnih ciljeva ili ciljeva sustava kao cjeline. Krajnji cilj komunikacije je u najvećem broju slučajeva koordinacija pojedinačnih akcija u cilju postizanja koherentnijeg sustava. Sustav je tim koherentniji, bolje koordiniran, što se bolje zaobilaze neželjene situacije, kao npr, sukob oko resursa, potpuni zastoj, neravnomjerna raspodjela agenata u okolini. Koordinacijski detalji su suštinsko svojstvo višeagentskog sustava pa se može reći da se višeagentska arhitektura uglavnom svodi na oblikovanje koordinacije među agentima.

Dva temeljna tipa koordinacije su suradnja (kooperacija) i natjecanje. Kod suradnje, agenti zajednički rade na postizanju ciljeva sustava, dok natjecanje pretpostavlja međusobno suprotstavljene (antagonističke) ili jednostavno sebične agente. Temeljni oblici koordinacije u višeagentskim sustavima su dani na sl. 2.2. U nastavku će biti dan samo kratki opis pojedinih shema za ostvarivanje sustava temeljenih na kooperativnoj interakciji, dok se više detalja se može naći u [nwana96], [huhns99] i [snajder04].

- organizacijsko strukturiranje:



Slika 2.2: Podjela oblika koordinacije u višeagentskim sustavima [nwana96, huhns99].

  - koristi se apriorna (statička) organizacijska struktura, koja propisuje okvir za interakciju kroz definiciju uloga, komunikacijskih puteva i hijerarhijskih odnosa;
  - tehnika je prikladna za sustave u kojima se broj agenata relativno malo mijenja (npr, kada su agenti odgovorni za fizičke resurse);
  - više izvedbenih inačica:
    * *hijerarhijska struktura*, kod koje nadređeni agent dijeli poslove podređenim agentima, te integrira njihove rezultate u konačno rješenje,
    * *obrazac oglasne ploče* (*engl.* blackboard) [pfleger98, huhns99], u kojem podređeni agenti upisuju parcijalne rezultate na zajedničku oglasnu ploču, dok upravljački agent prati napredovanje rješenja i po potrebi aktivira odnosno suspendira podređene agente;
  - nema načina za rekonfiguraciju sustava: područje je na granici višeagentskih i klasičnih porazdijeljenih sustava;
  - prednosti: efikasnost, jednostavnost;
  - nedostatci: ograničeno područje primjene, centralizirano upravljanje može postati usko grlo;

- ugovorna mreža:
  - poopćenje fiksne hijerarhijske organizacije na način da se omogućuje dinamička rekonfiguracija sustava;
  - prilikom pojave svakog novog zadatka u sustavu:
    * odgovorni agent particionira zadatak i raspisuje natječaj za rješavanje svakog od podzadataka,
    * agenti koji imaju mogućnosti za obavljanje raspisanog posla se javljaju na natječaj,
    * poslovi se dodjeljuju najpovoljnijim ponuđačima koji ih rješavaju vlastitim snagama ili dalje dijele uz raspisivanje novog natječaja,
    * nakon što su svi podzadatci obavljeni, agent koji je raspisao natječaj integrira parcijalne rezultate u konačno rješenje;

  - prednosti: dinamičko dodjeljivanje zadataka, prilagodljivost;
  - nedostatci: intenzivno komuniciranje može postati usko grlo;

- višeagentsko planiranje:
  - prikladno kada se sustav sastoji od više agenata čiji ciljevi su jasni, a problem je doći do plana koji razrješava konflikte pri njihovom zadovoljenju;
  - u prvoj fazi, svaki od agenata generira početni plan za zadovoljenje svojih ciljeva
  - dvije osnovne inačice druge faze:
    * *centralizirano višeagentsko planiranje*, gdje se pojedinačni planovi šalju centraliziranom arbitru koji ih onda usaglašava,
    * *porazdijeljeno višeagentsko planiranje*, gdje agenti međusobno komuniciraju i pri tom modeliraju planove ostalih agenata te modificiraju svoje planove na način da se ne kose s planovima ostalih;
  - prednosti: korištenje poznatog formalizma iz klasične umjetne inteligencije, prilagodljivost pri razrješavanju konflikata;
  - nedostatci: neefikasnost uslijed velikog opsega kućanskih poslova.

Nabrojeni temeljni obrasci višeagentske suradnje prekrivaju kontinuum mogućnosti od efikasnog, jednostavnog i nefleksibilnog (organizacijsko strukturiranje) do neefikasnog, složenog i vrlo fleksibilnog ponašanja (porazdijeljeno višeagentsko planiranje). O samoj konačnoj primjeni će ovisiti da li je problem moguće modelirati jednostavnijom interakcijom uz visoku efikasnost i slabiju prilagodljivost, ili će biti potrebna složenija interakcija i veća prilagodljivost uz gubitke na performansi sustava.

2.2 Područja primjene višeagentskog računarskog vida

Pored porazdijeljenog praćenja agentima–promatračima koje će biti razmatrano u sljedećem odjeljku, u literaturi se mogu naći višeagentski pristupi rješavanju nekih drugih područja računarskog vida. Iako je višeagentski pristup prikladan za razmatranje upravo takvih, složenih problema s više mogućih međusobno nezavisnih puteva k rješenju (npr, detekcija rubova, regija, pokreta), prethodna istraživanja nisu odmakla dalje od pojedinačnih eksperimentalnih sustava s ograničenim mogućnostima. Glavni razlog za relativno slabu dinamiku višeagentskog računarskog vida je u tome što višeagentska teorija nudi "samo" arhitektonske obrasce, a računarski vid je još uvijek relativno nezrelo područje u kojem se standardni postupci, čija svojstva bi u kontekstu nekog problema trebala favorizirati jedan ili drugi obrazac, tek počinju kristalizirati. Višeagentska teorija može ponuditi recept za udruživanje nezavisnih eksperata, ali ako ne postoje standardi što točno koji od eksperata može i treba napraviti, razvijene arhitekture je jako teško ponovno upotrijebiti pa je njihov značaj vrlo ograničen. Apstraktne višeagentske arhitekture u računarskom vidu stoga je vrlo teško i nezahvalno istraživati pa se u literaturi takvi pokušaji vrlo rijetko nalaze, a većina radova se usko fokusira na specijalizirane probleme kao što su segmentacija slike, prepoznavanje objekata, i praćenje objekata u pojedinačnim slijedovima slika.

2.2.1 Integracija heterogenih modula u složeni sustav

Bianchi i Rillo [bianchi96] su opisali višeslojnu višeagentsku arhitekturu računarskog vida, koja se temelji na Aloimonosovoj paradigmi svrhovitog vida. Prema toj paradigmi, rekonstrukcija scene se izbjegava kao pretežak i loše definiran problem, dok se funkcionalnost sustava usko vezuje uz njegove zadatke u fizičkom svijetu. U radu se razmatra domena upravljanja robotske ruke uz pomoć jedne nepokretne kamere, a opažanje okoline se svodi na postupke pronalaženja karakterističnih objekata i detekciju pokreta u sceni. Predloženi sustav se sastoji od tri agenta koji modeliraju tri osnovna ponašanja s fiksnim rastućim prioritetom: detekciju mogućeg sudara, pripremu radnog prostora i obavljanje željene operacije. Antagonistički agenti su organizirani u horizontalnoj arhitekturi, na način da u svakom trenutku zainteresirani agent s najvišim prioritetom dobiva pravo upravljanja robotskom rukom. Ocijenjeno je da je glavna prednost višeagentskog pristupa opis sustava u terminima ponašanja i zadataka, što omogućava eksplicitan opis interakcije između pojedinih ponašajnih modula.

Jedan od teoretski najsofisticiranijih višeagentskih sustava računarskog vida [graf00] je organiziran kao potpuno povezana ugovorna mreža agenata koji se dinamički organiziraju metodom raspisivanja natječaja. Sustav se sastoji od dvije temeljne vrste agenata: organizatora i radnika, a glavna razlika je u tome što organizatori mogu dio posla prepustiti nekom drugom agentu, koji bi se javio na raspisani natječaj. Nakon unošenja zadatka, sustav se samostalno organizira u hijerarhijsku stablastu strukturu, u kojoj svaki čvor predstavlja agenta organizatora koji je dijelove svog posla rasporedio drugim agentima. Korisnički upiti se u sustav unose putem specijalnog agenta komunikatora, u skladu sa sintaksom jezika za opis produkcijskih sustava CLIPS.

Autori su mogućnosti sustava opisali na primjeru sljedećeg zadatka: "pronađi sve šipke (*engl.* ledge) i crvene objekte u slici pribavljenoj sa servera Penelopa". Taj upit može se izraziti sljedećom konstrukcijom u CLIPS-u:

```
(extract ?dst ?src)
  (is-a ?dst object)
    (or (has-name ?dst ledge) (has-color ?dst red))
  (is-a ?src image)
    (has-source ?src camera)
    (has-server ?src penelope)
```

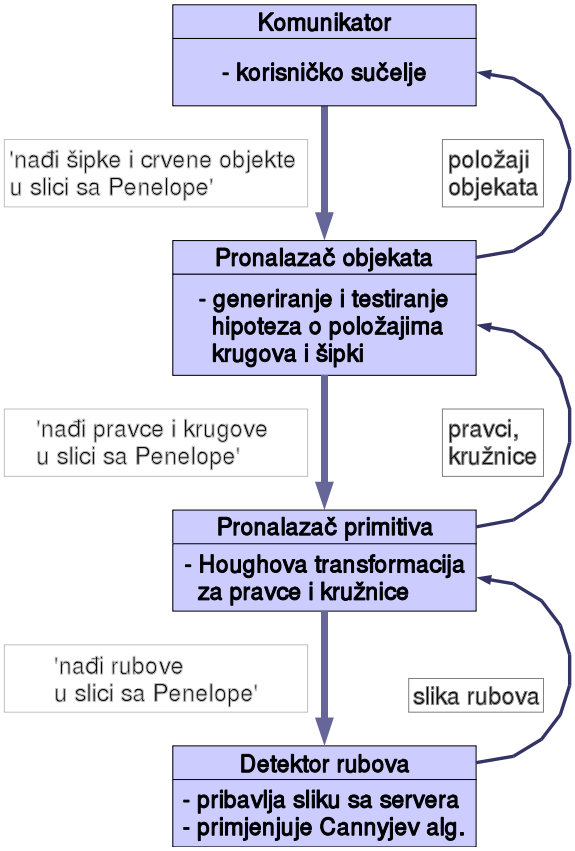
Nakon što je primio upit, agent komunikator raspisuje natječaj na kojem pobjeđuje agent za prepoznavanje objekata (zbog jednostavnosti, prepoznaju se samo šipke i krugovi) koji na ulazu treba popis geometrijskih primitiva, pa raspisuje novi natječaj:

```
(extract ?dst ?src)
  (is-a ?dst feature)
    (or (has-type ?dst line) (has-type ?dst circle))
  (is-a ?src image)
    (has-source ?src camera)
    (has-server ?src penelope)
```

Na drugi natječaj se javlja agent za pronalaženje krugova i dužina. On međutim zna raditi samo sa slikom u kojoj su rubovi već segmentirani, pa se raspisuje treći natječaj:

```
(extract ?dst ?src)
  (is-a ?dst image) (has-type ?dst edge)
  (is-a ?src image)
    (has-source ?src camera)
    (has-server ?src penelope)
```

Na treći natječaj se javlja agent za rubnu segmentaciju koji angažira agente radnike za pribavljanje izvorne slike i primjenu Cannyjevog rubnog detektora i vraća segmentiranu sliku agentu za pronalaženje geometrijskih primitiva. Taj agent detektira krugove i dužine te ih isporučuje agentu za prepoznavanje objekata koji sada ima sve preduvjete za obavljanje svog zadatka. Konačno, agent za prepoznavanje dojavi položaje i opise nađenih objekata komunikacijskom agentu koji rezultate prikaže korisniku i time obrada upita završava. Postupak stvaranja dinamičke hijerarhijske strukture za rješavanje zadanog problema u mnogome podsjeća na zaključivanje s ulančavanjem unatrag u produkcijskom sustavu. Dinamički stvorena hijerarhijska struktura je prikazana na sl. 2.3. Predloženom organizacijom postiže se eks-



Slika 2.3: Hijerarhijska struktura stvorena ugovornom mrežom (detaljnije u tekstu).

plicitno odvajanje područja odgovornosti pojedinih elementarnih algoritama što, u slučaju njihovog umješnog odabira, omogućava elegantnu rekonfiguraciju sustava i velike mogućnosti za višestruku ponovnu upotrebu dijelova kôda.

2.2.2 Raspoznavanje objekata suradnjom promatrača

Oswald i Levi [oswald01] su opisali pristup za prepoznavanje objekata grupom promatrača. Pristup se temelji na uvažavanju velikih razlika u izgledu 3D objekata kod prostorno udaljenih točaka promatranja, pa promatrači izvode prostornu integraciju na razini cjelokupnih objekata. Navodi se kako je takav pristup u prednosti pred drugom mogućnošću, ostvarivanjem korespondencije na razini komponenata objekata, utoliko što postiže rezultate i u dijelovima scene koji su vidljivi samo jednom promatraču, što je posebno bitno kod stvarnih primjena. Za razliku od prethodnih radova, gdje se korespondencija na razini objekata ostvaruje uglavnom pri "primopredaji" područja pažnje, kad bi se pokretni objekt približio granici

područja odgovornosti dvaju promatrača, u predloženom radu suradnja se koristi za verifikaciju hipoteza o identitetu i orijentaciji objekta. Svaki od promatrača pokušava prepoznati objekt statističkom metodom koja se temelji na izgledima objekta iz ravnomjerno raspoređenog skupa smjerova objekata. Fuzija dobivenih hipoteza se obavlja Bayesovom mrežom, uz korištenje apriornog znanja o položaju i orijentaciji pojedinih promatrača. Eksperimenti su pokazali se da se upotrebom većeg broja promatrača postiže znatno veća robustnost raspoznavanja u odnosu na konfiguraciju sa samo jednim promatračem.

2.2.3 Višeagentska obrada slike

Najviše radova na području višeagentskog računarskog vida razmatra paralelnu obradu slike nezavisnim agentima. Kod tih pristupa, svaki agent djeluje lokalno i po potrebi komunicira s ostalim agentima u svojoj blizini, a tipična operacija je klasificiranje slikovnih elemenata u skladu sa zadatkom sustava. Glavna područja primjene takvih sustava možemo podijeliti prema tome da li obrađuju jednu sliku ili cijeli niz slika, što se onda svodi na postupke segmentacije odnosno praćenja objekata (uz jednog promatrača i jedan niz slika).

Segmentacija slike

Sustav koji su predložili Liu i Tang [liu99], podrazumijeva velik broj reaktivnih agenata koji operiraju izravno po susjedstvu elementa slike na kojem se trenutno nalaze. Na početku simulacije, u slici se stvori početni skup slučajno raspoređenih agenata sa slučajno pridijeljenim smjerovima kretanja. U ovisnosti o vrijednosti slikovnih elemenata u svom neposrednom susjedstvu, agenti odabiru jedno od dva temeljna ponašanja, razmnožavanje ili traženje. Traženje se primjenjuje kada se agent nalazi na slikovnim elementima s nehomogenim susjedstvom. Kod tog ponašanja, agenti se pomiču u skladu sa smjerom kretanja, dok im ne istekne životni vijek, unaprijed zadan brojem "ciklusa" simulacije. Kada agent dođe na slikovni element s homogenim susjedstvom, prvo označava takav element kao segmentiran, te stvara svoje kopije u svim homogenim elementima susjedstva. (kopije nasljeđuju smjer kretanja od roditelja). Na taj način, homogene regije bivaju jednako označene, i time se postupak segmentacije završava. Predložena arhitektura je konceptualno jednostavna, ali nije jasno kakve bi bile njene prednosti nad drugim metodama segmentacije, osim efikasnosti na masovno paralelnim računalima.

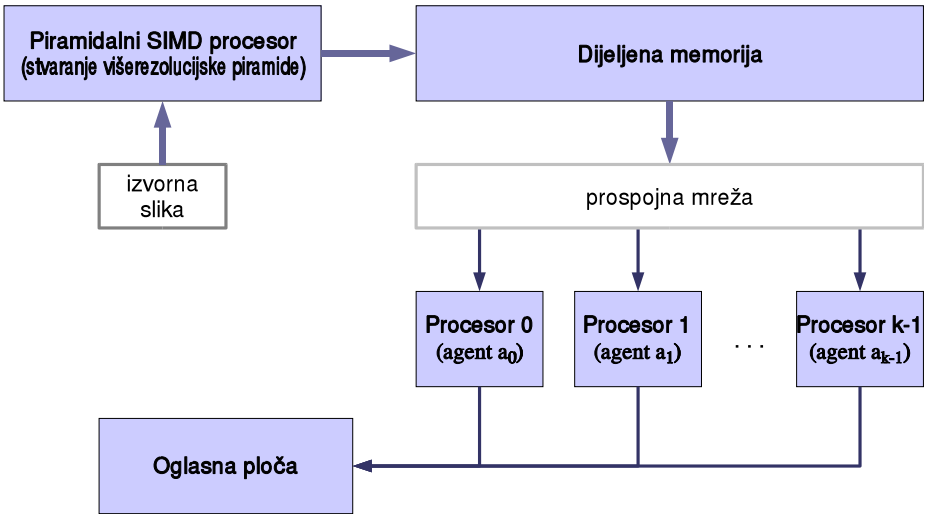
Sličan sustav s agentima čiji okoliš predstavljaju elementi slike (*engl.* image situated agents) su predložili Boucher et al. [boucher98], za segmentaciju mikroskopskih snimaka živih stanica. Sustav je usko prilagođen domeni problema pa tako postoje agenti za pronalaženje jezgre stanice, pozadine, rubova stanice i sl. Kao i kod prethodne metode, agenti su reaktivni i mogu poprimiti nekoliko ponašanja, ovisno o percipiranoj okolini: interakciju (prilikom stapanja dvaju područja), širenje (normalan rast područja) te reprodukciju (stvaranje agenata druge vrste). Autori navode kako je važno svojstvo ovakvih agenata mogućnost modeliranja nezavisnih ponašanja, koje onda koordinira upravljačka komponenta agenta. Glavna zamisao ovakvog sustava je uvesti odgovarajući redoslijed u postupak segmentacije, u skladu s apriornim znanjem o tome što je u razmatranoj klasi slika lakše a što teže segmentirati. Tako se prvo traže slikovni elementi koji pripadaju jezgrama stanica jer iskustvo pokazuje da ih je najlakše naći. Kada dođu do ruba svog područja, agenti za jezgru analiziraju svoje šire susjedstvo, i u odgovarajućim slikovnim elementima stvore agente pozadine i agente stanične membrane. Konačno, nakon što su segmentirane i jezgre i membrane i pozadina, segmentiraju se najteža područja citoplazmi. Autori navode kako bi se već raspoređeni agenti u segmentiranoj slici mogli koristiti i za praćenje kretanja stanica kroz

slijed slika.

Možda najrazrađenija višeagentska metoda segmentacije [duchesnay03] predviđa dvije temeljne vrste agenata, i to za amplitudnu te rubnu segmentaciju. Segmentacija se odvija na različitim razinama nepravilne piramidalne strukture, počevši od najzrnatije razine gdje su agentima pridruženi slikovni elementi. Agenti mogu poprimiti pet različitih strategija ponašanja: rast područja, otkrivanje susjeda, stapanje, suradnja (s agentima svoje ili druge vrste), priprema za sljedeću razinu. Sustav se inicijalizira podsegmentiranom slikom (slikom u kojoj je broj segmentiranih područja znatno veći od od željenog) koja se dobiva nekom od poznatih metoda. Nakon stapanja međusobno vrlo sličnih područja, suradnjom sa susjednim agentima se stvaraju mreže poznanstva regija i rubova koji su međusobno kompatibilni ali ne toliko da bi bili stopljeni u prethodnom koraku. Grupe agenata koji su povezani takvom relacijom sličnosti na sljedećoj grubljoj razini piramide biti će predstavljeni samo s jednim agentom, a procesiranje će se zaustaviti na onoj razini piramide na kojoj broj agenata ne bude znatno manji od prethodne razine. Autori navode da je predložena metoda za njihove slike dala bolje rezultate od ostalih poznatih općenitih metoda, u smislu kvalitete segmentacije i robustnosti na odabir parametara postupka.

Praćenje višestrukih objekata

Tan i Martin [tan89] su predložili paralelni sustav za praćenje više objekata u višerezolucijskoj piramidi. Obradu asinkrono obavlja veći broj procesora, organiziranih u arhitekturu tipa MIMD. Sklopovska arhitektura sustava je prikazana na sl. 2.4. Procesnim elementima se dodjeljuju agenti koji mogu imati ulogu ili nadgledanja dodijeljenog dijela slike ili identifikacije i praćenja objekata koje prijavljuju agenti iz prethodne skupine. Nadgledanje i praćenje se odvija na najvišoj razini piramide, dok se identifikacija odvija na finijim rezolucijama, prema potrebi. Sustav se sastoji od nepromjenljivog broja agenata za nadgledanje, koji po potrebi stvaraju agente za praćenje, čiji vijek traje sve dok ne izgube praćeni objekt. Agenti pratioci dojavljuju rezultate praćenja centraliziranom odlagalištu, u skladu s obrascem oglasne ploče [pfleger98], gdje se i razrješavaju konflikti u vezi sa stapanjem i naknadnim razdvajanjem objekata. Navodi se da je, unatoč mogućem uskom grlu u komunikaciji s oglasnom pločom, takva arhitektura bolja od shema u kojima agenti komuniciraju međusobno svaki sa svakim, kako zbog povećanja mrežnog prometa tako i zbog elegantnijeg rješavanja konflikata i dobivanja koherentnog prikaza scene.



Slika 2.4: Sklopovska arhitektura za praćenje objekata u višerezolucijskoj piramidi.

Remagnino et al. [remagnino98] su opisali konceptualno sličan sustav u kojem su agenti također pridruženi praćenim objektima. Pored agenata za praćenje pojedinačnih objekata, međutim, u ovom radu uvode se i specijalni agenti za analizu interakcije pratećih objekata. Prednost ovakvog pristupa je intuitivno jasan način particioniranja teškog problema interpretacije dinamičkih scena s više pokretnih objekata, ali iz ponuđenog šturog opisa je vrlo teško izvući konkretnije detalje o radu sustava što onemogućava njegovu usporedbu s ostalim rješenjima iz literature. Puno jasnija obrada istog područja primjene može se naći u [hongeng01], ali u tom pristupu se ne koristi višeagentska teorija pa ovdje neće biti dalje razmatran.

2.3 Porazdijeljeno praćenje višestrukim promatračima

U kontekstu ovog rada, najinteresantniji su prethodni radovi na području praćenja objekata većim brojem kamera. Radi se o području koje se može agentski obojiti iznimno jednostavno, na način da se obrada pojedinačnih slijedova slika pridijeli pojedinačnim agentima promatračima. Nadalje, zbog iznimne složenosti postupaka integracije pojedinačnih rezultata te korištenja ukupnog prikaza u povratnoj vezi, često je i te postupke prikladno izraziti u okviru višeagentske teorije. Ipak, eksplicitni višeagentski opis se koristi u samo malom broju radova dok većina koristi terminologiju iz starijeg područja porazdijeljene umjetne inteligencije. Rad jednog te istog sustava često se može opisati i u terminima agenata, i u terminima poslužitelja, klijenata i ostalih vrsta programa s mogućnošću komunikacije, pa nije čudo da se područja višeagentskih sustava i porazdijeljene umjetne inteligencije često poistovjećuju i u teoretskim raspravama [chaib95, huhns99].

Porazdijeljeno praćenje je iznimno široko područje, sa mnogim pažnje vrijednim varijacijama i specifičnim podpodručjima što potvrđuje velik broj znanstvenih radova u posljednjih nekoliko godina. Potencijalna područja primjene se kreću od praćenja ljudi [gavrila99] i pokretnih robota [sogo99] u zatvorenim prostorima, pa sve do nadgledanja prostranih vanjskih područja u heterogenoj senzorskoj mreži [zhao03]. U paragrafu 2.3.1 će biti razmatrani različiti važni aspekti porazdijeljenog praćenja, dok će arhitekture najzanimljivijih ostvarenja biti opširnije prikazane u nastavku odjeljka, uz grupiranje s obzirom na pokretljivost promatrača.

2.3.1 Aspekti porazdijeljenog praćenja

Pri oblikovanju arhitekture za porazdijeljeno praćenje potrebno je obratiti pažnju na niz aspekata tog problema koji značajno uvjetuju njen konačni oblik. Detaljnija rasprava će biti ograničena na aspekte u okviru temeljne konfiguracije s međusobno udaljenim kalibriranim perspektivnim kamerama čiji pogledi se većim dijelom preklapaju, dok će dodatne mogućnosti biti ukratko opisane u posljednjim paragrafima odjeljka. Najvažniji od tih aspekata su opisani u sljedećim paragrafima, dok su konkretne pretpostavke prethodnih radova iz literature sažete u tablici 2.3. Iako je porazdijeljeno praćenje predmet interesa svih članaka iz tablice, njihova uža orijentacija varira od pretežno teorijskih razmatranja bez eksperimentalnog rada, pa sve do isključivog razmatranja više tehničkih detalja bez ikakvih arhitektonskih pretpostavki. Radovi iz tablice su podijeljeni u tri grupe, prema značaju publikacija u kojima su objavljeni. Prva grupa sadrži članke iz referiranih časopisa, druga radove iz manje značajnih časopisa i zbornika prestižnih konferencija, dok su u trećoj grupi radovi s ostalih međunarodnih znanstvenih skupova.

	povezanost promatrača	pokretljivost promatrača	klasa scena	stvarno vrijeme	stupnjevi slobode gibanja objekata	broj promatrača u eksperimentima	velika zalihost promatrača	detekcija objekata
[cai99]	h.s.	pasivni	z.s.	da	2	3	ne	o.p.
[collins01]	h.s.	aktivni	p.s.	da	2	3	ne	o.p.
[dockstader01]	h.s.	pasivni	z.s.	ne	3	4	ne	o.t.
[matsuyama02]	p.m.	aktivni	z.s.	da	3	8	da	o.p.
[mittal03]	–	pasivni	z.s.	ne	2	16	da	m.o.
[zhao03]	p.m.	pasivni	p.s.	da	2	–	da	–
[kanade97]	h.s.	pokretni	p.s.	da	2	–	ne	–
[matsuyama98]	p.m.	pokretni	p.s.	da	3	–	–	–
[nakazawa98]	p.m.	pasivni	z.s.	da	2	3	ne	o.p.
[hoover99]	h.s.	pasivni	z.s.	da	2	4	da	o.p.
[sogo99]	h.s.	pasivni	z.s.	da	2	16	da	m.o.
[matsuyama00]	h.s.	aktivni	z.s.	da	3	4	da	–
[karuppiah01]	h.s.	pasivni	z.s.	da	2	2	ne	o.p.
[khan01]	–	pasivni	z.s.	ne	2	3	ne	o.p.
[tsutsui01]	–	pasivni	z.s.	ne	2	4	ne	o.t.
[yoshida01]	p.m.	pasivni	p.s.	ne	2	–	–	–
[gueziec02]	h.s.	pasivni	z.s.	da	3	2	ne	o.p.
[javed03]	–	pasivni	p.s.	da	2	3	ne	o.p.
[yang03]	h.s.	pasivni	p.s.	da	2	8	da	o.p.
[stillman99]	–	pasivni	z.s.	da	2	2	ne	m.o.
[krumm00]	h.s.	pasivni	z.s.	da	2	2	ne	o.p.
[chang01]	–	pasivni	z.s.	da	2	2	ne	o.p.
[menegatti01]	h.s.	pokretni	z.s.	da	2	–	–	–

p.m. — potpuno povezana mreža, h.s. — hijerarhijska struktura, z.s. — zatvorene scene, p.s. — prostrane scene, o.p. — oduzimanje pozadine, o.t. — optički tok, m.o. — model objekata.

Tablica 2.3: Prethodni radovi u kontekstu važnijih aspekata porazdijeljenog praćenja.

Podjela s obzirom na povezanost promatrača

Iz tablici 2.3 je vidljivo da su decentralizirana povezana mreža i hijerarhijska struktura dva glavna obrasca organiziranja grupe promatrača. Kod decentralizirane mreže [matsuyama02, zhao03], potrebni su sofisticirani komunikacijski protokoli kojima se ostvaruje pronalaženje objekata i ravnomjerno pokrivanje scene skupom promatrača. Glavna prednost ovakvog pristupa je prilagodljivost, a odatle i otpornost na pogreške, jer se u sustav mogu dodavati novi promatrači ili uklanjati stari bez potrebe za ikakvim prethodnim najavama.

Hijerarhijska arhitektura [collins01, yang03] s druge strane nudi optimalniju iskorištenost resursa, dakle jednako efikasan sustav uz manji broj promatrača. Pored toga, pogodnost hijerarhijskog pristupa je mogućnost jednostavne izgradnje ukupnog prikaza scene [dockstader01] jer su svi mjerni podatci automatski dostupni u čvoru na vrhu hijerarhijske strukture. Međutim, iako je ukupni promet u mreži smanjen u odnosu na decentralizirani pristup, nadređeni

čvorovi strukture mogu vrlo lako postati usko grlo jer prema njima teku rezultati obrade svih podređenih čvorova. Takvi problemi mogu se riješiti višerazinskom hijerarijom, čiji nedostatci su povećanje složenosti i specijaliziranosti sustava.

Iz izloženog se može zaključiti kako će decentralizirani pristupi biti najprikladniji u masovnim mrežama sa velikim brojem promatrača i velikom zalihošću pribavljenih informacija. U takvim mrežama sa stotinama promatrača [matsuyama98, zhao03] jednostavnost i prilagodljivost mrežne infrastrukture bez čvrstih hijerarhijskih veza se lako može pokazati odlučujućom. Suprotno, hijerarhijski pristupi će biti pogodniji gdje se želi izvući maksimalan učinak iz postojećeg ograničenog skupa promatrača. Mogući su i hibridni pristupi slični višeagentskom obrascu ugovorne mreže, u kojima se hijerarhijska struktura dinamički prilagođava trenutnom zadatku. Kao i kod višerazinske hijerarhije, i ovdje je problem povećanje složenosti sustava koje se u konkretnim primjenama može pokazati neopravdanim.

Podjela s obzirom na pokretljivost promatrača

S obzirom na pokretljivost, promatrači u sustavima za porazdijeljeno praćenje se mogu podijeliti na pokretne i nepokretne, u ovisnosti o tome jesu li sposobni za samostalnu navigaciju u prostoru. Nepokretni promatrači se još mogu podijeliti na aktivne i pasivne, u ovisnosti o tome mogu li upravljati smjerom gledanja i ostalim parametrima kamere ili ne. Ovdje valja naglasiti da u aktivne promatrače ubrajamo isključivo one koji svoje aktivne kamere koriste u cilju praćenja kretanja objekata, dakle bez pristupa u kojima aktivna kamera služi prvenstveno za dobivanje slike u visokoj rezoluciji objekta praćenog nepokretnim kamerama [stillman99, karuppiah01, collins01].

Većina postojećih radova razmatra isključivo probleme vezane uz nepokretne i pasivne promatrače. Razlozi za to su višestruki, od činjenica da neriješenih problema u klasičnim konfiguracijama ne manjka te da u strukturiranim okolišima aktivni promatrači uglavnom nisu isplativi (većina smjerova gledanja pokriva zidove), pa sve do skupoće pokretnih platformi. Nadalje, od radova koji spominju aktivne i pokretne promatrače, većina problem razmatra isključivo na apstraktnoj razini [kanade97, matsuyama98, menegatti01] ili uz vrlo ograničeno eksperimentiranje [collins01].

Aktivne kamere u sustavu porazdijeljenog praćenja su detaljnije razmatrane samo u radovima Matsuyame et al. [matsuyama00, matsuyama02]. Njihova decentralizirana arhitektura je isprobana na značajnom broju zahtjevnih eksperimenata, a detaljniji opis se nalazi u 2.3.3.

Podjela s obzirom na vrstu razmatranog okoliša

S obzirom na tip okoliša u kojem se objekti gibaju, razlikujemo primjene u prostranim vanjskim scenama [collins01, zhao03], uz prostorno udaljene promatrače povezane uglavnom bežičnim vezama, te primjene u zatvorenim prostorima gdje su promatrači spojeni na istu lokalnu mrežu [dockstader01, matsuyama02]. Pristupi za prostrane scene moraju uzimati u obzir brojne specifičnosti [zhao03] kao npr, veću cijenu komunikacije s udaljenijim promatračima, prisutnost velikog broja heterogenih promatrača, te potrebu za što rjeđom komunikacijom zbog produljenja trajanja baterijskog napajanja. Suprotno, pristupi prilagođeni zatvorenim scenama mogu pretpostaviti relativno maleno i stalno kašnjenje pri mrežnim komunikacijama, te ravninsko gibanje praćenih objekata po horizontalnoj površini nadgledanog prostora. Iako su u kontekstu ovog rada zanimljivi prvenstveno pristupi za rad u zatvorenim scenama, u sljedećim odjeljcima će biti dani detaljniji opisi najistaknutijih pristupa otvorenim scenama [collins01, zhao03].

Pretpostavke za rad u stvarnom vremenu

Većina realističnih primjena porazdijeljenog praćenja zahtijeva rad sustava u stvarnom vremenu[3], pri čemu se javlja niz dodatnih problema. Pribavljanje slika se najčešće obavlja na prostorno udaljenim računalima pa se, zbog nepredvidljivog kašnjenja pri mrežnim komunikacijama i asinkronosti slijedova, pri integraciji ne može pretpostaviti da su pojedinačne značajke izlučene iz *istovremenih* slika različitih slijedova. Dodatno, promatrači mogu imati različite i nestalne performanse pa će, općenito, odgovarajuće učestalosti značajki također biti različite. Integracijska metoda stoga mora sadržavati robustne algoritme prikladne za rad s takvim neusklađenim slijedovima značajki, opisanim točnim vremenskim trenutkom u kojem je pribavljena odgovarajuća slika. Konačno, potrebno je razviti metode za periodičku sinkronizaciju satova svih promatrača da bi vremenski opisnici značajki bili usklađeni.

Drugi problem je jednostavnije prirode i svodi se na potrebnu performansu cijelog sustava kako bi se ukupni rezultati mogli koristiti u povratnoj petlji, za korigiranje odgovornosti aktivnih promatrača ili predikciju budućih položaja pojedinih objekata u svakom od relevantnih slijedova. Subjektivna ocjena je da uz trenutno stanje tehnologije učestalost povratne informacije ne može biti jednaka učestalosti obrade ulaznih slijedova, nego za red veličine manja (npr, jedna poruka u povratnoj vezi na svakih 10 slika ulaznog slijeda). Ova ocjena se slaže s izvještajima o postignutim rezultatima iz literature [dockstader01].

Stupnjevi slobode gibanja objekta

Većina prethodnih autora se ograničava na praćenje objekata s dva stupnja slobode, jer ta pretpostavka vrijedi kod većine realističnih aplikacija. Ljudi i vozila kao najinteresantniji objekti uglavnom ne lete, pa je njihov 3D položaj moguće odrediti kao presjecište odgovarajuće svjetlosne zrake s horizontalnom podlogom. U slučajevima kada podloga nije horizontalna, položaj je moguće odrediti uz apriorno poznat model reljefa okoliša [collins01].

Glavno ograničenje pristupa koji pretpostavljaju mogućnost 3D gibanja objekata je u tome što je za određivanje točnog položaja u svakom trenutku potrebno da objekt prati najmanje dva promatrača [matsuyama02]. Ipak, takav pristup nudi kvalitativno veće mogućnosti kao praćenje ljudskog gibanja na razini dijelova tijela [dockstader01] (ruku, nogu, zglobova) ili praćenje gibanja lopte u televizijskim prijenosima sportskih susreta [gueziec02].

Brojnost i zalihost promatrača

U jednom od najvažnijih radova u području [matsuyama02] navodi se kako su broj promatrača N_p i broj objekata N_o temeljni faktori sustava za porazdijeljeno praćenje. Ipak, većina prethodnih pristupa su skalabilni, u smislu da dozvoljeni broj promatrača nije prethodno ograničen, pa se kao bolji parametar za opis rada sustava može odabrati *zalihost promatrača*. Zalihost promatrača, ζ, može se definirati kao prosječan broj promatrača koji tokom predviđenog rada sustava prati svaki od objekata u sceni, a iznosi od 1 [javed03] do 6 i više [mittal03]. Taj termin je spomenut i ranije u ovom odjeljku, gdje stoji da su decentralizirane arhitekture prikladnije u sustavima s većom zalihošću promatrača. U tom smislu postaje interesantan prag "velike zalihosti" ζ_p koji se proizvoljno ali intuitivno može postaviti u skladu sa sljedećom jednadžbom, uz uvođenje oznake λ za broj stupnjeva slobode praćenih objekata:

$$\zeta_p = \begin{cases} 2, & \lambda = 2 \\ 4, & \lambda = 3 \end{cases} \tag{2.1}$$

[3]jedna od rijetkih izuzetaka jest off–line analiza ljudskog kretanja [dockstader01].

Metoda detekcije objekata

Kod većine pristupa, detekcija objekata se temelji na vrlo općenitoj metodi oduzimanja pozadine. Temeljna zamisao je vrlo jednostavna a svodi se na određivanje slikovnih elemenata objekata oduzimanjem pribavljene slike od modela pozadine koji se može dobiti u fazi inicijalizacije, kad u sceni nema objekata. Brojne razrade temeljne zamisli uključuju statističko modeliranje šuma i promjena osvjetljenja u sceni, metode za izgradnju i održavanje modela scene kod aktivnih promatrača [matsuyama02], te proširenje modela dubinskom informacijom kod promatrača opremljenih stereo kamerama [krumm00]. Glavno ograničenje ove metode je nemogućnost određivanja granice preklapajućih objekata.

Druga često korištena općenita metoda se svodi na segmentaciju slike dobivene određivanjem rijetkog optičkog toka. Na žalost, računska složenost postupka uglavnom ne dopušta njegovu primjenu u stvarnom vremenu.

Konačno, objekti se mogu detektirati i upotrebom jednostavnih apriornih modela (moraju biti jednostavni zbog zahtjeva na rad u stvarnom vremenu), korištenjem značajki kao što su oblik objekta [krumm00], svjetlina ili boja [stillman99]. Valja napomenuti da se modeli objekata gotovo uvijek koriste i nakon prethodnih, općenitijih metoda, za identifikaciju objekata kroz slijed slika i u odgovarajućim slikama različitih slijedova.

Značajke korištene pri integraciji pojedinačnih rezultata

U prvoj fazi porazdijeljenog praćenja, objekti se pronalaze u pojedinačnim slijedovima slika. U drugoj fazi, pojedinačni rezultati se integriraju, ostvarivanjem korespondencije različitih značajki objekta. Najčešće korištena značajka je svakako položaj objekta (ako su dva promatrača registrirala objekt na istom mjestu, onda se najvjerojatnije radi o istom objektu). Pored toga koristi se brzina objekta [tsutsui01], te različita statistička svojstva slikovnih elemenata koji pripadaju objektu (npr, histogram boje) [collins01].

Zrnatost integracije

Prethodno je spomenuto da pristupi korespondenciji koji se temelje na malim područjima nisu prikladni kod konfiguracija sa znatno udaljenim točkama gledanja, pa je integraciju najjednostavnije ostvarivati na razini cjelokupnih objekata. Ipak, u nekim primjenama je najzanimljivije gibanje strukture praćenih objekata [dockstader01], pa je tada korespondenciju potrebno obavljati na razini dijelova objekta. Još ambicioznji je pristup koji se temelji na određivanju volumnih elemenata prostora koji su konzistentni sa slikama iz svih točaka gledanja [mittal03], a temelji se na postavljanju epipolarnog ograničenja kao što je navedeno u dodatku D, i to za sve parove promatrača, i sve odsječke odgovarajućih epipolarnih pravaca koji pripadaju korespondirajućim područjima.

Ti pristupi su ambiciozni jer se susreću s problemima međusobnog prekrivanja strukture objekata koji općenito ne moraju imati jednoznačno rješenje. Upravo zbog toga predložena rješenja [dockstader01] i [mittal03] zahtijevaju velik broj promatrača (4 odnosno 16) i sinkronizirane ulazne slijedove slika, a zbog složenosti postupaka ne rade u stvarnom vremenu.

Na ovom mjestu valja još spomenuti i mogućnost integracije na razini slikovnih elemenata ulaznih slika [hoover99]. U predloženom elegantnom pristupu, svaki promatrač otkriva siluete objekata na ravnoj podlozi zatvorene prostorije metodom oduzimanja pozadine. Promatrači su kalibrirani pa svaki od njih homografijom može preslikati detektirane siluete na tlocrt prostorije i tako dobiti svojevrsnu kartu zauzeća tlocrta. Pojedinačne karte nisu precizne jer promatrači ne mogu vidjeti površinu podloge *iza* detektiranih objekata pa su ta područja

označena kao zauzeta iako možda i nisu. Točna karta zauzeća se dobiva u nadglednoj komponenti, obavljanjem logičke operacije "I" nad odgovarajućim elementima pojedinačnih karata (element tlocrta je zauzet ako i samo ako je zauzet u svim pojedinačnim kartama). Objekti se iz ukupne karte zauzeća izlučuju jednostavnim pronalaženjem povezanih područja.

Korespondencija s nepreklapajućim pogledima

Ponekad, a posebno kod prostranih scena, nije isplativo pokriti senzorima cijelo područje u kojem se gibaju objekti. Tada se problem integracije svodi na ostvarivanje korespondencije između odsječaka trajektorija dojavljenih od strane različitih promatrača koji se u vremenu ne preklapaju. Kako nema načina da se kao princip korespondencije iskoristi položaj objekta (jer objekt u svakom trenutku vidi samo jedan promatrač), korespondenciju je potrebno ostvariti sofisticiranim vjerojatnosnim proračunima, koji se temelje na glatkom nastavljanju trajektorije i karakterističnim značajkama detektiranog objekta [collins01, javed03].

Upotreba aktivnih kamera za identifikaciju

U cilju identifikacije, često se u kombinaciji s nekom metodom praćenja javlja potreba za pribavljanjem slike koja prikazuje praćeni objekt (ili neki njegov dio) u visokoj rezoluciji [stillman99, karuppiah01]. Mogući kandidati za takav postupak su lice osobe ili registarska pločica vozila. Pored potpuno automatiziranih sustava, takav pristup je zanimljiv i kod manualnih metoda praćenja, kada operater zahtijeva uvećani prikaz dijela slike na posebnom ekranu [collins01].

Kalibracija promatrača

Kako bi se mogla baviti integracija, rezultate promatrača je potrebno moći interpretirati u zajedničkom referentnom koordinatnom sustavu svijeta, tj. potrebno je umjeriti unutrašnje i vanjske parametre odgovarajućih kamera. Dva su temeljna pristupa tom problemu: eksplicitno umjeravanje svakog od promatrača te razne metode samoumjeravanja. Eksplicitno umjeravanje će biti razmatrano u poglavlju 4, a još više podataka može se naći u literaturi [collins99, olsen01]. Samoumjeravanje (autokalibracija) kamere se svodi na automatizirani proces kojim se pri inicijalizaciji sustava omogućuje promatračima da sami odrede parametre svojih kamera. Najčešće se praćeni objekt provede pod nadzorom operatera navigacijskim prostorom, a promatrači onda usklađuju parametre koristeći epipolarna ograničenja [chang01], ograničenja na ravninsko [sogo99, chang01] ili poznato [menegatti03] gibanje objekata ili referentne objekte u okolišu [chang01]. Pored toga, moguć je i automatski pristup [stein00, khan01] kod kojeg se kalibracija temelji samo na ograničenjima gibanja u Euklidskom prostoru.

Upotreba panoramskih i stereo kamera

Panoramske kamere se ponekad koriste kao protuteža aktivnim kamerama, na način da se jednaki prostor može pokriti bez sofisticiranih upravljačkih algoritama, ali uz znatno slabiju rezoluciju dobivenih slika [karuppiah01, menegatti03]. Prethodni radovi uglavnom nisu koristili stereo kamere unatoč postojećim komercijalnim platformama i potencijalno velikim mogućnostima uslijed mogućnosti uparivanja volumnih slika [krumm00]. Razlog je vjerojatno u tome što su do sada postojeća komercijalna rješenja imala problema s postizanjem upotrebljivih rezultata u stvarnom vremenu (navodi se [krumm00] performansa od 4Hz za

slike dimenzija 320×240). Zbog stalnog eksponencijalnog rasta performanse računala, ti odnosi bi se uskoro mogli znatno promijeniti.

Sinkronizacija kamera

Signali dobiveni analognim kamerama mogu se sinkronizirati pa odgovarajuće slike pojedinih slijedova odgovaraju jedinstvenim trenutcima u vremenu. Time se izbjegava potreba za složenim shemama interpolacije odnosno predikcije istovremenog položaja slike objekta u međusobno asinkronim slijedovima slika. Ipak, ta se tehnika rjeđe koristi jer njeno korištenje prate sljedeća ograničenja:

- mreža promatrača mora biti vrlo homogena:
  - promatrači moraju imati približno jednake performanse (broj obrađenih slika u sekundi, Hz), jer sustav prati rad najsporijeg promatrača,
  - vrijeme propagacije signala između svaka dva čvora mreže moraju biti približno jednaka jer integracija može početi tek kad mjerenja svih promatrača budu dostupna (postojeća rješenja zato koriste hijerarhijske arhitekture promatrača te ne rade u stvarnom vremenu);
- sve kamere moraju biti spojene na isti sinkronizacijski signal.

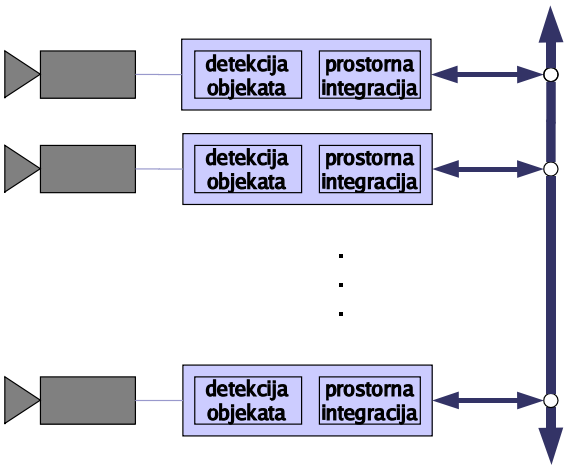
Navedena ograničenja poništavaju prednosti kod većine realističnih strukturiranih ili prostranih okoliša, pa se tehnika koristi uglavnom kod vrlo simetričnih laboratorijskih sustava koji zahtijevaju veliku preciznost ulaznih rezultata [dockstader01, mittal03].

2.3.2 Porazdijeljeno praćenje pasivnim promatračima

Potpuno porazdijeljeni pristup praćenju većim brojem promatrača je opisan u radu Nakazawe et al. [nakazawa98]. Pretpostavlja se sustav od više promatrača, svaki od kojih upravlja procesnim i senzorskim resursima (kamerom i dodijeljenim računalom). Promatrači su spojeni u potpuno povezanu mrežu: svaki promatrač šalje položaje detektiranih objekata svim ostalim promatračima u grupi kao što je prikazano na sl. 2.5. Ukupni prikaz scene se tako stvara i održava lokalno u sklopu svakog promatrača. Promatrači imaju kalibrirane kamere te apriorno znanje o strukturi scene (tlocrt okoliša), na temelju kojih u inicijalizacijskoj fazi određuju presjeke pogleda i strategiju inicijalizacije praćenja:

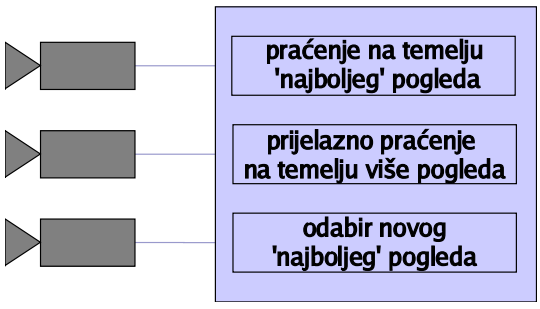
- ulazak objekata u vidno polje se testira samo u dijelovima slike koji odgovaraju rubovima vidnog polja, i to samo onim rubovima koji nisu u vidnom polju nekog drugog promatrača;
- ostale rubove vidnog polja nije potrebno provjeravati jer će eventualni dolazak objekata iz tog smjera dojaviti ostali promatrači.

Cai i Aggarwal [cai99] su predstavili monolitni pristup porazdijeljenom nadgledanju strukturiranog zatvorenog prostora. Predloženi sustav se temelji na korespondenciji značajki oblika detektiranog ljudskog tijela, te položaja i intenziteta karakterističnih točaka na ljudskom tijelu. Arhitektura sustava ne predviđa nikakvu autonomiju promatrača, slike pribavljene kalibriranim kamerama su digitalizirane istim računalom na kojem se obavlja i integracija rezultata. Organizacija postupka integracije slična je horizontalnoj agentskoj arhitekturi, sa tri osnovne strategije ponašanja (načina rada) pri praćenju svakog objekta (sl. 2.6):



Slika 2.5: Porazdijeljeno praćenje uz lokalne ukupne poglede [nakazawa98].

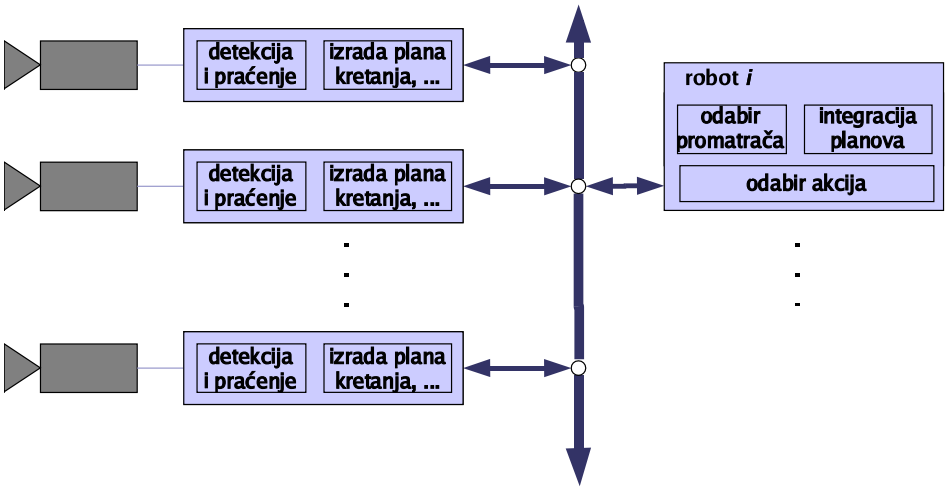
1. praćenje na temelju "najboljeg" pogleda:
   - koristi se kada je objekt daleko od rubova trenutnog "najboljeg" pogleda,
   - klasično 2D praćenje;
2. prijelazno praćenje na temelju više pogleda:
   - koristi se kada trenutni "najbolji" pogled nije dovoljan za praćenje objekta,
   - npr, kad se objekt približi rubu pogleda ili otiđe predaleko od kamere,
   - značajke se uparuju koristeći epipolarno ograničenje;
3. automatska promjena "najboljeg" pogleda:
   - nadgledna procedura koja testira da li neki od pogleda zadovoljava uvjete da postane novi "najbolji" pogled,
   - uvjeti: u toku je robustno prijelazno praćenje, objekt je daleko od ruba pogleda.



Slika 2.6: Monolitna arhitektura za porazdijeljeno praćenje [cai99].

Sogo et al. [sogo99] su predstavili porazdijeljeni sustav za navigaciju robota s predefiniranim putovima kretanja. U fazi inicijalizacije, roboti se gibaju kroz okoliš pod upravljanjem ljudskog operatera. Promatrači u toj fazi uče (i) pozadinsku sliku scene, (ii) dijelove scene po kojima se roboti mogu gibati, (iii) nagib pridružene kamere u odnosu na ravninu gibanja, i (iv), željene trajektorije gibanja robota. Tokom rada, svaki promatrač određuje plan gibanja detektiranih robota, kao kutnu korekciju θ trenutnog smjera gibanja u k.s. svijeta.

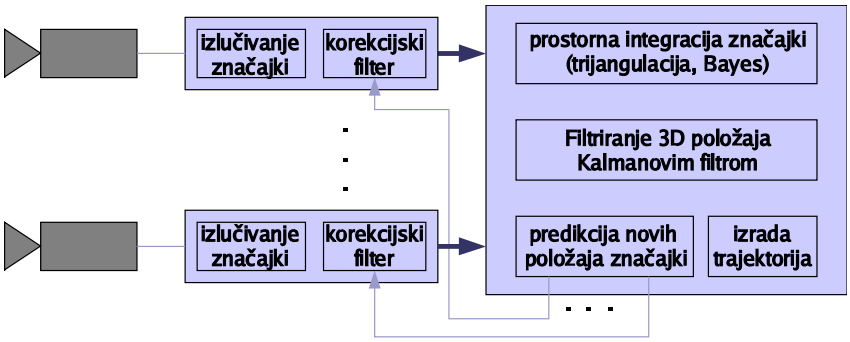
Korekcija trajektorije se obavlja u skladu s trenutnim položajem objekta te naučenim slobodnim dijelovima scene i željenim trajektorijama, a uzima u obzir i autokalibrirani nagib kamere te eventualno izbjegavanje drugih robota. Samostalni roboti nakon slanja zahtjeva za instrukcije o daljnjem kretanju, od porazdijeljenog sustava dobivaju listu planova svih promatrača koji taj robot vide (identifikacija robota se obavlja na temelju njegove boje). Roboti određuju korekciju trajektorije na temelju planova vrednovanih u skladu s veličinom robota u slici odgovarajućeg promatrača. Predložena arhitektura sustava je prikazana na sl. 2.7, a može se klasificirati kao centralizirano višeagentsko planiranje pri čemu promatrači odgovaraju agentima, a roboti arbitrima za rješavanje konflikata vezanih uz pojedine planove.



Slika 2.7: Porazdijeljeno praćenje u društvu nezavisnih promatrača [sogo99].

Dockstader i Tekalp [dockstader01] su opisali hijerarhijski sustav za porazdijeljeno praćenje gibanja više ljudi u zatvorenoj prostoriji. Praćenje se izvodi na razini pojedinih dijelova ljudskog tijela (praćenje tzv. artikuliranih objekata), uz razmatranje međusobnog prekrivanja različitih dijelova istog objekta. Gibanje se prati na temelju rijetkog optičkog toka, a vremenska i prostorna integracija se obavljaju uz pomoć Kalmanovog filtra odnosno Bayesove mreže. Izgrađeni ukupni prikaz scene se koristi u povratnoj vezi, za predikciju budućih položaja objekata u pojedinim slijedovima slika. Takav pristup nije problematičan jer se analiza obavlja na prethodno snimljenim nizovima slika pribavljenim sinkroniziranim kamerama, dakle zanemaruju se utjecaji kašnjenja pri pribavljanju slike i razmjeni poruka. Prema arhitekturi ilustriranoj na sl. 2.8, opisani pristup se može svrstati u višeagentske sustave sa fiksnom organizacijskom strukturom. U pretpostavljenom slučaju, s nepokretnim promatračima, naprednije sheme kao što je npr. ugovorna mreža ne bi donijele nikakve prednosti jer bi i u slučaju potrebe za dodatnim promatračima i složenijom hijerarhijom, povezanost bila uvjetovana njihovim prostornim rasporedom.

Monolitna arhitektura vrlo slična onoj na sl. 2.6 [cai99] je predložena u radu Yanga et al. [yang03], koji opisuje specifični problem brojanja velikog broja ljudi u malenom prostoru. Pojedini promatrači izlučuju objekte oduzimanjem pozadine i šalju komprimirane binarne slike programu koji obavlja integraciju rezultata, a izvodi se na zasebnom računalu. Integracijski program projicira položaje detektiranih nakupina na ravninu kretanja i pronalazi objekte presijecanjem projekcija dobivenih za slike iz svih slijedova. Pri oduzimanju pozadine koristi se vrlo niski prag koji propušta velik broj slikovnih elemenata koji pripadaju



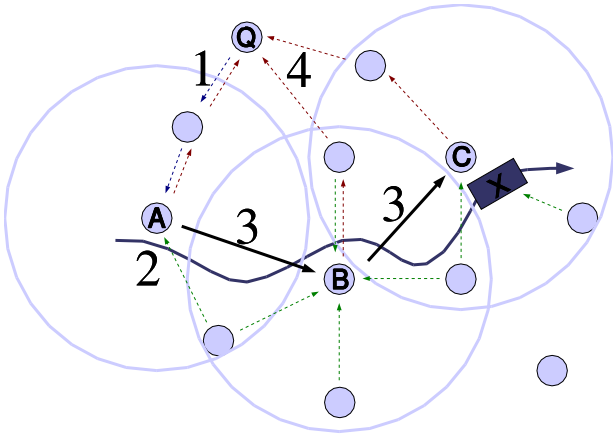
Slika 2.8: Hijerarhijska arhitektura za porazdijeljeno praćenje [dockstader01].

pozadini a zbog šuma se razlikuju od modela. Navodi se kako se upotrebom većeg broja promatrača (npr. 8) dobivene proširene siluete objekata istanje automatski, pri pronalaženju njihovog presjeka u postupku integracije. Kao logično proširenje arhitekture, predlaže se particioniranje svih promatrača u skupove čiji pogledi se međusobno ne preklapaju, te organiziranje sustava u više razina hijerarhije.

Zhao et al. [zhao03] razmatraju mogućnost oblikovanja masovne senzorske mreže za porazdijeljeno praćenje sa velikom zalihošću promatrača. Razmatrana mreža bi se sastojala od velikog broja heterogenih promatrača ili senzorskih čvorova (IR, vid, ultrazvuk), sa ograničenim resursima (baterijsko napajanje te propusnost i domet bežične veze). U takvoj mreži, javljaju se specifični zahtjevi jer se puno više energije troši za bežičnu komunikaciju nego za obradu podataka. Stoga je vrlo važno osigurati što lokalnije sheme komunikacije koja se ne može izbjeći zbog potrebe za prostornom integracijom mjerenja pojedinačnih promatrača. U radu se razmatraju tri temeljne arhitekture:

1. Arhitektura s čvrstom hijerarhijom:
   - nedostatak: ne može se izbjeći skupa komunikacija s udaljenim nadređenim čvorovima,
   - prednost: konceptualna jednostavnost;
2. Dinamička prilagodba hijerarhije stanju u sceni:
   - nadređeni čvor koji obavlja integraciju trajektorije objekta nije fiksan: uloga nadređenog čvora putuje mrežom zajedno s praćenim objektom,
   - takav postupak integracije se može izraziti i posebnim pokretnim agentom dodijeljenom praćenom objektu koji po potrebi putuje između nepokretnih agenata domaćina u senzorskoj mreži [qi03],
   - nedostatak: za veći broj objekata, usko grlo postaje integracija podataka o praćenju pojedinačnih objekata u ukupni prikaz scene,
   - prednost: arhitektura je posebno prikladna za praćenje objekata;
3. Ravnopravna potpuno povezana mreža:
   - metafora kolektivne svijesti: integracijski podatci se čuvaju u svakom čvoru mreže;
   - nedostatak: visoka cijena komunikacije u sustavu,
   - prednost: dobro za neke posebne zadatke, npr, određivanje broja objekata u okolišu;

U radu se najviše pažnje posvećuje arhitekturi sa prilagodljivom hijerarhijskom strukturom. Njen rad se može opisati na primjeru koji je ilustriran na sl. 2.9. Obojeni kružići u slici predstavljaju čvorove mreže (promatrače), veće kružnice označavaju domet komunikacije pojedinih čvorova, pune crte među čvorovima pokazuju preuzimanje odgovornosti za postupak integracije, dok isprekidane crte označavaju izmjenu poruka među promatračima.



Slika 2.9: Dinamička prilagodba arhitekture stanju scene [zhao03] (detaljnije u tekstu).

Tijek događaja u primjeru je sljedeći:

1. u mrežu se preko čvora **Q** upućuje zahtjev za praćenje objekta **X**;
2. promatrač koji prvi otkrije objekt (**A**) preuzima zadatak integracije, ostali senzori iz susjedstva mu dojavljuju tekuće položaje praćenog objekta;
3. kad se objekt udalji od promatrača koji trenutno obavlja integraciju, odgovornost i dotadašnje podatke o praćenju preuzima promatrač koji je u tom trenutku najbliži objektu (**A**→**B**, **B**→**C**);
4. opažena trajektorija objekta tako putuje mrežom i povremeno biva poslana čvoru koji je inicirao praćenje.

2.3.3 Porazdijeljeno praćenje aktivnim promatračima

Collins et al. [collins01] su predstavili rezultate višegodišnjeg rada na projektu za kooperativno nadgledanje prostranih scena. Iako se projekt uglavnom razmatrao u kontekstu vojnih primjena, navodi se da projekt ima vrlo široku primjenu u civilnom životu, uglavnom za automatsko osiguravanje trgovina, parkirališta i ostalih objekata od krađe ili drugih protuzakonitih radnji. Osnovna struktura sustava je jednorazinska fiksna hijerarhija, u kojoj se mjerenja svih promatrača integriraju u jedinstvenom nadređenom čvoru, gdje se i usklađuju njihove odgovornosti. U opisanoj fazi razvoja, sustav je sposoban za poluautomatski rad, na način da može obavljati razne zadatke zadane od strane operatera. Primjer podržanih zadataka su: "prati ovaj automobil" ili "izvijesti o prolasku crvenih kamiona kroz ova vrata". Korisničko sučelje objedinjuje rezultate praćenja i mogućnost unosa novih zadataka u sustav. Predviđena su dva glavna načina vizualizacije rezultata praćenja, i to izravnim superponiranjem rezultata na slike pribavljene od pojedinih promatrača, ili prikazom objekata na zemljopisnoj karti nadgledanog područja. Tako je moguće na karti označiti zanimljivo

područje, i sustavu naložiti da prati sve objekte koji u to područje uđu. Glavna razlika između opisanih zadataka i zadataka koji se razmatraju u ovom radu je utoliko što su ovdje zanimljivije mogućnosti za automatski rad sustava, bez intervencija operatera. Tipični primjer takvog zadatka je: "prati kretanje svih objekata u zadanom prostoru, uz što bolje iskorištavanje postojećih resursa (promatrača)".

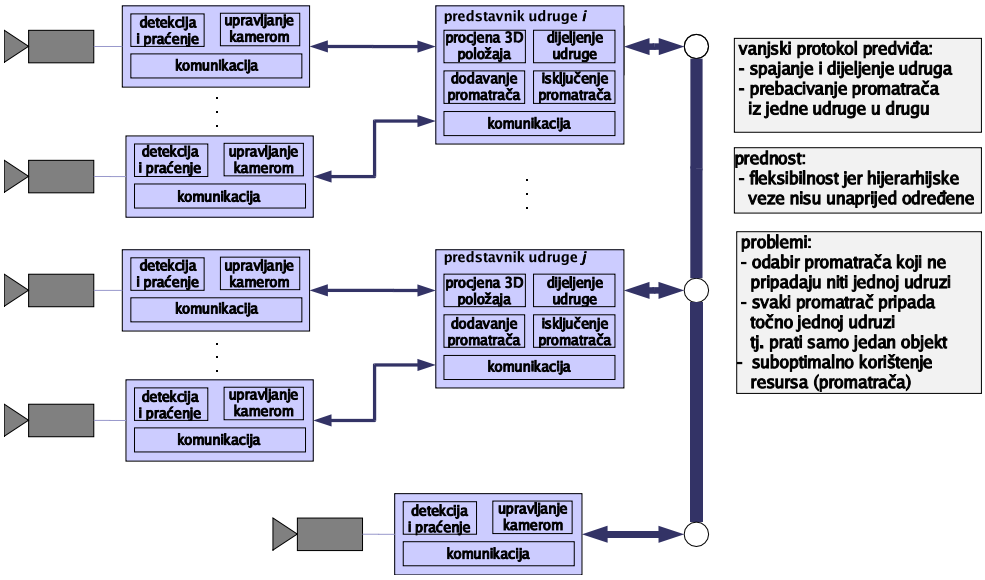
Jedan od najsofisticiranijih prethodnih višeagentskih sustava računarskog vida razmatra upravo problem praćenja objekata višestrukim aktivnim promatračima. Pristup Matsuyame i Ukite [matsuyama02] pretpostavlja zatvoreni prostor te veliku zalihost promatrača. Svaki od promatrača se prema njihovom modelu sastoji od modula za opažanje (analizu slike), djelovanje (upravljanje kamerom), komunikaciju sa ostalim promatračima te modula za sinkroniziranu izmjenu podataka (tzv. dinamička memorija). Svi moduli se izvode usporedno, u zasebnim tokovima izvođenja ili dretvama (*engl.* thread), a međusobno su povezani preko modula za izmjenu podataka čiji zadatci su interpolacija i predikcija vrijednosti varijable u zadanom trenutku, na temelju niza diskretnih mjernih rezultata. Promatrači mogu raditi u jednom od dva temeljna načina rada:

- *praćenje*: smjer gledanja kamere prati točno jedan objekt, kad objekt nestane prelazi se u *traženje*;
- *traženje*: smjer gledanja kamere sistematski se pomiče sve dok se ne nađe neki objekt, tada se prelazi u *praćenje*.

Na početku rada sustava, svi promatrači rade kao slobodni strijelci u stanju "traženje". Kada neki od promatrača nađe objekt, razglasi to svim ostalim promatračima, te s onima koji su taj isti objekt uspjeli naći formira "udrugu" za njegovo praćenje.[4] Zbog dostupnosti većeg broja pogleda, udruga može odrediti 3D položaj praćenog objekta. Kada neki od promatrača iz udruge izgubi kontakt sa objektom, ponovo postaje slobodni strijelac. Pored opisanog protokola za stvaranje udruga, predviđeni su i protokoli za njihovo restrukturiranje, spajanje i dijeljenje. Pregovore u ime udruge s ostalim udrugama obavlja njen predstavnik koji je početno osnivatelj udruge. Restrukturiranje se obavlja kada dvije udruge ustanove da imaju različit broj članova, pa se neki članovi brojnije udruge prepuštaju drugoj udruzi. Dvije udruge mogu ustanoviti da prate isti objekt i tada se primjenjuje protokol za njihovo spajanje u zajedničku udrugu. Ovakva situacija se javlja kada se dva objekta međusobno toliko približe da ih više nije moguće razlikovati. Konačno, dijeljenje udruge se obavlja kada se pojedinačna mjerenja više ne mogu pripisati jednom objektu. To se može dogoditi kada se dva prethodno stopljena objekta razdvoje i počnu međusobno udaljavati. Opisani rad predstavlja vrlo vrijedan doprinos razumijevanju područja porazdijeljenog praćenja, kako na arhitektonskoj razini tako i na razinama najsitnijih implementacijskih detalja. Arhitektura predloženog pristupa se može opisati kao višeagentski sustav na dvije razine (vidi sl. 2.10). Koordinacija na razini promatrača može se svrstati u agentske arhitekture temeljene na ugovornoj mreži (vidi sl. 2.2), jer pri stvaranju udruge slobodni strijelac koji je prvi pronašao objekt natječajem traži promatrače koji mu pri tome mogu pomoći. Koordinacija na razini udruga je manje razrađena, jer je izvedena tako da svaka udruga komunicira sa svakom

[4] U ovom trenutku problem predstavlja situacija u kojoj bi objekt uspjeli identificirati *svi* ostali promatrači iz skupa. Sve ostale promatrače nije povoljno primiti u udrugu jer bi cijela udruga pokrivala samo jedan potencijalno vrlo mali dio scene. Problem se rješava zahtjevom da u svakom trenutku bude apriorno odabran broj n_{SS} slobodnih strijelaca (u eksperimentima je korišteno n_{SS}=1), koji se onda ne bi primili u udrugu. Iako autori ne navode način izvedbe, takav zahtjev bi se mogao izvesti ili decentralizirano, kolektivnom sviješću o broju slobodnih strijelaca ili centraliziranom oglasnom pločom gdje bi svaki promatrač registrirao svoj način rada. U oba slučaja, rješenje negativno utječe na prilagodljivost sustava.

drugom što bi moglo postati usko grlo sustava u slučaju većeg broja praćenih objekata (eksperimenti su provedeni u konfiguraciji s deset promatrača i dva objekta).



Slika 2.10: Porazdijeljeno praćenje u društvu surađujućih agenata [matsuyama02].

2.3.4 Porazdijeljeno praćenje uz pokretne promatrače

Posebno zanimljivo područje porazdijeljenog praćenja sačinjavaju pristupi u kojima se pretpostavlja da su promatrači pokretni ili, specifičnije, imaju mogućnost samostalne navigacije. Takva mogućnost prvi put načelno je spomenuta u radu Kanadea et al. [kanade97] koji opisuje raniju fazu projekta koji je detaljnije opisan u [collins01]. U spomenutim radovima, navodi se česta potreba za integracijom rezultata praćenja nepokretnim senzorima, sa rezultatima dobivenim iz zraka (helikopter ili zrakoplov). Pri tome je ključni problem određivanje točnog položaja pokretnog senzora koji se uglavnom rješava za svaku sliku posebno, na temelju poklapanja referentnih objekata na pribavljenim slikama sa zemljopisnom kartom ili s podatcima dobivenim očitanjem nepokretnih senzora. Autori navode uspješnu integraciju podataka jednog pokretnog senzora iz zraka s mrežom heterogenih nepokretnih senzora na zemlji [collins01].

U opisu ranije faze prethodno predstavljenog sustava za porazdijeljeno praćenje aktivnim promatračima [matsuyama02], Matsuyama predlaže formalizaciju utjelovljenih i beztjelesnih agenata promatrača [matsuyama98]. Ta formalizacija je ovdje izložena u prerađenom obliku, u skladu s terminologijom predloženom u tablici 2.1. Pored usvojenih simbola S (skup stanja okoline), P (skup opažaja okoline), I (skup unutrašnjih stanja agenta) te A (skup akcija agenta), uvodimo sljedeće nove oznake:

P_I: pribavljene slike;

P_C: poruke primljene od drugih agenata;

I_S: programski dio unutrašnjeg stanja agenta;

I_V: parametri aktivne kamere pridružene agentu;

I_W: položaj pokretnog agenta;

A_C: slanje poruka drugim agentima;

Vrijedi, naravno: $P = P_I \times P_C$, $I = I_S \times I_V \times I_W$, i $A = A_C$.

Beztjelesni agenti se po definiciji ne mogu kretati, ali se dopušta da imaju upravljivu kameru; u tom slučaju vrijedi: $I_V \neq \varnothing$, $I_W = \varnothing$. Funkcionalnost pokretnog agenta promatrača u terminima elementarnih apstraktnih operacija sa sl. 2.1 može se zapisati kao što je navedeno u tablici 2.4.

operacija	definicija
gledaj	$S \times I_V \times I_W \rightarrow P$
razmisli	$I_S \times P \rightarrow I_S$
djeluj	$I_S \rightarrow A_C \times I_V \times I_W$

Tablica 2.4: Operacije pokretnog agenta promatrača.

Iz tablice se vidi da se opažanje promatrača (operacija "gledaj") sastoji od pribavljanja slika i poruka drugih promatrača, a ovisi o njegovom položaju i smjeru gledanja, te stanju okoline. Nadalje, programsko stanje promatrača se obnavlja u ovisnosti o opaženoj okolini (operacija "razmisli"). Konačno, promatrač djeluje u skladu sa svojim programskim stanjem, promjenom parametara kamere i svog položaja, te slanjem poruka drugim promatračima.

U formalizaciji opisanoj u tablici 2.4, pretpostavljeno je da su parametri kamere i položaj agenta u svakom trenutku dostupni i poznati. Međutim, zbog tromosti mehaničkih dijelova, sporosti komunikacije i integracijskih grešaka pri odometrijskoj lokalizaciji [segvic00], to u praksi najčešće nije slučaj. Zato je postavljanje i očitavanje tih parametara često povezano sa stohastičkim rezoniranjem koje se prikladnije uklapa u operacije gledaj i djeluj nego u operaciju razmisli. Uvodimo oznake:

P_V: očitani parametri aktivne kamere;

P_W: očitan položaj;

A_V: podešavanje parametara aktivne kamere;

A_W: promjena položaja.

Pokazuje se da "nova" formalizacija u potpunosti odgovara apstraktnom opisu danom u tablici 2.1, uz sljedeću razradu skupova P i A:

$$P = P_I \times P_C \times P_V \times P_W;$$

$$A = A_C \times A_V \times A_W.$$

U skladu s ovom formalizacijom, agent promatrač nije ništa drugo nego računalni program čije opažanje okoline se svodi na pribavljanje slika, primanje poruka, te očitanje pozicijskih i ostalih senzora pridruženog sklopovlja, dok mu mogućnosti djelovanja uključuju slanje poruka i upravljanje pridruženim sklopovljem (pokretna kamera, motori pogonskih kotača).

Jedan od rijetkih radova koji detaljnije razmatra probleme pokretnih promatrača u porazdijeljenom sustavu za praćenje, opisuje sustav za praćenje položaja lopte u okviru "momčadi" robotskog nogometa [menegatti01]. Svaki od robota promatrača ima dvije kamere, panoramsku i perspektivnu: prednost panoramske kamere je u širem vidnom polju, dok perspektivna nudi veću rezoluciju pribavljenih slika. Roboti periodički obavljaju apsolutnu lokalizaciju metodom umjetnih referentnih objekata, a u međuvremenu korigiraju položaj očitavanjem

odometrijskih senzora. Praćenje lopte se odvija po shemi sličnoj onoj koja je opisana u [matsuyama02], dinamičkim stvaranjem i održavanjem udruge promatrača koji loptu vide. Zbog kretanja promatrača i netočne lokalizacije, lopta se ne može tražiti oduzimanjem pozadine nego nekom od metoda koje koriste apriorno znanje o lopti. Pri integraciji pojedinačnih mjerenja, nesigurnost se modelira kao zbroj lokalizacijske greške i greške položaja lopte u koordinatnom sustavu robota. Lokalizacijska pogreška ovisi o sljedećim činiocima:

- vrsti kamere kojom je lokalizacija obavljena (panoramska ili perspektivna);
- apriornoj ocjeni pogreške apsolutne lokalizacije;
- vrijeme proteklo od posljednje apsolutne lokalizacije.

U postupku određivanja položaja lopte, greška ovisi o:

- vrsti kamere kojom je pribavljena slika na temelju koje je pronađena lopta;
- procijenjenoj udaljenosti od lopte.

2.3.5 Osnovni tipovi arhitektura za porazdijeljeno praćenje

Na temelju prethodnog pregleda relevantnih radova na području porazdijeljenog praćenja računarskim vidom, moguće je odrediti osnovne tipove arhitektura te ocijeniti njihove prednosti i nedostatke s obzirom na pretpostavke vezane uz konkretno područje primjene. Podjela će se temeljiti na aspektima porazdijeljenog praćenja koji su identificirani i razrađeni u paragrafu 2.3.1. Od svih navedenih aspekata, prema subjektivnoj ocjeni, najvažniji s arhitektonskog stajališta su klasa scena, pokretljivost promatrača, broj promatrača N_p, broj praćenih objekata N_o te zalihost promatrača ζ. Klasa razmatranih scena je bitan parametar jer izravno utječe na tehnologiju povezivanja promatrača koja onda definira latenciju i propusnost komunikacijskih veza. Kod većine primjena, latencija pri komunikaciji ne dozvoljava udaljeno upravljanje smjerom gledanja ili položajem promatrača jer se inače gubi reaktivnost kao važna prednost koncepta pokretnih promatrača. Zato aktivni i, posebno, pokretni promatrači unose zahtjeve za većom autonomijom promatrača te razmatranjem njihovog rada u kontekstu stvarnog vremena. Parametri N_p, N_o i ζ općenito nisu međusobno ovisni jer su, primjerice, pod pretpostavkom $N_p \approx N_o$, moguće kombinacije s različitim vrijednostima ζ, u ovisnosti o tome jesu li objekti zbijeni ili rasuti. Ocjene prilagodljivosti temeljnih oblika arhitekture za porazdijeljeno praćenje s obzirom na pojedine vrste scena te kretanja parametara N_p, N_o i ζ, sažete su u tablici 2.5. Tablica zorno prikazuje kako je područje primjene monolitnih sustava dosta ograničeno kako zbog slabe skalabilnosti u smislu rasta parametara N_p, N_o i ζ, tako i zbog problema s komunikacijskom latencijom pri upravljanju pokretnih promatrača.

Unatoč vrlo dobroj prilagodljivosti, pristup koji se temelji na planovima nezavisnih eksperata [sogo99] također ima ograničeno područje primjene. U takvom pristupu, ostvarivanje suradnje promatrača je vrlo neizvjesno jer ona ovisi o sofisticiranim metodama za usaglašavanje planova pojedinačnih promatrača. U opisanoj izvedbi [sogo99] koristi se vrlo jednostavna metoda usaglašavanja, pa je pristup pogodno primijeniti samo kod modeliranja jednostavnih reaktivnih ponašanja, gdje je lokalna informacija uglavnom dovoljna za ostvarivanje funkcije sustava (npr, jednostavnija shema navigacije robota). Konačno, opisani sustav ne održava ukupni prikaz scene, ali to ne mora biti problem jer se on može dobiti kombiniranjem s jednom od arhitektura koja to podržava.

arhitektura	p.s.†	p.p.†	u.p.†	$N_p^{\ddagger}\uparrow$	$N_o^{\ddagger}\uparrow$	$\zeta^{\ddagger}\uparrow$	$\zeta\downarrow$	prednosti i nedostatci
monolitni sustav [cai99], sl. 2.6	−	−	+	−	−	−	+	slaba prilagodljivost zbog duljine signalnih linija i ograničene procesne moći
fiksna hijerarhija [dockstader01, collins01], sl. 2.8	−	+/−	+	+/−	+	+	+	jednostavni protokol i dobro iskorištenje resursa uz ručnu konfiguraciju
prilagodljiva hijerarhija [matsuyama02, zhao03], sl. 2.9, 2.10	+	+/−	+/−	+	+/−	+	+/−	prilagodljivost uz slabiju efikasnost i neriješen zajednički prikaz
decentralizirana mreža [nakazawa98], sl. 2.5	−	+	+	+/−	+/−	+/−	+	mogućnost autonomne suradnje uz veliki mrežni promet
nezavisni eksperti [sogo99], sl. 2.7	+/−	−	−	+	+	+	+	skalabilnost uz neriješen ukupni prikaz te rudimentalnu suradnju

† p.s.: prostrane scene, p.p.: pokretni promatrači, u.p.: ukupni prikaz.
‡ N_p: broj promatrača, N_o: broj objekata, ζ: zalihost.

Tablica 2.5: Prilagodljivost arhitektura za porazdijeljeno praćenje.

Dok je kod nezavisnih eksperata komunikacija među promatračima svedena na najmanju moguću mjeru, situacija u decentraliziranoj mreži je dijametralno suprotna: kod ovog pristupa *svaki* promatrač ima ažurni prikaz ukupnog stanja scene. Ovakav pristup omogućava idealne uvjete za sofisticirane i robustne sheme suradnje jer je ukupno stanje dostupno u svim čvorovima pa nema potrebe za centraliziranim odlučivanjem. Međutim, visoki mrežni promet te potreba za komuniciranjem s prostorno udaljenim suradnicima ograničavaju skalabilnost pristupa i njegovu primjenljivost kod prostranih scena.

Jedino rješenje za smanjenje ukupnog prometa u mreži promatrača se svodi na njihovu organizaciju u hijerarhijskoj strukturi kako bi se postiglo sažimanje informacija na lokalnoj razini. Najjednostavnija takva struktura je unaprijed definirana fiksna hijerarhija, kod koje su uloge i veze unaprijed predodređene [dockstader01, collins01]. Takva organizacija je posebno prikladna kod razvedenih zatvorenih scena, gdje je topologija mreže definirana strukturom okoliša u kojem se navigacija odvija. Prolaskom kroz hijerarhijsku stablastu strukturu, informacije o objektima prelaze iz raspršenih i nesigurnih ulaznih podataka u listovima prema organiziranim i pouzdanijim rezultatima u korijenu strukture. Očiti nedostatci fiksne jednorazinske strukture koncentrirani su u nadređenim čvorovima hijerarhije: robustnost sustava je vrlo osjetljiva na njihove kvarove, dok konačna mrežna propusnost njihovih mrežnih veza uvjetuje slabiju skalabilnost sustava. Međutim, neki od ovih nedostataka se mogu riješiti višerazinskom hijerarhijskom organizacijom.

Suptilniji nedostatak fiksne hijerarhije je u složenom postupku konfiguracije, što kod sustava s vrlo velikim brojem promatrača može biti odlučujući faktor. U fiksnoj hijerarhiji, svakom čvoru (osim korijenu stabla) je potrebno reći adresu njegovog nadređenog čvora, što komplicira eksperimentiranje sa raznim položajima promatrača. Slični ali veći problemi se javljaju u višerazinskoj hijerarhiji s pokretnim promatračima, kada se javlja potreba da zbog lokalnosti integracije pojedinačnih mjerenja pokretni promatrač promijeni svoj dotadašnji nadređeni čvor. Neke od ovih problema moguće je rješiti u sustavu sa prilagodljivom hijerarhijom. Prethodni radovi [matsuyama00, zhao03] su razmatrali dvorazinsku hijerarhiju s prilagodljivom prvom i fiksnom drugom razinom (vidi sl. 2.9, 2.12). U prvoj razini, promatrači se grupiraju u udruge prema zajedničkom praćenom objektu, dok u drugoj razini

predstavnici udruga dojavljuju trajektoriju objekta korijenu hijerarhije. Ovakav pristup je posebno prikladan kod primjena sa velikim brojem promatrača i malim brojem praćenih objekata u prostranoj sceni, jer omogućava efikasnu organizaciju promatrača na principu udaljenosti od praćenog objekta.

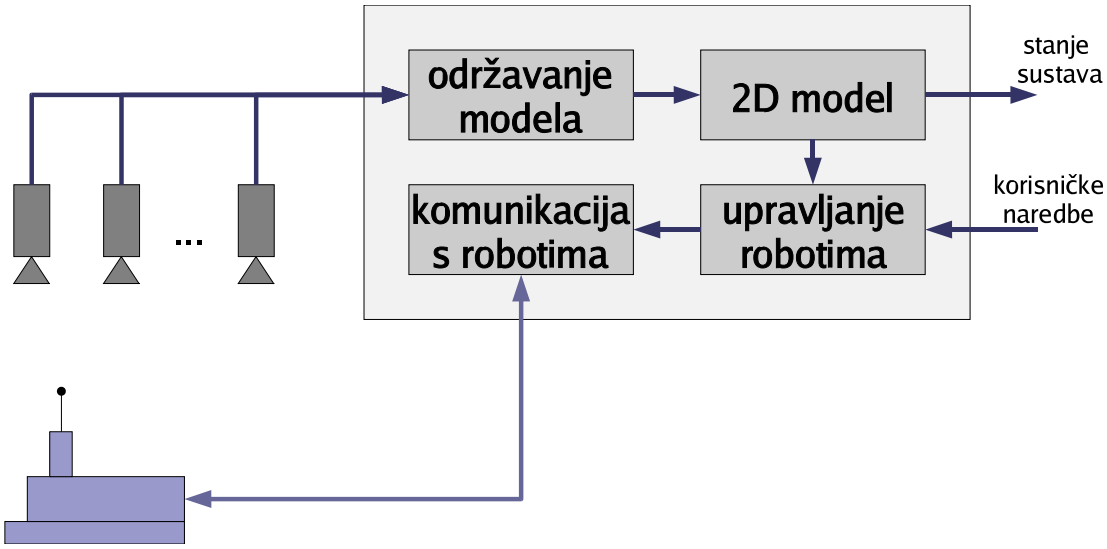
Prilagodljiva hijerarhija se nešto lošije ponaša kod malih vrijednosti ζ, jer je kod prethodnih pristupa sa particioniranjem skupa promatrača prema praćenim objektima, zalihost osnovni kriterij stvaranja hijerarhije. Čak i u slučaju drugih oblika dinamičkog udruživanja, protokol kojim bi se mogao pronaći optimalan raspored promatrača bio bi znatno složeniji nego kod fiksne hijerarije sa eksplicitnim ukupnim stanjem scene. Stoga je u takvim situacijama, u kojima su promatrački resursi ograničeni, predefinirana hijerarhija vjerojatno najprikladnije rješenje.

2.4 Lokalizacija globalnim vidom

Lokalizirati pokretnog robota znači odrediti njegov položaj s obzirom na neki koordinatni sustav [segvic00]. Temeljna podjela metoda lokalizacije je s obzirom na to da li se određeni položaj izražava apsolutno, u odnosu na neki vanjski referentni koordinatni sustav, ili relativno, u odnosu na položaj robota u prošlosti. Relativne metode se svode na integriranje pomaka robota odometrijskim i inercijskim senzorima. Njihove prednosti stoga su jednostavnost i pristupačnost dok su nedostatci nemogućnost interpretacije u nekom vanjskom koordinatnom sustavu te neograničeno gomilanje integracijske pogreške. Zato se najčešće koristi kombinacija odometrije sa nekom apsolutnom metodom, na način da se apsolutna lokacija određuje samo povremeno kada procijenjena integracijska greška postane veća od predefinirane tolerancije.

Zamisao o apsolutnoj lokalizaciji robota porazdijeljenim sustavom računarskog vida je prvi put izložena u radu Kaya i Luoa [kay93]. Unatoč vremenskom odmaku od deset godina, pristup se koristi i u novije vrijeme, kako kod natjecanja u robotskom nogometu [veloso98], tako i kod istraživačkih sustava na području autonomne navigacije [hoover00]. Izvorni rad pretpostavlja monolitni sustav u kojem se slijedovi slika pribavljeni od strane više prostorno razmještenih kamera integriraju u zajednički koherentni prikaz scene, slično opisanom kasnijem radu [collins01], kao što je ilustrirano na sl. 2.11. Zbog sporije tehnologije, objekti se na principu oduzimanja pozadine detektiraju i prate uz pomoć specifičnog sklopovlja. Dobiveni model okoliša koristi podsustav za upravljanje robotima koji korisničke naredbe (npr. "odnesi kutiju AB–14 u sobu 23") prevodi u nizove elementarnih naredbi (npr. "idi naprijed 2.5 metra"), šalje ih robotima te nadgleda njihovo izvođenje. Ukoliko se na planiranom putu javi neka prepreka ili neki drugi pokretni robot, podsustav treba izdati naredbe za pokušaj izbjegavanja zastoja. Iako korišteni roboti imaju samo najosnovnije ultrazvučne senzore za izbjegavanje sudara, upotrebom predloženog pristupa se omogućava njihova djelotvorna navigacija. Ovdje valja naglasiti da su procesne mogućnosti robota iz razmatrane senzorski skromno opremljene klase vrlo skromne pa se upravljanje uglavnom obavlja sa udaljenog računala, putem radio veze. Stoga se donekle može reći da opisani pristup rješava i problem samostalne navigacije, uz napomenu da se autonomna logika nalazi u okviru centraliziranog sustava za lokalizaciju.

Jedan od novijih sustava za navigaciju globalnim vidom [sogo99] pretpostavlja porazdijeljeno upravljanje robotima višeagentskim planiranjem, te mogućnost učenja jednostavnih trajektorija za svakog od upravljanih robota (vidi sl. 2.7). Roboti se u fazi učenja kreću pod upravljanjem ljudskog operatera po željenoj trajektoriji koju memoriraju svi promatrači. Tokom rada sustava, upravljači samostalnih robota (koji se ne izvode na samom robotu, nego

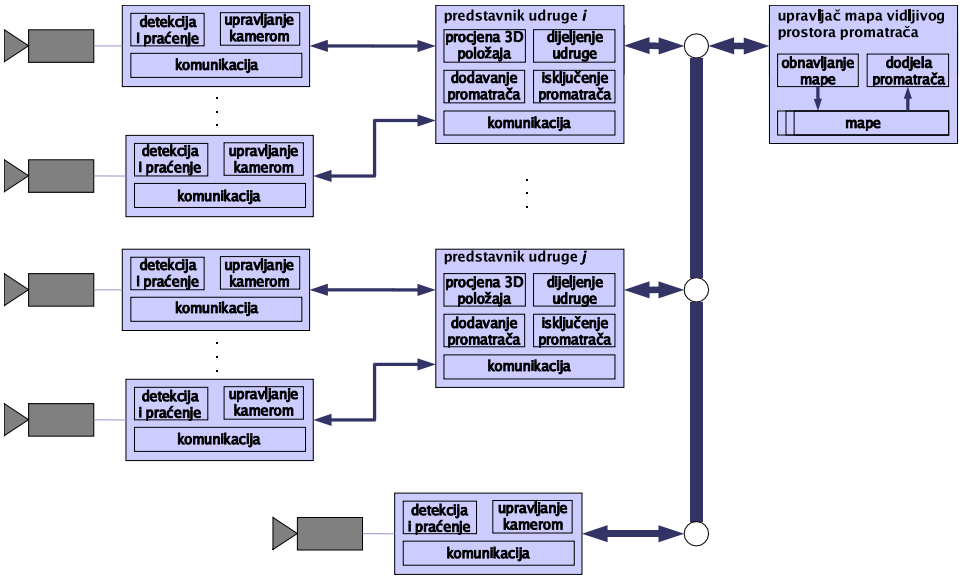


Slika 2.11: Izvorni sustav za lokalizaciju i navigaciju globalnim vidom [kay93].

mu samo šalju naredbe niske razine za upravljanje motorima i očitanje senzora) periodički zahtijevaju instrukcije o daljnjem kretanju. Sustav svakom takvom robotu šalje težinsku listu planova kretanja, pri čemu pojedine planove stvaraju promatrači kojima se dotični robot nalazi unutar vidnog polja. Planovi se generiraju na temelju naučenih željenih trajektorija, te na temelju položaja ostalih robota u neposrednoj blizini. Eventualne korekcije putanje zbog izbjegavanja drugih robota se obavljaju na temelju mape slobodnih područja u okolišu koja se uči također pri inicijalizaciji sustava, sistematskim kretanjem jednog od robota po svim dostupnim dijelovima okoliša. Logika upravljačkog programa tada integrira pojedinačne planove u složeni plan na temelju kojeg se izdaju naredbe pogonskim motorima. Glavni nedostatak ovakvog pristupa je relativna složenost integracije pojedinačnih planova te nemogućnost zadavanja složenijih zahtjeva u porazdijeljenom navigacijskom sustavu, dok je prednost u dobivenoj prilagodljivosti u smislu skalabilnosti i otpornosti na pogreške.

Mapa slobodnih područja kojima se roboti mogu slobodno kretati je ključan alat sustava za lokalizaciju opisanog u [hoover00]. Spomenuta mapa se dobiva na temelju slika dobivenih višestrukim kamerama, i to pronalaženjem presjeka slika dobivenim oduzimanjem pozadine [hoover99] kako je opisano u 2.3.1. U radu se razmatraju problemi dinamičkog planiranja puta, identifikacije robota te upravljanja njegovom brzinom. Identifikacija robota je potrebna jer se pokretni robot na mapi generiranoj porazdijeljenim sustavom za praćenje javlja samo kao jedno od povezanih zauzetih područja. U cilju identifikacije robota odnosno njegove lokalizacije, u radu se predlaže upravljanje kretanjem robota u prepoznatljivim figurama, npr. po kružnici (za određivanje položaja) ili naprijed–nazad (za određivanje orijentacije). U radu su prikazani rezultati uspješnih eksperimenata sa dinamičkim prepravljanjem planiranih putanja dvaju robota uslijed bliskih susreta s preprekama.

Slično održavanju mape slobodnih područja, moguće je održavati mapu vidljivog prostora za svakog agenta promatrača. Takva mogućnost razmatrana je u radu Ukite i Matsuyame [matsuyama00], koji opisuje proširenje temeljne arhitekture opisane u [matsuyama00, matsuyama02] (vidi sl. 2.10). Spomenuto proširenje podrazumijeva uvođenje mape vidljivog prostora za svakog od promatrača, upravljačke komponente koja obavlja operacije nad mapama te njeno dinamičko povezivanje s predstavnicima svih udruga, kao što je ilustrirano na sl. 2.12. Vrijednost elemenata mape može biti "vidljiv", "zaklonjen" ili, početno stanje, "nepoznat". Podatci u mapi se obnavljaju nakon svake poruke odgovarajućeg promatrača



Slika 2.12: Uvođenje mape vidljivog prostora promatrača [matsuyama00].

predstavniku svoje udruge: ukoliko je promatrač prijavio da vidi objekt koji se u sklopu njegove udruge prati, svi volumni elementi prostora između promatrača i objekta se u mapi tog promatrača označavaju kao "vidljivi". Suprotno, kada promatrač izgubi objekt praćen od strane njegove udruge, volumni elementi objekta određeni na temelju rezultata praćenja ostalih članova se u mapi promatrača označavaju kao "zaklonjeni". Upravljačka komponenta registrira promatrače koji su besposleni u svojim udrugama jer praćeni objekt trenutno nije vidljiv iz njihove točke gledanja, te im (preko predstavnika njihove trenutne udruge) pokušava dodijeliti poslove gdje će biti korisniji. Dvije najjednostavnije mogućnosti uključuju proglašavanje promatrača slobodnim strijelcem i dodjelu promatrača udruzi čiji objekt nije u zaklonjenoj zoni njegove mape. Predloženo proširenje omogućava dinamičku prilagodbu sustava strukturi scene na na vrlo primamljiv i intuitivan način. Ipak, noviji rad istih autora [matsuyama02] ne razmatra opisane preinake: njihovim uvođenjem prethodno decentralizirana arhitektura postaje značajnim dijelom potpuno centralizirana (sa svim prednostima i nedostatcima koje takav pristup nosi) što se vrlo vjerojatno ne slaže s prvotnom intencijom autora.

Poglavlje 3

Višeagentska arhitektura za porazdijeljeno praćenje

U ovom poglavlju, detaljno će se opisati arhitektura za porazdijeljeno praćenje u razvedenom zatvorenom prostoru, čija temeljna značajka je unaprijed određena hijerarhijska struktura. Predložena arhitektura je posebno prilagođena praćenju kretanja malih robota po horizontalnoj ravnini scene te mogućnostima primjene u sustavu za samostalnu navigaciju pokretnih robota temeljenom na zamisli globalnog vida..

Pretpostavke koje su uvjetovane razmatranom primjenom biti će detaljnije razrađene u odjeljku 3.1, u okviru aspekata porazdijeljenog praćenja koji su opisani u odjeljku 2.3. Odjeljak 3.2 opisuje detalje osnovne jednorazinske arhitekture, u okviru koje su izvedeni eksperimenti koji su opisani u kasnijim poglavljima. Konačno, neka zanimljiva proširenja osnovne arhitekture bit će prikazana u odjeljku 3.3.

3.1 Pretpostavke i zahtjevi

S obzirom na uopćeni opis problema porazdijeljenog praćenja koji je dan u Uvodu (vidi sl. 1.1) te prethodna arhitektonska razmatranja iz poglavlja 2, može se reći da područje primjene željenog sustava zadovoljava sljedeće pretpostavke:

- razmatra se razvedena zatvorena scena u kojoj su aktivni promatrači s prethodno umjerenim sklopovljem međusobno spojeni brzom lokalnom mrežom;
- ne pretpostavlja se velika zalihost promatrača;
- broj promatrača u svakoj razini hijerarhije je proizvoljan, ali ne prevelik (nije realistično imati 20 promatrača u jednoj prostoriji);
- tokovi komunikacije među promatračima i agentima koji obavljaju prostornu integraciju podataka tvore stablastu hijerarhijsku strukturu;
- promatrači međusobno nisu sinkronizirani i ne moraju imati jednake performanse;
- latencija pri komunikaciji prvenstveno ovisi o intenzitetu mrežnog prometa, pa ne ovisi o položajima agenata sudionika, te nije konstantna u vremenu;
- praćeni objekti su relativno mali u odnosu na dimenzije scene,

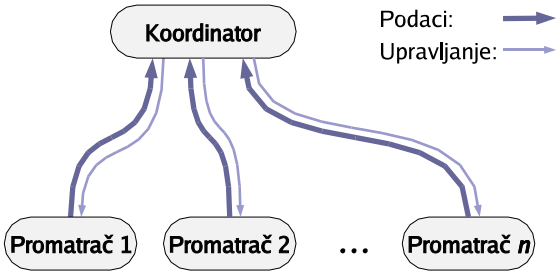
- prostorna integracija pojedinačnih mjerenja se odvija na razini cjelokupnih objekata i to samo na temelju procijenjenih položaja jer su praćeni objekti mali pa ih je teško identificirati.
- kretanje praćenih objekata ima dva stupnja slobode pa je njihov 3D položaj moguće procijeniti i iz jednog pogleda;

U skladu s izloženim pretpostavkama, ključne sposobnosti željenog sustava su:

1. praćenje jednog ili više objekata jednim promatračem;
2. integracija podataka dobivenih od promatrača u zajednički prikaz scene;
3. rad u stvarnom vremenu uz nepredvidljivu latenciju računalne mreže i različite performanse promatrača.
4. robustnost s obzirom na uklanjanje postojećih ili dodavanje novih promatrača u sustav;
5. koordinacija odgovornosti promatrača u cilju boljeg praćenja stanja u sceni;

3.2 Temeljna hijerarhijska arhitektura

Pri oblikovanju arhitekture željenog sustava, potreba za porazdijeljenom organizacijom se javlja već kao posljedica zahtjeva da se algoritmi računarskog vida izvršavaju nad višestrukim slijedovima slika u sustavu za rad u stvarnom vremenu. Zbog složenosti postupaka obrade slike, korisno je osigurati da svaki promatrač dobije svo vrijeme dodijeljenog procesora za analizu svog slijeda. U skladu s tom idejom, integracijski i koordinacijski zadatci mogu se obavljati na hijerarhijski višoj razini, u okviru agenta koordinatora koji također raspolaže vlastitim procesorom. Opisana arhitektura je ilustrirana na sl. 3.1: promatrači šalju mjerne podatke koordinatoru, koji ih koristi za upotpunjavanje i obnavljanje ukupnog prikaza scene. Na temelju ukupnog pogleda, agent koordinator korigira odgovornosti promatrača u cilju što boljeg praćenja stanja u sceni. Ukoliko se koordinator promatra kao dio komunikacijske infrastrukture, sustav organiziran prema sl. 3.1 zadovoljava agentski test [huhns99a] citiran u Uvodu: kad god se novi promatrač prijavi koordinatoru, odgovornosti svih ostalih promatrača se prilagođavaju u smislu bolje učinkovitosti ukupnog sustava.



Slika 3.1: Temeljna hijerarijska arhitektura za porazdijeljeno praćenje.

Agentska arhitektura sa sl. 3.1 pretpostavlja značajnu autonomiju komponenti koje u njoj sudjeluju te komunikaciju strukturiranim porukama na značajnoj razini apstrakcije. Važno svojstvo takve organizacije je da se zadatci pokušavaju obaviti u što lokalnijem kontekstu, što sa sobom nosi višestruke prednosti:

1. upravljanje sustavom je tim jednostavnije što pojedinačne komponente imaju veći stupanj autonomije: korištenjem autonomnih komponenata (agenata) izbjegava se opasnost da rast sustava bude ograničen složenošću upravljačke komponente, te se postiže povećanje prilagodljivosti sustava u smislu različitih uvjeta rada;

2. obrađeni podatci u simboličkom obliku su obično mnogo sažetiji od ulaznih ikoničkih podataka: komunikacija na takvoj apstraktnijoj razini postavlja manje zahtjeve na mrežnu propusnost, pa se zaobilazi opasnost da mreža postane ograničavajući čimbenik performanse sustava;

3. lokalna obrada za sobom povlači mogućnost usporednog rada što, u kombinaciji s prethodno spomenutim manjim mrežnim prometom i boljom prilagodljivosti, otvara mogućnosti za vrlo značajna povećanja performanse u odnosu na odgovarajuću monolitnu arhitekturu.

3.2.1 Agenti promatrači

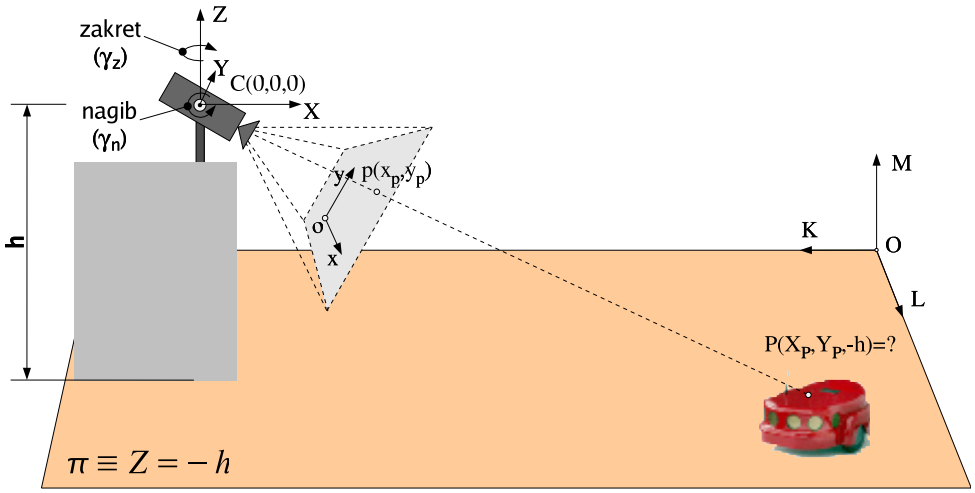
U predloženoj arhitekturi, svaki agent promatrač je opremljen kamerom pričvršćenom na upravljivo postolje s dva rotacijska stupnja slobode: zakretom i nagibom, kao što je prikazano na sl. 3.2. Kod svih poznatih izvedbi, os zakreta postolja je nepokretna dok se, zbog jednostavnosti mehaničke izvedbe, os nagiba pomiče u skladu s kutom zakreta zajedno s cijelom kamerom. Temeljni zadatci promatrača su detekcija i praćenje objekata zadane vrste, te upravljanje smjerom gledanja pridružene kamere u cilju poboljšanog praćenja tekućeg objekta ili traženja novih objekata.

Slika 3.2: Upravljiva kamera s dva stupnja slobode.

Geometrija promatrača

Željeni sustav za porazdijeljeno praćenje sadrži veći broj promatrača pa je tako, pored koordinatnih sustava slike (o, x, y) i promatrača (postolja) (C, X, Y, Z), potrebno definirati i zajednički referentni koordinatni sustav scene (O, K, L, M). Važno svojstvo razmatrane primjene je da se praćeni objekti kreću po poznatoj ravnini podloge scene π, pa je stoga s njom pogodno poravnati ravninu (O, K, L) k.s. svijeta, što ujedno znači i da je os M poravnata s vektorom normale ravnine π. U istom duhu, za svakog od promatrača pogodno je odabrati

koordinatni sustav za koji se uspravna os Z podudara s osi M k.s. svijeta, te približno poravnati os zakreta postolja kamere s tim smjerom. U tom slučaju, 3D Euklidska transformacija iz k.s. promatrača u k.s. svijeta ima 3 translacijska i samo jedan rotacijski stupanj slobode kao što je ilustrirano na sl. 3.3.



Slika 3.3: Geometrija agenta promatrača.

Agent promatrač može na temelju intrinsičnih parametara kamere te središta segmentirane regije $p(x_p, y_p)$ slikovnoj ravnini odrediti 3D položaj pravca u k.s. kamere na kojem se nalazi procijenjeni položaj objekta. Da bi se jednoznačno odredio položaj objekta, potrebno nam je dodatno ograničenje koje po pretpostavci predstavlja ravnina gibanja π. Međutim, da bi se ograničenje moglo primijeniti, potrebno je ravninu $\pi \equiv Z = -h$ izraziti u k.s. kamere. U idealnom slučaju, Euklidsku transformaciju M_{PK} iz k.s. promatrača u k.s. kamere bilo bi moguće opisati s dvije rotacije oko koordinatnih osi k.s. promatrača. Zbog mehaničkih neprikladnosti postolja (projekcijski centar kamere se ne može postaviti u presjecištu osi zakreta i nagiba) to u stvarnosti nije moguće postići pa je transformaciju potrebno kalibrirati za svaki par kutova zakreta i nagiba, kao što je opisano u poglavlju 4. Primjenom tih dvaju ograničenja određen je procijenjeni položaj objekta u k.s. kamere, a odatle se primjenom M_{PK}^{-1} lako dobiva položaj objekta u k.s. promatrača $P(X_P, Y_P, -h)$.

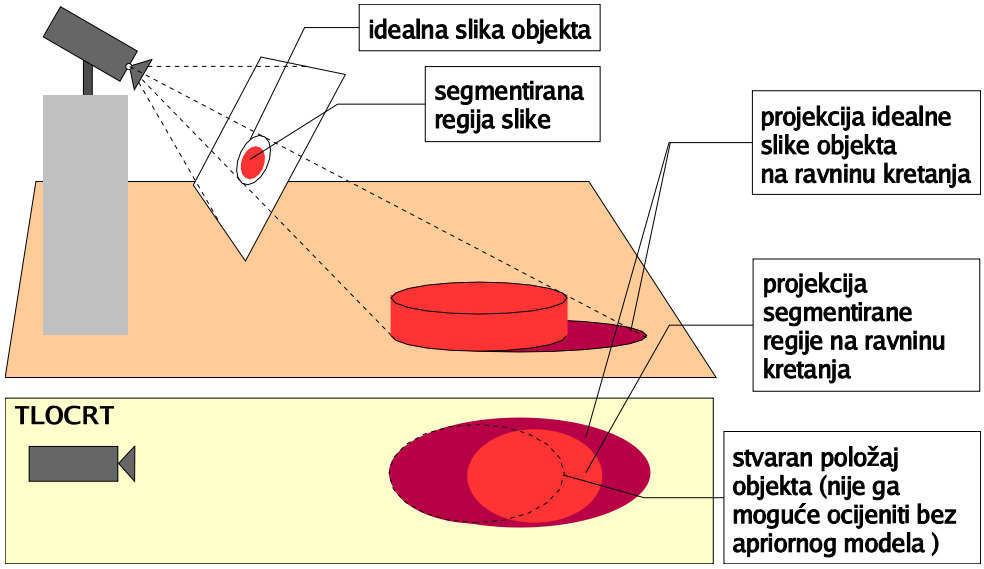
Zbog ravninskog gibanja, za položaj objekta su značajne samo vrijednosti koordinata (x, y) u k.s. promatrača, odnosno (k, l) u k.s. svijeta. Koordinatni sustav promatrača nema nikakav značaj izvan svog lokalnog konteksta, pa će promatrači sve svoje mjerne rezultate izražavati u koordinatama svijeta. Lako se vidi da je transformacija iz k.s. promatrača u k.s. svijeta relativno jednostavna, a svodi se na Euklidsku 2D transformaciju s dva translacijska i jednim rotacijskim stupnjem slobode.

Kao zaključak odjeljka, naveden je sažeti popis parametara koji su potrebni promatraču za određivanje položaja objekta u k.s. svijeta:

- unutrašnji parametri kamere (K, k_1, k_2);
- unutrašnji parametri postolja (M_{PK});
- vanjski parametri promatrača:
  - visina postolja u odnosu na ravninu gibanja h;
  - položaj postolja u ravnini gibanja $(k_0, l_0) \in (O, K, L)$;
  - orijentacija postolja $\phi_0 = \sphericalangle(X, K)$.

Detekcija objekata

Zbog zahtjeva za radom u stvarnom vremenu i malene veličine objekata, njihov položaj u slici se određuje nekom od jednostavnih metoda amplitudne segmentacije. Nažalost, dobro umjereno sklopovlje ne jamči jednostavno određivanje granica praćenog objekta, kako zbog konačne visine objekata, tako i zbog mogućnosti za njihovu nepotpunu segmentaciju. Taj problem je prikazan na sl. 3.4: iako je umjerenim sklopovljem moguće precizno odrediti projekciju segmentirane regije iz ravnine slike na ravninu kretanja objekata, za kvantitativno određivanje nesigurnosti dobivenog mjerenja ipak je potrebno imati apriorni model koji sadrži znanje o dimenzijama praćenih objekta.



Slika 3.4: Nesigurnost pri određivanju položaja praćenog objekta.

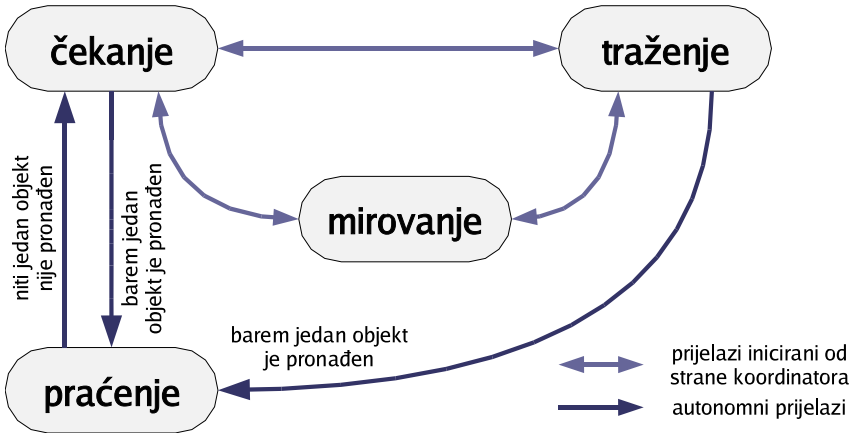
Međutim, u svim eksperimentima su kao praćeni objekti korišteni niski roboti kod kojih prosječna lokalizacijska pogreška uslijed efekta prikazanog na sl. 3.4 ne iznosi više od 20 cm. Ta pogreška nije bila kritična za rad eksperimentalnog sustava, tim više što se kod integracije mjerenja više promatrača pojedinačne greške barem djelomično poništavaju. Stoga su u eksperimentima, položaji objekata jednostavno procijenjeni kao projekcije središta segmentiranih regija na ravninu gibanja, bez ikakvog modela nesigurnosti, upravo kao što je to prikazano na sl. 3.3. Mogućnosti za precizniju lokalizaciju objekata u kojima se opisana greška ne zanemaruje su opisane u 3.3.1.

Funkcionalnost promatrača

Glavni zadatak agenata promatrača je detekcija i praćenje željenih objekata u trenutnom vidnom polju. Traženi objekti se trebaju detektirati relativno jednostavnim postupcima koji se mogu obaviti u stvarnom vremenu.
Svaki promatrač je povezan dvosmjernom vezom s predefiniranim koordinatorom. Gotovo sve odlazne poruke promatrača su asinkrone u smislu da se odgovor koordinatora ne očekuje, zbog potrebe za radom u stvarnom vremenu. Prilikom dodavanja novog promatrača u sustav, promatrač uspostavlja vezu s koordinatorom te utvrđenim protokolom usklađuje sat sa referentnim satom koordinatora. Postupak sinkronizacije se periodički ponavlja, na zahtjev promatrača.
Prilikom svake promjene smjera gledanja, promatrač treba obavijestiti koordinatora o dijelu scene koji se trenutno nalazi unutar vidnog polja promatrača. Nakon svake obrađene slike,

promatrač koordinatoru šalje popis detektiranih objekata potpisan vremenskim opisnikom koji odgovara trenutku pribavljanja analizirane slike. Prirodno, svi geometrijski podatci se izražavaju u referentnom 2D k.s. svijeta, dok se vremenski opisnici odnose na referentni sat koordinatora.
Promatrač je sposoban raditi u četiri različita načina rada (stanja, ponašanja), a prijelazi među stanjima mogu biti ili autonomni ili inicirani od strane koordinatora kao što je ilustrirano na sl. 3.5.

- `mirovanje`: promatrač ne mijenja smjer gledanja, odgovoran je za motrenje dijela scene koji mu se trenutno nalazi u vidnom polju;
- `čekanje`: promatrač ne mijenja smjer gledanja sve dok ne pronađe barem jedan objekt, a onda prelazi u stanje `praćenje`;
- `traženje`: promatrač sistematski mijenja smjer gledanja sve dok ne pronađe barem jedan objekt, a onda prelazi u `praćenje`;
- `praćenje`: promatrač mijenja smjer gledanja kamere u skladu s kretanjem dodijeljenog objekta sve dok ga ne izgubi, kada prelazi u `čekanje`.



Slika 3.5: Prijelazi među načinima rada agenta promatrača.

Agentska arhitektura promatrača

Opisana funkcionalnost agenata promatrača u mnogome određuje njihovu arhitekturu u kontekstu razmatrane primjene. Predložena arhitektura nasljeđuje shemu apstraktnog agenta danu na sl. 2.1: temeljna funkcionalnost se svodi na ciklus *gledaj* - *razmisli* - *djeluj*, pri čemu se apstraktne operacije sastoje od sljedećih konkretnih postupaka:

- **gledaj**: očitavanje smjera gledanja kamere, dohvat nove slike, pribavljanje poruka;
- **razmisli**: analiza pristiglih poruka, obrada slike, detekcija i praćenje objekata, interpretacija rezultata u k.s. svijeta;
- **djeluj**: izdavanje naredbi za promjenu smjera gledanja, slanje poruka s rezultatima obrade, dijagnostički ispis.

3.2.2 Agenti koordinatori

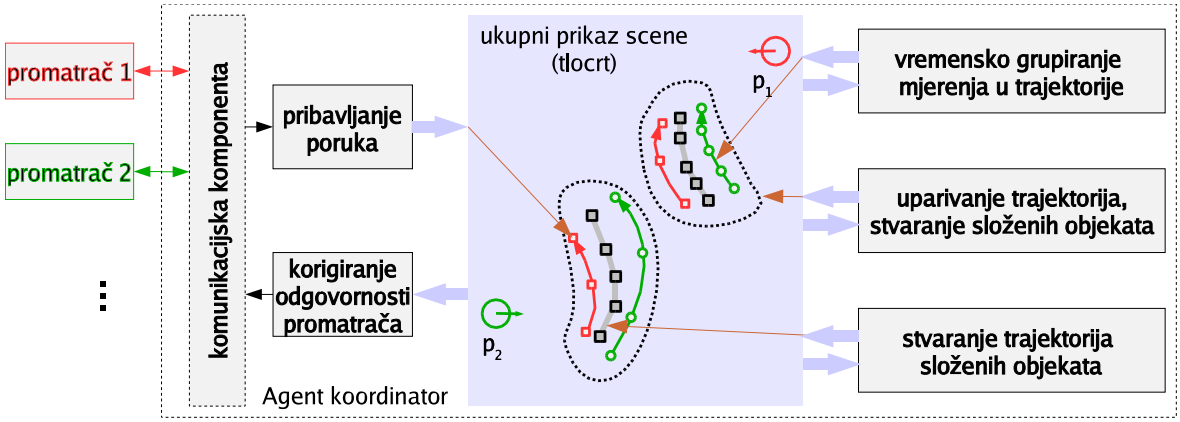
Agent koordinator je zadužen za integraciju (grupiranje) slijedova pojedinačnih položaja praćenih objekata koji su dojavljeni od strane promatrača, te njihovu interpretaciju u cilju prilagodbe cijelog sustava stanju u sceni. Integracijski zadatci se svode na iterativno obnavljanje ukupnog prikaza scene. Za razliku od klasičnih primjena raspoznavanja uzoraka gdje se uzorci analiziraju i klasificiraju jednokratno, u ovom slučaju je opaženo stanje scene kao rezultat analize potrebno tokom vremena stalno obnavljati i korigirati, u beskonačnoj petlji. Zbog toga podatkovna struktura za pohranjivanje ukupnog prikaza te pripadni mehanizmi za njenu podatkovnu apstrakciju čine središnji dio funkcionalnosti koordinatora. Struktura sadrži podatke na četiri hijerarhijske razine, uz postupni rast apstrakcije, kao što je ilustrirano na sl. 3.6.

1. razina mjerenja promatrača: sadrži pojedinačne položaje objekata koji su u slici označeni malim kvadratima, a dobiveni su iz mjernih poruka promatrača;

2. razina pojedinačnih trajektorija: sadrži trajektorije objekata koje su u slici označene tankim krivuljama, a dobivaju se vremenskom integracijom pojedinačnih mjerenja;

3. razina percipiranih objekata: sadrži koordinatorove ocjene položaja objekata koje su u slici označene većim kvadratima, a dobivene su prostornom integracijom podataka svih relevantnih promatrača, na principima bliskosti i sličnosti oblika pojedinačnih trajektorija;

4. razina konačnih trajektorija: sadrži trajektorije percipiranih objekata koje su u slici označene debljim krivuljama, a dobivene su vremenskom integracijom podataka iz prethodne razine.



Slika 3.6: Održavanje ukupnog prikaza scene u okviru koordinatora.

Iako i promatrači sadrže postupke za prilagodbu smjera gledanja razvoju situacije u sceni, odgovarajuća razmatranja su ograničena lokalnim kontekstom. Sofisticirana, strateška prilagodba sustava stanju u sceni zahtijeva dostupnost ukupnog prikaza scene pa je logičan izbor da se takvi postupci svrstaju u sklopu koordinatora. Tako koordinator na temelju više slijedova mjerenja gradi ukupan prikaz scene, i koristi ga za dinamičko prilagođavanje odgovornosti promatrača stanju u sceni. Za izgradnju i održavanje ukupnog prikaza, koordinator mora sadržavati četiri temeljna postupka koji definiraju uzlazne prijelaze među hijerarhijskim razinama prikaza scene, i to: (i) pribavljanje poruka promatrača, (ii) grupiranje mjerenja u trajektorije, (iii) integracija pojedinačnih trajektorija u percipirane objekte, te (iv) formiranje trajektorija percipiranih objekata.

Grupiranje trajektorija i koordinacija promatrača

Pribavljanje mjerenja te formiranje trajektorija na razinama mjerenja i percipiranih objekata su općeniti postupci koji se mogu sresti i izvan konteksta porazdijeljenog praćenja. To međutim ne vrijedi za grupiranje trajektorija i koordinaciju promatrača, pa će se ti postupci ovdje dodatno razjasniti.

Pri traženju korespondencije između pojedinačnih trajektorija, može se koristiti pretpostavka da su traženi objekti razmjerno niski u odnosu na dimenzije scene. U tom slučaju sigurno vrijedi da su prividni položaji jednog te istog objekta kod različitih promatrača međusobno bliski, bez obzira na položaj objekta u sceni. Slijedeći pretpostavku, skup preliminarnih hipoteza za uparivanje trajektorija se može dobiti primjenom kriterija blizine trenutnih položaja objekata. Najbolju hipotezu je moguće potražiti uz dodatni kriterij da recentni odsječci pojedinačnih trajektorija moraju imati međusobno sličan oblik. Nakon što je korespondencija ostvarena, precizniji položaj objekta u zadanom trenutku se može ocijeniti uzimajući u obzir i parametre modela nesigurnosti pojedinačnih mjerenja.

Za razliku od ostalih zadataka, kod koordinacije treba tražiti kompromis između međusobno proturječnih ciljeva. Za svaki od percipiranih objekata, najpreciznije praćenje se ostvaruje usmjeravanjem svih kamera u njegovom smjeru, međutim sustav treba pratiti sve objekte u sceni. Pored toga, sustav treba čim prije opaziti ulazak novog objekta u scenu, što se, bez daljnjih pretpostavki, može postići samo pravilnim rasporedom vidnih polja kamera po sceni. Svaki od ciljeva u izolaciji se jednostavno rješava, međutim, konačni postupak mora rješavati spomenute konflikte odnosno pronaći kompromisno rješenje.

Očito je da svako područje primjene ima specifične zahtjeve u vezi s prioritetom praćenja pojedinih situacija. Tako kod praćenja pokretnih robota prioritet kod praćenja imaju natovareni i sporiji roboti, te uska područja gdje je potrebno precizno mimoilaženje. Različita pravila vrijede kod praćenja nogometnih susreta, jer je tamo potrebno znati da su za televizijsku publiku vrlo zanimljivi igrači oko lopte, te da je pored toga posebno interesantan kazneni prostor momčadi na čijem dijelu terena se igra odvija. Ipak, pored takvih specifičnih zahtjeva, postoje i intrinsični zahtjevi porazdijeljenog praćenja, npr, da treba rasporediti promatrače da prate čim veći broj objekata, uz što veću preciznost praćenja. U okviru ovog odjeljka, razmatrat će se strategije u kojima figuriraju upravo takvi intrinsični zahtjevi, dok će se mogućnosti za izražavanje specifičnih zahtjeva opisati u odjeljku 3.3.3.

U kontekstu oblikovanja intrinsičnih koordinacijskih strategija, ravninsko gibanje praćenih objekata je vrlo važna pretpostavka jer se tada lokalizacija može obaviti samo jednim promatračem. Stoga se mogu razmatrati sljedeći koordinacijski scenariji, uz konvenciju da se broj promatrača i praćenih objekata označavaju s N_o i N_p:

$N_p \geq N_o$:
Odgovornost za praćenje svakog objekta je dodijeljena zasebnom promatraču, a eventualni slobodni promatrači nadziru scenu u cilju otkrivanja novih objekata. Kombinacija $N_p < N_o$ se u okviru ovog scenarija ne razmatra zbog problema s obradom situacije u kojoj se dva objekta koje zajednički prati samo jedan promatrač počnu gibati u suprotnim smjerovima. Zadatak koordinatora je svakom objektu dodijeliti promatrača koji će ga pratiti uz čim veću prosječnu preciznost praćenja u sustavu. Koordinacijski algoritam koji donekle zadovoljava izložene zahtjeve je opisan u poglavlju 5.

$N_p < N_o$:
Koordinator particionira promatrače na pratioce i motritelje, pri čemu motritelji imaju šire vidno polje. Smjerovi gledanja motritelja se postavljaju tako da pokrivaju što veći dio ravnine kretanja. Moguća je i izvedba u kojoj su motritelji izvedeni kao nepokretni

promatrači, čiji položaj i smjer gledanja su postavljeni manualnim metodama. Zadatak motritelja je gruba lokalizacija objekata, dok pratioci fiksiraju redom pojedine objekte i određuju njihove preciznije položaje i identitet. Zadatak koordinatora je dinamična prilagodba smjera gledanja motritelja gibanju objekata u sceni, na način da najbolje budu pokrivena područja u kojima se objekti zadržavaju najdulje. Pored toga, koordinator dodjeljuje objekte motriteljima, ponovo uz kriterij čim veće ukupne preciznosti praćenja u sustavu.

Funkcionalnost agenta koordinatora

Agent koordinator je povezan dvosmjernim vezama s proizvoljnim brojem promatrača koji međusobno nisu sinkronizirani i mogu imati različite učestalosti slanja mjerenja.
U postupku prijave, koordinator mora saznati položaj promatrača te podskup ravnine gibanja koji se nalazi u njegovom vidnom polju. Naknadna uključenja promatrača u rad sustava bi trebala biti moguća, kao i odjava nekog od promatrača dok sustav još radi.
Tijekom rada sustava, koordinator treba ispravno interpretirati sljedeće poruke od strane promatrača:

1. zahtjev za trenutnim odgovorom, tijekom sinkronizacije satova promatrača i koordinatora,

2. izvještaj o promjeni dijela ravnine gibanja koji se nalazi u vidnom polju promatrača uslijed pomaka smjera gledanja,

3. izvještaj o detektiranim objektima koji se sastoji od liste položaja objekata te vremenskog opisnika;

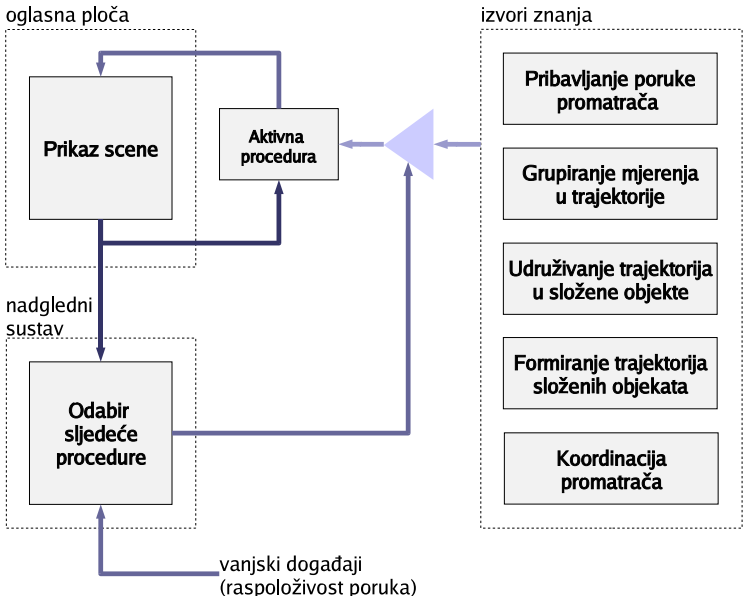
Prikaz scene se kontinuirano obnavlja grupiranjem mjerenja u pojedinačne trajektorije, njihovom integracijom u percipirane objekte te stvaranjem trajektorija percipiranih objekata. Koordinator korigira odgovornosti promatrača slanjem poruka koje iniciraju promjenu njegovog stanja, prema sl. 3.5. Sve odlazne poruke koordinatora su asinkrone: odgovori na koordinacijske poruke nisu predviđeni, kako se performansa sustava ne bi smanjila u slučaju velike latencije nekog od promatrača.

Unutrašnja arhitektura agenta koordinatora

Kod agenata promatrača, redoslijed obavljanja pojedinih procedura koje čine apstraktnu operaciju *razmisli* je unaprijed definiran: da bi se poslao izvještaj koordinatoru, prvo se mora odrediti položaj detektiranih objekata u k.s. svijeta, a još prije toga je u pribavljenoj slici potrebno naći objekte. Situacija nije tako jednostavna kod koordinatora, jer ritam postupaka integracije prvenstveno ovisi o protoku stvarnog vremena i ne mora se poklapati s dotokom promatračkih poruka. Stoga unutrašnja arhitektura koordinatora ne može biti oblikovana na izravan način kao što je to slučaj s promatračima, nego će biti potrebna detaljnija arhitektonska analiza.

Kao što se vidi na sl. 3.6, agent koordinator uključuje pet glavnih procedura, i to: pribavljanje poruka, grupiranje mjerenja u trajektorije (vremenska integracija), grupiranje trajektorija u percipirane objekte (prostorna integracija), stvaranje trajektorija percipiranih objekata (još jedna vremenska integracija) te dodjela odgovornosti promatračima. Ove procedure prevode podatke iz hijerarhijski nižih u hijerarhijski više razine zajedničkog prikaza scene. Redoslijed pokretanja procedura ovisi o dinamičkim uvjetima koje je moguće detektirati te prilikom izvođenja, npr, pribavljanje novog mjerenja, ili približavanje nekog od objekata rubu

vidnog polja nekog od promatrača koji ga prate. Potrebna oportunistička aktivacija se može izraziti prema obrascu oglasne ploče [pfleger98, huhns99], koji je već spomenut u 2.1.4 kao oblik višeagentske koordinacije temeljen na organizacijskom strukturiranju. Glavni subjekti u tom obrascu su izvori znanja (spomenute procedure), oglasna ploča (ukupni prikaz scene), te nadgledna komponenta koja upravlja redoslijedom izvršavanja izvora znanja (vidi sl. 3.7).



Slika 3.7: Unutrašnja arhitektura agenta koordinatora.

Kod opisanog arhitektonskog obrasca, izvori znanja ne komuniciraju izravno, nego svoja zapažanja bilježe na centraliziranom odlagalištu podataka (oglasnoj ploči) i tako nezavisno grade djeliće mozaika koji bi trebao konvergirati rješenju. Glavna razlika između predložene organizacije i klasičnih sustava oglasne ploče je u tome što se ovdje stalno radi na dotjerivanju jednog te istog rješenja (ukupnog stanja scene), dok klasični sustavi rade u epizodama koje su međusobno uglavnom nepovezane (npr, prepoznavanje izgovorene rečenice [pfleger98]).

Agentska arhitektura koordinatora

Kao i kod promatrača, vanjska arhitektura agenta koordinatora također nasljeđuje shemu apstraktnog agenta danu na sl. 2.1. Apstraktne operacije *gledaj* - *razmisli* - *djeluj*, pri tome sadrže od sljedeće konkretne postupke:

- **gledaj**: pribavljanje mjerenja promatrača;
- **razmisli**: održavanje ukupnog prikaza scene (vidi sl. 3.6)
- **djeluj**: koordinacija promatrača; dijagnostički ispis.

3.2.3 Komunikacijski protokol

Komunikacija u kontekstu stvarnog vremena

Protokol komunikacije među promatračima i koordinatorom je oblikovan u skladu s prethodno izloženim zahtjevima sustava. Vremenski interval između slanja i primanja poruke je vrlo teško predvidjeti jer sudionici komunikacije imaju velike apetite za procesorom računala: promatrači obrađuju sliku, a koordinator uparuje pojedinačne trajektorije. Agenti

stoga mogu samo povremeno ostaviti svoj glavni posao i pogledati je li u međuvremenu došla kakva poruka. Takvo primanje poruka je nažalost povezano s provjerama u diskretnim vremenskim trenutcima te vremenski skupim promjenama konteksta procesora, a sve to uzrokuje povećanje kašnjenja primitka poruke.

Svaka poruka se sastoji od svog informacijskog sadržaja i vremena kreiranja koje se može koristiti za interpretaciju poruke u stvarnom vremenu. Značaj poruka je obično vezan za neki određeni trenutak, kao što je trenutak pribavljanja slike u kojoj su detektirani objekti čije položaje poruka prijavljuje. Zato svaka poruka ima svoj rok trajanja, unutar kojeg mora biti obrađena ili inače zastarijeva. Rok trajanja ovisi o konkretnoj primjeni, pa tako, primjerice, poruku koja zahtijeva praćenje nekog pokretnog objekta a poslana je 500 ms prije primitka, ima smisla obrađivati samo kod relativno sporih objekata, koji se ne gibaju brzinom većom od oko 0.5 m/s.

Jedan od najograničavajućih zahtjeva koje agenti moraju zadovoljavati je rad u stvarnom vremenu, pa je temeljna značajka protokola njegova asinkronost, tj. primitci poruka u pravilu se ne potvrđuju. Poruke protokola su prvenstveno informativnog karaktera, tako da ispravan rad pošiljaoca može ne ovisiti ni o kakvim efektima poruke na primaoca. Štoviše, obzirom na to da poruka može do primaoca doći i nakon što joj istekne rok trajanja, pošiljaoc mora biti spreman i na najgori slučaj, u kojem primaoc neće nikad obraditi poruku.

Poruke promatrača

`sinkronizacija():`

Cilj ove poruke je usklađivanje sata promatrača prema referentnom satu koordinatora. Od koordinatora se nakon primanja ove poruke zahtijeva neposredan odgovor porukom `potvrda`. Promatrač šalje poruku `sinkronizacija` i mjeri vrijeme do primitka potvrde. Neka je trenutak slanja poruke $T_0^{(p)}$, trenutak primanja potvrde $T_1^{(p)}$, po vremenu agenta promatrača; zbog latencije mreže vrijedi $T_0^{(p)} < T_1^{(p)}$. Neka je nadalje $T_p^{(k)}$ trenutak kreiranja poruke potvrde po vremenu koordinatora. To vrijeme je dostupno i promatraču, jer se nalazi u zaglavlju poruke `potvrda`. Označimo trenutak primitka povratne poruke prema satu koordinatora s $T_1^{(k)}$, i ocijenimo ga kao sumu trenutka slanja poruke i polovine ukupnog vremena latencije:

$$T_1^{(k)} = T_p^{(k)} + \frac{T_1^{(p)} - T_0^{(p)}}{2}. \tag{3.1}$$

Tada se razlika među satovima ΔT može ocijeniti kao:

$$\Delta T = T_1^{(k)} - T_1^{(p)} = T_p^{(k)} - \frac{T_1^{(p)} + T_0^{(p)}}{2}. \tag{3.2}$$

Radi postizanja veće točnosti, cijeli postupak se ponavlja veći broj puta, a kao konačna vrijednost uzima se srednja vrijednost pojedinačnih vremena ΔT.

`prijava(ime, x,y)`

Promatrač se prijavljuje za rad koordinatoru i šalje svoje ime i položaj u referentnom k.s. svijeta (x,y).

`vidno_polje(lista_područja)`

Ovom porukom, promatrač izvještava koordinatora o dijelovima ravnine kretanja koji se nalaze unutar njegovog vidnog polja.

`mjerenje(lista_objekata)`
: Nakon obrade slikovnih okvira u kojima je detektiran jedan ili više objekata, promatrač ovom porukom šalje koordinatoru položaje detektiranih objekata u k.s. svijeta.

`odjava()`
: Ovom porukom promatrač eksplicitno javlja koordinatoru da prekida rad, te da koordinator više ne može računati na njega. Promatrač nije obavezan poslati ovu poruku.

Poruke koordinatora

`potvrda():`
: Neposredni odgovor na sinkronizacijsku poruku promatrača. Ovo je jedina sinkrona poruka u cijelom protokolu.

`usmjeri(x,y):`
: Dodjela odgovornosti promatraču za praćenje dijela ravnine gibanja oko točke (x,y) u referentnom k.s. svijeta. Pri tome promatrač i dalje izvještava o stanju u sceni.

`prati(x,y):`
: Dodjela odgovornosti promatraču za praćenje objekta koji se nalazi u blizini točke (x,y) u referentnom k.s. svijeta. Ukoliko promatrač ne detektira objekt u blizini zadane točke, treba preći u stanje `čekanje`.

`traži():`
: Uputa promatraču da sustavno mijenja smjer gledanja kamere dok ne pronađe bilo koji objekt. Nakon toga promatrač treba pratiti taj objekt.

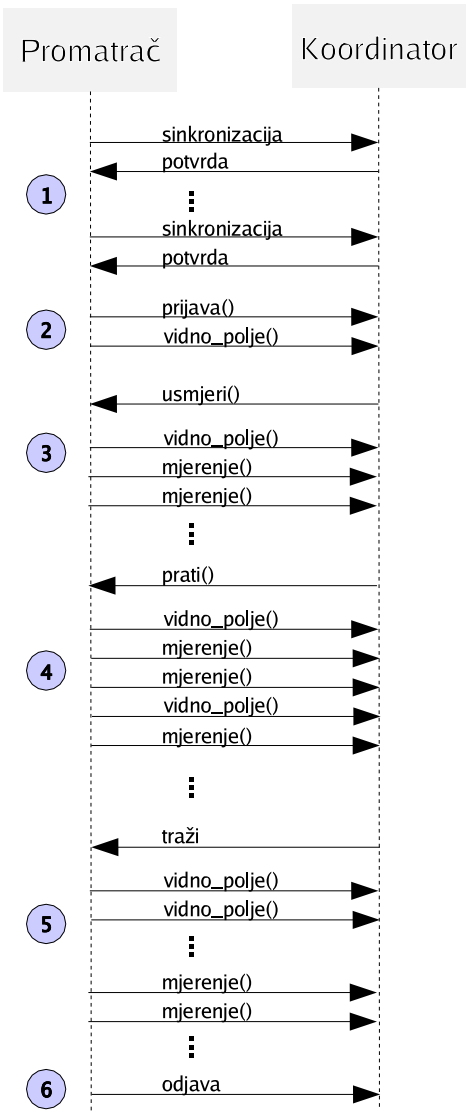
Primjer komunikacije

Na sl. 3.8 prikazan je primjer komunikacije između promatrača i koordinatora u zamišljenom eksperimentu. Zanimljivi trenutci su u lijevom dijelu slike označeni brojevima od 1 do 6, dok su odgovarajuća događanja detaljnije opisana u desnom dijelu slike.

3.2.4 Rasprava o temeljnoj arhitekturi

Prikazana temeljna hijerarhijska arhitektura za porazdijeljeno praćenje može integrirati pojedinačne slijedove položaja praćenih objekata u robustni ukupni prikaz scene. Ono što u prethodnim odjeljcima međutim nije pokazano, jest način upotrebe dobivenog prikaza za inteligentno dodjeljivanje odgovornosti pojedinim promatračima. Najinteresantnija bi svakako bila metoda koja bi bila toliko općenita da bi bila primjenljiva za sve probleme porazdijeljenog praćenja. Jedna takva strategija će biti prikazana u poglavlju 5, ali odmah valja reći da će njena primjena biti ograničena na prostore bez preklapanja strukture scene i ravnine gibanja. Područje primjene opisane metode moglo bi biti znatno veće pod uvjetom da koordinator od svakog promatrača dobije informaciju koji su dijelovi ravnine gibanja tom promatraču nevidljivi odnosno zaklonjeni [matsuyama00]. Proširenja temeljne arhitekture u smislu zadovoljenja tog uvjeta će biti razmatrana u 3.3.2.

Kao što je već navedeno, intrinsične kordinacijske strategije u mnogim slučajevima neće biti dovoljne za postizanje zadovoljavajućih rezultata praćenja. Zato bi bilo vrlo interesantno omogućiti neinvazivno uklapanje posebne aplikacijski specifične koordinacijske procedure koja bi imala apriorno znanje o svim pojedinostima konkretne primjene. Pri tome



1 **Sinkronizacija:** Promatrač određuje razliku ΔT između svog i koordinatorovog sata, mjerenjem vremena proteklog između slanja početne poruke i stvaranja poruke potvrde koordinatora.

2 **Prijava:** Na početku rada, promatrač šalje svoj položaj u k.s. svijeta i početni položaj kamere.

3 **Praćenje područja:** Koordinator zahtijeva da promatrač pokrije dio scene oko točke (x,y), pa šalje poruku `usmjeri()`. Promatrač je utvrdio da može okrenuti kameru prema toj točki te nakon izvršenja šalje koordinatoru novi smjer gledanja i počinje slati izvještaje o objektima u sceni.

4 **Praćenje objekta:** Koordinator promatraču dodjeljuje odgovornost za praćenje objekta koji se nalazi u blizini točke (x,y). Promatrač uspijeva fiksirati objekt, prati njegovo gibanje, te izvještava nakon svakog pomaka kamere te nakon svakog obrađenog slikovnog okvira.

5 **Traženje:** Nakon primitka poruke traži, promatrač sustavno mijenja smjer gledanja sve dok ne nađe neki objekt.

6 **Odjava:** Kad promatrač prekida s radom, o tome se obavještava koordinator da u daljnjoj strategiji praćenja ne računa na pomoć tog promatrača.

Slika 3.8: Primjer komunikacije između promatrača i koordinatora.

neinvazivno znači da prilikom unošenja nije potrebno ponovo prevoditi ili povezivati postojeći kôd, te da se u cilju testiranja mogu isprobati različite strategije bez potrebe za gašenjem ostatka sustava. Pokazuje se da se upravo spomenuta kao i neke druge zanimljive mogućnosti postižu višerazinskom hijerarhijom koja će se razmatrati u 3.3.3.

3.3 Proširenja temeljne arhitekture

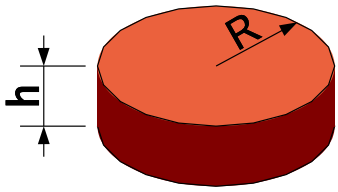
Eksperimentalni sustav koji je razvijen u sklopu ovog rada je ukazao na brojne mogućnosti poboljšanja temeljne arhitekture koja je opisana u prošlom odjeljku. Kao prvo poboljšanje, predlaže se uvođenje apriornog 3D modela praćenog objekta koji bi omogućio precizniju lokalizaciju objekata kako na razini promatrača, tako i na razini koordinatora. Nakon toga se predlaže postupak kojim bi do koordinatora mogla doći informacija koji dijelovi scene su zaklonjeni kojem pridruženom promatraču, što bi omogućilo sofisticiranije koordinacijske strategije [matsuyama00]. Predlaže se i višerazinsko proširenje temeljne arhitekture koje bi omogućilo veću skalabilnost cijelog sustava te dodavanje specifičnih koordinacijskih proce-

dura. Pokazat će se da mogućnost višerazinskog strukturiranja stvara pretpostavke za vrlo elegantnu primjenu porazdijeljenog praćenja u okviru sustava za lokalizaciju samostalnih robota na principu globalnog vida. Na samom kraju odjeljka, razmatra se mogućnost uvođenja pokretnih promatrača, koja bi mogla biti posebno zanimljiva kod vojnih primjena [collins01] i robotskog nogometa [menegatti01].

3.3.1 Modeliranje lokalizacijske pogreške

Model praćenih objekata

Kao što je navedeno i u razmatranju geometrijskih odnosa (pododjeljak 3.2.1), za modeliranje nesigurnosti određivanja 3D položaja promatrača potrebno je imati model praćenih objekata. Predlaže se jednostavni dvoparametarski model, u kojem se praćeni objekti modeliraju valjkom radijusa R i visine h, kao što je prikazano na sl. 3.9.



Slika 3.9: Model praćenog objekta.

Uz ovakav model praćenog objekta, iz sl. 3.4 se vidi da se, u slučaju idealne (potpune) segmentacije objekta, projekcija segmentirane regije na ravninu gibanja može dosta točno aproksimirati elipsom s dimenzijama osi

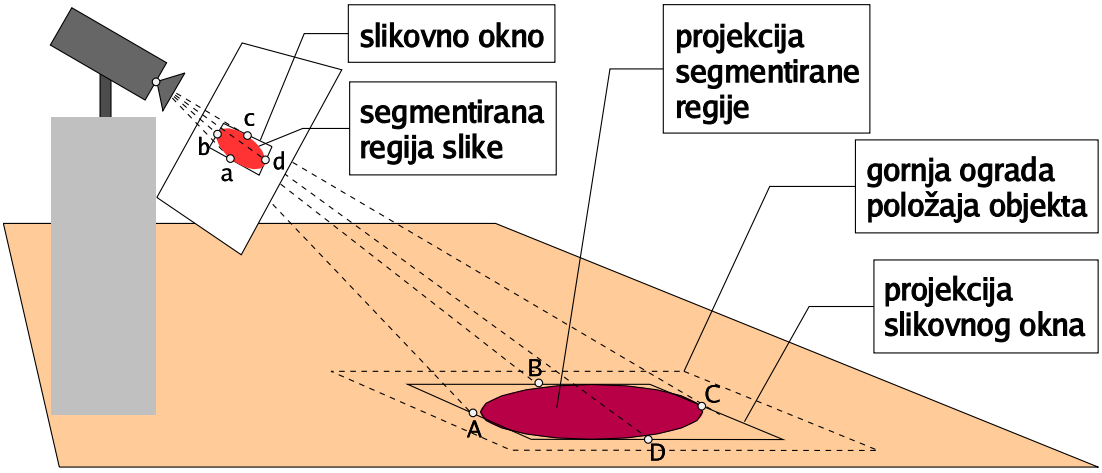
$$d_1^I = R + h \cdot \operatorname{arc\,tg}(\theta)/2) \tag{3.3}$$
$$d_2^I = R, \tag{3.4}$$
$$\tag{3.5}$$

pri čemu orijentacija glavne osi odgovara spojnici objekta i promatrača. Vrijednost θ u gornjoj jednadžbi se dobiva kao kut između pravca koji spaja robota i žarište kamere, te projekcije tog pravca na vodoravnu ravninu. Kako je tehnički lakše raditi s pravokutnicima nego s elipsama, u daljnjem tekstu će se projekcije objekata modelirati pravokutnicima.

Model nesigurnosti položaja praćenih objekata

Nažalost, segmentacija objekta je najčešće nepotpuna pri čemu obično "otpadnu" vertikalni dijelovi objekta, ali čest je slučaj da ni gornja površina ne bude detektirana u potpunosti. Točan položaj objekta u općem slučaju stoga neće biti moguće odrediti, pa ne preostaje drugo nego potražiti čim uže područje koje sigurno obuhvaća objekt. Takvo područje će se nazivati *gornjom ogradom* položaja objekta, a u nastavku paragrafa će biti opisan način za njeno određivanje. U prvom koraku, kompliciran oblik segmentirane regije je moguće aproksimirati njenim slikovnim oknom tj. minimalnim pravokutnikom koji obuhvaća cijelu regiju. Iako je projekcija slikovnog okna na ravninu gibanja najčešće trapez, zbog udaljenosti objekta od kamere njega će biti moguće prilično točno aproksimirati pravokutnikom. Projicirani pravokutnik je moguće parametrizirati središtem P_P, orijentacijom ϕ te dimenzijama glavne i sporedne osi d_1 i d_2. Ukoliko je sklopovlje promatrača kalibrirano, formalizmi opisani u poglavlju 4 omogućavaju pronalaženje projekcije elemenata slikovne ravnine na ravninu

gibanja jednostavnim množenjem vektora matricom 3×3. Stoga je parametre projiciranog pravokutnika moguće izračunati na temelju projekcija središta i simetrala stranica slikovnog okna prema slici sl. 3.10. Dobiveni parametri (d_1, d_2) će zbog podsegmentacije biti manji od



Slika 3.10: Procjena gornje ograde položaja objekta.

idealnih (d_1^I, d_2^I), pa se dimenzije gornje ograde položaja objekta mogu ocijeniti kao (d_1', d_2'):

$$d_1' = d_1 + 2(d_1^I - d_1) = 2d_1^I - d_1 \quad (3.6)$$
$$d_2' = d_2 + 2(d_2^I - d_2) = 2d_2^I - d_2 \quad (3.7)$$
$$(3.8)$$

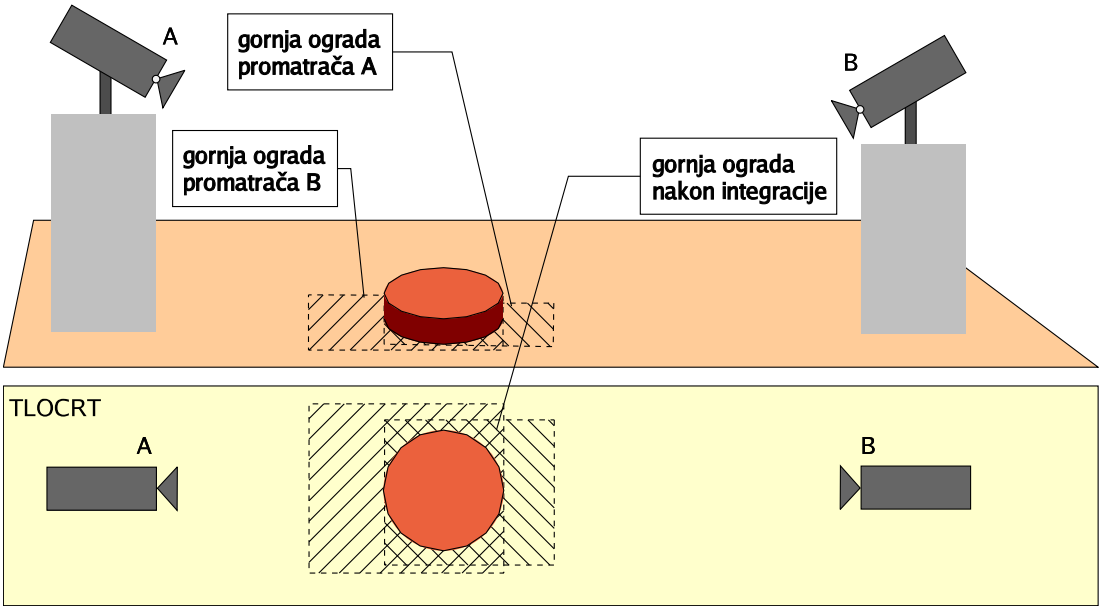
Iako se središte objekta sigurno nalazi unutar pravokutnika (P_P, ϕ, d_1', d_2'), središte pravokutnika nije ujedno i najvjerojatnija vrijednost središta objekta. Kao što je već rečeno, podsegmentacija je najčešća na vertikalnim plohama objekta jer na njih u prosjeku pada manje svjetlosti nego na gornju horizontalnu plohu. Stoga ima smisla ocijeniti najvjerojatnije središte objekta P_O kao projekciju središta segmentirane regije na ravninu koja je za visinu objekta h viša od ravnine gibanja.

Kao sažetak dosadašnjih razmatranja, može se reći da je rezultat lokalizacije objekta opisan s dva nezavisna koncepta: gornjom ogradom i najvjerojatnijim središtem. U skladu s tim, izvještaj promatrača o položaju svakog detektiranog objekta mogao bi sadržavati sljedeće podatke:

- najvjerojatnije središte, točka P_O,
- gornju ogradu položaja, pravokutnik (P_P, ϕ, d_1, d_2).

Prostorna integracija modela nesigurnosti

Ostaje još samo pitanje prostorne integracije većeg broja pojedinačnih izvještaja o položaju istog objekta. Upravo ovdje se vidi prednost koncepta gornje ograde, jer se integracija gornjih ograda provodi iznimno jednostavno. Ukoliko pretpostavimo da je korespondencija objekata u slikama pojedinih promatrača ispravno obavljena, vrijedi da sve gornje ograde zadanog objekta obuhvaćaju stvarni položaj tog objekta. Tada se postupak integracije svodi na traženje presjeka pojedinačnih gornjih ograda, kao što je ilustrirano na sl. 3.11. Sličan pristup prostornoj integraciji rezultata praćenja pojedinačnih promatrača korišten je i u [hoover99, mittal03, yang03].



Slika 3.11: Integracija gornjih ograda položaja objekta.

Vidi se da se integracijom većeg broja mjerenja, površina rezultantne gornje ograde objekta smanjuje što znači da se kvaliteta lokalizacije poboljšava. Taj podatak je do sada bio intuitivno prihvaćen kao jedna od značajnih prednosti porazdijeljenog praćenja, a sada dobiveno povećanje preciznosti možemo i kvantificirati. Nažalost, najvjerojatnije središte objekta se ne može integrirati na tako elegantan način bez primjene ad hoc distribucija u okvirima teorije vjerojatnosti ili neizrazitih skupova. Stoga se predlaže jednostavna upotreba aritmetičke sredine pojedinačnih najvjerojatnijih položaja, uz naznaku da bi se u tu svrhu moglo upotrijebiti i središte nove gornje ograde kao možda još i bolja procjena.

3.3.2 Detekcija zaklonjenih dijelova ravnine gibanja

Znanje o pokrivenosti različitih dijelova scene kamerama različitih promatrača je vrlo važno za uspješno koordiniranje njihovih odgovornosti. Kod realističnih scena gdje su prekrivanja znatna, koordinacija promatrača bez takvih informacija je bespredmetna jer je vidljivost pored prostorne bliskosti temeljni kriterij odabira promatrača koji će biti zadužen za praćenje nekog objekta. U sustavu koji pretpostavlja tri stupnja slobode kretanja praćenih objekata, pojam zaklonjenih dijelova scene je vrlo složen jer pretpostavlja listu povezanih podskupa 3D prostora [matsuyama00]. Situacija je nešto povoljnija pod pretpostavkama koje se razmatraju u ovom radu, kad je gibanje objekata ravninsko. Tada su interesantni zaklonjeni dijelovi ravnine gibanja, koji se mogu opisati listom povezanih podskupova ravnine. Neovisno o pretpostavkama vezanim za slobodu gibanja objekata, postoje dva temeljna konteksta u kojima se podatci o zaklonjenim dijelovima scene za pojedine promatrače mogu pribavljati, obrađivati i spremati: kontekst promatrača i kontekst koordinatora. Osobine tih dvaju pristupa biti će detaljnije razrađene u sljedećim paragrafima, zajedno s preporukama za odabir prikladnijeg pristupa u ovisnosti o relevantnim svojstvima područja primjene.

Detekcija zaklonjenih dijelova scene u okviru promatrača

Da bi ovakav pristup bio moguć, promatrači bi pored objekata trebali moći detektirati i postojanje zaklonjenih dijelova scene. U općenitom slučaju, to se ne može napraviti bez

dodatnih pretpostavki, kao npr, opremljenosti promatrača senzorima za stereo ili 3D vid. Međutim, u slučaju isključivo ravninskog gibanja objekata, situacija se znatno pojednostavnjuje jer postoji bijekcija između elemenata slikovne ravnine i svakog mogućeg položaja praćenog objekta. Za svaki slikovni element stoga je moguće postaviti sljedeći potpuni skup isključivih hipoteza:

1. slikovni element odgovara praćenom objektu;
2. slikovni element odgovara ravnini gibanja;
3. slikovni element odgovara dijelu scene koji zaklanja ravninu gibanja.

Naravno, čim se razmatra praćenje objekata, unaprijed se pretpostavlja da je moguće razviti metode za relativno robustno testiranje hipoteze 1. Odatle je međutim još samo jedan korak do testiranja hipoteze 2, što bi omogućilo klasifikaciju svakog slikovnog elementa po potpunom skupu hipoteza. Tehnički, testiranje bi se moglo izvesti nekom od metoda segmentacije koje bi se temeljile na boji ili teksturi ravnine gibanja. Oblikovanjem postupka za određivanje ispravne hipoteze u svakom slikovnom elementu, otvara se mogućnost detekcije zaklonjenih dijelova zanimljivog dijela scene odnosno ravnine gibanja objekata. Naime, za svaku regiju slike klasificiranu u skladu s hipotezom 3, vrijedi da je dio ravnine gibanja koji se po važećim jednadžbama također preslikava u tu regiju zapravo zaklonjen. Nastavak primjene zamisli je očigledan: promatrač tokom rada održava mapu zaklonjenih područja ravnine gibanja te povremeno šalje promjene koordinatoru kako ne bi dobivao zadatke u zaklonjenim područjima. Preliminarni rezultati u laboratorijskom okruženju potvrđuju mogućnost primjene ovakvog pristupa, a prikazani su u poglavlju 6.

Prednosti ovakvog pristupa detekciji zaklonjenih područja su brzina i preciznost dobivenih rezultata, dok su nedostatci slaba prilagodljivost različitim svojstvima ravnina gibanja, konceptualna neprilagođenost primjenama u kojima objekti nemaju ravninsko gibanje, te potreba za slanjem dodatnih podataka koordinatoru što uzrokuje povećanje složenosti komunikacijskog protokola te dodatno opterećenje komunikacijskih veza.

Detekcija zaklonjenih dijelova scene u okviru koordinatora

Ovakav pristup se temelji na bilježenju neuspješnih traženja lokaliziranih objekata, a posebno je prikladan kod sustava s velikom zalihošću promatrača. Ukupni navigacijski prostor se za svakog od promatrača dijeli na tri klase područja: *vidljivo*, *zaklonjeno* i *nepoznato*. Početno, cijeli prostor je pridijeljen klasi *nepoznato*, a dijelovi prostora se preciznije određuju tokom rada sustava. Kad god promatrač detektira prisutnost objekta, odgovarajući dio scene se za tog promatrača registrira kao vidljiv. Ukoliko sustav pretpostavlja objekte s tri stupnja slobode gibanja, u klasu vidljivih dijelova scene se dodatno uključuje i cijeli stožasti podskup prostora između kamere i objekta. Tijekom rada, koordinator često dodjeljuje odgovornosti nezaposlenim promatračima za praćenje objekta kojeg trenutno već prate i drugi promatrači. Ako novi promatrač ne detektira objekt na zadanom mjestu, odgovarajući dio scene se za tog promatrača registrira kao zaklonjen.

Prednosti opisanog pristupa su konceptualna jednostavnost, mogućnost primjene kod proizvoljnog gibanja objekata i to bez potrebe za specifičnim postupcima detekcije, te činjenica da se podatci detektiraju i spremaju tamo gdje će se i koristiti, dakle unutar koordinatora. Glavni nedostatci su sporije građenje mape vidljivosti, pogotovo kod sustava s malom zalihošću promatrača, te potreba za dodatnim procesnim resursima koordinatora.

Spremanje podataka o zaklonjenim dijelovima scene

Kod oba prethodno opisana pristupa, nakon detekcije zaklonjenih dijelova scene, nameće se potreba za umetanjem novih podataka u podatkovnu strukturu koja će biti korištena u postupku koordinacije promatrača. Glavni kriteriji koje ta podatkovna struktura mora zadovoljavati su jednostavan pristup postojećim podatcima, efikasno umetanje novodobivenih podataka, te preciznost i sažetost prikaza. U tom smislu moguća su dva osnovna pristupa, vektorski i rasterski, koji favoriziraju različite od spomenutih kriterija. Ti pristupi su detaljnije opisani u sljedećim odlomcima, pod pretpostavkom ravninskog gibanja objekata.

Vektorski pristup podrazumijeva poligonalni opis granica svake povezane komponente zaklonjenog područja scene. Vektoriziranje povezane komponente stoga se postiže aproksimiranjem njenih granica u obliku liste povezanih dužina. Dodavanje novih podataka u strukturu svodi se na kombiniranje novodobivenih podataka s postojećim primjenom metoda analitičke geometrije. Zbog velike složenosti pripadnih operacija, ovakav pristup vjerojatno ne bi bio opravdan za praćenje objekata s tri stupnja slobode kretanja.

Rasterski pristup, međutim, podrazumijeva zapis cjelokupnog zaklonjenog prostora u 2D matrici. Zapis matrice može se komprimirati rekurzivnim dijeljenjem nehomogenih kvadratnih područja u četiri manja kvadrata, što se može prikazati usmjerenim stablom u kojem svaki čvor–roditelj ima četiri čvora–djeteta (*engl.* quadtree) [matsuyama00]. Pri tome valja obratiti pažnju da se kvantizacija prostora mora napraviti sukladno orijentaciji koordinatnog sustava svijeta, jer će dobivene podatke koristiti koordinator. Glavni nedostatak ovog pristupa u odnosu na vektorski su manja konceptualna elegancija, slabiji omjer sažetosti i preciznosti, te neriješeni problem odabira najfinije rezolucije prikaza. Glavna prednost pristupa je praktičnost upotrebe, jer se omogućavaju relativno jednostavne operacije pristupa i obnavljanja podatkovne strukture.

Slično kao i kod prethodne dileme, da li je detekciju zaklonjenih područja bolje obavljati u sklopu koordinatora ili promatrača, ni za odabir formalizma prikaza zaklonjenih područja nije lako dati konačan odgovor bez daljnjeg eksperimentalnog rada. Zato se daljnja razrada mogućih opcija nameće kao jedan od najprioritetnijih nastavaka ovog rada.

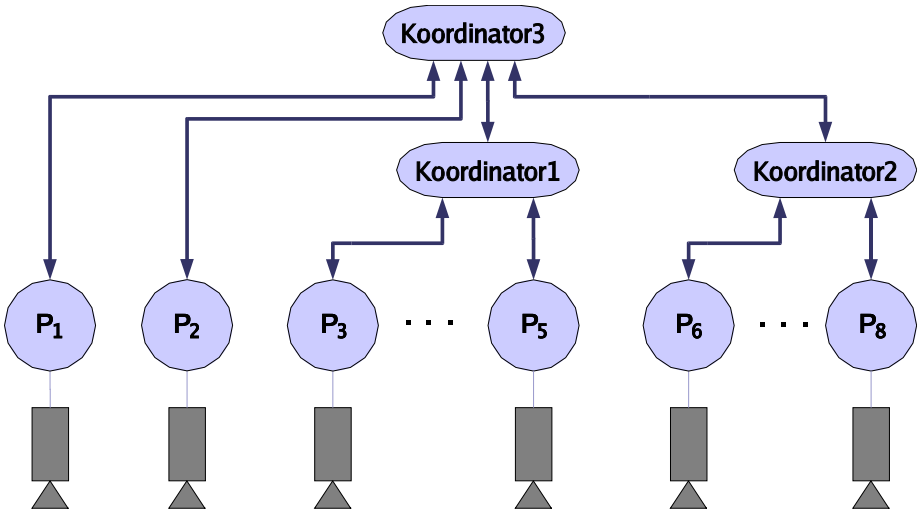
Modeliranje ograničenog radnog područja postolja promatrača

Većina postojećih upravljivih postolja nemaju neograničene mogućnosti podešavanja kutova zakreta i nagiba. Tako integrirano postolje sa sl. 3.2, ima radno područje od $\pm 25°$ po nagibu odnosno $\pm 100°$ po zakretu. Slično kao i kod detekcije zaklonjenih dijelova scene, koordinator bi te podatke morao znati kako promatračima ne bi dodjeljivao zadatke koje oni ne mogu obavljati. Pokazuje se da je informacije o radnom području moguće vrlo elegantno izraziti unutar nekog od opisanih modela zaklonjenih dijelova scene promatrača. Naime, bez obzira na prirodu gibanja objekata, moguće je dijelove scene koji se zbog ograničenog radnog područja postolja ne mogu vidjeti, jednostavno proglasiti zaklonjenima. Ovdje valja napomenuti da bi se istim postupkom u sustav mogli vrlo jednostavno integrirati i nepokretni promatrači.

Kod detekcije zaklonjenih područja u sklopu promatrača, podatci o ograničenom radnom području dolaze do koordinatora zajedno s podatcima o zaklonjenim područjima scene, pa nisu potrebna daljnja proširenja protokola. Međutim, ukoliko se zaklonjena područja detektiraju u sklopu koordinatora, potrebno je proširiti početnu prijavu promatrača na način da ona sadrži i informaciju o ograničenom radnom području upravljivog postolja.

3.3.3 Višerazinska hijerarhija

Veliki zahtjevi za procesnom moći i mrežnom propusnošću nadređenog čvora te složenost njegove izvedbe mogu postati ograničavajući čimbenik veličine jednorazinskih hijerarhijskih arhitektura za porazdijeljeno praćenje. Najelegantniji način za rješenje takvog problema je stvaranje višerazinske hijerarhije, kod koje u mreži za porazdijeljeno praćenje postoji veći broj koordinatora organiziranih u stablastu hijerarhijsku strukturu. U takvoj arhitekturi, koordinatori na nižim razinama integriraju podatke dobivene od sebi podređenih agenata te prosljeđuju integrirano stanje scene prema koordinatoru na višoj razini. Kako se ne bi izgubila mogućnost donošenja odluka na globalnoj razini, podređeni koordinator mora primati odgovornosti od nadređenog koordinatora, te ih naknadno raspodijeliti sebi podređenim agentima. Idejna shema opisane arhitekture prikazana je na sl. 3.12.



Slika 3.12: Primjer višerazinske hijerarhije za porazdijeljeno praćenje. Krugovi predstavljaju agente promatrače kojima su pridružene kamere, dok zaobljeni pravokutnici predstavljaju koordinatore.

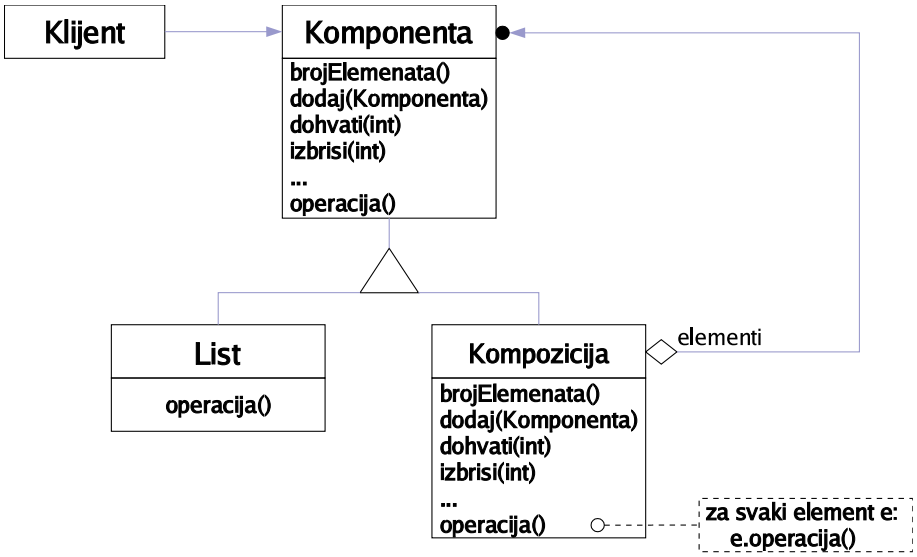
Dodatna prednost višerazinske arhitekture jest mogućnost dodavanja specijaliziranog koordinatora na vrhu hijerarhije. Specijalizirani koordinator bi u sustavu sa sl. 3.12 mogao biti postavljen odmah iznad koordinatora 3, a uključivao bi apriorno znanje o konkretnom zadatku praćenja. Time bi se postiglo eksplicitno odvajanje intrinsične strategije porazdijeljenog praćenja i specifičnih zahtjeva razmatrane aplikacije, slično modelu svjesnog i nesvjesnog ponašanja kod ljudi. U skladu s tom analogijom, intrinsična strategija koordinacije se primjenjuje po postavci, dok specifična strategija definira korekcije u određenim situacijama kada temeljno ponašanje nije prikladno.

Obrazac kompozicije

Elementi hijerarhijske arhitekture na sl. 3.12 mogu se podijeliti na agente promatrače koji izravno doprinose funkcionalnosti, te agente koordinatore koji predstavljaju čitave grupe agenata promatrača. Podređeni agenti svakog koordinatora mogu biti ili promatrači ili, rekurzivno, čitave podhijerarhije čiji arhitektonski obrazac odgovara obrascu glavne arhitekture. Potreba za takvim višerazinskim hijerarhijama se u programskom inženjerstvu često javlja, pa je za njihovu izvedbu predložen poseban strukturni oblikovni obrazac koji je originalno nazvan *composite* [gamma95]. U skladu s tim obrascem, višerazinsku hijerarhiju moguće je opisati podatkovnom strukturom u obliku jednorazinskog stabla čiji elementi mogu biti

ili konkretni čvorovi odnosno listovi stabla, ili još jedna takva struktura odnosno podstablo. Zbog svojstva da se kao elementi osnovnog stablastog strukturiranog tipa javljaju nove instance tog istog tipa, navedeni obrazac se može nazvati *kompozicijom*.

Dijagram odnosa tipova koji sudjeluju u obrascu je prikazan na sl. 3.13. Iz dijagrama se vidi da se cijeloj strukturi pristupa preko sučelja `komponenta` kojeg implementiraju i listovi i podstabla strukture (simbol: trokut okrenut prema gore). Podstabla su opisana tipom `Kompozicija`, a sadrže (simbol: romb) veći broj (simbol: crni krug) komponenata koje mogu biti ili listovi ili nova stabla. Konačno, dijagram prikazuje (simbol: kružnica) kako podstabla obavljaju operacije pozivanjem odgovarajućih operacija nad svim svojim elementima.



Slika 3.13: Strukturni dijagram odnosa tipova u obrascu kompozicije.

Obrazac kompozicije nudi mogućnost povezivanja elementarnih komponenti i njihovih grupa u proizvoljnu hijerarhijsku strukturu. Glavna prednost pristupa je mogućnost transparentnog rada s elementarnim objektima i njihovim grupama preko zajedničkog sučelja. Međutim, da bi se obrazac mogao primijeniti, potrebno je opisati operacije složenih objekata u terminima njihovih sastavnih dijelova. Rješavanje tog problema u potpunosti ovisi o konkretnoj primjeni te će najčešće uvjetovati primjenljivost obrasca.

Ostvarenje višerazinske hijerarhije obrascem kompozicije

Iako je obrazac kompozicije prvotno zamišljen u okviru objektno orijentiranog programiranja, pokazuje se da se analogna razmatranja mogu primijeniti i u okviru agentske paradigme. Naime, odnosi prikazani na sl. 3.13 vrijede i ako pretpostavimo da su sudionici obrasca *aktivni* objekti koji imaju nezavisan tok izvođenja odnosno vlastitu dretvu, čije sučelje se temelji na mrežnom protokolu. Takvi neformalni opisi agentskih arhitektura dobivaju na važnosti tim više jer pojam sučelja u agentskoj paradigmi nije tako dobro definiran kao kod objekata, pa jezici opće namjene u kojima bi se zahtjevi za sučeljem agenta mogli formalno izraziti ne postoje.

Kao što je rečeno prije, ključan korak u izvedbi kompozicije je izraziti operacije nadređenog čvora u terminima podređenih čvorova. U našem konkretnom slučaju, to znači da podređeni koordinator u komunikaciji s nadređenim koordinatorom mora moći igrati ulogu

vrlo blisku ulozi promatrača u komunikacijskom protokolu temeljne arhitekture. To se svodi na sljedeće dvije stvari:

- podređeni koordinator mora slati ukupni prikaz scene nadređenom agentu kao što bi ga slao i promatrač;
- podređeni koordinator mora pokušati poštovati odgovornosti koje su mu dodijeljene od strane nadređenog koordinatora.

Sljedeći paragrafi opisuju skicu implementacije akcija podređenog koordinatora, za svaku poruku komunikacijskog protokola.

Poruke podređenog koordinatora

`sinkronizacija():`
: Kao i promatrač, i podređeni koordinator periodički sinkronizira svoj sat prema satu nadređenog koordinatora. Vremenski podatci kod svih ostalih poruka su izraženi u skladu sa satom nadređenog koordinatora.

`prijava(ime, x,y)`
: Podređeni koordinator se prijavljuje nadređenom i šalje svoje ime, težište položaja pridruženih agenata promatrača te njihov broj. Podređeni koordinator je kalibriran pa sva mjerenja izražava u koordinatnom sustavu nadređenog koordinatora.

`vidno_polje(lista_područja)`
: Podređeni koordinator izvještava koordinatora o dijelovima ravnine kretanja koji se nalaze unutar njegovog vidnog polja. Poslano područje odgovara uniji vidnih polja podređenih agenata.

`zaklonjena_područja(lista_područja)`
: Podređeni koordinator izvještava koordinatora o zaklonjenim dijelovima ravnine kretanja. Poslano područje odgovara presjeku zaklonjenih površina podređenih agenata.

`mjerenje(lista_objekata)`
: Nakon postavljanja novih percipiranih objekata na oglasnu ploču, podređeni koordinator šalje nadređenom njihove položaje.

`odjava()`
: Neobavezna poruka o prekidu rada podređenog koordinatora.

Poruke nadređenog koordinatora

`potvrda():`
: Neposredni sinkroni odgovor na sinkronizacijsku poruku podređenog agenta.

`usmjeri(x,y):`
: Dodjela odgovornosti za praćenje područja ravnine gibanja oko točke (x,y). Ukoliko ima slobodnih podređenih agenata, podređeni koordinator dodjeljuje odgovornost agentu koji je najbliži danoj točci uz eventualnu preraspodjelu odgovornosti ostalih agenata.

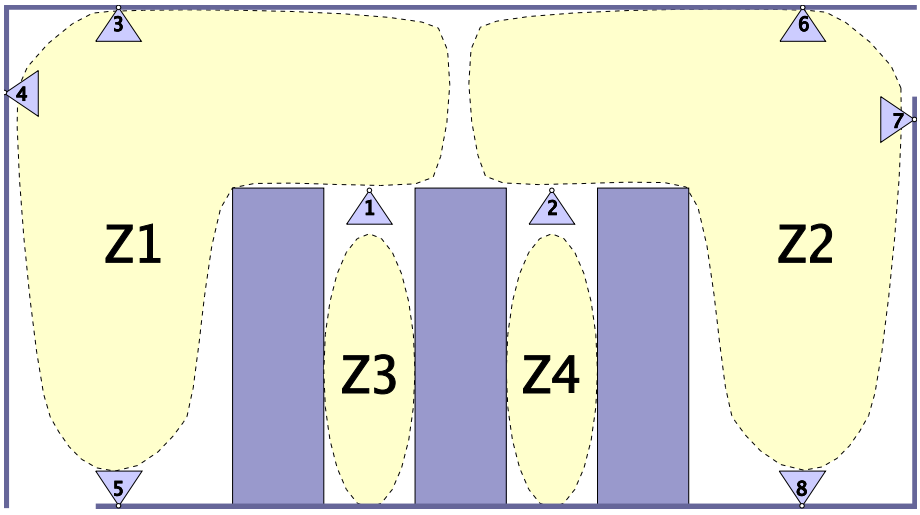
`prati(x,y)`:
: Dodjela odgovornosti za praćenje objekta koji se nalazi u blizini točke (x,y). Ukoliko ima slobodnih podređenih agenata, podređeni koordinator dodjeljuje odgovornost agentu koji je najbliži danoj točci uz eventualnu preraspodjelu odgovornosti ostalih agenata.

`traži()`:
: Dodjela odgovornosti za pretraživanje prostora. Podređeni agent šalje poruku `traži()` slobodnom podređenom agentu, ukoliko takav postoji te ako nijedan drugi podređeni agent nije u stanju `traženje`.

Iz opisanog se može zaključiti da pri slanju podataka o praćenju prema višim razinama hijerarhije ne treba očekivati nikakve probleme: podređeni koordinator integrira podatke dobivene od sebi podređenih agenata i jednostavno ih prosljeđuje nadređenom koordinatoru. Kod upravljačkog toka, koji se spušta niz hijerarhiju, situacija je nešto složenija. Naime, čini se da je tu komunikaciju vrlo teško napraviti transparentnom s obzirom na resurse podređenih agenata: nije smisleno iste odgovornosti dodijeliti jednom promatraču ili grupi od npr. pet promatrača. Zato prilikom prijave, svaki promatrač dojavljuje broj promatrača s kojima raspolaže. Očekuje se da taj broj tokom rada sustava neće biti konstantan (zbog kvarova ili dodavanja novih promatrača u sustav), pa će nadređeni promatrač morati biti spreman obraditi višestruke prijave za korekciju brojnog stanja.

Porazdijeljeno praćenje u sustavu s višerazinskom hijerarijom

Očito, strategije koordinacije cijelih *hijerarhija* promatrača pripadaju dosta zahtjevnom području koje nadilazi opseg ovog rada. Međutim, postoji izravna primjena višerazinske hijerarhije kod koje se ta koordinacija svodi na minimum, a odnosi se na hijerarhiju prilagođenu povezanosti nadgledane razvedene scene. Neka je, na primjer, zadan tlocrt nadgledanog prostora prema sl. 3.14, u kojem su raspoređeni promatrači označeni brojevima od 1 do 8. Skup promatrača se u tom slučaju može particionirati prema više–manje odvojenim "interes-



Slika 3.14: Tlocrt razvedene scene s particioniranim promatračima.

nim zonama" Z1 do Z4 koje zajednički pokrivaju promatrači iz pojedinih skupova. Nadalje, promatrače iz svake interesne zone moguće je organizirati u podhijerarhiju sa zasebnim pomoćnim koordinatorom. Konačno, pomoćne koordinatore i promatrače koji jedini pokrivaju svoju interesnu zonu moguće je povezati s glavnim koordinatorom skupine, prema sl. 3.12.

Potreba za koordinacijom promatrača koji pokrivaju različite zone se javlja isključivo u slučaju kada objekt prelazi granicu interesnih zona. Nadređeni koordinator takvu situaciju može detektirati prema bliskosti položaja objekta prijavljenim položajima pojedinih podređenih koordinatora, te prema odgovarajućim podatcima o zaklonjenim dijelovima scene. Nakon detekcije takve situacije, koordinator može pomoći na način da odgovornost za praćenje objekta prenese s grupe iz čije zone objekt izlazi, na grupu u čiju zonu objekt ulazi. Naravno tu može nastati problem u slučaju da su svi promatrači u odredišnoj grupi zauzeti, ali bez daljnjih pretpostavki u smislu apriornog prioriteta pojedinih objekata, ta situacija može biti "riješena" samo podjelom promatrača na pratioce i motritelje kao što je opisano u 3.2.2. Glavna prednost prikazane arhitekture u odnosu na jednorazinsku sa sl. 3.1, je u tome što se zahtijevaju manji mrežni i procesni resursi glavnog koordinatora. pri čemu su ostala svojstva sustava uglavnom nepromijenjena.

3.3.4 Navigacija robota porazdijeljenim sustavom globalnog vida

U ovom odlomku, opisat će se mogućnost primjene porazdijeljene arhitekture za praćenje objekata koja je opisana u prethodnom tekstu, u infrastrukturi globalnog vida za podršku autonomne navigacije jednostavnih robota. Ovdje valja napomenuti da funkcija tih robota nije ni na koji način vezana uz sustav za porazdijeljeno praćenje, za razliku od pokretnih promatrača o kojima će biti riječi u 3.3.5

Lokalizacija globalnim vidom se obično primjenjuje kod robota s vrlo ograničenim procesnim i senzorskim mogućnostima. Senzorska oprema najčešće uključuje samo odometriju i ultrazvučne daljinomjere, dok resursi pripadnog mikroupravljača omogućuju samo najprimitivnije navigacijske procedure uglavnom reaktivnog karaktera. Inteligentno ponašanje takvih robota je pod nadzorom programa koji se izvodi na odvojenom računalu koji s mikroupravljačem komunicira preko radio veze. U skladu s tim, može se reći da se funkcionalnost sustava za samostalnu navigaciju globalnim vidom sastoji od dvije temeljne operacije: percepcije okoline i upravljanja robotima. Kod originalnog sustava globalnog vida [kay93], obje temeljne operacije su bile integrirane u monolitnom programu, prema sl. 2.11. Unatoč monolitnoj organizaciji, veza između modula koji su odgovorni za obavljanje tih operacija u tom sustavu je bila jednosmjerne prirode: modul za upravljanje je donosio odluke na temelju opaženog stanja okoline, bez utjecaja na postupke opažanja.

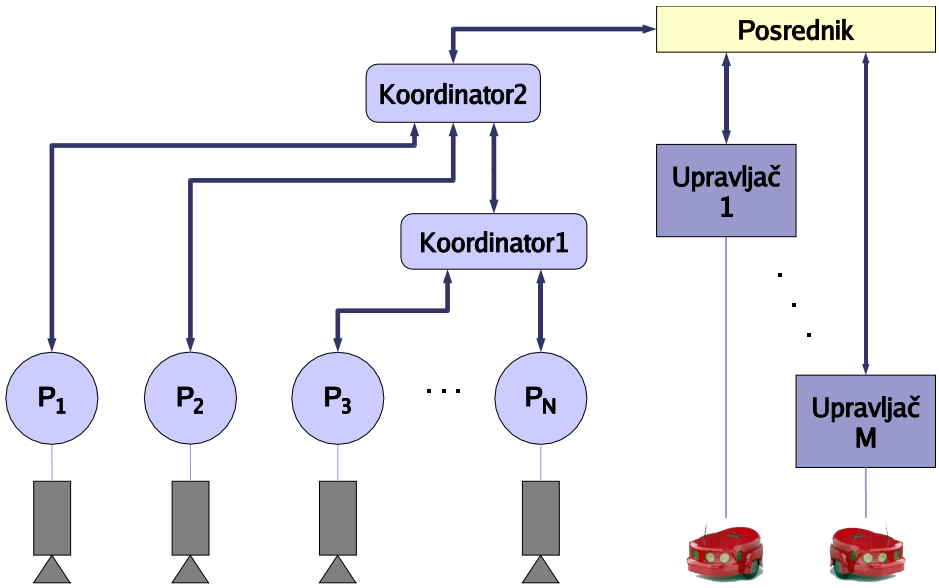
Monolitna rešenja obično imaju slabije mogućnosti prilagodbe većem korištenju resursa i novim uvjetima rada, pa su, kod novijeg sustava globalnog vida [hoover00], temeljni moduli izvedeni u zasebnim programskim jedinicama. U tom sustavu, modul za percepciju je izveden kao porazdijeljeni sustav računarskog vida koji nakon integracije mapa zauzeća dobivenih oduzimanjem pozadine za svaku od pridruženih kamera, na izlazu daje ukupni prikaz scene u obliku standardnog analognog video signala. Taj video signal se ponovo digitalizira u sklopu modula za upravljanje robotima koji donosi odluke o gibanju svih robota na temelju rezultata analize digitaliziranog signala. Problem identifikacije pojedinih robota u prikazu scene se rješava u okviru upravljača, kao što je opisano u 2.4.

Iako se rješenjem opisanim u [hoover00] postiže fizička odvojenost percepcije scene i upravljanja robotima sa svim svojim prednostima, ono ipak ne zadovoljava sljedeće bitne kriterije:

1. ne omogućava se porazdijeljeni koncept upravljanja robotima: stvarna samostalna navigacija se postiže tek kad se različitim robotima može upravljati *nezavisnim* programima;

2. nije predviđena povratna veza od kognitivnog modula za upravljanje robota prema opažajnom modulu, u skladu s principima aktivnog vida [bajcsy88].

Navedena svojstva bi se u sklopu predloženog sustava za porazdijeljeno praćenje mogla postići specijaliziranim agentom *posrednikom*. U skladu s prethodno navedenim zahtjevima, posrednik bi imao ulogu specijaliziranog koordinatora na vrhu hijerarhije sustava za porazdijeljeno praćenje, dok bi u isto vrijeme nezavisnim upravljačima pojedinih robota omogućavao uvid u stanje scene i podnošenje zahtjeva za povećanim prioritetom praćenja nekog promatrača ili dijela scene. Arhitektura opisanog pristupa navigaciji globalnim vidom je ilustrirana na sl. 3.15. Tako bi na primjer robot koji je zbog nedostatka resursa ostao izvan praćenog dijela scene, mogao neko vrijeme voziti "na slijepo", pazeći samo da izbjegne sudar očitavanjem ultrazvučnih senzora. Vožnja na slijepo bi se prekinula u slučaju nepredviđenih bliskih susreta ili kad bi nesigurnost položaja ocijenjenog odometrijom narasla iznad toleriranih vrijednosti. U tom slučaju, robot bi morao stati i zahtijevati od posrednika pokrivanje svog trenutnog procijenjenog položaja sa svrhom precizne lokalizacije.



Slika 3.15: Predložena arhitektura za samostalnu navigaciju globalnim vidom.

Iz izloženog se vidi da bi posrednik trebao igrati ulogu koordinatora (nadređenog) prema sustavu za porazdijeljeno praćenje, dok bi njegova uloga prema upravljačima robota bila podređenog karaktera i vrlo slična ulozi promatrača. Ovdje će se stvar unekoliko zakomplicirati jer do sada nismo imali situaciju da je jedan podređeni čvor odgovoran više nego jednom nadređenom čvoru. Posrednik će stoga imati osjetljiv zadatak usklađivanja zahtjeva većeg broja sebičnih agenata na opću dobrobit. Skica funkcionalnost posrednika bi se sastojala od sljedećih usporednih operacija:

- **Osluškivanje prijava za rad:**
  svaki nezavisni upravljač robota se prijavljuje na predodređenom mrežnom ulazu računala na kojem se izvodi agent posrednik;

- **Izvještaj o stanju scene:**
  posrednik periodički šalje rezultate praćenja svakom prijavljenom upravljaču, upotrebom poruka koje u sustavu za praćenje šalju promatrači: `vidno_polje()`, `mjerenje()` i `zaklonjena_područja()`.

- **Primanje zahtjeva za praćenje:**
  posrednik od prijavljenih upravljača prima poruke koje u sustavu za praćenje šalju koordinatori: `usmjeri()`, `prati()` i `traži()`.

- **Rješavanje konflikata:**
  posrednik mora uskladiti zahtjeve pojedinih upravljača s intrinsičnim zahtjevima porazdijeljenog praćenja, i po potrebi rješavati konflikte po principu kružnog prvenstva.

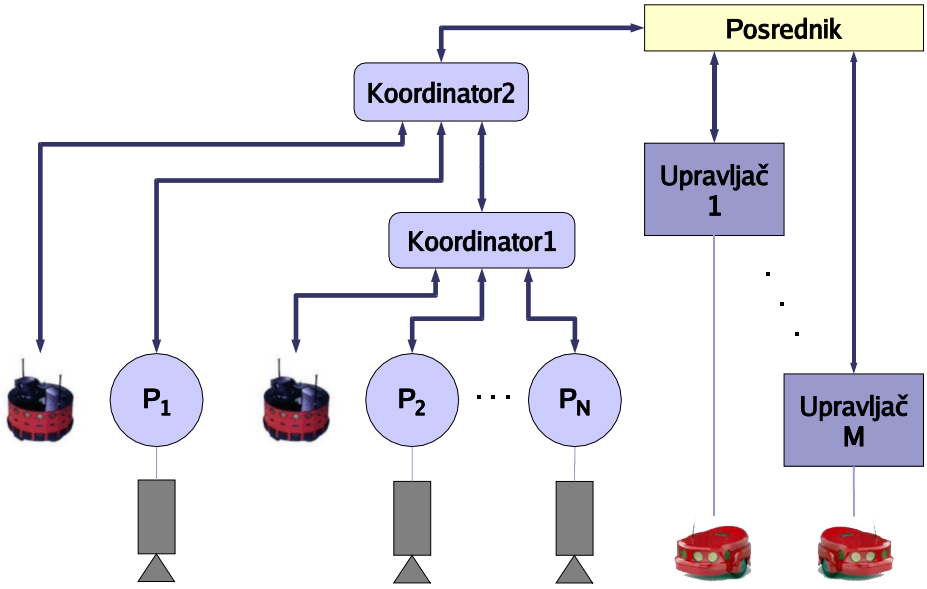
3.3.5 Pokretni promatrači

Kao posljednji aspekt porazdijeljenog praćenja, razmotrit će se mogućnosti korištenja agenata promatrača koji se mogu autonomno gibati kroz scenu. Općenito, može se očekivati da će preciznost mjerenja pokretnog promatrača biti znatno manja nego u slučaju nepokretnog promatrača opremljenog istom kamerom, i to zbog (i) veće nesigurnosti vanjskih parametara kamere, što je uvjetovano promjenljivim položajem promatrača, i (ii) relativno male najveće visine na kojoj se kamera može pričvrstiti na pokretnu platformu.

Slika 3.16: Nepovoljan položaj kamere kod pokretnog promatrača.

Prednosti upotrebe pokretnih promatrača se stoga očituju prvenstveno u većoj prilagodljivosti sustava za praćenje razvedenom okolišu u kojem se praćeni objekti kreću. Kako god domišljat raspored aktivnih promatrača bio, gotovo uvijek postoje slijepi zakutci okoliša koji se ne mogu pokriti niti jednim od njih. Ukoliko se tokom rada pokaže da je pokrivanje jednog od takvih zakutaka bitno (npr, ukoliko se jedan od praćenih robota ne može izvući iz slijepe ulice), jedan od koordinatora može poslati pokretnog promatrača da dojavi što se tamo zapravo zbiva. Način uklapanja pokretnih promatrača u predloženu porazdijeljenu infrastrukturu globalnog vida ilustriran je na sl. 3.17. Pored namjenskih pokretnih promatrača čiji jedini cilj je izvještavanje o stanju scene, sustav može koristiti usluge pokretnih platformi koje imaju i druge ciljeve, odnosno, nije ih moguće pomicati na proizvoljne lokacije okoliša. Tako se opisuje integriranje podataka dobivenih kamerama na tlu s podatcima dobivenim iz kamere postavljene na borbenom zrakoplovu koji mora brzo preletjeti zanimljivo područje

zbog opasnosti od neprijateljske protuzračne obrane [collins01]. Pored toga, dobar primjer je i praćenje lopte tijekom utakmice robotskog nogometa gdje je određivanje položaja lopte samo jedan od ciljeva porazdijeljenog robotskog sustava [menegatti01].



Slika 3.17: Uklapanje pokretnih promatrača u porazdijeljenu infrastrukturu globalnog vida.

Za sve opisane scenarije, ključni problem je potreba za periodičkim određivanjem točnog položaja za svakog pokretnog promatrača u sustavu. Pored toga, zanimljive su i ostale modifikacije višerazinske hijerarhije koja je opisana u prethodnim odjeljcima, potrebne za uklapanje namjenskih pokretnih promatrača. U nastavku će biti razmatrane najznačajnije od tih promjena, a one su vezane za proširenja osnovnog protokola, korištenje pokretnih promatrača u okviru koordinacijske strategije te mogućnosti promjene izravno nadređenog agenta koordinatora.

Problem lokalizacije pokretnog promatrača

Moguće pristupe lokalizaciji pokretnog promatrača u kontekstu sustava za porazdijeljeno praćenje moguće je svrstati u dvije velike grupe: one koji koriste infrastrukturu sustava i pristupe koji su nezavisni od ostalih promatrača u sustavu. Iako druga opcija uključuje veći spektar mogućnosti, od detekcije referentnih objekata pa sve do modernih uređaja za satelitsku navigaciju, najzanimljivije bi ipak bilo omogućiti određivanje položaja pokretnog robota bez upotrebe dodatnih resursa. To implicira povremenu primjenu apsolutne lokalizacije korištenjem rezultata praćenja nepokretnih promatrača u sustavu. U tom slučaju se javlja zanimljiva situacija da pokretni promatrač u sustavu obavlja dvostruku ulogu. Tijekom većeg dijela rada, on je ravnopravni sudionik u porazdijeljenom praćenju, međutim povremeno, tijekom periodičkih lokalizacija, pokretni promatrač je, što se ostatka sustava tiče, najobičniji pokretni objekt čiji položaj je potrebno što preciznije odrediti.

S obzirom na to da se zbog širine primjene arhitekture ne pretpostavlja mogućnost identifikacije praćenih objekata, pokretni promatrač može odrediti svoj položaj jedino na temelju uvida u ukupni prikaz scene. Tok informacija između takvog agenta i koordinatora je suprotan od uobičajenog jer podatci o sceni trebaju putovati od koordinatora prema (budućem) promatraču. Umetanje novog oblika komunikacije prilično bi zakompliciralo postojeći protokol pa je vjerojatno bolje rješenje razviti zasebni lokalizacijski protokol. Odgovarajuća lokalizacijska procedura stoga bi se odvijala u sljedećim koracima:

- **O**djava praćenja:
  ukoliko je agent već prijavljen kao promatrač nekog koordinatora treba se odjaviti s te funkcije, jer tijekom lokalizacije ne može sudjelovati u porazdijeljenom praćenju.

- **P**rijava lokalizacije:
  pokretni agent se prijavljuje kao sudionik u lokalizacijskom protokolu koordinatoru na vrhu hijerarhije koji jedini ima ukupni prikaz stanja scene. Koordinator odgovara slanjem prikaza scene u pravilnim vremenskim razmacima;

- **a**naliza:
  pokretni agent analizira razvoj događaja u sceni na temelju kojeg nakon stanovitog vremena nedvosmisleno određuje svoj položaj;

- **o**djava lokalizacije:
  pokretni agent odjavljuje lokalizacijski protokol, te se neposredno nakon toga prijavljuje kao pokretni promatrač koordinatoru kojeg mu dodjeljuje koordinator na vrhu hijerarhije.

Naravno, posao je kvalitativno najteži na samom početku rada sustava, kada položaj agenta nije ni približno poznat. Kritična faza je analiza razvoja događaja jer bi tada agent trebao odrediti koji od objekata u ukupnom prikazu scene ima dinamiku gibanja koja se poklapa s njegovom. Naravno, postoji mogućnost da je početni položaj agenta unutar dijela scene koji ne pokriva niti jedan promatrač. U tom slučaju, agent treba nasumičnim gibanjem pokušati ući u kolektivno vidno polje sustava, što bi se trebalo odraziti ulaskom novog objekta u ukupni prikaz scene. Kod svih narednih lokalizacija, posao je znatno lakši jer je tada približni položaj agenta poznat. Agent tada "zna" u kojem se smjeru mora gibati kako bi postao vidljiv ostalim agentima sustava, te u kojem dijelu ukupnog prikaza scene može očekivati svoje pojavljivanje.

Ostali dodatci temeljnoj arhitekturi

Osnovni protokol za komunikaciju promatrača s koordinatorom će biti dovoljan za zadovoljenje većine komunikacijskih potreba i pokretnih promatrača. Ipak, kako bi se iskoristile njihove specifične mogućnosti, korisno bi bilo unijeti sljedeće dodatke osnovnom protokolu:

- Tijekom prijave, svi promatrači dojavljuju svom koordinatoru da li imaju sposobnost kretanja ili ne. Pretpostavlja se da su u ovom trenutku pokretni promatrači već odredili svoj položaj lokalizacijskim protokolom, te da im je tom prilikom dodijeljen koordinator.

- Nakon svakog pomaka, pokretni promatrači dojavljuju koordinatoru svoj novi položaj koji se na temelju poznatog starog položaja određuje odometrijskim senzorima. Ovdje valja primijetiti da pomaci, na žalost, potpuno obezvrijeđuju do tada izgrađenu mapu zaklonjenih područja.

- Analogno mapi zaklonjenih područja, moguće je na temelju očitanja ultrazvučnih senzora pokretnog agenta graditi mapu *nedostupnih* područja navigacijskog prostora.

- Potrebna je nova koordinacijska poruka za zadavanje novog položaja pokretnom promatraču. Novi položaj se može odrediti na temelju učestalosti posjete praćenih objekata pojedinim dijelovima scene, uz razmatranje mape zaklonjenog prostora nepokretnih promatrača.

- Konačno, ponekad će biti potrebno dodijeliti pokretnog promatrača dijelu scene koji je pod izravnim nadzorom drugog koordinatora. Takvi transferi bi mogli biti inicirani od strane agenta koji je nadređen i koordinatoru kojem je pokretni agent trenutno odgovoran, i koordinatoru kod kojeg se javlja potreba za pokretnim promatračem.

Poglavlje 4

Umjeravanje elemenata aktivnog vida

Određivanje kvantitativnih odnosa u sceni na temelju podataka dobivenih aktivnim računarskim vidom temelji se na dva ključna preduvjeta. Prvi preduvjet u tom kontekstu je postavljanje prikladnog modela stvaranja slike u senzorskom elementu kamere, te modela gibanja senzora u ovisnosti o parametrima stupnjeva slobode pokretnog postolja. Postavljeni modeli će omogućiti razumijevanje odnosa između elemenata slikovne ravnine i stvarnih objekata 3D scene, te odrediti opseg problema koji su teoretski rješivi. Neki od parametara postavljenog modela nužno će biti proizvoljni zbog inherentnog svojstva stvaranja slike da se *mnoge* točke prostora preslikavaju u istu točku slikovne ravnine. Međutim, većina parametara modela će biti svojstvena kameri, pokretnom postolju ili položaju postolja u referentnom koordinatnom sustavu scene. Postavljenu teoriju nije moguće praktično upotrijebiti bez poznavanja točnih iznosa parametara sklopovlja pa je razvoj postupaka za njihovo određivanje drugi ključni preduvjet za obavljanje preciznih mjerenja pokretnim kamerama. Zadovoljenje opisanih preduvjeta predmet je živog interesa u znanstvenoj literaturi, što je rezultiralo solidnim teoretskim osnovama i domišljatim praktičnim postupcima umjeravanja. Veći dio izlaganja u ovom poglavlju stoga je pregled i eksperimentalna evaluacija postojećih ostvarenja, dok se tek u završnom dijelu razmatraju originalne mogućnosti pronalaženja ravninskog mjernog uzorka i umjeravanja pokretnih postolja.

4.1 Matematički alati

U ovom odjeljku biti će razmatrane temeljne teoretske postavke potrebne za modeliranje procesa stvaranja slike. Pokazuje se da se iznimno elegantan opis tog procesa dobiva aparatom projekcijske geometrije, jer on omogućava *linearan* opis stvaranja slike i transformacija među koordinatnim sustavima Zbog linearnosti, kombiniranje transformacija se svodi na elementarne matrične operacije pa će u različitim manipulacijama vrlo bitnu ulogu igrati neke metode linearne algebre. Važni matematički alati će u ovom odjeljku biti izloženi nevezano od ostatka teksta, dok će se njihova primjena objasniti u sljedećim odjeljcima.

4.1.1 Projekcijska geometrija

Projekcijska geometrija je grana geometrije koja proučava svojstva i invarijante projekcijskih preslikavanja. S obzirom da se radi o dobro razrađenom području, ovdje će se radi kompletnosti navesti samo temeljni pojmovi i rezultati koji će biti korišteni u nastavku teksta, dok se detaljniji prikaz može naći u [faugeras93, mohr96, segvic00, weisstein:www].

Projekcijsko preslikavanje ili kraće projekcija je svako preslikavanje između dva projekcijska prostora koje čuva kolinearnost točaka, odnosno u kojem se pravci preslikavaju u

pravce. Točke n–dimenzionalnog projekcijskog prostora $\mathbf{x} \in \mathbb{P}^n$ mogu se predstaviti (n+1)–dimenzionalnim vektorima različitim od nule, tako da vrijedi:

$$\mathbf{x} \in \mathbb{P}^n \Rightarrow \mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_{n+1} \end{bmatrix}^T, \ \mathbf{x} \neq \mathbf{0}. \tag{4.1}$$

Vektori koji predstavljaju elemente projekcijskog prostora su definirani do na konstantu odnosno imaju homogene koordinate. Ista točka se stoga obično može predstaviti s beskonačno mnogo različitih vektora, pa se zato uvodi relacija "sličnosti" koja povezuje sve različite zapise istog elementa projekcijskog prostora:

$$\mathbf{x} \sim \mathbf{y} \Longleftrightarrow \mathbf{x} = \lambda \cdot \mathbf{y}, \ \forall \lambda \in \mathbb{R} \backslash \{0\} \tag{4.2}$$

Pokazuje se da je svaka projekcija linearna transformacija $\mathbb{P}^m \rightarrow \mathbb{P}^n$ pa se može opisati matricom $\mathbf{P}_{m+1 \times n+1}$. Lako se vidi da su projekcijske matrice također definirane do na konstantu pa vrijedi $\mathbf{P} \sim \lambda \mathbf{P}$, uz $\lambda \neq 0$. Za razliku od uobičajenih transformacija, projekcije pored kolinearnosti čuvaju samo varijante križnog omjera (*engl.* cross ratio) četiri točke [mohr96]. Kutevi projiciranih likova stoga ne moraju biti jednaki originalima, pa dolazi do specifičnih projekcijskih "izobličenja" analizom kojih je moguće izvući neočekivane i dalekosežne zaključke [segvic00].

Projekcijski prostori su uglavnom korisni samo kao međukorak u elegantnijem postupku modeliranja projekcija, dok se položaji fizičkih objekata obično izražavaju u klasičnim Euklidskim prostorima $\mathbb{R}^2$ ili $\mathbb{R}^3$. Stoga je tipičan slijed operacija u modeliranju projekcije (i) predstavljanje ulaznih podataka iz izvorišnog Euklidskog prostora (tipično scena, $\mathbb{R}^3$) u odgovarajućem projekcijskom prostoru, (ii) primjena projekcija obavljanjem matričnih operacija, i (iii) vraćanje rezultata iz projekcijskog prostora u odredišni Euklidski prostor (tipično slika, $\mathbb{R}^2$). Kanonsko izomorfno preslikavanje $\mathbb{R}^n \rightarrow \mathbb{P}^n$ postoji u svakoj točki $\mathbf{x} \in \mathbb{R}^n$:

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \mapsto \begin{bmatrix} x_1 & x_2 & \dots & x_n & 1 \end{bmatrix}^T \tag{4.3}$$

Obratno kanonsko preslikavanje postoji za sve $\mathbf{y} \in \mathbb{P}^n : y_{n+1} \neq 0$:

$$\begin{bmatrix} y_1 & y_2 & \dots & y_{n+1} \end{bmatrix}^T \mapsto \begin{bmatrix} \frac{y_1}{y_{n+1}}, \frac{y_2}{y_{n+1}}, \dots, \frac{y_n}{y_{n+1}} \end{bmatrix}^T \tag{4.4}$$

Točke oblika $\mathbf{y} \in \mathbb{P}^n : y_{n+1} = 0$ se nazivaju točke u beskonačnosti i nemaju prikaz u $\mathbb{R}^n$, a skup svih takvih točaka čini tzv. hiperravninu u beskonačnosti. Stoga se može reći da se projekcijski prostor dobiva kao unija odgovarajućeg Euklidskog prostora i hiperravnine u beskonačnosti.

Od posebnog interesa će se pokazati ravninska projekcijska preslikavanja $\mathbb{P}^2 \rightarrow \mathbb{P}^2$ koja se često kraće nazivaju *homografijama* i označavaju s $\mathbf{H}_{3\times3}$. Geometrijski elementi projekcijske ravnine i njihova projekcijska svojstva su sažeti u tablici 4.1. Raspišimo za ilustraciju najsloženiju jednadžbu iz te tablice, projekcijski prikaz krivulje drugog reda (*engl.* conic). Neka je $\mathbf{q}$ element projekcijskog prostora koji ima prikaz u odgovarajućem kanonskom Euklidskom prostoru ($q_3 \neq 0$), te neka je $\mathbf{C}$ proizvoljna matrica:

$$\mathbf{q} = \begin{bmatrix} x & y & 1 \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \tag{4.5}$$

Tada se jednostavnim raspisivanjem lijeve strane jednadžbe $\mathbf{q}^T\mathbf{C}\mathbf{q} = 0$ dobiva:

$$f + 2\,c\,x + a\,x^2 + 2\,e\,y + 2\,b\,x\,y + d\,y^2 = 0, \tag{4.6}$$

što pokazuje da vrijedi izomorfizam dvaju prostora jer je (4.6) upravo jednadžba opće krivulje drugog reda (npr, kose elipse, parabole, hiperbole) u Euklidskoj ravnini.

objekt	prikaz	jednadžba	transformacija
točka	$\mathbf{q} = \begin{bmatrix} x & y & t \end{bmatrix}^T$		$\mathbf{Hq}$
pravac	$\mathbf{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$	$\mathbf{l}^T\mathbf{q} = 0$	$\mathbf{H}^{-T}\mathbf{l}$
krivulja 2. reda	$\mathbf{C}_{3\times3} = \mathbf{C}_{3\times3}^T$	$\mathbf{q}^T\mathbf{C}\mathbf{q} = 0$	$\mathbf{H}^{-T}\mathbf{C}\mathbf{H}^{-1}$

Tablica 4.1: Elementi projekcijske ravnine; koristi se konvencija $\mathbf{A}^{-T} = (\mathbf{A}^{-1})^T$.

4.1.2 Odabrane metode linearne algebre

Kao što će se pokazati u odjeljku 4.2, transformacije koje se događaju tijekom stvaranja slike se mogu elegantno opisati matričnim operacijama. Posebno će pri tome biti zanimljive dekompozicije matrica i metode rješavanja linearnih sustava s viškom ograničenja. Potrebni postupci će ovdje biti ukratko opisani sa stanovišta krajnjeg korisnika, dok se više detalja može naći u literaturi [weisstein:www, press93].

Dekompozicije matrica

Postupci dekompozicije propisuju postupke svođenja zadane matrice na odgovarajuću kanonsku formu, a te forme su najčešće umnošci matrica specijalnog oblika. Iako većina postupaka pretpostavlja kvadratnu ulaznu matricu, neki postupci su primjenjivi i na matrice proizvoljnog oblika. Većinu postupaka moguće je provesti bez ikakvih ograničenja na sadržaj zadane matrice što je posebno bitno kod numeričkih postupaka, kada se matrica dobiva na temelju ulaznih podataka. U nastavku teksta će biti navedeni svi takvi postupci koji su bili korišteni u izvedbi eksperimentalnog sustava.

Dekompozicija L–U

Za proizvoljnu kvadratnu matricu $\mathbf{A}$, rezultat dekompozicije su:

- donja trokutasta matrica $\mathbf{L}$,
- gornja trokutasta matrica $\mathbf{U}$.

Vrijedi:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \tag{4.7}$$

Ovo je jedna od najčešće korištenih dekompozicija, jer predstavlja najefikasniji način za rješavanje klasičnih linearnih sustava s n jednadžbi i n nepoznanica. Rezultat dekompozicije je jedinstven, uz proizvoljni dodatni uvjet $\mathbf{L}_{ii} = 1, \forall i$.

Dekompozicija Q–R

Za proizvoljnu kvadratnu matricu $\mathbf{A}$, jedinstveni rezultat dekompozicije su:

- gornja trokutasta matrica $\mathbf{R}$,
- ortogonalna matrica $\mathbf{Q}$ (vrijedi $\mathbf{Q}^T \cdot \mathbf{Q} = \mathbf{I}$).

Vrijedi:

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \tag{4.8}$$

Ova dekompozicija se koristi samo u specijalnim slučajevima jer je dvostruko sporija od dekompozicije L–U.

Singularna dekompozicija

Za proizvoljnu matricu $\mathbf{A}_{m\times n}$ (ne mora biti kvadratna), rezultat dekompozicije su:

- matrica s ortogonalnim stupcima $\mathbf{U}_{m\times n}$,
- dijagonalna pozitivno semidefinitna matrica $\mathbf{D}_{n\times n}$,
- ortogonalna matrica $\mathbf{V}_{n\times n}$.

Vrijedi:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^T \tag{4.9}$$

Matrice $\mathbf{D}$ i $\mathbf{V}$ imaju neka posebna svojstva:

- i–ti element dijagonalne matrice $\mathbf{D}$ odgovara drugom korijenu i–te svojstvene vrijednosti matrice $\mathbf{A}^T\mathbf{A}$;
- Vrijedi: $\mathbf{D}_{ii} \in \mathbb{R}$, $\mathbf{D}_{ii} \geq \mathbf{D}_{jj} \geq 0$, $\forall i, j : i > j$;
- i–ti stupac ortogonalne matrice $\mathbf{V}$ odgovara i–tom svojstvenom vektoru matrice $\mathbf{A}^T\mathbf{A}$.

S obzirom da ima ortogonalne stupce, matrica $\mathbf{U}$ čuva normu:

$$\|\mathbf{U}\mathbf{x}\| = (\mathbf{U}\mathbf{x})^T(\mathbf{U}\mathbf{x}) = \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x} = \mathbf{x}^T(\mathbf{U}^T\mathbf{U})\mathbf{x} = \|\mathbf{x}\| \tag{4.10}$$

Ova dekompozicija se najčešće koristi pri rješavanju linearnih sustava s pravokutnim ili singularnim matricama, tj. kod sustava s n nepoznanica i m jednadžbi, uz $n \neq m$.

Dekompozicija po Choleskom

Neka je zadana simetrična pozitivno definitna matrica $\mathbf{A}$. Rezultat dekompozicije je gornja trokutasta matrica $\mathbf{U}$, za koju vrijedi:

$$\mathbf{A} = \mathbf{U}^T \cdot \mathbf{U} \tag{4.11}$$

Općenito, dekompozicija se koristi kad god je to moguće jer je dvostruko brža od LU dekompozicije.

Homogeni linearni sustavi s viškom ograničenja

Neka je zadan homogeni linearni sustav s viškom ograničenja (*engl.* overconstrained, overdetermined linear system):

$$\mathbf{A}_{m\times n} \cdot \mathbf{x}_n = 0,\ m > n \tag{4.12}$$

Zadani sustav ima m jednadžbi i n nepoznanica, te ima jedinstveno netrivijalno rješenje definirano do množenja s proizvoljnom konstantom, ako i samo ako rang pravokutne matrice $\mathbf{A}$ iznosi $n-1$. Obično se zahtijeva da rješenje bude ortogonalni vektor s čim se izražavaju želje za njegovom netrivijalnošću i jedinstvenošću. Nadalje, gotovo svaka "stvarna" matrica ima puni rang, pa se u praksi obično traži "najbolje" rješenje u smislu minimalne kvadratne pogreške, koje uvijek postoji:

$$\min_x \|\mathbf{A}\mathbf{x}\|,\ \text{uz } \|\mathbf{x}\| = 1 \tag{4.13}$$

Traženo rješenje se dobiva iznenađujuće brzo, primjenom singularne dekompozicije.

Teorem 4.1. *Netrivijalno rješenje homogenog linearnog sustava s viškom ograničenja u skladu s* (4.13) *dobiva se kao svojstveni vektor matrice* $\mathbf{A}^T\mathbf{A}$ *koji odgovara njenoj svojstvenoj vrijednosti s najmanjim apsolutnim iznosom.*

Dokaz. Neka je $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$; tada tražimo $\mathbf{x}$ koji zadovoljava:

$$\min_x \|\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{x}\|, \text{ uz } \|\mathbf{x}\| = 1 \tag{4.14}$$

Kao što je prikazano u (4.10), $\mathbf{U}$ čuva normu zbog ortogonalnosti stupaca, pa se iz gornje jednadžbe može izostaviti. Neka je $\mathbf{q} = \mathbf{V}^T\mathbf{x}$. Tada vrijedi:

$$\|\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{x}\| = \|\mathbf{D}\mathbf{q}\|, \text{ uz } \|\mathbf{q}\| = 1 \tag{4.15}$$

Elementi $\mathbf{D}$ su pozitivni i padajući, pa $\mathbf{q}$ koji minimizira (4.15) iznosi $\mathbf{q}_s = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}^T$ Odatle slijedi ono što je trebalo dokazati:

$$\mathbf{x} = \mathbf{V}\mathbf{q}_s = \mathbf{V}_{:n} \tag{4.16}$$

□

Dotjerivanje rotacijske matrice

Zbog raznih numeričkih pogrešaka, rotacijska matrica $\mathbf{Q}$ koja se dobiva numeričkim postupcima ne mora biti ortogonalna kakva bi u idealnom slučaju trebala biti. Podatak o ortogonalnosti se stoga koristi kao dodatan zahtjev u postupku traženja rotacijske matrice kako bi se smanjila greška i izbjegle nekonzistentnosti u daljnjim proračunima. U skladu s tom zamisli, konačna rotacijska matrica $\mathbf{R}$ se izražava kao ortogonalna matrica koja je najbliža ulaznoj matrici $\mathbf{Q}$, u smislu minimalne Frobeniusove norme udaljenosti [zhang00]:

$$\min_R |\mathbf{R} - \mathbf{Q}|_F^2, \text{ uz } \|\mathbf{R}^T\mathbf{R}\| = \mathbf{I} \tag{4.17}$$

Neka je trag kvadratne matrice definiran sa $\mathrm{tr}(\mathbf{A}) = \sum_i a_{ii}$. Raspisivanjem (4.17) se dobiva:

$$\begin{aligned}
\sum_i \sum_j (r_{ij} - q_{ij})^2 &= \sum_i \sum_j r_{ij}^2 - 2r_{ij}q_{ij} + q_{ij}^2 = \\
&= \sum_i \sum_j r_{ij}^2 + \sum_i \sum_j q_{ij}^2 - 2\sum_i \sum_j r_{ij}q_{ij} = \\
&= k_1 - k_2 \sum_i \sum_j r_{ij}q_{ij} = \\
&= k_1 - k_2\,\mathrm{tr}(\mathbf{R}^T\mathbf{Q})
\end{aligned} \tag{4.18}$$

Trag je invarijantan na transformaciju sličnosti [weisstein:www] pa vrijedi: $\mathrm{tr}(\mathbf{B}^{-1}\mathbf{A}\mathbf{B}) = \mathrm{tr}(\mathbf{A})$. Uz singularnu dekompoziciju $\mathbf{Q}$, iz (4.18) slijedi:

$$\begin{aligned}
\min_R |\mathbf{R} - \mathbf{Q}|_F^2 &= \max_R \mathrm{tr}(\mathbf{R}^T\mathbf{Q}) = \\
&= \max_R \mathrm{tr}(\mathbf{R}^T\mathbf{U}\mathbf{D}\mathbf{V}^T) = \\
&= \max_R \mathrm{tr}(\mathbf{V}^T\mathbf{R}^T\mathbf{U}\mathbf{D})
\end{aligned} \tag{4.19}$$

Uvedimo $\mathbf{Z} = \mathbf{V}^T\mathbf{R}^T\mathbf{U}$. $\mathbf{Z}$ je ortogonalna pa vrijedi:

$$\mathrm{tr}(\mathbf{Z}\mathbf{D}) = \sum_i z_{ii}d_{ii} \le \sum_i d_{ii} \tag{4.20}$$

$\mathbf{Z}$ koji maksimizira (4.20) iznosi $\mathbf{I}$, pa je rješenje (4.19):

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T \tag{4.21}$$

Postupak opisan u [zhang00] je nepotpun, jer dozvoljava ishod $\|\mathbf{R}\| = -1$, koji je malo vjerojatan ali ipak moguć. U tom slučaju dobivena ortogonalna matrica $\mathbf{R}$ ne bi bila rotacija nego tzv. rotoinverzija što bi moglo obezvrijediti naknadne proračune. Ovakav problem se može riješiti množenjem apsolutno najmanjeg stupca ili retka matrice $\mathbf{R}$ s -1.

4.2 Model stvaranja slike

Tema ovog odjeljka je kvantificiranje geometrijskih odnosa u kontekstu izolirane kamere. Pojmovi stvaranja slike i umjeravanja kamere će se prvo ilustrirati na najopćenitijem slučaju, koji se može primijeniti na sve kamere s izoštrenom slikom. U nastavku će se postaviti uobičajeni perspektivni model stvaranja slike, te razraditi proširenja tog modela koja su potrebna u kontekstu vanjskog referentnog koordinatnog sustava. Konačno, opisat će se nelinearno proširenje modela koje je potrebno za ispravljanje radijalnih izobličenja. Praksa potvrđuje da predložena kombinacija perspektivnog modela i modela radijalnih izobličenja može vrlo dobro opisati većinu komercijalno dostupnih kamera.

4.2.1 Ponašajni model kamere

Neka su točke scene opisane Euklidskim 3D prostorom, i neka se elementi senzorskog polja kamere nalaze na Euklidskoj slikovnoj ravnini π. Stvaranje slike se tada može opisati kao *preslikavanje* $f : \mathbb{R}^3 \to \mathbb{R}^2$ točaka Euklidskog 3D prostora u elemente slikovne ravnine π. Vidi se da domena preslikavanja ima veći broj dimenzija od kodomene pa se radi o preslikavanju oblika "više na jedan". Preciznije, u svaki element slikovne ravnine preslikava se cijeli polupravac točaka prostora, zbog pravocrtnog širenja svjetlosti. Neka $\mathcal{P}$ označava skup svih polupravaca domene preslikavanja odnosno 3D scene. Tada vrijedi:

$$\forall q \in \pi \; \exists\, p \in \mathcal{P} : f(Q) = q, \; \forall Q \in p \tag{4.22}$$

Svaki polupravac $p \in \mathcal{P}$ korišten u gornjoj jednadžbi definira svjetlosnu zraku koja podražava odgovarajući senzorski element.

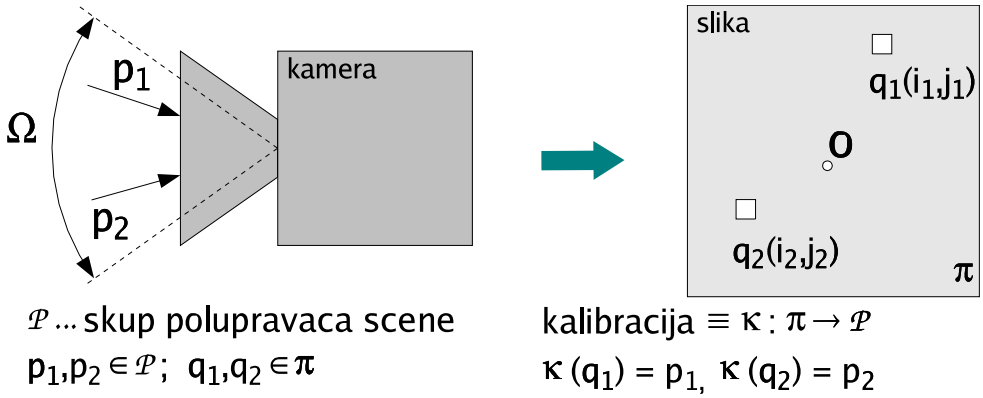
Sada možemo definirati problem umjeravanja ili kalibracije proizvoljne kamere kao pronalaženje preslikavanja koje je *inverzno* procesu stvaranja slike:

$$\kappa : \pi \to \mathcal{P} = ? \tag{4.23}$$

Preslikavanje κ je temelj za određivanje kvantitativnih odnosa u sceni, a tolerancija izvedenih zaključaka će ovisiti o točnosti parametara tog preslikavanja. Proces stvaranja slike te pojam umjeravanja proizvoljne nepoznate kamere ilustrirani su na sl. 4.1.

Model stvaranja slike možemo razmatrati u kontekstima dvaju referentnih koordinatnih sustava:

1. svojstveni koordinatni sustav kamere;
2. vanjski koordinatni sustav svijeta.



Slika 4.1: Proces stvaranja slike u kameri koja je predstavljena crnom kutijom.

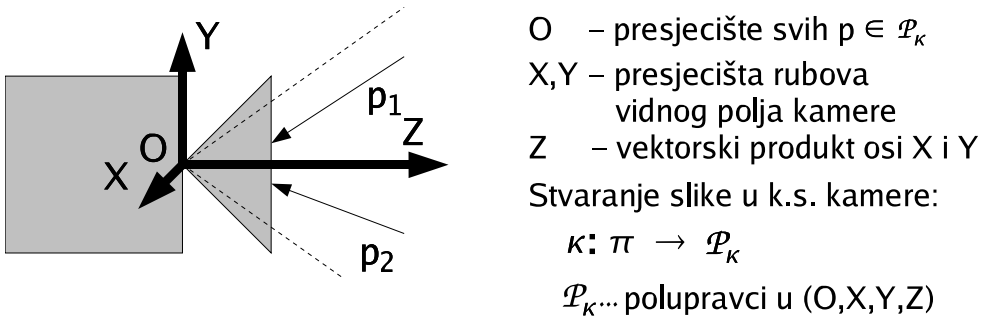
Ovakva podjela je smislena jer su parametri modela u kontekstu k.s. kamere svojstveni korištenoj kameri i ne ovise o eventualnim promjenama položaja kamere. Ti parametri se stoga obično nazivaju *unutrašnjim* ili intrinsičnim parametrima kamere. Stvaranje slike u kontekstu vanjskih koordinatnih sustava se može izraziti položajem kamere u referentnom k.s. scene. Taj pomak odgovara Euklidskoj transformaciji s ukupno šest stupnjeva slobode, a odgovarajući parametri te transformacije se obično nazivaju vanjskim parametrima kamere.

Postojanje svojstvenog koordinatnog sustava se može pokazati zamišljenim algoritmom za njegovu konstrukciju. Neka je π_v "vidljivi" podskup slikovne ravnine, odnosno onaj njen pravokutnik kojeg zauzima senzor kamere (rezolucija senzorskog elementa u ovom trenutku nije važna). Neka je nadalje $\mathcal{P}_{\kappa v}$ "vidljivi" podskup kodomene funkcije κ, koji sadrži one polupravce scene čija slika pada unutar π_v. Tada se ishodište svojstvenog k.s. kamere može konstruirati kao točka za koju je suma udaljenosti do svakog od polupravaca $\mathcal{P}_{\kappa v}$ minimalna:

$$O = \min_{Q} \iint_{q \in \pi_v} d(Q, \kappa(q)), \text{ uz } Q \in \mathbb{R}^3 \tag{4.24}$$

Na sličan način mogu se konstruirati četiri ruba vidnog polja, kao ravnine koje najbolje opisuju unije polupravaca koji se preslikavaju u horizontalne odnosno vertikalne rubove pravokutnika π_v. Osi X i Y k.s. kamere se tada mogu dobiti kao presjecišta parova ravnina koje odgovaraju vodoravnim odnosno uspravnim rubovima vidnog polja. Konačno, glavna os k.s. kamere Z može se dobiti kao vektorski produkt osi X i Y.

Opisani postupak konstrukcije svojstvenog k.s. kamere ilustriran je na sl. 4.2. Iako je zbog brojnih izvedbenih problema u praksi neizvediv, postupak ilustrira činjenicu da je u principu moguće umjeriti i proces stvaranja slike i vanjske parametre bilo koje kamere isključivo korištenjem geometrijskih metoda.



Slika 4.2: Konstrukcija svojstvenog koordinatnog sustava kamere (detaljnije u tekstu).

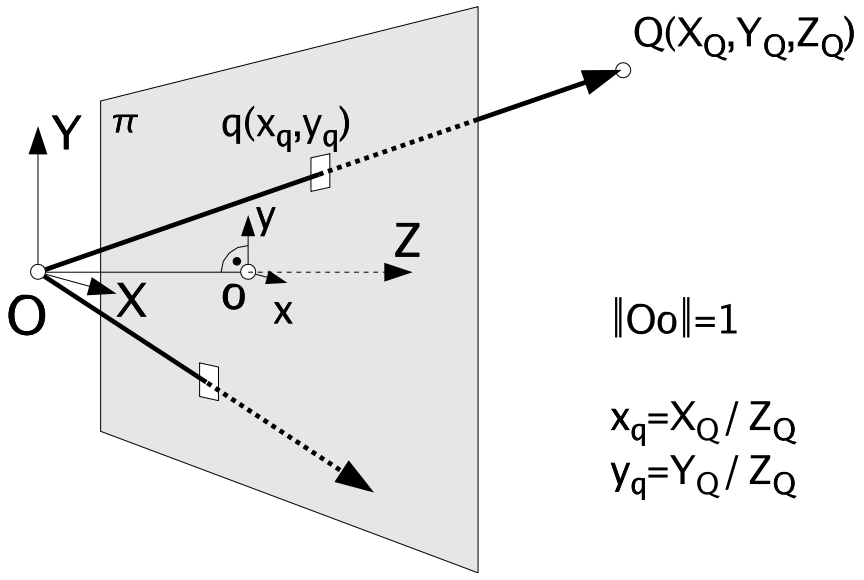
4.2.2 Linearni model stvaranja slike

U prethodnoj raspravi nije korištena niti jedna pretpostavka o izvedbenim detaljima korištene kamere. Takav općeniti pristup u praksi najčešće nije potreban, jer se svi proizvođači trude proizvesti kamere koje daju sliku sa što manjim *nelinearnim izobličenjima*. Nelinearna izobličenja u slici se manifestiraju zakrivljenim linijama koje odgovaraju ravnim bridovima scene. Nažalost, većina stvarnih kamera ipak daje donekle izobličenu sliku, pri čemu su obično najizraženija radijalna izobličenja. Stoga se stvarne kamere obično opisuju perspektivnim modelom koji definira stvaranje neizobličene slike, koji se po potrebi proširuje modelom korekcije radijalnih izobličenja.

Temeljni model idealne kamere

Definirajmo *idealnu* kameru kao kameru koja daje slike bez nelinearnih izobličenja, kod kojih se pravocrtni bridovi scene preslikavaju u odsječke pravaca odnosno dužine. Pokazuje se da se kod takvih kamera svi polupravci iz $\mathcal{P}_{kv}$ sijeku u tzv. glavnoj točki projekcije[1] u kojoj se onda nalazi i ishodište k.s. kamere, dok je glavna os projekcije Z okomita na slikovnu ravninu i prolazi kroz njeno ishodište. Temeljni model idealne kamere prikazan je na sl. 4.3, a zasniva se na sljedećim dodatnim pretpostavkama:

- središte senzora postavljeno je točno u projekciji ishodišta k.s. kamere odnosno žarišta leće na ravninu senzora;
- elementi senzora su idealni kvadrati;
- jedinica u k.s. slike dobiva se kao udaljenost senzora od žarišta.



Slika 4.3: Temeljni model idealne kamere.

Izravno iz definicije idealne kamere slijedi da se odgovarajući postupak stvaranja slike može opisati nekom projekcijskom transformacijom na koje je kolinearnost invarijantna. Pod spomenutim dodatnim pretpostavkama, međutim, stvaranje slike modelira kanonska perspektivna projekcija kao najjednostavnija projekcijska transformacija $\mathbb{P}^3 \to \mathbb{P}^2$. U skladu s tom

[1] Zbog načina izvedbe kamera, glavna točka projekcije se često naziva i žarištem leće kamere.

transformacijom, točke k.s. kamere $(X_Q, Y_Q, Z_Q) \in (O, X, Y, Z)$ preslikavaju se u odgovarajuće točke k.s. slike $(x_q, y_q) \in (o, x, y)$ prema sljedećoj matričnoj jednadžbi:

$$Z_Q \cdot \begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_Q \\ Y_Q \\ Z_Q \\ 1 \end{bmatrix}, Z_Q \neq 0. \tag{4.25}$$

Jednadžba (4.25) se može zapisati u još elegantnijem obliku, ako točke zapišemo kao elemente odgovarajućih projekcijskih prostora $\mathbf{q} \in \mathbb{P}^2$ i $\mathbf{Q} \in \mathbb{P}^3$, te uvedemo pokratu za matricu kanonske perspektivne projekcije:

$$\mathbf{q} = \lambda_1 \begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix}, \ \mathbf{Q} = \lambda_2 \begin{bmatrix} X_Q \\ Y_Q \\ Z_Q \\ 1 \end{bmatrix}, \ \forall \lambda_1, \lambda_2 \neq 0; \quad \mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.26}$$

Konačni oblik temeljnog modela kamere glasi:

$$\mathbf{q} = \mathbf{S} \cdot \mathbf{Q}. \tag{4.27}$$

Zanimljivo je napomenuti da jednadžba (4.27) vrijedi u svim točkama ravnine (X, Y) osim ishodišta, dok jednadžba (4.25) nije definirana ni u jednoj od tih točaka.

Opći perspektivni model idealne kamere

U praksi, udaljenosti u slici se obično izražavaju u dimenzijama slikovnih elemenata, dok središte senzora obično nije dobro poravnato s glavnom osi k.s. kamere. Zbog toga se uvodi opći perspektivni model kao linearno proširenje temeljnog modela (4.27) s pet nezavisnih parametara:

$$\mathbf{q} = \begin{bmatrix} s_x & s_\theta & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{S} \cdot \mathbf{Q}. \tag{4.28}$$

Zbog jednostavnosti zapisa, uvodi se oznaka za matricu parametara općeg perspektivnog modela:

$$\mathbf{K} = \begin{bmatrix} s_x & s_\theta & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.29}$$

Raspisivanjem jednadžbe (4.28) došlo bi se do sljedećeg značenja pojedinih parametara:

- s_x, s_y – iznos jedinice na osima k.s. slike u odnosu na žarišnu udaljenost:
  - mogu se interpretirati kao iznos žarišne daljine izražen u jedinicama horizontalne odnosno vertikalne dimenzije senzorskih elemenata;
  - oblik senzorskih elemenata je dan sa $r = \frac{s_y}{s_x}$, elementi su obično kvadrati ($r = 1$);
  - horizontalna širina vidnog polja se dobiva po formuli: $\phi_{hfov} = 2 \cdot arctg(\frac{w/2}{s_x})$, za senzor dimenzija $w \times h$.

- t_x, t_y – pomak ishodišta k.s. slike u odnosu na središte senzora.
- $s_\theta = tan(\theta)/s_x$ – kut među osima k.s. slike ($s_\theta \approx 0$)

Iz jednadžbe (4.28) se vidi da se primjenom matrice $\mathbf{K}^{-1}$ elementi slike dobivene svakom idealnom kamerom mogu preslikati u prikaz koji bi se dobio temeljnim modelom. Takav prikaz je ponekad potreban, pa se odgovarajući koordinatni sustav naziva *normaliziranim* k.s. slike. Da bi se razlikovali od regularnih slikovnih elemenata, elementi normaliziranog k.s. slike se obično označavaju "kapom", tako da vrijedi:

$$\mathbf{q} = \mathbf{K} \cdot \hat{\mathbf{q}}, \quad \hat{\mathbf{q}} = \mathbf{S} \cdot \mathbf{Q}. \tag{4.30}$$

Vanjski parametri kamere

Pokazuje se da je projekcijsku geometriju prikladno koristiti i kod modeliranja položaja kamere u k.s. svijeta, zbog elegantnog linearnog zapisa pojedinih transformacija. Svaka opća Euklidska transformacija $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ sa šest stupnjeva slobode da se zapisati kao projekcijska transformacija među odgovarajućim projekcijskim prostorima $\mathbb{P}^3 \rightarrow \mathbb{P}^3$. U konkretnom slučaju, interesantno je dobiti prikaz točke scene u k.s. kamere, ako je poznat njen prikaz u k.s. svijeta $(X_{Q_W}, Y_{Q_W}, Z_{Q_W})$, ako je poznato da su koordinate ishodišta k.s. kamere u k.s. svijeta dane translacijskim vektorom $\mathbf{t}$, te da je rotacija iz k.s. svijeta u k.s. kamere dana rotacijskom matricom $\mathbf{R}$. Vrijedi sljedeća formula čija ispravnost se dokazuje elementarnim raspisivanjem:

$$\begin{bmatrix} X_Q \\ Y_Q \\ Z_Q \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{3x3} & -\mathbf{R}_{3x3}\mathbf{t}_{3x1} \\ 0_{1x3} & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{Q_W} \\ Y_{Q_W} \\ Z_{Q_W} \\ 1 \end{bmatrix} \tag{4.31}$$

Kao i prije, elegantniji zapis jednadžbe (4.31) se dobiva ako točke zapišemo kao elemente odgovarajućih projekcijskih prostora te uvedemo pokratu za matricu transformacije:

$$\mathbf{Q_W} = \lambda \begin{bmatrix} X_{Q_W} \\ Y_{Q_W} \\ Z_{Q_W} \\ 1 \end{bmatrix}, \ \forall \lambda \neq 0; \quad \mathbf{M} = \begin{bmatrix} \mathbf{R}_{3x3} & -\mathbf{R}_{3x3}\mathbf{t}_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}. \tag{4.32}$$

Konačni projekcijski oblik tražene Euklidske transformacije glasi:

$$\mathbf{Q} = \mathbf{M} \cdot \mathbf{Q_W} \tag{4.33}$$

Rotacijska matrica ima tri stupnja slobode, a svaki od njih opisuje elementarnu rotaciju oko jedne od koordinatnih osi. Redoslijed primjena rotacija je pri tome bitan jer množenje matrica nije komutativno. Stoga se u literaturi može naći velik broj *parametrizacija* rotacijske matrice, koje zapravo propisuju redoslijed primjena elementarnih rotacija. U eksperimentima je korištena subjektivno najplauzibilnija parametrizacija (nagib,zakret,okret), u kojoj se prvo primjenjuje rotacija oko osi X, zatim oko osi Y, i konačno oko osi Z.

$$\mathbf{R} \ = \ \mathbf{R_z}(\gamma) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha) \tag{4.34}$$

Ovdje valja napomenuti da prikazana parametrizacija nije jedinstvena, jer za svaku zadanu matricu postoje barem dvije trojke (α,β,γ) za koje izraz (4.34) poprima vrijednost te matrice. Raspisivanje izraza (4.34) povjereno je specijaliziranom računalnom programu, a konačni rezultat je dan sljedećim izrazom:

$$\mathbf{R} = \begin{bmatrix} \cos(\beta)\cos(\gamma) & \cos(\gamma)\sin(\alpha)\sin(\beta) - \cos(\alpha)\sin(\gamma) & \cos(\alpha)\cos(\gamma)\sin(\beta) + \sin(\alpha)\sin(\gamma) \\ \cos(\beta)\sin(\gamma) & \cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\beta)\sin(\gamma) & \cos(\gamma)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\alpha) & \cos(\alpha)\cos(\beta) \end{bmatrix}$$

Kao sažetak prethodne rasprave, može se reći da vanjski parametri kamere određuju položaj kamere jednadžbom (4.33), te da se mogu predstaviti uređenom šestorkom $(\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z, \alpha, \beta, \gamma)$. Ponekad je zanimljivo dobiti inverzno Euklidsko preslikavanje, koje opisuje položaj k.s. svijeta u k.s. kamere. Rješenje se dobiva invertiranjem matrice vanjskih parametara koje se jednostavno provodi zbog ortogonalnosti rotacijske matrice:

$$\begin{aligned} \mathbf{M}_i &= \mathbf{M}^{-1} = \begin{bmatrix} \mathbf{R_i}_{3x3} & -\mathbf{R_i}_{3x3}\mathbf{t_i}_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}, \\ \mathbf{R_i} &= \mathbf{R}^{-1} = \mathbf{R}^T, \\ \mathbf{t_i} &= \mathbf{R}^{-1} \cdot \mathbf{t}. \end{aligned} \tag{4.35}$$

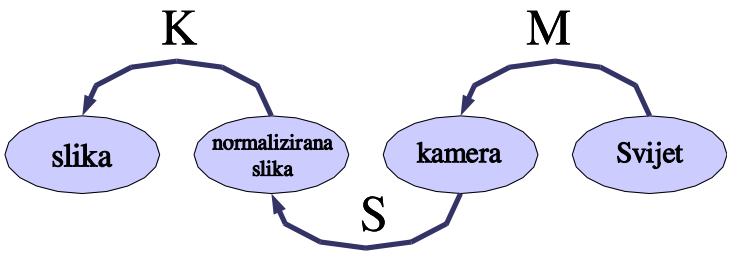
Sažetak ukupnog linearnog modela kamere

Ukupni linearni model podrazumijeva sljedeći niz transformacija:

1. Euklidska transformacija $\mathbf{M}$ iz k.s. svijeta u k.s. kamere, definirana sa šest vanjskih parametara kamere;
2. kanonska perspektivna projekcija $\mathbf{S}$ iz k.s. kamere u normalizirani k.s. slike;
3. opća perspektivna transformacija $\mathbf{K}$ iz normaliziranog k.s. slike u k.s. slike, koja je definirana s pet unutrašnjih parametara kamere.

Ukupna transformacija dana je sljedećim izrazom, a ilustrirana je na sl. 4.4:

$$\mathbf{q} = \mathbf{K} \cdot \mathbf{S} \cdot \mathbf{M} \cdot \mathbf{Q_W} = \mathbf{P}_{3\times4} \cdot \mathbf{Q_W} \tag{4.36}$$

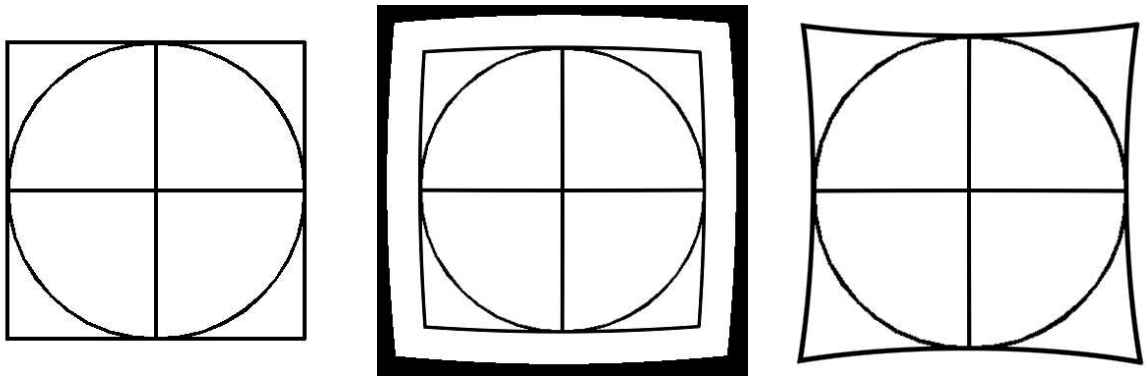


Slika 4.4: Ukupni linearni model stvaranja slike (detaljnije u tekstu).

Ukupni linearni model se sastoji od unutrašnjih i vanjskih parametara. Unutrašnji parametri svakom slikovnom elementu pridružuju polupravac kroz žarište, dok vanjski parametri određuju interpretaciju tih pravaca u nekom vanjskom koordinatnom sustavu.

4.2.3 Model radijalnog izobličenja leće

Linearni model stvaranja slike (4.36) može se primijeniti kod svih kamera s izoštrenom slikom i idealnom ("tankom") lećom. Nažalost, posljednji uvjet često nije zadovoljen, pogotovo kod kamera sa širokim kutem vidnog polja, pa je kod takvih kamera linearni model potrebno proširiti postupkom ispravljanja radijalnih izobličenja. Postoje dva temeljna oblika manifestacije radijalnih izobličenja kao što je prikazano na sl. 4.5: bačvasti koji je češći, i šiljati. Kod



(a) (b) (c)

Slika 4.5: Originalna slika (a), te bačvasti (b) i šiljati (c) oblici radijalnog izobličenja.

oba oblika, i korekcija i izobličenje se modeliraju istim jednadžbama, i to nad elementima normaliziranog k.s. slike. Nelinearna transformacija kojom se opisuju i izobličenja i njihova korekcija glasi [tamaki02]:

$$\begin{aligned} f_{ab}(\hat{x}_a) &= \hat{x}_a \cdot (1 + k_1^{ab} \cdot \hat{r}_a^2 + k_2^{ab} \cdot \hat{r}_a^4) \\ f_{ab}(\hat{y}_a) &= \hat{y}_a \cdot (1 + k_1^{ab} \cdot \hat{r}_a^2 + k_2^{ab} \cdot \hat{r}_a^4) \\ \hat{r}_a^2 &= \hat{x}_a^2 + \hat{y}_a^2 \end{aligned} \tag{4.37}$$

U opisanim jednadžbama, generički indeksi a i b mogu poprimiti vrijednosti u odnosno d, ili obratno. Pri tome u označava korigirane vrijednosti koordinata (*engl.* undistorted), dok d označava njihove izobličene vrijednosti (*engl.* distorted). Nadalje, smjer ud označava model izobličenja, dok smjer du označava model korekcije. U skladu s tim, postoje četiri parametra modela, dva za izobličenje, k_1^{ud} i k_2^{ud} te dva za korekciju k_1^{du} i k_2^{du}. Primjena jednadžbi je ista u oba smjera, i svodi se na pomak svakog slikovnog elementa u skladu s rezultatima odgovarajuće funkcije nad obje koordinate:

$$\begin{aligned} \hat{x}_d &= f_{ud}(\hat{x}_u) \\ \hat{y}_d &= f_{ud}(\hat{y}_u) \\ \hat{x}_u &= f_{du}(\hat{x}_d) \\ \hat{y}_u &= f_{du}(\hat{y}_d) \end{aligned} \tag{4.38}$$

Kod nekih kamera se mogu primijetiti i neka druga nelinearna izobličenja. Eksperimentalni rezultati su pokazali da ta izobličenja nisu toliko izražena na korištenim kamerama ili da nisu negativno utjecala na preciznost proračuna. Ostala izobličenja stoga nisu detaljnije razmatrana te se ovdje navode samo radi kompletnosti teksta.

- Zatamnjenje površina na periferiji slike:
  Efektivni presjek leće $S_{\text{eff}} = S \cdot \cos(\varphi)$ ovisi o upadnom kutu φ zrake u odnosu na glavnu os projekcije Z. Posljedica toga je zatamnjenje ravnomjerno osvijetljenih površina na periferiji slike, ali nema utjecaja na geometrijsku preciznost podataka.

- Tangencijalna nelinearna izobličenja [zhang00]:
  Na korištenim kamerama nisu zamjetna.

- Kromatska aberacija:
  Zbog nesavršene leće, dijelovi slike na granici raznobojnih regija budu zamućeni. Na korištenim kamerama nije zamjetna.

4.3 Umjeravanje unutrašnjih parametara kamere

Tema ovog odjeljka su postupci za umjeravanje unutrašnjih parametara kamere. U prvom dijelu navedeni su najvažniji aspekti tog problema, kao i pristupi za njihovo rješavanje koji su opisani u literaturi. Postupak temeljen na većem broju pogleda na ravninski uzorak [zhang00] je u kontekstu ovog rada ocijenjen kao najprikladniji, te je detaljno opisan u drugom dijelu odjeljka. Na kraju odjeljka opisana je originalna metoda detekcije značajki mjernog uzorka, kao neizostavni izvedbeni detalj postupka umjeravanja. Dobiveni eksperimentalni rezultati uvršteni su u poglavlje 6.

4.3.1 Aspekti umjeravanja unutrašnjih parametara

Pristupi umjeravanju unutrašnjih parametara kamere se na najgrubljoj razini mogu podijeliti na metode koje podrazumijevaju mjerni uzorak, i metode samoumjeravanja koje takvo što ne pretpostavljaju. Prednost prvog pristupa je velika preciznost dobivenih parametara, dok je prednost samoumjeravanja primjenljivost u kompliciranim slučajevima, kad se vanjski ili unutrašnji parametri tijekom obrade mijenjaju. Međutim, pored te osnovne podjele, postoji niz općenitih aspekata tog problema koji su vrlo bitni za kvalitetnu izvedbu postupka, kao npr, potreba za nelinearnom optimizacijom, ili odabir strukture mjernog uzorka te oblika mjernih značajki. Ti aspekti će biti uvođeni prirodnim redoslijedom, kada se ukaže prilika za opis njihovog značaja.

Postupak umjeravanja korištenjem 3D uzorka

Konceptualno najjednostavnija metoda umjeravanja unutrašnjih parametara kamere određuje parametre linearnog modela upotrebom 3D mjernog uzorka. U takvom pristupu, točke mjernog uzorka $\mathbf{Q}_i$ čiji precizni položaji su unaprijed poznati se uparuju s detektiranim projekcijama u slici $\mathbf{q}_i$. Uvjet za uspješnost postupka je postavljanje uzorka u vidno polje kamere na takav način da vidljive mjerne točke ne budu sve koplanarne te da budu ravnomjerno raspoređene po cijeloj slici. U prvom koraku, traži se projekcijska matrica $\mathbf{P}_{3\times4}$ koja najbolje opisuje detektirani raspored mjernih točaka, u smislu minimalne sume kvadrata pojedinačnih projekcijskih pogrešaka:

$$\min_P \sum_i d(\mathbf{P}\cdot\mathbf{Q}_i, \mathbf{q}_i)^2. \tag{4.39}$$

Za svaki par točaka vrijedi jednadžba:

$$\mathbf{P}\cdot\mathbf{Q} = \lambda\cdot\mathbf{q}, \quad \lambda \neq 0. \tag{4.40}$$

Parametar λ u toj jednadžbi je proizvoljan, zbog prirode elemenata projekcijskog prostora. Stoga se $\mathbf{q}$ i $\mathbf{P}\cdot\mathbf{Q}$ mogu interpretirati kao paralelni vektori, pa se isti uvjet može prikazati sljedećim vektorskim produktom:

$$\mathbf{q}\times(\mathbf{P}\cdot\mathbf{Q}) = 0. \tag{4.41}$$

To je ujedno i najelegantniji oblik uvjeta iz jednadžbe (4.40), jer se izbjegava opasnost dijeljenja s nulom [spies03]. Uz konvenciju da $\mathbf{P}_{i:}$ predstavlja i–ti redak matrice $\mathbf{P}$, razradom

prethodne jednadžbe dobiva se:

$$\begin{aligned}\mathbf{q}\times(\mathbf{PQ}) &= \begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{P}_{1:}\mathbf{Q} \\ \mathbf{P}_{2:}\mathbf{Q} \\ \mathbf{P}_{3:}\mathbf{Q} \end{bmatrix} \\ &= \begin{bmatrix} y_q\cdot\mathbf{P}_{3:}\mathbf{Q} - 1\cdot\mathbf{P}_{2:}\mathbf{Q} \\ 1\cdot\mathbf{P}_{1:}\mathbf{Q} - x_q\cdot\mathbf{P}_{3:}\mathbf{Q} \\ x_q\cdot\mathbf{P}_{2:}\mathbf{Q} - y_q\cdot\mathbf{P}_{1:}\mathbf{Q} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_3^T & -\mathbf{Q}^T & y_q\mathbf{Q}^T \\ \mathbf{Q}^T & \mathbf{0}_3^T & -x_q\mathbf{Q}^T \\ -y_q\mathbf{Q}^T & x_q\mathbf{Q}^T & \mathbf{0}_3^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}_{1:}^T \\ \mathbf{P}_{2:}^T \\ \mathbf{P}_{3:}^T \end{bmatrix} = \mathbf{0}_{12}\end{aligned} \tag{4.42}$$

Iz dobivenog se vidi da se za svaku mjernu točku uzorka uparenu s detektiranom točkom u slici dobivaju dvije linearno nezavisne jednadžbe. Slaganjem jednadžbi za n uparenih točaka, dobivamo matricu homogenog linearnog sustava

$$\mathbf{A}_{2n\times 12}\cdot\mathbf{p} = \mathbf{0}_{12} \tag{4.43}$$

Za dobivanje linearnog rješenja treba upariti bar šest točaka, ali u praksi je zbog dobivanja boljih rezultata potrebno koristiti puno više (npr. 100). Rješenje linearnog sustava u smislu minimalne apsolutne vrijednosti rezultata jednadžbe (4.43), dobiva se poznatim metodama i omogućava rekonstrukciju matrice $\mathbf{P}_{3\times 4}$. Nažalost, ovako dobiveno linearno rješenje ima dva nedostatka:

1. dobiveno rješenje ne mora zadovoljavati početni uvjet (4.39);
2. ovakvim pristupom nije moguće pronaći parametre nelinearnih izobličenja.

Dekompozicija projekcijske matrice

Ukoliko je umjerena projekcijska matrica dobivena ovim ili nekim drugim postupkom točna, pojedinačni parametri linearnog modela mogu se odrediti sljedećim postupkom. Množenjem $\mathbf{S}$ i $\mathbf{M}$ u jednadžbi (4.36), dobiva se:

$$\lambda\cdot\mathbf{P} = \mathbf{K}\cdot\mathbf{S}\cdot\mathbf{M} = \mathbf{K}\cdot\begin{bmatrix}\mathbf{R} & -\mathbf{R}\cdot\mathbf{t}\end{bmatrix} \tag{4.44}$$

Označimo lijevu kvadratnu podmatricu od $\mathbf{P}$ s $\mathbf{P}_3$, te i–ti stupac od $\mathbf{P}$ s $\mathbf{P}_{:1}$. Tada vrijedi:

$$\mathbf{P}_3 = \begin{bmatrix}\mathbf{P}_{:1} & \mathbf{P}_{:2} & \mathbf{P}_{:3}\end{bmatrix} = \frac{1}{\lambda}\mathbf{K}\cdot\mathbf{R} \tag{4.45}$$

Dobivena jednadžba pokazuje da je upotrebom QR dekompozicije prva tri stupca projekcijske matrice dobivene umjeravanjem, moguće dobiti i intrinsične i vanjske parametre nepoznate kamere.

Nelinearna optimizacija

Linearno rješenje minimizira rezidual sustava $\mathbf{Ap} = \mathbf{0_{12}}$, odnosno kvadratno odstupanje zadanih ograničenja po elementima projekcijske matrice. To ne zadovoljava dva kriterija [tsai87, zhang00, heikkila00]:

- u skladu s jednadžbom (4.39), treba minimizirati projekcijsku pogrešku

$$\epsilon_p = \sum_i d(\mathbf{P} \cdot \mathbf{Q_i} - \mathbf{q_i})^2, \tag{4.46}$$

- treba odrediti parametre nelinearnih izobličenja.

Zato se linearno rješenje koristi kao početna procjena za gradijentni optimizacijski postupak, uz funkciju cilja ϵ_p Tipično, koristi se Levenberg-Marquardtov optimizacijski postupak implementiran u bibliotekama `minpack` (Fortran) odnosno `cephes` (C) koje su dostupne na `http://www.netlib.org`. Varijable koje postupak tipično optimira su:

- linearni intrinsični parametri kamere ($\mathbf{K}$),
- intrinsični parametri nelinearnih izobličenja (k_1^{ud}, k_2^{ud}),
- vanjski parametri Euklidske transformacije ($\mathbf{M}$).

Upotreba ravninskog mjernog uzorka

Iako je opisani postupak umjeravanja kamere 3D uzorkom konceptualno najjednostavniji, u praksi se često traže alternative zbog komplikacija s preciznom izradom 3D uzorka. Ravinski mjerni uzorak je u tom kontekstu posebno interesantan jer se modernim laserskim pisačem mogu dobiti iznimno precizni 2D uzorci, čiju planarnost je potrebno osigurati lijepljenjem na ravnu podlogu. Noviji eksperimenti pokazuju da se najbolji rezultati dobivaju ukoliko se željeni 2D uzorak iscrta na kvalitetnom LCD ekranu računala [latin04]. Nažalost, upotreba ravninskih uzoraka je povezana sa znatnim teoretskim poteškoćama. Naime, zbog linearne zavisnosti točaka uzorka, nije moguće odrediti linearno rješenje za sve elemente projekcijske matrice na temelju samo *jednog* pogleda! Ovaj problem se može rješavati na najmanje dva načina:

- Pojednostavniti matricu intrinsičnih parametara $\mathbf{K}$, tako da: $s_x = s_y = f$, i $s_\theta = t_x = t_y = 0$ [tsai87]. Pokazuje se daje u tom slučaju linearno rješenje moguće dobiti i na temelju samo jednog pogleda na ravninski uzorak, po cijenu manje izražajnosti korištenog modela.
- Korištenjem domišljatog postupka [zhang00] koji se temelji na *više* različitih pogleda na ravninski mjerni uzorak. Taj postupak je korišten u svim eksperimentima, a detaljnije opisan u 4.3.2.

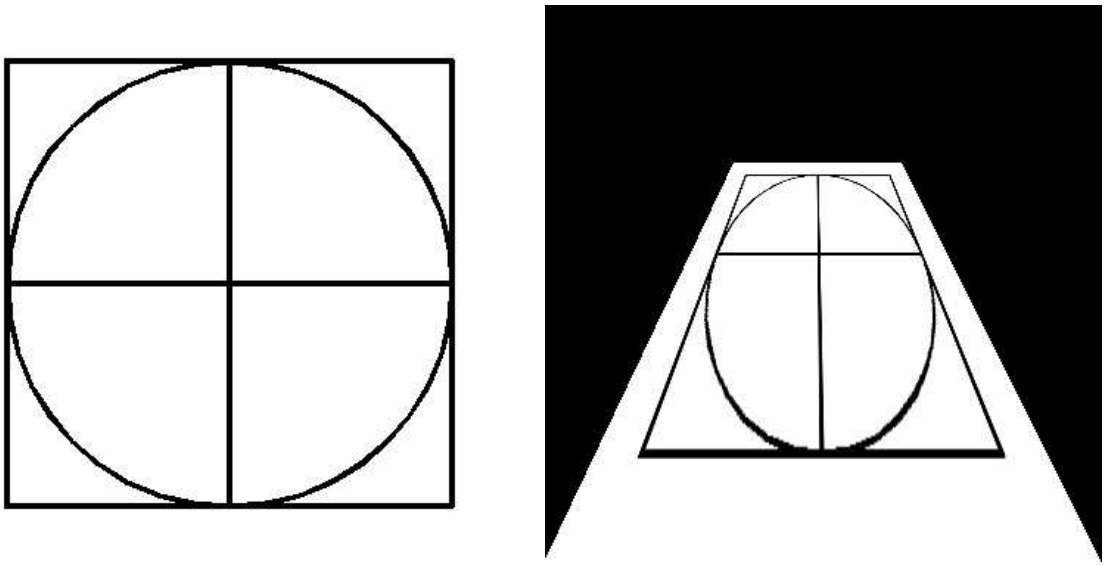
Odabir značajki mjernog uzorka

U dosadašnjim razmatranjima, metoda detekcije značajki mjernog uzorka u pribavljenim slikama nije razmatrana. Iako se radi o tehničkom izvedbenom detalju, postupak je ključan za preciznost dobivenih parametara pa će se ovdje razmotriti mogućnosti za njegovu izvedbu, u ovisnosti o različitim oblicima mjernih značajki. Temeljni zahtjevi na oblik mjerne značajke su donekle kontradiktorni:

- mala površina:
  u vidno polje treba smjestiti velik broj značajki;
- robustnost na šum:
  značajka ne smije biti pre–lokalnog karaktera, njen položaj se mora odrediti na temelju što većeg broja slikovnih elemenata.

- dobra lokalizacija:
  jednoznačna detekcija značajke nije dovoljna, potrebno je moći odrediti njen što točniji položaj.

U literaturi se obično koriste crno–bijele slike krugova [heikkila00] i kvadrata [zhang00], u kojima je položaj značajki određen težištem krugova odnosno vrhovima kvadrata. Prednost vrhova kvadrata je vrlo dobra lokalizacija, dok je prednost težišta krugova veća robustnost uslijed većeg broja slikovnih elemenata koji glasaju za položaj značajke. Težišta krugova imaju slabiju lokalizaciju jer se uslijed homografije težište projiciranog lika ne mora poklopiti s projekcijom težišta originala. Taj problem je opisan na sl. 4.6.



(a) (b)

Slika 4.6: Originalni krug (a), te njegova homografijska slika (b) u kojoj se težište projiciranog lika ne poklapa s projekcijom težišta originalnog kruga.

Korištenje težišta krugova kao mjernih značajki

U skladu s prethodnom raspravom, težišta krugova predstavljala bi najbolji odabir mjernih značajki, ukoliko se uspije korigirati pomak težišta projiciranih krugova u odnosu na projekcije težišta originala. Model te korekcije je opisan u literaturi [heikkila00], a u nastavku teksta će se dati kratka skica tog postupka.

Model preslikavanja točaka

Neka je ravnina uzorka Υ dana ishodištem $\mathbf{u_o}$ te koordinatnim vektorima $\mathbf{u_x}$ i $\mathbf{u_y}$ koji su svi izraženi u k.s. svijeta. Tada se točke iz projekcijske ravnine uzorka $\mathbf{Q}_U = \begin{bmatrix} Q_{Ux} & Q_{Uy} & 1 \end{bmatrix}$ preslikavaju u točke projekcijskog k.s. svijeta $\mathbf{Q}_W$ prema jednadžbi:

$$\mathbf{Q}_W = \begin{bmatrix} \mathbf{u_x} & \mathbf{u_y} & \mathbf{u_o} \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{Q}_U. \tag{4.47}$$

Zbog jednostavnosti zapisa, uvodi se posebna oznaka za matricu transformacije iz k.s. uzorka u k.s.svijeta:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u_x} & \mathbf{u_y} & \mathbf{u_o} \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.48}$$

Transformirane točke se iz k.s. svijeta preslikavaju u k.s. slike matricom projekcijske transformacije $\mathbf{P}_{3\times4}$, prema jednadžbi (4.36). Pokazuje se stoga da postoji homografija $\mathbf{H}$ iz k.s. uzorka u k.s. slike, koja svaku značajku uzorka preslikava u odgovarajući element slikovne ravnine:

$$\begin{aligned} \mathbf{q}_i &= \mathbf{P}\cdot\mathbf{U}\cdot\mathbf{Q}_{Ui} \\ &= \mathbf{H}\cdot\mathbf{Q}_{Ui} \end{aligned} \tag{4.49}$$

Homografija $\mathbf{H}$ poprima posebno jednostavan oblik ako se ravnina uzorka Υ poklapa s $X-Y$ ravninom k.s. svijeta:

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{P}_{:1} & \mathbf{P}_{:2} & \mathbf{P}_{:4} \end{bmatrix} \tag{4.50}$$

Model preslikavanja krugova

Kružnica polumjera r u točki $(Q_{Uix}, Q_{Uiy}) \in \Upsilon$ ima sljedeću jednadžbu, prema tablici 4.1:

$$\mathbf{Q}_U^T\cdot\mathbf{C}\cdot\mathbf{Q}_U = \mathbf{0}. \tag{4.51}$$

Lako se pokazuje da matrica kružnice iznosi:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & -Q_{Uix} \\ 0 & 1 & -Q_{Uiy} \\ -Q_{Uix} & -Q_{Uiy} & Q_{Uix}^2 + Q_{Uiy}^2 + r^2 \end{bmatrix} \tag{4.52}$$

Za odgovarajuću kružnicu u slikovnoj ravnini vrijedi:

$$\mathbf{q}^T\cdot\mathbf{H}^{-T}\mathbf{C}\mathbf{H}^{-1}\cdot\mathbf{q} = \mathbf{q}^T\cdot\mathbf{S}\cdot\mathbf{q} = \mathbf{0} \tag{4.53}$$

Vrlo složenim proračunom se pokazuje [heikkila00] da je (4.53) elipsa s težištem u:

$$\mathbf{q}_i = \mathbf{S}^{-1}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T = \mathbf{H}\mathbf{C}^{-1}\mathbf{H}_{3,:}^T \tag{4.54}$$

Izmnožimo posljednja dva faktora jednadžbe (4.54):

$$\begin{aligned} \mathbf{q}_i &= \mathbf{H}\cdot\mathbf{C}^{-1}\mathbf{H}_{3,:}^T = \mathbf{H}\cdot\begin{bmatrix} Q_{Uix} - \frac{H_{31}\cdot r^2}{\mathbf{H_{3,:}^T Q_{Ui}}} \\ Q_{Uiy} - \frac{H_{32}\cdot r^2}{\mathbf{H_{3,:}^T Q_{Ui}}} \\ 1 \end{bmatrix} \\ &= \mathbf{H}\cdot\mathbf{Q_{Ui}} - \mathbf{H}\cdot\varepsilon_H(\mathbf{H}, \mathbf{Q_{Ui}}, r) \end{aligned} \tag{4.55}$$

Iz dobivene jednadžbe slijedi da kad polumjer kružnice teži nuli, pomak također teži nuli što je u skladu s fizičkom stvarnošću:

$$\begin{aligned} &\lim_{r\to0} \quad \varepsilon_H = 0, \\ &\lim_{r\to0} \quad \mathbf{q}_i = \mathbf{H}\cdot\mathbf{Q_{Ui}} \end{aligned} \tag{4.56}$$

Zaključak

Tražena translacijska korekcija detektiranih težišta krugova ravnine uzorka dana je jednadžbom (4.55). Bez primjene te korekcije, sve detektirane mjerne značajke u slici imale bi sistematsku pogrešku, što bi općenito smanjilo preciznost postupka. Nažalost, korekcija ε_H ovisi o $\mathbf{H} = \mathbf{P}\cdot\mathbf{U}$ (kojeg upravo umjeravamo!), pa nju nije moguće primijeniti prije ulaska u postupak rješavanja linearnih jednadžbi, Stoga se primjena korekcije mora ostaviti za zadnji korak umjeravanja, dakle tek u funkciji cilja nelinearne optimizacije ukupnog modela.

Umjeravanje bez mjernog uzorka

Metode umjeravanja kamere bez mjernog uzorka su poznate kao "samoumjeravanje" kamere (*engl.* self-calibration, auto-calibration). Obično se razmatra scenario s nepomičnom scenom i pokretnom kamerom, a zaključci se izvode na temelju uparivanja točaka pojedinih slika slijeda. Postupci su razvijani pod različitim pretpostavkama, pri čemu kretanje kamere može biti poznato ili ne, dok intrinsični parametri mogu biti konstantni ili ne. Ipak, gotovo svi postupci se temelje na primjeni epipolarnog ograničenja[2] na svakom uzastopnom paru slika iz pribavljenog slijeda. Čini se da je krajnja motivacija za ovakve postupke mogućnost naknadne rekonstrukcije 3D scene na temelju prethodno snimljenih materijala uz proizvoljna pomicanja smjera gledanja te podešavanja širine vidnog polja kamere.

Jedan od temeljnih scenarija samoumjeravanja pretpostavlja umjeravanje konstantne matrice unutrašnjih parametara $\mathbf{K}$, na temelju slijeda slika dobivenih uz nepoznato kretanje kamere. Potreba za takvim scenarijem se javlja ukoliko se žele saznati unutrašnji parametri i dinamika kretanja nepoznate kamere kojom je snimljen neki video materijal, što je prvi korak prema dalekom cilju 3D rekonstrukcije snimljene scene. Za svaki par uzastopnih slika iz pribavljenog slijeda, postavlja se generalizirani stereo problem. Fundamentalna matrica $\mathcal{F}$ se umjerava na temelju epipolarnih ograničenja koja se postavljaju za barem 8 uparenih parova točaka u dvije slike, što rezultira sljedećim linearnim sustavom jednadžbi:

$$\mathbf{q}_i^{(k)}\mathbf{F}\mathbf{q}_i^{(k+1)} = 0, i = 1, 2, \ldots, n,\ n \geq 8 \tag{4.57}$$

S obzirom da se unutrašnji parametri kamere ne mijenjaju, algebarski izraz za umjerenu fundamentalnu matricu glasi:

$$\mathcal{F} = \mathbf{K}^{-T} \cdot [\mathbf{t}_\times]\mathbf{R} \cdot \mathbf{K}^{-1}. \tag{4.58}$$

Fundamentalna matrica ima više stupnjeva slobode od Euklidskog pomaka, pa se uz više parova uzastopnih slika može postaviti sustav nelinearnih jednadžbi u parametrima matrice $\mathbf{K}$.

4.3.2 Umjeravanje upotrebom više pogleda na ravninski uzorak

U ovom pododjeljku će se opisati metoda umjeravanja unutrašnjih parametara kamere upotrebom više pogleda na ravninski uzorak [zhang00]. Glavna zamisao postupka je odrediti parametre linearnog modela postavljanjem apstraktnih ograničenja dobivenih na temelju umjerenih pojedinačnih homografija. Linearni rezultat se može poboljšati nelinearnom optimizacijom bez straha od degeneriranih rezultata uslijed planarnog uzorka, jer se lako osigurava da položaji uzorka u pojedinim pogledima budu raznoliki.

Neka je k.s. svijeta (O, X, Y, Z) orijentiran tako da je os Z okomita na ravninu uzorka. U skladu s jednadžbom (4.50), konstrukcija homografije iz ravnine uzorka u slikovnu ravninu vrlo je jednostavna:

$$\mathbf{q} = \mathbf{P} \cdot \begin{bmatrix} X_Q \\ Y_Q \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{:1} & \mathbf{P}_{:2} & \mathbf{P}_{:4} \end{bmatrix} \cdot \begin{bmatrix} X_Q \\ Y_Q \\ 1 \end{bmatrix} = \mathbf{H} \cdot \mathbf{q}^u, \tag{4.59}$$

gdje je $\mathbf{q}$ položaj detektirane značajke u projekcijskoj ravnini slike, $\mathbf{H}$ tražena homografija, a $\mathbf{q}^u$ položaj originalne mjerne značajke u projekcijskoj ravnini uzorka. Ako se projekcijska matrica $\mathbf{P}$ izrazi prema (4.36), dobiva se sljedeći izraz za homografiju $\mathbf{H}$:

$$\mathbf{H} = \lambda \cdot \mathbf{K} \begin{bmatrix} \mathbf{R}_{:1} & \mathbf{R}_{:2} & \mathbf{t} \end{bmatrix} \tag{4.60}$$

[2] Jednadžbe epipolarnog ograničenja su izvedene u dodatku D.

Teoretska motivacija

Za svaki pogled na mjerni uzorak, moguće je umjeriti po jednu homografiju s 8 stupnjeva slobode. Međutim, Euklidski pomak uzorka u svakom pogledu opisuje se sa samo 6 parametara. Teoretski je stoga moguće iz svakog pogleda izvući $8-6=2$ ograničenja na intrinsične parametre kamere. Ukoliko se navedena ograničenja uspješno matematički izraze, tri pogleda na mjerni uzorak biti će dovoljno za rješenje matrice $\mathbf{K}$ (6 ograničenja za 5 parametara). Ako se model samo malo pojednostavni pretpostavkom da je parametar zakošenosti $s_\theta = 0$, unutrašnje parametre će biti moguće odrediti i samo na temelju dva pogleda. Konačno, na temelju jednog pogleda mogu se procijeniti samo glavni unutrašnji parametre s_x i s_y, uz pretpostavke $s_\theta = t_x = t_y = 0$. U praksi, nema nikakvih prepreka da se postupak ponovi i više od tri puta pa uobičajena laboratorijska procedura umjeravanja podrazumijeva korištenje serije od devet pogleda.

Određivanje homografija

Neka $\mathbf{q}^u \in \Upsilon$ i $\mathbf{q} \in \pi$ označavaju položaj mjerne značajke u k.s. uzorka, odnosno položaj detektirane značajke u k.s. slike. Homografiju $\mathbf{H} : \Upsilon \to \pi$ umjeravamo uz isti uvjet kao i matricu $\mathbf{P}$ kod postupka koji pretpostavlja 3D mjerni uzorak, a opisan je u 4.3.1:

$$\lambda\mathbf{q} = \mathbf{H}\cdot\mathbf{q}^u \Longleftrightarrow \mathbf{q}\times(\mathbf{H}\cdot\mathbf{q}^u) = \mathbf{0}. \tag{4.61}$$

Za svaku uspješno identificiranu i lokaliziranu mjernu značajku uzorka, dobivaju se dvije linearne nezavisne jednadžbe u parametrima matrice homografije:

$$\begin{aligned}\mathbf{q}\times(\mathbf{H}\mathbf{q}^u) &= \left[\ q_x, q_y, 1\ \right]\times\left[\ \mathbf{H}_{1:}\mathbf{q}^u, \mathbf{H}_{2:}\mathbf{q}^u, \mathbf{H}_{3:}\mathbf{q}^u\ \right]\\ &= \begin{bmatrix}\mathbf{0}_3^T & -\mathbf{q}^{uT} & y_q\mathbf{q}^{uT}\\ \mathbf{q}^{uT} & \mathbf{0}_3^T & -x_q\mathbf{q}^{uT}\\ -y_q\mathbf{q}^{uT} & x_q\mathbf{q}^{uT} & \mathbf{0}_3^T\end{bmatrix}\cdot\begin{bmatrix}\mathbf{H}_{1:}^T\\ \mathbf{H}_{2:}^T\\ \mathbf{H}_{3:}^T\end{bmatrix} = \mathbf{0}_9\end{aligned} \tag{4.62}$$

Za n detektiranih značajki, dobiva se sustav $\mathbf{A}_{2n\times 9}\cdot\mathbf{h}_9 = 0_9$. Četiri značajke su stoga dovoljne za jednoznačno rješenje, dok je u praksi potrebno imati stotinjak značajki ili više, uz rješenje koje minimizira ukupnu kvadratnu pogrešku. Radi poboljšanja numeričke točnosti, i originalne i detektirane značajke se prije određivanja homografije mogu prevesti u k.s. gdje bi bile ravnomjerno raspoređene [zhang00]. Ovako dobiveno rješenje potrebno je poboljšati nelinearnom optimizacijom, iz istih razloga kao i kod umjeravanja 3D uzorkom.

Postavljanje ograničenja na matricu K

Veza između stupaca matrice homografije i parametara perspektivnog modela kamere slijedi iz jednadžbi (4.59) i (4.60):

$$\mathbf{H} = \left[\ \mathbf{H}_{:1}\ \ \mathbf{H}_{:2}\ \ \mathbf{H}_{:3}\ \right] = \left[\ \lambda\mathbf{K}\mathbf{R}_{:1}\ \ \lambda\mathbf{K}\mathbf{R}_{:2}\ \ \lambda\mathbf{K}\mathbf{t}\ \right] \tag{4.63}$$

Odatle slijede izrazi za stupce rotacijske matrice, koji će biti korišteni u nastavku:

$$\mathbf{R}_{:i} = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{H}_{:i}, i = 1,2 \tag{4.64}$$

Korištenjem konvencije $\mathbf{K}^{-T} = (\mathbf{K}^{-1})^T$, uvedimo pokratu:

$$\mathbf{B} = \mu\cdot\mathbf{K}^{-T}\mathbf{K}^{-1} \tag{4.65}$$

Matrica $\mathbf{B}$ je simetrična i pozitivno definitna, pa se matrica $\mathbf{K}^{-1}$ dobiva njenom dekompozicijom po Choleskom. Ograničenja na matricu unutrašnjih parametara sada slijede iz ortonormalnosti stupaca rotacijske matrice:

$$\begin{aligned} \mathbf{R}_{:1}^T\mathbf{R}_{:2} = 0 &\equiv \mathbf{H}_{:1}^T\mathbf{B}\mathbf{H}_{:2} = 0 \\ \mathbf{R}_{:1}^T\mathbf{R}_{:1} = \mathbf{R}_{:2}^T\mathbf{R}_{:2} &\equiv \mathbf{H}_{:1}^T\mathbf{B}\mathbf{H}_{:1} = \mathbf{H}_{:2}^T\mathbf{B}\mathbf{H}_{:2}. \end{aligned} \tag{4.66}$$

Sada je potrebno organizirati dobivena ograničenja u linearni sustav. Neka je:

$$\mathbf{b} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \mathbf{B}_{13} & \mathbf{B}_{22} & \mathbf{B}_{23} & \mathbf{B}_{33} \end{bmatrix}^T \tag{4.67}$$

Sada se osnovni građevni element ograničenja $\mathbf{H}_{:i}^T\mathbf{B}\mathbf{H}_{:j}$ može raspisati po pojedinim elementima matrice $\mathbf{B}$, tako da se dobije linearni sustav jednadžbi:

$$\mathbf{H}_{:i}^T\mathbf{B}\mathbf{H}_{:j} = \begin{bmatrix} \mathbf{H}_{1i}\mathbf{H}_{1j} \\ \mathbf{H}_{1i}\mathbf{H}_{2j} + \mathbf{H}_{2i}\mathbf{H}_{1j} \\ \mathbf{H}_{1i}\mathbf{H}_{3j} + \mathbf{H}_{3i}\mathbf{H}_{1j} \\ \mathbf{H}_{2i}\mathbf{H}_{2j} \\ \mathbf{H}_{2i}\mathbf{H}_{3j} + \mathbf{H}_{3i}\mathbf{H}_{2j} \\ \mathbf{H}_{3i}\mathbf{H}_{3j} \end{bmatrix} \cdot \mathbf{b} = \mathbf{v}_{ij} \cdot \mathbf{b} \tag{4.68}$$

Temeljna ograničenja svakog pogleda koja su zadana jednadžbama (4.66), stoga se mogu zapisati kao dvije homogene linearne jednadžbe u elementima $\mathbf{b}$:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \cdot \mathbf{b} = \mathbf{0} \tag{4.69}$$

Ako je umjereno n nezavisnih homografija, dobiva se sustav s $2n$ jednadžbi i šest nepoznanica od kojih je samo pet linearno nezavisnih:

$$\mathbf{V}_{2n\times 6} \cdot \mathbf{b} = \mathbf{0}_6. \tag{4.70}$$

Određivanje matrice K

Iz sustava opisanog jednadžbom (4.70), da se odrediti simetrična matrica $\mathbf{B}$. Jedini mogući problem u tom postupku je da numerički dobivena matrica nije pozitivno definitna što bi po fizičkim odnosima svakako trebala biti. Opisana komplikacija je više simptom nego problem jer upućuje na to da odabrani pogledi nisu dovoljno raznoliki pa je numerički šum uvjetovao velike devijacije u izračunatim elementima matrice. U tom slučaju postupak umjeravanja je najbolje prekinuti, i pokušati dobiti bolje rezultate raznolikijim skupom pogleda ili kvalitetnijom izradom mjernog uzorka. Konkretno, eksperimentiranjem je utvrđeno da se problemi javljaju u sljedećim slučajevima:

- položaji ravnina uzorka u različitim pogledima su skoro paralelni,
- broj detektiranih značajki mjernog uzorka je relativno malen,
- uzorak je postavljen daleko od kamere pa zauzima relativno mali dio vidnog polja.

Vidi se da su, na sreću, situacije u kojima se problem javlja upravo suprotne scenariju za dobro umjeravanje kamere.

Ako opisanog problema nema, dekompozicijom po Choleskom moguće je naći $\mathbf{K}^{-1}$ tako da vrijedi:

$$\mathbf{B} = \mu \cdot \mathbf{K}^{-T}\mathbf{K}^{-1} \tag{4.71}$$

Pri tome je nepoznati parametar μ moguće odrediti na temelju zahtjeva

$$\mathbf{K}^{-1}_{3,3} = 1. \tag{4.72}$$

Konačno, matrica unutrašnjih parametara linearnog modela dobiva se trivijalnim invertiranjem gornje trokutne matrice $\mathbf{K}^{-1}$.

Određivanje matrica pomaka ravnine uzorka u pojedinim pogledima

Kad je matrica $\mathbf{K}$ poznata, mogu se odrediti svi elementi pomaka ravnine uzorka u svakom od pogleda. Prva dva stupca rotacijske matrice te translacijski vektor mogu se odrediti prema jednadžbi (4.64):

$$\begin{aligned} \mathbf{R}_{:i} &= \frac{1}{\lambda} \cdot \mathbf{K}^{-1}\mathbf{H}_{:i}, i = 1, 2 \\ \mathbf{t} &= \frac{1}{\lambda} \cdot \mathbf{K}^{-1}\mathbf{H}_{:3} \end{aligned} \tag{4.73}$$

Treći stupac rotacijske matrice, te nepoznati parametar λ, mogu se odrediti na temelju ortogonalnosti stupaca rotacijske matrice:

$$\begin{aligned} \mathbf{R}_{:3} &= \mathbf{R}_{:1} \times \mathbf{R}_{:2} \\ \lambda &= \frac{1}{2} \sum_{i=1,2} \|\mathbf{K}^{-1}\mathbf{H}_{:i}\|. \end{aligned} \tag{4.74}$$

Izvedbeni detalji postupka nelinearne optimizacije

Postupak nelinearne optimizacije se provodi sukladno općenitom postupku opisanom u 4.3.1. Sa stanovišta postupka optimizacije, relevantne su sljedeće varijable (n_p označava broj pogleda, a n_{qi} broj detektiranih značajki u i–tom pogledu):

- nepromjenljive liste uparenih značajki $\{\mathbf{q}^u_{ij}\}, \{\mathbf{q}_{ij}\}, i = 1..n_p, j = 1..n_{qi}$;
- parametri Euklidskih pomaka $\{\mathbf{R}^P_i\}$, $\{\mathbf{t}_i\}$, $i = 1..n_p$;
- intrinsični parametri: $\mathbf{K}$, $(\mathbf{k}^{ud}_1, \mathbf{k}^{ud}_2)$, $(\mathbf{k}^{du}_1, \mathbf{k}^{du}_2)$,

Postupak minimizira ukupno 9 unutrašnjih te $n_p \cdot 6$ vanjskih parametara modela kamere. Ciljna funkcija postupka $\text{fcn} : \mathbb{R}^{9+6n_p} \rightarrow \mathbb{R}^{4n_p\bar{n}_t}$ na ulazu prima trenutne vrijednosti za $n_p \cdot 6$ parametara koje treba optimizirati, a na izlazu daje dvosmjernu projekcijsku pogrešku (dva puta po dvije koordinatne vrijednosti) u svakoj detektiranoj značajki svakog pogleda.

$$\text{fcn}(\{\mathbf{R}_i\}, \{\mathbf{t}_i\}, \mathbf{K}, \{\mathbf{k}^k_l\}) = \{\delta^d_{ij}, \delta^u_{ij}\}, i = 1..n_p, j = 1..n_{qi} \tag{4.75}$$

Dvosmjerna projekcijska pogreška se računa u cilju istovremenog određivanja svih parametara [tamaki02] modela nelinearnog radijalnog izobličenja koji je opisan u 4.2.3. Oba elementa funkcije cilja δ^d_{ij} i δ^u_{ij} određuju projekcijsku grešku u slikovnoj ravnini. Faktori radijalnog izobličenja $k^{ud}_{1,2}$ primjenjuju se pri računanju uobičajene projekcijske greške δ^d_{ij} u izobličenoj slici, koja odgovara udaljenosti između detektirane značajke i točke u koju se originalna značajka uzorka preslikava u skladu s važećim modelom. Suprotno, faktori korekcije radijalnog izobličenja $k^{du}_{1,2}$ utječu na rezultat funkcije cilja pri računanju projekcijske greške δ^u_{ij} u ispravljenim slikovnim koordinatama, koja odgovara udaljenosti između ispravljenog položaja detektirane značajke i točke u koju se originalna značajka uzorka preslikava

u skladu s linearnim modelom. Oblici pojedinih komponenti funkcije cilja su dani sljedećim jednadžbama:

$$\begin{aligned} \delta_{ij}^{d} &= d(\mathbf{K} \cdot \mathbf{f}_{ud}(\mathbf{S} \cdot \mathbf{M}_i' \cdot \mathbf{q}_{ij}^{u'}), \mathbf{q}_{ij}), \\ \delta_{ij}^{u} &= d(\mathbf{K} \cdot \mathbf{S} \cdot \mathbf{M}_i' \cdot \mathbf{q}_{ij}^{u'}, \mathbf{K} \cdot \mathbf{f}_{du}(\mathbf{K}^{-1} \cdot \mathbf{q}_{ij})). \end{aligned} \tag{4.76}$$

Pri tome je $\mathbf{M}_i'$ matrica koja se sastoji od prvog, drugog i četvrtog stupca matrice Euklidskog pomaka i–tog pogleda, dok je $\mathbf{q}_{ij}^{u'}$ korigirani položaj kruga uzorka koji odgovara težištu regije u slici dobivene projekcijom. Korekcija se obavlja prema sljedećoj formuli, u skladu s razmatranjima iz 4.3.1:

$$\mathbf{q}_{ij}^{u'} = \mathbf{q}_{ij}^{u} - \varepsilon_H(\mathbf{q}_{ij}^{u}, \mathbf{K} \cdot \mathbf{S} \cdot \mathbf{M}_i', r). \tag{4.77}$$

Zaključak

Funkcija cilja vraća vektor dvosmjernih projekcijskih pogrešaka δ_k koje odgovaraju značajkama uparenim u svim pogledima, u ovisnosti o unutrašnjim i vanjskim parametrima kamere:

$$\text{fcn}(\{\mathbf{R}_i\}, \{\mathbf{t}_i\}, \mathbf{K}, \mathbf{k}_l^{du}, \mathbf{k}_l^{ud}) = [\delta_1, \ldots, \delta_{n_q}]^T, \text{ uz } n_q = 2 \cdot \sum_{i=1}^{n_p} n_{qi}. \tag{4.78}$$

Korišteni model obuhvaća linearne parametre $(\{\mathbf{R}_i\}, \{\mathbf{t}_i\}, \mathbf{K})$, parametre radijalnog izobličenja projiciranih točaka mjernog uzorka $\{\mathbf{k}_l^{du}\}$, te odgovarajuće parametre korekcije točaka detektiranih u slikovnoj ravnini $\{\mathbf{k}_l^{ud}\}$. Optimizacijski postupak pronalazi optimalne vrijednosti parametara kamere, u smislu minimizacije kvadratne norme funkcije cilja.

Kvaliteta dobivenog rješenja može se ocijeniti analizom dobivene projekcijske pogreške, za veliki broj pogleda na kvalitetno izrađeni mjerni uzorak. Prosječna kutna pogreška umjeravanja se na temelju dobivenog parametra linearnog modela s_x može ocijeniti kao:

$$\theta_\mu = \text{arc tg}(\frac{\bar{\delta}_{\text{fcn}}}{s_x}), \text{ uz } \bar{\delta}_{\text{fcn}} = \frac{1}{n_q} \sum_{k=1}^{n_q} \|\delta_k\|_2, \tag{4.79}$$

gdje $\bar{\delta}_{\text{fcn}}$ označava prosječnu projekcijsku pogrešku umjerenog modela. Više od prosječne pogreške, često će biti interesantna najveća kutna pogreška koja se ocjenjuje kao:

$$\theta_{\max} = \text{arc tg}(\frac{\delta_{\text{fcn}}^{(\max)}}{s_x}), \text{ uz } \delta_{\text{fcn}}^{(\max)} = \|\delta_k\|_2 : \|\delta_k\|_2 \geq \|\delta_l\|_2 \ \forall l \tag{4.80}$$

gdje je $\delta_{\text{fcn}}^{(\max)}$ maksimalna pojedinačna projekcijska pogreška po svim uparenim točkama. Eksperimentalni rezultati su pokazali da gradijentni postupak pronalazi minimum funkcije cilja vrlo efikasno i pouzdano. Performansa eksperimentalnog kalibracijskog programa je omogućavala izvršavanje u stvarnom vremenu, a usko grlo je naravno bio postupak pronalaženja značajki mjernog uzorka u pribavljenim slikama.

4.3.3 Pronalaženje značajki mjernog uzorka

U ovom pododjeljku se u grubim crtama opisuje postupak pronalaženja značajki mjernog uzorka u pribavljenim sivim slikama. Pored samog pronalaženja značajki, potrebna je i njihova identifikacija, odnosno uparivanje svake pronađene značajke u slici s odgovarajućom originalnom značajkom u ravnini mjernog uzorka. Postupak uparivanja treba biti čim robustniji kako bi se omogućilo nesmetano eksperimentiranje i eventualna naknadna upotreba

postupka u manje kontroliranim uvjetima. Odabrana je jednostavna struktura mjernog uzorka, u obliku pravokutnog polja crnih krugova na bijeloj pozadini. Parametri uzorka stoga su međusobne udaljenosti redaka i stupaca, d_x odnosno d_y, te radijus krugova r koji se mogu zadati u proizvoljnim mjernim jedinicama. Obično je najpraktičnije odabrati uzorak s kvadratnom strukturom $d_x = d_y$.

Dva su temeljna konteksta upotrebe algoritma. Lakši posao odnosno bolji uvjeti za ispravnu detekciju mogu se očekivati u kontekstu umjeravanja unutrašnjih parametara kamere. U tom kontekstu, uzorak bi uvijek trebao zauzimati najveći do dio vidnog polja kamere, pa je mala opasnost od fiksacije krivog uzorka. U skladu s postupkom umjeravanja opisanim u 4.3.2, nije bitno znati sve parametre rasporeda značajki, nego će dovoljan biti omjer d_y/d_x, a vanjski parametri kamere se mogu izraziti uz mjernu jedinicu $d_x = 1$. U kontekstu određivanja vanjskih parametara kamere, posao će biti relativno teži jer uzorak može biti značajno udaljen od kamere. Nadalje, u cilju jednoznačnosti umjerenih parametara, značajku koja definira ishodište k.s. uzorka biti će potrebno specijalno označiti, što je u izvedbi osigurano upisivanjem bijelog kružića unutar željenog crnog kruga. Konačno, u cilju interpretacije položaja u vanjskom referentnom sustavu, biti će potrebno znati točne vrijednosti parametara uzorka u referentnim jedinicama, npr. centimetrima.

Izlučivanje značajki

Pribavljena slika se prvo binarizira uz korištenje praga prema sljedećoj formuli:

$$\mathrm{th} = v_{min} + c \cdot (v_{max} - v_{min}), \tag{4.81}$$

gdje su v_{max} i v_{min} ekstremne vrijednosti sive razine u pribavljenoj slici, a c je slobodni parametar koji ovisi o korištenoj kameri. Nakon toga se u slici izlučuju povezane nakupine za čije slikovne elemente vrijedi $v_p < \mathrm{th}$. Mnoge od tih nakupina odgovaraju pojedinim crnim krugovima mjernog uzorka, ali je vrlo teško oblikovati općeniti diskriminacijski kriterij.

Grupiranje značajki

Temeljna zamisao postupka je korištenje simetrije mjernog uzorka kao kriterija detekcije. Uspješno grupiranje nakupina u pravokutno polje stoga je temeljni kriterij odluke da li neka nakupina odgovara značajki ili ne. Grupiranje nakupina se temelji na pretpostavci da se svaki redak odnosno stupac može inkrementalno rekonstruirati, korak po korak, zbog svojstva da je vektor pomaka do sljedeće značajke retka kod susjednih značajki gotovo konstantan. Razlog tome je relativna blizina susjednih elemenata polja koje čini mjerni uzorak, zbog čega se homografija lokalno može opisati kao afina transformacija.

Za korišteni postupak grupiranja ključan je pojam *baze*, odnosno jedinstvene uređene četvorke indeksa nakupina koja se pridružuje svakoj nakupini. Svaka nakupina n_i ujedno je i prvi element svoje baze koja onda ima oblik (n_i, n_j, n_k, n_l). Ostali elementi nakupine se određuju prema sljedećem algoritmu:

- n_j se bira kao nakupina koja je najbliža n_i,
- n_k se bira iz preostalog skupa nakupina kao najbliža n_i, pod uvjetom da nije približno kolinearna s n_i i n_j,
- n_l se bira iz preostalog skupa nakupina kao najbliža n_i, pod uvjetom da nije približno kolinearna ni s n_i i n_j, ni s n_i i n_k.

Svaka baza generira jedinstveno grupiranje nakupina u pravokutno polje prema sljedećem postupku:

- generira se prvi rubni redak uzorka r_1 kao polje indeksa nakupina počevši od nakupine n_i prema n_j i dalje;
- generira se drugi rubni redak uzorka r_2 kao polje indeksa nakupina počevši od nakupine n_k prema n_l i dalje;
- generiraju se stupci mjernog uzorka, počevši od odgovarajućih elemenata dvaju rubnih redaka: npr. za i–ti stupac, počinje se od nakupine r_{1i} prema r_{2i} i dalje.

Kao konačna baza za grupiranje nakupina, odabire se naravno ona koja generira "matricu" s najviše elemenata mjernog uzorka. U idealnom slučaju, postoje četiri takve baze, za svaki kutni element po jedna, a odabir je proizvoljan.

Završne napomene

Pokazuje se da ovakva jednostavna shema pokazuje iznenađujuće dobre rezultate i kada uvjeti pribavljanja slike nisu strogo kontrolirani. Jedini problem koji može omesti ispravno pronalaženje uzorka se javlja kada se rubnom elementu uzorka pridruži "kriva" baza uslijed blizine nekog crnog objekta u pozadini scene. Ovo ne mora biti kritično jer postoji redundancija: za ispravan rad algoritma je dovoljno da baza bude "ispravno" određena u barem jednoj od kutnih nakupina uzorka. Ovakav problem se može u potpunosti izbjeći oblikovanjem uzorka s marginom koja je veća od proreda redaka odnosno stupaca.

Ako je u nekoj od kutnih nakupina detektiran bijeli krug, pojedinim elementima uzorka potrebno je dodijeliti koordinate koje bi imali u desno orijentiranom k.s. uzorka koji ima ishodište u toj nakupini. Inače treba paziti da orijentacija sustava bude konzistentna s orijentacijom koja se očekuje u ostatku programskog sustava. U programskoj izvedbi, dileme se razrješuju tako da se uvijek postavlja k.s. uzorka koji je desno orijentiran.

4.4 Umjeravanje i primjena vanjskih parametara kamere

Vanjski parametri kamere su potrebni ukoliko je rezultate obrade slike potrebno interpretirati u nekom vanjskom referentnom koordinatnom sustavu, što je gotovo uvijek slučaj. Vanjski parametri su od posebnog značaja kod porazdijeljenih sustava koji se sastoje od većeg broja promatrača, jer je precizna interpretacija pojedinačnih pogleda u referentnom sustavu ključna za njihovu kvalitetnu prostornu integraciju. Vanjski parametri tim više dolaze do izražaja kad je kamera pričvršćena za pokretno postolje, jer je temeljni model tada potrebno proširiti vremenski promjenljivom transformacijom iz k.s. promatrača u k.s. kamere, čak i kod primjena sa samo jednim promatračem.

4.4.1 Određivanje vanjskih parametara u sustavu s više promatrača

Neka je zadan porazdijeljeni sustav s više agenata promatrača koji su opremljeni kamerama s poznatim unutrašnjim parametrima Promatrači analiziraju pribavljene slike te međusobno komuniciraju u cilju izgradnje zajedničkog prikaza scene. Pri izgradnji zajedničkog prikaza odnosno prostornoj integraciji, javlja se već najavljeni problem interpretacije pojedinačnih mjerenja u zajedničkom referentnom koordinatnom sustavu. Drugim riječima, problem se

svodi na pitanje "gdje su kamere?", odnosno na određivanje matrica Euklidskih transformacija iz k.s. svijeta u k.s pojedinih promatrača:

$$\mathbf{M}_i = ?,\ i = 1, 2, \ldots, n_p. \tag{4.82}$$

Manualni pristupi rješavanju tog problema nisu prihvatljivi jer podrazumijevaju dugotrajno i neprecizno mjerenje udaljenosti od kućišta pojedinih kamera do ishodišta referentnog k.s. svijeta. Dodatne poteškoće pri tome postavlja nemogućnost određivanja pomaka žarišta leće u odnosu na neki izraženi k.s. kućišta kamere. Problem postaje još izraženiji ako se uvaži činjenica da je kalibraciju potrebno obaviti prije *svakog* eksperimenta, pogotovo u laboratorijima s većim brojem korisnika.

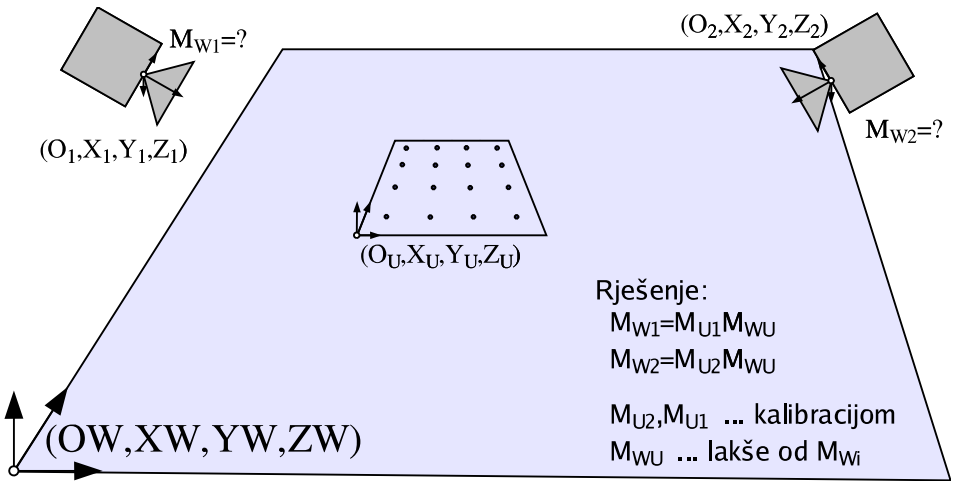
Kratki opis prethodnih rješenja

Iako literatura ne obiluje radovima s područja umjeravanja vanjskih parametara skupa kamera, ipak su pronađena tri rada koji analiziraju problem s različitih stanovišta. Lee et al. [stein00] opisuju automatsku metodu kalibracije više kamera koje promatraju istu urbanu scenu iz ptičje perspektive. Metoda se temelji na korespondenciji trajektorija pokretnih objekata, pa su joj općenitost i robustnost glavne prednosti, dok glavni nedostatak predstavlja dugotrajno izvođenje Collins i Tsin [collins99] su predložili metodu koja se temelji na lociranju svake aktivne kamere iz skupa trijangulacijom, na temelju predefiniranog skupa referentnih objekata čiji položaj je precizno utvrđen uz pomoć specijalne mjerne opreme. Prednost metode je velika preciznost, dok su nedostatci neriješena detekcija referentnih objekata u slici, te potreba za utvrđivanjem njihovog točnog položaja.

Najzanimljivija je metoda Olsena i Hoovera [olsen01], koja se temelji na postavljanju mjernih uzoraka na podlogu scene, i to u strateškim točkama koje su vidljive iz točaka gledanja većeg broja promatrača. Uzorci su vrlo slični uzorcima koji se inače koriste pri umjeravanju unutrašnjih parametara kamere, ali su dimenzijama znatno veći kako bi bili vidljivi i iz udaljenih dijelova scene. Uzorci su pričvršćeni za podlogu pa je njihov precizan položaj potrebno odrediti samo jednom. U jednostavnijim prostorijama dovoljan je samo jedan takav uzorak pa se ishodište referentnog k.s. svijeta može poravnati s njegovim položajem. Prednost ovakvog pristupa je što omogućava razmještanje kamera po sceni bez potrebe za obavljanjem ikakvih mjerenja jer se kalibracija u odnosu na vidljivi mjerni uzorak obavlja automatski. Opisani pristup umjeravanju vanjskih parametara skupa kamera ilustriran je na sl. 4.7.

Određivanje položaja kamere u odnosu na mjerni uzorak

Tijek temeljnog postupka umjeravanja vanjskih parametara je donekle sličan umjeravanju unutrašnjih parametara korištenjem više pogleda na mjerni uzorak na način opisan u 4.3.2, ali je znatno izravniji i jednostavniji. Pod pretpostavkom da su unutrašnji parametri kamere poznati odnosno prethodno umjereni, vanjske parametre je moguće odrediti samo na temelju poznatih koordinata mjernih značajki u referentnom k.s. svijeta. Tehnički, postupak se odvija u dva koraka: prvo se odredi homografija na temelju uparenih značajki slike i uzorka, kao rješenje homogenog linearnog sustava 4.62. U drugom koraku, rotacijska matrica $\mathbf{R}$ i translacijski vektor $\mathbf{t}$ određuju se na temelju umjerene homografije $\mathbf{H}$ i prethodno poznate matrice unutrašnjih parametara $\mathbf{K}$, prema jednadžbama (4.73) i (4.74).



Slika 4.7: Određivanje položaja skupa kamera na temelju zajedničkog mjernog uzorka.

4.4.2 Parametri postolja s dva stupnja slobode

Razmatra se upravljivo postolje s dva rotacijska stupnja slobode, zakretom γ_z i nagibom γ_n, prema sl. 3.2. Utjecaj kretanja kamere na geometrijske odnose pri stvaranju slike može se modelirati uvođenjem novog k.s. promatrača koji se dosada nije razmatrao jer je uvijek bio u istom odnosu s k.s. kamere. Sada je opći perspektivni model dan jednadžbom (4.36) potrebno doraditi na način da se ukupna matrica Euklidskog pomaka iz k.s. svijera u k.s. kamere $\mathbf{M}$ izrazi umnoškom dvaju matrica:

$$\mathbf{q} = \mathbf{KSM} \cdot \mathbf{Q} = \mathbf{KS}[\mathbf{M}_{OC}(\gamma_z, \gamma_n)\mathbf{M}_{WO}] \cdot \mathbf{Q} \tag{4.83}$$

Nove matrice pomaka imaju sljedeća značenja:

- $\mathbf{M}_{WO}$ – stalan pomak iz k.s. svijeta u k.s. promatrača odnosno postolja;
- $\mathbf{M}_{OC}$ – pomak iz k.s. promatrača u k.s. kamere koji izravno ovisi o parametrima postolja (γ_z, γ_n).

Korisno je uvesti konvenciju da se k.s. promatrača i kamere poklapaju za $\gamma_z = \gamma_n = 0$:

$$\mathbf{M}_{OC}(0, 0) = \mathbf{I} \tag{4.84}$$

U idealnom slučaju, kad bi leća kamere bila montirana točno u presjecištu osi zakreta i nagiba, pomak kamere u odnosu na promatrača bi se mogao opisati čistom rotacijom:

$$\mathbf{M}_{OC} = \begin{bmatrix} \mathbf{R}_{OC} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{4.85}$$

Matrica rotacije bi se mogla dekomponirati po osima stupnjeva slobode postolja, vodeći računa o redoslijedu elementarnih rotacija:

$$\mathbf{R}_{OC}(\gamma_z, \gamma_n) = \mathbf{R}_z(0)\mathbf{R}_x(\gamma_n)\mathbf{R}_y(\gamma_z). \tag{4.86}$$

Umjeravanje vanjskih parametara upravljivog postolja

Prije daljnjih razmatranja, odmah valja napomenuti da odvojena kalibracija pokretnog postolja neovisno o pričvršćenoj kameri nije najbolja ideja. Položaj kamere na postolju se u

većini slučajeva definira *rukom*, te kasnije učvršćuje vijkom, pa je svako postavljanje kamere vezano s jedinstvenim i neponovljivim pomakom kamere u odnosu na pokretno postolje. Stoga je precizno umjeravanje para postolje–kamera najdalji domet kojem se možemo nadati, uz ograničenje da životni vijek dobivenih parametara traje do prvog skidanja kamere s postolja.

Kao i za kameru, i za pokretno postolje možemo reći da ima intrinsične $\mathbf{M}_{OC}$ i vanjske parametre $\mathbf{M}_{WO}$. Pri tome intrinsični parametri ovise o parametrima stupnjeva slobode postolja, dok vanjski parametri ovise isključivo o položaju i orijentaciji postolja u referentnom k.s. svijeta. Nastavljajući analogiju s umjeravanjem parametara kamere, prirodno je prvo umjeriti unutrašnje parametre postolja, a tek onda vanjske. Problem umjeravanja vanjskih parametara postolja potpuno je ekvivalentan umjeravanju vanjskih parametara kamere. Ako su poznati unutrašnji parametri postolja $\mathbf{M}_{OC}$, vanjski parametri postolja $\mathbf{M}_{WO}$ se mogu odrediti pod uvjetom da se vanjski parametri kamere M za neki smjer gledanja (γ_z, γ_n) odrede npr. metodom iz prethodnog pododjeljka:

$$\begin{aligned} \mathbf{M} &= \mathbf{M}_{OC}(\gamma_z, \gamma_n)\mathbf{M}_{WO} \Longrightarrow \\ & \mathbf{M}_{WO} = \mathbf{M}_{OC}(\gamma_z, \gamma_n)^{-1}\mathbf{M} \end{aligned} \tag{4.87}$$

Umjeravanje unutrašnjih parametara upravljivog postolja

Unutrašnji parametri postolja se mogu umjeriti iterativnim određivanjem relativnih pomaka između položaja dobivenih za različite vrijednosti kutova γ_z i γ_n. Postupak bi mogao započeti postavljanjem mjernog uzorka u trenutnom smjeru gledanja kamere, tako da zauzima 1/4 vidnog polja. Sada je moguće kalibrirati relativne pomake postolja u okolini trenutnog položaja (γ_{z0}, γ_{n0}), za sve ($\Delta\gamma_z$, $\Delta\gamma_n$) za koje je uzorak vidljiv. Opisani temeljni postupak potrebno je ponoviti za "razne" položaje kalibracijskog uzorka, te iz dobivenih podataka odrediti kalibraciju $\mathbf{M}_{OC}(\gamma_z, \gamma_n)$. Postupak je moguće nastaviti na temelju dva različita izvedbena pristupa.

Izvedbeni pristup "gruba sila"

Temeljni postupak umjeravanja se ponavlja sve dok umjerene okoline ne prekriju ukupno radno područje postolja. Konačno, $\mathbf{M}_{OC}(\gamma_z, \gamma_n)$ se može jedinstveno odrediti uz već uvedenu pretpostavku $\mathbf{M}_{OC}(0, 0) = \mathbf{I}$. Prednost pristupa je konceptualna jednostavnost, dok su nedostatci dugotrajnost te potreba za postavljanjem mjernog uzorka u vrlo velikom broju točaka prostora. Ovaj problem bi se mogao u potpunosti izbjeći u scenariju u kojem bi se pomak kamere određivao bez mjernog uzorka, samo na temelju uparivanja točaka susjednih slika, u skladu s principima samoumjeravanja koji su skicirani u 4.3.

Izvedbeni pristup "precizno postolje"

Ovaj pristup se može odabrati ukoliko mehanička preciznost postolja zadovoljava željene kriterije kvalitete kalibracije, što obično jest slučaj. U tom slučaju se $\mathbf{M}_{OC}$ može predstaviti kao kombinacija konstantne translacije koja definira pomak leće pričvršćene kamere u odnosu na presjecište rotacijskih osi postolja, i idealne rotacije:

$$\mathbf{M}_{OC}(\gamma_z, \gamma_n) = \begin{bmatrix} \mathbf{I} & \mathbf{t}_{OC} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{OC}(\gamma_z, \gamma_n) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4.88}$$

Temeljni postupak u tom pristupu mogao bi se primijeniti u samo nekoliko pogleda, koji bi bili dovoljni za postavljanje linearnog sustava s viškom ograničenja u parametrima $\mathbf{t}_{OC}$.

Matrica $\mathbf{R}_{OC}$ pri tome bi se mogla računati prema formuli (4.86). Jedinstveno rješenje bi se moglo dobiti uz već uvedenu pretpostavku $\mathbf{t}_{OC}(0,0) = \mathbf{0}$. Prednost ovakvog pristupa je jednostavnost izvedbe, dok je nedostatak ovisnost postupka umjeravanja o mehaničkoj preciznosti postolja.

4.4.3 3D položaj objekta na poznatoj ravnini

Na kraju poglavlja, izvest će se jednostavna posljedica dosadašnjih razmatranja, da je objektu čije je gibanje ograničeno poznatom ravninom Υ moguće odrediti 3D položaj na temelju samo jedne slike dobivene umjerenom kamerom. Neka je ravnina Υ dana ishodištem $\mathbf{u_o}$ te koordinatnim vektorima $\mathbf{u_x}$ i $\mathbf{u_y}$. Tada prema (4.49) i (4.50) postoji homografija između točaka uzorka $\mathbf{q}^u \in \Upsilon$ i točaka slikovne ravnine $\mathbf{q} \in \pi$ po formuli:

$$\mathbf{q} = \mathbf{P} \cdot \begin{bmatrix} \mathbf{u_x} & \mathbf{u_y} & \mathbf{u_o} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{q}^u = \mathbf{P} \cdot \mathbf{U} \cdot \mathbf{q}^u = \mathbf{H} \cdot \mathbf{q}^u \tag{4.89}$$

Ako $\mathbf{H}_{3\times3}$ nije singularna, ravnina Υ nije okomita na ravninu slike, pa inverzno preslikavanje $\Upsilon \to \pi$ postoji i dano je izrazom:

$$\mathbf{q}^u = \mathbf{H}^{-1} \cdot \mathbf{q}, \ \forall \mathbf{q} \in \pi \tag{4.90}$$

Drugim riječima, 3D položaj objekta na poznatoj ravnini se na temelju njegove slike dobiva iznimno jednostavno, primjenom jednog množenja matrice 3×3, uz eventualno još dva dijeljenja prilikom prevođenja rezultata iz projekcijske u Euklidsku ravninu.

Poglavlje 5

Izvedba eksperimentalnog sustava

Tema ovog poglavlja su detalji izvedbe eksperimentalnog sustava za porazdijeljeno praćenje. U prvom dijelu poglavlja, navode se ključni elementi potrebni za razvoj i ispitivanje, te normalno funkcioniranje sustava. U skladu s tim, relevantni sklopovski resursi detaljno su specificirani u odjeljku 5.1, dok su odabrane programske platforme, biblioteke i razvojni alati opisani u odjeljku 5.2. U drugom dijelu poglavlja, opisuje se programska izvedba eksperimentalnog sustava. Izgrađeni sustav implementira jednorazinsku višeagentsku arhitekturu opisanu u odjeljku 3.2, pa se u daljnjem tekstu opisuju izvedbe sudionika u takvoj arhitekturi. Odjeljak 5.3 stoga sadrži detalje izvedbe agenta promatrača, dok je izvedba agenta koordinatora opisana u odjeljku 5.4.

5.1 Sklopovski resursi eksperimentalnog sustava

Eksperimentalni sustav je razvijan i ispitivan u laboratoriju Zavoda za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta. Računalne resurse sustava čine tri računala koja su povezana lokalnom mrežom Ethernet preko preklopnika propusnosti 100 Mbps. S obzirom na namjenu željenog sustava, najvažniji sklopovski resursi računala su optički sustav kamere koji se apstrahira njenim unutrašnjim parametrima, upravljivo postolje koje određuje vanjske parametre kamere te međusklop za pribavljanje slike. Unutrašnji parametri kamere te parametri upravljivog postolja su razmatrani u poglavlju 4, pa će u nastavku teksta detaljnije biti opisani samo izvedbeni detalji rada s međusklopom za pribavljanje slike. Većina specifičnog sklopovlja se koristi samo u okviru odgovarajućih agenata promatrača, jer koordinatori opažaju vanjski svijet te djeluju na njega isključivo preko sklopova za mrežnu komunikaciju koji imaju standardizirana sučelja.

5.1.1 Računalni resursi eksperimentalnog sustava

Eksperimentalni sustav se sastoji od tri IBM PC kompatibilna računala, a svako od njih je opremljeno upravljivom kamerom i odgovarajućim međusklopom za pribavljanje slike. Pored navedenog sklopovlja specifičnog za računalni vid, za performansu računala su značajni i procesor računala, operacijski sustav te prevodioc programskog jezika C++ koji je korišten za stvaranje izvršnog kôda. Važne infrastrukturne komponente za svako od triju računala eksplicitno su popisane u nastavku teksta.

1. Radna stanica Asus A7M266–D:

   - kamera: VIDERE DCAM
     (digitalna kamera na sabirnici IEEE 1394);

- upravljivo postolje: Directed Perception PTU–46–17.5
  (rs232, 38400 bps, poseban upravljački protokol);
- pribavljanje slike: Procomp 1394r
  (međusklop za sabirnicu IEEE 1394 sa standardnim programskim sučeljem);
- procesor: 2×AMD Athlon MP, 1.4 GHz, 530 spec cint2000;
- operacijski sustav: Linux 2.4.18;
- prevodioc: g++ v3.0;
- ime računala na IP mreži: dupli.

2. Radna stanica Compaq Evo W4000

   - kamera: Sony EVI–D31
     (analogna kamera s video izlazom po PAL standardu);
   - upravljivo postolje: integrirano u kameri
     (rs232, 9600 bps, upravljački protokol VISCA);
   - pribavljanje slike: Imagenation PXC 200 AL
     (digitalizator PAL signala s posebnim programskim sučeljem)
   - procesor: Pentium 4, 2 GHz, 710 spec cint2000;
   - operacijski sustav: MS Windows 2000;
   - prevodioc: g++ v3.2, MSVC v6.0;
   - ime računala na IP mreži: ocalinko.

3. Osobno računalo Matsonic 7127C

   - kamera: Sony EVI–D31
     (analogna kamera s video izlazom po PAL standardu);
   - upravljivo postolje: integrirano u kameri
     (rs232, 9600 bps, upravljački protokol VISCA);
   - pribavljanje slike: HaupPauge ImpactVCB
     (digitalizator PAL signala s programskim sučeljem MCI);
   - procesor: Intel Celeron, 700 MHz;
   - operacijski sustav: MS Windows 2000;
   - prevodioc: g++ v3.2, MSVC v6.0;
   - ime računala na IP mreži: misko.

Na svakom od triju računala izvršava se po jedan agent promatrač, dok se agent koordinator izvršava na radnoj stanici Asus A7M266–D. Zbog dostupnog drugog procesora, prisustvo koordinatora ne utječe na performansu promatrača koji se izvršava na istom računalu.

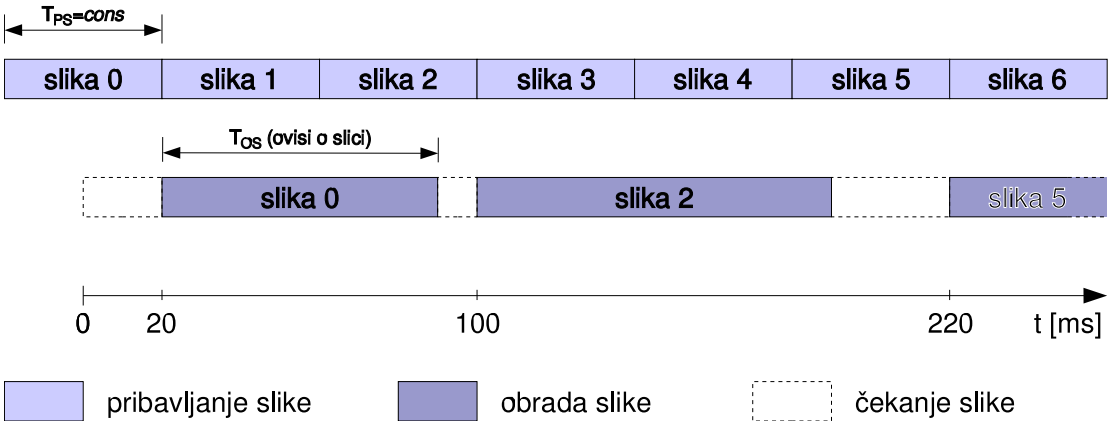
5.1.2 Međusklop za pribavljanje slike

Međusklop za pribavljanje slike prihvaća slijed slika iz priključene kamere i omogućava transparentni pristup najnovijoj slici iz slijeda. Pribavljanje slike je vrlo zahtjevna operacija jer zahtijeva vrlo veliku propusnost. Primjerice, evropski video standard PAL pretpostavlja sliku dimenzija 768×576 elemenata, uz učestalost osvježavanja od 25 Hz. Kod uobičajenih

formata boje RGB i YUV, svaki slikovni element se predstavlja trobajtnim kôdom, pa je za prijenos nekomprimiranog PAL signala potrebna propusnost od oko 250 Mbps što je blizu graničnoj propusnosti najčešće korištenog sabirničkog standarda PCI:

$$P_{PAL} = 768 \cdot 576 \cdot 24 \cdot 25. \tag{5.1}$$

U praksi se međutim pokazuje da kod primjena u stvarnom vremenu procesorsko vrijeme još više ograničava performansu. Stoga se obično eksperimentira sa dvostruko manjom rezolucijom 384×238 uz koju se tehnologijom iz 2002. godine postiže prihvatljiva performansa od oko 10 Hz uz prosječno trajanje obrade slike $T_{OS} \approx 100$ ms. Time se potrebna propusnost smanjuje četiri puta, na oko 60 Mbps što je znatno manje od ograničenja modernih sabirnica. Opisani vremenski odnosi su ilustrirani na sl. 5.1: slike dolaze brže nego što ih program stiže obraditi, pa se stoga barem svaka druga slika "preskoči".



Slika 5.1: Vremenski odnosi pribavljanja i obrade slike u stvarnom vremenu.

Trajanje pribavljanja slike T_{PS} dosta ovisi o vrsti veze između kamere i međusklopa, ali općenito nije zanemarivo i kod PAL signala iznosi 40 ms. Izvođenje programa nije sinkrono s ritmom stvaranja slika u kameri pa je i pored korištenja modernih DMA protokola potrebno čekati neko vrijeme da se dovrši pribavljanje slike koja bi odgovarala što skorijem stanju u sceni. Jednostavnim proračunom se dobiva da je moguće vrijeme čekanja unutar $(0, T_{PS})$ za međusklop s DMA prijenosom odnosno unutar $(T_{PS}, 2T_{PS})$ ukoliko međusklop ne podržava DMA prijenos. To čekanje je problematično jer uzrokuje smanjenje performanse koja je ionako na donjoj granici tolerancije.

Međutim, još veći problem predstavlja latencija djelovanja u odnosu na opažanje koja može značajno oslabiti rezultate praćenja, pogotovo kada je potrebno upravljati smjerom gledanja u stvarnom vremenu. Iz sl. 5.1 se vidi da su rezultati analize slike u najboljem slučaju dostupni tek $T_{PS} + T_{OS}$ nakon trenutka koji odgovara stvaranju slike u kameri! To znači da u sustavu računarskog vida s netrivijalnom obradom i konvencionalnom tehnologijom iz 2002. godine nije moguće reagirati na podražaj u vremenu kraćem od 100 ms. Štoviše, sve akcije u sustavu se poduzimaju na temelju stanja scene iz prošlosti pa je kod dinamičkih scena potrebno ekstrapolirati razvoj događaja za procijenjenu latenciju odnosno trajanje pribavljanja i obrade slike. Dodatan problem se javlja kod međusklopova s generičkim i eksperimentalnim programskim sučeljima u kojima nije predviđeno zahtijevati čekanje pribavljanja nove slike nego se isporučuje slika koja je posljednja prebačena u glavnu memoriju. Primjerice, kod takvih sučelja bi u sl. 5.1 nakon obrade slike 0 slijedila obrada slike 1, bez čekanja da se završi pribavljanje slike 2. Iako takvo ponašanje zapravo poboljšava performansu sustava, ono nije

poželjno jer povećava latenciju i, što je gore, onemogućava određivanje njenog iznosa i time povećava nesigurnost procjene *trenutnog* stanja u sceni.

Dva su temeljna načina za prijenos slike od kamere prema međusklopu za pribavljanje slike, i to analogni i digitalni. Do prije nekoliko godina, gotovo sve kamere su na izlazu davale analogni signal (kompozitni ili odvojeni svjetlina – boja), pa su odgovarajući međusklopovi morali digitalizirati signal u stvarnom vremenu što je uvjetovalo njihovu veliku cijenu i donekle usporavalo prodor industrijskih primjena računalnog vida. S obzirom da gotovo sve kamere koriste neku od digitalnih senzorskih tehnologija (npr. CCD ili CMOS), takav pristup se čini vrlo neefikasnim jer se digitalna slika u kameri pretvara u analogni signal, kojeg je u računalu ponovo potrebno digitalizirati. Pojavom digitalnih kamera, stiču se pretpostavke za kvalitetniju sliku po nižoj cijeni, ali te pretpostavke su do danas tek donekle ispunjene. Naime, tržište analognog video signala vjerojatno je još i danas jače od digitalnog pa se dobiva paradoksalna situacija da se zbog solidnije izvedbe, analognom kamerom u pravilu dobiva bolja slika nego digitalnom unatoč uzastopnim D/A i A/D konverzijama te šumu u signalnim vezama. Nadalje, zbog nepostojanja jedinstvenog standarda, prvi međusklopovi za pribavljanje digitalne slike su bili još skuplji od klasičnih digitalizatora. Tek u posljednjih nekoliko godina, pojavom standarda IEEE 1394, stekle su se pretpostavke za masovniju proizvodnju što je uzrokovalo da cijene digitalnih sučelja konačno postanu minimalne kakve bi i trebale biti po logici stvari.

Kod gotovo svih modernih međusklopova, prebacivanje slike u memorijski prostor korisničkog procesa se obavlja sofisticiranim metodama uz upotrebu sklopovlja DMA. Zbog složenosti izvedbe, ta funkcionalnost je često sadržana u programskom sučelju koje se isporučuje zajedno s međusklopom. Kod digitalnih međusklopova po standardu IEEE 1394, ta podrška je većim dijelom ugrađena u jezgru novijih operacijskih sustava, dok se manji dio obavlja bibliotekama koje ne moraju ovisiti o proizvođaču kamere. Postoje standardna sučelja i za analogne međusklopove, međutim ona su najčešće prilagođena primjenama u kojima dominira prikaz slike na ekranu računala a ne njena obrada. Stoga većina digitalizatora nude specifična programska sučelja koja nude značajno veću funkcionalnost od standardnih. Zbog otvorenosti izvedbe programskog sustava, zahtijeva se da program pribavlja sliku transparentno, neovisno o priključenom sklopovlju. Stoga je za svako od korištenih specifičnih sučelja potrebno napisati prilagodnik (*engl.* adapter) [gamma95] koji implementira metode jedinstvenog apstraktnog sučelja, kao što je opisano u dodatku B.

5.2 Programske platforme, biblioteke i razvojni alati

Programska izvedba eksperimentalnog sustava se sastoji od više donekle neovisnih elemenata. Najdalekosežnija odluka u tom kontekstu svakako je odabir programskog jezika jer svaki jezik implicira specifični stil koji najviše određuje konačni oblik programa. U praksi međutim, gotovo značajniji od samog jezika je skup dostupnih programskih biblioteka jer one omogućuju da se postupak razvoja fokusira na specifičnu funkcionalnost željenog sustava. Na proces razvoja u mnogome utječe i skup korištenih pomoćnih alata koji ubrzavaju svakodnevne zadatke kao što su arhiviranje verzija izvornog kôda, automatizacija postupka prevođenja ili traženje grešaka u kôdu. Konačno, na oblikovanje utječe i odabir ciljnih operacijskih sustava jer svaki od njih ima svoje više ili manje opravdane specifičnosti u terminima kojih je potrebno izraziti željenu funkcionalnost.

5.2.1 Ciljni operacijski sustavi

Zbog porazdijeljene prirode sustava, zahtijevana je mogućnost izvođenja pojedinih agenata na različitim operacijskim sustavima. Prvenstveno su razmatrani najrašireniji operacijski sustavi Linux i MS–Windows, dok se predviđa da bi podrška ostalih operacijskih sustava (npr, MacOS) trebala biti relativno lagan zadatak s obzirom na to da se postojeći standardi za biblioteke operacijskog sustava uvjerljivo najdosljednije ne poštuju kod MS–Windowsa. U skladu sa zahtjevima, programska izvedba agenata najvećim dijelom ne ovisi o ciljnom operacijskom sustavu jer je oblikovana u skladu s važećim standardom programskog jezika C++ [cxx98]. Ipak, izvedbe nekih komponenti [lakos96] nije bilo moguće napisati na prenosiv način, pa su za te komponente oblikovane zasebne izvedbe zajedničkog sučelja za svaki od ciljnih operacijskih sustava. Nepremostive prepreke pisanju prenosivog kôda su raznolike i uključuju:

- Ignoriranje sučelja programskih biblioteka operacijskog sustava u skladu s prethodnim standardima:
  - biblioteka za upravljanje s dretvama, odnosno višestrukim tokovima izvođenja programa
    `<pthread.h>` (POSIX, Linux) – `<windows.h>` (Microsoft);
  - sučelje neblokirajućih ulazno–izlaznih operacija koje su potrebne za robustnu komunikaciju preko serijske veze
    `<sys/select.h>` (POSIX, Linux) – `<windows.h>` (Microsoft);
  - biblioteka za precizno mjerenje vremena
    `<sys/time.h>` (System V, BSD, Linux) – `<windows.h>` (Microsoft);
  - biblioteka za rad s dinamičkim izvršnog modulima
    `<dlfcn.h>` (Solaris, Linux) – `<windows.h>` (Microsoft).
- Opravdane poteškoće u dizajniranju jedinstvenog sučelja za grafičke operacije zbog različitih temeljnih pretpostavki:
  - biblioteka za rad s prozorima i grafiku
    `<X11/Xlib.h>` (X Consortium, Linux) – `<windows.h>` (Microsoft);
  - međusklop za pribavljanje slike
    `<linux/videodev.h>` (Linux) – `<vfw.h>` (Microsoft);
- Vlasnički formati datoteka:
  - biblioteka za rad s komprimiranim slijedovima slika
    `<libavi.h>` (Linux) – `<vfw.h>` (Microsoft).

Ipak, komponenata čije izvedbe ovise o platformi u sustavu ima relativno malo: tek 14 od ukupno 216 komponenti nije prenosivo što je malo više od 6%.

5.2.2 Prevodioci programskog jezika C++

Eksperimentalni sustav je gotovo u potpunosti oblikovan u programskom jeziku C++ koji je trenutno, po subjektivnoj procjeni, najbolji jezik za oblikovanje primjena računarskog vida. Dobre osobine odabranog jezika uključuju mogućnost objektno orijentiranog i generičkog programiranja, prenosivost osigurana međunarodnim standardom, dostupnost velikog broja

biblioteka te efikasnost izvršnog kôda. Iako posljednje od navedenih svojstava u posljednje vrijeme kod velikog broja primjena gubi na važnosti, to najčešće nije slučaj i na području računarskog vida, posebno kod primjena u stvarnom vremenu koje su uži predmet interesa ovog rada[1].

Za prevođenje komponenti sustava korištena su dva prevodioca programskog jezika C++:

- Microsoft Visual C (MSVC) v6.0, za MS–Windows;
- g++ v3.0 za Linux odnosno v3.2 za MS–Windows.

Kod razvijanja velikih programskih sustava, općenito je vrlo korisno koristiti veći broj prevodioca izvornog kôda [meyers97]. Razlog je u tome što se kvaliteta prevodioca sastoji od više ortogonalnih kriterija, kao što su usaglašenost s važećim standardom jezika [cxx98], brzina prevođenja, podrška dijagnostičkih biblioteka i performansa generiranog izvršnog kôda. Standard programskog jezika C++ propisuje neke vrlo zahtjevne mogućnosti koje su ključne za razvoj podatkovnih struktura i odgovarajućih pristupnih metoda koje su i općenite i robustne na pogreške i iznimno brze. Nažalost, cjelokupni standard implementiraju samo neki prevodioci koji su dosta skupi i stoga nedostupni. Stoga je kod različitih prevodioca potrebno pribjegavati različitim ružnim zaobilaznim metodama za postizanje funkcionalnosti koja bi trebala biti podržana izravno u jeziku. Od dvaju korištenih prevodioca programskih jezika, po pitanju podrške standarda je daleko bolji slobodno dostupni prevodioc g++. Nedostatci u MSVC-u su djelomično popravljeni tek u verziji koja je izašla tek 2003. godine, kada je izvorni kôd već bio u potpunosti napisan.

Korisnost prevođenja projekta različitim prevodiocima se očituje kako u fazi testiranja pojedinih komponenti tako i u fazi eksperimentacije s cjelokupnim sustavom. U fazi testiranja, jedna te ista greška se često potpuno drukčije manifestira kod programa generiranih različitim prevodiocima što omogućava bolji uvid u okolnosti kod kojih do greške dolazi. Naravno, prethodna rečenica se odnosi na greške koje se ne manifestiraju deterministički u svakom prolazu programa preko dijela izvornog kôda koji sadrži pogrešku nego na greške koje se najteže pronalaze i najdulje ispravljaju. Manifestacije takvih grešaka ovise o ulaznim podatcima, pribavljenoj slici ili poruci, vremenskim odnosima među višestrukim tokovima izvođenja, fizičkom rasporedu objekata u radnoj memoriji računala ili potpuno nepredvidljivim zakulisnim operacijama, npr. vraćanju oslobođene memorije operacijskom sustavu od strane biblioteke memorijskog upravljača. Pokazalo se da dijagnostičke mogućnosti dvaju prevodioca uspijevaju razotkriti različite greške (nažalost ne i sve), što je značajno ubrzalo fazu testiranja komponenti sustava.

Nažalost, sofisticirane dijagnostičke mogućnosti nije moguće koristiti prilikom eksperimentiranja sa cijelim sustavom jer one značajno usporavaju izvođenje te onemogućuju rad u stvarnom vremenu. Tako je za produkciju konačnog sustava najvažnija performansa generiranog izvršnog kôda vremenski najzahtjevnijih algoritama obrade slike na niskoj razini kao što su glađenje ili rast područja. Zanimljivo je da se performanse pojedinih važnih algoritama generiranih različitim prevodiocima rasipaju i do 50%, i to ne uvijek u korist istog prevodioca. Razlog tome vjerojatno je intenzivno korištenje predložaka (*engl.* template) kao relativno nove tehnike programiranja jer se tom tehnikom omogućuju agresivne metode optimizacije generiranog kôda, a izvedbe tih optimizacija vjerojatno još nisu sasvim dorađene.

[1]Nažalost, procesorsko vrijeme ograničava opcije i kod primjena kod kojih ne postoje eksplicitni zahtjevi na brzinu obrade. Naime, ispitivanje ispravnog rada kao vrlo važna komponenta razvoja podrazumijeva primjenu više verzija algoritma na velikom broju slika, a to nije moguće kvalitetno obaviti ukoliko obrada jedne slike predugo traje.

Ipak, premoć u najzahtjevnijoj operaciji glađenja bila je dostatna da agent promatrač dobiven prevodiocem g++ u konačnici bude brži od suparnika za oko 30%, na istom računalu i operacijskom sustavu. Kad se ne bi radilo o eksperimentalnom sustavu, pravi pristup bi bio prevesti svaki kritični algoritam prevodiocem kojim se postiže njegova bolja performansa te objektne datoteke s izvršnim kôdom dobivenim različitim prevodiocima povezati u zajedničku izvršnu datoteku. Nažalost to nije jednostavna operacija zbog komplikacija vezanih uz nepostojanje standarda za sučelja objektnih datoteka (*engl.* ABI – application binary interface) dobivenih prevođenjem jezika C++. Jedini način za zaobilaženje tih komplikacija je izrada posebnog sučelja kritičnih algoritama u programskom jeziku C koji kao zreliji jezik ima standardizirano sučelje objektnih datoteka pa se izvršne datoteke različitih prevodioca mogu nesmetano povezivati. Ocijenjeno je da zbog ograničenog vremena potencijalni dobitak u performansi ne opravdava eksperimentiranje u tom smjeru.

5.2.3 Biblioteke opće namjene

Razvoj eksperimentalnog sustava ne bi bio moguć u zadanom roku bez ogromnog broja biblioteka opće namjene koje je moguće naći i dobaviti putem Interneta. Radi izbjegavanja ograničenja na kasnije korištenje programa, razmatrane su samo vanjske biblioteke s odgovarajućim licencama (npr, Free, Public domain te Gnu LGPL). Odabrane biblioteke opće namjene ukratko su opisane u nastavku teksta:

1. Standardna biblioteka jezika C++

   - uključuje temeljne komponente za strukturiranje podataka (dinamičko polje, lista, stog, rep, stablo, asocijativna tablica) i odgovarajuće temeljne algoritme (iteraciju, prebrojavanje, pretraživanje, sortiranje), te transparentno baratanje tokom znakova neovisno o izvorištu i odredištu (memorija, konzola ili datoteka);
   - sučelje je definirano standardom [cxx98] programskog jezika C++, a postoje zatvorene i otvorene izvedbe.

2. boost – skup biblioteka opće namjene za programski jezik C++

   - URL `http://www.boost.org/`;
   - projekt uključuje veliki broj biblioteka od kojih su korištene sljedeće:
     - `smart_ptr`: pokazivači s automatskom dealokacijom memorije (*engl.* reference counted pointer, smart pointer) [meyers97],
     - `ublas`: temeljni postupci linearne algebre (manipuliranje matricama);
     - `threads`: biblioteka za manipulaciju višestrukih usporednih tokova izvođenja programa;
   - projekt sadrži recenzirane biblioteke raznih autora, a postupak recenzije je detaljnije opisan u daljnjem tekstu;
   - licenca: Free.

3. Template Numerical Toolkit (TNT) – biblioteka složenijih matričnih operacija

   - `URL http://math.nist.gov/tnt/`;
   - korišteni su algoritmi za svojstvene vrijednosti i dekompozicije matrica LU, QR, Cholesky, SVD;

- biblioteka je razvijena na američkoj državnoj instituciji National Institute of Technology;
- licenca: Public domain.

4. cephes – skup matematičkih biblioteka
   - URL `http://www.netlib.org/cephes/`;
   - korišten je prijevod Levenberg-Marquardtovog gradijentnog optimizacijskog postupka u programski jezik C (originalni postupak je pisan u Fortranu, a nalazi se u biblioteci minpack koja je također dostupna);
   - autor prijevoda je Steve Moshier, dok je original razvijen na francuskoj državnoj instituciji Argonne National Laboratories;
   - licenca: Public domain.

Subjektivni dojam jest da izbjegavanje zatvorenih biblioteka uglavnom nije negativno utjecalo na kvalitetu izvedbe sustava. To se može činiti paradoksalno, ali čini se da donedavno uobičajeni model razvoja programske podrške sa plaćenim licencama i skrivenim izvornim kôdom u mnogim područjima ne uspijeva izdržati test vremena. Srž problema jest u modernim izazovima programskog inženjerstva: pokazuje se da je razvoj biblioteka za kojima postoji ogromna komercijalna potreba često vrlo složena i dugotrajna. Moderne biblioteke se razvijaju godinama, a tokom tog razvoja često dolazi do nepredvidljivih obrata uslijed spoznaja do kojih se dolazi isključivo njihovim korištenjem [hunt99]. Cijena razvoja svih građevnih komponenata neke složene aplikacije (jednostavne aplikacije nisu zanimljive jer su uglavnom već napisane!) stoga može biti znatno veća od cijene po kojoj bi se aplikacija mogla prodati krajnjim korisnicima. Kompanija koja bi platila savjesno izrađene biblioteke po njihovoj stvarnoj cijeni morala bi više godina raditi s negativnim profitom što na nesigurnom tržištu programske opreme često nije prihvatljivo. Resursi samo jedne kompanije stoga često nisu dovoljni da pokriju kvalitetan razvoj svih potrebnih biblioteka opće namjene pa se pribjegava raznim kratkoročnim rješenjima koja kod sposobnih razvojnih timova i inkrementalnog modela razvoja *mogu* evoluirati u kvalitetno konačno rješenje [brooks95].

Pored opisanog problema, postoji i još jedan razlog zbog kojeg se čini da otvorene licence pružaju plodnije tlo za izradu kvalitetnih programskih rješenja od uobičajenog modela s plaćenim licencama. Naime, pokazuje se da je vrijednost nekog programa izravno povezana sa sljedećim svojstvima:

- brojem kompetentnih korisnika sposobnih za prevladavanje početnih teškoća i slanje suvislih izvještaja o greškama i sugestijama za daljnji razvoj,
- mogućnosti uvida u izvorni kôd koja kompetentnim korisnicima omogućava preciznije dijagnosticiranje problema.

Sve licence koje ograničavaju područje upotrebe programa (među njima je nažalost i otvorena licenca Gnu GPL) slabe proizvod jer smanjuju broj korisnika proizvoda, sve licence koje ne dozvoljavaju uvid u izvorni kôd slabe proizvod jer otežava korisnicima sudjelovanje u daljnjem razvoju proizvoda. Opisani problem se očitava i u promjeni fokusa mnogih računalnih kompanija koje odustaju od naplate licenci na programska rješenja nego stvaraju dobit prodajom usluga. Drugim riječima, u takvom otvorenom modelu ne prodaje se sam program, nego konzultantsko vrijeme utrošeno na poduku o pravilnom korištenju programa te razvoj njegovih dodatnih svojstava.

Vrlo zanimljiv predstavnik otvorenog modela je projekt po imenu `boost`. Radi se o projektu čiji cilj je stvaranje kolekcije kvalitetnih prenosivih biblioteka opće namjene za programski jezik C++. Iako je početni skup biblioteka doprinesen od strane začetnika projekta, većinu ostalih biblioteka su napisali nezavisni autori i tvrtke. Sva komunikacija među administratorima projekta, autorima i ostalim subjektima koji su zainteresirani za projekt se obavlja preko triju listi za diskusiju koje se distribuiraju putem elektronske pošte. Proces prihvaćanja nove biblioteke u kolekciju donekle podsjeća na postupak recenzije radova u znanstvenim publikacijama. Autori najprije objave svoju namjeru doprinosa neke funkcionalnosti (npr, biblioteke za leksičku analizu teksta) i onda ostali sudionici rasprave mogu dati svoj komentar u smislu kolika je potreba za takvom bibliotekom, da li tako nešto već postoji i koji detalji bi bili od posebnog interesa. Ukoliko dovoljno mnogo sudionika rasprave izrazi svoj interes za rješenje problema, autori postavljaju preliminarnu verziju biblioteke na www stranicu projekta. Nakon iterativnih poboljšanja uslijed korisnih sugestija sudionika liste, autori predaju konačni prijedlog biblioteke i zahtijevaju formalnu recenziju. Konačni prijedlog mora zadovoljiti niz formalnih zahtjeva uključujući prisustvo ispitnih programa, primjera i dokumentacije. Formalnu recenziju odobrava trenutni administrator projekta, zadužuje jednog od aktivnih sudionika projekta kao glavnog recenzenta te određuje termin i trajanje recenzije. Formalne recenzije tipično traju 10 dana, a u njima sudjeluju svi zainteresirani sudionici rasprave zbog čega je u svakom trenutku u postupku najviše jedna recenzija. Po završetku postupka recenzije, glavni recenzent donosi odluku o (ne)prihvaćanju biblioteke na temelju svih pristiglih komentara. Uvjeti za pozitivnu recenziju su dosta strogi jer se zahtijevana funkcionalnost uglavnom utvrđuje konsenzusom koji se stvara ravnopravnom raspravom svih zainteresiranih strana. Pozitivna recenzija podrazumijeva uspješno prevođenje i testiranje na većini od 15 podržanih prevodioca i desetak operacijskih sustava. Nakon pozitivne recenzije, biblioteka postaje dostupna na www stranici projekta (licenca je potpuno slobodna, Free), a eventualni nedostatci i sugestije za proširenjima se autorima i dalje mogu dojavljivati elektronskom poštom na odgovarajuću listu za diskusiju. Ovakav postupak se može činiti previše anarhičan jer sudionici rasprave ne trebaju predočiti formalne dokaze da se razumiju u prirodu problema čije rješenje recenziraju. U praksi se međutim pokazuje da pisanje suvisle recenzije zahtijeva vrlo veliki angažman pa se neozbiljni doprinosi vrlo rijetko događaju i lakoćom filtriraju. Ovakav postupak donosi korist svim sudionicima jer autori dobivaju besplatnu povratnu informaciju o kvaliteti proizvedenog kôda, dok zajednica dobiva još jednu kvalitetnu i prenosivu komponentu koja se može koristiti s razumnom količinom sigurnosti u njezinu ispravnost.

5.2.4 Pomoćni alati

Slična razmatranja kao i kod biblioteka vrijede i za pomoćne alate za razvoj programske podrške. S obzirom da se alati ne ugrađuju u izvršni kôd, ovdje licenca Gnu GPL ne predstavlja nikakvo ograničenje pa se najčešće i koristi. Za razliku od klasičnih zatvorenih programa s plaćenim licencama, otvorena rješenja su najčešće prenosiva, i mogu se kombinirati te tako bolje prilagoditi potrebama konkretne primjene. Subjektivni je dojam da otvorene aplikacije uglavnom daju superiorna rješenja u odnosu na zatvorene alternative, s izuzetkom interaktivnog ispravljanja grešaka gdje alternativa nudi bolje korisničko sučelje uz približno istu funkcionalnost.

1. CVS – concurrent versions system
   - `URL http://www.cvshome.org/`;

- alat za baratanje verzijama izvornog kôda;
- koristi se za praćenje promjena i sinkronizaciju izmjena u izvornom kôdu;
- posebno je potreban kada se program razvija na više platformi ili kad više ljudi doprinosi razvoju istog programa;
- licenca: Gnu GPL;
- zatvorena alternativa: Microsoft SourceSafe.

2. meld

   - `URL http://meld.sourceforge.net/`;
   - program za vizualnu usporedbu tekstualnih datoteka;
   - omogućava usporedbu pojedinačnih datoteka, cijelih direktorija te verzija datoteka spremljenih u cvs odlagalištu;
   - licenca: Gnu GPL;
   - zatvorena alternativa: Microsoft windiff.

3. tmake

   - `URL http://www.trolltech.com/developer/download/tmake.html/`;
   - alat za stvaranje i održavanje datoteka koje određuju tok prevođenja programskog sustava (tzv. *makefile*ova);
   - licenca: TrollTech Free;
   - zatvorena alternativa: razvojno okruženje MSVC.

4. Source Navigator

   - `URL http://sourcenav.sourceforge.net/`;
   - alat za pretraživanje i unakrsno referenciranje identifikatora, te vizualni prikaz međuovisnosti komponenti sustava, klasne hijerarhije i strukture pojedinih klasa;
   - licenca: Gnu GPL;
   - zatvorena alternativa: razvojno okruženje MSVC.

5. ddd+gdb

   - `URL http://sources.redhat.com/gdb/`;
   - program za interaktivno praćenje izvođenja programa (gdb) te grafičko korisničko sučelje (ddd);
   - licenca: Gnu GPL;
   - zatvorena alternativa: razvojno okruženje MSVC.

6. skriptni jezici Perl i Python

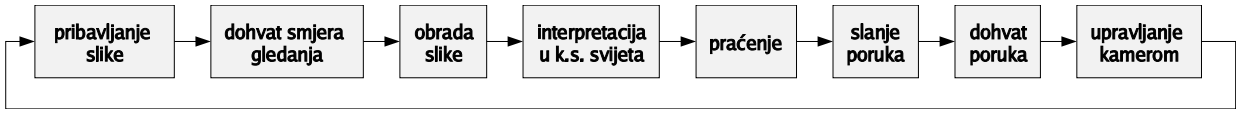
   - `URL http://www.python.org/`, `URL http://www.perl.org/`;
   - interpretirani programski jezici s iznimno bogatim bibliotekama;
   - korisni za povezivanje pojedinih alata (npr: tmake+cvs), u jedinstveno rješenje prilagođeno konkretnoj primjeni;

- licenca: Gnu GPL;
- zatvorena alternativa: Visual Basic (?).

7. ImageMagick
   - URL `http://www.imagemagick.org/`;
   - skup alata za neinteraktivno baratanje slikom;
   - korisni kao građevni elementi skripti za manipuliranje većim brojem slika, kad interaktivni pristup nije prikladan;
   - licenca: Gnu GPL;
   - zatvorena alternativa: ?
8. editori teksta gvim i nedit
   - URL `http://www.vim.org/`, URL `http://www.nedit.org/`;
   - licenca: Gnu GPL;
   - zatvorena alternativa: razvojno okruženje MSVC.

5.3 Izvedba promatrača

Program koji obavlja funkciju promatrača se sastoji od niza procedura koje se izvršavaju slijedno u beskonačnoj petlji, prema sl. 5.2. Redoslijed procedura je uvjetovan protokom informacija u skladu s funkcionalnošću agenta promatrača. Svaki ciklus obrade započinje pribavljanjem nove slike. Istovremeno sa slikom, potrebno je dohvatiti i smjer gledanja kamere kako bi se rezultati obrade slike mogli ispravno interpretirati u vanjskim kontekstima. Nakon toga, odrađuje se vremenski najzahtjevnija operacija obrade pribavljene slike, koja se svodi na pronalaženje objekata u 2D koordinatnom sustavu slike. Trenutni položaj postolja jedinstveno određuje vanjske parametre kamere, pa se u ovom trenutku uz poznate unutrašnje parametre i položaj ravnine kretanja, dobiveni položaji objekata interpretiraju u referentnom 3D koordinatnom sustavu svijeta. Pojedinačni 3D položaji objekata dobiveni tijekom vremena tada se grupiraju u trajektorije na temelju kojih je moguće filtrirati "lažne" objekte čije pojavljivanje u vremenu i prostoru nije konzistentno. Nakon toga, obavlja se eventualno slanje poruka koordinatoru u skladu s komunikacijskim protokolom. Sada smo došli do trenutka u kojem je najelegantnije provjeriti ima li novih koordinatorovih poruka, jer njihovo eventualno prisustvo može utjecati isključivo na konačnu proceduru slijeda obrade, odnosno na pomicanje smjera gledanja kamere.



Slika 5.2: Slijed procedura agenta promatrača.

5.3.1 Pribavljanje slike

Pribavljaju se slike u boji dimenzija 320×240 slikovnih elemenata u RGB formatu, u kojem je svaki slikovni element predstavljen trobajtnim kôdom. Zbog konačne propusnosti veze, pribavljena slika u najboljem slučaju odgovara trenutku od prije 40 ms, u skladu s raspravom iz 5.1.2.

5.3.2 Dohvat smjera gledanja

Smjer gledanja kamere je potreban za određivanje njenih vanjskih parametara na temelju kojih se računa homografijska matrica transformacije iz slikovnih koordinata u referentni k.s. svijeta. Ovdje treba obratiti pozornost da je za ispravnu interpretaciju pribavljene slike potrebno ocijeniti smjer gledanja koji je kamera imala u trenutku stvaranja slike. To je posebno složen posao ukoliko se kamera kreće uslijed prethodne naredbe. U tom slučaju, položaj kamere je potrebno ocijeniti na temelju trenutne kutne brzine postolja te latencije pribavljanja slike uvećanu za jednosmjernu latenciju komunikacije s pokretnim postoljem. Latencija komunikacije ovisi o brzini serijske veze i iznosi od nekoliko milisekundi na 9600 bps do ispod milisekunde na 38400 bps.

5.3.3 Obrada slike

Obrada slike se temelji na pretpostavci da se praćeni objekti mogu segmentirati na temelju boje. Procedura se može razložiti na sljedeće korake:

1. Glađenje Gaussovim filterom u cilju smanjenja šuma u slici. Eksperimentiranjem je utvrđeno da se optimalni rezultati dobivaju uz $\sigma = 0.5$;

2. Pretvaranje slike iz RGB formata u format HSV, u kojem se slikovni elementi opisuju trojkom (nijansa, zasićenje, svjetlina).

   Format HSV je perceptualno uniformniji od RGB formata, što znači da su Euklidske udaljenosti među vektorima boja više usklađene s intuitivnim ljudskim osjećajem za različitost boja. Štoviše, format HSV je posebno pogodan za detekciju istaknutih boja spektra jer se pokazuje da je njihova nijansa relativno invarijantna na varijabilne uvjete osvjetljenja. Komponente HSV formata se obično izražavaju realnim brojevima u intervalu $[0, 1]$. Pretvorba slikovnih elemenata iz uobičajenog RGB formata u spomenuti HSV format može se izraziti jednadžbama (5.2).

$$
\begin{aligned}
V &= \max(R, G, B) \\
S &= \frac{V - \min(R, G, B)}{V} \\
H &= \begin{cases}
0, & S = 0 \\
1/6 \cdot \frac{(G-B)}{R-\min(R,G,B)}, & V = R, G \geq B \\
1 + 1/6 \cdot \frac{(G-B)}{R-\min(R,G,B)}, & V = R, G < B \\
1/3 + 1/6 \cdot \frac{(B-R)}{G-\min(R,G,B)}, & V = G \\
2/3 + 1/6 \cdot \frac{(R-G)}{B-\min(R,G,B)}, & V = B
\end{cases}
\end{aligned}
\tag{5.2}
$$

3. Maskiranje pretamnih i presvijetlih područja koji se detektiraju na temelju malenih komponenata svjetline odnosno zasićenja.

4. Izlučivanje objekata na temelju zadane nijanse i dozvoljenog odstupanja. Svi slikovni elementi ulazne slike čija nijansa nije zadovoljavajuća u ovom koraku se postavljaju na crnu boju. Ovdje valja obratiti pozornost da je nijansa zapravo kutna vrijednost što znači da su boje s nijansama 0.01 i 0.99 vrlo slične i po nijansi udaljene za 0.02. Praćeni su crveni objekti, pa je zadana nijansa $H = -0.02$, uz dozvoljenu udaljenost od 0.08.

5. Pronalaženje povezanih regija koje pripadaju objektima. Korišten je algoritam za pronalaženje binarnih područja u slici dobivenoj u prošlom koraku. Na kraju postupka, objekti manji od 20 slikovnih elemenata odbacuju se kao nepouzdani.

5.3.4 3D interpretacija

Homografijska matrica transformacije iz slikovne ravnine u ravninu gibanja praćenih objekata određuje se na temelju unutrašnjih i vanjskih parametara kamere te poznatog položaja ravnine gibanja, kao što je opisano u 4.4.3. Vanjski parametri kamere se određuju na temelju poznatih kuteva zakreta i nagiba te poznatih kalibracijskih parametara kao što je opisano u 4.4.2. Položaj objekta u ravnini gibanja se dobiva izravno iz njegovih slikovnih koordinata primjenom dobivene homografijske matrice.

5.3.5 Praćenje

Detektirani objekti se pridružuju trajektorijama koje su izgrađene tokom obrade prethodnih slika. Pri udruživanju se uzimaju u obzir samo oni dijelovi trajektorija koji su dobiveni unutar protekle dvije sekunde. Uparivanje se temelji na matrici udaljenosti između položaja svih detektiranih objekata i projiciranih trenutnih položaja za svaku od trajektorija. Algoritam slijedi "pohlepni" iterativni pristup u kojem se u svakoj iteraciji uparuju objekt i trajektorija čiji odgovarajući položaji su međusobno najbliži. Algoritam završava kad je pronađena najmanja udaljenost veća od unaprijed zadanog praga koji se modulira u ovisnosti o broju alternativnih uparivanja kako za objekt tako i za projicirani položaj trajektorije. Iako ovakav algoritam nije optimalan (npr, u smislu minimizacije ukupne udaljenosti pridruženih parova), u praksi za odabranu primjenu postiže zadovoljavajuće rezultate u smislu kvalitete praćenja i brzine izvođenja.

5.3.6 Slanje poruka

Položaji objekata koji su detektirani i u barem tri prethodne slike se šalju koordinatoru porukom `mjerenje`. Ukoliko se kamera pomiče, potrebno je odrediti novo vidno polje promatrača homografijom njegovih rubnih točaka (homografija čuva pravocrtnost!) i poslati ga koordinatoru porukom `vidno_polje`. Vremenski opisnik poruka se postavlja u skladu s procijenjenim trenutkom stvaranja slike, u skladu s referentnim satom koordinatora. Pored toga, rezultati obrade se ispisuju i u lokalnom prozoru, zbog dijagnostičkih razloga.

5.3.7 Primanje poruka

Provjerava se da li je na zauzeti mrežni ulaz računala pristigla poruka od strane koordinatora. U pozitivnom slučaju, prelazi se u odgovarajući režim rada i po potrebi obavlja predbilježba operacije pomaka kamere.

5.3.8 Upravljanje kamerom

Smjer gledanja kamere se mijenja u sljedećim situacijama:

1. Trenutni način rada je praćenje, a praćeni objekt se približava rubu slike. Smjer gledanja se pomiče u smjeru prividnog gibanja objekta.

2. Trenutni način rada je traženje, a prošlo je više od 10 s nakon posljednjeg pomaka kamere bez da je detektiran ijedan objekt. Kut zakreta se pomiče za pola vidnog polja u skladu s tekućim smjerom traženja. Kad se dođe do kraja radnog područja kamere, smjer traženja se mijenja. Nakon završenog punog ciklusa, varira se kut nagiba po analognom algoritmu.

3. Dobivena je koordinatorova poruka `usmjeri()` ili `traži`. Smjer gledanja se pomiče što je bliže moguće traženom položaju.

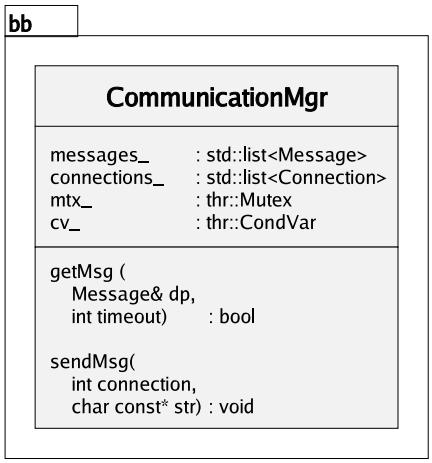
Zbog inercije postolja, njegovo pomicanje u skladu sa zadanom naredbom redovno traje više stotina milisekundi. Posljedica toga je da će se postolje pomicati prilikom obrade većine slika iz pribavljenog slijeda pa je na to potrebno obratiti posebnu pažnju.

5.4 Detalji izvedbe koordinatora

Kao što je opisano u 3.2.2 i ilustrirano na sl. 3.7, unutrašnja arhitektura koordinatora slijedi obrazac oglasne ploče [pfleger98]. U tom obrascu, rješenje složenog zadatka se tvori suradnjom međusobno nezavisnih procedura ili izvora znanja. Svaka procedura djeluje nad globalnom strukturom podataka ili oglasnom pločom koja sadrži sve ulazne podatke i međurezultate te je odgovorna za rješenje podzadataka na pojedinom stupnju apstrakcije. Izvršavanje procedura traje relativno kratko, dok je za njihovu aktivaciju odgovorna nadgledna komponenta na temelju testiranja tzv. preduvjeta aktiviranja koji je specifičan za svaku proceduru. U nastavku teksta, bit će opisani izvedbeni detalji i aktivacijski uvjeti za svaku od temeljnih pet procedura.

5.4.1 Pribavljanje poruka promatrača

Sudjelovanje koordinatora u komunikacijskom protokolu sastoji se od primanja podatkovnih poruka od strane pridruženih promatrača i slanja upravljačkih poruka istom skupu promatrača, kao što je opisano u 3.2.3. Općenito, primanje poruka je složeniji dio komunikacije, jer primaoc ne zna kad će mu poruka doći pa mora ili čekati ili više puta provjeravati je li poruka stigla. Primanje se posebno komplicira kada ima više mogućih pošiljaoca, te kada primaoc mora obraditi primljene podatke u stvarnom vremenu. Čest način za rješavanje tog problema je smještanje komunikacije sa svakim pojedinačnim promatračem u zasebni prijemni tok izvođenja programa odnosno dretvu, koji pribavljene poruke proširuje opisnikom pošiljaoca i smješta u zajedničku listu poruka. Prijemni tok izvođenja se dodjeljuje svakom komunikacijskom zahtjevu, iz posebnog dodjeljivačkog toka izvođenja koji prima zahtjeve na za to rezerviranom mrežnom ulazu računala (dodjeljivački tok otvara komponenta za komunikaciju prilikom pokretanja programa). Komunikacija se tako može obaviti jednostavnim sinkronim pozivima koji blokiraju odgovarajući prijemni tok sve dok poruka ne postane dostupna na ulazu. U skladu s opisanim rješenjem, glavni program obavlja svoju obradu neovisno o komunikaciji, te povremeno proziva komunikacijsku komponentu kako bi saznao je li u međuvremenu stigla neka poruka, te u skladu s odgovorom poduzima odgovarajuće akcije (npr, postavlja mjerenje na oglasnu ploču). Skica komunikacijske komponente programa prikazana je na sl. 5.3. Pored liste pristiglih neobrađenih poruka, komponenta vodi evidenciju o otvorenim komunikacijskim vezama, dok sinkronizacijski objekti služe za osiguravanje konzistentnog pristupa listi poruka koja se referencira i iz glavnog i iz prijemnih tokova izvođenja.



Slika 5.3: Skica komunikacijske komponente koordinatora `CommunicationMgr`. Komponenta je sadržana u paketu oglasne ploče `bb`.

5.4.2 Grupiranje mjerenja u trajektorije

Pojedinačna trajektorija je slijed mjerenja istog promatrača za koja se vjeruje da odgovaraju istom objektu. Svako mjerenje se sastoji od položaja objekta i od vremena stvaranja slike u kojoj je objekt detektiran. Za svakog priključenog promatrača, ova procedura grupira mjerenja dobivena u različitim vremenskim trenutcima u trajektorije. Pri tome se koristi algoritam koji je već opisan u 5.3.5, te se na oglasnu ploču bilježe samo najrecentniji odsječci trajektorija koji su zabilježeni u posljednje dvije sekunde (kao i u izvedbi promatrača).

5.4.3 Integracija trajektorija u percipirane objekte

Pojedinačne trajektorije koje su dobivene različitim promatračima a odgovaraju istom fizičkom objektu nisu identične zbog raznih sistematskih pogrešaka koje su uglavnom uzrokovane nesavršenom kalibracijom opreme i konačnom visinom objekta. Pored toga, pojedinačna mjerenja u različitim trajektorijama gotovo uvijek odgovaraju različitim trenutcima jer promatrači nisu međusobno sinkronizirani i imaju različite performanse, kao što je prikazano na sl. 5.4. Postupak pronalaženja grupa trajektorija koje odgovaraju istom objektu stoga nije jednostavan zadatak, a donekle je sličan postupku grupiranja jer se obavlja prema principima prostorne bliskosti i sličnosti oblika. Nažalost, zbog specifičnosti problema, korespondenciju nije prikladno tražiti poznatim općenitim metodama grupiranja, jer su parovi trajektorija istog promatrača međusobno nekompatibilni, tj. ne mogu odgovarati istom fizičkom objektu. Štoviše, tijekom napredovanja postupka parovi trajektorija različitih trajektorija također mogu postati nekompatibilni. Ovo se događa kad god su dvije trajektorije pridružene korespondencijskim skupovima u kojima figuriraju trajektorije istog promatrača.

Prvi korak pri pronalaženju korespondencije je inicijalizacija matrice udaljenosti D_{ij}, koja za svaki par trajektorija (i,j) sadrži procjenu njihove međusobne različitosti. Ukoliko su trajektorije para međusobno nekompatibilne, njihova udaljenost nije relevantna, pa se takva situacija označava s D_{ij}=NK. Elementi matrice D_{ij} se računaju prema sljedećem postupku:

- ako trajektorije i i j pripadaju istom promatraču, D_{ij}:=NK;
- inače, postupak se nastavlja kako slijedi (ilustracija je na sl. 5.4):

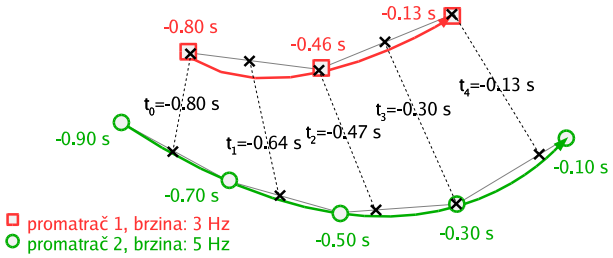
1. pronalazi se zajednički vremenski interval (t_0, t_N), kao presjek područja definicija trajektorija (na slici, $t_0 = -0.80\,\mathrm{s}$, $t_N = -0.13\,\mathrm{s}$);
2. određuju se trenutci $t_i = i \cdot \frac{t_N - t_0}{N}$, $i = 1, 2, \ldots, N-1$, koji početni interval dijele na $N-1$ jednakih dijelova (na slici, N=5);
3. za obje trajektorije A i B, interpoliraju se 2D vektori $\mathbf{P_{Ai}}$ i $\mathbf{P_{Bi}}$, koji odgovaraju položajima objekta u trenutcima $t_i, i = 0, 1, 2, ..., N$ (na slici su ti položaji označeni križićima);
4. formira se skup vektora udaljenosti odgovarajućih interpoliranih položaja $\mathbf{d_i} = \mathbf{P_{Ai}} - \mathbf{P_{Bi}}$, $i = 0, 1, ..., N$;
5. daljnja razmatranja se temelje na statističkim svojstvima komponenata x i y vektora udaljenosti $\mathbf{d_i} = x_i \cdot \mathbf{i} + y_i \cdot \mathbf{j}$, koja se izražavaju vektorima $\mathbf{m_d}$ i $\mathbf{s_d}$:

$$\mathbf{m_d} = \begin{bmatrix} \mu_x & \mu_y \end{bmatrix}^T \tag{5.3}$$

$$\mathbf{s_d} = \begin{bmatrix} \sigma_x & \sigma_y \end{bmatrix}^T, \tag{5.4}$$

   gdje μ i σ označavaju srednju vrijednost i standardnu devijaciju po komponentama x i y vektora $\mathbf{d_i}$, $i = 0, 1, ..., N$;
6. konačno, različitost trajektorija se izražava pomoću procjene njihove prostorne udaljenosti $\mathbf{m}_d$ i procjene različitosti njihovih oblika $\mathbf{s_d}$:

$$D_{ij} = \sqrt{||\mathbf{m_d}||^2 + ||\mathbf{s_d}||^2}. \tag{5.5}$$



Slika 5.4: Priprema za računanje mjere različitosti dvaju nesinkroniziranih trajektorija.

Kao što je već spomenuto, položaji istog objekta zabilježeni od strane različitih promatrača mogu se razlikovati uslijed višestrukih izvora grešaka. Eksperimenti su pokazali da se dominantni efekt tih grešaka na kratke segmente trajektorije može modelirati jednostavnom translacijom. Korespondencija se stoga zasniva na funkciji udaljenosti koja ovisi o srednjoj vrijednosti i standardnoj devijaciji odstupanja između odgovarajućih točaka trajektorije. Motivacija za takav oblik funkcije udaljenosti je modeliranje ortogonalnih utjecaja oblika i prostorne udaljenosti na mjeru različitosti dvaju trajektorija.

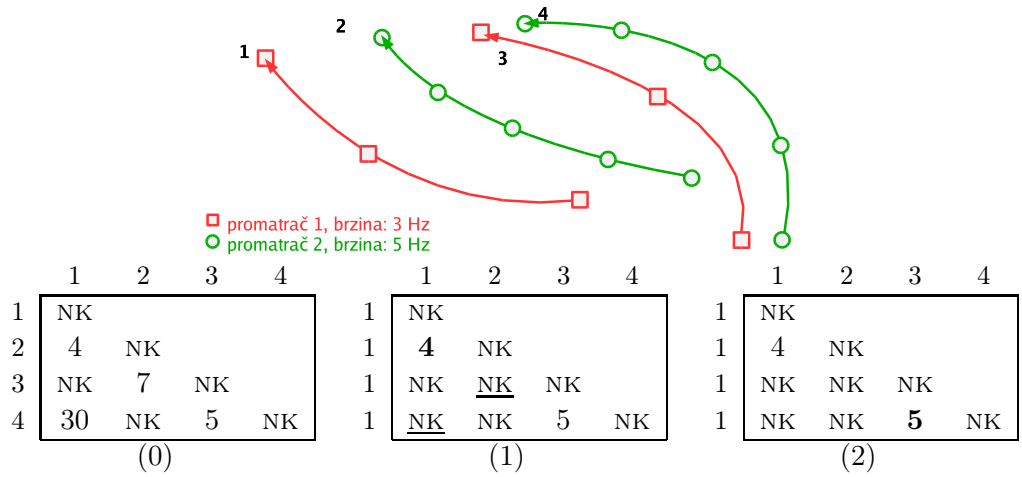
Nakon određivanja matrice udaljenosti, pojedinačne trajektorije se udružuju prema jednostavnom "pohlepnom" algoritmu koji je prikazan na sl. 5.5. Bolji rezultati bi se vjerojatno postigli algoritmom koji minimizira srednju kvadratnu pogrešku, ali zbog vremenskih ograničenja nije bilo moguće napraviti eksperimente u tom smjeru.

Pretpostavimo situaciju u kojoj dva promatrača prate dio scene u kojem se kreću dva pokretna objekta, tako da se na oglasnoj ploči dobivaju četiri pojedinačne trajektorije, kao što je ilustrirano na sl. 5.6. U prikazanoj situaciji, algoritam za određivanje korespondencije pojedinačnih trajektorija bi trebao zaključiti da trajektorije 1 i 2 odgovaraju percipiranom

```
zauvijek{
  nađi(i,j):¬isti_objekt(k,i)∧D_ij →min;
  ako (D_ij>prag){
    kraj;
  }
  udruži_trajektorije(i,j)

  ∀(trajektorija k): isti_objekt(k,i)
    ∀(trajektorija l): isti_promatrač(k,l){
       D[l,j]=D[j,l]=NK
  }

  ∀(trajektorija k): isti_objekt(k,j)
    ∀(trajektorija l): isti_promatrač(k,l){
       D[l,i]=D[i,l]=NK
  }
}
```

Slika 5.5: Skica algoritma za određivanje korespondencije pojedinačnih trajektorija.

objektu A, odnosno da trajektorije 3 i 4 odgovaraju percipiranom objektu B. Tijek izvođenja algoritma se u tom slučaju može prikazati slijedom vrijednosti matrica udaljenosti nakon svake iteracije algoritma, što je također prikazano na sl. 5.6. Matrice udaljenosti su simetrične, pa su im u slici prikazani samo elementi donjeg trokuta. U nultoj iteraciji, elementi



Slika 5.6: Pojedinačne trajektorije oglasne ploče u zamišljenom eksperimentu, i elementi matrice udaljenosti tijekom tri iteracije korespondencijskog algoritma (detaljnije u tekstu).

matrice udaljenosti su određeni vrijednostima koje su im dodijeljene tijekom inicijalizacije. Trajektorije $\{3,1\}$, te trajektorije $\{4,2\}$ su odmah označene kao nekompatibilne jer potiču od istih promatrača, dok se pretpostavlja da su izrazom (5.5) dobivene vrijednosti $D_{12}=4$, $D_{32}=7$, $D_{41}=30$, $D_{43}=6$. U prvoj iteraciji, udružuju se trajektorije 1 i 2, zbog minimalne udaljenosti. Kao rezultat toga, trajektorije $\{3,2\}$ i $\{4,1\}$ su označene kao nekompatibilne, jer korespondencijski skupovi $\{1,2,3\}$ i $\{1,2,4\}$ nemaju smisla. U drugoj iteraciji, udružuju se trajektorije 3 i 4, kao najbliži još neupareni par. Postupak se ovdje zaustavlja, jer nema više kompatibilnih trajektorija koje nisu već u zajedničkom korespondencijskom skupu. Vidi se da algoritam primjenom logičkih ograničenja uspijeva zaobići dilemu između uparivanja

$\{2,1\}$ i $\{2,3\}$ koja imaju vrlo slične mjere udaljenosti, dok predefinirani prag dolazi do izražaja samo kada su bliski dijelovi scene zaklonjeni različitim promatračima.

Svaka grupa trajektorija koja je dobivena opisanim postupkom sadrži povijest gibanja jednog fizičkog objekta viđenu kroz kamere različitih promatrača. Međutim, konačni cilj ove procedure jest ocijeniti *trenutne* položaje fizičkih objekata u sceni te ih upisati na oglasnu ploču kao percipirane objekte. Traženi položaji se mogu odrediti na temelju trenutnih pojedinačnih položaja u svakoj od trajektorija grupe, koji se mogu ocijeniti linearnom ekstrapolacijom, na temelju posljednja dva mjerenja trajektorije. U primijenjenoj jednostavnoj izvedbi, položaj percipiranog objekta oglasne ploče određuje se kao srednja vrijednost procijenjenih trenutnih pojedinačnih položaja. Sofisticiranije sheme određivanja položaja percipiranih objekata razmatrane su u 3.3.1.

5.4.4 Formiranje trajektorija percipiranih objekata

Vremenski slijedovi percipiranih objekata mogu se grupirati u percipirane trajektorije po istom algoritmu koji je korišten i u formiranju pojedinačnih trajektorija i u izvedbi agenta promatrača. Po završavanju grupiranja novih položaja percipiranih objekata, percipirane trajektorije se upisuju na oglasnu ploču.

5.4.5 Strategija za koordinaciju promatrača

Kao što je rečeno prije, cilj procedure za koordinaciju je postići optimalno praćenje stanja u sceni, slanjem koordinacijskih poruka (uputa) promatračima. Optimalno praćenje pri tome znači nalaženje kompromisa između preciznog praćenja pojedinih objekata i pokrivanja praznog prostora scene u cilju brzog otkrivanja novih objekata.

Predložena minimalistička izvedba je zamišljena za postizanje robustne kordinacije razvedenoj sceni s malim brojem objekata. U svakom pozivu procedure razmatra se mogućnost boljeg praćenja stanja u sceni slanjem koordinacijske poruke točno jednom promatraču uz korištenje poruka `traži` i `prati`. Kao što je već obrazloženo u 3.2.2, na koordinacijske poruke se ne predviđaju odgovori zbog zahtjeva za rad u stvarnom vremenu. Neka N_{Op} označava trenutni broj percipiranih objekata oglasne ploče, koji odgovara broju fizičkih objekata u sceni koji su vidljivi barem jednom od promatrača. Tada je svakom promatraču moguće zadati $N_O + 1$ različitih odredišta odnosno koordinacijskih poruka. Neka odredišta od 0 do $N_O - 1$ označavaju odgovarajuće objekte, a odredište NO neka označava pretraživanje. Sveukupan broj mogućih poruka u danom pozivu procedure tako iznosi $N_P \cdot (N_O + 1)$, gdje je N_P broj promatrača, a svaka poruka se može izraziti kao uređen par (promatrač, odredište).

- objekt kojeg prati samo jedan promatrač definira se kao *slabo praćeni objekt*;
- promatrač koji prati barem jedan slabo praćen objekt definira se kao *vezan promatrač*;
- promatrač koji prati barem jedan objekt kojeg prate ukupno dva promatrača definira se kao *važan promatrač*;
- promatrač koji ne prati niti jedan objekt definira se kao *besposlen promatrač*;
- promatrač koji nije ni vezan ni važan ni besposlen definira se kao *slobodan promatrač*;

Ponašanje procedure definirano je sljedećim skupom pravila:

1. promatračima koji se nalaze u načinu rada "traženje", vezanim, važnim, slobodnim i besposlenim promatračima dodjeljuju se prioriteti 0, 0, 1, 2 odnosno 3;

2. ako nema promatrača s prioritetom većim od 0, procedura završava bez slanja koordinacijske poruke;

3. inače, odabire se promatrač s najvišim prioritetom i označava s O_C (ukoliko postoji više takvih promatrača, konflikt se rješava shemom s kružnim prvenstvom);

4. ako postoji slabo praćeni objekt A koji je izvan vidnog polja promatrača O_C, promatraču O_C se dodjeljuje praćenje objekta A (kad bi A bio u vidnom polju promatrača O_C, to bi značilo da mu je A zaklonjen jer O_C nije vezan i ne prati A);

5. inače, samo ako O_C nije važan, šalje mu se poruka `traži`.

5.4.6 Nadgledni sustav oglasne ploče

Kod obrasca oglasne ploče u skladu sa sl. 3.7, nadgledni sustav određuje koji će se izvor znanja (procedura) izvesti u sljedećem koraku obrade. Donesena odluka se temelji na ispitivanju preduvjeta aktiviranja za svaki izvor znanja, a oblikovanje tih preduvjeta može biti dosta složeno jer sasvim ovisi o konkretnoj proceduri. Najlakše je donijeti odluku kod procedura čija obrada se svodi na obradu nekog događaja. Tako je dostupnost poruke u listi komunikacijske komponente prirodan preduvjet procedure pribavljanja poruka. Slično, proceduru grupiranja mjerenja će imati smisla aktivirati tek kada na red dođe poruka s novom grupom detektiranih objekata nekog od promatrača.

Kod ostalih procedura, stanje nije tako jasno, jer se tijekom izvođenja programa događa da neobrađenih ulaznih podataka na ploči ima češće nego što je procedure moguće pokretati a da se ne iscrpe dostupni procesni resursi. Stoga, kad se na ploču upiše novo mjerenje program ne pokušava odmah udružiti dotičnu trajektoriju s trajektorijama ostalih promatrača jer je to dosta zahtjevna operacija. Jedno rješenje tog problema bilo bi pokretanje procedure tek kad se oblici udruženih trajektorija značajno promijene od posljednjeg aktiviranja procedure, ali obavljanje svih relevantnih testova bi trajalo gotovo kao i izvršavanje same procedure pa to ne bi bio veliki dobitak. Općenito, aktivacijski preduvjeti u obrascu oglasne ploče moraju biti jednostavniji od složenosti odgovarajućih procedura, jer bi se inače moglo dogoditi da sustav većinu vremena bezuspješno testira preduvjete procedura koje se rjeđe aktiviraju. Stoga je odabran pragmatičan pristup rješenju tog problema u kojem se ne inzistira na složenim preduvjetima, nego se aktiviranje problematičnih procedura obavlja u fiksnim vremenskim intervalima. U konkretnom slučaju, procedure za udruživanje trajektorija i održavanje trajektorija percipiranih objekata se slijedno pozivaju tri puta u sekundi, dok se koordinacija promatrača obavlja svakih 2 sekunde. Preduvjeti aktivacije najvažnijih procedura oglasne ploče su sažeti u tablici 5.1

	naziv procedure	preduvjet aktiviranja
1	pribavljanje poruka	dostupnost poruke
2	grupiranje mjerenja	pribavljena mjerna poruka
3	udruživanje trajektorija	svakih 0.33 s
4	formiranje složenih trajektorija	nakon svakog poziva procedure broj 3.
5	prikaz stanja u sceni	nakon svakog poziva procedure broj 3 ili svake 2 s.
6	koordinacijska strategija	svake 2 s

Tablica 5.1: Preduvjeti aktiviranja procedura oglasne ploče.

Pored temeljnih pet procedura koje su prikazane na sl. 3.7, izvedba koordinatora sadrži i dijagnostičku proceduru, čiji zadatak je pregledan prikaz stanja oglasne ploče na zaslonu

računala. Ona se aktivira u skladu s ritmom postavljanja percipiranih objekata na oglasnu ploču, ili najkasnije nakon svake 2 s, u slučaju da u sceni ne postoji niti jedan objekt.

Poglavlje 6

Eksperimentalni rezultati

Izvedba eksperimentalnog sustava obuhvaća cjelokupnu funkcionalnost propisanu temeljnom višeagentskom arhitekturom koja je opisana u odjeljku 3.2. Detaljna provjera funkcionalnosti obavljena je za sve komponente sustava osim procedure za koordinaciju promatrača, koja pretpostavlja minimalna preklapanja u sceni pa ne bi mogla biti isprobana bez zahtjevnih intervencija u postojećem okolišu. Ocijenjeno je da je značaj te procedure prvenstveno teorijski, kao osnova za robustnije postupke koji bi koristili i znanje o zaklonjenim dijelovima scene pojedinih promatrača. Stoga je nauštrb eksperimenata s koordinacijskom procedurom provedena opsežnija teoretska razrada geometrijskih odnosa u porazdijeljenom aktivnom vidu, koja je predstavljena u poglavlju 4.

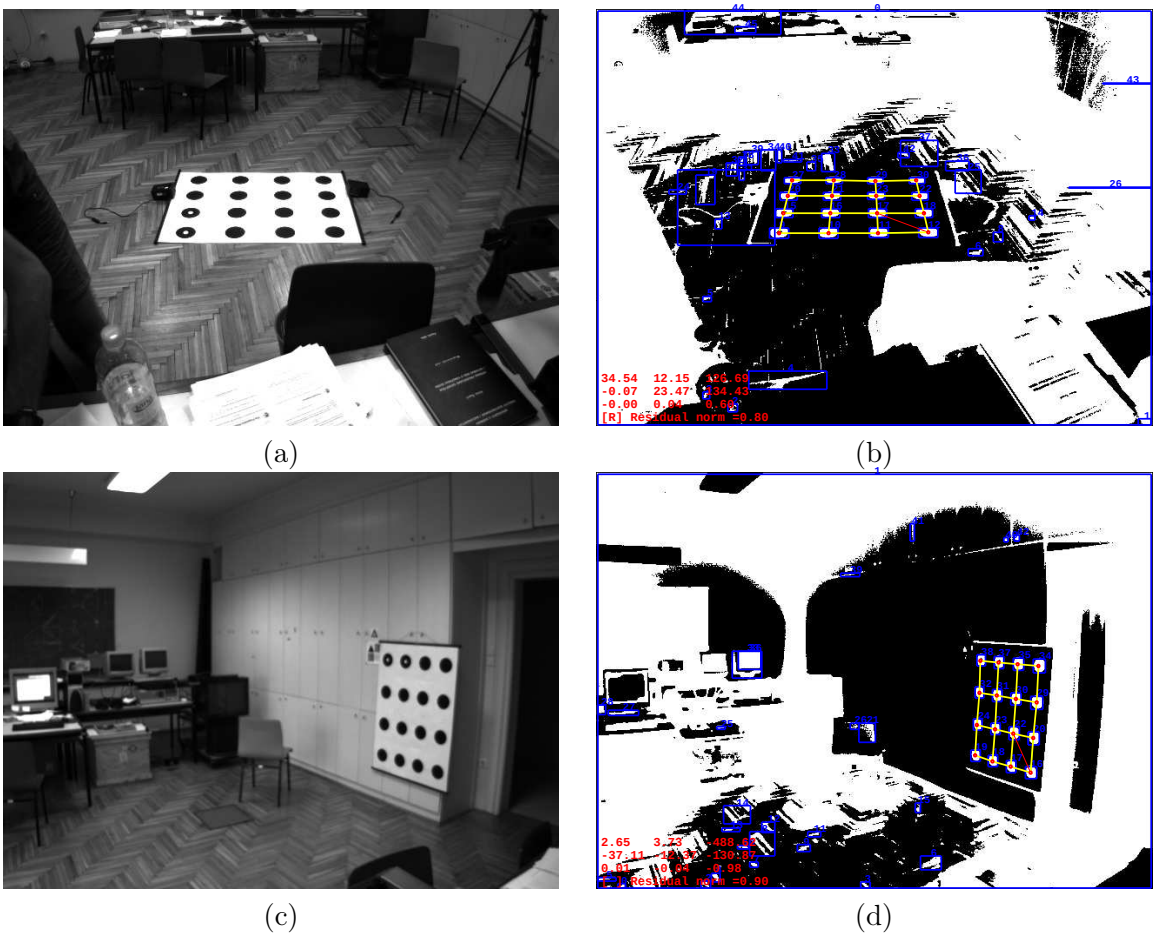
Ovo poglavlje je organizirano kako slijedi. Eksperimenti vezani uz umjeravanje korištenih kamera, opisani su u odjeljku 6.1. Odjeljak 6.2 prikazuje rezultate praćenja pokretnih objekata pojedinačnim promatračima. Konačno, eksperimenti s porazdijeljenim praćenjem u sustavu s tri promatrača i jednim koordinatorom koji pojedinačne poglede integrira u zajednički prikaz, predstavljeni su u odjeljku 6.3.

6.1 Umjeravanje parametara kamere

Na području umjeravanja parametara kamera, obavljene su tri vrste eksperimenata. Najprije je ispitana robustnost algoritma za traženje mjernog uzorka s obzirom na zadani prag binarizacije. Nakon toga, predstavljena je grupa eksperimenata vezanih uz umjeravanje unutrašnjih parametara kamere. Konačno, preliminarni eksperimenti u vezi s umjeravanjem vanjskih parametara kamere navedeni su u završnici odjeljka.

6.1.1 Pronalaženje značajki mjernog uzorka

Algoritam za pronalaženje mjernog uzorka i preciznog položaja pojedinih njegovih značajki predstavljen je u 4.3.3. Rad algoritma ilustriran je na sl. 6.1, gdje su analizom ulaznih slika (a) i (c) dobiveni položaji uzoraka u binariziranim slikama (b) i (d), u kojima su zanimljiva područja označena bijelom bojom. U tim slikama, izlučene regije su označene težištima, identifikacijskim brojevima i slikovnim oknima, susjedne regije pronađenog uzorka su povezane dužinama, dok su u lijevom donjem kutu upisane važeće matrice homografije iz k.s. uzorka u k.s. slike. Glavno svojstvo predloženog algoritma je optimalnost postupka za početno lociranje uzorka kojeg se želi analizirati. Neovisno o redoslijedu analize izlučenih regija, algoritam će mrežu uzorka početi graditi upravo od one regije čija "baza" ima najveći potencijal u smislu broja značajki izgrađene mreže. Barem na prvi pogled međutim, postoji nekoliko mogućnosti za neispravan rad algoritma, a one će se razmotriti u nastavku teksta.



(a) (b)

(c) (d)

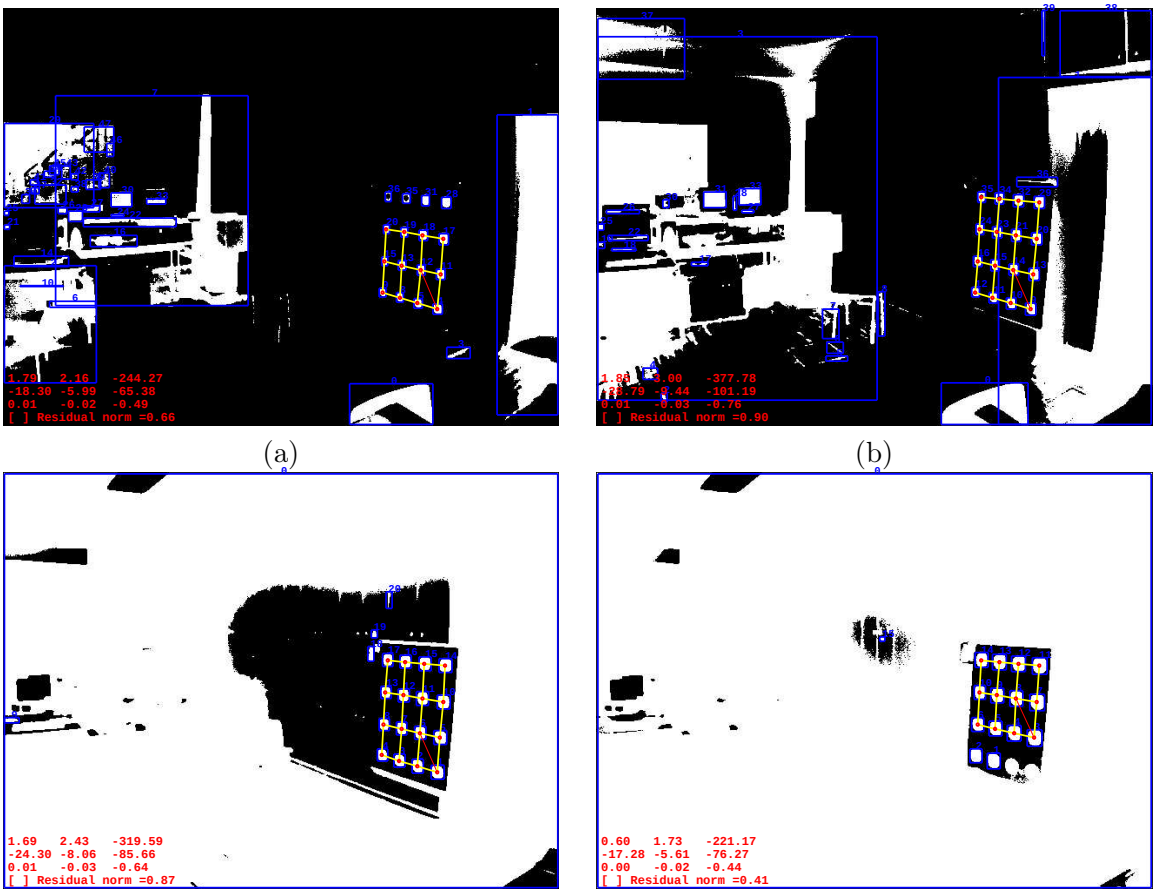
Slika 6.1: Pronalaženje uzorka u slučajevima kad je uzorak postavljen vodoravno (a–b), odnosno okomito (c–d).

Prvi potencijalni problem u vezi s opisanim algoritmom jest bojazan da bi njegova praktična vrijednost bila umanjena slabom performansom zbog dosta velike složenosti ocjene potencijala baze uzorka koja se mora obaviti počevši od svake izlučene regije. Pokazalo se još jednom da je (autorova) intuicija nepouzdan pokazatelj odnosa složenosti algoritama, te da je takva razmatranja potrebno vršiti nad čvrstim eksperimentalnim podatcima kad god je to moguće. Tri su glavne cjeline razmatranog algoritma: (i) binarizacija, (ii) izlučivanje binariziranih regija, te (iii) grupiranje izlučenih regija. Eksperimenti su pokazali da vrijeme izvršavanja pojedinih cjelina algoritma mnogo ovisi o ulaznoj slici, broju značajki mjernog uzorka te odabranom parametru binarizacije c. Stoga je usporedna analiza složenosti cjelina napravljena na temelju *najgorih* vremena u pojedinim cjelinama preko svih pribavljenih slika. Analiza pokazuje da se u kombiniranom najgorem slučaju 5% vremena izvođenja algoritma potroši na binarizaciju, 55% vremena na rast područja, te samo 40% vremena na grupiranje. Time postaje jasno da grupiranje regija nije najintenzivnija cjelina algoritma što znači da su bojazni s početka paragrafa neutemeljene. Trajanje obrade sive slika dimenzija 640×480 na radnoj stanici Asus A7M266–D tipično traje oko 200 ms (zbog sekvencijalne prirode postupka koristi se samo jedan procesor), što odgovara performansi od 5 Hz koja omogućava obavljanje umjeravanja u stvarnom vremenu. Drugi potencijalni problem predstavlja mogućnost neželjene detekcije neke pravilne strukture iz pozadine okoliša u kojem se umjeravanje odvija. Pokazalo se međutim da su u stvarnim slikama laboratorijskih okoliša pravilne dvodimenzionalne strukture vrlo rijetka pojava, jer dobiveni rasporedi značajki iz okoliša nikad nisu omogućili izgradnju mreže s više od šest značajki, a najčešće niti toliko.

Očito je stoga da već odabir mjernog uzorka s 16 značajki onemogućava neželjene pozitivne rezultate, barem što se tiče razmatranih okoliša.

Konačno, možda najslabija točka algoritma je njegova ovisnost o parametru binarizacije koji bi trebao osigurati "vidljivost" svih značajki mjernog uzorka i nakon binarizacije ulazne slike. Prva pretpostavka da bi se išta moglo zaključiti na temelju binarizacije jest postojanje jakog kontrasta između crnih i bijelih regija uzorka. U praksi, ta pretpostavka se ostvaruje korištenjem laserskog pisača za ispis crnih točaka na bijeloj pozadini, pa ona ne predstavlja problem, za ne pretjerano loše osvijetljene dijelove scene. Nadalje, osvjetljenje cijelog uzorka trebalo bi biti ravnomjerno, jer u ekstremnom suprotnom slučaju ne bi bilo moguće odabrati jedinstveni prag koji bi bio prikladan za cijelu sliku. U praksi, ovakav problem se rijetko javlja, i to uglavnom kod umjeravanja unutrašnjih parametara kada uzorak zauzima veći dio slike. Međutim, umjeravanje unutrašnjih parametara je općenito lakši problem jer pruža veće mogućnosti za kontrolu uvjeta osvjetljenja jer je uzorak tada bliže kameri.

Posljednja karika u lancu uspješne binarizacije je prag th koji se dobiva na temelju ulaznog parametra $c \in [0, 1]$, prema (4.81). Pokazuje se da se u velikom broju slučajeva zadovoljavajuća binarizacija slike postiže odabirom parametra $c = 0.3$, koji je korišten i pri dobivanju rezultata na sl. 6.1. Kod uobičajene rasvjete u dijelu scene u kojem se uzorak nalazi, eksperimenti su štoviše pokazali da je algoritam dosta robustan s obzirom na varijacije parametra c. To je ilustrirano na sl. 6.2, koja prikazuje rezultate dobivene uz četiri različite vrijednosti parametra c. Na priloženim slikama se vide dva oblika lošeg ponašanja algoritma uslijed neprikladnog parametra binarizacije. Ukoliko je prag prenizak kao na slici (a), neke



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

Slika 6.2: Osjetljivost algoritma za pronalaženje uzorka s obzirom na prag binarizacije: rezultati za ulaz sa sl. 6.1(c), ako je zadano c=0.1 (a), 0.2 (b), 0.5 (c) i 0.7 (d).

značajke bivaju samo djelomično vidljive i u binariziranoj slici, što je posebno izraženo kod značajki s bijelim krugom u sredini. Uslijed toga, težište značajke se pomiče od idealnog položaja pa se odgovarajući redak ne prepoznaje kao dio uzorka. Suprotno, u slici (d) gdje je prag prevelik, izlučene regije su nešto veće nego što bi trebale biti, dok je donja desna kutna regija u binariziranoj slici spojena s tamnom pozadinom u neposrednoj blizini uzorka. Treba primijetiti da je i pored pojedinačnih neuspjeha, algoritam u oba slučaja ipak uspio pronaći položaj nepotpunog uzorka u slici.

6.1.2 Umjeravanje unutrašnjih parametara

Metoda umjeravanja unutrašnjih parametara kamere opisana u 4.3.2, određuje matricu parametara perspektivnog modela $\mathbf{K}$, te četiri parametra modela radijalnog izobličenja k_1^{ud}, k_2^{ud}, k_1^{du} i k_2^{du}. Prvi način za ocjenu kvalitete tako dobivenih parametara temelji se na analizi prosječne i najveće kutne pogreške umjeravanja, koje se računaju prema izrazima (4.79) i (4.80), iz projekcijske greške dobivene u skladu s funkcijom cilja optimizacijskog postupka. Kvaliteta umjerenih parametara radijalnog izobličenja može se dodatno provjeriti i numerički, jer bi dobivene funkcije izobličenja odnosno korekcije na području slike trebale biti međusobno inverzne. Konačno, ocjena kvalitete parametara radijalnog izobličenja može se dati i subjektivnom metodom, usporedbom izobličenja u izvornoj i korigiranoj slici.

Analiza temeljena na projekcijskoj grešci funkcije cilja

Za ispravno umjeravanje parametara modela stvaranja slike, kritična je izrada kvalitetnog mjernog uzorka. Iako to prvenstveno vrijedi za metode koje podrazumijevaju 3D mjerni uzorak, izrada 2D uzorka također je vrlo osjetljiv zadatak, posebno u fazi lijepljenja uzorka na ravnu podlogu. Dokaz za to su rezultati umjeravanja u kojima projekcijska greška jako varira od uzorka do uzorka, kao što je prikazano na tablici 6.1. Tablica prikazuje projekcijsku grešku umjeravanja kamere Basler A301f s lećom `CNG 1.8/4.8-0301` tvrtke Schneider-Kreuznach, korištenjem triju mjernih uzoraka A, B i C, uz rastuću preciznost tehnike izrade. Stupci tablice redom sadrže oznaku uzorka, format papira na kojem je ispisan uzorak, broj korištenih slika uzorka (pogleda), broj uparenih značajki u svakoj od slika, prosječnu projekcijsku pogrešku, najveću pojedinačnu projekcijsku pogrešku i kut širine vidnog polja kao najizraženiji parametar kamere. Najbolji rezultati su dobiveni korištenjem uzorka C. Za taj uzorak,

uzorak	veličina	n_p	n_{qi}	$\bar{\delta}_{\text{fcn}}$	$\delta_{\text{fcn}}^{(\max)}$	θ_{hfov}
A	A4	4	140	0.64	1.67	64.01
B	A4	5	140	0.34	0.98	63.58
C	CD	5	100	0.20	0.69	63.47
A,B,C	A4,CD	14	100,140	0.48	1.69	63.82

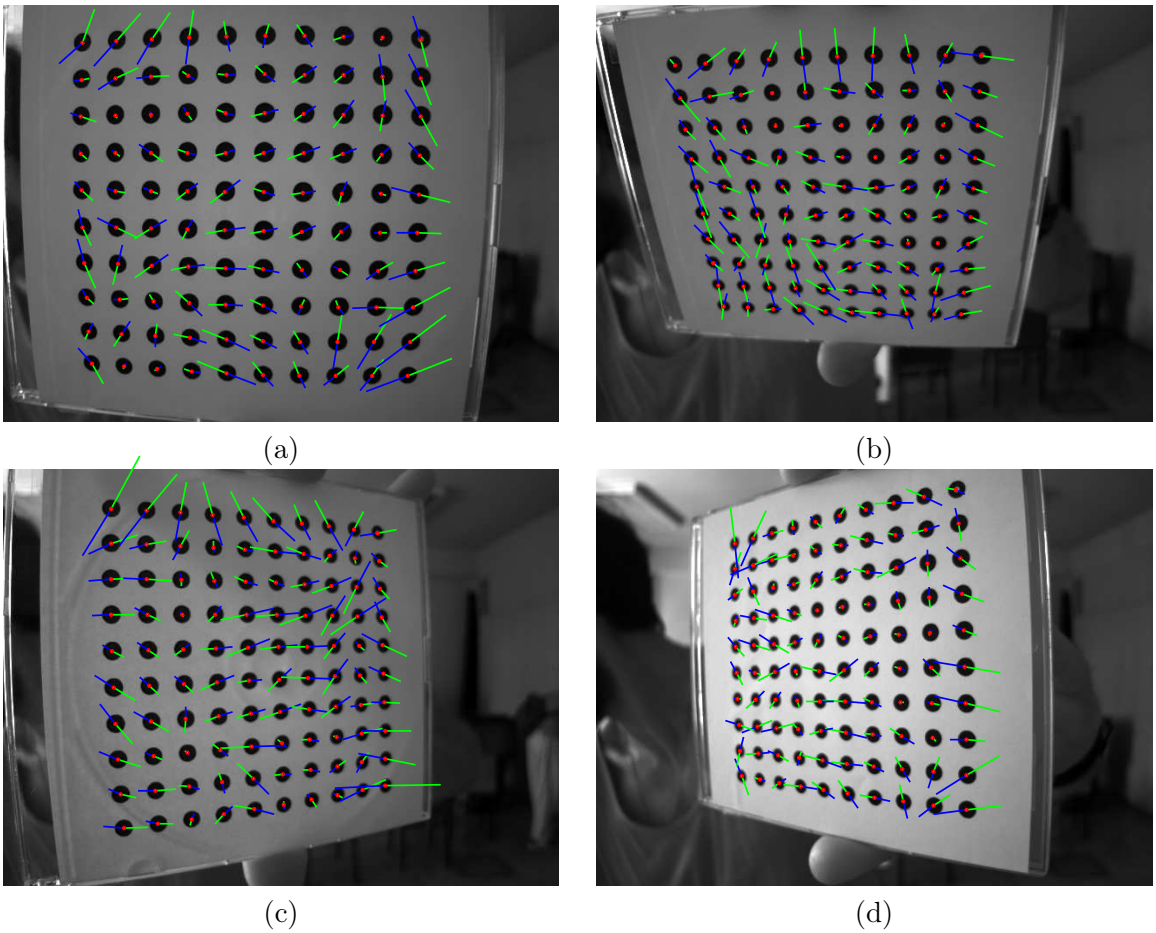
Tablica 6.1: Ovisnost devijacije o mjernom uzorku; A4=210×297 mm, CD=125×125 mm.

prema tablici 6.1, prosječna i maksimalna projekcijska pogreška u slikovnoj ravnini iznose 0.20 odnosno 0.70 slikovnih elemenata. Zbog zanimljivog slučaja da je vodoravna dimenzija slike po iznosu oko deset puta veća od kuta širine vidnog polja kamere, odgovarajuće kutne pogreške u stupnjevima su oko deset puta manje, 0.022° odnosno 0.078°. Dobiveni kutevi odgovaraju pomacima od 3.8 cm odnosno 13.6 cm na udaljenosti od 100 m. Usporedbe radi, isti eksperimenti su obavljeni bez nelinearne optimizacije, dakle bez modela radijalnog izobličenja. Dobivene projekcijske pogreške u tom slučaju su znatno veće, pa prosječna i

maksimalna kutna pogreška iznose 0.26° odnosno 0.8°, a to odgovara pomacima od 45 cm odnosno 140 cm na udaljenosti od 100 m.

Zanimljivo je da je kompenzacija homografijskog odstupanja središta krugova mjernog uzorka prema (4.55) u obavljenim eksperimentima bila toliko mala, da praktično nije utjecala na rezultat. Projekcijska izobličenja se javljaju samo u slučajevima kada se točke istog objekta scene istovremeno javljaju i u neposrednoj blizini, i vrlo daleko od kamere, kao npr. u slici željezničke pruge ili u zamišljenoj sceni koja bi se mogla projicirati u sl. 4.6(b).

Konačno, razmatrano je i kretanje smjera i iznosa projekcijskih pogrešaka u značajkama kalibracijskog slijeda slika u ovisnosti o položaju značajke mjernog uzorka, kao što je prikazano na sl. 6.3 za četiri slike iz slijeda uzorka C. U svim mjernim značajkama četiriju slika, ucrtani su modulirani vektori projekcijskih pogrešaka $100 \cdot \delta_{ij}^d$ i $-100 \cdot \delta_{ij}^u$, pri čemu su δ_{ij}^u i δ_{ij}^d definirani u (4.76). Protufazna modulacija (množenje sa 100 odnosno -100) je potrebna da bi vektori uopće bili vidljivi u slici, te da se ne bi međusobno prekrivali. Vizualna analiza pojedinačnih pogrešaka pokazuje da su odstupanja veća na periferiji slike gdje je ukupan broj bliskih projiciranih značajki uzorka manji. Iako se čini da raspored pogrešaka pokazuje stanovitu ovisnost o položaju mjerne značajke (primjerice, greške značajki najdesnijeg stupca uzorka su gotovo uvijek usmjerene u lijevo), uočene pravilnosti nisu ukazale na vidljive propuste u izradi uzorka.



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

Slika 6.3: Ovisnost projekcijske pogreške o položaju projiciranih značajki uzorka.

Numerička ocjena preciznosti parametara radijalnog izobličenja

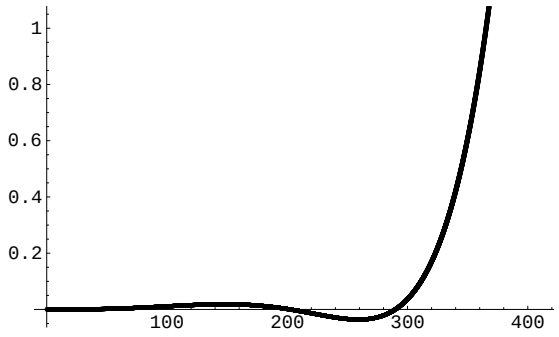
Prisjetimo se jednadžbi modela radijalnog izobličenja:

$$\begin{aligned}\hat{\mathbf{q}}_d &= [\hat{x}_d, \hat{y}_d, 1]^T = [f_{ud}(\hat{x}_u), f_{ud}(\hat{x}_u), 1]^T = \mathbf{f}_{ud}(\hat{\mathbf{q}}_u),\\ \hat{\mathbf{q}}_u &= [\hat{x}_u, \hat{y}_u, 1]^T = [f_{du}(\hat{x}_d), f_{du}(\hat{x}_d), 1]^T = \mathbf{f}_{du}(\hat{\mathbf{q}}_d).\end{aligned} \tag{6.1}$$

Uvrštavanjem izraza za $\hat{\mathbf{q}}_u$ u izraz za $\hat{\mathbf{q}}_d$, dobiva se: $\hat{\mathbf{q}}_d = \mathbf{f}_{ud}(\mathbf{f}_{du}(\hat{\mathbf{q}}_d))$, što znači da je f_{ud} inverzna funkcija f_{du}, odnosno da vrijedit $f_{ud} = f_{du}^{-1}$. Mjera odstupanja od tog odnosa može se izraziti formulom:

$$f_\varepsilon(w) = w - f_{du}(f_{ud}(w)). \tag{6.2}$$

Za nelinearne parametre dobivene analizom slijeda od pet slika uzorka C, graf ovisnost f_ε o w dan je na sl. 6.4. Korištena kamera je kalibrirana za sliku 640×480, što znači da su koordinatne vrijednosti svih vidljivih točaka slikovne ravnine manje od 320. Iz prikazanog grafa se stoga vidi da su umjerene funkcije doista vrlo blizu tome da budu inverzne unutar slike. Izvan područja slike međutim, odnos funkcija se rapidno kvari, što je bilo i za očekivati s obzirom na to da se sve vidljive projekcije mjernih značajki nužno nalaze unutar slike.



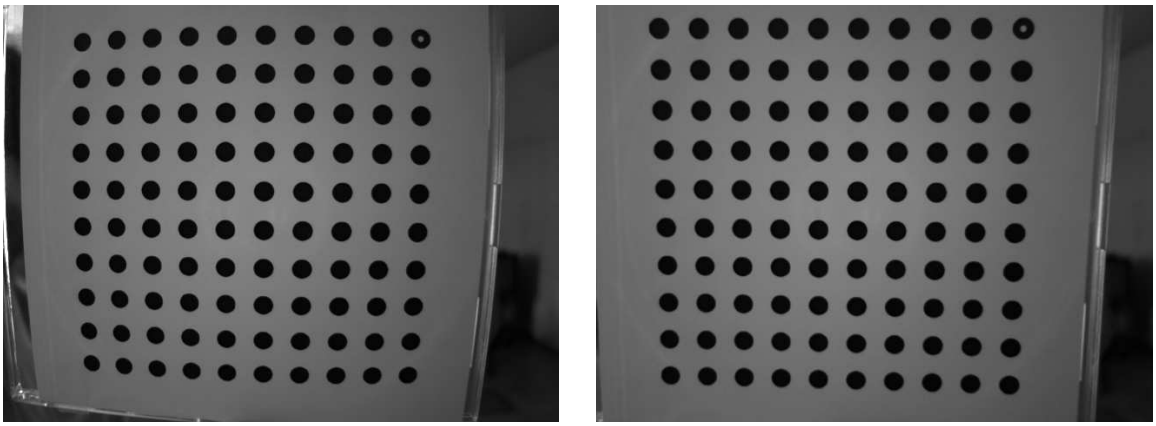
Slika 6.4: Preciznost parametara radijalne distorzije kao graf $f_\varepsilon(w) = w - f_{du}(f_{ud}(w))$.

Provedba korekcije radijalne distorzije

Da bi se provela korekcija radijalne distorzije, potrebno je znati sve unutrašnje parametre kamere, zbog činjenice da se radijalni model primijenjuje u normaliziranim slikovnim koordinatama, u kojima jedinica po obim osima slike odgovara žarišnoj daljini. Korekcija se može provoditi na dva temeljna načina, a svaki od njih ima i prednosti i nedostatke.

U prvom scenariju, korekcija se obavlja nad čitavom slikom. Primjena takvog postupka ilustrirana je na sl. 6.5, gdje se ulazna slika s vidljivim bačvastim radijalnim izobličenjem transformira u sliku u kojoj su izobličenja subjektivno potpuno nezamjetna. Vrijednosti elemenata korigirane slike $\mathbf{q}_u$ računaju se bilinearnom transformacijom, u ovisnosti o četiri elementa izobličene slike koji se nalaze u neposrednoj blizini odgovarajuće izobličene točke $\mathbf{f}_{ud}(\mathbf{q}_u)$. Glavni nedostatak ovog scenarija je velika računska složenost, uslijed koje obrada sive slike 640×480 na radnoj stanici Asus A7M266–D traje oko 100 ms (zbog sekvencijalne prirode postupka koristi se samo jedan procesor). S obzirom na to da se glavna obrada treba dogoditi tek nakon korekcije radijalnih izobličenja, ovakav pristup će se kod primjena u stvarnom vremenu moći koristiti tek u daljoj budućnosti.

U drugom scenariju, segmentacija slike se obavlja u izobličenoj slici, dok se tek nakon toga položaji zanimljivih značajki interpretiraju u neizobličenim koordinatama. Ovim postupkom



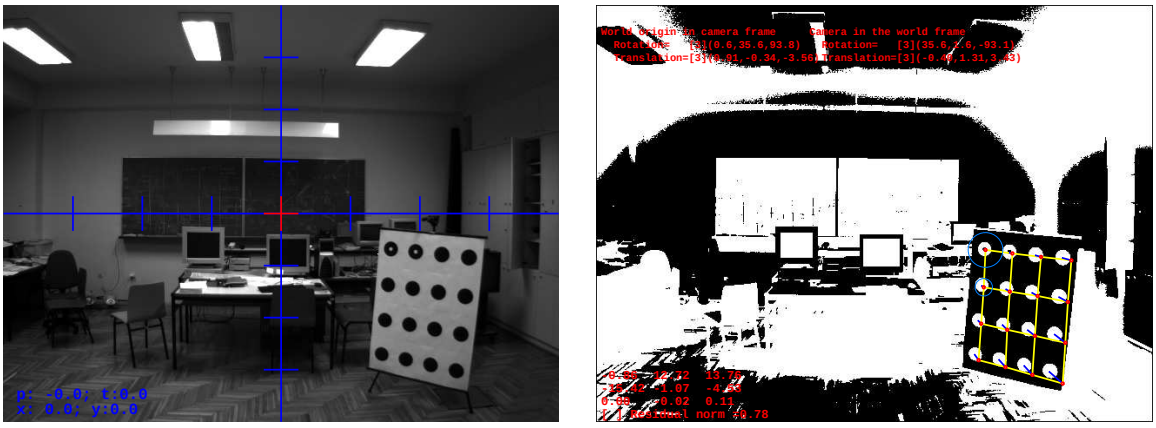
(a) (b)

Slika 6.5: Korekcija radijalnog izobličenja: ulazna slika (a) i rezultat (b).

se stoga korigiraju samo neki elementi izobličene slike $\mathbf{q}_d$, i to izravnom primjenom funkcije $\mathbf{f}_{du}(\mathbf{q}_d)$. Ovakav pristup je neusporedivo brži od prethodnog, pa je u kontekstu eksperimentalnog višeagentskog sustava korišten u izvedbi agenta promatrača, te u sklopu procedure za umjeravanje vanjskih parametara kamere kao što je prikazano na sl. 6.6. Glavni nedostatak ovakvog pristupa je ograničeno područje primjene jer ne može pomoći pri detekciji "velikih" značajki (npr, dugih pravocrtnih bridova).

6.1.3 Umjeravanje vanjskih parametara kamere

Umjeravanje vanjskih parametara više kamera upotrebom zajedničkog mjernog uzorka, moguće je obaviti prema postupku opisanom u 4.4.1. U obavljenim eksperimentima, korišten je mjerni uzorak dimenzija 840×1188 mm, dobiven spajanjem 16 listova papira formata A4, na svakom od kojih je ucrtana po jedna mjerna značajka. Dogovorno, ishodište k.s. uzorka je desno orijentirano, a nalazi se u onom kutnom elementu uzorka koji je označen bijelim krugom u sredini crnog kruga (drugi tako označeni element u ovoj verziji postupka nema nikakav posebni značaj). Do sada razmatrani k.s. kamere sa sl. 4.3 je bio lijevo orijentiran što je najlakše promijeniti invertiranjem glavne osi projekcije, tako da se pozitivne vrijednosti koordinate z sada nalaze iza kamere. Ovu promjenu je moguće i formalno ozvaničiti promjenom predznaka trećeg dijagonalnog elementa matrice $\mathbf{S}$ u jednadžbi (4.27), uz što sva dosadašnja razmatranja ostaju valjana. Tipičan primjer korištenja procedure prikazan je na sl. 6.6 u kojem mjerni uzorak zauzima oko 1/16 vidnog polja kamere. Procedura pronalazi parametre Euklidskih transformacija koje povezuju ishodišta k.s. uzorka i k.s. kamere. Svaka od te dvije transformacije ima šest stupnjeva slobode, koji su navedeni u gornjem dijelu odredišne slike. Na temelju primjene procedure na samo jednoj slici, može se zaključiti da se ishodište mjernog uzorka nalazi oko 3.5 m ispred, oko 90 cm desno, i oko 35 cm ispod trenutnog položaja kamere. Naravno, za ispravan rad procedure moraju biti zadovoljeni zahtjevi da je mjerni uzorak strogo planaran, te da su njegove dimenzije poznate s velikom preciznošću. Rezultati preliminarnih eksperimenata su potvrdili mogućnost lociranja mjernog uzorka s centimetarskom točnošću, dok je preciznost dobivenih orijentacija uzorka od oko 10° nešto manje zadovoljavajuća. Detaljniji eksperimenti navedeni su u [latin04], gdje je razmatrano poboljšanje rezultata sužavanjem vidnog polja kamere (*engl.* zoom), u cilju povećanja prividne veličine mjernog uzorka



(a) (b)

Slika 6.6: Određivanje vanjskih parametara kamere: ulazna slika (a) i rezultati (b).

6.2 Praćenje pojedinačnim promatračima

Iako je praćenje objekata inače vrlo složen i ne u potpunosti riješen problem računarskog vida, ono u kontekstu ovog rada ima samo sporednu ulogu pribavljanja ulaznih podataka za kooperativnu porazdijeljenu obradu. Stoga su pri oblikovanju odgovarajućih postupaka postavljeni specifični zahtjevi. Najvažnije je bilo u zadanom relativno kratkom vremenu postići funkcionalnost i performansu koje bi omogućile obavljanje realističnih eksperimenata u porazdijeljenom okruženju. U skladu s tim nisu razmatrani ni zahtjevniji aspekti problema praćenja (npr, razmatranje dodira dvaju ili više objekata), ni teoretske osnove za daljnja poboljšanja (npr, bolja predikcija budućih položaja objekata korištenjem teorije vjerojatnosti).

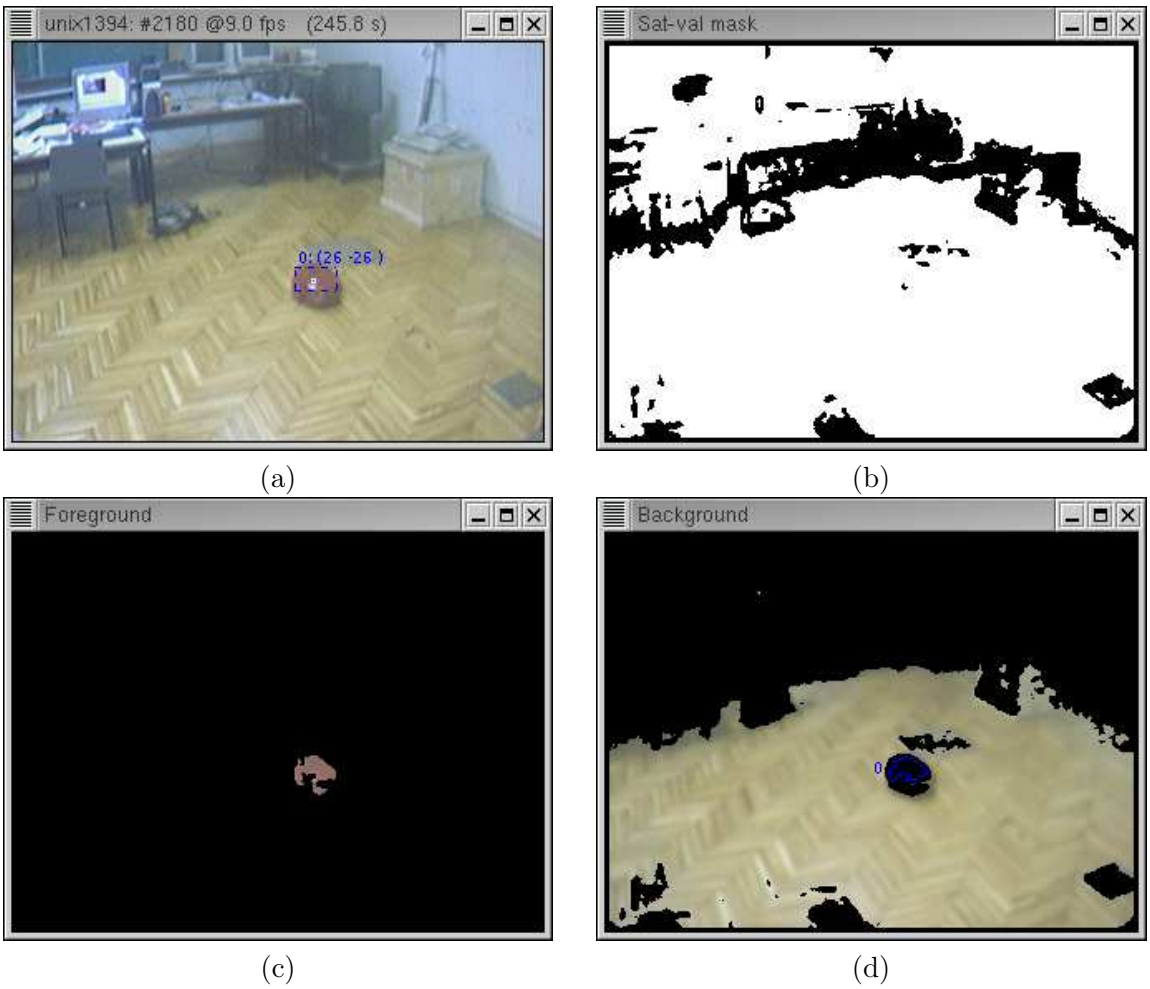
Kao što je navedeno u odjeljku 5.3, upotrijebljen je jednostavni pristup praćenju u dva međusobno neovisna koraka: prepoznavanje objekata, te praćenje projekcija njihovih težišta na ravninu gibanja. Prednost takvog pristupa je konceptualna jednostavnost koja se odražava u lakoći strukturiranja programske izvedbe, međutim ta jednostavnost ujedno i ograničava kvalitetu jer se kretanje ne koristi kao kriterij prepoznavanja. Metode koje pri prepoznavanju koriste kretanje načelno se zasnivaju na računanju optičkog toka što nije sasvim istraženo područje, pogotovo ukoliko je proračune potrebno obaviti u stvarnom vremenu. S obzirom da takve metode ne mogu pratiti objekte koji se kreću jako sporo ili stoje na mjestu, idealno rješenje bi vjerojatno podrazumijevalo kombinaciju oba temeljna pristupa.

Odjeljak je organiziran prema koracima odabranog pristupa praćenju objekata. Eksperimentalni rezultati sa stanovišta prepoznavanja objekata u pojedinačnim slikama stoga su opisani u prvom dijelu odjeljka, dok se u drugom dijelu opisuju eksperimenti s praćenjem objekta kroz slijed od više slika.

6.2.1 Prepoznavanje objekata

Najčešće korištena metoda prepoznavanja objekata s ciljem njihovog praćenja zasniva se na oduzimanju pribavljene slike od posebno generirane slike pozadine scene. Iako se radi o teoretski vrlo jednostavnom konceptu, u praksi je potrebno razraditi postupak prilagođavanja modela pozadine osvjetljenju scene, što može biti dosta zahtjevan problem. Stoga je odabran još jednostavniji pristup u kojem se prepoznavanje objekata temelji na detekciji njihove boje u koordinatnom sustavu HSV. Pokazalo se da nijansa kao H komponenta sustava HSV daje relativno dobar opis boje objekata scene pri raznim uvjetima osvjetljenja, iako su dobivene siluete praćenih objekata najčešće daleko od savršenih. Zbog odbljesaka, neravnomjerne

obojenosti i površine kućišta, te nedovoljnog osvjetljenja i sjena, objekt se često manifestira kao više međusobno odvojenih regija. Takve regije se ponekad (ali ne i uvijek) mogu udružiti heurističkim postupcima, korištenjem kriterija veličine i bliskosti. Nedetektirano "pucanje" praćenog objekta stvara posebne probleme proceduri za praćenje objekata koja se onda mora baviti pojavom novog objekta koji će u skoroj budućnosti najvjerojatnije ponovo nestati. U najvećem broju slučajeva, postupci ipak uspijevaju ispravno klasificirati dovoljno velik dio površine objekta, tako da se objekt donekle nedvosmisleno može prepoznati kao dominantna regija klasificiranih slikovnih elemenata. Tijek prepoznavanja objekta na radnoj stanici Asus A7M266–D prikazan je na sl. 6.7. Originalna slika s ucrtanim slikovnim oknom



(a) (b)

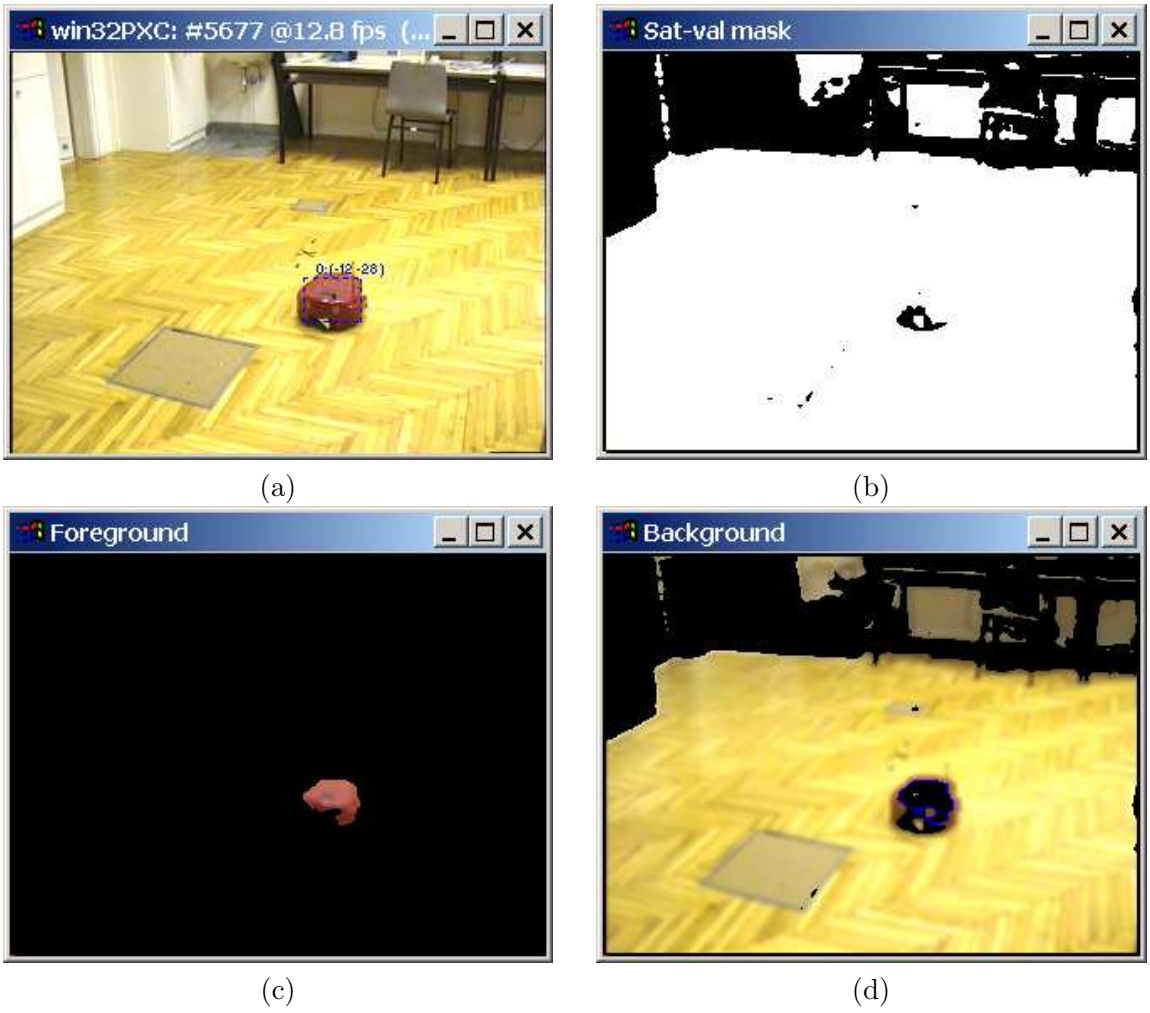
(c) (d)

Slika 6.7: Prikaz toka obrade promatrača koji se izvršava na radnoj stanici Asus A7M266–D.

i kutnim položajem detektiranog objekta prikazana je u okviru (a). Okvir (b) prikazuje binarnu sliku u kojoj su bezbojna i presvijetla područja detektirana na temelju premalih S odnosno V komponenti HSV sustava i označena crnom bojom. Vrijednost komponente H u odgovarajućim slikovnim elementima je gotovo proizvoljna, pa se te regije izbacuju iz daljnjih razmatranja upotrebom operacije maskiranja ulazne slike. Praćeni objekti se detektiraju u izglađenoj slici kao povezana područja slikovnih elemenata čija koordinata H odgovara apriorno zadanim očekivanim vrijednostima, a ta područja su prikazana u okviru (c). Konačno, razmotrena je i mogućnost izlučivanja ravnine gibanja prema njenoj žutoj boji, a postignuti rezultat je prikazan u okviru (d), zajedno s konturom izlučenog objekta.

Rezultati obrade promatrača koji se izvršava na radnoj stanici Compaq Evo W4000 prikazani su na sl. 6.8. Drugi promatrač promatra istu scenu, korištenjem kamere koja je

vidljiva u lijevom gornjem kutu sl. 6.7(a), na kućištu računala desno od monitora. Raspored okvira je identičan rasporedu za sl. 6.7 koji je opisan u prethodnom odlomku. Izlučena silueta na sl. 6.8(c) je dosta bolja od odgovarajuće siluete na sl. 6.7(c), zbog kvalitetnije kamere koja unosi manje šuma, što se može vidjeti i usporedbom izvornih slika koje se nalaze u okvirima (a) na sl. 6.7, 6.8.



(a) (b)

(c) (d)

Slika 6.8: Prikaz toka obrade promatrača koji se izvršava na računalu Compaq Evo W4000.

Čest problem kod računarskog vida je da ambijentalno osvjetljenje koje sasvim zadovoljava kriterije ljudskog oka nije dovoljno dobro da pribavljene slike mogu biti uspješno interpretirane strojnim metodama. Poseban problem u tom smislu stvaraju veliki kontrasti i odbljesci uzrokovani izravnim osvjetljenjem, te kutovi scene koji su često slabije osvijetljeni. Velik dio problema pri tome je zapravo u domeni senzora koji ne mogu prikazati velike kontraste nego ulaze u zasićenje prema bijeloj odnosno crnoj boji. Tako se u slikama eksperimenata može vidjeti da ni objekt ni referentna ravnina nisu izlučeni idealno, jer se u slikama sl. 6.7(c,d) javljaju crna područja koja nisu ispravno prepoznata. Ipak, ako se izuzmu sporadični neuspjesi, regularno osvjetljenje je uglavnom dovoljno za prepoznavanje crvenih objekata u pojedinačnim slikama zatvorenog prostora, uz korištenje opisanih postupaka i navedene opreme.

6.2.2 Praćenje objekata

U kontekstu praćenja, vrlo važno svojstvo svakog od promatrača je njegova performansa koja se obično iskazuje kao broj obrađenih slika u sekundi, a pri tome se koristi mjerna jedinica Hz ili fps (*engl.* frames per second). Agenti promatrači rade u stvarnom vremenu, pa kvaliteta praćenja umnogome ovisi o performansi odgovarajućih programskih izvedbi, jer učestalost obrade slika odgovara učestalosti uzorkovanja scene. Performanse agenta promatrača na različitim računalima eksperimentalnog sustava su sažete u tablici 6.2.

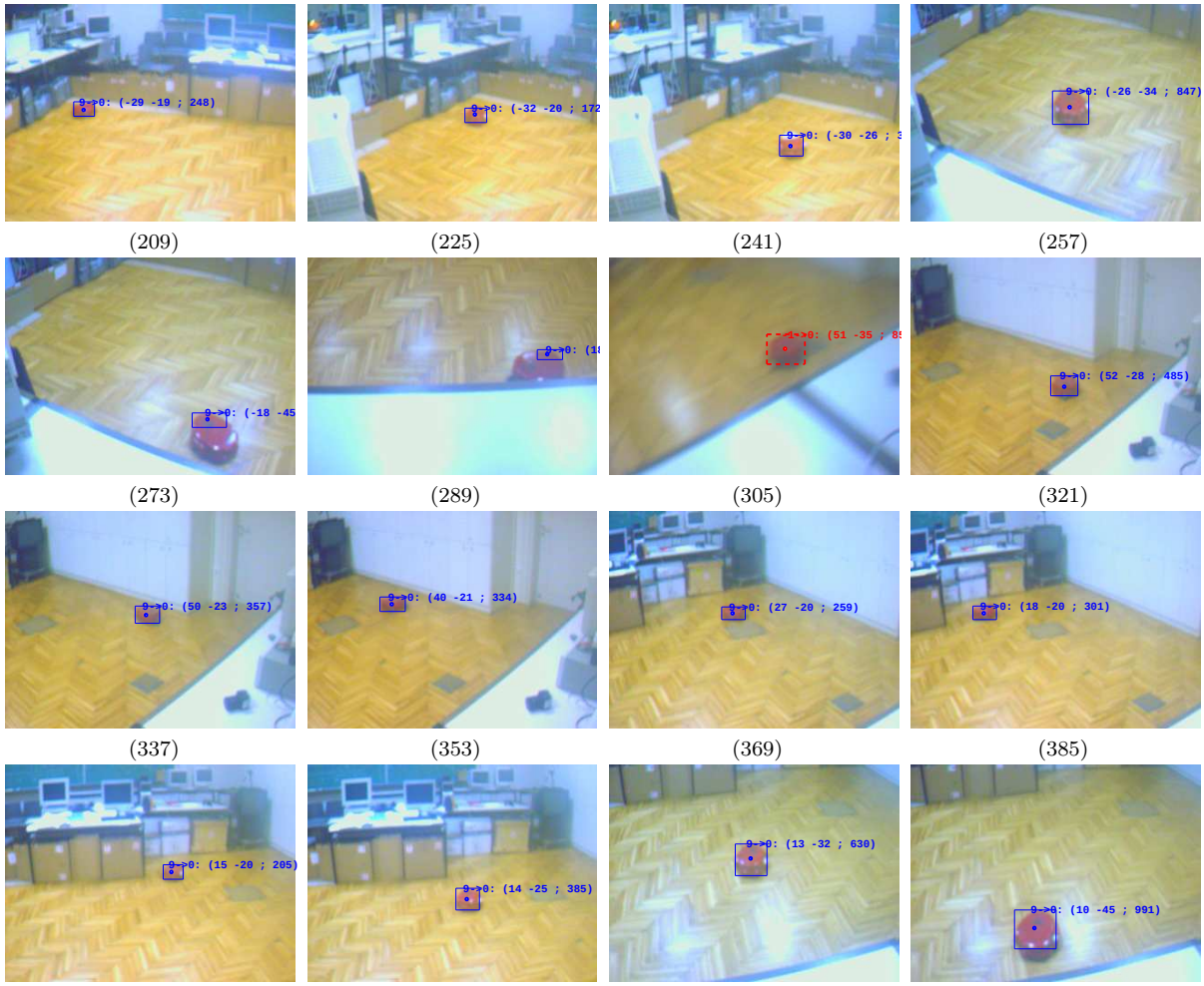
računalo	performansa [Hz]
Asus A7M266–D	9.1
Compaq Evo W4000	12.5
Matsonic 7127C	3.1

Tablica 6.2: Performanse promatrača na različitim računalima eksperimentalnog sustava.

Praćenje objekata se obavlja na temelju kriterija prostorne bliskosti težišta prepoznatih objekata u tekućem i prethodnim okvirima, postupkom koji je opisan u 5.3.5. Praćenje je ispitivano na scenama u kojima se jedan robot tipa AmigoBot nasumično kreće po prostoriji u kojoj su raspoređeni promatrači. Glavni problem pri praćenju predstavlja sporost upravljivih kamera Sony EVI D-31 što znatno otežava praćenje brzih objekata. Postolje PTU-46-17.5 omogućava puno okretnije upravljanje, dovoljno brzo za nesmetano praćenje robota AmigoBot čija maksimalna brzina iznosi 1 m/s.

Postignuti rezultati su ilustrirani slijedovima obrađenih slika na dva različita računala. Slijed slika promatrača koji se izvodi na radnoj stanici Asus A7M266–D prikazan je na sl. 6.9. Praćenje je dosta pouzdano unatoč slabijoj kameri koja pruža sliku s relativno mnogo šuma, a najveći problemi se javljaju kada se robot nalazi u položaju u kojem reflektira stropno osvjetljenje. Ispod svakog okvira na sl. 6.9 naveden je redni broj odgovarajuće slike u originalnom slijedu. Razmak između okvira je 16 slika izvornog slijeda što odgovara vremenu od oko 1.8 s. Prepoznati objekt je u svakom okviru označen slikovnim oknom i tekstom oblika `p`→`q`: (ϕ θ; `A`), koji ukratko opisuje stanje praćenja objekta. Kutevi ϕ i θ označavaju zakret i nagib pravca od glavne točke projekcije kamere prema težištu objekta u stupnjevima, u odnosu na smjer optičke osi kad su zakret i nagib pokretne kamere jednaki nuli. Površina `A` označava broj slikovnih elemenata regije koja je prepoznata kao objekt. Relacija `p`→`q` označava uparivanje koje je algoritam praćenja pronašao između do sada praćenog objekta `p`, i objekta prepoznatog u tekućem okviru `q`. Identifikator praćenog objekta se kroz slijed ne mijenja (uvijek je jednak 9), što znači da je algoritam praćenja ispravno zaključio da se u cijelom slijedu javlja samo jedan objekt. Zbog eksperimentalnog programskog sučelja za pribavljanje slike preko sabirnice IEEE 1394 na operacijskom sustavu Linux, kompenzacija latencije pribavljenih slika na način opisan u 5.1.2 nije implementirana, pa se kamera pokreće u inkrementalnim pomacima, a praćenje se ne obavlja dok pomaci traju. Takva situacija se vidi u okviru 305 na sl. 6.9: `p` iznosi `-1`, dok je slikovno okno objekta označeno isprekidanom crtom, jer se praćenje ne može obaviti dok se kamera giba zbog nepoznavanja latencije pribavljenih slika. To ne stvara probleme jer zbog velike brzine postolja pomaci traju oko 1/4 s, pa se za vrijeme pomaka propušta praćenje rezultata obrade samo dvije slike.

Slike iz slijeda koji je zabilježen tijekom obrade na radnoj stanici Compaq Evo W4000 prikazane su na sl. 6.10. Zbog sporije brzine integriranog upravljivog postolja, praćenje je dosta slabije u odnosu na prethodnu konfiguraciju, unatoč boljoj kameri. Oznake u pojedinim okvirima slike odgovaraju navedenom u opisu sl. 6.10. Razmak između okvira na sl. 6.10 je oko 10 slika izvornog slijeda što odgovara vremenu od oko 0.8 s. Iako pripadni digitalizator

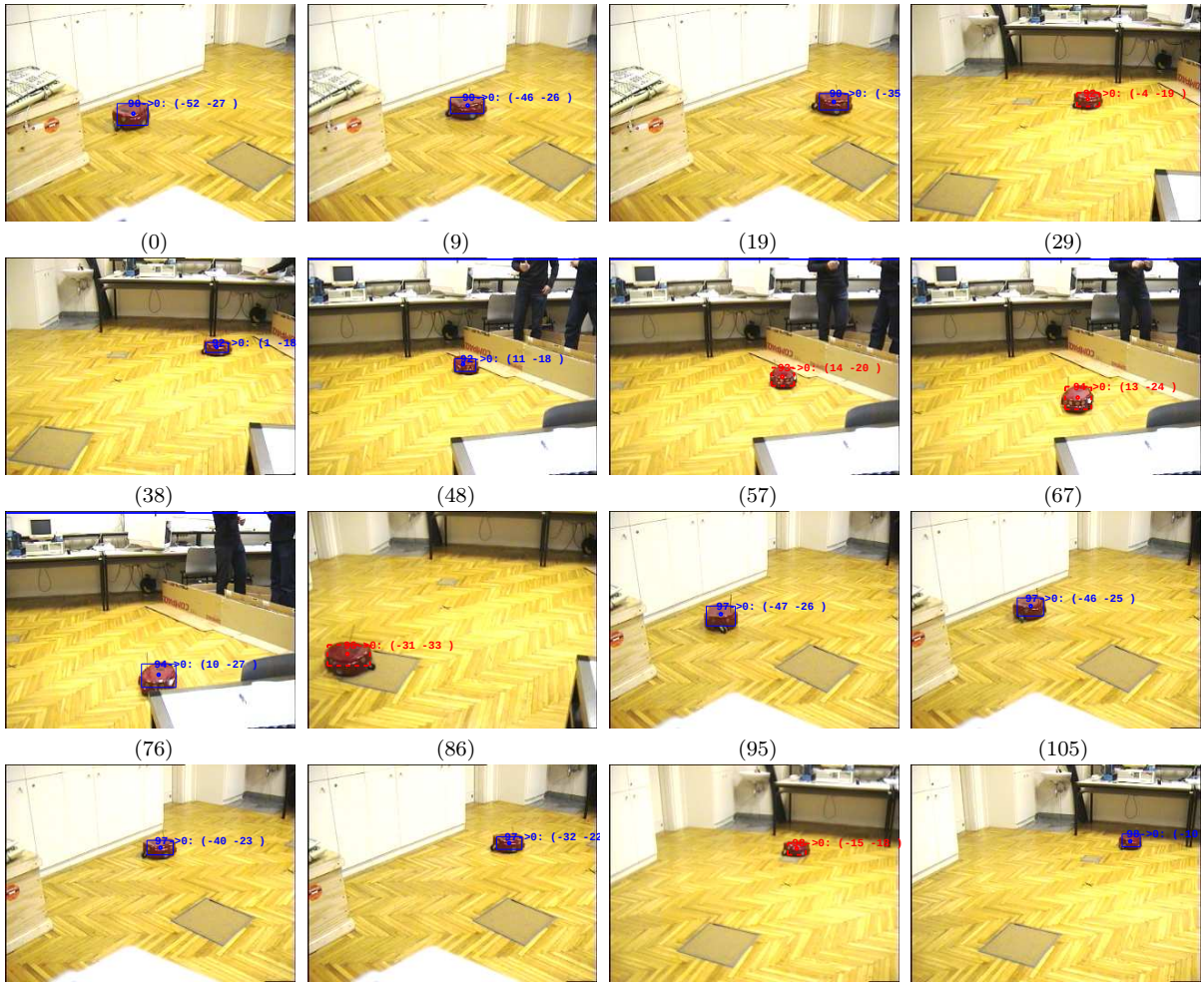


Slika 6.9: Slijed obrađenih slika na računalu Asus A7M266–D. Brojevi u zagradama označavaju redni broj slike iz pribavljenog slijeda.

slike omogućava ocjenu latencije pribavljenih slika, obrada s pokretnom kamerom i dalje nije moguća zbog spore upravljačke veze čija najveća podržana brzina iznosi tek 9600 bps. Svaki pomak kamere traje nešto manje od 1 s, a tokom tog vremena praćeni objekt često pređe toliku udaljenost da ga postupak praćenja ne može svrstati u postojeću trajektoriju, pa se registrira nestanak "starog" i dolazak "novog" objekta. To se vidi na sl. 6.10, gdje se opisani scenarij dogodio u okvirima 29, 57, 67, 86 i 134. Vrijednost parametra `p` tijekom obrade prikazanog slijeda raste od `90` do `98`, što znači da se scenarij dogodio ukupno osam puta. Te situacije ipak ne utječu pogubno na rezultate porazdijeljenog praćenja, s obzirom na to da koordinator zbog uvida u cijelu scenu može tolerirati znatno veća odstupanja pojedinačnih mjerenja trajektorije.

6.3 Prostorna integracija pojedinačnih pogleda

Izvedba agenta koordinatora je isprobana u laboratorijskom okruženju, u konfiguraciji s tri agenta promatrača, koji se izvode na računalima Asus A7M266–D, Compaq Evo W4000 i Matsonic 7127C. Navedenim računalima su na lokalnoj TCP/IP mreži dodijeljena imena `dupli`, `ocalinko` i `misko`. Agent koordinator se pri tome izvodio na dvoprocesorskoj radnoj stanici Asus A7M266–D, zajedno s odgovarajućim agentom promatračem. Izmjerena per-



Slika 6.10: Slijed obrađenih slika na računalu Compaq Evo W4000. Brojevi u zagradama označavaju redni broj slike iz pribavljenog slijeda.

formansa promatrača je bila jednaka kao i u slučaju kada koordinatoru nije bilo dodijeljeno isto računalo.

Prostorna integracija kao najvažniji zadatak koordinatora je izgradnja ukupnog prikaza scene na temelju više parcijalnih pogleda, kao što je opisano u odjeljku 5.4. Preciznost postupka ovisi o točnosti vanjskih parametara kamere svakog od promatrača, pa je prije svakog eksperimenta parametre potrebno provjeriti zbog mogućnosti pomaka ili još gore rotacije kamera tijekom vremena. Nažalost, ograničeno vrijeme nije dozvolilo uhodavanje poluautomatskih metoda umjeravanja opisanih u 4.4.1, pa su vanjski parametri kamera određivani dugotrajnim ručnim premjeravanjem prostora. Stoga su parametri provjeravani ad-hoc metodom, u kojoj bi se objekt pomicao po ravnini gibanja, a uvidom u ukupni prikaz scene provjerilo bi se poklapanje položaja dojavljenih od pojedinih promatrača.

Rezultati postupka se prikazuju u stvarnom vremenu na računalu na kojem se izvodi agent koordinator. Kao i kod agenata promatrača, eksperimentalne rezultate je najzornije prikazati slijedom slika koje prikazuju kretanje rezultata postupka u vremenu. Takav slijed je prikazan na sl. 6.11, za eksperiment u kojem se dva pokretna objekta s brzinom od oko 0.5 m/s mimoilaze u dijelu scene kojeg pokrivaju sva tri promatrača. Svaki okvir slike prikazuje ukupni prikaz scene koji je određen i iscrtan od strane koordinatora u trenutku koji je označen ispod okvira, a vidljiv je i u donjem lijevom kutu unutrašnjosti okvira. U pozadini

svakog okvira su ucrtani tlocrt laboratorija i referentna kvadratna mreža dimenzija 1×1 metar. Pored toga, za svakog promatrača su posebnom bojom ucrtani sljedeći podatci:

- velika kružnica koja definira položaj promatrača, te vektor iz njenog središta koji definira referentni zakret promatrača (referentni nagib iznosi 0 stupnjeva);
- tekst u blizini položaja promatrača koji sadrži identifikacijski broj, mrežno ime i trenutne vrijednosti zakreta i nagiba pokretnog postolja u stupnjevima;
- četverokut koji definira granice vidnog polja promatrača;
- jedna veća i niz manjih kružnica, povezanih sivim dužinama i obojenih bojom promatrača, koje prikazuju posljednji položaj detektiranog objekta te slijed njegovih prošlih položaja u zadnje 2 s (ukupan broj povezanih kružnica podijeljen s dva stoga može biti interpretiran kao gruba ocjena performanse promatrača u Hz).

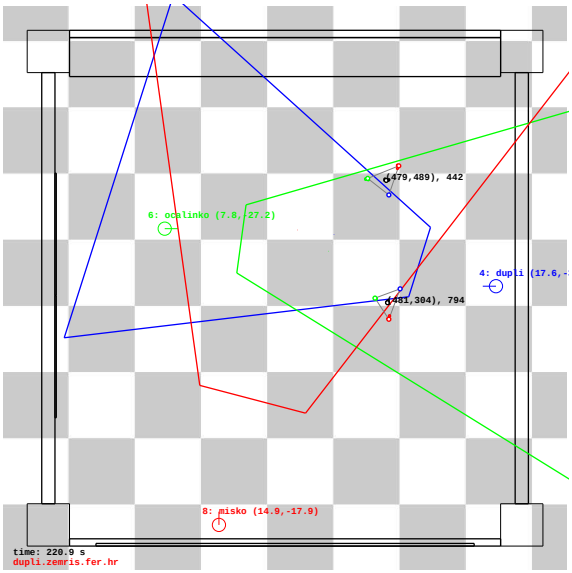
Za svaki percipirani objekt oglasne ploče, crnom bojom su prikazani sljedeći podatci:

- tekst koji označava koordinate posljednjeg položaja objekta u k.s. svijeta i njegovu procijenjenu površinu u cm^2;
- jedna veća i niz manjih crnih kružnica povezanih sivim dužinama, koje prikazuju trenutni položaj percipiranog objekta te odgovarajuću trajektoriju tokom posljednje 2 s.
- posljednja mjerenja u trajektorijama promatrača koje čine percipirani objekt povezana su sivim dužinama.

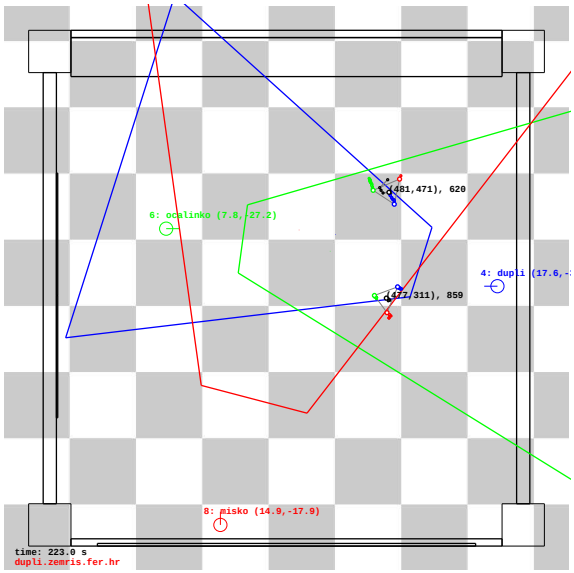
Treba napomenuti da tehnike umjeravanja parametara radijalnog izobličenja leće nisu korištene u eksperimentima koji se ovdje opisuju, jer nije bilo vremena za njihovu integraciju s ostalim komponentama eksperimentalnog sustava. Nedostatak kvalitetne kompenzacije radijalnih izobličenja se manifestira kao veliko odstupanje prijavljenih položaja u blizini rubova vidnog polja odgovarajućeg promatrača.

Prikazi obrade sa slika 6.9, 6.10 i 6.11, dostupni su i u elektroničkom obliku, kao animacije u formatu GIF kojeg zna prikazati svaki preglednik Interneta. Odgovarajuće datoteke su dostupne na sljedećim adresama:

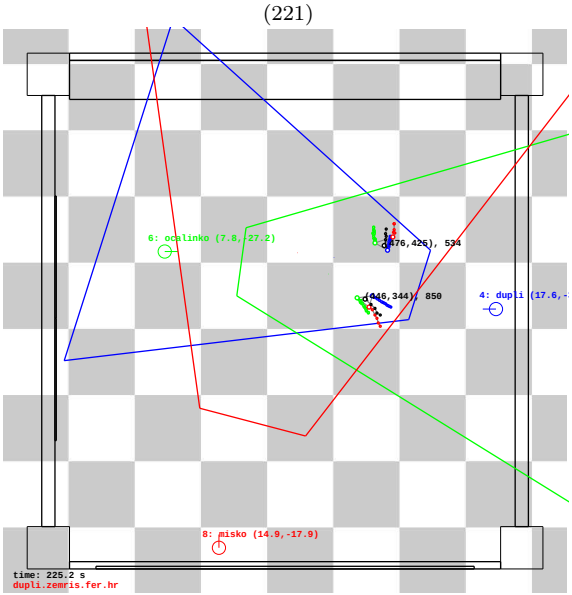
- `http://www.zemris.fer.hr/~ssegvic/macv/figures/results/1/unix1394.gif`
- `http://www.zemris.fer.hr/~ssegvic/macv/figures/results/1/win32PXC.gif`
- `http://www.zemris.fer.hr/~ssegvic/macv/figures/results/1/animatedWithBg.gif`



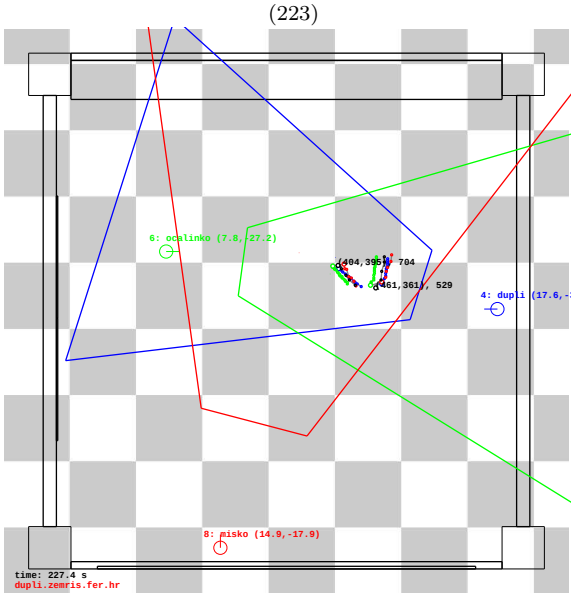
(221)



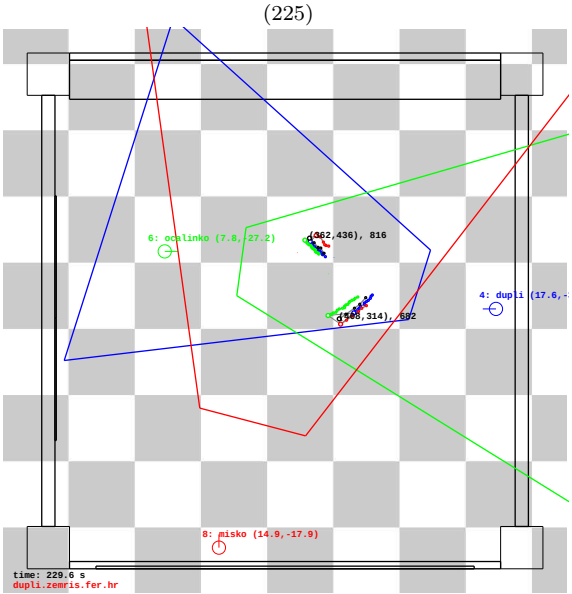
(223)



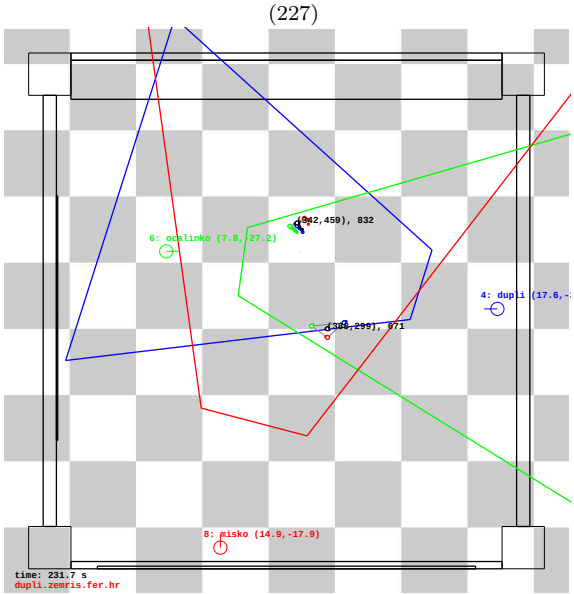
(225)



(227)

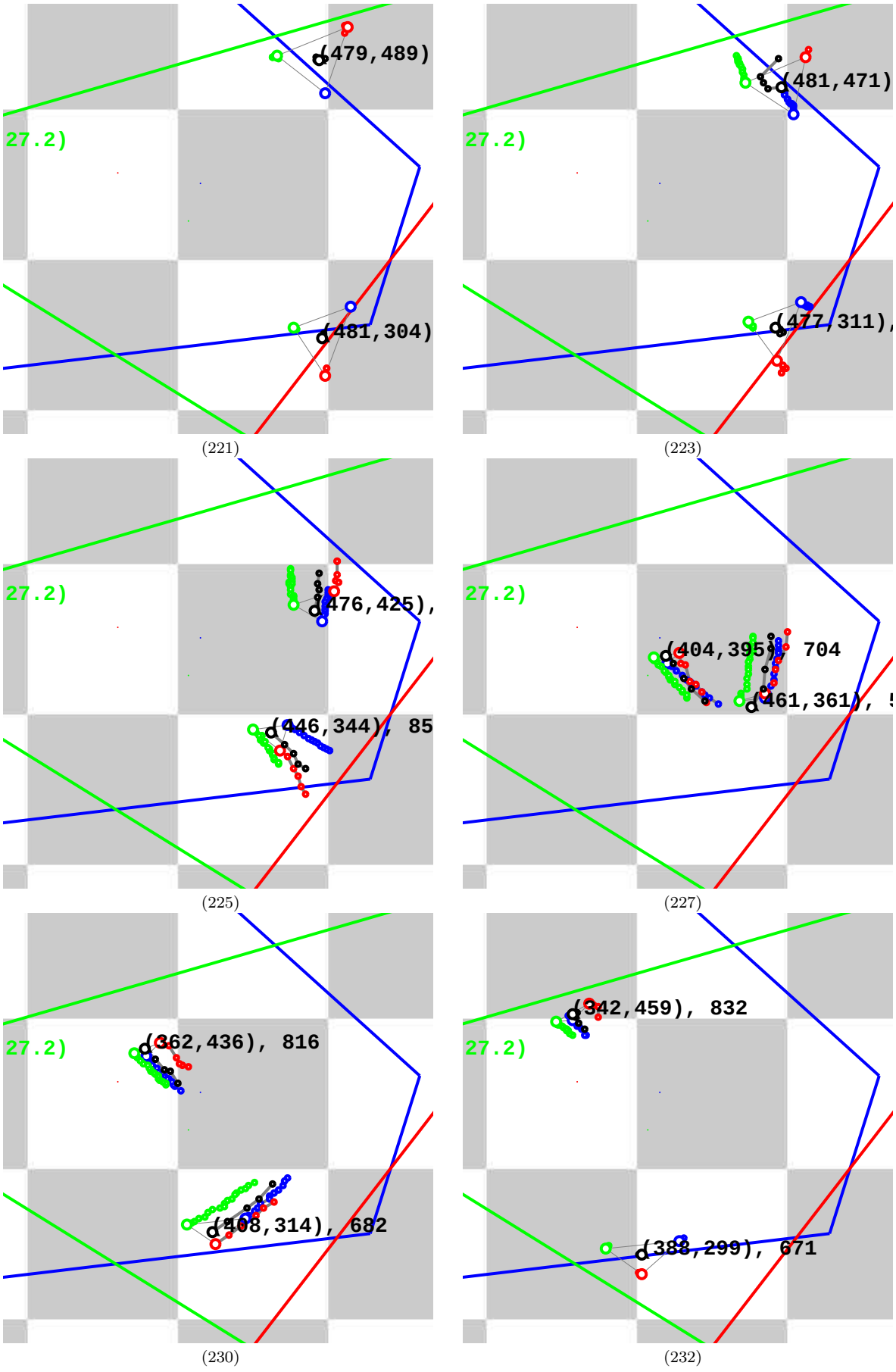


(230)



(232)

Slika 6.11: Slijed prikaza scene s dva objekta koji koordinator stvara i održava u stvarnom vremenu.



Slika 6.12: Slijed uvećanih prikaza sa sl. 6.11.

Poglavlje 7

Zaključak

Praćenje gibanja pokretnih objekata jedno je od najčešćih područja primjene računarskog vida. Velik broj konkretnih zadataka na tom području razmatra prostrane ili razvedene scene koje se ne mogu pokriti samo jednom kamerom. Ponekad se problemi vezani uz takve scene mogu ublažiti korištenjem aktivnih i panoramskih kamera, ali najopćenitije rješenje se postiže tek u porazdijeljenom sustavu, gdje se konačni odgovori dobivaju usaglašavanjem rezultata praćenja većeg broja strateški raspoređenih promatrača. U ovom radu, razmatrani su različiti načini organiziranja promatrača u porazdijeljenom sustavu. Posebna pažnja usmjerena je na kooperativne višeagentske arhitekture, u kojima su promatrači donekle autonomni a funkcionalnost se postiže kroz njihovu suradnju. Takve arhitekture su posebno prikladne kada promatrači raspolažu resursima kojima je potrebno upravljati u stvarnom vremenu, jer se strogo centralizirani pristupi u takvim situacijama suočavaju s nizom praktičnih i fizičkih ograničenja.

Razmatrane su tri temeljne klase promatrača: pasivni, aktivni i pokretni. Pasivni promatrači su opremljeni nepokretnim kamerama, aktivni promatrači mogu upravljati parametrima kamere kao što su smjer gledanja i širina vidnog polja, dok se pokretni promatrači izvode na robotskoj platformi pa se mogu kretati kroz scenu. Arhitektonski pristupi su vrednovani prema različitim kriterijima od kojih su najvažniji što bolje iskorištavanje specifičnih resursa aktivnih i pokretnih promatrača, te učinkovitost pri stvaranju i korištenju ukupnog prikaza scene. Od posebnog značaja za razmatrani problem su tipovi arhitektura temeljeni na fiksnoj i prilagodljivoj hijerarhiji. Ocijenjeno je da je prilagodljiva hijerarhija bolje rješenje kad veliki broj promatrača prati mali broj objekata u prostranoj sceni. Međutim, u realističnim slučajevima kada se promatrači kreću kroz izrazito strukturirane i razvedene prostore a zalihost promatrača nije velika, prednosti prilagodljive hijerarhije se uglavnom gube pa je jednostavniji pristup s čvrstom hijerarhijom prikladniji.

U sustavu za distribuirano praćenje ključna je usklađenost prostornih i vremenskih referentnih koordinatnih sustava sudionika. Dok je kod sinkronizacije satova problem moguće riješiti izmjenom niza poruka u toku normalnog rada, kod prostornih koordinata zadatak je znatno složeniji i mora se riješiti prije pokretanja sustava. Usklađenost referentnih prostornih koordinata pretpostavlja umjeravanje unutrašnjih i vanjskih parametara kamera promatrača, za što su korišteni postupci iz literature. Proračuni se posebno kompliciraju kod aktivnih promatrača kod kojih je pored modela kamere potrebno postaviti i model upravljivog postolja, te umjeriti odgovarajuće parametre.

Predložena je višerazinska hijerarhijska arhitektura za porazdijeljeno praćenje računarskim vidom, prikladna za široku klasu realističnih problema. Glavne značajke te arhitekture su (i) lokalno sažimanje informacija na svakoj razini hijerarhije, (ii) korištenje ukupnog prikaza scene pri koordiniranju aktivnih i pokretnih promatrača, te (iii) kratkoročne auto-

nomne te dugoročne centralizirane odluke. Funkcionalnost opisane temeljne koordinacijske procedure je usporediva s rješenjem koje je nedavno predloženo u [matsuyama02]. Međutim dostupnost ukupnog prikaza pruža potencijal i za sofisticiranija ponašanja, posebno u kombinaciji s održavanjem mape zaklonjenih dijelova scene za svakog od promatrača. Predložena arhitektura omogućava uklapanje pokretnih promatrača, pri čemu se njihov apsolutni položaj može odrediti na temelju opažanja nepokretnih promatrača sustava.

Razvijen je eksperimentalni višeagentski sustav koji obuhvaća funkcionalnost propisanu jednorazinskom inačicom predložene arhitekture. Dobiveni eksperimentalni rezultati potvrđuju da je predložena arhitektura valjan pristup za povezivanje potrebnih programskih komponenti na održiv, prilagodljiv i proširiv način. Pravci za budući rad uključuju kako proširenje eksperimentalnog sustava naprednijim značajkama predložene arhitekture, tako i nadgradnju postojeće arhitekture novim zamislima od kojih je možda najprivlačnija razrada zahtjevnijih taktika za koordinaciju promatrača.

Dodatak A

Programska arhitektura eksperimentalnog sustava

Izvedba eksperimentalnog sustava sastoji se od skupa programskih komponenti vlastite izrade koje izravno rješavaju pojedine aspekte željene funkcionalnosti i vanjskih biblioteka koje su opisane u 5.2.3. Sve komponente vlastite izrade su pohranjene u mrežnoj biblioteci verzija izvornog kôda koja se gradi i održava alatom `cvs`, koji je ukratko opisan u 5.2.4. Organizacija komponenti u biblioteci je hijerarhijska, pri čemu se organizacijske jedinice nazivaju paketima (*engl.* package) [lakos96]. Programska izvedba eksperimentalnog sustava sastoji se od pet izvršnih datoteka ili *ciljeva*, koji se grade od pojedinih paketa biblioteke.

Neka opća razmatranja o pisanju dokumentacije za velike programske sustave navedena su u odjeljku A.1. Raspodjela odgovornosti među paketima biblioteke razrađena je u odjeljku A.2, kao glavna značajka programske arhitekture eksperimentalnog sustava. Odjeljak A.3 detaljnije opisuje načine korištenja i proširivanja razvojne ljuske računarskog vida koja je temeljni cilj biblioteke. Konačno, kratak opis ostalih izvršnih datoteka dan je u odjeljku A.4.

A.1 Općenito o programskoj dokumentaciji

Pisanje programske dokumentacije je iznimno odgovoran i težak zadatak. Odgovoran je zato što kvaliteta dokumentacije u mnogome određuje upotrebljivost konačnog proizvoda, dok je težak jer napisati nešto o uspješnom velikom programu što neće zastariti prije izdanja konačnog dokumenta graniči s nemogućim. Brooks [brooks95] navodi tri metafore stvaranja programske podrške: pisanje (*engl.* writing), izgradnja (*engl.* building, construction) i uzgajanje (*engl.* growing). Prva metafora je prikladna za vrlo kratke programe koji se mogu napisati i ispitati za jedan dan. Druga metafora uspoređuje oblikovanje programa s graditeljstvom: zadatku se pristupa u nekoliko faza s čvrstim poretkom: izrada zahtjeva, oblikovanje projektne dokumentacije, izvedba fizičkih radova te dobivanje uporabne dozvole. Mnogi autori tvrde da je i ta druga metafora zastarjela, jer su konceptualne strukture koje je danas potrebno izraditi presložene da bi se točno unaprijed specificirale, i prezahtjevne da bi se realizirale bez greške [brooks95]. Dodatno, okoliši u kojima današnji programi trebaju funkcionirati tipično su iznimno dinamični, pa je, čak i u slučaju da je početna specifikacija bila potpuno ispravna u trenutku donošenja, vrlo malo vjerojatno da bi ona vrijedila i u trenutku isporuke, desetak mjeseci kasnije. Zato se predlaže nova metafora, u kojoj se razvoj programske podrške uspoređuje s uređenjem vrta [hunt99], gdje stvaraoc nema sve konce u rukama, nego naume i planove učestalo prilagođava razvoju događaja koje nije moguće unaprijed predvidjeti. U skladu s tom metaforom, oblikovanje počinje od apstraktnih ciljeva

i njihovih prioriteta, koji se tijekom razvoja revidiraju, u ovisnosti o uspjehu na pojedinim aspektima problema. Razvoj programske podrške je iterativni proces u kojem se funkcionalnost inkrementalno nadograđuje i koji gotovo nikad ne prestaje. Što je proizvod uspješniji, to je manje vjerojatno da će njegov razvoj stati, jer više korisnika znači veći pritisak za proširenje postojeće funkcionalnosti.

Biblioteka komponenti izvornog kôda eksperimentalnog programskog sustava sastoji se od preko 50 000 linija teksta i preko 100 000 riječi. Za usporedbu, izvorni kôd gotovo konačne verzije ovog rada bez slika sastoji se od nešto manje od 12 000 linija teksta, odnosno 50 000 riječi. Očito, postupak pisanja potpune dokumentacije velikog programskog sustava vrlo je smion poduhvat, koji enormno povećava ukupno vrijeme stvaranja proizvoda. Sveobuhvatnu dokumentaciju stoga sebi mogu priuštiti samo veliki razvojni timovi, dok je u ostalim slučajevima potrebno ocijeniti točku optimalnog odnosa uloženih resursa i vrijednosti dobivene dokumentacije.

U opisanom složenom i dinamičnom okruženju, dokumentacija mora naći teški kompromis između opisivanja pravog stanja na terenu i produljenja vlastitog roka trajanja. Mnoga eksperimentalna svojstva trebaju biti izuzeta jer je njih često najteže za opisati, a najveća je vjerojatnost da se u nekoj od budućih verzija korjenito promijene zbog novih nepredviđenih spoznaja. Slično, neka manje bitna odstupanja od idealnog ponašanja mogu biti preskočena jer je vrlo važno da se složeni odnosi izraze na najjednostavniji i najprirodniji način. Dokumentacija ujedno predstavlja i plan budućeg rada pa će dobro uobličene i obrazložene novosti brzo naći svoj put i do same izvedbe odgovarajućih komponenti sustava.

Tri dodatna poglavlja ovog rada A, B i C, predstavljaju pokušaj pisanja temeljne dokumentacije za skup programskih komponenti od kojih se sastoji eksperimentalni višeagentski sustav za porazdijeljeno praćenje. Ovaj dodatak opisuje temeljne arhitektonske značajke biblioteke komponenti na najapstraktnijoj razini. Dodatak B daje detaljniji prikaz komponenti za rad sa specifičnim sklopovljem. Konačno, s upotrebne strane, dodatak C opisuje sustav za automatizirano prevođenje komponenata biblioteke u ciljeve eksperimentalnog sustava, neovisno o ciljnoj platformi.

A.2 Arhitektura biblioteke izvornog kôda

Građa biblioteke se sastoji od 210 sučeljnih datoteka s ekstenzijom `.h` i 216 izvedbenih datoteka s ekstenzijom `.cpp`. Svaka izvedbena datoteka definira po jednu komponentu, te u pravilu ima odgovarajuću sučeljnu datoteku čiji naziv se od naziva izvedbene datoteke razlikuje samo po ekstenziji. Srodne grupe komponenata su organizirane u pakete, i smještene u zasebne direktorije prema imenu paketa. Neki veliki paketi su dodatno razvrstani u podpakete, i onda je svakom od njih dodijeljen zasebni poddirektorij. Zajednički dio naziva dviju datoteka komponente sastoji se od prefiksa koji sadržava imena paketa i podpaketa kojima komponenta pripada, te opisnog naziva datoteke. Zbog konvencije da naziv komponente u prefiksu sadrži i naziv paketa, imena paketa su tipično skraćenice 2–4 slova, dok su opisni nazivi datoteka u pravilu dulji. Tako npr. datoteka `math_cal_homography.cpp` koja se nalazi u direktoriju `math/cal`, sadrži komponentu koja pripada kalibracijskom podpaketu `cal` paketa matematičkih komponenti `math`, a omogućava određivanje homografije između zadanih skupova točaka dviju ravnina. Popis paketa biblioteke dan je u tablici A.1, gdje je za svaki paket navedena ocjena složenosti kao broj linija izvornog kôda pripadnih komponenti.

paket	BLIK†	kratki opis
alg	7717	postupci obrade slike
bb	2243	glavni program agenta koordinatora
bbs	306	dijagnostički klijent za agenta koordinatora
cal	1799	glavni program i pojedini postupci umjeravanja kamera
ccam	2708	pristup upravljivim kamerama
cli	970	analiza parametara komandne linije
cvsh	548	glavni program razvojne ljuske
dib	3876	prikaz slike, pristup slikovnim elementima
ext	4240	vanjski izvorni kôd
geom	2904	geometrijske operacije u diskretiziranoj ravnini
ip	3769	postupci razvojne ljuske
mas	2544	izvedba agenta promatrača
math	4673	matematički postupci
mm	1089	elementi višeagentskog komunikacijskog protokola
pd	420	komponente ovisne o odredišnoj platformi
rs232	1283	komunikacija preko serijskog protokola `rs232`
sock	695	komunikacija preko mrežnog protokola TCP/IP
thr	1759	rad s višestrukim tokovima izvođenja
ui	1015	tekstualno korisničko sučelje
util	2381	razne neklasificirane komponente
vd	775	ponor slike, npr. datoteka, prozor, ...
vs	3809	izvor slike, npr. datoteka, sklopovlje, ...
win	3642	grafički prikaz u zasebnom prozoru

† BLIK: broj linija izvornog kôda.

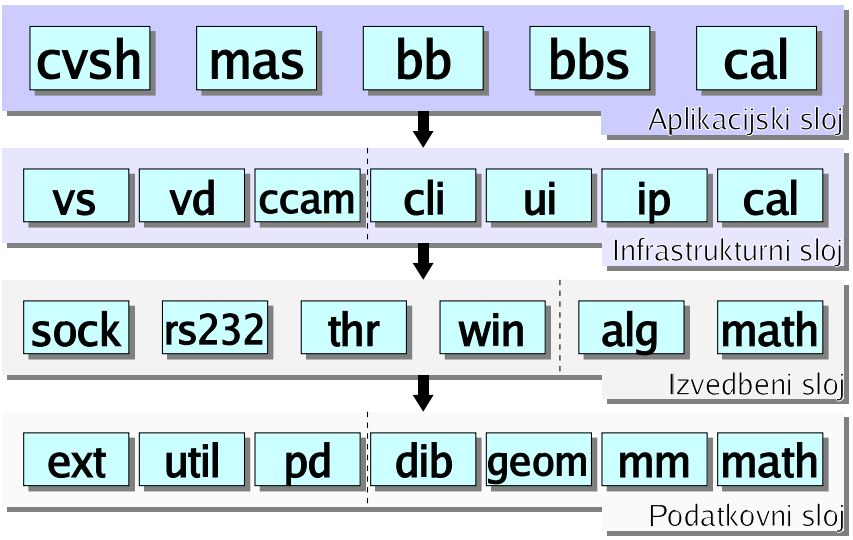
Tablica A.1: Popis paketa iz biblioteke izvornog kôda.

A.2.1 Grupiranje paketa prema apstraktnosti domene

Paketi biblioteke su organizirani prema često korištenom obrascu oblikovanja prema kojem se programske cjeline grupiraju u slojeve (*engl.* layers) [buschmann96], tako da sve komponente istog sloja pružaju usluge koje su na otprilike istoj razini apstrakcije. Komponente svakog sloja obavljaju svoje zadatke upotrebom usluga komponenata iz nižih slojeva. Osnovna zamisao obrasca je smanjiti kombinatornu eksploziju ukupnog broja veza među komponentama postupnim raščlanjivanjem složenih zadataka na višim slojevima, sve dok se ne dođe do jednostavnijih problema koji se rješavaju na nižim slojevima. U primjeni obrasca, paketi su prema apstraktnosti domene grupirani u sljedeća četiri sloja, od najapstraktnijeg prema konkretnijima: aplikacijski, infrastrukturni, izvedbeni i podatkovni, kao što je ilustrirano na sl. A.1. Tipične međusobne eksplicitne ovisnosti paketa iz pojedinih slojeva u slici su naznačene strelicama. Iznimno, paketi se mogu protezati i kroz dva susjedna sloja, kao što je to slučaj s paketima `cal` i `math`.

Aplikacijski sloj

Paketi aplikacijskog sloja sadrže komponente koje definiraju tokove izvođenja programa pojedinih ciljeva biblioteke na najvišoj razini apstrakcije. Konceptualno, postoji bijekcija između ciljeva i aplikacijskih paketa, pa ciljevi u pravilu uključuju samo po jedan aplikacijski paket. Aplikacijski paketi najčešće sadrže komponentu s glavnim programom cilja,



Slika A.1: Grupiranje paketa prema međusobnim ovisnostima i apstraktnosti domena. Crtkana granica dijeli "tehnološke" od "problemskih" paketa (detaljnije u tekstu).

koja definira funkciju `main()`. Alternativno, aplikacijski paket može uključiti komponentu `ip_dllAdapter`, koja će omogućiti dinamičko povezivanje jedne od komponenti paketa `ip` sa glavnim programom razvojne ljuske. Takav pristup se koristi u paketu koji implementira izvedbu agenta promatrača.

Infrastrukturni sloj

Infrastrukturni sloj sadrži pakete biblioteke koji enkapsuliraju temeljne zadatke komponenata aplikacijskog sloja: ulaz slike, izlaz slike, upravljanje kamerom, korisničko sučelje, te postupci računarskog vida visoke razine. Pojedini postupci su sadržani u komponentama paketa `ip` i `cw` (npr, `ip_canny` definira Cannyjev detektor rubova), a njihova izvedba se temelji na komponentama izvedbenog sloja koji sadrže komponente niže razine računarskog vida (npr, `alg_colourRGB2HSV` opisuje prevođenje slike iz formata RGB u format HSV). Taj paket predstavlja područje najveće razvojne aktivnosti sloja jer se svaki novi postupak u biblioteku dodaje kao nova komponenta paketa.

Izvedbeni sloj

Izvedbeni sloj sadrži partikularne postupke niže razine, koji su svoj status paketa "zaslužili" više složenošću izvedbe nego konceptualnom istaknutošću u kontekstu rješenja problema računarskog vida. Tako su se ovdje našli paketi koji enkapsuliraju neku sklopovsku aktivnost (npr, mrežnu komunikaciju, `sock`), pristup naprednim mogućnostima operacijskog sustava (npr, rad s prozorima, `win`), ili neke specifične aspekte konkretnog problema s područja obrade slike, `alg`, ili raspoznavanja uzoraka, `math`.

Podatkovni sloj

Podatkovni sloj sadrži pakete čije su komponente najbliže fizičkim podatcima u memoriji računala. Najvažniji paket ovog sloja, `dib`, propisuje postupke za spremanje slike i načine za manipulaciju pojedinim slikovnim elementima. Pored toga, paket `ext` sadrži vanjske procedure koje su distribuirane u obliku golih algoritama pa ih je bilo potrebno prilagoditi

suživotu sa ostalim komponentama biblioteke. Sve komponente iz ovog sloja ovise isključivo o standardnim bibliotekama, te ih je moguće po volji koristiti i u drugim projektima.

Problemski i tehnološki paketi

Paketi svih slojeva osim aplikacijskog mogu se podijeliti u dvije grupe: problemsku i tehnološku. Na sl. A.1, ta podjela je naznačena crtkanom uspravnom crtom, pri čemu su problemski paketi smješteni lijevo od crte. Problemski paketi izravno definiraju korake rješenja razmatranog problema računarskog vida na različitim razinama apstrakcije. Postupci ljuske `ip_xxx`, implementiraju svoju funkcionalnost u terminima postupaka za obradu slike `alg_yyy` i raspoznavanja uzoraka `math_zzz`, koji opet svoje operacije obavljaju na temelju slikovnih elemenata `dib_www` i operacijama nad njihovim skupovima `geom_qqq`. Nasuprot tome, tehnološki paketi pružaju unificirana sučelja za funkcionalnosti koje inače već postoje, ali se postižu na nepraktične načine, ili ovise o tipu priključenog sklopovlja i ciljnom operacijskom sustavu. Zajednička značajka tehnoloških paketa je da se mogu vrlo jednostavno ponovo upotrijebiti u nekom programskom projektu koji je posve nevezan uz porazdijeljeno praćenje objekata. Zbog svoje strateške važnosti, tehnološki paketi imaju vrlo visoki prioritet pri razvoju novih svojstava i ispravljanju nedostataka, zbog čega su relativno stabilni i vrlo prikladni za ponovnu upotrebu.

Za većinu komponenata problemskih paketa nije vjerojatno da bi se koristili u većini izvršnih programa biblioteke. Štoviše, neke od tih komponenata se ne koriste ni u jednom cilju eksperimentalnog sustava, nego su tu ostale iz vremena drugih projekata[1] Živost na području dodavanja novih komponenti je kod svih slojeva najintenzivija upravo u problemskim paketima, jer se stalno javljaju potrebe za novim postupcima i algoritmima. Pored toga, za problemske komponente vrijedi i da se u prosjeku najrjeđe koriste, pa ih je zato prije svake nove upotrebe potrebno dobro ispitati.

A.2.2 Struktura složenih paketa

Neki paketi su vremenom postali toliko složeni da se javila potreba za njihovim daljnjim raščlanjivanjem. U sljedećim paragrafima, ukratko će se opisati struktura takvih paketa, kao popis pripadnih podpaketa i njihovih složenosti.

Paket `ccam`

Paket opisuje različite koncepte vezane uz upravljive kamere i njihove parametre. Valja napomenuti da komponente ovog paketa ne opisuju postupke umjeravanja, nego samo nude praktičan način pristupa skupovima već umjerenih parametara. Struktura paketa opisana je u tablici A.2.

paket	BLIK	kratki opis
par	321	unutrašnji parametri kamere s promjenljivim vidnim poljem
pt	1043	upravljanje postoljem s 2 stupnja slobode
visca	1344	upravljanje kamerom preko protokola VISCA

Tablica A.2: Popis podpaketa paketa `ccam`.

[1] Temeljni koncepti biblioteke postavljeni su 2000. godine, pri čemu su mnogi tehnički detalji uvedeni ili dorađeni na temelju rasprava s dr.sc. Zoranom Kalafatićem i mr.sc. Vladimirom Stanisavljevićem.

Paket `dib`

Paket opisuje smještanje slike u memoriji računala na najnižoj razini. Sastoji se od dva podpaketa i nekoliko komponenti koje nisu dalje klasificirane. Struktura paketa opisana je u tablici A.3.

paket	BLIK	kratki opis
access	1689	pristup elementima slika različitih formata
io	1607	učitavanje i spremanje slika u različitim datotečnim formatima
ostatak	580	smještanje slike u memoriji, elementarna obrada

Tablica A.3: Popis podpaketa paketa `dib`.

Paket `ext`

Paket sadrži vanjske algoritme koji su prepravljeni na način da se omogući njihova integracija s ostatkom sustava, uz proizvoljno korištenje u skladu s mogućnostima koje pruža sučelje. Intervencije uključuju pretvaranje u ANSI C da bi se kôd uopće mogao prevesti prevodiocem jezika C++, oslobađanje alocirane memorije, te eksportiranje samo onih simbola koji su definirani sučeljem da tijekom povezivanja ne bi došlo konflikta zbog višestruko definiranih simbola. Struktura paketa opisana je u tablici A.4.

paket	BLIK	kratki opis
susan	2341	detektor kutova i rubova SUSAN
lmdif	1899	optimizacijski postupak po Levenbergu i Marquardtu

Tablica A.4: Popis podpaketa paketa `ext`.

Paket `geom`

Paket sadrži operacije s točkama kao elementima diskretizirane slikovne ravnine, s kojima se mogu prikazati pojedini slikovni elementi. Taj prikaz je najčešće potreban pri izvlačenju simboličkih informacija iz obrađene slike. Struktura paketa opisana je u tablici A.5.

paket	BLIK	kratki opis
draw	2400	iscrtavanje skupova točaka na prikaznoj jedinici ili u `.eps` datoteci
ostatak	504	prikaz točaka i elementarne operacije nad njima

Tablica A.5: Popis podpaketa paketa `geom`.

Paket `math`

Paket sadrži razne operacije s područja matematike i raspoznavanja uzoraka. Struktura paketa opisana je u tablici A.6.

paket	BLIK	kratki opis
cal	1645	postupci potrebni pri umjeravanju kamera
group	286	sučelje za razne algoritme grupiranja
track	1375	sučelje za razne postupke praćenja objekata ili značajki
ostatak	1367	razne neklasificirane operacije

Tablica A.6: Popis podpaketa paketa `math`.

A.3 Razvojna ljuska računarskog vida

Cilj `cvsh` biblioteke izvornog kôda omogućava interaktivno ispitivanje raznih postupaka iz domene računarskog vida. Zbog određene sličnosti s korisničkom ljuskom koja je uobičajena na operacijskim sustavima UNIX, program je nazvan ljuskom računarskog vida (*engl.* `cvsh` – computer vision shell). Program se izvodi u jednom ili više izvedbenih ciklusa, koji se zadaju ili pri pozivu programa, u komandnoj liniji, ili interaktivnim dijalogom. U svakom ciklusu dohvaća se jedna ili više slika iz izvora definiranog odabranom komponentom paketa `vs`, slike se obrađuju postupkom kojeg definira odabrana komponenta paketa `ip`, dok se odredišna slika šalje u jedan ili više ponora slike koji su definirani željenim komponentama paketa `vd`. Postupci obično imaju parametre koji se mogu interaktivno podešavati između svaka dva izvedbena ciklusa. Ukoliko je barem jedan od ponora prozor, postupak ima mogućnost iscrtavanja pronađenih značajki preko odredišne slike. Konfiguracija programa se može obaviti i pri pozivu programa i interaktivno, a obuhvaća odabire izvora slike, postupka obrade, ponora slike, te parametara postupka obrade.

Iako je izložena zamisao konceptualno vrlo jednostavna, njena izvedba je dosta zahtjevna jer podrazumijeva izgradnju velikog broja komponenti tehnoloških paketa. Odmah je jasno da to obuhvaća barem sve željene izvore slike (sklopovlje, datoteke, mreža) i sve željene ponore slike (prozori, datoteke, mreža), na svim ciljnim operacijskim sustavima. Dodavanje novih komponenti u pakete `vs`, `vd` i `ip` je iznimno jednostavno, jer se komponente napisane u skladu s konvencijama povezuju s ostatkom aplikacije automatskim metodama. Svaka od tih komponenti se prilikom pokretanja programa, u fazi statičke inicijalizacije, upisuje u registar svoje klase, koji je odgovoran za stvaranje konkretnih objekata u ovisnosti o zadanom opisnom nizu znakova. Za detalje pogledati izvedbu komponenti `vs_grab` ili `ccam_pt`.

Sadašnja implementacija razvojne ljuske podržava samo tekstualno korisničko sučelje, dok se grafičke mogućnosti modernih računala koriste samo pri prikazu rezultata obrade. To se na prvi pogled može činiti nazadno, ali, oblikovanje dovoljno općenitog i prenosivog grafičkog sučelja zahtjevan je i dugotrajan zadatak. Dosadašnja publika uglavnom nije imala problema sa savladavanjem tekstualnog sučelja, pa će razvoj novih komponenti tehnoloških paketa te inkrementalna poboljšanja već postojećih komponenti još neko vrijeme imati prioritet.

A.3.1 Parametri komandne linije

Ljuska omogućuje dva temeljna načina rada: neinteraktivni i interaktivni. U slučaju neinteraktivnog rada, parametri komandne linije u potpunosti određuju tijek izvođenja, dok u suprotnom oni definiraju samo početnu konfiguraciju programa koja se tokom interaktivnog rada može promijeniti. Komandnom linijom se može utjecati na sljedeće parametre izvođenja programa:

- **izvor slijeda slika** (opcija `-s`) — sklopovlje ili datoteka;
- **postupak obrade** (opcija `-a`) — ugrađeni algoritam ili put do dinamičke biblioteke;

- **konfiguracijski niz** (opcija `-c`) — ovisan o postupku obrade.
- **interaktivni rad** (opcija `-i`) — može se zadati početna naredba interaktivnog rada;
- **ponor slijeda slika** (opcija `-d`) — datoteka ili prozor;
- **maksimalna performansa** (opcija `-m`) — definira minimalno vrijeme obrade svake slike;

Kod interaktivnog rada, podrazumijeva se prozor kao jedini ponor slike. Sasvim općenito, program se može pozvati upotrebom jednog od sljedećih sintaksnih oblika:

1. **neinteraktivni rad:**

```
cvsh -s<Source>[#<Range>] -a=<Algorithm> -c=<ConfigString>
     -d<Dest> [-d<Dest> ...  ]
```

2. **interaktivni rad:**

```
cvsh -s<Source>[#<Range>] -a=<Algorithm> -c=<ConfigString>
     -i[=<CommandString>]
```

Sintaksa pojedinih opcija je opisana u nastavku teksta, a njen kraći prikaz može se dobiti pozivanjem programa bez argumenata.

Opcija `-s<Source>[#<Range>]`

Specificira se izvor slijeda slika `<Source>`, te interval slika `<Range>` koji se želi obraditi. Izvor slika općenito može biti datoteka, mrežni poslužioc ili međusklop za pribavljanje slike pa prema tome sintaksa tijela opcije `<Source>` može imati jedan od sljedećih oblika:

1. `f=filename`, gdje `filename` predstavlja valjanu putanju do željene izvorne datoteke (dozvoljeni formati datoteka su opisani u odjeljku A.3.1);
2. `n=host:port`, gdje je `host` IP adresa udaljenog poslužioca koji prima IP pozive na portu <Port> (u ovoj verziji ljuske, opcija nije podržana);
3. `g=name[@board[.channel[.mode]][:wxh[xszpix]]]`, gdje je:
   - `name`: ime komponente koja opisuje rad s međusklopom za pribavljanje slike;
   - `board`: indeks međusklopa (ako ih ima više);
   - `channel`: ulazni kanal međusklopa;
   - `mode`: način rada, ovisi o tipu međusklopa;
   - `wxh`: željene dimenzije slike;
   - `szpix`: broj bajta korištenih za prikaz svakog slikovnog elementa: tipično 1 označava sivu sliku, dok 3 i 4 označava sliku u boji;

Naravno, komponenta ne može ispuniti ovako zadane želje u svakom slučaju, jer sklopovi podržavaju različite formate slike, nego mora pokušati naći dozvoljenu konfiguraciju koja je najbliže zadanoj. U ovom trenutku, podržana su po dva sučelja za međusklopove za prikaz slike na oba operacijska sustava, kao što je prikazano u tablici A.7.

Interval slika `<Range>` se uzima u obzir samo ako je izvor slike datoteka i služi za ograničavanje obrade na slike datoteke čiji redni brojevi su u intervalu sadržani. `<Range>` ima sintaksu `#From-To`, gdje su `From` i `To` redni brojevi prve odnosno neposrednog sljedbenika posljednje slike iz intervala kojeg je potrebno obraditi.

komponenta	OS	kratki opis
`vs_grab_unixV4L`	Linux	standardno sučelje pod Linuxom
`vs_grab_unix1394`	Linux	pribavljanje slike preko sabirnice IEEE 1394
`vs_grab_w32MCI`	MS Windows	posebno sučelje tvrtke Imagenation
`vs_grab_w32PXC`	MS Windows	standardno sučelje pod MS Windows

Tablica A.7: Podržana sučelja za pribavljanje slike.

Opcija `-a=<Algorithm>`

Znakovni niz `<Algorithm>` definira postupak obrade. Moguće je odabrati jedan od ugrađenih postupaka obrade (npr. `histogram` za pronalaženje histograma slike), ili valjanu putanju izvršne datoteke s vanjskim postupkom. U posljednjem slučaju, izvršna datoteka mora biti prevedena kao dinamička biblioteka, i mora sadržavati barem jednu komponentu izvedenu iz `ip_base`. Za detalje proučiti komponente `ip_dllAdapter`, `ip_dllStub` te `ip_registry`, i pri tome obratiti posebnu pažnju na način upotrebe enumeracijskog parametra `ip_creator_base::DllEntry`.

Opcija `-c=<ConfigString>`

Specificira se konfiguracijski niz znakova za postupak određen opcijom `-a`. Konfiguracijski nizovi ovise o postupku.

Opcija `-d<Dest>`

Specificira se odredište slijeda slika `<Dest>`. koji općenito može biti datoteka, mrežni poslužioc ili prikazna jedinica pa prema tome sintaksa tijela opcije `<Source>` može imati jedan od sljedećih oblika:

1. `f=filename`, gdje `filename` predstavlja valjanu putanju do željene odredišne datoteke (dozvoljeni formati datoteka su opisani u odjeljku A.3.1);
2. `n=port`, gdje je `port` mrežni ulaz na kojem će se obrađene slike staviti na raspolaganje udaljenim klijentima (ova opcija trenutno nije dostupna);
3. `w`, odredište je prozor koji se iscrtava na prikaznoj jedinici računala.

Opcija `-i[=<CommandString>]`

Specificira se interaktivni način rada i, eventualno, prva interaktivna naredba. Obrada svih slika slijeda mogla bi se tražiti naredbom `-i="process next -1"`. Popis interaktivnih naredbi se nalazi u odjeljku A.3.2.

Opcija `-m=<MaxHz>`

Specificira se maksimalna performansa programa, u Hz. Parametar efektivno određuje minimalno vrijeme obrade, na način da se eventualna brža obrada neke slike kompenzira čekanjem. Naredba je potrebna u situacijama kada je računalo "prebrzo" za zadani postupak pa se rezultati obrade ne mogu pratiti u stvarnom vremenu.

Formati ulaznih datoteka

Kod opcija `-sf=...` i `-df=...`, ekstenzija specificirane datoteke mora odgovarati njenom formatu, kao što slijedi:

- ekstenzija **.avi**: datoteka je u Microsoftovom AVI formatu (format nije podržan na operacijskim sustavima koji nisu proizvedeni od strane Microsofta);
- ekstenzija **.srs**: datoteka se sastoji od slijeda slika u nekomprimiranom *Sun raster* formatu;
- ekstenzije **.ppm**, **.bmp**, **.sr**: datoteka se sastoji od jedne slike u *portable pixmap*, *Microsoft bmp* odnosno *Sun raster* formatu;
- ekstenzija **.txt**: datoteka je tekstualna i specificira više nizova slika iz drugih datoteka.

Tekstualna izvorna datoteka se sastoji od zaglavlja i liste datoteka s opcionalnim odgovarajućim intervalima. Svaka od datoteka sa slijedom slika za obradu smješta se u zasebnu liniju tekstualne datoteke sa sintaksom `<FileName>[<Range>[<Range> ...]]` gdje je `<FileName>` ime datoteke, a `<Range>` interval sa sintaksom opisanom u odjeljku A.3.1. Sintaksa dozvoljava specificiranje više intervala, npr, `ime_datoteke#100-200#400-500`. Prazne linije tekstualne datoteke se zanemaruju, kao i linije u kojima je u prvom stupcu znak "`#`" (komentari). Opcionalno zaglavlje ima sintaksu `@<Dir>` i definira prefiks koji će se prilikom interpretacije nadodati svakom od imena datoteka `<FileName>`.

Opisanim načinom mogu se zadati posebno zanimljivi dijelovi različitih datoteka, što je posebno korisno pri analizi dugačkih slijedova dinamičke scene. Primjer kojim se zadaje izvorni slijed koji se sastoji od 2. i 3. slike datoteke `skcng0.srs` te 0., 1. i 4. slike datoteke `skcng1.srs` pokazan je na sl. A.2

A.3.2 Interaktivne naredbe ljuske

Ako je u komandnoj liniji specificiran interaktivni način rada (opcija `-i`), ljuska prilikom pokretanja izvršava početnu naredbu (ako je ona zadana) i prepušta daljnje upravljanje korisniku, preko tekstualnog korisničkog sučelja. Naredbe koje sučelje podržava se mogu svrstati u tri skupine:

- naredbe za podešavanje parametara izvođenja (`algorithm`, `source` i `config`);
- naredbe za prikaz i obradu slika ulaznog slijeda (`process` i `show`);

```
# ovako se označavaju komentari

# oznaka direktorija u kojem se nalaze datoteke
@/home/sinisa/work/data/calib/latin/calib.txt

# skcng0.srs je slijed u kojem su zanimljive slike 2 i 3
skcng0.srs#2-4

# može se čitati više intervala istog slijeda
skcng1.srs#0-2#4-5
```

Slika A.2: Primjer tekstualne datoteke izvornog slijeda.

- ostale naredbe (`help` i `quit`).

Naredbe iz prve skupine utječu na parametre čije početne vrijednosti se zadaju iz komandne linije (izvor slijeda slika, postupak obrade i konfiguracija postupka obrade) kao što je dokumentirano u odjeljku A.3.1. Sve naredbe je moguće skraćivati; u slučaju višeznačne kratice, ljuska odabire prvu naredbu prema leksikografskom uređaju. Tako su i `proc` i `p` valjani oblici naredbe `process`, `s` određuje naredbu `show`, a minimalan zapis naredbe `source` je `so`. Dozvoljeni sintaksni oblici pojedinih naredbi su opisani u sljedećim odjeljcima.

Naredba `source`

Naredba omogućava promjenu izvora slijeda slika. Sintaksa naredbe je `source <Source>`, gdje `<Source>` može poprimiti iste oblike kao i kod opcije komandne linije `-s` (vidi A.3.1).

Naredba `algorithm`

Tom naredbom se može interaktivno specificirati novi postupak obrade. Sintaksa naredbe je `algorithm <Algorithm>`, gdje `<Algorithm>` može poprimiti iste oblike kao i kod opcije komandne linije `-a` (vidi A.3.1).

Naredba `config`

Naredba omogućava interaktivni uvid u parametre postupka obrade i njihovu promjenu. Sintaksa naredbe je `config [<ConfigString>]`, gdje je `<ConfigString>` konfiguracijski niz i ovisi o postupku obrade. Ako konfiguracijski niz nije zadan, konfiguracija algoritma se obavlja interaktivno.

Naredba `process`

Naredbom se pokreće postupak obrade slika iz izvornog slijeda. Interval koji je potrebno obraditi je određen argumentima naredbe i može biti apsolutan ili relativan u odnosu na poziciju posljednje obrađene slike iz slijeda *SSP*. Sintaksa naredbe je `source <Location>`, a podržani formati argumenta `<Location>` i odgovarajući intervali za obradu su sažeti u tablici A.8. Podrazumijevana vrijednost za `[<howMany>]` je 1. Ukoliko je neka od granica dobivenog intervala obrade manja od nule ili veća od broja slika izvornog slijeda, ona se svodi na dozvoljenu vrijednost operacijom modulo n_{IS}, gdje je n_{IS} broj slika izvornog slijeda. Specijalni slučaj parametra `<howMany>` jest vrijednost -1, i tada se kao vrijednost parametra

oblik naredbe	interval obrade
`next [<howMany>]`	`[SSP+1,SSP+<howMany>+1>`
`previous [<howMany>]`	`[SSP-<howMany>,SSP>` (unatrag!)
`this [<howMany>]`	`[SSP,SSP+1>` (`<howMany>` ponavljanja)
`address <frame>`	`[<frame>,<frame>+1>`

Tablica A.8: Oblici interaktivne naredbe `process` (detalji u tekstu).

podrazumijeva n_{IS}. Tako je naredbom `process next -1` moguće postići obradu svih slika iz slijeda točno jednom. Izvođenje neke duge naredbe može se opozvati pritiskom tipke `<ESC>` u *odredišnom* prozoru ljuske, tj. u prozoru čiji naslov sadržava naziv izvora slijeda slika i brzinu obrade. Tipka `<ESC>` može se koristiti i za prekidanje neinteraktivne obrade.

Ključne riječi `next`, `previous`, `this` i `address` mogu se skraćivati kao i naredbe ljuske (vidi odjeljak A.3.2). Kod izvora slika bez mogućnosti pozicioniranja (npr. međusklop za pribavljanje slike), naredbe s ključnim riječima `previous`, `this` i `address` se interpretiraju kao i odgovarajuća naredba s ključnom riječi `next`. Numerički parametar `[<howMany>]` kod riječi `this` označava koliko puta za redom će se obraditi tekuća slika iz slijeda. To može biti korišteno sa svrhom preciznog mjerenja trajanja izvođenja postupka obrade.

Naredba `show`

Naredba ima istu sintaksu kao i naredba `process` (odjeljak A.3.2), a razlikuje se od nje samo u tome što se slike iz intervala izvornog slijeda prikazuju bez obrade.

Naredba `help`

Tom naredbom se ispisuje sažeti popis interaktivnih naredbi ljuske.

Naredba `quit`

Naredbom se prekida rad programa i vraća upravljanje ljusci operacijskog sustava.

Upotreba miša

Ljuska omogućava pohranjivanje podataka koji su trenutno prikazani u nekom od prozora u datoteku formatiranu u skladu sa standardom *encapsulated postscript*. Rezultate simboličke obrade je moguće pohraniti ili s praznom pozadinom ili na način da se kao pozadina postavi slika koja je u prozoru trenutno prikazana. Za tu namjenu se koristi desna tipka miša, kao što je prikazano u tablici A.9. Komponenta za prevođenje rezultata simboličke obrade u grafički format `.eps` značajno povećava vrijednost razvojne ljuske a napisao ju je dr.sc. Zoran Kalafatić.

tipka miša	modifikator	opis
`desna`	-	pohranjuju se samo rezultati obrade;
`desna`	`<Shift>`	pohranjuju se rezultati obrade i odredišna slika
`desna`	`<Ctrl>`	pohranjuje se samo odredišna slika

Tablica A.9: Načini korištenja desne tipke miša u bilo kojem od prozora.

Ljuska ne definira funkcionalnost ostalih tipki miša i tastature (osim `<ESC>`), pa se ona može zasebno definirati u okviru pojedinih postupaka obrade. Valaj obratiti pažnju na to da

se postupcima prosljeđuju samo tipke koje su pritisnute unutar *odredišnog* prozora ljuske, tj. prozora čiji naslov sadržava naziv izvora slijeda slika i brzinu obrade.

A.3.3 Primjer interaktivnog korištenja programa

U sljedećem primjeru, "`#`" označava komentare, "`>`" oznaku za unos naredbi ljuske operacijskog sustava, dok je "`file[i/n] $`" oznaka za unos naredbi ljuske računarskog vida.

```
# kratak opis parametara komandne linije
> cvsh

# pokretanje programa u interaktivnom načinu rada
# korištenjem postupka za pronalaženje kalibracijskog uzorka
# nad slijedom slika koji je definiran datotekom skcng.txt
> cvsh -sf=skcng.txt -a=matchGrid -c=".5" -i

# popis interaktivnih naredbi ljuske
skcng.txt[0/19]$ help

# prikaz svih slika iz slijeda
skcng.txt[0/19]$ s n -1

# obrada slika 1-5
skcng.txt[0/19]$ p n 5

# promjena praga binarizacije
skcng.txt[5/19]$ c .6

# obrada slika 10-15
skcng.txt[5/19]$ s a 10
skcng.txt[10/19]$ p n 5

# promjena postupka
skcng.txt[15/19]$ a canny_w0

# interaktivna konfiguracija postupka
skcng.txt[15/19]$ c

# obrada slika 8-4 (unatrag)
skcng.txt[15/19]$ s a 9
skcng.txt[9/19]$ p p 5

# promjena izvorne datoteke
skcng.txt[4/19]$ so f=bug.txt

# obrada 5. slike 25 puta
bug.txt[0/7]$ s a 5
bug.txt[5/7]$ p t 25

# kraj rada
bug.txt[5/7]$ quit
```

A.3.4 Proširivanje ljuske novim postupcima

Razvojna ljuska je zamišljena na način da ostvari optimalni kompromis između jednostavnosti upotrebe i ograničenja područja primjene. U najjednostavnijem slučaju, korisniku je dovoljno kreirati svoju vlastitu komponentu paketa `ip`, prema konvencijama koje će biti opisane u ovom dijelu rada. Zahtjevniji korisnici uvijek mogu odustati od predložene zamisli, te koristiti komponente tehnoloških paketa pri izgradnji programa s drukčijom problemskom arhitekturom, koja će bolje odražavati odnose specifičnog problema. Ipak, uz stanovite konceptualne ustupke, predloženi formalizam omogućava pisanje dosta složenih postupaka što pokazuje i izvedba agenta promatrača koja je predstavljena u odjeljku 5.3.

Konvencije za izradu novih postupaka

Svi postupci implementiraju zajedničko sučelje `ip_base`, čiji je pojednostavljeni oblik prikazan na sl. A.3, pri čemu je značenje objekata pojedinih tipova sažeto u tablici A.10.

```
class ip_base{
//operations
public:
  // mandatory
  virtual void process(
    const dib_base& src,
    const win_eventVector& events,
    dib_base& dib_dst,
    win_annotation& ann) =0;
  // may be omitted
  virtual void config(cli_wrap& cli);
  virtual void profile(ui_profileData& ){}

//attributes
public: //mandatory
  virtual char const* name()=0;
  const dib_fmt& dstFmt();

//  ...
};
```

Slika A.3: Pojednostavljeni oblik zajedničkog sučelja `ip_base`.

tip	opis	sastavni dijelovi ili detaljniji opis
`dib_fmt`	format slike	dimenzije i veličina slikovnih elemenata
`dib_base`	slika	format slike i matrica slikovnih elemenata
`win_eventVector`	ulazni podatci	popis pritisnutih tipki tastature i miša
`win_annotation`	rezultati obrade	popis geometrijskih značajki
`cli_wrap`	konzolni u-i	objekt pruža mogućnost konzolnog ulaza i ispisa
`ui_profileData`	profil izvođenja	vremena izvođenja pojedinih dijelova algoritma

Tablica A.10: Tipovi koji se koriste u sučelju komponente `ip_base`.

Zahtjevi sučelja sa sl. A.3 prema konkretnim izvedbama mogu biti sažeti kako slijedi:

- Glavna operacija sučelja `process` prima od ljuske ulaznu sliku i popis korisničkih akcija u odredišnom prozoru ljuske, obavlja operaciju za koju je komponenta namijenjena, te vraća ljuski obrađenu sliku i simboličke rezultate obrade. Kod postupka koji enkapsulira postupak za traženje mjernog uzorka, odredišna slika se dobiva binarizacijom, dok pronađena okna izlučenih crnih regija i detektirana mreža uzorka čine simboličke rezultate (vidi npr. sl. 6.6).

- Neobavezne operacije sučelja `config` i `profile`, omogućuju interaktivnu konfiguraciju postupka preko konzolnog ispisa i unosa, te izvještanje ljuske o razdiobi potrošenog vremena na pojedine korake postupka.

- Vrlo važan atribut sučelja, `name`, koristi se za automatsku prijavu postupka odgovarajućoj registracijskoj komponenti. Jedini način da bilo koji postupak postane aktivan jest da se ime zatraženo preko sučelja ljuske poklapa s imenom postupka određenog atributom `name`.

- Konačno, konkretan objekt nekog postupka mora u svakom trenutku znati format izlazne slike, kako bi ljuska mogla na samom početku kreirati datotečne ponore slike koji taj podatak zahtijevaju. Implikacija toga je da postupkovni objekt mora u svom konstruktoru dobiti format ulazne slike (kojeg ljuska dobiva od izvora slike).

Pored implementiranja spomenutih zahtjeva, konkretan postupak mora zadovoljiti i zahtjeve koji omogućuju automatsko povezivanje postupka s korisničkim sučeljem ljuske:

- Postupak mora imati konstruktor sa samo jednim argumentom tipa `const dib_fmt&`.

- Postupak mora imati statički atribut `s_name` koji vraća isto što i `name`.

- Konačno, postupak mora instancirati statički objekt čiji tip se dobiva instanciranjem predloška `ip_creator<>` s tipom konkretnog postupka. Registracija objekta se obavlja iz konstruktora statičkog objekta, u fazi inicijalizacije statičkih varijabli.

Konkretni postupci se kreiraju dinamički, preko registracijskog objekta, dok ljuska s njima komunicira preko zajedničkog sučelja. Stoga, odgovarajuće komponente ne moraju imati sučeljnu datoteku, nego se i deklaracija i implementacija glavne klase postupka mogu nalaziti unutar jedne implementacijske datoteke (za primjer, vidi `ip_radial`). U skladu s navedenim zahtjevima i razmatranjima, početak implementacijske datoteke konkretnog postupka bi nakon uključivanja potrebnih biblioteka `#include` trebalo izgledati kao što je prikazano na sl. A.4.

A.4 Ostale izvršne datoteke eksperimentalnog sustava

Od pet ciljeva biblioteke, u dosadašnjem tekstu opisana je samo razvojna ljuska `cvsh` kao najkarakterističniji predstavnik grupe. U ovom odjeljku će se vrlo ukratko opisati namjena i upotreba ostalih ciljeva.

A.4.1 Izvedba agenata promatrača

Cilj `mas.cvsh` sadrži izvedbu agenata promatrača višeagentskog sustava. Izveden je kao postupak razvojne ljuske `ip_mas`, a sastoji se od komponenata paketa `mas`, te koristi usluge brojnih problemskih komponenti izvedbenog i podatkovnog sloja. Pokretanje cilja se

```
class ip_concrete:
  public ip_base
{
  // ... (data) ...

public:
  ip_concrete(const dib_fmt&);
  virtual ~ip_concrete();
public:
  virtual void process(
    const dib_base& src,
    const win_eventVectorProtocol& events,
    dib_base& dst,
    win_annotationProtocol& ann);
  virtual void profile(ui_profileDataProtocol& p);
  virtual void config(cli_wrapProtocol& cli);
public:
  static char const* s_name(){return "concrete";}
  virtual char const* name(){return s_name();}
  const dib_fmt& dst_fmt();

  // ...
};

namespace{
  // the static creator object
  ip_creator<ip_concrete> creator_;
}
```

Slika A.4: Karakteristični dijelovi implementacijske komponente konkretnog postupka.

izvodi preko sučelja razvojne ljuske, pri čemu je u pozivu ljuske potrebno navesti odgovarajući međusklop za pribavljanje slike. Tako se, pod pretpostavkom da se obje izvršne datoteke `cvsh` i `mas.cvsh` nalaze u istom direktoriju, pokretanje programa na radnoj stanici `Asus A7M266--D` može obaviti sljedećom naredbom:

```
cvsh -sg=unix1394@320x240x3 -a=mas.cvsh -i="p n -1"
```

Razvojna ljuska je prilagođena postupcima pasivnog vida, pa ne predviđa mogućnost konfiguriranja upravljive kamere iz komandne linije. Zbog ograničenog vremena, to je privremeno riješeno čvrstim pristupom stvaranju i konfiguriranju objekata za upravljanje kamerom, koji se temelji na mrežnom imenu računala na kojem se program izvodi. Dugoročno rješenje je napraviti proširenje ljuske na način da njena domena obuhvati i problem stvaranja objekata za apstrakciju pristupa upravljivim kamerama (to je tek odnedavno postalo moguće uslijed temeljite revizije paketa `ccam`). Komponenta `ip_base` u tom slučaju bi bila proširena sa dodatnom konfiguracijskom metodom, kojom bi ljuska nakon analize parametara komandne linije mogla postupku poslati objekte za enkapsulaciju upravljanja smjerom gledanja i širinom vidnog polja. Takav pristup već je primijenjen u cilju `cal` koji objedinjuje postupke umjeravanja parametara kamera, a opisan je u odjeljku A.4.3.

Programska konfiguracija postupka uključuje mrežnu adresu pridruženog koordinatora, parametre obrade slike (npr, raspon nijansi crvene boje), te unutrašnje i vanjske parametre kamere. Ti parametri se mogu podešavati preko sučelja ljuske, dok se valjane početne

vrijednosti također postavljaju u ovisnosti o mrežnom imenu računala na kojem se agent izvodi.

A.4.2 Izvedba agenta koordinatora

Za razliku od agenata promatrača koji moraju raditi sa različitim kamerama, pokretnim postoljima i međusklopovima za pribavljanje slike, koordinatori ne trebaju komunicirati sa egzotičnim sklopovljem. Stoga nije potrebna nikakva posebna konfiguracija, pa se odgovarajuća izvršna datoteka `bb` pokreće bez ikakvih parametara komandne linije.

Pored izvedbe samog agenta kordinatora, oblikovan je i pomoćni program `bbs`, koji može ostvariti upravljačku vezu s koordinatorom i omogućiti konfiguraciju internih parametara koordinatora kao što je promjena koordinacijske strategije. Nadalje, program se koordinatoru može predstaviti i kao agent, te mu slati proizvoljne mjerne podatke, u skladu s komunikacijskim protokolom. Takva mogućnost je vrlo korisna kod dijagnosticiranja neželjenog ponašanja agenta koordinatora. Sintaksa poziva programa jest:

```
bbs [-a] [server]
```

Gdje je `server` mrežna adresa agenta koordinatora, a opcija određuje hoće li se program predstaviti kao agent promatrač (`-a`) ili kao upravitelj. Detaljniji popis mogućnosti programa `bbs`, može se dobiti pozivom `bbs -h`.

A.4.3 Izvedba postupaka za umjeravanje kamera

Cilj `cal` predstavlja jednostavno unificirano sučelje za razne postupke umjeravanja opreme aktivnog vida. Slično kao i u paketu `ip`, konkretni postupci trebaju naslijediti `cal_worker`, definirati statički atribut `s_name` koji vraća jedinstveni identifikator postupka, te instancirati statički objekt za povezivanje s glavnim programom. Do sada su implementirani postupci za umjeravanje unutrašnjih i vanjskih parametara kamere, kao što je opisano u 4.3.2 i 4.4.1.

Kao što je već najavljeno u A.4.1, program omogućava sofisticirano konfiguriranje priključenog sklopovlja. Za početak, tu su opcije `-s`, `-a` i `-c` koje su analogne odgovarajućim opcijama cilja `cvsh`, a omogućuju odabir ulaznog slijeda slika (to je u pravilu sklopovlje), te konkretnog postupka umjeravanja i njegovih eventualnih parametara. Pored toga, postupak definira nove dvije opcije, koje će biti opisane u nastavku teksta.

Opcija `--ptu`

Opcija omogućava zadavanje konkretnog postolja za upravljanje smjerom gledanja kamere. Puna sintaksa opcije jest: `--ptu=<PtuModel>[@port][:id]`. Pri tome `PtuModel` označava tip postolja, dok je `port` broj koji odgovara serijskom ulazu na kojeg je postolje spojeno. Ako se `port` ne navede, podrazumijeva se vrijednost 0, koja odgovara vratima `COM1` na operacijskom sustavu MS–Windows, odnosno `/dev/ttyS0` na Linuxu. Konačno, `id` je serijski broj postolja koji se u trenutnoj verziji biblioteke ne koristi. Buduća namjena tog parametra jest indeksiranje kalibracijskih podataka za dano postolje, koji su potrebni za preračunavanje trenutnog položaja u matricu Euklidskog pomaka, kao što je to opisano u odjeljku 4.4.2. Podržani tipovi postolja su Directed Perception PTU s oznakom `dpAsc` i integrirano postolje kamere Sony Evi–D31 s oznakom `visca`.

Opcija `--cam`

Opcija omogućava naznačavanje korištene kamere, kako bi program mogao odabrati pravi skup unutrašnjih parametara. Sintaksa podržava i pasivne i upravljive kamere; kod uprav-

ljivih kamera obično su najzanimljiviji parametri širine vidnog polja (*engl.* zoom) i žarišne daljine leće (koju ne treba miješati sa žarišnom daljinom projekcije!). Puna sintaksa opcije jest: `--cam=<CamModel>[@port]:id`, pri čemu `CamModel` označava tip kamere, `port` je broj koji označava serijski ulaz na kojeg je spojen upravljački priključak kamere, dok je `id` serijski broj kamere koji se koristi za dohvat umjerenih parametara. Kao i kod upravljivog postolja, podrazumijevana vrijednost ulaznih vrata je 0. U ovom trenutku nisu podržane kamere sa upravljivim parametrima, uglavnom zato jer još nije oblikovan dovoljno fleksibilan postupak njihove kalibracije. Za pasivne kamere je u polju `CamModel` potrebno navesti niz `fixed`, dok polje `id` u tom slučaju sadrži i tip i serijski broj kamere. Trenutno su umjerene po jedna laboratorijska kamera Basler s oznakom `baslerA301f-skcng-109210017345` i Sony Evi–D31 s oznakom `evid31-PXC-110561`. Konačno, podržana je i oznaka `dumb` kojom se naznačuje da kamera nije kalibrirana.

Primjer poziva kalibracijskog programa

Kod kalibracije ekstrinsičnih parametara na radnoj stanici `Asus A7M266--D`, cilj `cal` se tipično poziva kako slijedi:

```
cal -sg=unix1394 -c=0.3 -a=extrinsic
    --cam=fixed:baslerA301f-skcng-109210017345 --ptu=dpAsc
```

Dodatak B

Apstrakcija pristupa specifičnom sklopovlju

U skladu s dobrim običajima programskog inženjerstva, stroga separacija logike programa od izvedbenih detalja pristupa različitom sklopovlju postavljena je kao jedan od značajnijih zahtjeva pri oblikovanju eksperimentalnog sustava. Time se, pored osiguravanja jasnije raspodjele odgovornosti među komponentama i paketima programa [lakos96], postiže mogućnost transparentnog uklapanja heterogenog sklopovlja i odgovarajuće programske opreme u funkcionalni sustav. To je posebno značajno u laboratorijskim uvjetima, kad je eksperimente potrebno obaviti korištenjem računalnih i ostalih sklopovskih resursa opće namjene, različitih proizvođača i godina proizvodnje.

Na području praćenja objekata aktivnim računarskim vidom, javlja se potreba za radom s tri različite vrste sklopovlja, i to s međusklopovima za pribavljanje slike, upravljivim postoljima za podešavanje smjera gledanja kamere, te kamerama s upravljivim unutrašnjim parametrima. Za svaku od navedenih vrsta sklopovlja razvijeno je apstraktno sučelje, te odgovarajuće konkretne izvedbe koje implementiraju metode apstraktnog sučelja upotrebom odgovarajućih specifičnih programskih biblioteka. Važniji detalji paketa koji obuhvaćaju komponente specifičnog sklopovlja opisani su u sljedećim odjeljcima.

B.1 Motivacija

Ako se zanemari stabilnost i lakoća održavanja koje su implicitna posljedica jasne strukture velikog programskog sustava, motivacija za apstrahiranje sučelja različitih proizvođača opreme može se činiti upitna. Naime, radi se o netrivijalnom poslu koji oduzima puno vremena, a ne dodaje nikakvu novu funkcionalnost konačnom programskom sustavu. Razmotrimo zato alternativni pristup kojim bi se unaprijed ograničio spektar sklopovlja koje će raditi s programskim sustavom, na proizvode koji odgovaraju jednoj, unaprijed odabranoj specifikaciji sučelja. Najbolje bi bilo kad bi takvo sučelje prema istovrsnom sklopovlju različitih proizvođača bilo propisano važećim industrijskim standardom, ali, kod navedene tri vrste sklopovlja aktivnog računarskog vida, to nažalost nije slučaj. Primjerice, na području pribavljanja slike iz analognog video signala postoji velik broj programskih sučelja, ali svako od njih je primjenljivo na samo mali podskup proizvođača i ciljnih operacijskih sustava, kao što je prikazano u tablici B.1. Nadalje, kvaliteta usluge različitih sučelja znatno varira, kao što je opisano u 5.1.2. Konačno, na dugoročnu podršku bilo kojeg od tih sučelja ne može se računati, jer iskustvo pokazuje da se postojeća sučelja često napuštaju u korist novih rješenja koja nisu unatrag kompatibilna.

sučelje	proizvođač opreme	operacijski sustav
MCI	razni	MS–Windows
V4L	razni	Linux
VIS	Sun	Solaris
MIL	Matrox	MS–Windows
PXC	Imagenation	MS–Windows

Tablica B.1: Neka programska sučelja za pribavljanje slike iz analognog signala.

Situacija kod ostalih sklopovskih resursa je vrlo slična onoj koja je prikazana u tablici B.1. Na tržištu ima više međusobno nekompatibilnih rješenja, svako od kojih nudi različitu razinu kvalitete po različitoj cijeni. Stoga se pokazuje da alternative apstrahiranju sučelja zapravo nema, kad god konačna aplikacija podrazumijeva korištenje većeg broja kamera, ili ako je predviđeni životni vijek projekta duži od nekoliko godina. Kod takvih aplikacija, manjkavosti jedinstvene platforme se mogu pokazati tek u kasnijoj fazi oblikovanja, kada promjene tehnologije mogu znatno usporiti razvoj, a to se može pokazati kobnim za uspješni završetak projekta. Zato je apstrakciju potrebno napraviti što prije, čim se stekne dovoljno dobar uvid u izvođenje ključnih operacija konkretnog sklopovlja.

B.2 Pribavljanje slike

Pribavljanje slike je najvažnija od spomenutih sklopovskih operacija, jer se koristi kod svih primjena računarskog vida u stvarnom vremenu, neovisno o tome ima li se pristup parametrima kamere ili ne. Zahtijeva se da željeni paket programskih komponenata ima izolirajuće apstraktno sučelje [lakos96], koje bi korisnicima omogućilo iscrpno ispitivanje ostalih komponenata sustava, neovisno o konkretnim implementacijama koje bi također trebale biti dio paketa. Vrlo važno svojstvo takve organizacije jest mogućnost paralelnog razvoja razmatranog paketa i ostalih komponenata sustava, pod pretpostavkom da je izolirajuće sučelje dobro oblikovano. U skladu s raspravom u tablici 5.1.2, glavna operacija tog sučelja bila bi dohvat nove slike iz slijeda, a ona bi trebala omogućiti i pristup trenutku stvaranja slike u senzoru kamere.

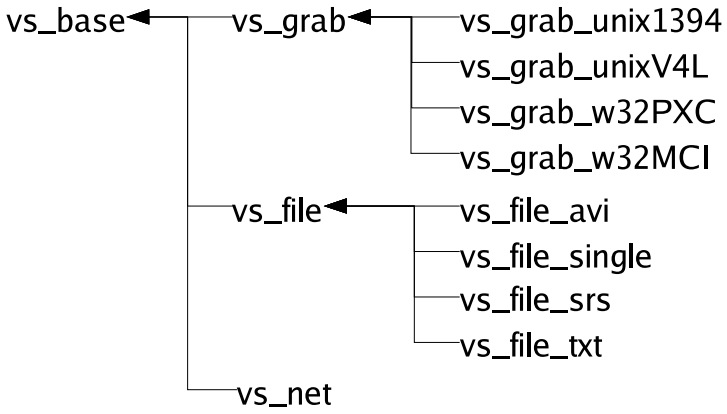
B.2.1 Struktura paketa

Pored izravnog pristupa sklopu za pribavljanje slike, također se zahtijevaju konkretne izvedbe tog sučelja koje bi omogućavale transparentno pribavljanje slike iz zadane datoteke. Konačno, može se razmišljati i o trećoj izvedbi sučelja, u kojoj bi se slike pribavljale s udaljenog mrežnog poslužioca, u skladu s obrascem oblikovanja koji je poznat pod imenom `Proxy` [gamma95]. Takva infrastruktura omogućila bi razvoj porazdijeljenih postupaka obrade slike, što nije tema ovog rada pa se neće dalje razmatrati. Mogućnost dohvaćanja slika iz datoteke je međutim uvijek interesantna, pogotovo u fazi ispitivanja novih postupaka kada je potrebno osigurati ponovljivost ulaznih podataka. U opisanim zahtjevima mogu biti identificirani sljedeći ključni koncepti:

- `izvor slike`: apstraktno sučelje za pribavljanje slike. Omogućava dohvat sljedeće slike slijeda i trenutka njenog stvaranja, te prethodni pristup njenom formatu i organizaciji u memoriji računala, u skladu s komponentama paketa `dib`.

- `sklopovski izvor`: razrada koncepta `izvor slike`, koristi se kao zajednička apstrakcija za sve komponente koje opisuju dohvat slike iz priključenog sklopovlja.
- `datotečni izvor`: razrada koncepta `izvor slike`, koristi se kao zajednička apstrakcija za sve komponente koje opisuju dohvat slike iz datoteke.
- `mrežni izvor`: konkretna izvedba koncepta `izvor slike`, u kojoj se svi zahtjevi prosljeđuju izvoru slike koji se nalazi u okviru mrežnog poslužitelja na udaljenom računalu.

Razmatrani paket je nazvan `vs` prema engleskoj skraćenici koncepta "izvor slike" (*engl.* video source). Organizacija komponenata paketa prikazana je na sl. B.1, dok su zaduženja pojedinih komponenata sažeta u tablici tablici B.2. Korisnici paketa trebali bi usluge konkretnih komponenata koristiti preko što apstraktnijih sučelja u skladu s mogućnostima, jer se tako "besplatno" dobiva neovisnost programa o konkretnom mehanizmu za pribavljanje slike. Koncept apstraktnog izvora slike uveden je od strane mr.sc. Vladimira Stanisavljevića, prema odgovarajućem konceptu biblioteke VIS koja je dostupna za operacijski sustav Solaris.



Slika B.1: Organizacija komponenata paketa za pribavljanje slike `vs`.

komponenta	kratki opis	korišteni resursi
`vs_base`	koncept `izvor slike`	—
`vs_grab`	koncept `sklopovski izvor`	—
`vs_file`	koncept `datotečni izvor`	—
`vs_grab_unixV4L`	analogni digitalizatori	biblioteka Video for Linux
`vs_grab_unix1394`	kamere po DCAM specifikaciji	biblioteka dc1394
`vs_grab_w32MCI`	analogni digitalizatori	biblioteka Video for Windows
`vs_grab_w32MCI`	sklopovlje tvrtke Imagenation	vlasničke biblioteke
`vs_file_avi`	datoteke formata `avi`	biblioteka Video for Windows
`vs_file_single`	datoteke formata `bmp`,`ras`,`ppm`	paket `dib_io`
`vs_file_srs`	datoteka sa slijedom slika `ras`	standardna biblioteka jezika
`vs_file_txt`	opisna datoteka, vidi A.3.1	standardna biblioteka jezika

Tablica B.2: Konkretne komponente paketa za pribavljanje slike `vs`.

B.2.2 Stvaranje konkretnih objekata

Stvaranje novih konkretnih objekata je najosjetljiviji trenutak u programu s apstraktnim sučeljima [alexandrescu01] jer je korištenje polimorfnih poziva dostupno tek kad je stvaranje objekta već dovršeno. Tipičan način rješavanja tog problema je dokumentiran kao obrazac tvornice (*engl.* Abstract factory) [gamma95], koja u ovisnosti o ulaznom parametru stvara konkretni objekt odgovarajućeg tipa. Taj obrazac je korišten npr. u komponenti `vs_fileUtil` za stvaranje objekata koji nasljeđuju sučelje `vs_file`, gdje se u ovisnosti o ekstenziji zadane datoteke i ciljnom operacijskom sustavu poziva odgovarajući konstruktor ili generira programska iznimka. Dobra strana ovog rješenja je što su klijenti izolirani od promjena u hijerarhiji za koju stvaranje objekata obavlja dodijeljena tvornica. Nedostatak ovakvog pristupa se očituje prilikom dodavanja novih konkretnih tipova koji nasljeđuju osnovnu klasu, jer je tada u kôd tvornice potrebno uvrstiti proceduru za testiranje ulaznog parametra i eventualno stvaranje objekta novog tipa. Primjerice, nakon dodavanja komponente za pristup MPEG2 datotekama s ekstenzijom `.mpg`, u komponentu `vs_fileutil` bi trebalo uvrstiti odsječak koji testira ime datoteke i po potrebi stvara novi objekt novog tipa `vs_file_mpg`. To nije sasvim zadovoljavajuće jer promjene kôda nisu potpuno lokalizirane: dodavanje jedne funkcionalnosti se uvijek odražava promjenama većeg broja komponenata. Objektno orijentirani pristup oblikovanju preporučuje upravo suprotnu praksu, da se operacije izvode nad apstraktnim podatcima, tako da najveći dio biblioteke bude potpuno imun na promjene komponenata niže razine.

Modernije rješenje tog problema koristi predloške da bi se izbjeglo ručno kodiranje uvjeta u tvornici. Jedan recept za takvo rješenje opisan je u [alexandrescu01], ali nažalost još nije upotrebljiv jer dostupni prevodioci još ne podržavaju neke sofisticirane zahtjeve standarda jezika [cxx98] na kojima se recept temelji. Stoga je oblikovano manje elegantno rješenje koje koristi uhodanije tehnike pa se može koristiti na svim prevodiocima. Spomenuto rješenje je sadržano u komponenti `util_registry`, a temelji se na statičkoj identifikacijskoj metodi `s_name()` te posebnom statičkom kreacijskom objektu kojeg moraju instancirati sve konkretne klase. Tip kreacijskog objekta je parametriziran i s tipom bazne klase i s tipom izvedene klase, tako da njegova instancijacija u slučaju komponente `vs_grab_w32PXC` izgleda kako slijedi:

```
util::RegistryCreator<vs_grab,vs_grab_w32PXC> creator;
```

Konstruktor statičkog kreacijskog objekta šalje svoju adresu centralnom registru koji je izveden prema obrascu jedinstvenog objekta (*engl.* singleton) [gamma95]. Centralni registar sadrži asocijativno polje `map_` čiji se elementi adresiraju prema identifikatoru bazne klase `typeid(Base)`. Elementi tog polja su ponovo asocijativna polja koja se ovaj put adresiraju prema identifikatoru konkretne klase `s_name()` (kreacijski objekti su parametrizirani s konkretnom klasom pa mogu pristupiti njenim statičkim metodama). Elementi svakog ugniježdenog polja su pokazivači na kreacijske objekte klasa koje nasljeđuju dotičnu baznu klasu. Tako svaka konkretna komponenta instancira statički objekt koji iz svog konstruktora obznani svoje postojanje centralnom registru, koji onda pohrani referencu na kreacijski objekt u polje koje odgovara baznoj klasi hijerarhije. Implementacija predloška za kreacijske objekte izgleda kako slijedi (bazna klasa predloška je potrebna za polimorfno baratanje kreacijskim objektima iz centralnog registra, a izuzeta je zbog konciznosti prikaza):

```
class Registry;
template <class Base, class Derived>
class RegistryCreator:
  public RegistryCreatorAbstract<Base>
{
```

```
public:
  RegistryCreator(){
    Registry::instance().registerCreator(
      typeid(Base).name(), this);
  }
private:
  virtual char const* name_derived(){
    return Derived::s_name();
  }
  typedef typename Base::CtorArg CtorArg;
  virtual Base* create(const CtorArg& p){
    return new Derived(p);
  }
};
```

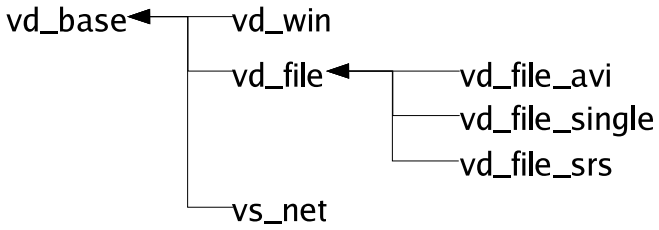
Kreiranje novih objekata tipično se izvodi iz predloška metode `create` centralnog registra koja se poziva s identifikatorom konkretne klase (parametar predloška je naravno bazna klasa). Centralni registar pristupa kreatoru konkretne klase preko primljenog identifikatora: ako uspije pronaći odgovarajući objekt, registar poziva njegovu kreacijsku metodu, inače generira programsku iznimku s dijagnostičkim tekstom. Kreacijskom objektu se prosljeđuje drugi argument predloška kreacijske klase, čiji tip mora biti definiran u baznoj klasi razmatrane hijerarhije pod imenom `CtorArg`. Pojednostavljena implementacija predloška kreacijske metode registra izgleda kako slijedi (izuzeti su detalji potrebni za uspješno prevođenje na jednom neispravnom prevodiocu):

```
class Registry{
  //...
  typedef std::map<Key, RegistryCreatorBase*> CreatorMap;
  typedef std::map<Key, CreatorMap>           RegistryMap;
  RegistryMap map_;

  // ...
  template<class Base>
  Base* create(
    char const* name_derived,
    const typename Base::CtorArg& par)
  {
    //dohvat polja pokazivača na kreatore
    RegistryMap::iterator it=map_.find(typeid(Base).name());
    if (it==map_.end()){
      std::string str("Registry::getCreators - illegal base class: ");
      str += typeid(Base).name();
      throw std::runtime_error(str);
    }
    CreatorMap& cm=it->second;

    //dohvat konkretnog kreatora
    CreatorMap::iterator it=cm.find(name_derived);
    if (it==end()){
      std::string err("util::registry::create<");
      err+=typeid(Base).name();
      err+="> - unsupported type: ";
      err+=name_derived;
```

```
      throw std::runtime_error(err);
    }

    //stvaranje konkretnog objekta
    RegistryCreatorAbstract<Base>& creator=
      dynamic_cast<RegistryCreatorAbstract<Base>&>(*it);
    return creator.create(par);
  }
};
```

B.2.3 Spremanje slike

Analogno konceptu izvora slike, može se se uvesti i koncept ponora slike, jer je slike ponekad potrebno prikazati na ekranu računala, a nekad ih je zgodno i pohraniti radi daljnje obrade. Motivacija je rasterećenje klijenata od izvedbenih detalja različitih programskih tehnologija, te omogućavanje što jednostavnijeg korištenja. Pored prozora i datoteka, moguće je razmišljati i o mrežnom ponoru slike, koji bi dobivene slike privremeno pamtio te stavljao na raspolaganje udaljenim klijentima koji bi se spajali preko prethodno spomenute komponente `vs_net`. Razmatrani paket je nazvan `vd` prema engleskoj skraćenici koncepta "ponor slike" (*engl.* video destination). Organizacija svih komponenata paketa prikazana je na sl. B.2.



Slika B.2: Organizacija komponenata paketa za spremanje slike `vd`.

Gotovo sva dodatna razmatranja vezana uz korištenje i stvaranje konkretnih komponenata paketa `vs` vrijede i za paket `vd`, pa se ovdje neće zasebno navoditi.

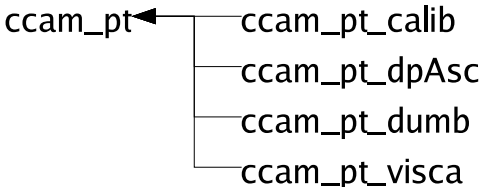
B.3 Upravljanje parametrima kamera

Kao što je prikazano u tablici A.2, paketi za enkapsulaciju vanjskih i unutrašnjih parametara kamere smješteni su unutar zajedničkog nadpaketa `ccam`. Struktura sastavnih dijelova tog nadpaketa biti će ukratko opisana u nastavku teksta ovog odjeljka.

B.3.1 Upravljanje postoljem s dva stupnja slobode

Domena paketa `ccam_pt` je upravljanje postoljem s dva stupnja slobode te modeliranje utjecaja parametara postolja na vanjske parametre kamere, a od ta dva zadatka, za sada je riješen samo prvi. Kao što je bio slučaj i s paketom `vs`, zahtijeva se izolirajuće apstraktno sučelje koje omogućava klijentima testiranje svog kôda neovisno o konkretnim izvedbama koje su također dio paketa. Sučelje bi trebalo podržavati očitavanje i postavljanje vrijedosti obaju parametara postolja, a od posebnog interesa su konkretne izvedbe za pokretne platforme koje su korištene u eksperimentalnom sustavu, a navedene su u 5.1.1. Gotovo sva

upravljiva postolja se spajaju na računalo preko serijske veze po standardu RS232, pa će važan parametar konkretnih izvedi sučelja biti identifikator serijskih vrata na koja je postolje priključeno. Taj identifikator se postavlja u skladu s konvencijama komandne linije cilja za kalibraciju kamera `cal`, koje su opisane u A.4.3. Organizacija komponenata paketa prikazana je na sl. B.3, dok je kratak opis konkretnih komponenata paketa dan u tablici B.3. Stvaranje objekata konkretnih komponenata se obavlja iz centralnog registra, pomoću komponente `util_registry`, kao što je opisano u odjeljku B.2.



Slika B.3: Organizacija komponenata paketa za upravljanje postoljem s dva stupnja slobode.

komponenta	kratki opis	korišteni resursi
`ccam_pt_calib`	kalibracijski podatci postolja	(nije implementirano)
`ccam_pt_dpAsc`	postolje DP PTU–46–17.5	standardna biblioteka jezika
`ccam_pt_dumb`	analogon /dev/null	standardna biblioteka jezika
`ccam_pt_visca`	standard tvrtke Sony	paket `ccam_visca`

Tablica B.3: Konkretne komponente paketa za upravljanje postoljem s dva stupnja slobode.

U pojednostavljenom obliku, sučelje komponente `ccam_pt` koja je u korijena hijerarhije paketa izgleda kako slijedi:

```
class ccam_pt{
protected:
  ccam_pt();
public:
  virtual ~ccam_pt();

// position
public:
  virtual void setPanTilt(double pan, double tilt);
  virtual void getPanTilt(double& p, double& t);
  virtual void setPanTiltSpeed(double ps, double ts);
  virtual void getPanTiltSpeed(double& ps, double& ts);

// synchronization
public:
  virtual bool blocking(){return false;}
  virtual bool moving(){return false;}
  virtual void waitCompletion(){}

//concrete ccam_pt construction
public:
  virtual char const* name()=0;
  // concrete classes ctor argument
  struct CtorArg{
```

```
    int port;
    CtorArg(int p):port(p){}
  };
  //factory for concrete classes
  static ccam_pt* create(const char* name, const CtorArg& arg);
};
```

Veći dio prikazanog sučelja je intuitivno jasan, pa ovdje neće biti dalje razmatran, s izuzetkom metoda `blocking`, `moving` i `waitCompletion`. Naime, konkretna pokretna postolja omogućuju sinkroni i asinkroni upravljački protokol s računalom domaćinom. Protokoli se razlikuju u načinu povratka iz operacija pomaka postolja: kod asinkronog protokola povratak se događa trenutno, dok se kod sinkronog protokola povratak događa tek nakon pomaka kamere na što se zbog trome mehanike tipično troši više stotina ms. Postolje DP PTU–46–17.5 omogućava potpuni asinkroni protokol sa nezavisnim postavljanjem kutnih brzina po osima slobode, dok primitivi standarda VISCA tvrtke Sony samo djelomično podržavaju takav protokol (npr, promjenu položaja moguće je specificirati samo kao apsolutne koordinate zakreta i nagiba). Stoga je u ovoj verziji biblioteke asinkroni protokol izveden u komponenti `ccam_pt_dpAsc`, dok komponenta `ccam_pt_visca` implementira sinkroni protokol.

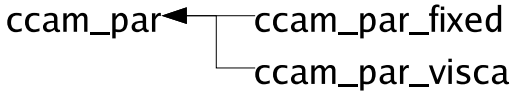
Vrstu implementiranog protokola moguće je saznati pozivom metode `blocking`: ako metoda vrati istinu, protokol je sinkroni. Sučelja koja implementiraju asinkroni protokol, definiraju i posebne izvedbe ostale dvije metode. Metoda `moving` vraća istinu, ako se pokretno postolje u trenutku poziva giba, dok se metodom `waitCompletion` omogućava čekanje izvršenja prethodne naredbe, tj. asinkroni protokol se privremeno pretvara u sinkroni. Definicije metoda u baznoj klasi odgovaraju sinkronom protokolu, pa takve konkretne komponente ne moraju uključivati izvedbu niti jedne od triju metoda.

B.3.2 Predstavljanje unutrašnjih parametara kamere

Unutrašnji parametri kamere predstavljaju domenu paketa `ccam_par`. Prvi zadatak tog paketa jest omogućavanje pristupa umjerenim parametrima za svaku konkretnu kameru, i on je ispunjen na zadovoljavajuć način. Drugi zadatak paketa nije riješen u trenutnoj verziji biblioteke, a svodi se na enkapsulaciju upravljanja kamerom s promjenljivim vidnim poljem, te pristupa njenim internim parametrima u ovisnosti o odabranom vidnom polju. Popis konkretnih komponenata paketa je dan u tablici B.4, dok je njihov međusobni odnos ilustriran na sl. B.4.

komponenta	kratki opis	korišteni resursi
`ccam_par_fixed`	kalibracija pasivne kamere	standardna biblioteka jezika
`ccam_par_visca`	standard tvrtke Sony	paket `ccam_visca` (nije dovršeno)

Tablica B.4: Konkretne komponente za upravljanje unutrašnjim parametrima kamere.



Slika B.4: Organizacija paketa za upravljanje unutrašnjim parametrima kamere.

Kao što se vidi u sl. B.4, korijen hijerarhije paketa `ccam_par` sadržan je u istoimenoj komponenti, čije sučelje u pojednostavljenom obliku izgleda kako slijedi:

```
class ccam_par{
protected:
  ccam_par();
public:
  virtual ~ccam_par();
//changing the field of view,
public:
  virtual double fov() const =0;
  virtual void setFov(double){};
//pinhole parameters
public:
  virtual int width() const =0;
  virtual double f() const =0;
  virtual double alpha() const =0;
  virtual double beta() const =0;
  virtual double gamma() const =0;
  virtual double u0() const =0;
  virtual double v0() const =0;
//radial distortion parameters
public:
  virtual double k1() const =0;
  virtual double k2() const =0;
  virtual double k1rev() const =0;
  virtual double k2rev() const =0;
//radial distortion operations
public:
  math::Point correct(const math::Point& pt) const;
  math::Point distort(const math::Point& pt) const;
//ccam_par construction
public:
  virtual char const* name() const =0;
  // concrete classes ctor argument
  struct CtorArg{
  public:
    int port;      // the port wrt rs232 package
    char const* id;// the calibration id
  public:
    CtorArg(int p, char const* i):port(p), id(i){}
  };
  //factory for the concrete classes
  static ccam_par* create(const char* name, const CtorArg& arg);
};
```

Prvi dio sučelja sadrži operacije za postavljanje i dohvat vodoravnog vidnog polja kamere (većini kamera će odgovarati ponuđena prazna implementacija). Nakon toga su navedene pristupne metode za pojedine unutrašnje parametre kamere, te operacije za primjenu i korigiranje radijalnog izobličenja čije su definicije sadržane u izvedbenoj datoteci komponente. Konačno, sučelje sadrži i infrastrukturu za kreiranje objekata konkretnih klasa što se obavlja upotrebom centralnog registra koji je definiran u komponenti `util_registry`, kao što je opisano u odjeljku B.2.

Dodatak C

Automatizirano prevođenje izvedbe eksperimentalnog sustava

Predmet ovog dodatka je detaljan opis koraka koji su potrebni za dohvat izvornog kôda biblioteke, te postupak prevođenja njenog najvažnijeg cilja, razvojne ljuske `cvsh`. Dodatak je organiziran prema sljedećim logičkim fazama postupka prevođenja: dohvat potrebnih alata, dohvat izvornog kôda, konfiguracija postupka prevođenja, prevođenje izvornih datoteka, te povezivanje prevedenih datoteka s potrebnim bibliotekama.

Temeljni postupak je prilagođen klasičnoj metodi prevođenja programskog sustava koja se temelji na alatu `make`. Pored takvog načina, međutim, omogućeno je i korištenje integrirane okoline prevodioca `msvc`, a odgovarajuća odstupanja od temeljnog postupka će u tekstu biti posebno naznačena. Preporuča se korištenje klasičnog pristupa jer je bolje razrađen, omogućava veći stupanj automatizacije postupka, te manje ovisi o verzijama korištenih programa. Dodatna prednost klasičnog pristupa je što se može koristiti na gotovo svakom operacijskom sustavu, dok je pristup s integriranom okolinom ograničen na operacijske sustave za koje se isporučuje integrirana okolina.

C.1 Potrebni alati i vanjske biblioteke

Svi alati i biblioteke koji se spominju u ovom dodatku detaljnije su opisani u odjeljku 5.2, a u ovom odjeljku će samo biti ukratko navedeni prema logičkoj fazi prevođenja u kojoj se koriste. Potrebni alati su sastavni dijelovi većine distribucija operacijskog sustava Linux, te bi se morali moći instalirati na bilo kojem UNIX-u. Kao najjednostavniji način za dobiti takvu okolinu na operacijskom sustavu MS–Windows, preporuča se instaliranje razvojnih okruženja Cygwin[1] i MinGW[2].

- alati za dohvat izvornog kôda: `cvs`, `python`;
- alati koji se koriste pri konfiguraciji postupka prevođenja:
  klasični `makefile`: `python`, `perl`, `tmake`,
  integrirana okolina: MSVC;
- alati i biblioteke koji se koriste pri prevođenju izvornog kôda:
  klasični `makefile`: `boost`, `tnt`, `make`, `gcc`,
  integrirana okolina: MSVC;

[1] http://sources.redhat.com/cygwin/
[2] http://www.mingw.org/

- biblioteke koje se koriste pri povezivanju prevedenih datoteka:
  Linux: `libImlib.so`,
  MS–Windows: `vfw32.lib`.

C.2 Dohvat izvornog kôda

U ovom odjeljku će biti opisani detalji dohvata potrebnih paketa iz mrežne biblioteke kojom upravlja poslužioc `cvs`.

C.2.1 Potrebne predradnje

Prije samog dohvata paketa biblioteke, potrebno je instalirati i konfigurirati klijent programskog paketa `cvs`. Potrebne operacije detaljno su razrađene u obliku sljedeće liste:

1. Po potrebi dohvatiti i instalirati neki klijent programskog paketa `cvs`.
2. Postaviti `CVSROOT` varijablu klijenta na:

   ```
   :pserver:anonymous@dupli.zemris.fer.hr:/home/cvs/pubcvs.
   ```

   Kod tekstualnih klijenata, to se postiže postavljanjem istoimene varijable korisničke ljuske, dok kod grafičkih klijenata prikazani niz treba upisati u odgovarajuću kućicu konfiguracijskog dijaloga.
3. Prijaviti se za rad s mrežnim poslužiocem biblioteke, uz praznu lozinku: za tekstualnog klijenta, upisati naredbu: `cvs login`.
4. Provjeriti konfiguraciju dohvaćanjem modula `cvsh_docs` u kojem se nalazi dokumentacija projekta: kod tekstualnog klijenta, upisati naredbu: `cvs checkout cvsh_docs`.
5. Važna napomena: firewall na računalu `dupli.zemris.fer.hr` ne dozvoljava spajanje s računala izvan ZEMRIS-a.

C.2.2 Dohvat skripti

Pored izvornog kôda, biblioteka definira i nekoliko pomoćnih skripti pisanih u Perlu i Pythonu, koje služe za automatiziranje konfiguracije procesa prevođenja. Te skripte je moguće dohvatiti u skladu sa sljedećom procedurom, uz pretpostavku tekstualnog klijenta `cvs`:

```
cvs login
cvs checkout cvs_scripts
export PATH=$PATH:$PWD/cvs_scripts
```

C.2.3 Dohvat opisnih datoteka projekta i izvornog kôda

Iako je u ovom trenutku za dohvat kompletne biblioteke dovoljno reći `cvs checkout src`, nema potrebe za pribavljanje paketa koji se u tekućem projektu ne koriste. Selektivni dohvat pojedinih paketa može se obaviti skriptom `ss_checkout.py`, čiji dohvat je opisan u C.2.2. Kao argument skripte, navodi se opisna datoteka projekta koju je prethodno potrebno dohvatiti, u skladu s uputama u nastavku teksta. Pored izvornog kôda, potrebno je pribaviti i modul `tmake_scripts` koji sadrži konfiguracijske skripte čiji zadatak je otkrivanje svojstava računala na kojem se prevođenje odvija. Skripte ovise o konkretnom operacijskom

sustavu, pa mogu detektirati prisustvo pojedinih izvršnih biblioteka, te po potrebi uključiti ili isključiti odgovarajuće komponente projekta. Stoga preporučena procedura glasi:

```
mkdir projects; cd projects
cvs checkout tmake_scripts
mkdir cvsh; cd cvsh
cvs checkout cvsh_shell
ss_checkout.py cvsh_shell/project.conf
```

Nakon opisanog slijeda naredbi, paketi izvornog kôda bi trebali biti u direktoriju:

```
.../projects/cvsh/src,
```

dok su datoteke potrebne za prevođenje projekta sadržane u direktoriju:

```
.../projects/cvsh/cvsh_shell.
```

Prednost ovakve organizacije je da jasno odjeljuje izvorni kôd od detalja prevođenja na različitim ciljnim operacijskim sustavima. Direktorij `cvsh_shell` sadrži poddirektorij:

```
.../projects/cvsh/cvsh_shell/tmake
```

u kojem se kreiraju `makefile`ovi, ovisno o ciljnoj konfiguraciji, dok su opisne datoteke za razvojnu okolinu prevodioca `msvc` sadržane u direktoriju:

```
.../projects/cvsh/cvsh_shell/msvc.
```

Datoteka `project.conf` sastoji se od više odjeljaka. Odjeljak `NAME` definira ime izvršne datoteke. Odjeljak `PACKAGES` navodi potrebne komponente i pakete izvornog kôda biblioteke: u pravilu, prevode se sve komponente navedenih paketa, ali je u odjeljku `FILTER` moguće navesti iznimke. U odjeljku `UNFILTER` moguće je eksplicite navesti komponente koje bi tijekom automatskog stvaranja `makefile`a bile izuzete, npr, zato što opisuju ispitne programe. Većina tih komponenti definira vlastitu funkciju `main()`, pa se ukupni program ne može povezati uz njihovo prisustvo. Međutim, neke ispitne komponente su izvedene kao postupci ljuske (nasljeđuju sučelje `ip_base`), pa se mogu nesmetano uklopiti u glavni program, kao što je to slučaj s komponentom `alg_matchGrid_test`. Konačno, eventualni specifični konfiguracijski zadaci navode se u odjeljku `CONFIG`. Datoteka `project.conf` za cilj `cvsh` izgleda kako slijedi:

```
NAME=cvsh
PACKAGES=
  alg
  cli
  cvsh
  dib
  ext/susan
  ext/lmdif
  geom
  ip
  math/math_planar.cpp
  math/math_planar.h
  math/math_linalg.cpp
  math/math_linalgTnt.cpp
  math/math_linalgTnt.h
  math/math_linalg.h
  math/cal
  pd
```

```
  thr
  ui
  util
  vd
  vs
  win
FILTER=
  thr_IPserver
  ip_dllAdaptor
  ip_menu
UNFILTER=
  alg_matchGrid_test
CONFIG= ieee1394
```

Datoteka `project.conf` se koristi i iz skripte `ss_checkout.py` prilikom dohvata izvornog kôda, kao pokazatelj komponenti koje je potrebno dohvatiti, i u kasnijoj fazi, prilikom automatiziranog generiranja `makefile`a.

C.3 Konfiguracija postupka prevođenja

Ukoliko se koristi integrirana okolina msvc, u ovom trenutku podrška prestaje: potrebno je učitati opisnu datoteku projekta:

`.../projects/cvsh/cvsh_shell/msvc/<verzija>/<ime_datoteke>,`

te ručno podesiti parametre koji ne odgovaraju (npr, direktorij s izvršnom datotekom). U slučaju problema, potrebno je analizirati dijagnostičke poruke te pokušati pronaći problem, npr, izuzimanjem problematične komponente iz projekta.

Inače, potrebno je slijediti sljedeću proceduru:

1. Dohvatiti i instalirati alat `tmake`, staviti odgovarajuću datoteku u izvršni put (`PATH`), te postaviti varijablu `TMAKEPATH` u skladu s platformom. Na Linuxu, varijabla bi se mogla postaviti sljedećom naredbom:

   ```
   export TMAKEPATH=/usr/local/tmake/lib/linux-g++/
   ```

2. Otići u odgovarajući radni direktorij:

   ```
   cd .../projects/cvsh/cvsh_shell/tmake/<platform>/<build-type>.
   ```

   U gornjoj naredbi, `<platform>` označava ciljni operacijski sustav, a podržane vrijednosti su `linux-g++` i `w32-g++`. Dijagnostičke procedure mogu usporiti program do neupotrebljivosti, pa je ustaljena praksa da se istovremeno održavaju barem dvije inačice izvršnog programa, `release` za ispitivanje u stvarnom vremenu, i `debug` za traženje grešaka. Svakoj inačici programa tako odgovara po jedan poddirektorij koji je u gornjoj naredbi označen s `build_type`, u kojem će se u sljedećem koraku generirati zaseban `makefile`. Trenutno su za svaki izvršni program biblioteke dostupne samo inačice `debug` i `release`.

3. Pokrenuti skriptu `./tmake_gen.sh`, koja uz pomoć skripte `ss_tmake_gen.py` modula `cvs_scripts` generira odgovarajući `makefile`.

Skripta `ss_tmake_gen.py` generira `makefile` koji prevodi samo one komponente stabla koje su specificirane datotekom `project.conf`. To znači da je moguće paralelno razvijanje više ciljeva nad istim stablom izvornog kôda. Iz `makefile`a se izuzimaju sve datoteke koje u nazivu imaju nizove `_test` i `_old`, osim ako nisu eksplicite spomenute u odjeljku `UNFILTER` opisne datoteke.

Konfiguracija postupka prevođenja se svodi na određivanje opcija korištenog prevodioca, i odvija se na tri razine. Na prvoj razini, opcije se postavljaju u ovisnosti o datoteci

`.../projects/cvsh/cvsh_shell/tmake/<platform>/config,`

neovisno o računalu na kojem se prevođenje odvija. Skripta može inkrementalno utjecati na opcije programa za prevođenje i povezivanje, te na skup datoteka koje se izuzimaju iz `makefile`a.

Na drugoj razini, pozivaju se skripte koje postavljaju opcije koje ovise o svojstvima računala na kojem se program prevodi i na kojem će se program izvršavati, a nalaze se u direktoriju

`.../projects/tmake_scripts/<platform>.`

Skripta `basic` se poziva uvijek, a ostale skripte se pozivaju samo ako su navedene u odjeljku `CONFIG` datoteke `project.conf`. Kao i u prethodnoj razini, skripte mogu utjecati na izuzimanje pojedinih komponenti, te na opcije prevođenja i povezivanja. Tako datoteka `linux-g++/basic` izuzima sve datoteke koje su specifične za drugu platformu (komponente u nazivu imaju niz `_w32`), dok datoteka `linux-g++/ieee1394` izuzima komponentu `unix1394` ako nije detektirana odgovarajuća podrška operacijskog sustava.

Konačno, na trećoj razini, na konfiguraciju utječe i inačica izvršne datoteke, i to preko skripte:

`.../projects/cvsh/cvsh_shell/tmake/<platform>/<build_type>/tmake_gen.sh.`

Primjerice, skripta u direktoriju namijenjenom prevođenju inačice `debug` koja se koristi pri ispravljanju grešaka, postavlja varijablu `TMAKEDBG` na temelju koje skripta `ss_tmake_gen.py` odabire odgovarajuće opcije prevodioca. Prevedene datoteke i konačna izvršna datoteka se uvijek smještaju u direktorij:

`~/tmp/tmake/<projekt>/<build-type>,`

što omogućava usporedno eksperimentiranje s više inačica istog programa.

Zbog povećanja brzine prevođenja, disk na kojem se smještaju prevedene datoteke bi trebao biti brz, dok mu pouzdanost nije kritična. Te zahtjeve idealno zadovoljavaju konfiguracije više diskova bez redundancije (RAID 0). Smještanje prevedenih datoteka na proizvoljnom logičkom disku računala može se postići vrlo jednostavno, kreiranjem simboličkog linka u `~/tmp/tmake`.

C.4 Nezavisno ispitivanje komponenata

Nezavisno ispitivanje komponenata se navodi kao vrlo važna tehnika pri razvoju velikih programskih sustava [lakos96, hunt99]. U principu, svaka komponenta bi trebala imati svoj ispitni program (*engl.* test driver), koji bi trebalo dati uvid u korektno izvršavanje i performansu komponente kojoj je pridružen. Ispitne komponente se obično fizički smještaju zajedno s ispitivanom komponentom, od koje preuzimaju i naziv, uz dodavanje dogovornog sufiksa, npr. `_test` ili `_t`. Ispitne datoteke nisu predviđene za korištenje iz drugih komponenti pa nikad nemaju sučeljnu datoteku, odnosno sastoje se samo od izvedbene datoteke

koja u pravilu definira vlastitu funkciju `main()`. Slično kao i s dokumentacijom, količina ispitnog kôda ne doprinosi novu funkcionalnost proizvodu, iako u znatnoj mjeri određuje njegovu kvalitetu. Sličnosti tu ne prestaju, jer ispitni kôd eksplicite pokazuje način korištenja komponente kojoj je pridružen, što znači da su postupci dokumentiranja i ispitivanja programskih sustava međusobno isprepleteni i srodni. Nažalost, kao i kod dokumentacije, sveobuhvatno ispitivanje mogu sebi priuštiti samo veliki razvojni timovi gdje ljudski resursi nisu strogo ograničeni. U većini stvarnih situacija stoga je potrebno naći optimalan odnos između uloženog truda i dobivenih rezultata. U svom trenutnom stanju, biblioteka izvornog kôda sadrži samo deset ispitnih komponenti, što znači da se otprilike svaka dvadeseta komponenta biblioteke može nesmetano nezavisno testirati. Poboljšanje tog omjera je jedan od prioritetnih ciljeva daljnjeg rada.

C.4.1 Ispitna komponente paketa za prikaz slike

Postupak ispitivanja će se ilustrirati na jednom od temeljnih podpaketa biblioteke. Paket `dib_access` sadrži komponente koje omogućavaju pristup elementima slike upotrebom statičkog polimorfizma [czarnecki00], koji se u programskom jeziku C++ ostvaruje predlošcima. U konkretnoj izvedbi, omogućava se efikasan pristup slikovnim elementima, kao kod izravne primjene pokazivača, uz prilagodljivost i sigurnost koje se klasičnim tehnikama mogu postići samo uz gubitak efikasnosti. Očito, radi se o vrlo osjetljivom području, pa je odgovarajuće komponente nakon svake nove preinake potrebno dobro ispitati na različitim prevodiocima. Ispitni postupak je izveden u komponenti `dib_access_test`, čiji se postupak prevođenja u mnogome poklapa se postupkom prevođenja ciljeva biblioteke koji je opisan u prethodnim odjeljcima. Prvo je potrebno dohvatiti opisne datoteke ispitnog projekta, koji je nazvan `dibbench`, jer između ostalog ispituje i performansu izvođenja temeljnih operacija nad slikom. Pored opisnih datoteka projekta, dohvaćaju se i potrebne datoteke izvornog kôda.

```
cd .../projects/
mkdir -p test/dibbench
cd test/dibbench
cvs co dibbench
ss_checkout.py dibbench/project.conf
```

Iz opisne datoteke projekta se vidi da su za prevođenje ispitnog programa dovoljni paket `dib` i neke komponente paketa `util`:

```
NAME = dibbench
PACKAGES=
  dib
  util
FILTER=
  util_avi
  util_dll
  util_math
UNFILTER=
  dib_access_test
```

Sada je još samo potrebno napraviti makefile, te prevesti i pokrenuti ispitni program. Rezultati ispitivanja su pri tome (i) uspješno prevođenje i (ii) dobivanje očekivanih rezultata.

```
cd .../projects/test/dibbench/dibbench/tmake/linux-g++/release
./tmake_gen.sh
make
/tmp/tmake/dibbench/release/dibbench
```

C.4.2 Automatsko ispitivanje projekta

Opisane naredbe bi se mogle izdavati i automatski, iz skripte čije izvođenje se pokreće svake večeri, i to za svaku komponentu koja ima ispitni program. Automatski program za ispitivanje bi mogao sve rezultate sažeti u preglednu tablicu te je nakon večernjeg testiranja slati elektronskom poštom svim sudionicima projekta. U ovakvom okruženju, ljudi koji razvijaju nova svojstva se ne moraju bojati da bi mogli unijeti nove greške u prethodno ispravan kôd, jer se kompletna funkcionalnost programa svakodnevno ispituje, dok je čovjeku još na umu što je i zašto dodavao ili mijenjao. Konačno, ovakvo agresivno ispitivanje omogućava konstantan uvid u stanje razvoja projekta, što je posebno važno za njegovu komercijalizaciju odnosno planiranje isporuke konačnog proizvoda. Zbog svih spomenutih razloga, automatizirano ispitivanje se sve više koristi u ozbiljnijim projektima [hunt99].

C.5 Završne napomene

C.5.1 Prevođenje projekta

Projekt se prevodi ili pozivanjem odgovarajuće naredbe integriranog okruženja, ili izvršavanjem naredbe `make` iz korisničke ljuske. Obratiti pozornost da je za uspješno povezivanje potrebno imati barem biblioteke `libImlib.so` pod Linuxom odnosno `vfw.lib` pod Windowsima, što ne bi smio biti problem. Dobiveni program će imati podršku za pribavljanje slike preko generičkih sučelja `V4L` te `MCI`. Ukoliko se žele koristiti komponente `vs_grab_unix1394` i `vs_grab_w32PXC`, potrebno je instalirati odgovarajuće biblioteke. U prvom slučaju, dovoljno bi trebalo biti slijediti upute na `http://www.linux1394.org/`, dok je u drugom slučaju potrebno (i) uključiti komponentu u projekt, (ii) podesiti putove do direktorija sučelja i izvedbe biblioteka, te (iii) dodati potrebne biblioteke u dijalogu s postavkama programa za povezivanje prevedenih datoteka.

C.5.2 Pokretanje izvršne datoteke

Ispravno prevođenje programa se može provjeriti navođenjem posebne konkretne komponente za pribavljanje slike, koja uvijek vraća istu sliku:

```
.../cvsh -sg=test -a=homography -dw
```

Navedeni postupak je izveden u komponenti `ip_homography`, a zadatak mu je (i) određivanje homografije iz izvorne slike u zadani četverokut odredišne slike, te (ii), primjena dobivenog preslikavanja metodom koja se inače koristi za korekciju radijalnog izobličenja, a opisana je u 6.1.2. Četverokut odredišne slike može se zadavati i u toku izvođenja programa, pritiskom na lijevu tipku miša u odredišnom prozoru ljuske prema tablici C.1.

akcija	definirana točka četverokuta
`<Ctrl>` + lijeva tipka miša	donja – lijeva
`<Shift>` + lijeva tipka miša	gornja – lijeva
lijeva tipka miša	gornja – desna
`<Ctrl>+<Shift>` + lijeva tipka miša	donja – desna

Tablica C.1: Podešavanje parametara postupka `ip_homography`.

C.5.3 Ispitivanje performanse izvršne datoteke

Ispitivanje performanse složenih programa je dosta osjetljivo područje, jer na performansu, pored samog procesora, utječe cijeli niz činioca: brzina memorije, brzina diskova, propusnost sabirnice i brzina iscrtavanja. Međutim, pristup diskovima i iscrtavanje rezultata su obično važni samo u fazi razvoja, kao uvid u rezultate koji se dobivaju na dobro odabranom skupu prethodno pribavljenih slika. Kod ozbiljnih industrijskih primjena, slike se obično pribavljaju izravno iz sklopovlja, dok se rezultati obrade mogu po volji sažeti, ili ih nije ni potrebno prikazati u stvarnom vremenu. Stoga bi se ispitivanje performanse pojedinih algoritama računarskog vida trebalo odvijati pod uvjetima u kojima su relevantni samo bitni činioci: procesor, memorija i sabirnica. To je moguće postići odabirom neinteraktivnog rada ljuske, uz korištenje posebnog konkretnog izvora slike koji uvijek vraća istu sliku, te posebnog konkretnog ponora slike, koji s dobivenom slikom ne radi ništa. U slučaju postupka koji opisuje Cannyjevo detektiranje rubova, operacija bi bila zadana kako slijedi (zahtijeva se obrada sto sivih slika dimenzija 384×288, u kojima je svaki slikovni element predstavljen jednim bajtom):

```
.../cvsh -sg=test@0:384x288x1#0-100 -a=canny_w0 -df=/dev/null
```

Cannyjev detektor je jedan od najčešće korištenih postupaka računarskog vida, pa je zanimljivo vidjeti koliko traje njegova primjena na različitim računalima. Tablica C.2 sadrži vremena izvršavanja navedene naredbe podijeljena sa 100, koliko iznosi broj obrađenih slika.

procesor	radna frekvencija	vrijeme izvođenja	učestalost obrade
AMD Opteron	2.0 GHz	15.8 ms	64 Hz
Intel Xeon	3.0 GHz	17.2 ms	58 Hz
Intel Pentium 4m	2.2 GHz	25.0 ms	40 Hz
AMD Athlon	1.4 GHz	34.5 ms	29 Hz

Tablica C.2: Trajanje izvođenja postupka biblioteke koji sadrži izvedbu Cannyjevog detektora rubova, za sliku dimenzija 384×288 uz 1 bajt po slikovnom elementu, na računalima s različitim procesorima.

C.5.4 Sažetak korištenih varijabli korisničke ljuske

```
export CVSROOT=:pserver:anonymous@dupli.zemris.fer.hr:/home/cvs/pubcvs
export PATH=$PATH:~/projects/cvs_scripts
export TMAKEPATH=/usr/local/tmake/lib/linux-g++/
export PATH=$PATH:/usr/local/tmake/bin/
```

C.5.5 Rješenja čestih problema

- Problemi s instanciranjem komponente `vs_grab_unix1394` mogu biti uzrokovani nedovoljnim ovlastima korisnika nad datotekama `/dev/video*`.
- Modul biblioteke `tmake_scripts` mora biti jednu razinu direktorija iznad modula `src` i `cvsh_shell`, inače ga skripta `ss_tmake_gen.py` neće naći.

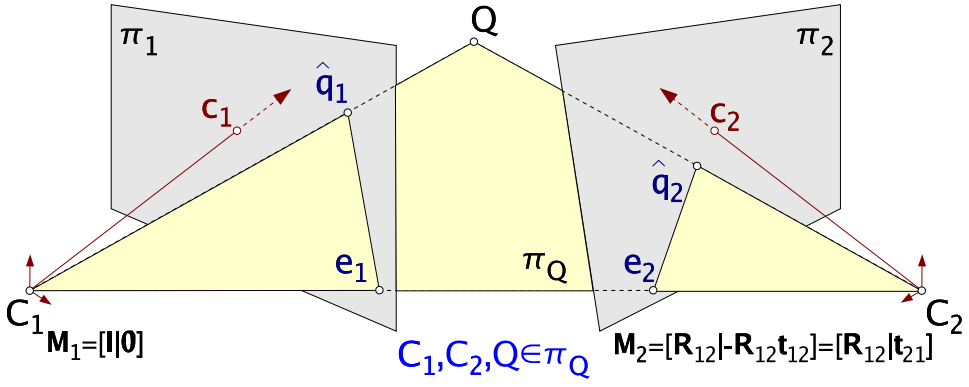
Dodatak D

Epipolarno ograničenje generaliziranog stereo vida

Neka je zadan generalizirani problem stereo vida uz korištenje normaliziranih koordinata slikovnih elemenata kao što je prikazano na sl. D.1. Neka p_q označava sve točke prostora koje se preslikavaju u zadanu točku slike $\hat{\mathbf{q}}$, tako da vrijedi:

$$\hat{\mathbf{q}} = \mathbf{S} \cdot \mathbf{M} \cdot \mathbf{Q}_{p_q}, \quad \forall \mathbf{Q}_{p_q} \in p_q. \tag{D.1}$$

U takvom sustavu vrijedi pravilo, da se za proizvoljnu točku prve slike $\hat{\mathbf{q}}_\mathbf{1} \in \pi_1$, svi mogući originali iz k.s. svijeta $\mathbf{Q} \in p_{q_1}$ preslikavaju u točno određeni *epipolarni* pravac druge slike.



Slika D.1: Epipolarno ograničenje stereo vida.

Prema sl. D.1, vektori $\overrightarrow{C_1Q}$, $\overrightarrow{C_2Q}$ i $\overrightarrow{C_1C_2}$ su koplanarni. Glavno svojstvo normaliziranog koordinatnog sustava slike je da jedinica po obje koordinatne osi odgovara udaljenosti slikovne ravnine od glavne točke projekcije. Posljedica toga je da se elementi n.k.s. slike mogu interpretirati kao 3D vektori, čiji smjer je paralelan s odgovarajućom svjetlosnom zrakom 3D prostora. Ishodište k.s. svijeta se poklapa s ishodištem k.s. prve kamere C_1, pa se spomenuti vektori mogu izraziti pomoću sljedećih elemenata projekcijske ravnine:

$$\begin{aligned} \overrightarrow{C_1Q} &= \lambda_1 \cdot \hat{\mathbf{q}}_1 \\ \overrightarrow{C_2Q} &= \lambda_2 \cdot \mathbf{R_{21}} \hat{\mathbf{q}}_2 \\ \overrightarrow{C_1C_2} &= \lambda_3 \cdot \mathbf{t}_{21} \end{aligned} \tag{D.2}$$

Uvjet koplanarnosti triju vektora može se izraziti sljedećom jednadžbom:

$$\hat{\mathbf{q}}_1^T \cdot (\mathbf{t}_{21} \times \mathbf{R_{21}} \cdot \hat{\mathbf{q}}_2) = 0 \tag{D.3}$$

Daljnjom razradom jednadžbe, dobiva se jednadžba epipolarnog pravca u π_2, koji odgovara proizvoljnoj točki $\hat{\mathbf{q}}_1 \in \pi_1$:

$$\begin{aligned} \hat{\mathbf{q}}_1^T \cdot ([\mathbf{t}_{21\times}] \cdot \mathbf{R_{21}}) \cdot \hat{\mathbf{q}}_2 &= \\ \hat{\mathbf{q}}_1 \cdot \mathcal{E} \cdot \hat{\mathbf{q}}_2 &= 0 \end{aligned} \tag{D.4}$$

Matrica $\mathcal{E}$ se naziva *esencijalnom* matricom i opisuje stereo korespondenciju u normaliziranim k.s. slike pa se može primijeniti isključivo kod kalibriranih kamera. Sadržaj te matrice definiran je prostornim rasporedom dvaju kamera, te se u slučaju da su pomak $\mathbf{t}_{21}$ i rotacija $\mathbf{R}_{21}$ koji opisuju prijelaz među k.s. dvaju kamera poznati, može dobiti algebarskim putem:

$$\begin{aligned} \mathcal{E} &= [\mathbf{t}_{21\times}] \cdot \mathbf{R}_{21} \\ &= \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \cdot \mathbf{R}_{21} \end{aligned} \tag{D.5}$$

Prikaze točaka u normaliziranom k.s. slike je moguće izraziti odgovarajućim prikazima u uobičajenom k.s. slike, uz množenje s odgovarajućom invertiranom matricom unutrašnjih parametara $\mathbf{K}$. Ograničenje (D.4) u tom slučaju glasi:

$$\begin{aligned} (\mathbf{K}_1^{-1}\hat{\mathbf{q}}_1)^T \cdot \mathcal{E} \cdot (\mathbf{K}_1^{-1}\hat{\mathbf{q}}_1) &= \\ \mathbf{q}_1^T \cdot (\mathbf{K}_1^{-T} \cdot \mathcal{E} \cdot \mathbf{K}_2^{-1}) \cdot \mathbf{q}_2 &= \\ \hat{\mathbf{q}}_1^T \cdot \mathcal{F} \cdot \hat{\mathbf{q}}_2 &= 0 \end{aligned} \tag{D.6}$$

Matrica $\mathcal{F}$ se naziva fundamentalnom matricom i opisuje stereo korespondenciju u k.s. slike kao što je prikazano u jednadžbi (D.6), pa se može umjeriti i kada kamere nisu kalibrirane. Za algebarski izvod te matrice međutim, potrebno je znati unutrašnje parametre dvaju kamera:

$$\mathcal{F} = \mathbf{K}_1^{-T} \cdot \mathcal{E} \cdot \mathbf{K}_2^{-1} = \mathbf{K}_1^{-T} \cdot [\mathbf{t}_{12\times}] \cdot \mathbf{R}_{12} \cdot \mathbf{K}_2^{-1} \tag{D.7}$$

Bibliografija

[alexandrescu01] Andrei Alexandrescu, *Modern C++ Design*, Addison-Wesley, 2001.

[bajcsy88] Ruzena K. Bajcsy, Active Perception, *Proceedings of the IEEE*, 76(8):996–1005, August 1988.

[bianchi96] Reinaldo A.C. Bianchi and Anna H.R.C. Rillo, A Purposive Computer Vision System: a Multi-Agent Approach, in *Proceedings of the Workshop on Cybernetic Vision*, 225–230, IEEE, São Carlos, Brazil, 1996.

[borenstein96] J. Borenstein, H. R. Everett and L. Feng, *Navigating Mobile Robots: Sensors and Techniques*, A. K. Peters, Ltd., Wellesley, MA, January 1996.

[boucher98] Alain Boucher, Anne Doisy et al., A society of goal-oriented agents for the analysis of living cells, *Artificial Intelligence in Medicine*, 14(1–2):183–199, September 1998.

[bradshaw99] Jeffrey M. Bradshaw, Mark Greaves et al., Agents for the Masses?, *IEEE Intelligent Systems*, 14(2):53–63, March 1999.

[brooks95] Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*, Addison Wesley Longman, Reading, Massachusetts, USA, 2 edn., 1995 (1975).

[buschmann96] Frank Buschmann, Regine Meunier et al., *A System of Patterns*, John Wiley & Sons, West Sussex, England, 1996.

[cai99] Q. Cai and J.K. Aggarwal, Tracking Human Motion in Structured Environments Using a Distributed-Camera System, *IEEE Transactions on Pattern recognition and Machine Intelligence*, 21(11):1241–1247, November 1999.

[cambridge:www] Cambridge University Press, Advanced Learner's Dictionary of English Language, british english.
URL `http://dictionary.cambridge.org`

[chaib95] B. Chaib-draa, Industrial Applications of Distributed AI, *Communications of the ACM*, 38(11):49–53, November 1995.

[chaib02] B. Chaib-draa and F. Dignum, Trends in agent communication language, *Computational Intelligence*, 18(2):89–101, May 2002.

[chang01] Ting-Hsun Chang and Shaogang Gong, Tracking Multiple People with a Multi-Camera System, in *Proceedings of Workshop on Multi-Object Tracking*, 19–28, Vancouver, Canada, August 2001.

[collins01] Robert Collins, Alan Lipton et al., Algorithms for cooperative multisensor surveillance, *Proceedings of the IEEE*, 89(10):1456–1477, October 2001.

[collins99] Robert Collins and Yanghai Tsin, Calibration of an Outdoor Active Camera System, in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 528–534, Fort Collins, Colorado, June 1999.

[czarnecki00] Krysztof Czarnecki and Ulrich Eisenecker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.

[dockstader01] A. Murat Dockstader, Shiloh L.; Tekalp, Multiple Camera Tracking of Interacting and Occluded Human Motion, *Proceedings of the IEEE*, 89(10):1441–1455, October 2001.

[duchesnay03] Edouard Duchesnay, Jean-Jacques Montois and Yann Jacquelet, Cooperative agents society organized as an irregular pyramid: A mammography segmentation application, *Pattern Recognition Letters*, 24(14):2435–2445, October 2003.

[faugeras93] O.D. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, 1993.

[ferber99] Jacques Ferber, *Multi-Agent Systems*, Addison Wesley Longman, New York, USA, 1999.

[gamma95] Erich Gamma, Richard Helm et al., *Design Patterns: Elements of Reusable Object-Oriented systems*, Professional Computing Series, Addison-Wesley, Reading, Massachusetts, USA, 1995.

[gavrila99] Dariu M. Gavrila, The Visual Analysis of Human Movement: A Survey, *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[genesereth97] Michael R. Genesereth and Steven P. Ketchpel, Software Agents, *Communications of the ACM*, 37(7):48–53, 1997.

[graf00] Thorsten Graf and Alois Knoll, A multi-agent system architecture for distributed coputer vision, *International Journal on Artificial Intelligence Tools*, 9(2):305–319, 2001.

[gueziec02] André Guéziec, Tracking Pitches for Broadcast Television, *Computer*, 35(3):38–43, March 2002.

[heikkila00] Janne Heikkilä, Geometric Camera Calibration Using Circular Control Points, *IEEE Transactions on Pattern recognition and Machine Intelligence*, 22(10):1066–1077, October 2000.

[hongeng01] Somboon Hongeng and Ramakant Nevatia, Multi-agent event recognition, in *Proceedings of the International Conference on Computer Vision*, 84–91, IEEE, Vancouver, Canada, 2001.

[hoover99] Adam Hoover and Bent David Olsen, A Real-Time Occupancy Map From Multiple Video Streams, in *Proceedings of the International Conference on Robotics and Automation*, 2261–2266, IEEE, Detroit, USA, May 1999.

[hoover00] Adam Hoover and Bent David Olsen, Sensor Network Perception for Mobile Robotics, in *Proceedings of the International Conference on Robotics and Automation*, vol. I, 342–348, IEEE, San Francisco, California, April 2000.

[howe:www] Dennis Howe, Free On-Line Dictionary Of Computing, computing terms.
URL `http://foldoc.doc.ic.ac.uk/foldoc/index.html`

[huhns97] Michael N. Huhns and Munindar P. Singh, The Agent Test, *IEEE Internet Computing*, 38(1):78–79, October 1997.

[huhns99a] Michael N. Huhns and Munindar P. Singh, A Multiagent Treatment of Agenthood, *Applied Artificial Intelligence*, 13(1–2):3–10, January 1999.

[huhns99] Michael N. Huhns and Larry M. Stephens, Multiagent systems and Societies of Agents, in Gerhard Weiss, ed., *Multiagent Systems*, 79–121, MIT Press, 1999.

[hunt99] Andrew Hunt and David Thomas, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley, 1999.

[javed03] Omar Javed, Zeeshan Rasheed et al., Tracking Across Multiple Cameras With Disjoint Views, in *Proceedings of the International Conference on Computer Vision*, 952–957, Nice, France, October 2003.

[kanade97] Takeo Kanade, Robert T. Collins et al., Cooperative Multisensor Video Surveillance, in *Proceedings of Image Understanding Workshop*, 3–10, New Orleans, USA, May 1997.

[karnik98] Neeran M. Karnik and Anand R. Tripathi, Design Issues in Mobile-Agent Programming Styles, *IEEE Concurrency*, 6(3):52–61, July 1998.

[karuppiah01] D.R. Karuppiah, Z. Zhu et al., A Fault-Tolerant Distributed Vision System Architecture for Object Tracking in a Smart Room, in *Proceedings of the International Workshop on Computer Vision Systems*, 201–219, Vancouver, Canada, July 2001.

[kay93] M.G. Kay and R.C. Luo, Global Vision for Intelligent AGVs, *SME Journal of Vision*, 9(2):1–4, 1993.

[khan01] S. Khan, O. Javed et al., Human Tracking in Multiple Cameras, in *Proceedings of the International Conference on Computer Vision*, I: 331–336, IEEE, Vancouver, Canada, July 2001.

[krumm00] John Krumm, Steve Harris et al., Multi-Camera Multi-Person Tracking for EasyLiving, in *Proceedings of Workshop on Visual Surveillance*, 3–10, Dublin, Ireland, July 2000.

[lakos96] John Lakos, *Large Scale C++ software Design*, Professional Computing Series, Addison-Wesley, Reading, Massachusetts, USA, 1996.

[latin04] Vedran Latin, Lokalizacija robota upotrebom referentnih objekata, 2004, diplomski rad, Fakultet elektrotehnike i računarstva, Zagreb.

[stein00] Lily Lee, Raquel Romano and Gideon Stein, Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame, *IEEE Transactions on Pattern recognition and Machine Intelligence*, 22(8):758–767, August 2000.

[lesser83] Victor R. Lesser, The Distributed Vehicle Monitoring Testbed, 1983, a Distributed Artificial Intelligence project developed at the University of Massachusetts. URL `http://dis.cs.umass.edu/research/dvmt/`

[liu99] Jiming Liu and Yuan Y. Tang, Adaptive Image Segmentation With Distributed Behavior-Based Agents, *IEEE Transactions on Pattern recognition and Machine Intelligence*, 21(6):544–551, June 1999.

[matsuyama98] Takashi Matsuyama, Cooperative Distributed Vision, in *Proceedings of Image Understanding Workshop*, 365–384, Monterey, USA, November 1998.

[matsuyama02] Takashi Matsuyama and Norimichi Ukita, Real-Time Multitarget Tracking by a Cooperative Distributed Vision System, *Proceedings of the IEEE*, 90(7):1136–1150, July 2002.

[menegatti01] Emanuele Menegatti and Enrico Pagello, Cooperative Distributed Vision for Mobile Robots, in *Proc. of AI*IA 2001 Workshop on Artificial Intelligence, Vision and Pattern Recognition*, September 2001.

[menegatti03] Emanuele Menegatti, Enrico Pagello et al., Toward knowledge propagation in an omnidirectional distributed vision system, in *Proceedings of 1st International Workshop on Advances in Service Robotics*, 1–6, March 2003.

[webster:www] Merriam-Webster, Online Dictionary of English Language, american english. URL `http://www.m-w.com`

[meyers97] Scott Meyers, *More Effective C++*, Professional Computing Series, Addison-Wesley, Reading, Massachusetts, USA, 1997.

[mittal03] Anurag Mittal and Larry S. Davis, M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene, *International Journal of Computer Vision*, 51(3):189–203, March 2003.

[mohr96] Roger Mohr and Bill Triggs, Projective Geometry for Image Analysis, July 1996, a tutorial given at the International Symposium of Photogrammetry and Remote Sensing, Vienna.
URL `http://www.inrialpes.fr/movi/people/Triggs/isprs96/isprs96.html`

[nakazawa98] A. Nakazawa, H. Kato and S. Inokuchi, Human Tracking Using Distributed Vision Systems, in *Proceedings of the International Conference on Pattern Recognition*, vol. I, 593–596, IEEE, Brisbane, Australia, August 1998.

[nwana96] Hyacinth S. Nwana, Lyndon Lee and Nick Jennings, Co-ordination in software agent systems, *BT Technology Journal*, 14(4):79–89, October 1996.

[olsen01] Bent David Olsen and Adam Hoover, Calibrating a camera network using a domino grid, *Pattern Recognition*, 34(5):1105–1117, May 2001.

[oswald01] Norbert Oswald and Paul Levi, Cooperative Object Recognition, *Pattern Recognition Letters*, 22(12):1273–1282, December 2001.

[pfleger98] Karl Pfleger and Barbara Hayes-Roth, An Introduction to Blackboard-Style Systems Organization, Tech. Rep. KSL-98-03, Stanford University, California, 1998.

[press93] W.H. Press, S.A. Teukolsky et al., *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 1993.

[qi03] Hairong Qi, YingyueXu and XIaoling Wang, Mobile-agent-based Collaborative Signal and Information Processing in Sensor Networks, *Proceedings of the IEEE*, 91(8):1172–1183, August 2003.

[remagnino98] Paolo Remagnino, Tieniu Tan and Keith Baker, Multi-agent visual surveillance of dynamic scenes, *Image and Vision Computing*, 16(8):529–532, June 1998.

[segvic00] Siniša Šegvić, *Uporaba projekcijske geometrije i aktivnog vida u tumačenju scena*, Master's thesis, Fakultet elektrotehnike i računarstva, Zagreb, Hrvatska, May 2000.

[shoham93] Yoav Shoham, Agent-oriented programming, *Artificial Intelligence*, 60(1):51–92, 1993.

[snajder04] Jan Šnajder, Višeagentska komunikacija i kooperacija, Tech. Rep., ZEMRIS, Fakultet elektrotehnike i računarstva, Zagreb, 2004.

[sogo99] Takushi Sogo, Hiroshi Ishiguro and Toru Ishida, Mobile Robot Navigation by Distributed Vision Agents, *Lecture Notes on Computer Science*, 1733:96–111, January 1999.

[spies03] Hagen Spies, Geometric Aspects of Computer Vision, 2003, phD course.
URL `http://www.isy.liu.se/~hspies/course/course.html`

[cxx98] International Standard, *Programming Languages – C++, ISO/IEC 14882*, International Standardization Organization (ISO), Geneve, Switzerland, 1998.

[steiner98] Donald Steiner, Industrial Applications of Multi-Agent Technology, in *Proceedings of the International Conference on Multi-Agent Systems*, 12–13, Paris, France, 1998.

[stillman99] Scott Stillman, Rawesak Tanawongsuwan and Irfan Essa, A system for Tracking and Recognizing Multiple People with Multiple Cameras, in *Proceedings of International Conference on Audio- and Video-Based Person Authentication*, 96–101, Washington, USA, April 1999.

[tamaki02] Toru Tamaki, Tsuyoshi Yamamura and Noboru Ohnishi, Unified Approach to Image Distortion, in *Proceedings of the International Conference on Pattern Recognition*, vol. 2, 584–589, Quebec, Canada, August 2002.

[tan89] C.L. Tan and W.N. Martin, An analysis of a distributed multiresolution vision system, *Pattern Recognition*, 22(3):257–265, 1989.

[tsai87] Roger Y. Tsai, A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf Cameras and Lenses, *IEEE Journal of Robotics and Automation*, 3(4):323–345, August 1987.

[tsutsui01] Hideki Tsutsui, Jun Miura and Yoshiaki Shirai, Optical Flow-Based Person Tracking by Multiple Cameras, in *Proceedings of International Conference of Multisensor Fusion and Integration*, 91–96, Baden Baden, Germany, August 2001.

[matsuyama00] Norimichi Ukita and Takashi Matsuyama, Incremental Observable-Area Modeling for Cooperative Tracking, in *Proceedings of the International Conference on Pattern Recognition*, 4192–4196, Barcelona, España, September 2000.

[veloso98] Manuela Veloso, Peter Stone et al., The CMUnited-97 Small Robot Team, in Hiroaki Kitano, ed., *RoboCup-97: Robot Soccer World Cup I*, 242–256, Springer Verlag, Berlin, 1998.

[wegner95] Peter Wegner, Interactive foundations of object-based programming, *Computer*, 28(10):70–72, 1995.

[wegner97] Peter Wegner, Why interaction is more powerful than algorithms, *Communications of the ACM*, 40(5):80–91, 1997.

[weisstein:www] Eric Weisstein, Eric Weisstein's World of Mathematics, a Wolfram Web Resource.
URL `http://mathworld.wolfram.com/`

[wooldridge99] Michael Wooldridge, Intelligent Agents, in Gerhard Weiss, ed., *Multiagent Systems*, 27–79, MIT Press, 1999.

[yang03] Danny B. Yang, Héctor H. Gonzaléz-Baños and Leonidas J. Guibas, Counting People in Crowds with a Real-Time Network of Image Sensors, in *Proceedings of the International Conference on Computer Vision*, 122–129, Nice, France, October 2003.

[yoshida01] Norihiko Yoshida, Multi-Target Tracking by Cooperation of Stationary and Mobile Agents, in *Proceedings of International Workshop on Cooperative Distributed Vision*, 183–203, Kyoto, Japan, March 2001.

[zhang00] Zhengyou Zhang, A flexible New Technique for Camera Calibration, *IEEE Transactions on Pattern recognition and Machine Intelligence*, 22(11):1330–1334, November 2000.

[zhao03] Feng Zhao, Jie Liu et al., Collaborative Signal and Information Processing: An Information Based Approach, *Proceedings of the IEEE*, 91(8):1199–1209, August 2003.

Sažetak

U višeagentskom pristupu projektiranju programske podrške, funkcionalnost se postiže interakcijom autonomnih procesnih entiteta ili agenata. Pristup je posebno prikladan kod problema koji su ili inherentno porazdijeljeni, pa se autonomni čimbenici ističu već u specifikaciji, ili iznimno složeni te ih je pogodno izraziti kroz što autonomnije specijalizirane podsustave. Komunikacija u višeagentskom sustavu obično se modelira prema poznatim obrascima interakcije među ljudima ili životinjama, pa agenti često imaju antropomorfna svojstva poput vjerovanja, želja ili namjera. Analiza praktičnih aspekata višeagentskog organiziranja na području računarskog vida podrazumijeva eksperimentiranje s vrlo složenim programskim sustavima. To djelomično objašnjava slabiju zastupljenost tog područja u znanstvenim publikacijama odnosno sporiji prodor višeagentskih pristupa na području industrijskih primjena.

Razmatra se višeagentski sustav iz domene porazdijeljenog računarskog vida, u kojem su agenti organizirani u višerazinskoj hijerarhijskoj strukturi. Cilj sustava je praćenje pokretnih objekata u stvarnom vremenu, temeljeno na kooperaciji skupa upravljivih kamera. Svakoj kameri pridruženo je zasebno umreženo računalo kojim upravlja agent promatrač, tj. program koji analizira pribavljene slike i upravlja smjerom gledanja kamere. Promatrači podešavaju smjer gledanja u smislu ostvarivanja čim boljih rezultata praćenja u skladu sa svojim odgovornostima koje se mogu odnositi ili na pojedine objekte ili na određene dijelove scene. Koordinaciju promatrača obavljaju tzv. agenti koordinatori, koji udružuju pojedinačne poglede u ukupni prikaz scene, te formuliraju globalne ciljeve porazdijeljenog sustava u obliku odgovornosti pojedinih promatrača. Korekcije odgovornosti promatrača obično obezvrijeđuju njihove dotadašnje kontekste praćenja, pa kvaliteta dojavljivanih podataka privremeno pada dok nesigurnost informacija o sceni raste. Učestalost korekcija stoga mora biti pažljivo odmjerena, što upućuje na to da je robustna autonomna prilagodba promatrača stanju scene iznimno važna za ispravan rad sustava.

Radi postizanja veće fleksibilnosti, predloženo je proširenje arhitekture s promatračima koji imaju mogućnost kretanja. Zbog potrebe za njihovom povremenom apsolutnom lokalizacijom, pogodno je postaviti dodatni zahtjev da promatrači, pored detekcije praćenih objekata, budu sposobni i za detekciju pokretnih promatrača. Tako bi se moglo reći da pokretni promatrači imaju dvostruku ulogu, i promatrača i pokretnog objekta. Konačno, predviđeno je i višerazinsko proširenje temeljne arhitekture u kojem podređeni koordinatori imaju ulogu promatrača prema koordinatoru na višoj razini. U takvoj organizaciji, podređeni koordinatori ispunjavaju svoje odgovornosti raspodjelom poslova pridruženim agentima.

Sustav za porazdijeljeno praćenje je razmatran u kontekstu lokalizacije jednostavnih pokretnih robota metodom globalnog vida. U takvoj organizaciji, upravljanje pojedinim robotima temelji se na zajedničkoj infrastrukturi koja pruža informacije o položaju svih robota u zadanom području, kao mrežni servis dostupan svim zainteresiranim stranama. Klijenti predložene infrastrukture bi bili nezavisni upravljači pojedinih robota, koji bi tada mogli stvarati autonomne planove akcija robota. Pokazuje se da je takvo sučelje moguće elegantno uklopiti u postojeću arhitekturu, u okviru specijaliziranog koordinatora na vrhu hijerarhije.

Ključne riječi: višeagentski sustavi, porazdijeljena umjetna inteligencija, trodimenzionalni i aktivni računarski vid, praćenje objekata, lokalizacija globalnim vidom.

Summary

In the multi-agent approach to software design, the functionality of a complex system is expressed through interaction between autonomous processing entities called agents. The approach is particularly suited for problems which are either inherently distributed, where autonomous subjects emerge directly from the specification, or extremely complex, so that it is very beneficial to express the solution through decoupled autonomous specialized subsystems. Communication in a multi-agent system is usually modelled after known interaction patterns between humans or animals, so that agents often feature anthropomorphic qualities such as beliefs, desires or intentions. Analysis of practical aspects of multi-agent architectures in the field of computer vision implies experiments with large scale software designs. That explains, in part, a rather weak attention in scientific publications and a relatively slow progress towards industrial applications.

This work considers a multi-agent system in the domain of distributed computer vision, in which the agents are organized within a multilevel hierarchical structure. The objective of the system is to track multiple moving objects in real time using cooperation of several controllable cameras. Each camera is assigned a separate network connected computer which is controlled by an observer agent, i.e. a program which processes acquired images and controls the viewing direction of the camera. Observers adapt their viewing directions in order to achieve optimal tracking results according to their responsibilities, which are related either to individual objects or to particular parts of the scene. The cooperation of observers is managed by the second kind of agents in the system, which are called coordinators. The tasks of a coordinator consist of integrating individual views into the comprehensive representation of the scene, and formulating global objectives of the distributed system into responsibilities of individual observers. Corrections of observer responsibilities often invalidate their tracking contexts which is reflected by a decreasing quality of provided data and an increasing uncertainty of the scene representation. The frequency of corrections therefore must be carefully regulated, which suggests that a robust autonomous adaptation of the observers to the changes in the scene is extremely beneficial for the correct functioning of the system.

In order to achieve a greater flexibility, an extension of the basic architecture is proposed, which deals with observers being capable of autonomous movement. Following the need for their occasional absolute localization, it is suitable to introduce an additional requirement, that observers should be able to detect their moving peers as well. Consequently, it can be stated that, in such a system, moving observers have a dual role: as an observer and as a tracked object. Finally, a multilevel extension of the basic architecture is also introduced, in which each inferior coordinator acts as an observer to the assigned superior coordinator. In such an arrangement, inferior coordinators fulfill their observer responsibilities by delegating jobs to assigned agents.

The system for distributed visual tracking is considered in the context of localizing a group of simple moving robots by a global vision approach. The approach implies building a common infrastructure capable of providing positions of all robots in the area, as a network service available to general public. The infrastructure could be used by independent robot controllers, which would then be able to make autonomous plans of robot actions. It is shown that such network interface can be fitted into proposed architecture, within a specialized coordinator on the top of the hierarchy.

Keywords: Multi-agent systems, distributed artificial intelligence, three-dimensional and active computer vision, distributed visual tracking, localization by global vision.

Title: Multi-agent Object Tracking by Active Vision

Životopis

Rođen sam 1971. godine u Splitu, a matematičku gimnaziju sam završio u Zadru 1990. godine. Iste godine upisao sam studij na Elektrotehničkom fakultetu Sveučilišta u Zagrebu i diplomirao u veljači 1995. godine. Za izuzetan uspjeh, i na trećoj i na četvrtoj godini studija primio sam Fakultetsko priznanje "Josip Lončar". Od 1996. godine do danas, zaposlen sam na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva u Zagrebu.

Sudjelovao sam u radu više znanstvenih projekata te bio voditelj jednog projekta primjene informacijske tehnologije Ministarstva znanosti i tehnologije. U svibnju 2000. godine obranio sam magistarski rad pod naslovom "Uporaba projekcijske geometrije i aktivnog vida u tumačenju scena". Područja mog znanstvenog, istraživačkog i profesionalnog interesa uključuju aktivni, trodimenzionalni i distribuirani računarski vid, višeagentske sustave, obradu slike, programsko inženjerstvo te objektno i generičko programiranje. Samostalno odnosno kao koautor objavio sam više članaka u časopisima sa međunarodnom recenzijom te na međunarodnim znanstvenim skupovima.

Tijekom rada na Zavodu, pomagao sam u izvođenju vježbi iz kolegija Arhitektura i organizacija računala, Digitalna elektronika, Inteligentni sustavi te Projektiranje digitalnih sustava, u svojstvu mlađeg asistenta odnosno asistenta. Također, sam sudjelovao u izvođenju vježbi iz kolegija Arhitektura računalnih sustava na Elektrotehničkom fakultetu u Osijeku, u akademskim godinama 1998/99 i 1999/2000.

Član sam strukovne udruge IEEE, i aktivno se služim engleskim i talijanskim jezikom.