Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Alan Lukežič

Avtonomno vodenje kvadrokopterja z računalniškim vidom

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Kristan

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00024/2012
Datum: 10.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALAN LUKEŽIČ**

Naslov: **AVTONOMNO VODENJE KVADROKOPTERJA Z RAČUNALNIŠKIM VIDOM**

COMPUTER-VISION-BASED AUTONOMOUS CONTROL FOR QUADROCOPTER

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V diplomski nalogi obdelajte problem avtonomnega vodenja kvadrokopterja z računalniškim vidom. Osredotočite se na nalogo zasledovanja objekta po prostoru z uporabo barvne kamere, ki je nameščena na kvadrokopterju. Kvadrokopter mora biti sposoben zasledovati splošne objekte, torej objekte, ki niso označeni z vnaprej določenimi markerji. Izberite in implementirajte primerne sledilnike za sledenje objektov z monokularno kamero in jih po potrebi priredite za delovanje na kvadrokopterju. Izdelajte tudi sistem avtonomnega vodenja kvadrokopterja, ki bo zasledoval objekt na podlagi implementiranega algoritma za sledenje. Sistem implementirajte na realni platformi in analizirajte njegovo delovanje.

Mentor:

doc. dr. Matej Kristan

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Alan Lukežič, z vpisno številko **63090089**, sem avtor diplomskega dela z naslovom:

Avtonomno vodenje kvadrokopterja z računalniškim vidom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mateja Kristana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 11. januarja 2011 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Mateju Kristanu za strokovno pomoč in napotke pri izdelavi diplomske naloge. Zahvaljujem se tudi as. mag. Luki Čehovinu za tehnično pomoč pri izdelavi aplikacije.

Posebna zahvala gre mojim staršem, ki so me tekom študija podpirali tako moralno kot tudi finančno. Hvala tudi Nini za strpnost in moralno pomoč tekom študija.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Sorodna dela 1

1.2 Cilj diplomske naloge 5

1.3 Zgradba diplomske naloge 6

2 Opis mobilne platforme Parrot AR.Drone **7**

2.1 Kvadrokopter AR.Drone 7

2.2 Uradna programska podpora 12

3 Metode za sledenje objektom **17**

3.1 Sledilnik CamShift 17

3.2 Naprednejši sledilnik LGT 19

4 Izvedba vodenja s kamero **25**

4.1 Model kamere 26

4.2 Kalibracija kamere 27

4.3 Sprotno ocenjevanje pozicije objekta 30

5 Rezultati **37**

5.1 Senzorji 37

5.2 Kalibracija kamere 41

5.3 Sledenje pri idealnih pogojih 43
5.4 Sledenje realnemu objektu 46

6 Sklep **51**
6.1 Možnosti za izboljšave 52

Povzetek

Cilj diplomske naloge je implementacija zasledovanja objektom z mobilno platformo Parrot AR.Drone. Parrot je kvadrokopter, zračno plovilo, ki je podobno helikopterju, le da ima štiri propelerje. Ker je na njem nameščena kamera in se je nanj mogoče brezžično povezati s prenosnikom, sodi sistem na področje računalniškega vida in mobilne robotike. V nalogi smo izdelali sistem, ki je sposoben samostojno slediti objektu, ki ga ročno označimo. Za sledenje objektom smo uporabili dva obstoječa sledilnika in analizirali njuno primernost v realnem sistemu. Sledilnika smo integrirali z mobilno platformo in implementirali ustrezne metode za navigacijo kvadrokopterja v prostoru. Naš postopek na podlagi lokacije objekta v sliki in notranjih parametrov kamere oceni položaj objekta v prostoru. S pomočjo te informacije predlagana metoda let kvadrokopterja regulira tako, da ohranja konstanten položaj glede na objekt. Delovanje sistema smo ovrednotili empirično na realnih primerih zasledovanja objektov in z meritvami v kontroliranih pogojih. Na koncu predlagamo še možnosti za nadaljne izboljšave, ki bi naredile sistem robustnejši.

Ključne besede: vizualno sledenje objektom, kvadrokopter, Parrot AR.Drone, računalniški vid, robotika

Abstract

The main goal of the thesis is presenting the implementation of the mobile platform Parrot AR.Drone for object tracking. Parrot is a quadrocopter – an aerial vehicle similar to a helicopter, but with four propellers. As there is a camera attached to it and there is the possibility of wireless connection via laptop, the system belongs to the computer vision and robotics field. We created a system which is capable of autonomous tracking a manually selected object. For tracking we used two existent trackers and analyzed their suitability in real system. We integrated the trackers into the mobile platform and implemented suitable methods for navigating the quadrocopter in space. Using only the object's location in the image, the quadrocopter is able to continually estimate it's position in space. Using this piece of information our method regulate the flight route of the quadrocopter, so that it keeps the constant position to the object. We assessed the system with the help of empiric experiments on real examples of object tracking and measurements in controlled conditions. At the end we suggested improvements which should make the system more robust.

Key words: visual object tracking, quadrocopter, Parrot AR.Drone, computer vision, robotics

Poglavje 1

Uvod

V zadnjem času so na področju mobilne robotike postali zelo popularni tako imenovani kvadrokopterji. To so zračna plovila podobna helikopterjem, le da imajo štiri propelerje nameščene vodoravno na ogrodje. Taka plovila imajo lahko nameščene tudi različne senzorje (npr.: GPS sprejemnik, senzor višine, nagiba, IR oddajnike) ter druge naprave (npr. kamero, laser za pridobitev informacije o prostoru, dostopno točko WiFi). Tako opremljeni kvadrokopterji se lahko uporabljajo tudi v praksi, saj bi ga lahko uporabljali za varovanje objektov namesto fiksnih kamer, za potrebe v vojski, za snemanje športnih prireditev na mestih, ki so za kamere in kamermane neprimerna ali v zabavni industriji. Posebej zanimiva aplikacija je sledenje gibajočim se objektom po prostoru. V tem poglavju bomo podrobneje opisali take vrste aplikacij in njihove značilnosti.

1.1 Sorodna dela

Sledenje objektom izhaja iz področja računalniškega vida, natančneje iz področja analize vsebine videa. Na tem področju je bilo v preteklosti izvedenih že veliko raziskav in objavljene literature [39, 40, 41, 42, 25]. Sledenje je uporabljeno tudi v številnih aplikacijah na področju video nadzorovanja, video editiranja, sledenje objektov v športu, medicini, igrah in spoznavnih siste-

mih. Algoritmi za sledenje objektom v videu podajo informacijo o lokaciji objekta v sliki za vsako sliko v sekvenci. Običajno je ta informacija položaj središča objekta v sliki podana v slikovnih elementih in velikost regije, ki jo zavzema objekt na sliki. Algoritmi za sledenje objektu v sekvenci slik uporabljajo različne značilnice. Najpogosteje so to barvni histogrami in robovi. S podobnostjo med histogrami se išče podobnost med regijami v sliki. Robove se določi s pomočjo detektorjev (npr. Canny-jev detektor robov [9]).

Na področju kvadrokopterjev je bilo izvedenih precej raziskav. Kvadrokopter je posebna vrsta helikopterja s štirimi propelerji, kot primer na Sliki 1.1. Prvega je konstruiral George de Bothezat leta 1922 za potrebe ameriške vojske. V članku [2] so opisani osnovni principi delovanja in krmiljenja kvadrokopterja.

Slika 1.1: Model kvadrokopterja. Slika je povzeta po [43].

Delovanje helikopterja s štirimi propelerji, dinamični model in stabilizacijo kvadrokopterja v zraku opisujeta [3, 4]. Sistem, kjer je vključena še

kamera in omogoča samostojno gibanje v prostoru, je opisan v [1]. Sistem mora preko kamere dobiti informacijo o položaju v prostoru in hitrosti s katero se premika. Kamera mu služi kot osnovni senzor in ni pritrjena na kvadrokopter, ampak se nahaja na fiksnem mestu v prostoru. Na podlagi informacije o položaju kvadrokopterja lahko sistem samostojno navigira plovilo po prostoru.

V povezavi z računalniškim vidom je posebej zanimivo samostojno navigiranje po prostoru s kvadrokopterji. Pri navigaciji običajno pomaga tudi informacija iz GPS senzorjev o položaju, v notranjosti stavb pa se gradi mapa s pomočjo algoritmov za hkratno lokalizacijo in izgradnjo map (angl. Simultaneous Localization and Mapping, SLAM [7]). V [5] je opisano grajenje mape prostora s kvadrokopterjem, ki je opremljen s kamero, stereo sistemom kamer in laserjem. Sistem dobi preko laserja oblak točk in s pomočjo algoritmov SLAM zgradi mapo prostora. Podoben problem je opisan tudi v [6], kjer so namesto laserja uporabili Kinect [27, 28] ter stereo sistem kamer. Za navigacijo so poleg barvne slike za vsak slikovni element dobili tudi informacijo o oddaljenosti v prostoru. Sistemi za gradnjo mape prostora so sicer dokaj robustni, vendar je potrebna draga oprema, časovna kompleksnost pa tudi ni nezanemarljiva. Na Fakulteti za elektrotehniko v Ljubljani je bil razvit sistem, ki uporablja slikovno ter informacijo s senzorjev kvadrokopterja za oceno položaja v prostoru in stabilizacijo v zraku [8]. Sistem deluje ob pomoči markerja, ki je postavljen na fiksnem mestu v prostoru. Iz slike markerja lahko oceni položaj in orientacijo v prostoru. Kadar ga ni na voljo (npr. izpade iz vidnega polja kamere), uporabi informacijo s senzorjev - žiroskopov, merilcev pospeška in višine. Tak sistem je nerealen, saj je omejen na vnaprej določen marker.

Zanimive teme na področju kvadrokopterjev raziskujejo tudi v laboratoriju GRASP [32] v Pennsylvaniji. Njihovi najzanimivejši projekti so samostojno učenje kvadrokopterja izvajanja figur v zraku [37], žongliranje s teniško žogico in prenašanje raznih predmetov [33]. Omeniti velja tudi sodelovanje kvadrokopterjev v skupini, kot je na primer prenašanje večjih predmetov

z več kvadrokopteji [35] ali pa grajenje objektov s posebnih opek [34], za katere menijo, da bi lahko mogoče v prihodnosti celo zamenjali gradbene dalavce. Zanimivo je tudi obnašanje več kvadrokopterjev v skupini, ki deluje kot celota [36], izvajanje raznih figur in skupinsko načrtovanje poti. Problemi, ki si jih zastavljajo v laboratoriju GRASP, so omejeni na določene pogoje. Kamera ni nameščena na kvadrokopterju, ampak v prostoru. Kamere, ki jih uporabljajo, niso navadne, ampak visokofrekvenčne infrardeče, ki omogočajo hitro osveževanje in posledično večjo natančnost. Na kvadrokopterjih so nameščeni odsevniki, ki odbivajo infrardečo svetlobo, ki jo seva kamera. Ker je v prostoru več kamer in njihov položaj je v naprej znan, je iz več slik mogoče določiti položaj objekta v prostoru. Tak sistem je veliko natančnejši, je pa tudi manj splošen, omejen na zelo specifične pogoje in zahteva natančno kalibracijo.

1.2 Cilj diplomske naloge

Cilj diplomske naloge je razviti metodo, ki bi omogočala avtonomno zasledovanje objekta v prostoru s kvadrokopterjem. Problem postane še posebej zanimiv, ker je kamera pritrjena na kvadrokopterju in se skupaj z njim premika. To pa vnaša v model dodatno dimenzijo in več priložnosti, da podatki postanejo šumni. Številne metode za vizualno sledenje so že razvite, zato smo se posvetili navigaciji kvadrokopterja in poizkušali informacijo o objektu, ki mu sledimo, čim bolje uporabiti. Končni izdelek diplomske naloge je sistem, ki je sposoben zasledovati objekt na podlagi vizualne informacije. Z mobilno napravo (npr. prenosnikom) se povežemo na kvadrokopter in iz njega dobimo video s kamere. Označimo objekt, kateremu želimo, da sledi in na koncu varno pristane na tleh. Sistem smo realizirali na realni platformi AR.Drone [21]. Shema našega sistema je prikazana na Sliki 1.2. Njegova prednosti je splošna uporabnost, saj ni omejen na drage naprave ali strogo kontrolirane pogoje. Kvadrokopter je potrebno povezati s prenosnikom, sledenje je potem možno v splošnem okolju.

Slika 1.2: Shema sistema.

1.3 Zgradba diplomske naloge

Preostanek naloge je razdeljen v šest poglavij. V Poglavju 2 je predstavljena mobilna platforma Parrot AR.Drone. Opisani so vsi sestavni deli, ki se nahajajo v kompletu in njihove specifikacije. Opisali smo tudi uradno programsko opremo, ki skrbi za povezavo in komunikacijo kvadrokopterja z računalnikom.

V Poglavju 3 smo opisali metodi za sledenje objektom, ki smo jih uporabili pri integraciji. Najprej smo se lotili z enostavnejšim in tudi manj učinkovitim sledilnikom, CamShift [18]. Ta sledilnik nam je služil v zgodnji fazi razvoja našega sistema. Opisali smo tudi robustnejši sledilnik LGT [26], ki je bil razvit na Fakulteti za računalništvo in informatiko v Laboratoriju za umetne vizualne spoznavne sisteme, s katerim smo izvedli sledenje kompleksnejšemu objektu v prostoru.

Poglavje 4 opisuje implementacijo sledenja, ustrezne odzive kvadrokopterja in probleme, ki so se pri tem pojavili. Najprej opišemo model kamere in kalibracijo kamere. Vodenje kvadrokopterja smo razdelili v dva dela in ju opisali v podpoglavjih. Prvi del je ocena odklona, drugi pa premik vzdolž osi x.

Poglavje 5 je namenjeno ovrednotenju sistema. Osredotočili smo se na posamezne faze, ki se zgodijo od vzleta kvadrokopterja pa do zaključka sledenja in pristanka na tleh. Z meritvami smo podali natančnost senzorjev ter vrednost notranjih parametrov kamere, ki smo jih pridobili s postopkom kalibracije kamere. Delovanje in robustnost celotnega sistema smo preizkusili na primeru sledenja osebi v večjem prostoru. V Poglavju 6 smo strnili naše ugotovitve v sklep in predstavili nekaj možnosti za izboljšave.

Poglavje 2

Opis mobilne platforme Parrot AR.Drone

V tem poglavju je podrobneje predstavljena mobilna platforma Parrot AR.Drone. Opisana je vsebina kompleta z vsemi sestavnimi deli ter njihove specifikacije. Drugi del je namenjen programski opremi, s katero se povežemo na kvadrokopter in z njim komuniciramo. Na kvadrokopter se lahko povežemo z različnimi mobilnimi napravami (npr. telefonom ali prenosnikom). Opisana je tudi vrsta povezave in način komunikacije mobilne naprave s kvadrokopterjem.

2.1 Kvadrokopter AR.Drone

Kvadrokopter izdeluje francosko podjetje Parrot, ki se ukvarja predvsem z izdelavo brezžičnih naprav (npr. slušalk) za mobilne telefone. Kvadrokopter Parrot AR.Drone je bil predstavljen na sejmu zabavne elektronike (CES) v Las Vegasu leta 2010. Na trgu je sicer na voljo že različica 2.0, ki je bila predstavljena leta 2012. V diplomski nalogi smo uporabili prvo. Namenjen je letenju, nadzorujemo ga pa tako, da se povežemo nanj preko brezžične dostopne točke, ki se nahaja na samem kvadrokopterju, z mobilno napravo. Na voljo so uradne aplikacije za vodenje naprave za operacijska sistema iOS

(Apple) in Android (Google) ter neuradne za operacijska sistema Samsung BADA in Symbian. Kvadrokopter je prikazan na Sliki 2.1, njegova dolžina je približno 30 centimetrov, masa pa 420 gramov. Ogrodje in štirje propelerji so narejeni iz plastike. Zaščita za letenje v prostoru je iz stiropora, ogrodje za propelerje v obliki črke x pa iz karbonskih vlaken.

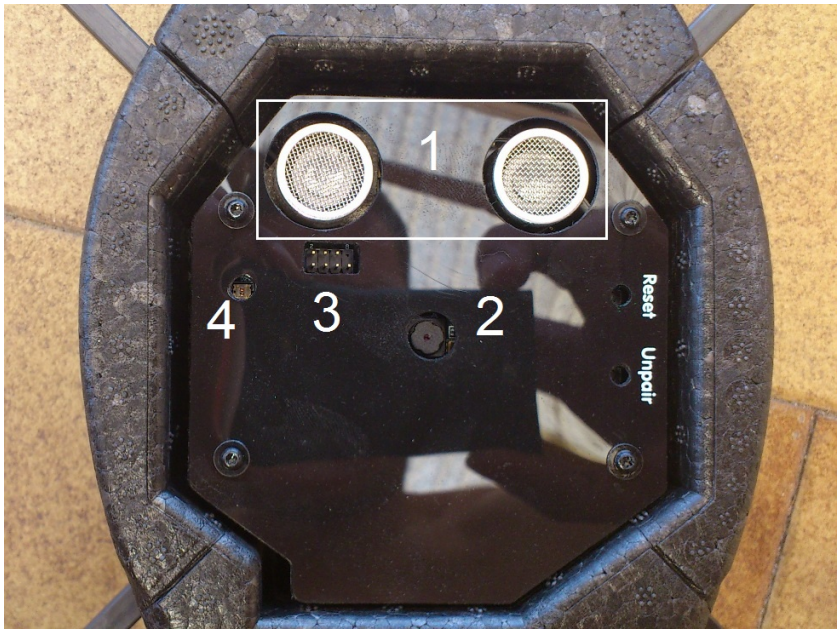
Slika 2.1: Parrot AR.Drone z zaščito.

Za procesiranje informacij, ki jih prejema in pošilja po brezžičnem omrežju WiFi, skrbi mikrokontroler ARM9 s 468 MHz ter 128 MB pomnilnika RAM, na katerem je nameščena programska oprema (angl. firmware) na osnovi operacijskega sistema Linux. Letenje omogočajo štirje propelerji, ki so pritrjeni na električne motorje z močjo 15W. Prikazani so na Sliki 2.2 in označeni s številko 2. Električno energijo zagotavlja 1000 mAh litij-ionska polimerska baterija z napetostjo 11.1 V, kar zadostuje za 12 – 15 minut letenja. Na Sliki 2.2 je s številko 3 označen priklop za baterijo.

Slika 2.2: Kvadrokopter brez zaščite s sprednje strani. S številkami so prikazani: 1 - sprednja kamera, 2 - propeler, 3 - priklop za baterijo.

Najvišja hitrost, ki jo kvadrokopter lahko doseže, je 5 metrov na sekundo (18km/h). Kvadrokopter potrebuje za optimalno letenje tudi informacijo o trenutni orientaciji v prostoru. Za to skrbijo senzorji za rotacije (žiroskopi) okoli vseh treh osi, merilnik pospeška okoli vseh treh osi ter ultrazvočni senzor višine z dometom 6 metrov (na Sliki 2.3 označen s številko 1). Senzor deluje tako, da ena enota pošilja ultrazvočne signale, ki se odbijajo od okolice, druga enota pa jih sprejema in na podlagi časa, ki se porabi za potovanje signalov, izračuna oddaljenost ovire oziroma višino. V kvadrokopter sta vgrajeni tudi dve kameri. Prva je nameščena spredaj in je usmerjena naprej, druga pa na spodnjem delu in usmerjena navzdol. Prednja kamera zajema slike v ločljivost 640x480 pikslov, njen vidni kot pa je 93°. Prednja kamera je prikazana na Sliki 2.2 in označena s številko 1.

Spodnja kamera zajema slike s hitrostjo 60 slik na sekundo, vidni kot pa znaša 64°, prikazana pa je na Sliki 2.3 in označena s številko 2. Na spodnji strani kvadrokopterja je nameščen priklop USB, ki nam omogoča povezavo z računalnikom, prikazan je na Sliki 2.3 s številko 3. Poleg priklopa se nahaja LED dioda, ki v stanju pripravljenosti sveti zeleno, ob morebitni napaki pa rdeče. Na Sliki 2.3 je označena s številko 4.



Slika 2.3: Spodnja stran kvadrokopterja Parrot AR.Drone. S številkami so prikazani: 1 - ultrazvočni senzor višine, 2 - spodnja kamere, 3 - priklop USB, 4 - LED dioda.

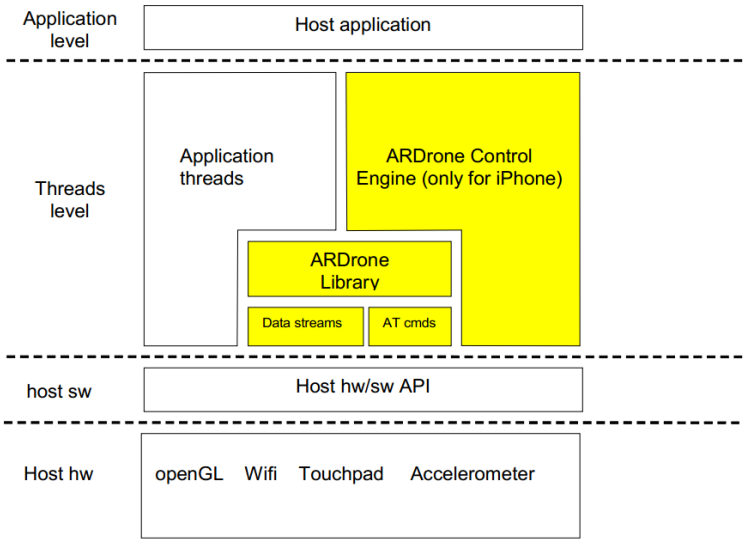
Tabela 2.1 prikazuje podrobnejše tehnične specifikacije modela Parrot AR.Drone. V Poglavju 5 so prikazane tudi meritve posameznih senzorjev in njihova natančnost.

Tabela 2.1: Tehnične podrobnosti kvadrokopterja AR.Drone.

Dimenzije	
z zaščito	52,5 x 51,5 cm
brez zaščite	45 x 29 cm
teža (z zaščito)	420 g
teža (brez zaščite)	380 g
hitrost	5 m/s; 18 km/h
Baterija	
moč	1000 mAh
napetost	11,1 V
moč motorjev	15 W
število vrtljajev	35000 rpm
čas letenja	12 - 15 min
čas polnjenja	90 min
Računalniški sistem	
procesor	ARM9 468 MHz
pomnilnik RAM	DDR 128 MB 200 MHz
povezljivost	WiFi b/g
operacijski sistem	Linux
Senzorji	
3 - osni merilnik pospeška 2 - osni žiroskop 1 - osni žiroskop (yaw)	
ultrazvočni višinometer	
maximalna višina	6 m
frekvenca	40 kHz
Sprednja kamera	
ločljivost	640 x 480 pikslov
razdalja detektiranja	0,3 - 5m
vidni kot	93°
Spodnja kamera	
ločljivost	176 x 144 pikslov
vidni kot	64°
hitrost zajemanja	60 fps

2.2 Uradna programska podpora

Podjetje Parrot skrbi tudi za razvoj programske opreme za kvadrokopter AR.Drone in podporo razvijalcem na tem področju. Na spletni strani [22] je mogoče prenesti uradno verzijo programskega paketa ARDrone SDK (angl. Software Development Kit), ki nam omogoča, da se na kvadrokopter povežemo tudi s prenosnim računalnikom in nam nudi ogrodje za ustvarjanje lastnih aplikacij. Shemo zgradbe programske opreme vidimo na Sliki 2.4. Prva plast na sliki predstavlja uporabnikovo apliakcijo, druga pa programe, ki so del programskega paketa SDK. Tretja plast ponazarja programsko opremo, ki je nameščena na kvadrokopterju. Spodnja plast predstavlja strojno opremo na AR.Drone-u in tisto, s katero upravljamo kvadrokopterju.



Slika 2.4: Shema programskega paketa SDK. Slika je povzeta po [24]

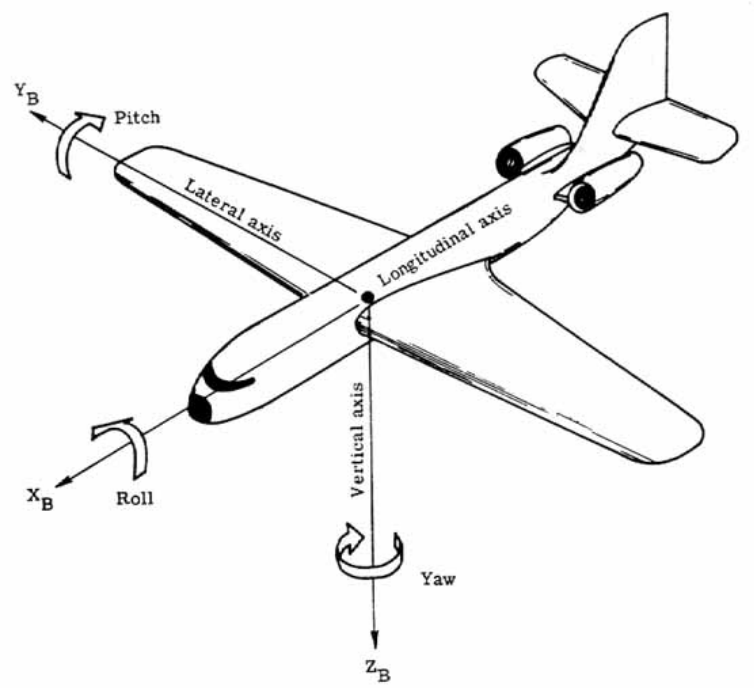
Pri izdelavi aplikacije s pomočjo ogrodja nam je olajšan dostop do vseh potrebnih podatkov, ki nam jih posreduje kvdrokopter. Tako je že implementiran sprejem slike v določeni programski strukturi, ki jo lahko poljubno obdelujemo. Dostopamo lahko tudi do podatkov o vseh treh rotacijah, višini na kateri se AR.Drone nahaja ter hitrosti, s katero se premika.

Na kvadrokopter se lahko povežemo in ga upravljamo z vsako napravo, ki omogoča WiFi način povezovanja, saj komunikacija poteka na način klient-strežnik. Pri povezovanju mobilne naprave na kvadrokopter AR.Drone se zgodijo naslednji koraki:

1. Kvadrokopter ustvari WiFi omrežje in rezervira prost IP naslov.
2. Uporabnik se poveže kot klient na ustvarjeno omrežje.
3. Klient zahteva IP naslov od kvadrokopterjevega strežnika preko protokola DHCP [31].
4. Kvadrokopterjev strežnik priskrbi klientu IP naslov, ki je:
   - Za ena večji od IP naslova strežnika (za programsko opremo na kvadrokopterju pred verzijo 1.1.3).
   - Večji od naslova strežnika za število med 1 in 4 (za programsko opremo na kvadrokopterju vključno z verzijo 1.1.3 in novejšo).
5. Klient lahko začne pošiljati pošiljati ukaze strežniku na kvadrokopterju preko dobljenega IP naslova in uporabljati njegova vrata (angl. ports).

Komunikacija klienta s kvadrokopterjem poteka na več načinov. Krmiljenje kvadrokopterja je realizirano tako, da klient pošilja strežniku ukaze preko protokola UDP na vratih 5556. Ukazi se pošiljajo 30 krat na sekundo. S tem je zagotovljeno delovanje brez časovnih zamikov. Primer ukaza iz izvorne kode, ki smo ga uporabljali za pošiljanje informacije kam in za koliko naj se kvadrokopter premakne:
ardrone_at_set_progress_cmd(int flag, float roll, float pitch, float gaz, float yaw);
Ukaz dobi kot argumente števila s plavajočo vejico med 0 in 1. To število predstavlja za kolikšen del največjega možnega premika, kar je del specifikacij, se bo kvadrokopter premaknil. Argumenti *roll, pitch, yaw* spreminjajo položaj kvadrokopterja tako kot kaže Slika 2.5, *gaz* spreminja višino, kjer se nahaja, *flag* pa postavi kvadrokopter v posebno stanje. V nadaljevanju

naloge bomo za rotacije *roll*, *pitch* in *yaw* uporabljali slovenske izraze, ki so kotaljenje, nagib in odklon.



Slika 2.5: Prikaz treh osi okrog katerih lahko rotiramo kvadrokopter. Rotacije so tudi poimenovane (*roll, pitch, yaw*). Slika je povzeta po [24].

Informacije o kvadrokopterju strežnik pošilja klientu preko protokola UDP na vratih 5554. Te informacije obsegajo stanje, v katerem se nahaja kvadrokopter, višino, na kateri se nahaja, hitrost, s katero se premika, naklon v vse tri smeri in stanje baterije. Kvadrokopter pošilja klientu tudi sliko s kamer preko protokola UDP na vratih 5555. Slika je lahko zakodirana na dva načina, UVLC (zelo podoben kodeku JPEG [29]) ali kodekom P264 (podoben kodeku H264 [30]). Zadnji način komunikacije poteka preko protokola TCP na vratih 5559 in je namenjen pošiljanju pomembnejših podatkov na strežnik. Preko te povezave lahko spreminjamo konfiguracijske nastavitve kvadrokopterja.

Dela na mobilni platformi Parrot AR.Drone smo se prvotno lotili z SDK-jem, ki je namenjen operacijskem sistemu Windows in delu v razvojnem okolju Visual Studio 10. Težave so tukaj nastopile že pri prevajanju izvorne kode. Po nekaj popravkih v sami kodi smo lahko pričeli z delom, toda tukaj so se prave težave šele začele. Najprej naj izpostavimo slabo kvaliteto povezave med kvadrokopterjem in računalnikom, ki je po navadi trajala le nekaj sekund, nato pa se je samodejno prekinila. Glavna težava dela z SDK-jem za operacijski sistem Windows pa je bila procesorska zahtevnost. Že za sam prikaz slike iz kvadrokopterja je bil procesor obremenjen s 100%, slika pa je vsakih nekaj sekund za prbližno sekundo obstala. To je bil tudi glavni razlog, da smo opustili prvotni načrt in celoten projekt prestavili na operacijski sistem Linux. SDK za Linux je veliko bolje podprt, saj nismo imeli nobenih težav s prevajanjem izvorne kode, izgubljanjem povezave, video pa se je prikazoval brez vmesnih premorov.

Poglavje 3

Metode za sledenje objektom

Za izvedbo sledenja objektom smo preizkusili dva obstoječa sledilnika. Preizkusili smo enostavnejšega, a ne tako natančnega in robustnega sledilnika CamShift [18]. Uporabili smo implementacijo, ki se nahaja v odprotokdni knjižnici OpenCV [13]. Njegova prednost je hitra integracija s sistemom in enostavna uporaba. Drugi sledilnik (LGT) [26], ki smo ga uporabili, je bil razvit na Fakulteti za računalništvo in informatiko v Laboratoriju za umetne vizualne spoznavne sisteme. Sledilnik predstavlja trenutno najsodobnejše metode s področja vizualnega sledenja.

3.1 Sledilnik CamShift

Algoritem CamShift (Continuously Adaptive Mean Shift Algorithm) je izpeljan iz algoritma Mean Shift [17]. Prvotno je bil namenjen za sledenje obrazom, vendar je lahko uporabljen tudi splošneje. Algoritem deluje tako, da najprej s pomočjo algoritma Mean Shift poišče center regije, nato pa poda še rotacijo in velikost objekta. Mean Shift je iterativen algoritem, ki v sliki išče regijo, ki se čim bolj ujema z referenčno. Iskanje temelji na podobnosti barvnega histograma regij v sliki. Uporabljen je barvni prostor HSV in sicer H (hue) komponenta. Za vsak slikovni element se izračuna verjetnost, da slikovni element pripada sledeni regiji. Dobi se mapa verjetnostne porazdeli-

tve. Algoritem Mean Shift na tej mapi poišče maksimum in na ta način dobi središče objekta. Da algoritem ne bi upošteval pretemnih in presvetlih točk, je potrebno določiti tudi mejni vrednosti za H barvno komponento. Algoritem dinamično prilagaja barvno porazdelitev ciljne regije med zaporednimi slikami. Ko algoritem CamShift določi regijo v sliki, ki se najbolj ujema z refrenčno, si zapomni njeno barvno porazdelitev in jo uporabi pri iskanju ciljne regije v naslednji sliki za referenčno. Med slikami se lahko spreminja barva in lokacija regije. Za mero podobnost, $d(K_1, K_2)$, med histogrami se uporablja Bhattacharyya-va razdalja

$$d(K_1, K_2) = \left(1 - \sum_{n=1}^{N} \sqrt{K_{1(n)} \cdot K_{2(n)}}\right)^{1/2}. \tag{3.1}$$

Notranji koren predstavlja skalarni produkt histogramov $K_1 = \{K_{1(n)}\}_{n=1:N}$ in $K_2 = \{K_{2(n)}\}_{n=1:N}$, ki sta predstavljena kot vektorja. Spremenljivka N predstavlja število celic v histogramu. Izraz je sorazmeren kosinusu kota med vektorjema K_1 in K_2. Delovanje sledilnika v praksi je prikazano na Sliki 3.1. Sledilnik z rdečo elipso označi sledeni objekt.

Slika 3.1: Delovanje sledilnika CamShift na primeru sledenja modre kroglice.

Algoritem lahko razdelimo na naslednje štiri korake:

1. Označimo regijo v kateri se nahaja objekt, ki mu želimo slediti. V nadaljevanju regijo poimenujemo regija zanimanja.

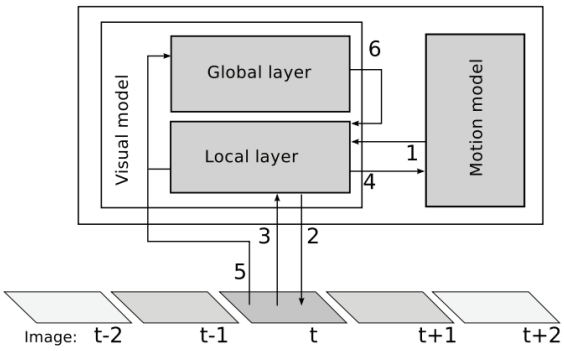
2. Izračuna se barvna porazdelitev regije zanimanja.

3. Iterativni algoritem Mean Shift najde na sliki regijo, ki se najbolj ujema z regijo zanimanja. Določi tudi center regije.

4. Regija se shrani kot regija zanimanja. Shrani se tudi središče regije. Ob naslednji sliki v zaporedju algoritem nadaljuje s točko 2.

3.2 Naprednejši sledilnik LGT

Algoritem CamShift za iskanje ujemajoče regije v sliki upošteva samo barvno informacijo. Zaradi preprostosti metode smo predvidevali, da bo za sledenje osebi v prostoru premalo robusten in nenatančen. Zato smo uporabili robustnejši sledilnik LGT [25], ki smo ga pridobili od avtorjev in uporablja naprednejše metode računalniškega vida. Sledilnik je sestavljen iz dveh nivojev, kot prikazuje Slika 3.2. Na njej so s številkami označeni: 1 - predikcija iz modela gibanja, 2 - ujemanje lokalne plasti, 3 - posodabljanje uteži regij iz lokalne plasti, 4 - posodabljanje modela gibanja, 5 - posodabljanj globalne plasti in 6 - dodajanje novih regij v lokalno plast.



Slika 3.2: Shema sledilnika LGT. Slika je povzeta po članku [25].

Lokalni nivo $\mathcal{L}_t$ je sestavljen iz regij, tako, da čim bolje opiše sledeni objekt ob času t

$$\mathcal{L}_t = \{\mathrm{x}_t^{(i)}, w_t^{(i)}\}_{i=1:N_t}. \tag{3.2}$$

Regija z zaporedno številko i je definirana s koordinatami $\mathbf{x}_t^{(i)}$ pozicije v sliki in uteži $w_t^{(i)}$, ki predstavlja "pomembnost" regije za opis objekta. Središče objekta $\mathbf{c_t}$ je predstavljeno kot uteženo povprečje regij

$$\mathbf{c_t} = \frac{1}{W_t} \sum_{i=1}^{N_t} w_t^{(i)} \cdot \mathbf{x}_t^{(i)}. \tag{3.3}$$

Element W_t v enačbi 3.3 predstavlja normalizacijski faktor, definiran kot $W_t = \sum_{i=1}^{N_t} w_t^{(i)}$. Predpostavlja se, da je pozicija posameznih regij v modelu odvisna samo od pozicij svojih neposrednih sosednjih regij, ki so določene z Delaunayevo triangulacijo [38]. Algoritem za vsako sliko poišče vrednosti uteži, tako da maksimizira skupno verjetnost $p(Y_t, X_t|\hat{X}_t)$. $\hat{X}_t$ predstavlja začetno oceno za množico regij, Y_t množico regij iz trenutne slike, X_t pa možno konfiguracijo regij. Ocena za slučajno spremenljivko $\tilde{X}_t$ je definirana kot

$$\tilde{X}_t = \arg\max_{X_t} p(Y_t, X_t|\hat{X}_t). \tag{3.4}$$

Vizualna verjetnost vsake regije je definirana z izrazom

$$p(Y_t|\mathrm{x}_t^{(i)}) = e^{-\lambda_v \rho(\mathbf{h}_{ref}^{(i)}, \mathbf{h}_t^{(i)})}. \tag{3.5}$$

Funkcija $\rho(.,.)$ predstavlja Bhattacharryya-evo razdaljo (3.1) med sivinskimi histogrami $\mathbf{h}_{ref}^{(i)}$ in $\mathbf{h}_t^{(i)}$. Prvi je referenčni histogram i-te regije, izračunan ob inicializaciji regije in se med sledenjem ne spreminja, drugi pa je histogram i-te regije izračunan ob trenutni sliki. Člen λ_v predstavlja konstanten faktor.

Predsavitev objekta z majhni regijami je robustna na nekatere deformacije (npr. rotacijo), vendar je uspešna za kratek čas, ponavadi le za nekaj slik. Zato je potrebno lokalno plast osveževati in regije, ki postanejo "zastarele", zamenjati z novimi. Regije, ki imajo vrednost uteži $w_t^{(i)}$ nižjo od pragovne vrednosti T_R, so odstranjene iz množice. Regije, ki so si preblizu skupaj, so združene v novo regijo, tako da ta predstavlja uteženo povprečje pozicije

starih regij, njena utež pa je povprečje uteži starih regij. Nove regije, ki se v lokalno plast dodajo iz globalne, dobijo ob inicializaciji vrednost uteži, ki je dvakrat večja od pragovne vrednosti T_R. Algoritem med delovanjem spreminja tudi največje število regij N_T^{cap}, ki se lahko nahajajo na lokalni plasti. Spreminjanje meje števila regij omogoča boljše delovanje algoritma, ker se s tem lahko prilagaja velikosti objekta. Mejna vrednost regij se določi po enačbi

$$N_{t+1}^{cap} = \alpha_{cap} N_t^{cap} + (1 - \alpha_{cap}) N_t. \tag{3.6}$$

N_t^{cap} predstavlja mejno vrednost pri prejšnji sliki, N_t število regij na lokalni lasti pri prejšnji sliki, α_{cap} pa eksponentni faktor pozabljanja.

Globalna plast sledilnika $\mathcal{G}_t$ je sestavljena iz treh vizualnih lastnosti ali značilnic: barve C_t, modela gibanja M_t in oblike S_t

$$\mathcal{G}_t = \{C_t, M_t, S_t\}. \tag{3.7}$$

Barvni model je predstavljen z dvema HSV histogramoma $\mathbf{h}_t^F$ in $\mathbf{h}_t^B$. Prvi predstavlja histogram objekta, drugi pa ozadja. Če z $I(\mathbf{x})$ označimo vrednost slikovnega elementa na poziciji $\mathbf{x}$ v sliki I, potem je verjetnost da pripada objektu $p(\mathbf{x}|F) = \mathbf{h}_t^F(I(\mathbf{x}))$ in verjetnost da pripada ozadju $p(\mathbf{x}|B) = \mathbf{h}_t B(I(\mathbf{x}))$. Verjetnost, da slikovni element pripada objektu je enaka

$$p(C_t|\mathbf{x}) = \frac{p(\mathbf{x}|F)p(F)}{p(F)p(\mathbf{x}|F) + (1 - p(F))p(\mathbf{x}|B)}. \tag{3.8}$$

Oba histograma se posodabljata med sledenjem. Histogram $\hat{\mathbf{h}}_t^F$ se izračuna iz regij na lokalni plasti, histogram ozadja $\hat{\mathbf{h}}_t^B$ pa se izračuna iz regije, ki predstavlja konveksno ovojnico okoli regij na lokalni plasti. Ta dva histograma posodabljata barvni model na globalni plasti po enačbah

$$\begin{aligned} \mathbf{h}_{t+1}^F &= \alpha_F \mathbf{h}_t^F + (1 - \alpha_F)\hat{\mathbf{h}}_t^F, \\ \mathbf{h}_{t+1}^B &= \alpha_B \mathbf{h}_t^B + (1 - \alpha_B)\hat{\mathbf{h}}_t^B. \end{aligned} \tag{3.9}$$

Člena α_F in α_B predstavljata stopnjo prilagodljivosti.

Model gibanja določajo značilne točke, ki jih dobimo s Harrisovim detektorjem robov [10]. Model $p(M_t|\mathbf{x})$ je definiran z enačbo

$$p(M_t|\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{N_s} p(\mathbf{x}_i|M_t)\Phi_\Sigma(\mathbf{x}-\mathbf{x}_i). \tag{3.10}$$

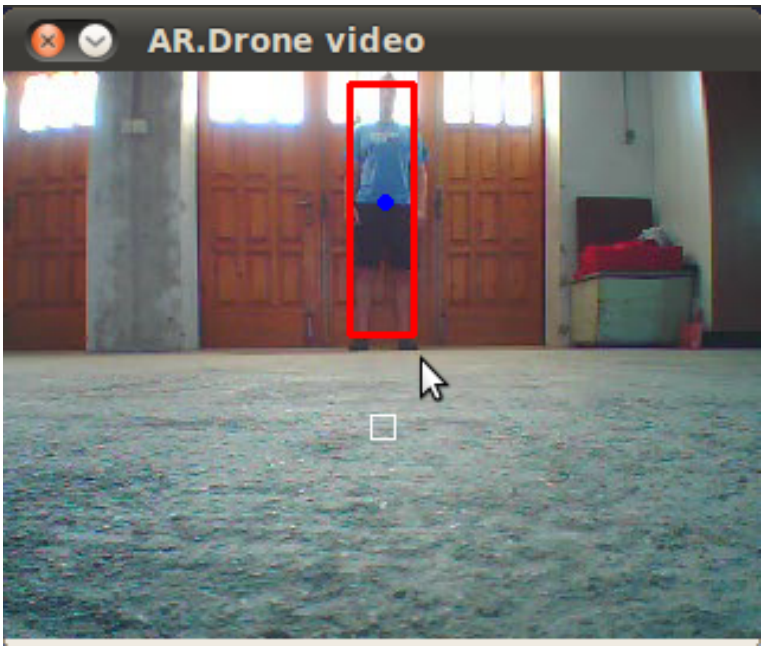
Člen $\Phi_\Sigma(.)$ predstavlja gausovo jedro s kovarianco Σ, N_s je število vseh značilnih točk v modelu, $p(\mathbf{x}_i|M_t)$ pa verjetnost gibanja posamezne značilne točke $\mathbf{x}_i$. Verjetnost $p(\mathbf{x}_i|M_t)$ je definirana z enačbo

$$p(\mathbf{x}_i|M_t) = (1-w_{noise})e^{-\lambda_M(d(v(\mathbf{x}_i),v_t))} + w_{noise}. \tag{3.11}$$

Funkcija $d(v(\mathbf{x}_i), v_t))$ predstavlja razdaljo med dvema hitrostima značilnih točk, w_{noise} konstanten šum in λ_M konstanto.

Oblika predstavlja oceno oblike objekta, definirano kot orientirana regija P_t. Izračuna se kot konveksna ovojnica čez regije iz lokalne plasti.

Ker je sledilnik napisan v programskem jeziku C++, naš projekt pa v jeziku C, smo morali za integracijo napisati vmesnik, ki nam omogoča, da lahko funkcije iz jezika C++ kličemo v našem projektu. Vsaka funkcija vmesnika samo kliče obstoječo funkcijo iz sledilnika z ustreznimi argumenti. Slika 3.3 prikazuje okno z videom iz kamere na kvadrokopterju in delovanje sledilnika LGT. Prikazana je pozicija središča objekta (označena z modro piko) in regije v kateri se nahaja objekt (rdeč pravokotnik).

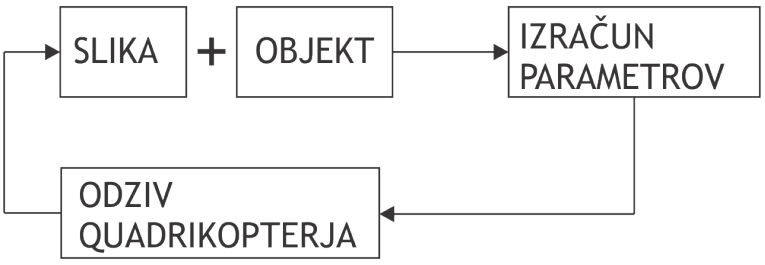


Slika 3.3: Delovanje sledilnika na videu iz kvadrokopterja.

Poglavje 4

Izvedba vodenja s kamero

V tem poglavje opišemo naš postopek, s katerim kvadrokopter na podlagi lokacije objekta v sliki avtonomno regulira svoj let. Ker se objekt iz 3D prostora preslika v slikovno ravnino kamere v 2D, je potrebno iz dobljene informacije oceniti položaj objekta v prostoru. V Podpoglavju 4.1 opišemo, kako se točke podane v treh dimenzijah preslikajo v točke na slikovnem senzorju. V nadaljevanju opišemo naš postopek regulacije leta kvadrokopterja s pomočjo ocene o položaju objekta v prostoru. Sledenje smo razdelili v dva dela: (i)ocena odklona, ter (ii)premik plovila vzdolž osi x. Shema našega pristopa je prikazana v Sliki 4.1.



Slika 4.1: Shema našega pristopa k avtonomnemu vodenju.

4.1 Model kamere

Podpoglavje opisuje model kamere ter kako se točke iz 3D koordinat preslikajo na slikovni senzor v 2D koordinate. Iz notranjih parametrov kamere, ki jih pridobimo s kalibracijo in polažaja objekta v sliki lahko ocenimo položaj objekta v prostoru. Vse točke so zapisane v homogenih koordinatah. V (4.1) vidimo, kako se točka v prostoru X_w s pomočjo projekcijske matrike kamere P preslika v točko na slikovnem senzorju X_i

$$X_i = P \cdot X_w. \tag{4.1}$$

Projekcijsko matriko lahko zapišemo tudi kot

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix}. \tag{4.2}$$

Matrika P predstavlja produkt kalibracijske matrike kamere K in rotacijske matrike R ter translacijskega vektorja t

$$P = K \cdot [R|t]. \tag{4.3}$$

Kalibracijska matrika kamere je sestavljena iz notranjih parametrov kamere, ki jih dobimo pri kalibraciji. Ti parametri so goriščni razdalji f_x in f_y, ter koordinati središča slikovnega senzorja c_x in c_y

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.4}$$

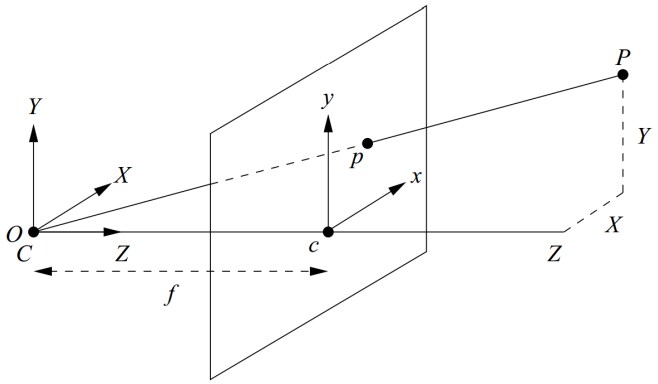
Rotacijska matrika R je sestavljena iz orientacije in položaja kamere v prostoru. Orientacijo predstavljajo koti, za katere je kamera zasukana okoli vseh treh osi in so označeni z ϕ, ψ in θ. V našem primeru dobimo te podatke iz senzorjev kvadrokopterja

$$R = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}. \tag{4.5}$$

Informacijo o položaju kvadrokopterja glede na objekt nosi translacijski vektor t.

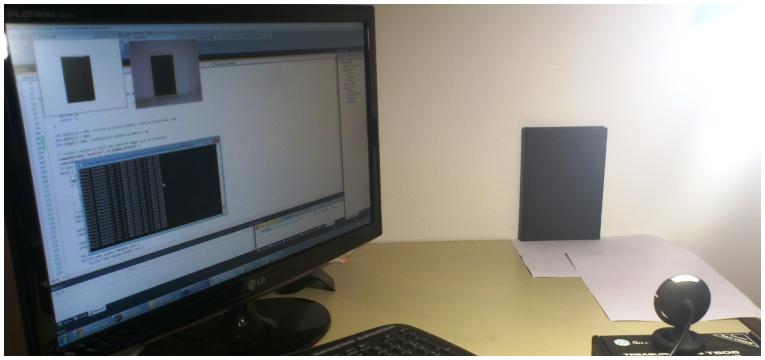
4.2 Kalibracija kamere

Postopek kalibracije kamere se uporablja za pridobitev notranjih parametrov kamere. Ti parametri so: goriščna razdalja, pozicija središča kamere, zamik slikovnih elementov ter ukrivljenost slike. V enačbi (4.4) vidimo, da od teh parametrov za določitev matrike K potrebujemo goriščno razdaljo in pozicij središča v sliki. Gorišče predstavlja točko, v kateri se zberejo vsi svetlobni žarki, goriščna razdalja pa nam pove, koliko je ta točka oddaljena od leče. Na Sliki 4.2 je označena s črko f. Ker ni nujno, da so slikovni elementi pravilne kvadratne oblike, računamo dve goriščni razdalji f_x in f_y, njuno razmerje $\frac{f_x}{f_y}$ pa predstavlja razmerje med dimenzijama enega slikovnega elementa. Podobno je tudi pri postavitvi središča kamere c_x in c_y, ki je ponavadi v središču slike, v praksi pa lahko pride do odstopanj. Središče je na Sliki 4.2 označeno s črko C. Na Sliki 4.2 so prikazani še točka objekta v prostoru P, svetovni koordinatni sistem (X, Y, Z), središče slike c, koordinatni sistem slike (x, y) in točka p preslikana iz prostora na slikovno ravnino.



Slika 4.2: Shema kamere in njenih notranjih parametrov.

Za kalibracijo smo preizkusili dve metodi [11, 14], ki se razlikujeta po kompleksnosti. Za preprostejšo smo potrebovali raven, enobarven, pravokoten predmet. Temu je najbolj ustrezala črna knjiga. Kalibracijo prikazuje Slika 4.3.



Slika 4.3: Preprosta kalibracija kamere.

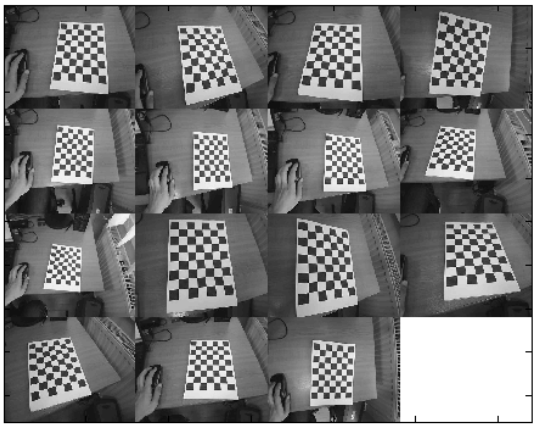
Knjigi smo izmerili višino dY in širino dX in jo postavili na ravno podlago. Izmerili smo tudi oddaljenost knjige dZ od kamere, katero smo kalibrirali. Ozadje in podlaga sta morala biti svetlejše barve, tako da sta se brez težav ločila od objekta, ki je bil poravnan vzporedno s slikovno ravnino. S slike, ki smo jo zajeli s kamero, smo pridobili še mere objekta v slikovnih elementih dx in dy. S temi podatki smo lahko določili goriščno razdaljo kamere po

enačbah

$$\begin{aligned} f_x &= \frac{dx}{dX} \cdot dZ, \\ f_y &= \frac{dy}{dY} \cdot dZ. \end{aligned} \tag{4.6}$$

Kalibracijo smo najprej izvedli z navadno spletno kamero, rezultate pa primerjali s kalibracijo iste kamere z naprednejšo metodo. Rezultati preproste kalibracije se nahajajo v Poglavju 5, v Tabeli 5.4.

Za drugo metodo kalibracije smo uporabili aplikacijo Matlab Calibraton Toolbox, ki je prosto dostopna na spletu [14]. Za razliko od prejšnje preproste metode s to dobimo vse notranje parametre kamere. Kalibracija se izvaja na vzorcu, ki je podoben črno-beli šahovnici. Vse, kar moramo tukaj storiti, je posneti sekvenco slik šahovnice pod različnimi zornimi koti ter z različnih razdalj. Nato označimo vse štiri vogale vzorca na vsaki sliki in dobimo vse notranje parametre kamere. Na Sliki 4.4 je prikazana sekvenca 15 slik, s katerimi smo izvedli postopek kalibracije z orodjem Matlab Calibration Toolbox. Rezultati so predstavljeni v Poglavju 5 v Tabeli 5.5.



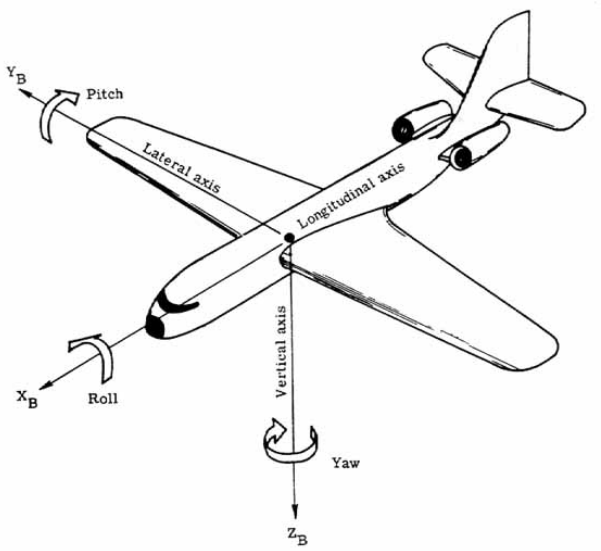
Slika 4.4: Sekvenca 15 slik, na kateri je bila izvedena kalibracija kamere z orodjem Matlab Calibration Toolbox [14].

4.3 Sprotno ocenjevanje pozicije objekta

Ocenjevanje pozicije objekta v prostoru je razdeljeno v dva dela. V prvem (Poglavje 4.3.1) je opisana ocena odklona - regulacija leta kvadrokopterja, ko se objekt premakne levo ali desno. V drugem delu (Poglavje 4.3.2) je opisan premik plovila vzdolž osi x (naprej ali nazaj) - ko se objekt v prostoru približa ali oddalji.

4.3.1 Ocena odklona

Najlažji del sledenja je ohranjanje objekta v središču slike po horizontalni osi. Na Sliki 4.5 vidimo, da gre pri tem za spreminjanje odklona (rotacije *yaw*).

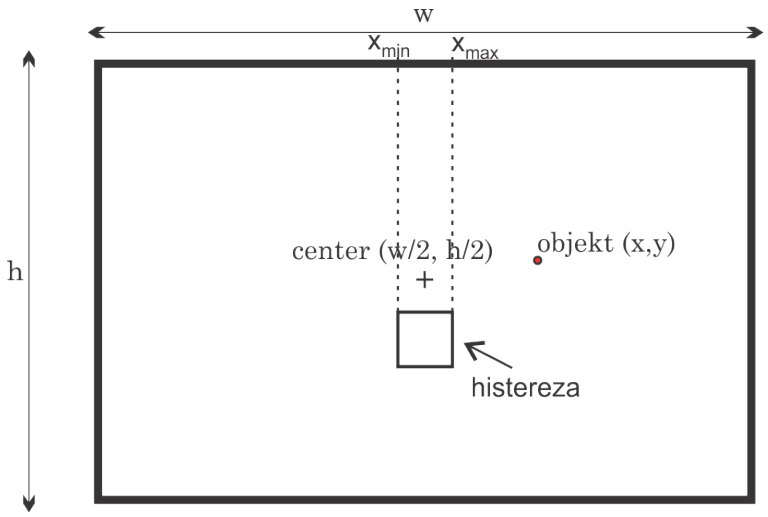


Slika 4.5: Rotacije okrog vseh treh osi. Slika je povzeta po [24].

Kot je opisano v Poglavju 2, kvadrokopter premikamo tako, da spreminjamo argumente ukaza za premik. Argumenti sprejmejo števila s plavajočo vejico med 0 in 1, ki predstavljajo za koliko se bo kvadrokopter premaknil. Ker želimo, da se objekt vedno nahaja čim bližje slike, izračunamo intenziteto odklona z enačbo

$$yaw = \frac{x - \frac{w}{2}}{\frac{w}{2}}. \tag{4.7}$$

Spremenjlivka x predstavlja trenutno x-koordinato centra objekta v sliki, w pa širino slike v slikovnih elementih. Da bi zagotovili večjo *stabilnost* kvadrokopterja v zraku, smo uporabili histerezo. To pomeni, da spremembe položaja središča objekta v sliki, ki se dogajajo v področju slike med x_{min} in x_{max}, ne upoštevamo. V tem primeru znaša odklon 0.0. S tem povečamo robustnost na šumenje v meritvi položaja objekta. Shema na Sliki 4.6 prikazuje histerezo in spremenljivke iz enačbe (4.7).



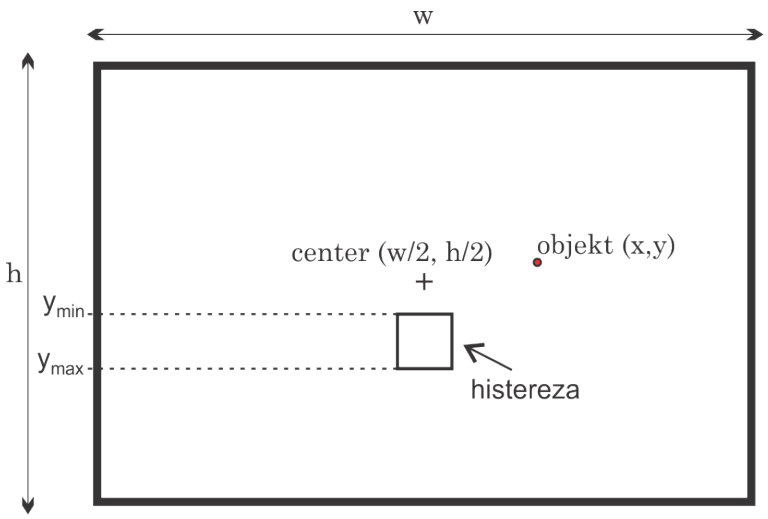
Slika 4.6: Shema histereze v sliki.

4.3.2 Premik vzolž osi x

Ohranjanje položaja v sliki

Na tem mestu smo predpostavili, da sledimo objektu, kateremu se ne spreminja višina in da je kvadrokopter vedno v vodoravnem položaju. To v praksi ni čisto res, saj se mu ob povečevanju hitrosti v smeri naprej oziroma nazaj spreminja tudi naklon (angl. pitch). Zato smo pošiljali ukaze, ki so dovolj "majhni" tako, da smo se tej predpostavki čim bolj približali. To je seveda botrovalo tudi k manjši odzivnosti sistema. Princip delovanja je zelo podoben kot pri oceni odklona. Tudi tukaj uporabimo histerezo, ki pa je postavljena nekoliko pod središče slike. Zato je tudi histereza na Sliki 4.7 postavljena pod središče slike. Takšno postavitev smo morali uporabiti, ker se spremembe položaja objekta v prostoru, v sliki ne poznajo, če je središče kamere na isti višini kot središče objekta.

Koraki, ki se izvedejo od vzleta, so naslednji: kvadrokopter se dviga oziroma spušča toliko časa, dokler se ne nahaja na višini tako, da je objekt, ki smo ga predhodno označili, na ustrezni višini v sliki (v kvadratku, ki predstavlja histerezo na Sliki 4.7). Ko je ta pogoj izpolnjen, si sistem zapomni



Slika 4.7: Shema histereze v sliki.

višino, ki jo izmeri z ultrazvočnim senzorjem višine in jo ohranja skozi celotno fazo leta. Stalno skrbi tudi, da se objekt nahaja v območju histereze. Torej če se spusti pod mejo y_{min}, to pomeni, da se je razdalja med objektom in kvadrokopterjem povečala, torej se mora premakniti naprej in obratno. Intenziteto premika vzdolž x-osi se izračuna z enačbo

$$pitch = \begin{cases} -\frac{y_{min}-y}{y_{min}\cdot\varepsilon} & ; y < y_{min} \\ \frac{y-y_{max}}{(h-y_{max})\cdot\varepsilon} & ; y > y_{max} \end{cases} \tag{4.8}$$

Spremenljivka y označuje y-koordinato trenutnega položaja objekta v sliki, y_{min} predstavlja y-koordinato vrhnje stranice, y_{max} pa y-koordinato spodnje stranice histereze. Spremenljivka h predstavlja višino slike podano v slikovnih elementih. Spremenljivke so prikazane na shemi v Sliki 4.7. V imenovalcu ulomka smo množili z ε, da smo upoštevali manjše premike kot bi bili v resnici, zaradi predpostavke o vodoravni legi kvadrokopterja. S poizkusi smo določili, da je najboljša vrednost $\varepsilon = 4.0$.

Sprotno računanje in ohranjanje razdalje do objekta

Ideja, ki smo jo uporabili na tem mestu, je sledeča. Sistem na začetku sledenja izračuna razdaljo do objekta na podlagi informacije iz slike in senzorjev ter nato za vsako sliko na podoben način računa razdaljo in se ustrezno odzove. Za implementacijo smo morali razumeti delovanje in teorijo sistema kamere, ki je opisana na začetku poglavja. Najprej smo morali določiti položaj koordinatnega sistema prostora glede na koordinatni sistem kamere. Izhodišče koordinatnega sistema prostora smo postavili v izhodišče koordinatnega sistema kamere, le da smo ga pomaknili na tla. Zato je edina neničelna komponenta translacijskega vektorja t višina h, ki jo sistem dobi iz ultrazvočnega senzorja višine kvadrokopterja

$$t = \begin{bmatrix} h \\ 0 \\ 0 \end{bmatrix}. \tag{4.9}$$

Položaj objekta X_w v prostoru smo definirali kot

$$X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}. \tag{4.10}$$

Komponenta x_w predstavlja višino objekta, z_w pa oddaljenost od kamere. V tem delu nas komponenta y_w ne zanima, saj jo obravnavamo v Poglavju 4.3.2. Zato za njeno vrednost vzamemo 0. Sledenje je v tem primeru sestavljeno iz dveh faz. Najprej se izračuna višina objekta v svetovnih koordinatah, tako da se določi referenčno razdaljo, na primer $z_w = 1$,

$$x_w = \frac{y - P_{23} - P_{24}}{P_{21}}. \tag{4.11}$$

V (4.11) in (4.12) predstavlja spremenljivka y, y-koordinato središča objekta v sliki, členi P_{23}, P_{24} in P_{21} pa elemente matrike P, tako kot smo opisali na začetku poglavja v (4.2). Nato sistem za vsako sliko ob znani višini objekta

računa razdaljo, jo primerja z referenčno in naredi ustrezen premik. Razdalja se računa po enačbi

$$z_w = \frac{y - (P_{21} \cdot x_w) - P_{24}}{P_{23}}. \tag{4.12}$$

Če je izračunan z_w večji od 1, pomeni, da se je objekt oddaljil in moramo kvadrokopter premakniti naprej, sicer pa obratno. Tudi v tem primeru smo dovolj majhne spremembe zanemarili.

Algoritem 1 predstavlja zaporedje izračunov, ki se zgodijo za eno sliko v sekvenci slik iz kamere kvadrokopterja.

Algorithm 1: Iteracija za eno sliko

Input: Slika I_i iz zaporedja

Initialize: $yaw \leftarrow 0$, $pitch \leftarrow 0$

Sledilnik na sliki I_i poišče središče sledenega objekta $\mathbf{x}_i(x_i, y_i)$

if $(x_i < x_{min}) \vee (x_i > x_{max})$ **then**

  $yaw = \frac{x - \frac{w}{2}}{\frac{w}{2}}$

if $y_i < y_{min}$ **then**

  $pitch = -\frac{y_{min} - y}{y_{min}\varepsilon}$

if $y_i > y_{max}$ **then**

  $pitch = \frac{y_i - y_{max}}{(h - y_{max})\varepsilon}$

Output: Vrednosti premikov yaw in $pitch$

Poglavje 5

Rezultati

Poglavje je razdeljeno na štiri dele. Najprej opišemo natančnost senzorjev in navedemo nekaj meritev, ki smo jih izvedli. Sledi del, ki je namenjen rezultatom kalibracije kamere in primerjavi dveh postopkov kalibracije, ki smo jih opisali v Poglavju 2. Za testiranje delovanja sledenja smo opisali sledenje pod "idealnimi" pogoji. Za robustnost sledilnika pa sledenje osebi v prostoru. Oba načina smo izvedli z dvemi sledilniki s Poglavja 3 in ju primerjali. Rezultate smo predstavili s pomočjo tabel in grafov. Vsi testi in eksperimenti so bili izvedeni na prenosniku Lenovo 3000 N500, s procesorjem Intel Core2Duo T6400 2.0GHz, 4GB pomnilnika RAM in grafično kartico NVidia GeForce 9300M GS.

5.1 Senzorji

Meritve smo izvedli za senzor višine ter za obe rotaciji, ki jih uporabljamo, nagib in odklon. Pri ultrazvočnemu senzorju višine je zanimivo, da kvadrokopter na tleh ne kaže višine 0, ampak okoli 210mm. Do te napake pride najverjetneje zaradi načina delovanja senzorja, saj ultrazvočni senzor deluje tako, da oddajnik pošilja ultrazvočne impulze, ki se od ovir odbijajo, sprejemnik pa meri čas od poslanega impulza do takrat, kadar pride odbit impulz nazaj. Če sta oddajnik in sprejemnik preblizu tal, se impulzi razpiršijo in

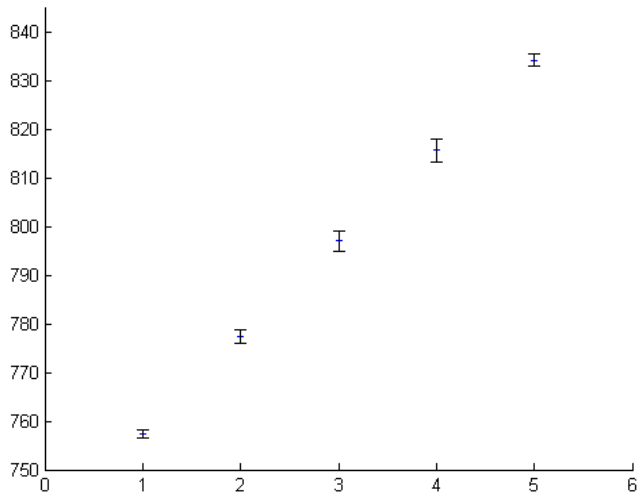
ne pridejo neposredno do senzorja. Meritve višine smo izvajali tako, da smo kvadrokopter postavljali na različne višine (Slika 5.1) in odčitali senzor, ki je kazal višino od kvadrokopterja do tal.

Slika 5.1: Merjenje višine s senzorjem višine kvadrokopterja.

Meritve smo izvedli za pet različnih višin, vsaka je bila dva centimetra višja od prejšnje. Za vsako višino smo meritev ponovili deset krat, izračunali srednjo vrednost in standardni odklon. Tabela 5.1 kaže meritve za višinometer. Na Sliki 5.2 je prikazan graf meritev za senzor višine. Vsaka navpična črta prikazuje standardni odklon za posamezno meritev, srednja črtica pa srednjo vrednost desetih meritev za posamezno višino. Glede na velikost standardnega odklona je mogoče opaziti, da so meritve, ko je kvadrokopter bližje tlem, natančnejše od tistih, ko je kvadrokopter višje.

Izmerjena višina [mm]	**Povprečje - μ [mm]**	**Standardni odklon - σ [mm]**
800	757.3	1.6
820	777.4	2.6
840	797.0	4.3
860	815.6	4.6
880	834.1	2.6

Tabela 5.1: Tabela meritev za senzor višine.



Slika 5.2: Graf rezultatov meritve za senzor višine.

Podobno kot za višino smo natančnost izmerili tudi za obe rotaciji, ki sta še posebej pomembni za naš problem sledenja. Leseno podlago smo postavili ob zid, ji spreminjali naklon ter odčitali senzor. Primer prikazuje Slika 5.3. Najprej smo meritve izvedli za odklon. Tabela 5.2 prikazuje rezultate meritev. V prvem stolpcu so dejanske vrednosti kotov za katere je bil kvadrokopter nagnjen, v drugem stolpcu so povprečne vrednosti desetih meritev, v tretjem pa standardni odklon. Podobno je tudi v Tabeli 5.3, le da smo merili natančnost izmerjenega nagiba. Iz obeh tabel lahko zaključimo, da je natančnost žiroskopov relativno visoka, saj je rezlika s pravo vrednostjo zanemarljiva, prav tako je tudi standardni odklon majhen.

Dejanski naklon [°]	**Povprečje -** μ [°]	**Standardni odklon** σ [°]
0	2.4	0.4
12	13.8	0.7
49	50.5	0.8
61	62.4	1.2

Tabela 5.2: Meritve za senzor odklona.

Dejanski naklon [°]	**Povprečje -** μ [°]	**Standardni odklon** σ [°]
0	2.5	0.5
12	13.4	0.8
49	50.5	0.8
61	62.2	1.2

Tabela 5.3: Meritve za senzor nagiba.

Slika 5.3: Merjenje natančnosti senzorjev rotacij.

5.2 Kalibracija kamere

V tem delu bomo predstavili rezultate kalibracije kamere, ki smo jo opisali v Poglavju 2. Testirali smo dve metodi, ju primerjali ter izbrali primernejšo za naš problem. Najprej smo izvedli eksperiment s preprosto metodo. V Tabeli 5.4 so rezultati kalibracije spletne kamere s preprosto metodo. Ker smo kamero postavljali na različne razdalje od objekta, predstavlja prvi stolpec razdaljo kamere do objekta v milimetrih. Drugi in tretji stolpec predstavljata vrednosti za goriščni razdalji f_x in f_y. Povprečni vrednosti za obe goriščni razdalji sta $\mu_{f_x} = 421.3$ in $\mu_{f_y} = 417.5$, varianci pa $\sigma^2_{f_x} = 3.9$ in $\sigma^2_{f_y} = 3.5$.

Za drugo metodo smo uporabili orodje Matlab Calibration Toolbox [14]. Orodje izračuna več parametrov kot pa samo goriščno razdaljo, zato Tabela 5.5 prikazuje vse parametre, ki jih je program izračunal. Kalibrirali smo isto kamero kot v prejšnem primeru s preprosto metodo. Ker aplikacija Matlab Calibration Toolbox upošteva tudi orientacijo kamere in njen položaj glede na objekt, je natančnejša, daje nam pa tudi rezultate za parameter središča kamere (c_x in c_y). Zato smo se odločili, da za določitev notranjih

Razdalja [mm]	$\mathbf{f}_x$	$\mathbf{f}_y$
485	421	418
520	418	414
535	422	419
560	422	419
580	421	417
620	424	418
povprečna vrednost	421.3	417.5

Tabela 5.4: Rezultati preproste kalibracije kamere.

parametrov kamere kvadrokopterja AR.Drone uporabimo to aplikacijo. Sicer se tudi rezultati preproste metode kalibracije ne bistveno razlikujejo od natančnejše metode, kar pomeni, da lahko kamero kalibriramo tudi brez posebnega predznanja o delovanju aplikacij za kalibriranje.

Parameter	**vrednost**
Goriščna razdalja f_x	430
Goriščna razdalja f_y	427
Središče kamere c_x	165
Središče kamere c_y	128
Zamik α	0.0°

Tabela 5.5: Rezultati kalibracije navadne spletne kamere z aplikacijo Matlab Calibration toolbox.

Tabela 5.6 prikazuje rezultate kalibracije kamere na kvadrokopterju. Parametre smo potrebovali pri ocenjevanju položaja objekta v prostoru glede na informacijo o položaju objekta v sliki, ki jo je posredoval sledilnik.

Parameter	vrednost
Goriščna razdalja f_x	208
Goriščna razdalja f_y	206
Središče kamere c_x	170
Središče kamere c_y	115
Zamik α	0.0°

Tabela 5.6: Rezultati kalibracije kamere kvadrokopterja AR.Drone z aplikacijo Matlab Calibration toolbox.

5.3 Sledenje pri idealnih pogojih

Problem sledenja smo poenostavili tako, da smo si za objekt izbrali pravokoten, enobarven (zelen) predmet in ga postavili na določeno višino. Prostor smo ogradili tako, da smo čim bolj izničili vpliv ozadja. Postavitev v prostoru je prikazana na Sliki 5.4.

Slika 5.4: Ograjen prostor z zelenim predmetom.

Želeli smo testirati odziv kvadrokopterja na spremembo položaja objekta v prostoru. Pri premikanju objekta levo in desno je delovanje zelo dobro in sledenje deluje tudi ob hitrejših premikih. Pri premikanju naprej in nazaj potrebuje kvadrokopter določen čas za reakcijo in za dosego končne pozicije.

Zato smo v Tabeli 5.7 prikazali natančnost sledenja, ko se objekt premakne za znan premik naprej oziroma nazaj. V prvem stolpcu so izmerjene vrednosti po pristanku kvadrokopterja, v drugem pa razlika z referenčno vrednostjo (2 metra) premika objekta. Zelen predmet smo za dva metra premaknili proti kvadrokopterju in poizkus ponovili dese krat. Pri meritvah je potrebno upoštevati, da smo razdaljo merili na tleh, torej pred vzletom in po pristanku. Ker kvadrokopter ne vzleti popolnoma navpično, se tukaj pojavi določena napaka, ki je nismo mogli ustrezno upoštevati. Za poizkus smo uporabili sledilnik CamShift.

Izmerjen premik [cm]	**Razlika [cm]**
236	36
260	60
203	3
244	44
277	77
229	29
282	82
253	53
206	6
207	7

Tabela 5.7: Rezultati poizkusa merjenja natančnosti sledenja za znan premik s sledilnikom CamShift.

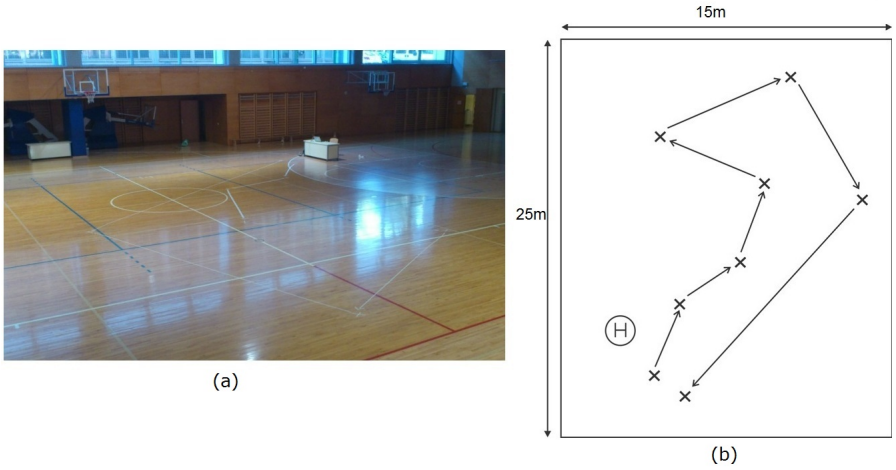
Enak poizkus smo naredili še z naprednejšim sledilnikom LGT. Ugibanja so se izkazala za pravilna, saj rezultati niso bili vidno boljši. Pogoji pri tem poizkusu so bili dovolj omejeni, da je tudi preprostejša metoda za sledenje dala dobre rezultate. Rezultati poizkusa so prikazani v Tabeli 5.8. Pri poizkusu s sledilnikom CamShift je bila povprečna vrednost desetih premikov 239.5 cm (standardni odklon 29.1 cm), z naprednejšim sledilnikom LGT pa 237.5 cm (standardni odklon 28.8 cm). Glede na povprečno vrednost in standardni odklon obeh poizkusov opazimo, da ni bistvenih razlik v natančnosti glede na sledilnik, ki smo ga uporabili.

Izmerjen premik [cm]	**Razlika [cm]**
214	14
233	33
208	8
261	61
210	10
237	37
251	51
204	4
283	83
274	74

Tabela 5.8: Rezultati poizkusa merjenja natančnosti sledenja za znan premik z naprednejšim sledilnikom LGT.

5.4 Sledenje realnemu objektu

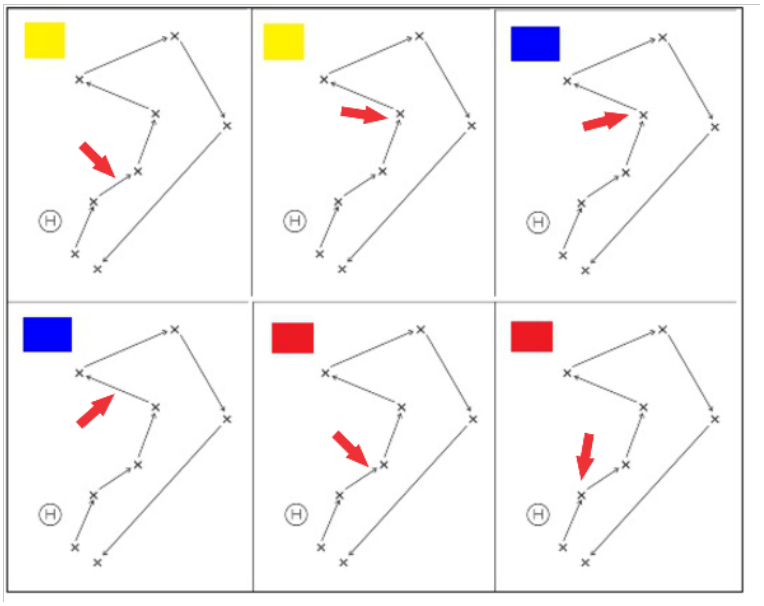
V zadnji fazi testiranja smo želeli preizkusiti robustnost sledenja v realnem okolju. Zato smo izvedli sledenje osebi v večjem prostoru. Sledenje smo izvajali v telovadnici, v kateri je dovolj prostora za gibanje. Prav tako so tudi pogoji za letenje optimalni. Temu najbolj pripomore ravna podlaga in velikost prostora zaradi nemotenega pretoka zraka, saj se v manjših prostorih pogosto dogaja, da kvadrokopter zaradi "vetra", ki ga ustvarjajo propelerji, zanaša iz smeri letenja. To smo opazili, da se dogaja predvsem ob stenah in podobnih ovirah. Na Sliki 5.5 sta prikazana telovadnica, v kateri je začrtan poligon, na katerem smo izvajali eksperimente in shema poligona. S puščicami je označena smer gibanja sledene osebe.



Slika 5.5: Poligon v telovadnici(a) in njegova shema(b).

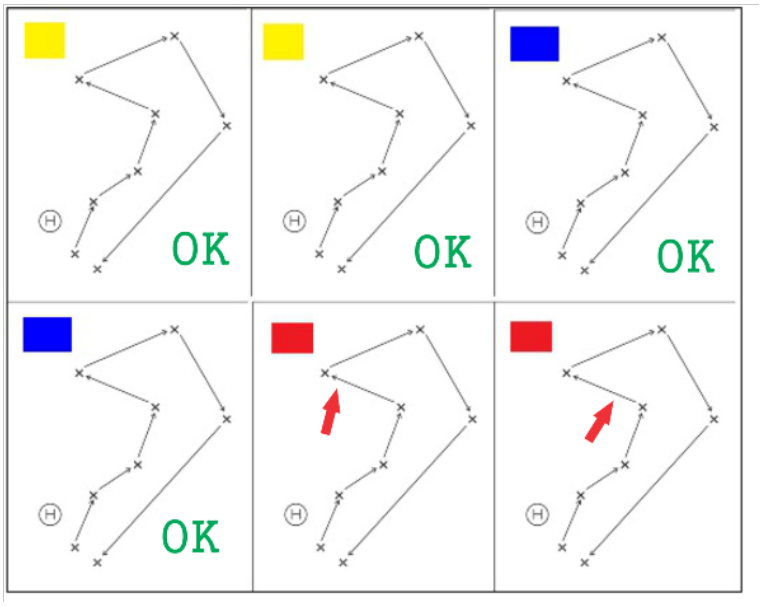
Kvadrokopter je gibanju sledil le, če ni bilo prehitro. Da bi testirali robustnost sistema smo za vsak sledilnik izvedli šest poizkusov. S poizkusi smo ugotovili, da na delovanje sledenja najbolj vpliva barva objekta v primerjavi z barvo prostora v katerem se nahaja. V vsakem poizkusu je oseba prehodila poligon, merili pa smo, če kvadrokopter uspešno sledi osebi do konca. Za vsak poizkus je prikazan poligon, mesto, na katerem je sistem odpovedal in v levem zgornjem kotu barva majice, ki jo je imela oseba oblečeno. V

primeru, da sistem ni odpovedal in je kvadrokopter prišel do konca poligona je v desnem spodnjem kotu sheme poligona znak OK. Na Sliki 5.6 je prikazana uspešnost sledilnika CamShift pri sledenju osebi. Sistemu s sledilnikom CamShift ni uspelo nikoli priti do konca poligona. S tem se je naša domneva o potrebi po robustnejšem sledilniku izkazala za pravilno.



Slika 5.6: Shema šestih poizkusov s sledilnikom CamShift.

Sistem z naprednejšim sledilnikom se je veliko bolje odrezal. Na Sliki 5.7 vidimo uspešnost sistema in mesta, kjer je odpovedal. Od šestih poizkusov je v štirih prišel uspešno do konca, odpovedal pa je v dveh primerih. To se je zgodilo, ko je imela oseba, kateri je sledil, oblečeno rdečo majico. To je mogoče pojasniti s tem, da je ozadje v telovadnici rdeče barve. Primer, ko je sistem zaradi sledilnika odpovedal, je prikazan na Sliki 5.8. Prikazana je sekvenca slik, ki jih je posnela kamera kvadrokopterja med letom.



Slika 5.7: Shema šestih poizkusov s sledilnikom LGT.

Kako izgleda sledenje osebi s kvadrokopterjem v resnici, si je mogoče ogledati na spletu [44]. Slika 5.9 prikazuje zaporedje slik sledenja osebi z naprednejšim sledilnikom LGT, ko je imela oblečeno rumeno majico, na Sliki 5.10 pa modro. Povprečen čas štirih poizkusov, v katerih je kvadrokopter preletel celoten poligon, je bil ena minuta in 16 sekund. Na Sliki 5.11 je prikazano zaporedje slik pri sledenju s sledilnikom CamShift do mesta, kjer je sistem odpovedal. V vsaki vrsti je prikazan poizkus z različno barvo majice.

Slika 5.8: Sekvenca slik posnetih s kvadrokopterjem. Uporabljen je bil sledilnik LGT. Primer, ko je sistem odpovedal.

Slika 5.9: Sekvenca slik posnetih s kvadrokopterjem. Uporabljen je bil sledilnik LGT.

Slika 5.10: Sekvenca slik posnetih s kvadrokopterjem. Uporabljen je bil sledilnik LGT.

Slika 5.11: Sekvenca slik posnetih s kvadrokopterjem, ki prikazuje tri poizkuse. Uporabljen je bil sledilnik CamShift.

Poglavje 6

Sklep

V diplomski nalogi smo izdelali robotski sistem, ki omogoča avtonomno vizualno zasledovanje objekta v prostoru z zračnim plovilom.Preizkusili smo metode za sledenje in jih integrirali v sistem. Razvili smo metode za regulacijo leta na podlagi slikovne informacije iz kamere na kvadrokopterju.

Opisali in primerjali smo dve metodi za sledenje - CamShift [18] in naprednejši sledilnik LGT [25]. Sledilnik CamShift smo uporabili za sledenje preprostemu enobarvnemu objektu v začetnih fazah razvoja sistema. Uporabili smo ga zaradi preproste integracije v sistem in hitrega delovanja. Pri poizkusu sledenja osebi je sledilnik odpovedal, zato smo uporabili naprednejšega (LGT). Njegova prednost je bila robustnost in sposobnost sledenju osebi v prostoru. Glavni tehnični izziv v diplomski nalogi je bil regulacija leta plovila na podlagi vizualne informacije, tako da je plovilo sledilo osebi, ki se je premikala po prostoru. Problem smo razdelili na dva dela in sicer oceno odklona in premik kvadrokopterja naprej, oziroma nazaj. Sistem smo testirali v večjem zaprtem prostoru tako, da je oseba hodila po začrtani poti. Izkazalo se je, da je delovanje odvisno od barvne podobnosti oblačil osebe in ozadja. Sistem je pogosteje odpovedoval, ko se je oseba gibala po prostoru z barvno podobnim ozadjem. V ostalih primerih problemov ni bilo in kvadrokopter je uspešno preletel poligon.

6.1 Možnosti za izboljšave

Ker je mobilna platforma opremljena s precej senzorji in dvema kamerama je dovolj prostora za izboljšave in morebitne dodatne funkcionalnosti. Predstavili bomo dve izboljšavi, ki bi bili za sistem najprimernejši.

Za začetek bi lahko omejili večja odstopanja, do katerih prihaja pri poziciji centra objekta, ki mu sledimo. To bi bilo smiselno narediti s pomočjo Kalmanovega filtra [12], ki skrbi za "zadušitev" večjih odstopanj pri vhodnem signalu. To bi nam omogočalo robustnejše delovanje sledilnika.

Kot drugo pomembno izboljšavo pa smatramo nadgradnjo sistema do te mere, da bi bil zmožen slediti tudi objektu, ki se premika veliko hitreje, npr. osebe pri športnih dejavnostih, tek, kolesarjenje, plavanje. Pri tem bi bilo treba ovreči predpostavko, da se kvadrokopter vedno nahaja v vodoravnem položaju in ustrezno upoštevati senzor rotacije. Ideja, ki bi lahko bila uspešna je, da regija v sliki, ki nam služi za histerezo, ne bi bila fiksno določena, ampak bi se premikala po sliki glede na rotacijo kvadrokopterja. Ob vodoravnem položaju kvadrokopterja bi bile meje histereze določene, ko pa bi se začel premikati po prostoru, bi se glede na informacijo iz žiroskopa v sliki ustrezno spreminjale.

S temi izboljšavami ter z boljšo strojno opremo bi bil sistem primernejši zahtevnejšim komercialnim aplikacijam, na primer sledenje športnikom v ekstremnih razmerah, na športnih prireditvah in snemanje na težje dostopnih mestih.

Literatura

[1] E. Altug, J. Ostrowski, R. Mahony, "Control of a quadrotor helicopter using visual feedback", na: *Proceedings of the IEEE International Conference on Robotics and automation*, IEEE, 2002, pp. 72-77.

[2] S. Bouabdallah, R. Siegwart. "Full Control of a Quadrotor", *Intelligent robots and systems, IROS 2007. IEEE/RSJ international conference*,p.p. 153-158, 2007.

[3] L. Derafa, A. Ouldali, T. Madani, A. Benallegue, "Four Rotors Helicopter Yaw and Altitude Stabilization," *Proceedings of the World Congress on Engineering 2007*, vol I, WCE 2007, July 2 - 4, 2007, London, U.K.

[4] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics*, Espoo, August 22, 2011.

[5] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice, N. Roy, "RANGE - Robust Autonomous Navigation in GPS-denied Environments," *IEEE International Conference on Robotics and Automation*, May 3-8, 2010, Anchorage, Alaska, USA, pp.1096-1097.

[6] L. Heng, G. Hee Lee, F. Fraundorfer, M. Pollefeys, "Real-Time Photo-Realistic 3D Mapping for Micro Aerial Vehicles," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 25-30, 2011. San Francisco, CA, USA.

[7] M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba, "A Solution to the Simultaneous Localization and

Map Building (SLAM) Problem", *IEEE transactions on robotics and automation*, vol. 17, no. 3, June 2001.

[8] M. Bošnak, D. Matko, S. Blažič, "Quadrocopter control using a non-board video system with off-board processing," *Robotics and Autonomous Systems 60*, p.p. 657–667, Ljubljana, 2012.

[9] J. Canny, "A computational approach to edge detection, "*Pattern Analysis and Machine Intelligence*, IEEE Transactions, p.p. 679-698, 1986.

[10] C. Harris, M.J. Stephens, "A combined corner and edge detector," *Alvey Vision Conference*, p.p. 147-152, Manchester, UK, 1988.

[11] J. E. Solem, "Programming Computer Vision with Python," 2012.

[12] R.E. Kalman, "A new Aproach to linear filtering and prediction problems," *Transactions of the American Society of the Mechanical Engineers*, Journal of the Basic Engineering, 82:34-45, 1960.

[13] OpenCV (2012). Dostopno na:
`http://opencv.willowgarage.com/wiki/`

[14] Matlab Calibration toolbox (2010). Dostopno na:
`http://www.vision.caltech.edu/bouguetj/calib_doc/`

[15] Rotacijska matrika (2012). Dostopno na:
`http://en.wikipedia.org/wiki/Rotation_matrix`

[16] Roll, Pitch in Yaw rotacije (2012). Dostopno na:
`http://planning.cs.uiuc.edu/node102.html`

[17] D. Comaniciu, P. Meer, "Mean Shift Analysis and Applications," *IEEE Int'l Conference Computer Vision*, Kerkyra, Greece, 1197-1203, 1999.

[18] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," *Intel Technology Journal Q2 '98*, Microcomputer Research Lab, Santa Clara, CA, Intel Corporation.

[19] CamShift algoritem - poenostavljeno (1998). Dostopno na: `http://www.robinhewitt.com/research/track/camshift.html`

[20] G. Depoyant, M. Ludmann, "Leading Units & Drone Enabled Probing" *Master project*, Aarhus, Denmark, August, 2011.

[21] Quadrikopter Parrot AR.Drone. Dostopno na: `http://ardrone.parrot.com/parrot-ar-drone/en/`

[22] Ar.Drone firmware (2012). Dostopno na: `http://drone-apps.com/support/ar-drone-firmware/`

[23] Ar.Drone API (2012). Dostopno na: `https://projects.ardrone.org/`

[24] Navodila za razvijalce in uporabo uradneka programskega paketa (2012). Dostopno na: `https://projects.ardrone.org/attachments/download/365/ARDrone_SDK_1_7_Developer_Guide.pdf`

[25] A. Leonardis, L. Cehovin, M. Kristan,"Robust Visual Tracking using an Adaptive Coupled-layer Visual Model,"IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society, Julij 2012.

[26] L. Čehovin, M. Kristan, A. Leonardis, "An adaptive coupled-layer visual model for robust visual tracking," *Computer Vision (ICCV)*, 2011 IEEE International Conference, p.p. 1363-1370, 2011.

[27] Kinect uradna spletna stran (2012). Dostopno na: `http://www.xbox.com/en-GB/KINECT`

[28] Kinect - predstavitev in osnovni principi delovanja (Wikipedia) (2012). Dostopno na: `http://en.wikipedia.org/wiki/Kinect`

[29] JPEG kodek za kodiranje slik v videu (Wikipedia) (2012). Dostopno na: `http://en.wikipedia.org/wiki/JPEG`

[30] H264 kodek za kodiranje slik v videu (Wikipedia) (2012). Dostopno na: http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[31] DHCP omrežni protokol (Wikipedia) (2012). Dostopno na: http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

[32] GRASP Laboratory, University of Pennsylvania, Philadelphia, USA. Dostopno na: https://www.grasp.upenn.edu/

[33] GRASP Laboratory: Robotic Quadcopters Work In Autonomous Swarms to Build Towers (2011). Dostopno na: http://www.popsci.com/technology/article/2011-01/grasp-lab-quadcopters-construct-towers-autonomous-swarms

[34] GRASP Laboratory: An Art Installation Sculpted by a Team of Swarming Autonomous Flying Robots (2011). Dostopno na: http://www.popsci.com/technology/article/2011-11/art-installation-sculpted-entirely-team-swarming-autonomous-flying-robot

[35] GRASP Laboratory: UPenn's Amazing Quad-Rotor Drones Now Work in Teams to Lift Heavy Payloads Together (2010). Dostopno na: http://www.popsci.com/technology/article/2010-07/video-upenns-quadcopters-now-work-teams-lift-heavy-payloads

[36] GRASP Laboratory:UPenn's GRASP lab unleashes a swarm of Nano Quadrotors (2012) Dostopno na: http://www.gizmag.com/grasp-nano-quadrotor-robots-swarm/21302/

[37] GRASP Laboratory: Learning acrobatic maneuvers for quadrocopters (2012). Dostopno na: http://www.autonomousrobotsblog.com/learning-acrobatic-maneuvers-for-quadrocopters/

[38] Delaunay triangulation - opis algoritma (Wikipedia) (2012). Dostopno na:
`http://en.wikipedia.org/wiki/Delaunay_triangulation`

[39] N.P. Papanikolopoulos, P.K. Khosla, T. Kanade, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision," *IEEE Transactions on Robotics and automatic*, vol. 9, no. 1, februar 1993

[40] M. Isard, A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision 29(1)*, 5–28 (1998)

[41] S. Zhou, R. Chellappa, B. Moghaddam,"Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters," *IEEE Transactions on Image Processing*, 13:11, pps. 1491-1506, 2004

[42] J.M. Rehg, T. Kanade, "Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking," *Third European Conf. on Computer Vision*, Stockholm, Sweden, May 1994, p.p. 35-46

[43] EVA Robotics: Quad-rotor UAV primer (2010). Dostopno na:
`http://evarobotics.com/latest-articles/quad-rotor-uav-primer`

[44] Demo posnetek sledenja osebi (YouTube) (2012). Dostopno na:
`http://www.youtube.com/watch?v=acUO_e6R7os&feature=plcp`