Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jon Natanael Muhovič

Vizualno sledenje objektov s kvadrokopterjem

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matej Kristan

Ljubljana, 2017

Zahvala

Zahvaljujem se mentorju dr. Mateju Kristanu za vodstvo pri izdelavi naloge, svojim bližnjim za vztrajno podporo in članom laboratorija ViCoS za produktivno okolje, v katerem je nastalo to delo.

Jon Natanael Muhovič, 2017

Kazalo

Povzetek

Naslov: Vizualno sledenje objektov s kvadrokopterjem

Zaradi vedno večje dostopnosti in kvalitete manjših letečih robotov za zasebno uporabo smo želeli nadgraditi njihovo funkcionalnost s sposobnostjo samodejnega sledenja želenemu objektu. To lahko naredimo z uporabo sledilnih algoritmov, ki delujejo na podlagi vizualne informacije. Taki algoritmi v vsaki sliki locirajo poljuben objekt in se prilagajajo spremembam velikosti in izgleda objekta. Formulirali smo sistem, ki uporabi informacije, pridobljene s takim algoritmom, za nadzor kvadrokopterja. Predlagani sistem tako ohranja razdaljo do sledenega objekta in hkrati skrbi, da je ta vedno blizu središča slike. Uporabili smo platformo s premično kamero, kar nam je omogočilo visoko odzivnost in sledenje objektu tudi, ko se ta nahaja niže od kvadrokopterja. Naš sistem nam poleg tega omogoča sledenje objektu, kadar se ta premika po neravnih površinah, kar s sistemi s fiksno kamero ni izvedljivo.

Ključne besede

kvadrokopter, vizualno sledenje objektov, računalniški vid, teorija krmiljenja

Abstract

Title: Visual object tracking with a quadcopter

Because of increasing accessibility and quality of small aerial robots for personal use we wanted to upgrade their functionality with the ability to autonomously follow objects. This can be achieved by using tracking algorithms that work with visual information. Such algorithms locate the object in every image and can adapt to size and appearance changes of the object. We have formulated a system that uses information from a tracking algorithm to control the quadcopter. The proposed system is thus able to maintain the distance to the object as well as keep the object near the center of the image. By using a platform with a movable camera we have achieved high responsiveness as well as the ability to follow an object that is located lower than the quadcopter itself. Our system also manages to follow targets that move on non-planar surfaces which is not possible using a fixed camera.

Keywords

quadcopter, visual object tracking, computer vision, control theory

Poglavje 1

Uvod

1.1 Motivacija

V zadnjih letih so postali manjši roboti za zasebno rabo precej popularni. Hkrati so se pojavile tudi težnje po nadgradnji njihovih funkcionalnosti, ne le njihovih tehničnih atributov. Govorimo o funkcionalnostih kot so avtonomna navigacija brez globalnega sistema za pozicioniranje (GPS), izogibanje oviram, planiranje poti ter samodejno sledenje objektom. V tem magistrskem delu želimo nasloviti zadnjo izmed njih, torej samodejno sledenje izbranemu objektu. Ker leteči roboti ponujajo unikaten zorni kot, so takorekoč vsi komercialni kvadrokopterji opremljeni s kvalitetnimi kamerami. Zaradi tega so primerna platforma za uporabo sledilnih algoritmov, ki so nastali na področju računalniškega vida. Takemu algoritmu lahko ročno označimo želeni objekt v sliki, ki ga bo skušal locirati v prihodnjih slikah. Raznolike okoliščine in visoka dinamika sistema pa lahko povzročijo težave, ki se pri drugih aplikacijah sledilnih algoritmov ne pojavljajo. Za dobro delovanje sistema je potrebna kontrola kvadrokopterja, ki reagira na premike objekta znotraj slike, prav tako je v dinamičnem videu inicializacija sledilnika s strani uporabnika zahtevnejša kot sicer. Tudi inicializacija sledilnika med tem, ko kvadrokopter še ne leti, se lahko izkaže za problematično, saj kratkoročni sledilniki objekt pogosto izgubijo še preden se robot v zraku stabilizira. Ker je ena od možnih

aplikacij tudi sledenje osebam, je treba nasloviti še psihološki aspekt tega, da robot leti v višini človekovega obraza, kar je lahko neprijetno in potencialno nevarno. Opazimo lahko tudi, da je večina eksperimentov s kvadrokopterji in sledilniki [1, 2, 3, 4] omejena na odprte in ravne prostore z enostavnimi trajektorijami, kar ne predstavlja nujno realističnih pogojev uporabe. Te probleme želimo nasloviti z implementacijo sistema, ki z uporabo podatkov, pridobljenih s sledilnikom, vgrajenimi senzorji ter znanjem o geometrijskih razmerjih med kvadrokopterjem in sledenim objektom skuša ohraniti objekt v centru vidnega polja kamere in tako omogoči dolgoročno sledenje izbrani tarči.

1.2 Pregled sorodnih del

Problem sledenja objektu s kvadrokopterjem so v dveh člankih reševali Pestana et al. [5, 4]. V članku [5] avtorji razvijejo metodologijo za vodenje kvadrokopterja brez uporabe globalnega sistema za pozicioniranje (GPS), v članku [4] pa ta regulator združijo z vizualnim sledilnikom openTLD [6] [1] in ga uporabijo za avtonomno sledenje poljubnemu objektu. Svoj sistem so implementirali na kvadrokopterju ARDrone 2.0 s fiksno kamero[2]. Uporaba fiksne kamere vnese nekatere omejitve, ki jih avtorji naslovijo. Za premik kvadrokopterja po oseh x in y je namreč potreben nagib celotne naprave, kar ob odsotnosti stabilizacije kamere neizogibno privede do občutnih sprememb v sliki. Za pravilno oceno spremembe višine objekta je tako treba upoštevati vpliv nagiba kvadrokopterja naprej ali nazaj. Fiksna kamera tudi omeji vidno polje med letenjem (sploh pri višjih hitrostih, saj je vertikalni odmik slike objekta direktno vezan na velikost nagiba kvadrokopterja) in oteži inicializacijo sledilnika pred vzletom. Zaradi težavnosti inicializacije sledilnika med letom je včasih lažje najprej izvesti inicializacijo sledilnika in šele nato sprožiti vzlet. Tu nastane problem, saj je programska oprema, ki regulira

[1] C++ koda dostopna na: https://github.com/gnebehay/OpenTLD

[2] https://www.parrot.com/si/en/drones/parrot-ardrone-20-elite-edition

vzlet komercialnih kvadrokopterjev, last izdelovalca naprave. Uporabnik med vzletom zato nima nadzora nad napravo, da bi vzlet upočasnil ali nanj kako drugače vplival, in mora to upoštevati pri načrtovanju sistema. Pri kvadrokopterju ARDrone 2.0 se izkaže, da se naprava med vzletom nagne nekoliko naprej, kar skoraj gotovo povzroči začasno izgubo sledenega objekta. Pestana et al. [4] problem te odpovedi rešijo z uporabo dolgoročnega sledilnika, ki ob izgubi objekta začne preiskovati sliko in v primeru ponovne detekcije objekta ponovno sproži sledenje. V veliko primerih se to sicer zgodi takoj po stabilizaciji kvadrokopterja po vzletu, če pa se vizualne okoliščine pri tem močneje spremenijo ta rešitev ni nujno učinkovita. Avtorji za določanje premika kvadrokopterja naprej oz. nazaj, ki je vezan na spremembo velikost objekta v sliki, predpostavijo velikost 40×30 cm in nastavijo želeno razdaljo do objekta na 3 metre. To naslovi omejitev, ki nastane zaradi uporabe ene same kamere, kar preprečuje določitev absolutne razdalje do posamezne točke v sliki.

Haag et al. so v članku [2] predstavili podoben sistem kot Pestana et al. [4], razvili so več sledilnikov, osnovanih na dveh kratkoročnih sledilnikih (DSST [7] in KCF [8]), katerima so dodali funkcionalnost za redetekcijo, ki deluje na osnovi dolgoročnega sledilnika TLD [6]. Nato so se osredotočili na zbiranje baze video posnetkov za testiranje, zajetih s kvadrokopterjem in na evalvacijo večjega števila različnih sledilnikov na tej bazi. Za ekspliciten nadzor kvadrokopterja na podlagi izhodnih podatkov sledilnika so uporabili regulator, opisan v članku [4], njihovo delo pa daje širši vpogled v tipične probleme, ki se pojavijo pri sledenju objektov zaradi lastnosti kvadrokopterja (predvsem zaradi fiksne kamere), kot so izguba objekta ob vzletu, večje spremembe izgleda ipd. Svojo zbirko posnetkov in kodo za uporabljene sledilnike so tudi javno objavili [3].

Danelljan et al. so v članku [3] za sledenje objektu uporabili sledilnik ACT (Adaptive Color Tracker [9]), svoj sistem pa so implementirali na platformi

[3] https://sites.google.com/site/trackingpursuit/

LinkQuad[4]. V svojem delu so sledilnik ACT kombinirali z detektorjem objektov kot načinom za verifikacijo rezultatov sledenja in hkrati za omogočenje sledenja večim tarčam naenkrat. Detektor objektov, osnovan na metodi HOG [10] so optimizirali za zaznavanje oseb, ta omejitev pa jim je omogočila tudi uporabo predpostavke o višini sledenih tarč. Če predpostavimo, da so ljudje v povprečju približno enako visoki, lahko z uporabo trigonometrije ocenimo absolutno razdaljo kamere do sledene osebe. Avtorji poleg tega omenijo tudi strategijo preiskovanja prostora z drsečim premikom za redetekcijo izgubljene oz. detekcijo še ne zaznane osebe.

M. Mueller et al. so v svojem delu [11] s pomočjo grafičnega pogona Unreal Engine 4[5] implementirali fotorealističen simulator kvadrokopterja, v katerem je mogoče testirati različne sledilne algoritme in kontrolne metode. Hkrati so generirali tudi obsežno zbirko posnetkov, ki simulirajo pogled kamere, pritrjene na kvadrokopter. V zadnjem času je zaradi visoke kvalitete računalniške grafike to postalo pogostejša praksa (za generiranje učnih primerov za strojno učenje [12, 13] ali za pridobivanje zbirk slik oz. videoposnetkov), saj zelo poenostavi tako pridobivanje kot tudi anotacijo posnetkov.

Isti avtorji so v članku [14] predstavili tudi svoj sistem za avtonomno sledenje objektov z letečim robotom, ki ga imenujejo PAT (angl. Persistent Aerial Tracking). Sistem za predikcijo lokacije tarče uporablja vizualni sledilnik Struck [15], modul za redetekcijo, osnovan na značilnicah HOG, in premično kamero skupaj z regulatorji PID za centriranje sledenega objekta v sliki kamere. Poleg predstavitve sistema za sledenje in evalvacije večjega števila modernih sledilnih algoritmov na zbirki zračnih posnetkov iz prej omenjenega članka [11] so avtorji predlagali tudi metodo za t.i. izročitev (angl. handover) sledenega objekta drugi napravi (konkretno, drugemu dronu). Ta pristop je mogoče uporabiti za nadaljevanje sledenja ob nizkem nivoju baterije prve naprave. Trenutno aktivna naprava pošlje svoje GPS koordinate, višino, smer in vizualni model tarče kontrolni postaji na tleh, ta pa informa-

[4]https://www.ida.liu.se/ jonkv82/cuas/platform/linkquad

[5]https://www.unrealengine.com

cije posreduje drugi napravi. Druga naprava lahko nato zavzame pozicijo na (skoraj) istih koordinatah, iz horizontalne razdalje med obema napravama pa se izračuna pravilna smer (angl. heading), da sledeni objekt pade v vidno polje druge naprave in lahko le-ta objekt tudi detektira in prevzame sledenje.

Lim et al. so v svojem članku [1] predstavili metodo za rekonstrukcijo trajektorij kamere in sledene osebe v 3D prostoru. Njihov pristop temelji na značilnih točkah (Harrisov detektor kotov [16] in deskriptor BRIEF [17]), ki jih zaznavajo v vsaki sliki, nato pa iz njihovih ujemanj med zaporednimi slikami ocenijo pozicijo kamere. Na tak način dobijo trajektorijo, ki jo opiše kvadrokopter, z uporabo detekcije pešcev (učenje klasifikatorja z algoritmom AdaBoost [18]) in vizualnega sledilnika Struck [15] pa pridobijo relativno pozo sledene osebe glede na kamero. Poleg tega z uporabo predpostavke o višini osebe (podobno kot [3]) in detekcijo ravnine tal (angl. ground plane) ocenijo tudi absolutno skalo pridobljenih meritev.

Trenutno so različni pristopi k opisanemu problemu precej odvisni od izbrane platforme in nekaterih močnih predpostavk. Platforme se med seboj precej razlikujejo po konfiguraciji kvadrokopterja in kamere (statične/premične kamere, morebitna stabilizacija ipd.), glavna predpostavka pa je ponavadi omejitev na predmete, ki se gibljejo po ravnini oz. omejitev na sledenje pešcev, kar omogoči oceno absolutne razdalje do zasledovanega objekta. Eksperimenti se večinoma izvajajo v razmeroma enostavnih okoljih, ki so v večjem delu statična in vsebujejo le zanemarljive odmike po višini (stopnice, klančine ipd.). Prav tako nobeden od omenjenih člankov ne uporablja dinamične kamere za sledenje objektom v primerih, ko bi bilo zaradi večje odzivnosti ali stabilnosti kvadrokopterja to bolj smiselno.

1.3 Prispevki

Prispevek magistrskega dela je razvoj sistema, ki kvadrokopterju omogoča robustno zasledovanje objekta. V ta namen naslovimo problem inicializacije sledilnika, robustne lokalizacije objekta in uporabo rezultata lokalizacije

v modulu za vodenje kamere in modulu za vodenje kvadrokopterja. Tak sistem uporabniku omogoča izbor poljubnega objekta v videu, pretočenem s kvadrokopterja. Kvadrokopter temu objektu nato sledi po prostoru. Po želji lahko uporabnik nastavi tudi druge parametre, kot sta recimo želeni kot med kvadrokopterjem in objektom ter relativna razdalja med njima. Z nastavitvijo želenega kota med kamero in objektom hkrati dosežemo možnost sledenja po neravnih površinah in zmanjšamo občutek vsiljivosti za sledeno osebo. Sistem nato na vhodnih slikah zažene kratkoročni sledilni algoritem, iz njegovih izhodnih podatkov generira vrednosti za vodenje kvadrokopterja in kamere, hkrati pa kontrolne podatke v povratni zanki posreduje tudi sledilniku in s tem skuša minimizirati potencialne težave, ki bi jih le-ta lahko imel zaradi hitrih premikov v sliki (sledenje hitrejših objektov, hiter dvig ob vzletu). Ob tem predlagamo še nekatere rešitve za praktične probleme, ki nastanejo pri delu z dinamičnim sistemom kot je kvadrokopter (problem ročne inicializacije sledilnika, odpovedi sledilnika ob vzletu in pristanku). Predstavimo tudi eksperimente, ki preizkusijo sposobnosti našega sistema, česar se zaradi visoke težavnosti drugi avtorji na področju izogibajo.

1.4 Struktura naloge

V Poglavju 2 je opisano teoretično ozadje metod in konceptov, ki so uporabljeni v sistemu. Ti obsegajo več področij, ki niso direktno povezana, a pomembna za razumevanje delovanja opisanega sistema. Obsegajo osnove kamer, robotike, fizikalnih principov kvadrokopterjev, kontrolne teorije in vizualnih sledilnikov.
Poglavje 3 opisuje naš pristop k problemu, kjer predstavi glavna dela našega sistema (regulator kvadrokopterja in regulator kamere) ter njuno delovanje.
V Poglavju 4 so opisane konkretne lastnosti strojne opreme, ki smo jo uporabili pri implementaciji in programskega ogrodja ROS.
Poglavje 5 vsebuje opise eksperimentov, ki smo jih izvedli za preizkus delovanja našega sistema in za kvantitativno evalvacijo posameznih delov sistema.

Eksperimenti so razdeljeni na tiste, ki so namenjeni regulatorju kamere, tiste za vodenje kvadrokopterja in eksperimente, ki preizkusijo delovanje celotnega sistema (oz. pokažejo, da naša rešitev omogoči nekatere nove možnosti uporabe).

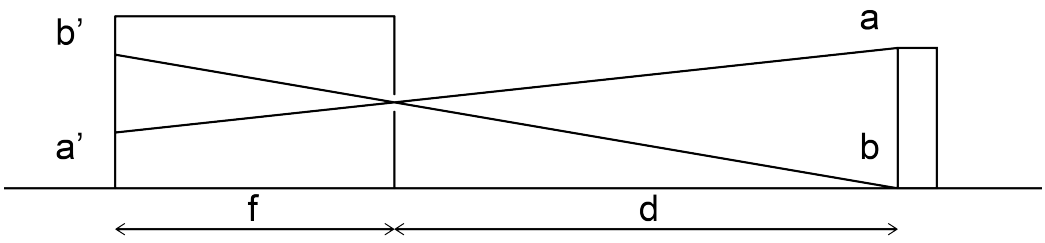
Poglavje 2

Teoretično ozadje

V tem poglavju bomo predstavili teoretično podlago, ki je potrebna za uspešno reševanje našega problema. Začeli bomo s predstavitvijo glavnih sestavnih delov mobilnega robota in opisali osnove kamer, mobilnih robotov in kvadrokopterjev. Nato bomo opisali še glavne elemente naše rešitve, torej geometrični odnos med kamero in objektom, teorijo regulatorjev in vizualne sledilnike.

2.1 Kamera

Kamera je glavni senzor, ki se uporablja v računalniškem vidu. Deluje na principu projekcije tridimenzionalnega prostora na slikovno ravnino. Za nastanek slike morajo svetlobni žarki potovati skozi majhno odprtino oziroma lečo, ki jih zbere in obrnjeno sliko projicira na slikovno ravnino. Prvi poskusi takih projekcij so bili narejeni s *camero obscuro*. To je zaprt prostor, ki ima v eni od sten majhno odprtino, skozi katero iz zunanjosti potujejo svetlobni žarki in ustvarijo sliko na nasprotni steni, kot je prikazano na Sliki 2.1. Ker mora biti za nastanek slike odprtina razmeroma majhna, so tako pridobljene slike precej temne, zaradi česar se v praksi uporabljajo leče, ki prepuščajo več svetlobe. V računalniškem vidu se zaradi enostavnosti večinoma uporablja model kamere z luknjično odprtino (angl. pinhole camera model). To je

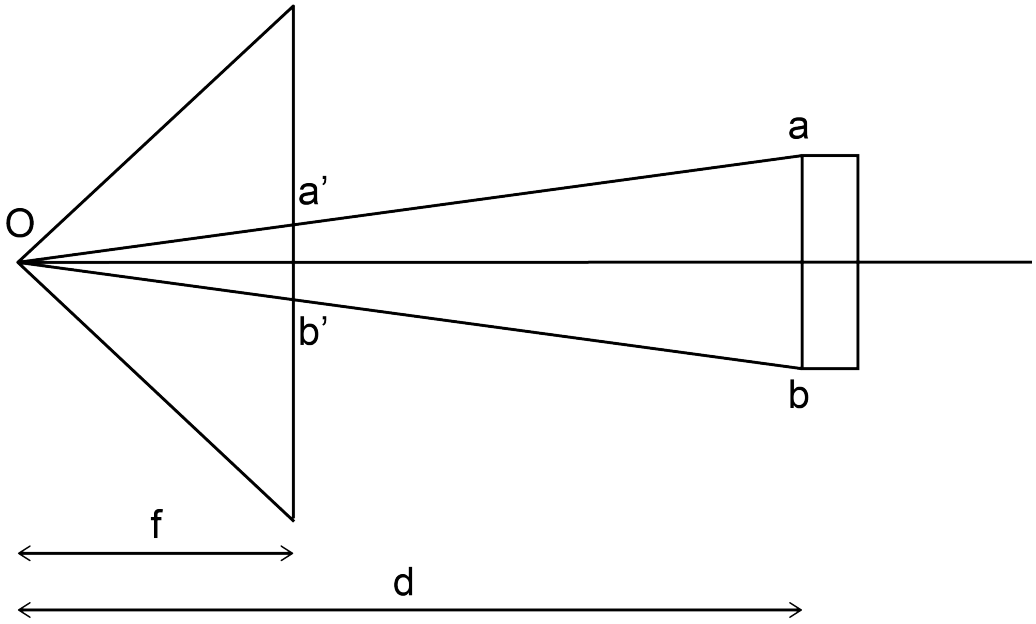


Slika 2.1: Prikaz delovanja *camere obscure*. Točki a in b sta najvišja in najnižja točka na objektu, točki a' in b' pa njuni sliki. Razdalja med kamero in objektom je označena z d, goriščna razdalja pa s f.

geometrijski model, ki nam služi za lažje razumevanje nastanka slike in kompleksnejših odnosov med večimi slikam in/ali kamerami (opazovanje istega objekta z večih zornih kotov ipd.).

2.1.1 Kamera z luknjično odprtino

Poenostavljen model kamere je sestavljen iz luknjice (angl. pinhole), ki ji rečemo tudi optični center, in slikovne ravnine. V realni kameri bi luknjica (ali leča) morala ležati med slikovno ravnino in opazovanim objektom, v tem teoretičnem modelu pa lahko zaradi enostavnosti kot slikovno ravnino obravnavamo kar ravnino, ki leži med luknjico in objektom, kot to prikazuje Slika 2.2. Razdalji med luknjico in slikovno ravnino rečemo goriščna razdalja, premici, ki povezuje luknjično odprtino O in center slikovne ravnine pa optična os (angl. optical axis). Centru slikovne ravnine drugače rečemo tudi principelna točka (angl. principal point). Slika točke $\mathbf{p} = [u, v, w]^\intercal$ v tridimenzionalnem prostoru nastane tako, da poiščemo točko, kjer premica, ki povezuje točko $\mathbf{p}$ z optičnim centrom, seka slikovno ravnino. V tej točki nastane $\tilde{\mathbf{p}}$, ki je slika točke $\mathbf{p}$. Točka $\tilde{\mathbf{p}}$ ima koordinati p_x, p_y definirani v koordinatnem sistemu slikovne ravnine, katerega izhodišče leži na optični osi.



Slika 2.2: Kamera z luknjično odprtino. Oznake so enake tistim na Sliki 2.1.

Ta postopek nastanka slike imenujemo perspektivna projekcija.

Z uporabo lastnosti podobnih trikotnikov lahko ugotovimo, da sta koordinati točke $\tilde{p}$ enaki

$$p_x = \frac{fu}{w},$$
$$p_y = \frac{fv}{w}, \tag{2.1}$$

kjer je f goriščna razdalja (c.f. Slika 2.8, kjer je prikazan enak princip). Zaradi enostavnejšega računanja ponavadi delamo s homogenimi koordinatami, kar pomeni, da koordinatam točk v tridimenzionalnem prostoru dodamo četrto koordinato, ki je vedno enaka 1. To nam omogoči predstavitev točk v neskončnosti, hkrati pa projekcijo poljubne točke v prostoru na sli-

kovno ravnino poenostavi v množenje z matriko **K**:

$$\tilde{\mathbf{p}} = \mathbf{K} \cdot \mathbf{p} \tag{2.2}$$

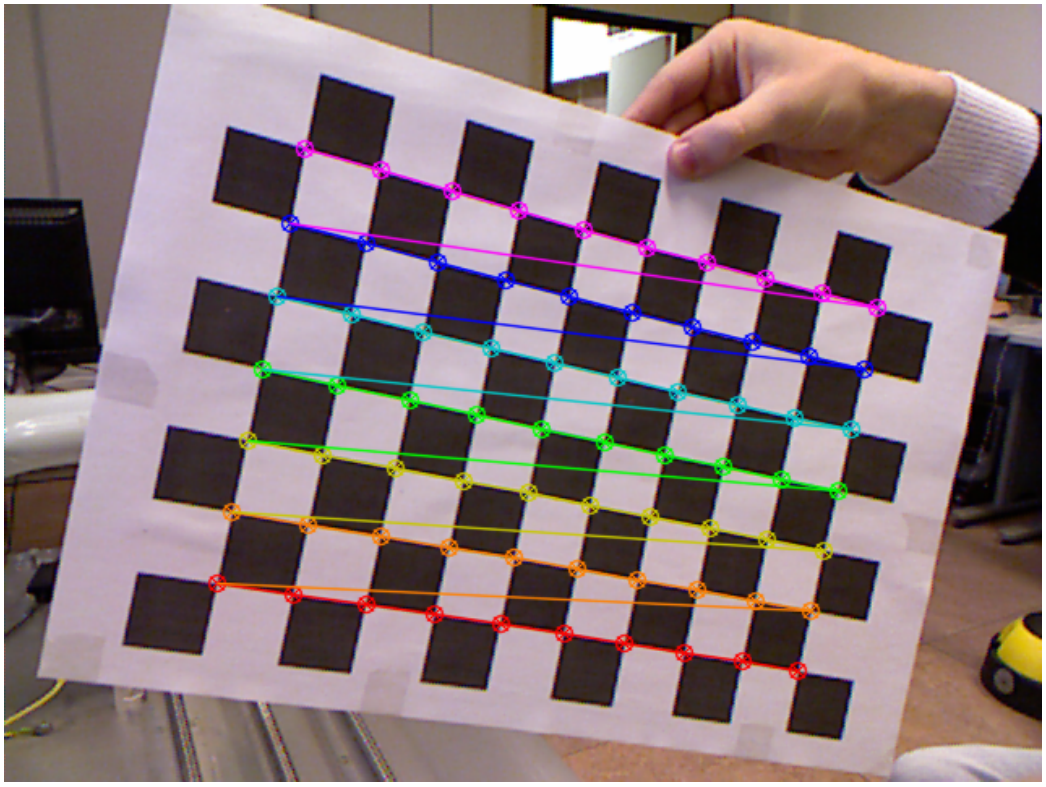
Matrika **K**, ki jo uporabimo za perspektivno projekcijo, predstavlja lastnosti kamere in jo imenujemo intrinzična matrika. Njena sestava je opisana v naslednjem poglavju.

2.1.2 Kalibracija kamere

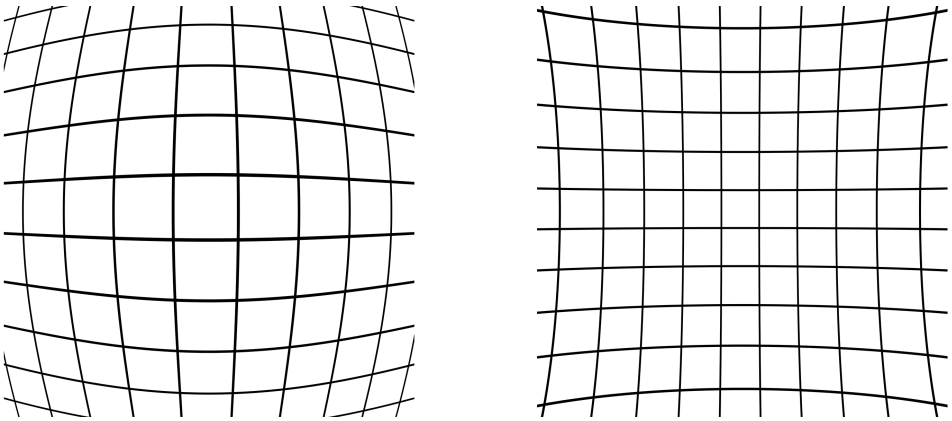
Model z luknjično odprtino predstavlja idealiziran pogled na nastanek slike v kameri, v resnici pa moramo biti pri zajemanju digitalnih slik pozorni tudi na fizične omejitve uporabljenih senzorjev. To pomeni, da moramo upoštevati fizično velikost svetlostnih senzorjev v kameri, možnost neenakih velikosti stranic posameznih pikslov ali njihovo izkrivljenost (angl. skew) in napako, ki jo zaradi svoje oblike povzroči leča (distorzija). Te dejavnike pri izdelavi kompleksnejših modelov kamere imenujemo intrinzični parametri in so bistveni pri zajemu in obdelavi slik, ki so primerne za obdelavo z metodami računalniškega vida.

Ker pa kamere večine podatkov o teh lastnostih nimajo zapisanih v specifikacijah, jih je treba oceniti empirično. To izvedemo s postopkom, imenovanim kalibracija kamere. Pri njem s kamero naredimo več slik vzorca z znanimi lastnostmi, po navadi je to vzorec šahovnica ali mreža pik, kot je prikazano na Sliki 2.3. V slikah nato detektiramo oglišča posameznih kvadratov oziroma centre pik. Iz primerjave med izmerjenimi in znanimi pozicijami točk nastavimo sistem linearnih enačb, ki ga nato lahko rešimo recimo z metodo najmanjših kvadratov.

Kot rezultat dobimo ocenjene vrednosti intrinzičnih parametrov, ki jih lahko uporabimo za rektifikacijo zajetih slik. Med najopaznejšimi napakami, ki jih povzroči uporaba leč v kamerah, je radialna distorzija. Zaradi njenih nelinearnih lastnosti se je sicer ne da vključiti v intrinzično matriko, marveč se rektifikacija izvaja v kasnejšem procesiranju slik. Distorzija nastane zaradi uporabe leče, ki svetlobnih žarkov ne ukloni popolnoma uniformno, ampak



Slika 2.3: Vzorec šahovnice za kalibracijo kamere.



Slika 2.4: Primera radialne distorzije. Slika a) prikazuje barrel, slika b) pa pincushion distorzijo.

povzroči, da se slike objektov, ki so bolj oddaljeni od središča slike, upognejo. Ravne črte ob robu slike zaradi tega ne izgledajo več ravne. Najpogostejša tipa distorzije sta t.i. sodček in blazinica (angl. barrel in pincushion), ki sta prikazana na Sliki 2.4. Parametri radialne distorzije so shranjeni v obliki koeficientov kvadratnih funkcij, s katerimi modeliramo nastalo napako. Preostale intrinzične parametre kamere lahko zapišemo v intrinzično matriko:

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

kjer sta f_x in f_y goriščni razdalji za osi x in y, x_0 in y_0 odmika izhodiščne točke slikovne ravnine, s pa predstavlja faktor izkrivljenosti. Ker želimo mrežo pikslov v digitalnih slikah definirati le za pozitivna števila, moramo izhodišče kamere premakniti z ustreznim odmikom. To predstavljata vrednosti x_0 in y_0, ki sta večinoma enaki polovici višine in širine celotne slike in premakneta izhodišče slikovne ravnine v zgornji levi kot slike.

2.1.3 Model realne kamere

Za analizo vpliva premikov kamere na sliko oz. za kakršno koli resnejše računanje s kamerami, jih moramo umestiti v isti koordinatni sistem kot točke, ki jih s kamerami opazujemo. To nam omogočijo t.i. ekstrinzični parametri kamere, ki opisujejo njeno trenutno pozicijo in usmeritev v prostoru. Ekstrinzične parametre zapišemo v matriko

$$\mathbf{E} = \left[\begin{array}{ccc|c} & \mathbf{R} & & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right], \tag{2.4}$$

kjer je $\mathbf{R}$ 3×3 matrika rotacije kamere, vektor $[t_x\ t_y\ t_z]^\intercal$ pa pozicija kamere v svetovnem koordinatnem sistemu. Sliko poljubne točke $\mathbf{p}$ v 3D prostoru

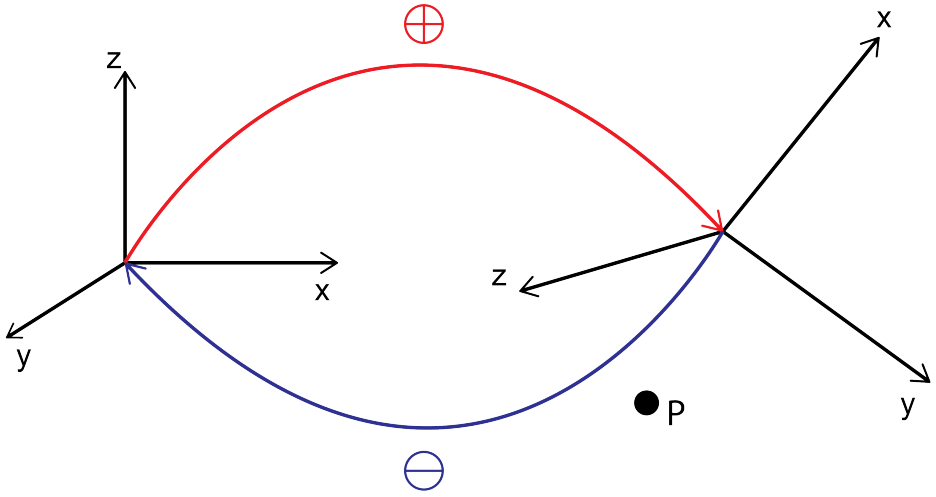
tako dobimo z enačbo

$$\tilde{\mathbf{p}} = \mathbf{K} \cdot \mathbf{E} \cdot \mathbf{p}. \tag{2.5}$$

2.2 Koordinatni sistemi in Lieva algebra

Vsakršno delo z roboti zahteva najprej dobro definiran prostor in način določanja pozicije in orientacije robota oz. njegovih sestavnih delov znotraj njega. Za osnovo nam služi kartezični koordinatni sistem v treh dimenzijah, glede na njegovo izhodišče pa lahko definiramo pozicijo (angl. position) drugih objektov, ki jih obravnavamo. Pozicija točke v prostoru v odvisnosti od koordinatnega izhodišča je ponavadi definirana kot vektor odmikov na posamezni osi koordinatnega sistema. Izraz poza (angl. pose) pa označuje pozicijo in hkrati orientacijo objekta znotraj prostora. Ker so roboti sestavljeni iz večih delov, se za njihovo obravnavo uporablja koncept togih objektov in togih transformacij. To pomeni, da vsak objekt definiramo kot samostojen koordinatni sistem, ki ga imenujemo koordinatni okvir (angl. coordinate frame). To nam omogoči, da obravnavamo premik vseh delov objekta naenkrat z izračunom premika koordinatnega okvirja glede na koordinatni sistem prostora. Pozicijo posameznega dela lahko nato dobimo z ustrezno transformacijo koordinatnega okvirja objekta in se tako izognemu računanju premika vsakega dela posebej.

Opis togih transformacij in Lievih grup v nadaljevanju je povzet po tehničnem poročilu Joseja-Luisa Blanca [19]. Rotacija v 3D prostoru je izvedena z uporabo matrik, ki pripadajo algebraični grupi SO(3) (angl. special orthogonal group). To so ortogonalne matrike velikosti 3×3, katerih determinanta je enaka 1 in predstavljajo prave izometrične transformacije. Ker so izometrične, ohranjajo razdalje med katerim koli parom točk, pozitivna determinanta pa pove, da grupa ne vsebuje zrcaljenja. Rotacijo točke $\mathbf{a} = [x_1, y_1, z_1]^\intercal$ lahko tako predstavimo kot množenje z matriko R, ki pripada tej grupi: $\mathbf{a'} = [x_2, y_2, z_2]^\intercal = \mathbf{Ra}$. Če želimo upoštevati tudi translacijo, moramo uporabiti homogene koordinate, ki jih dodamo rotacijski matriki in



Slika 2.5: Sprememba koordinatnega sistema.

dobimo matriko velikosti 4×4 oblike

$$\left[\begin{array}{ccc|c} & \mathrm{R} & & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right], \tag{2.6}$$

kjer je matrika $\mathbf{R} \in SO(3)$, vektor $[t_x, t_y, t_z]^\intercal$ pa predstavlja translacijo. Opazimo lahko, da je matrika identična matriki $\mathbf{E}$ iz Poglavja 2.1.3, ker obe predstavljata preslikavo med koordinatnima sistemoma v treh dimenzijah. Take matrike sestavljajo grupo, imenovano SE(3) (angl. special Euclidean). Z njimi lahko opišemo poljuben tog premik (in zasuk) v tridimenzionalnem prostoru. Tako opisane poze imenujemo tudi 6D poze, saj imajo šest prostostnih stopenj (tri za translacijo in tri za rotacijo).

Za delo s pozami moramo nad njimi definirati dve operaciji, kompozicijo $\oplus$ in inverzno kompozicijo $\ominus$. Ti dve operaciji predstavljata prehod med koordinatnimi sistemi. Če znotraj globalnega koordinatnega okvirja defi-

niramo nov koordinatni sistem P, je ta odvisen od koordinatnega sistema sveta (ki je definiran kot identiteta). Če želimo izračunati koordinate točke $\mathbf{a}$ v odvisnosti od koordinatnega sistema P, nam to predstavlja kompozicija $\mathbf{a} \oplus P$. To nam koristi, če recimo poznamo globalno pozicijo neke točke, ki jo opazujemo s kamero in želimo ugotoviti njeno pozicijo relativno na kamero. Prehode med koordinatnimi sistemi prikazuje Slika 2.5. Ker so matrike v SO(3) obrnljive, za vsak togi premik obstaja tudi njegov obratni premik:

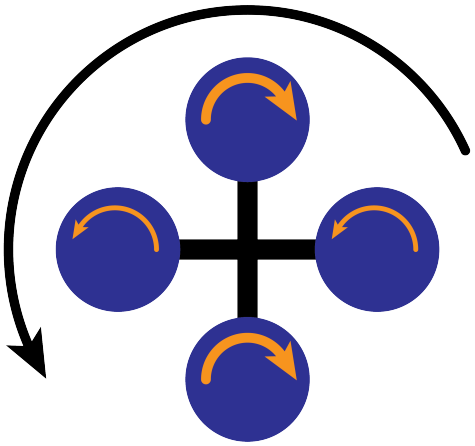
$$\begin{aligned} \mathbf{a}' &\equiv P \oplus \mathbf{a}, \\ \mathbf{a} &\equiv \mathbf{a}' \ominus P, \end{aligned} \tag{2.7}$$

kjer sta $\mathbf{a}$ točka in P poza v globalnem koordinatnem sistemu, $\mathbf{a}'$ pa koordinate točke $\mathbf{a}$ glede na pozo P. Ker so tovrstne preslikave izvedene z matričnim množenjem, je treba paziti na to, da operaciji nista komutativni in je vrstni red zaporednih kompozicij pomemben.

2.3 Kvadrokopterji

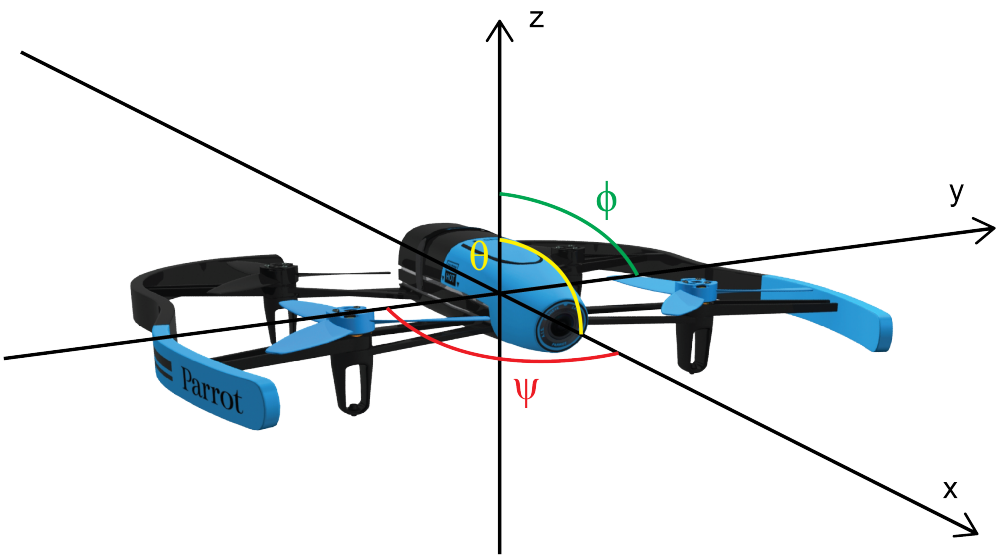
Kvadrokopterji (imenovani tudi kvadrotorji oziroma angl. UAV - Unmanned aerial vehicle) so brezpilotna letala s štirimi propelerji, pritrjenimi na križno ogrodje. Kvadrokopter leti tako, da z reguliranjem hitrosti motorjev ustvarja dovolj potiska (angl. thrust), ki negira silo gravitacije in mu omogoča, da obstane v zraku. Z manipulacijo relativnih hitrosti motorjev lahko regulator inducira premike v štirih smereh: po treh Eulerjevih kotih ϑ, φ in ψ (angl. pitch, roll, yaw) ter po višini. Kvadrokopter svojih premikov po oseh x in y ne izvaja direktno, marveč so taki premiki posledica spremembe kotov ϑ in φ. Konkretno, kvadrokopter se lahko premakne v smeri x le tako, da s povečanjem hitrosti zadnjih dveh motorjev poveča kot ϑ (torej se nagne naprej), kar spremeni smer vektorja potiska in njegovemu gibanju doda horizontalno komponento. Ker se pri tem nekaj sile preusmeri v horizontalni smeri, to navadno pomeni, da kvadrokopter ob horizontalnih

premiki izgubi nekoliko višine. Povsem iste značilnosti držijo za koordinato y, kot φ in stranske motorje. Zasuk okrog osi z (oz. spreminjanje kota ψ) je nekoliko zahtevnejši. Namreč, če bi se vsi propelerji vrteli v isto smer, bi to zaradi zakona o ohranitvi vrtilne količine povzročilo vrtenje celotnega sistema v nasprotno smer. Zaradi tega so kvadrokopterji sestavljeni tako, da imajo sosednji propelerji nasprotne smeri vrtenja, kar izniči ta efekt in omogoča stabilen let. Zasuk okrog osi z torej lahko dosežemo z manipulacijo tega ravnovesja tako, da povečamo hitrost motorjev, ki se vrtita v nasprotno smer želenega zasuka, kot je prikazano na Sliki 2.6. Višino regulator kvadrokopterja regulira direktno z enakomernim spreminjanjem hitrosti vseh motorjev, saj mora biti za stabilno lebdenje (angl. hover) vsota vseh sil nanj enaka nič. Vzgon, ki ga povzroča vrtenje propelerjev, mora biti za ohranjanje višine enak gravitacijski sili, ki deluje na sistem. Če je ta vzgon večji od gravitacijske sile se kvadrokopter dvigne, sicer se spusti.



Slika 2.6: Shema zasuka okrog osi z.

Za lažje razumevanje naslednjih poglavij bomo na tej točki definirali koordinatni sistem kvadrokopterja in ustrezne kote, ki se jih bomo držali do konca dela. Prikazuje jih Slika 2.7.

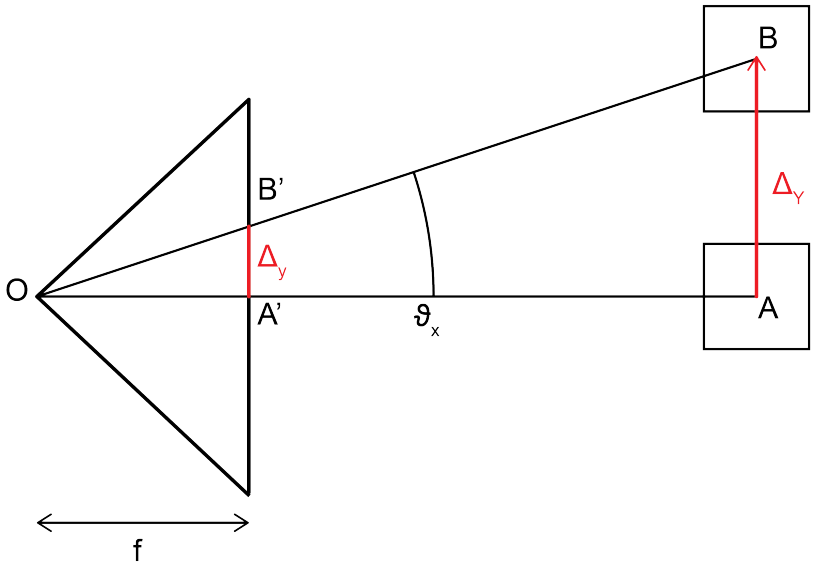


Slika 2.7: Koordinatni sistem kvadrokopterja.

2.4 Geometrija med kamero in objektom

Močan poudarek tega dela je na eksplicitni formulaciji geometričnega odnosa med kamero in sledenim objektom, zato bomo v tem podpoglavju predstavili nekatere lastnosti, ki jih lahko pridobimo z analizo projekcije objekta na slikovno ravnino ob premikih po prostoru. Ker bomo delali s premično kamero, velja najprej opisati izračun kota, za katerega se je objekt premaknil med dvema zaporednima slikama, kar nam bo kasneje omogočilo implementacijo regulatorja, ki bo skrbel, da je objekt vedno v centru kamere. V tem razdelku bomo za označitev osi v tridimenzionalnem (realnem) prostoru uporabljali oznake X, Y in Z, koordinatni sistem slikovne ravnine pa bomo označevali z oznakama x in y. V tem primeru sta osi X in Y realnega prostora vzporedni slikovni ravnini, os Z pa je nanjo pravokotna.

Če predpostavimo, da kamera gleda naravnost naprej, objekt leži zno-



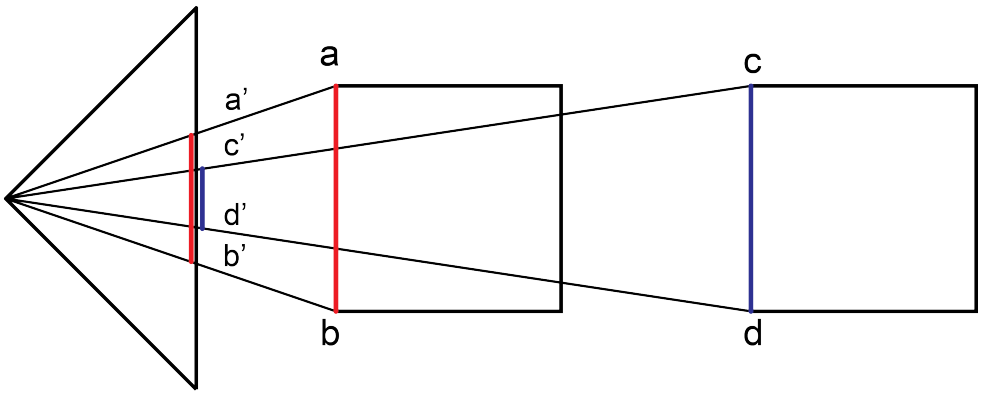
Slika 2.8: Premik objekta v sliki.

traj njenega vidnega polja, nato pa se premakne po osi x ali y (glede na slikovno ravnino), lahko izračunamo vektor premika njegove slike, kar prikazuje Slika 2.8. Opazimo lahko dva trikotnika, ki ju določa ta premik: ΔOAB in $\Delta OA'B'$, ki sta si podobna. Če poznamo goriščno razdaljo kamere lahko izračunamo kot, za katerega se je objekt premaknil po osi Y glede na kamero. Kot ϑ_y med centrom objekta in optično osjo kamere je tako enak

$$\vartheta_y = tan^{-1}\Big(\frac{\Delta_y}{f}\Big), \tag{2.8}$$

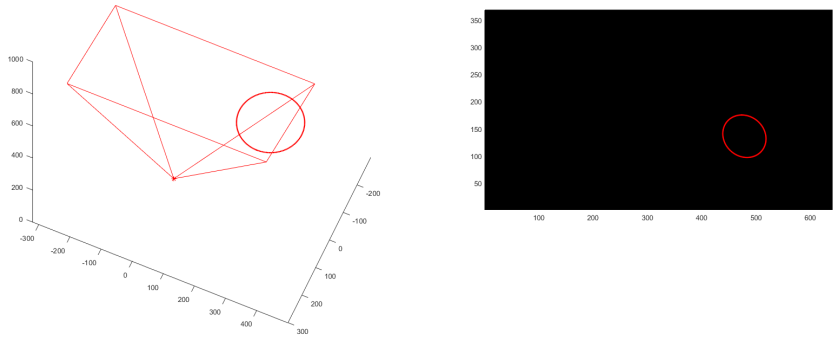
kjer je Δ_y premik slike objekta po osi y v pikslih in f goriščna razdalja kamere. Isto velja za premik po osi x in njemu ustrezen kot.

Za pravilno izpeljavo regulatorja kvadrokopterja moramo ugotoviti, kako se premiki v tridimenzionalnem prostoru odslikujejo na slikovni ravnini. Take premike bomo označevali z Δ_X, Δ_Y in Δ_Z, njim ustrezne spremembe v sliki kamere pa z Δ_x, Δ_y in Δ_d, kjer je Δ_d sprememba velikosti objekta v sliki (trenutno jo obravnavamo kot abstraktno mero velikosti slike v objektu, konkre-



Slika 2.9: Sprememba skale.

tne formule predstavimo v Poglavju 3). Iz lastnosti perspektivne projekcije zlahka opazimo, da so vektorji premikov slike objekta le vektorji premikov dejanskega objekta v prostoru, projicirani na slikovno ravnino. V primeru, da kamera in objekt ležita v isti ravnini in kamera gleda v smeri osi Z, je enostavno ugotoviti, da Δ_x ustreza Δ_X, Δ_y ustreza Δ_Y in Δ_d ustreza Δ_Z. Prikaz spremembe velikosti objekta je na Sliki 2.9. Če pa kamera in objekt ne ležita na isti (horizontalni) ravnini, moramo upoštevati tudi kot med njima. Obravnavali bomo le primer, ko objekt leži nižje od kamere, torej ima manjšo koordinato Y. Letenje kvadrokopterja niže od objekta je namreč smiselno le v redkih primerih, poleg tega pa lahko letenje blizu tlom močno negativno vpliva na stabilnost sistema. V tem primeru Δ_x še vedno ustreza Δ_X, Δ_y pa ima sedaj dva različna vira: če opazimo premik po osi y, je to lahko posledica premika objekta po osi Y ali po osi Z. V tem primeru premik objekta po koordinati Y v pozitivni smeri povzroči premik slike v isti smeri kot povečanje koordinate Z. Hitrost premika slike objekta po osi y je odvisna od kota med kamero in objektom.



Slika 2.10: Simulacija v okolju MATLAB.

2.4.1 Simulacija kamere in objekta

Za enostavnejše razumevanje geometrijskih razmerij med kamero in objektom smo v okolju MATLAB tudi nastavili in izvedli nekaj simulacij. Najprej smo morali nastaviti okolje, zato smo definirali kamero, katere parametri ustrezajo kameri na konkretnem kvadrokopterju (opisano v Poglavju 4.1), jo umestili v prostor in na njeno slikovno ravnino projicirali različne objekte, kar prikazuje Slika 2.10.

Ena glavnih nalog je bila ugotoviti razmerje med kotom, ki ga opisujeta objekt in kamera ter spremembami pozicije in velikosti slike objekta glede na njegove premike. Za izpeljavo vpliva kota se moramo spomniti, da je premik slike objekta le projekcija vektorja njegovega premika na slikovno ravnino. To je očitno, če pomislimo, da v primeru, da objekt leži točno pred centrom kamere in se od nje oddaljuje, njegova slika miruje, ker je projekcija njegovega premika na slikovno ravnino točka (c.f. Slika 2.9).

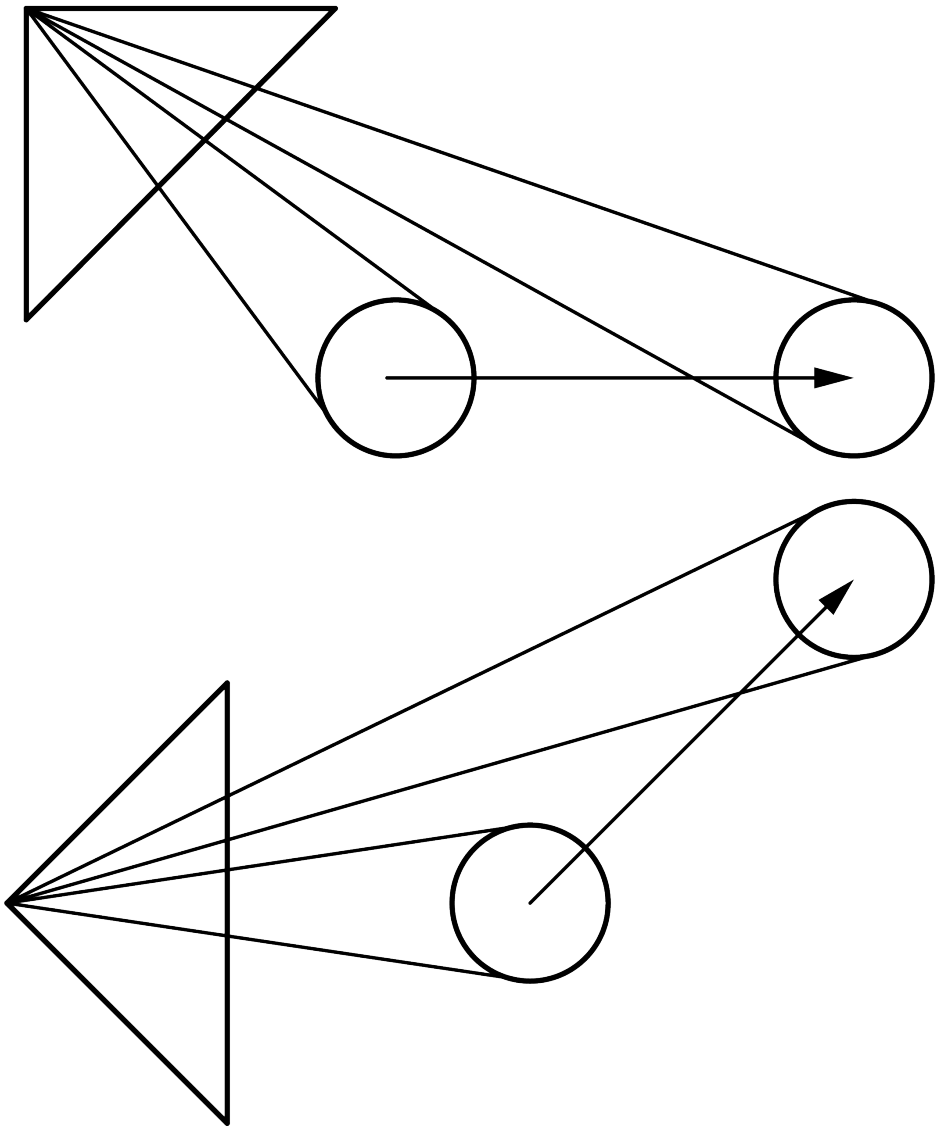
Za lažji razmislek o kompleksnejših premikih lahko najprej poenostavimo pogled na problem, kot prikazuje Slika 2.11. Vidimo lahko, da je premik objekta vzporedno s horizontalno ravnino, kadar ga kamera opazuje pod

α	Δ_d	Δ_y
-180	-1	0
-135	-0,7071	-0,7071
-90	0	-1
-45	0,7071	-0,7071
0	1	0
45	0,7071	0,7071
90	0	1
135	-0,7071	0,7071
180	-1	0

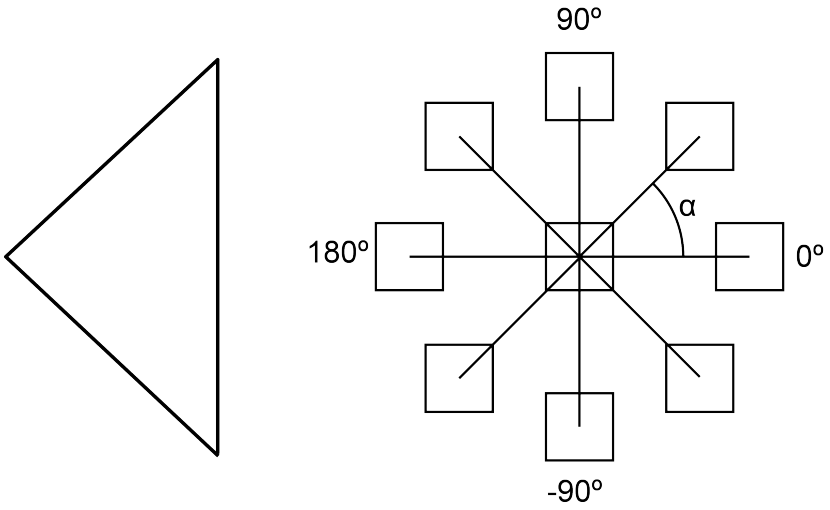
Tabela 2.1: Premiki objekta in ustrezne vizualne značilke.

nekim kotom (recimo 45°), ekvivalenten premiku objekta pod istim kotom glede na horizontalno ravnino, kadar je kamera usmerjena naravnost. Ker se mora naš sistem odzivati na dejanske premike objekta, ki so enaki ne glede na nagib kamere, bomo zato premike vedno obravnavali na enak način, razlika bo le v tem, kako se ti premiki odražajo v značilkah sledilnika.

Omejili se bomo le na premike objekta po vertikalni ravnini, torej tiste, ki v sliki inducirajo spremembo koordinate y in spremembo velikosti objekta. V primeru, ko kamera gleda direktno naravnost, lahko definiramo kot α, ki predstavlja odmik smeri premika objekta glede na optično os kamere. Če se objekt premakne naravnost stran od kamere, to opišemo kot premik z $\alpha = 0$, kot je prikazano na Sliki 2.12. Tako lahko opišemo premike objekta za kote na intervalu $[-180°, 180°]$. Iz razmerja obeh opaženih značilk lahko tako ocenimo kot, za katerega se je objekt premaknil v realnem svetu. Kadar je optična os kamere vzporedna s horizontalno ravnino, so za analizo najenostavnejši koti, ki so večkratniki $\frac{\pi}{2}$. Pri teh kotih je namreč (idealno) doprinos ene značilke enak 1 (oz. -1), doprinos druge pa je enak 0. Z analizo simulacije lahko zgradimo tabelo, ki prikazuje opažene vizualne značilke pri različnih vrednostih kota α. Enostavna trigonometrija nam nato pove, da se



Slika 2.11: Preslikava problema premika objekta pod kotom.



Slika 2.12: Premiki za kot α glede na optično os kamere.

vrednosti značilk spreminjata glede na kot α po formulah

$$\begin{aligned} \Delta_y &\propto \sin(\alpha), \\ \Delta_d &\propto \cos(\alpha). \end{aligned} \tag{2.9}$$

Naš cilj je ugotoviti, kako se spreminjajo odnosi med vidnimi značilkami in premiki objekta v realnem prostoru, kadar je kot med kamero in objektom, imenujmo ga γ, večji od 0, torej če kvadrokopter leti višje od sledenega objekta. Če privzamemo, da je kot γ enak 45°, lahko opazimo, da se spremenijo značilnice, ki jih zaznamo s kamero. Vidimo lahko, da se sprememba velikosti objekta brez spremembe koordinate y ne pojavi več pri $\alpha = 0°$, ampak pri $\alpha = -45°$. Enako se zgodi pri ostalih kotih, kot prikazujeta Slika 2.13 in Tabela 2.2. Enostavno je ugotoviti, da se enačbi (2.9) v tem

α	Δ_d	Δ_y
-180	-0,7071	-0,7071
-135	0	-1
-90	0,7071	-0,7071
-45	1	0
0	0,7071	0,7071
45	0	1
90	-0,7071	0,7071
135	-1	0
180	-0,7071	-0,7071

Tabela 2.2: Premiki objekta in ustrezne vizualne značilke za kot $\gamma = 45°$

primeru spremenita v

$$\Delta_y \propto \sin(\alpha + \gamma),$$
$$\Delta_d \propto \cos(\alpha + \gamma). \tag{2.10}$$

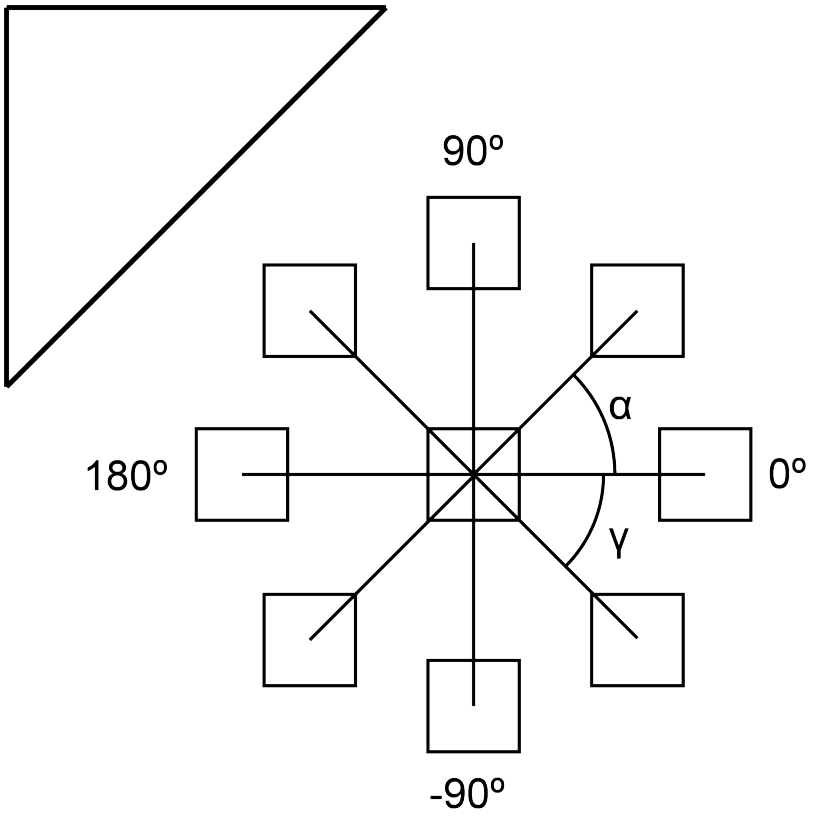
Ta uvid nam bo kasneje omogočil razvoj regulatorja, ki bo skušal oceniti premike objekta glede na kamero in jih negirati.

2.5 Osnove vodenja z uporabo slik

V našem sistemu želimo generirati vrednosti za nadzor kamere in kvadrokopterja izključno iz vizualnih značilk, zato potrebujemo ustrezen regulator, zatem pa še metodo preslikave vizualnih značilk v napako, ki jo ta regulator lahko popravi.

2.5.1 PID regulator

PID regulatorji (angl. proportional-integral-derivative) so v industriji med najpogosteje uporabljanimi kontrolerji s povratno zanko za regulacijo raznih



Slika 2.13: Premiki za kot α glede na optično os kamere pri nagibu kamere za 45°.

procesov. Ti regulator računajo napako kot razliko med trenutnim in ciljnim stanjem sistema. Za ta namen uporabljajo tri ločene vrednosti, ponavadi označene s K_p, K_i in K_d. To so koeficienti, ki upoštevajo trenutno stanje, pretekla stanja in skušajo napovedati prihodnja stanja sistema. Delovanje regulatorja prikazuje enačba

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}, \tag{2.11}$$

kjer je $u(t)$ t.i. kontrolna spremenljivka, s katero vodimo proces ob času t. Vrednost $u(t)$ nato služi kot vhod v sistem, ki ga skuša pripeljati v ciljno stanje. Na začetku delovanja moramo regulatorju nastaviti parametre (angl. gain) in ciljno stanje. Med delovanjem sistema regulator v vsakem časovnem koraku izračuna napako med trenutnim in ciljnim stanjem (lahko je to razlika med vrednostima, ali pa regulator upošteva še dodatne podatke), s katero najprej posodobi interno stanje in nato vrne ustrezno vrednost kontrolne spremenljivke.

Proporcionalni faktor skalira napako v trenutnem časovnem koraku, da ta na izhodu povzroči ustrezno spremembo. Velike vrednosti lahko povzročijo oscilacije ali nestabilnost sistema, saj se ta zelo močno odzove na spremembo napake, majhne vrednosti pa lahko slabo vplivajo na odzivnost sistema.

Integralni del skuša popraviti napako, ki se je nabrala v preteklih meritvah tako, da izračuna integral (oziroma vsoto zadnjih k napak za diskretne implementacije) in na ta način pospeši premik sistema proti ciljnemu stanju, če je napaka prisotna dalj časa.

Odvodni del pa se izračuna kot prvi odvod napake po času in tako skuša predvideti obnašanje sistema. S tem se ob večjih napakah poveča tudi hitrost premikov proti ciljnemu stanju, kar pripomore k odzivnosti sistema, a ob prevelikih vrednostih lahko povzroči prekoračitve in oscilacije okrog ciljnega stanja.

V praksi je integralni del implementiran z vsoto zadnjih n napak, odvodni del pa kot obtežena vsota preteklih diferenc napake (torej razlik med trenutno napako in napako v prejšnjem časovnem koraku), kot prikazujeta enačbi

(2.12).

$$\int_0^t e(\tau)d\tau \approx \sum_{i=1}^{k} e(t_i)\Delta t,$$
$$\frac{de(t_k)}{dt} \approx \frac{e(t_k) - e(t_{k-1})}{\Delta t}. \tag{2.12}$$

V teh enačbah Δt predstavlja časovni korak, $e(t_k)$ pa napako regulatorja pri času t_k.

2.5.2 Vizualni servo nadzor z uporabo slik

Vodenje robotov na podlagi informacij, pridobljenih s kamero, se izvaja s sistemi, imenovanimi IBVS (angl. image-based visual servo control). Taki sistemi se med sabo razlikujejo po razmerju kamere (oz. kamer) in robota. V industriji se lahko uporabljajo fiksne kamere, ki opazujejo delovno površino, za naš problem pa je bolj primeren pristop, kjer sta premikajoči se del robota (vrh robotske roke, center kvadrokopterja) in kamera v isti točki (angl. eye-in-hand). V tem primeru obstaja znano (in večinoma konstantno) razmerje med robotom in kamero. V literaturi se taki sistemi večinoma pojavljajo v kontekstu robotskih rok, a princip brez težav ustreza tudi mobilnim robotom. V jedru vizualnega servo vodenja je minimizacija funkcije napake, ki jo avtorji v zelo splošni obliki predstavijo v članku [20]:

$$\mathbf{e}(t) = s(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*, \tag{2.13}$$

kjer je $\mathbf{m}(t)$ množica meritev v času, $s(\cdot, \cdot)$ je funkcija, ki aplicira parametre $\mathbf{a}$ (to so naprimer intrinzični parametri kamere) na vektor $\mathbf{m}(t)$, kar proizvede vektor vizualnih značilk. Vektor $\mathbf{s}^*$ pa so želene vrednosti značilk. Rezultat $\mathbf{e}(t)$ je torej vrednost napake za trenutni časovni korak, kar lahko uporabimo za vhod v PID regulator. Ker v našem sistemu uporabljamo več PID regulatorjev in računamo napako za vsakega posebej, imajo v naši formulaciji vektorji v enačbi (2.13) en sam element. Zaradi tega lahko napako $\mathbf{e}(t)$ direktno vstavimo v enačbo (2.11) in pridobimo ustrezno kontrolno vrednost,

ki bo skušala napako minimizirati. Ker sta pozicija kamere in posledično pozicija slike opazovanega objekta odvisni od pozicije robota, lahko z manipulacijo robotove pozicije v prostoru minimiziramo odstopanje meritev od želenega stanja.

2.6 Sledilni algoritmi

Sledilni algoritmi so programi, ki jim podamo zaporedje slik in območje za inicializacijo (po navadi je to pravokotnik, ki označuje objekt, ki mu želimo slediti). Zgodnejši sledilniki so delovali na podlagi vizualne podobnosti [21], sčasoma pa so avtorji v svoje algoritme za izboljšanje robustnosti vpeljali veliko kompleksnejših pristopov. Mednje sodijo predstavitve objekta s histogrami [22], sledenje z večimi deli [23], z diskriminativnimi klasifikatorji [24], rojem delcev [25] ipd. Novi pristopi so bili nujni zaradi velikih sprememb, ki so jim lahko objekti podvrženi med sledenjem. To so v glavnem netoge deformacije objektov, spremembe v izgledu objekta, delna ali popolna zakrivanja in spremembe osvetlitve.

Sledilne algoritme lahko razdelimo na kratkoročne in dolgoročne. Kratkoročni delujejo od inicializacije do izgube objekta, torej morajo vključevati mehanizme za detekcijo odpovedi, fokusirajo pa se na prilagajanje spremembam izgleda objekta. Dolgoročni sledilniki pa poleg metod za sledenje vključujejo tudi mehanizme za ponovno detekcijo ob potencialni odpovedi sledilnika (tipičen primer takega sledilnika je sledilnik TLD [6]).

Sledilni algoritem, ki ga bomo uporabili pri implementaciji sistema, temelji na korelacijskih filtrih [8, 26], Lukežič [27] pa je v svojem magistrskem delu isti koncept uporabil za sledenje deformabilnih objektov. Ta algoritem in njegova izboljšana različica [28] sta dosegla odlične rezultate na težavnih eksperimentih ter delujeta v realnem času. Zaradi tega ocenjujemo, da sta dovolj primerna za uporabo na kvadrokopterju.

2.6.1 Korelacijski filtri

Ena najstarejših metod za detekcijo znanih vzorcev v slikah je primerjava s predlogo (angl. template matching). To pomeni, da nad sliko izvedemo križno korelacijo s predlogo in pogledamo, kje je odziv najmočnejši. Pri vizualno zelo podobnih vzorcih to proizvede močan odziv, za manj jasne primere pa so rezultati lahko nepredvidljivi. Eden od načinov, kako narediti take primerjave robustnejše, so korelacijski filtri [29]. To so predloge, naučene iz množice učnih slik, ki skušajo modelirati različne izglede želenega objekta oz. vzorca. Za pridobitev takih filtrov je bilo v preteklosti predstavljenih več metod, npr. [30], vse pa so zahtevale predčasno učenje filtrov. V članku [26] so Bolme et al. prvi predstavili metodo za kreiranje korelacijskih filtrov MOSSE (Minimum Output Sum of Squared Error), ki za inicializacijo potrebujejo eno samo sliko, in tako odprli podkategorijo sledilnikov s korelacijskimi filtri.

Sledilniki, ki uporabljajo korelacijske filtre, modelirajo izgled sledenega objekta s filtri, ki so naučeni na slikovnih primerih. Ob inicializaciji se kreirajo filtri iz označene regije v sliki, ki je centrirana na sledeni objekt, nato pa se sledenje in treniranje filtrov izvajata vzporedno. V vsaki naslednji sliki se objekt lokalizira s korelacijo filtrov in manjše preiskovane regije znotraj slike, nato pa se model izgleda objekta posodobi glede na trenutni izgled objekta. Zaradi zaželene čim višje hitrosti se korelacija izvede v Fourierjevem prostoru, kjer se korelacija poenostavi v produkt po elementih. Odziv tako predstavlja enačba

$$\mathbf{G} = \mathbf{F} \odot \mathbf{H}^H, \tag{2.14}$$

kjer je F Fourierjeva transformacija vhodne slike f, H pa Fourierjeva transformacija filtra $\mathbf{h}$ (simbol $(\cdot)^H$ tukaj predstavlja kompleksno konjugacijo, simbol $\odot$ pa produkt po elementih). Za izračun filtra $\mathbf{h}$, ki nam bo za določeno sliko proizvedel idealen rezultat, moramo rešiti enačbo (2.14), za to pa moramo najprej definirati idealni odziv. Tipično je to kompaktna dvodimenzionalna Gaussova funkcija z vrhom v centru slike in majhno standardno deviacijo.

Avtorji za izračuna filtra MOSSE minimizirajo vsoto kvadratne napake (angl. sum of squared error - SSE), kar zapišejo kot naslednjo enačbo:

$$\min_{H^H} \sum_i |\mathbf{F}_i \odot \mathbf{H}^H - \mathbf{G}_i|^2,$$

kjer gre indeks i po elementih slike. Filter, ki proizvede idealen odziv, se na ta način izračuna v vsakem od korakov, sledilniki pa pri posodabljanju filtra uporabljajo parameter α za nastavitev hitrosti učenja oz. pozabljanja. Posodobitev se izvede z linearno kombinacijo filtra v prejšnjem časovnem koraku in novo izračunanega filtra po enačbi

$$\mathbf{H}_i = \alpha \frac{\mathbf{G}_i \odot \mathbf{F}_i^H}{\mathbf{F}_i \odot \mathbf{F}_i^H} + (1 - \alpha)\mathbf{H}_{i-1}^H. \tag{2.15}$$

Na ta način se filter prilagaja spremembam izgleda tarče, hkrati pa skuša ohranjati robustnost.

Avtorji Henriques et al. [8] v svojem članku naslovijo nekatere probleme korelacijskih filtrov z uporabo linearne regresije in trika z jedrom ter za deskriptor uporabijo HOG [10]. Predlagani sledilnik imenujejo KCF (Kernelized Correlation Filters).

Za izziv VOT2014 [31] so isti avtorji sledilniku dodali še nekaj izboljšav, med katerimi je za naš namen ključna ocena skale. Ta se izvede tako, da se v vsakem koraku lokalizacije iz slike izrežejo trije deli, ki vsebujejo širše področje prejšnje lokacije objekta. Sledilnik interno shranjuje trenutno skalo objekta (relativno glede na začetek sledenja) in v vsakem koraku lokalizacije izvede korelacijo s trenutnim filtrom najprej nad delom, ki ustreza trenutni skali, nato pa še nad deloma, ki sta eno skalo večja oziroma manjša. Tako se izračunajo trije odzivi, za oceno spremembe skale in lokalizacijo pa se uporabi najmočnejši izmed njih. Sprememba skale tako ni poljubna, marveč se za vsako vhodno sliko lahko spremeni največ za eno stopnjo. Faktor za večjo in manjšo skalo je nastavljen na pet odstotkov trenutne velikosti tarče.

2.6.2 Sledilnik z deli LDP

Lukežič [27] v svojem magistrskem delu predstavi sledilnik LDP, ki nadgradi sledilnik KCF za boljše sledenje deformabilnim objektom. Sledilnik LDP lokalizira tarčo z grobo predstavitvijo, nato pa na srednjem nivoju inicializira dele, nad katerimi se zgradi sistem vzmeti, katerega minimizacija omogoči fino lokalizacijo centra tarče. Končno se vsi deli na obeh nivojih posodobijo in algoritem vrne trenutno regijo. V našem sistemu uporabljamo poenostavljeno obliko tega sledilnika, ki za lokalizacijo uporablja zgolj grobo predstavitev, torej pristop sledilnika KCF, združen z segmentacijo na podlagi barve.

Sledilnik modelira izgled objekta z uporabo globalne predloge in globalnega barvnega modela. Ker LDP izhaja iz sledilnika KCF, se globalna predloga uporabi tako, kot je opisano v članku [8]. Globalni barvni model pa za barvno segmentacijo uporablja histograma ospredja in ozadja.

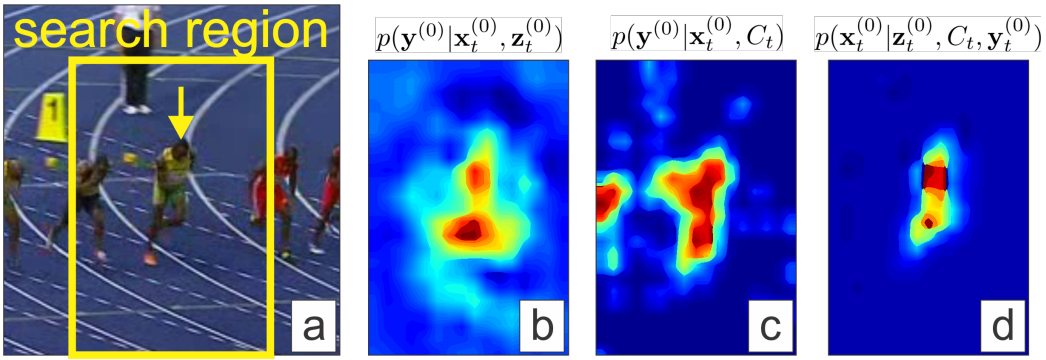
Center regije v trenutnem časovnem koraku se torej oceni z maksimizacijo gostote

$$p(x_t^{(0)}|I_t, C_t) \propto p(I_t|x_t^{(0)})p(C_t|x_t^{(0)}), \tag{2.16}$$

kjer faktor $p(I_t|x_t^{(0)})$ odraža vizualno podobnost med trenutno regijo in predlogo v obliki filtra. Faktor $p(C_t|x_t^{(0)})$ pa v enačbi predstavlja verjetnost, da posamezen piksel pripada objektu. To verjetnost pridobimo s povratno projekcijo barvnega histograma objekta v trenutno regijo, rezultat pa je nato regulariziran z uporabo Markovih slučajnih polj, kot je opisano v članku [32]. Oba pristopa in rezultat njunega produkta so prikazani na Sliki 2.14.

Detekcija odpovedi sledenja

Za praktični del našega pristopa je zelo pomembna detekcija odpovedi sledilnika. Za korelacijske filtre je to mera PSR (angl. peak to sidelobe ratio), kot jo v članku [26] opišejo Bolme et al. Ta mera označuje strmost odziva korelacijskega filtra in jo lahko uporabimo za določanje kvalitete odziva oz. za zaznavanje izgube sledenega objekta ali zakrivanja. Če trenutna slika jasno ustreza naučenemu filtru, bo vrh odziva zelo izstopal, če pa je objekt zakrit



Slika 2.14: Lokalizacija v sledilniku LDP, povzeto po [27]. a) izvorna slika z regijo preiskovanja, b) vizualna podobnost s filtrom, c) gostota verjetnosti, da piksel pripada objektu, d) združena gostota verjetnosti obeh modelov.

ali ni prisoten, bo vrh odziva precej manj izrazit. Mera PSR se izračuna tako, da izhod korelacije $\mathbf{g}$ razdelimo na vrh g_{max} in stranski valček (angl. sidelobe), ki vključuje vrednosti, ki niso v vrhu ali v njegovi okolici velikosti 11x11 pikslov. Vrednost PSR dobimo po enačbi

$$\frac{g_{max} - \mu_{sl}}{\sigma_{sl}}, \tag{2.17}$$

kjer sta μ_{sl} in σ_{sl} povprečje in standardni odklon stranskega valčka. Vrednosti se med normalnim sledenjem gibajo med 20 in 60, kadar pa padejo pod cca. 7, lahko sklepamo, da je sledilnik izgubil sledeni objekt. Če se to zgodi, je treba ročno reinicializirati sledilnik, ali pa zagnati sistem za redetekcijo, če je ta na voljo.

Poglavje 3

Naš pristop

V tem poglavju bomo opisali rešitve, ki smo jih uporabili za reševanje problema, predvsem regulator za premik kamere in regulator za nadzor kvadrokopterja. Poleg tega bomo predstavili še nekatere rešitve praktične nareve, ki smo jih morali dodati, da sistem zanesljivo deluje.

3.1 Regulacija pogleda kamere

Namen regulatorja kamere je prilagajati kot virtualne kamere tako, da bo objekt čim bližje centru slike. Regulator kamere rešuje enačbo (2.13) tako, da minimizira razdaljo med centrom objekta in centrom slike. Napaka se izračuna po enačbah:

$$\begin{aligned} \varepsilon_x &= \frac{x_r - \frac{w}{2}}{w}, \\ \varepsilon_y &= \frac{y_r - \frac{h}{2}}{h}, \end{aligned} \tag{3.1}$$

kjer sta x_r in y_r trenutni koordinati centra sledenega objekta, w in h pa širina in višina slike. Referenčna točka regulatorja je tako center slike (v splošnem lahko referenčno točko izberemo poljubno).

Če je evklidska razdalja med centrom objekta in centrom slike dovolj velika, se po enačbi (2.8) napaka pretvori v kot odmika, ki je potreben za

zmanjšanje napake, ta pa se posreduje regulatorju kvadrokopterja, ki izvede premik kamere. Za gladko delovanje sistema smo za pridobitev primernih vrednosti uporabili dva PID regulatorja, enega za vsako os. Kot je opisano v Poglavju 2.5.1, je namen PID regulatorjev poskrbeti za generiranje kontrolnih spremenljivk, ki kar se da učinkovito pripeljejo sistem v želeno stanje. Skrbijo torej, da ne prihaja do prekoračitev (oz. so te čim manjše) ali oscilacij, hkrati pa mora ostati sistem dovolj odziven.

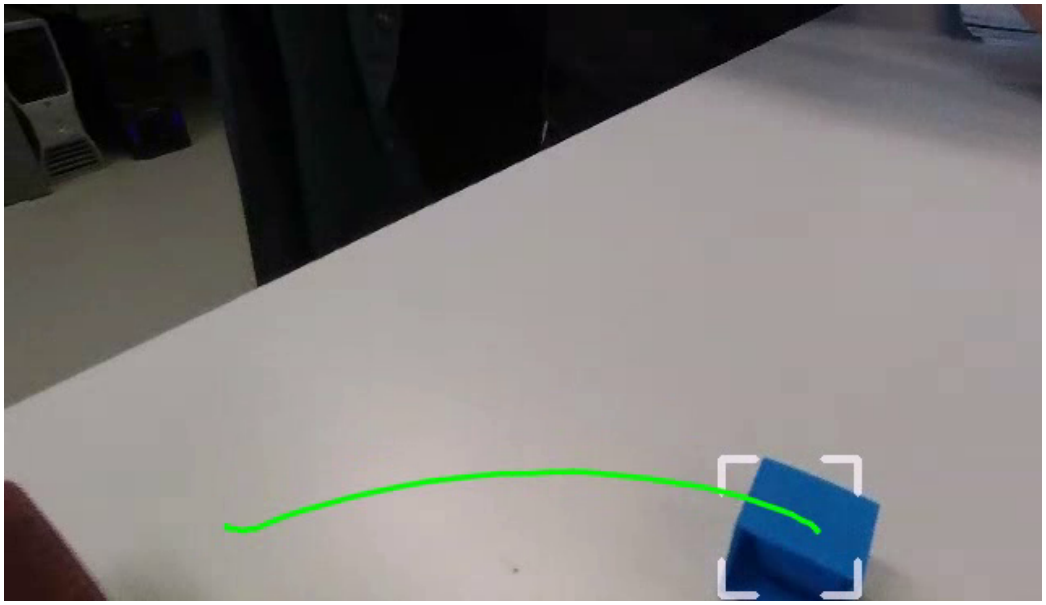
Problem globalne rektifikacije kamere

Pri sledenju objektov, ki ležijo blizu roba slike, lahko opazimo, da je njihova trajektorija ob premikih kamere ukrivljena. To je posledica dejstva, da je kamera širokokotna in zato nastala slika vsebuje močno radialno distorzijo, kot se vidi na Sliki 4.1. Kvadrokopter sicer z uporabo grafične procesne enote sliko interno izreže in rektificira, ne poravna pa tudi sosednjih pozicij kamere. Če premikamo pozicijo kamere po eni osi, pričakujemo premike slike objekta v kameri le po drugi osi. Izkaže pa se, da to drži le blizu optičnega centra kamere, dlje od njega pa se ravne črte ukrivijo, kot je prikazano na Sliki 3.1. Iz tega lahko sklepamo, da je izrez iz surove slike sicer brez vidnih napak, ki so posledica širokokotne kamere, globalno pa različne pozicije virtualne kamere ne ležijo na pravokotni mreži. Z eksperimentiranjem lahko opazimo, da trajektorije opaženih premikov v grobem ležijo na črtah, kot jih povzroči optična distorzija, prikazana na Sliki 2.4.

3.2 Regulacija položaja kvadrokopterja

Regulator, uporabljen v našem sistemu, se zgleduje in nadgradi tistega, ki je opisan v članku [4]. Ta vsebuje štiri povratne zanke, ki delujejo na značilnicah, pridobljenih s sledilnim algoritmom. Te tri značilnice so x in y koordinati centra objekta ter velikost sledene regije. Velikost objekta avtorji izračunajo po enačbi

$$\Delta_d = \sqrt{\frac{w \cdot h}{w_r \cdot h_r}}, \tag{3.2}$$



Slika 3.1: Distorzija zaradi rektifikacije surove slike.

kjer sta w in h dimenziji slike, w_r in h_r pa dimenziji regije objekta. Regulator je nato implementiran z uporabo štirih PID regulatorjev (roll, pitch, yaw, višina), c.f. [4].

Najprej moramo predstaviti oznake: ε_ϑ, ε_φ, ε_ψ in ε_z so napake, ki služijo kot vhodi v posamezen PID regulator. Značilke iz sledilnika pa bomo označevali z Δ_x, Δ_y in Δ_d, kjer prvi dve ustrezata spremembi koordinate centra objekta, zadnja pa spremembi velikosti sledenega objekta (relativno na velikost celotne slike). Vertikalni kot med objektom in optično osjo kamere bomo označevali s črko γ.

Pestana et al. [4] nastavijo vhode regulatorjev tako (oznake so spreme-

njene, da so primerljive z našimi, c.f. [4])

$$\begin{aligned} \varepsilon_\psi &= \Delta_x, \\ \varepsilon_\varphi &= \Delta_x - \frac{\psi_r - \psi}{\alpha_x}, \\ \varepsilon_z &= \Delta_y - \frac{\vartheta_r - \vartheta}{\alpha_y}, \\ \varepsilon_\vartheta &= \Delta_d. \end{aligned} \tag{3.3}$$

Avtorji pri vhodu regulatorja višine zaradi statične kamere upoštevajo trenuten nagib kvadrokopterja (kot ϑ), kar je pri stabilizirani kameri nepotrebno. Zaradi dvoumnosti premika slike objekta po osi x (ker ta ustreza tako premiku vstran kot tudi zasuku okrog osi z) avtorji za nastavitev ε_φ upoštevajo razliko med trenutnim in referenčnim kotom zasuka ψ, ki se nastavi le pri večjih spremembah ($> 25°$). Kvadrokopter se na Δ_y najprej odzove z zasukom po osi z, premik po osi Y pa se sproži šele po večji spremembi kota ψ.

Ker želimo naš regulator prilagoditi za sledenje z nagnjeno kamero, moramo uvesti nekatere spremembe. Najprej skušamo iz primerjave opaženih vrednosti Δ_y in Δ_d oceniti premik objekta po vertikalni ravnini, tako da izračunamo opaženi kot α. To naredimo s preoblikovanjem enačbe (2.8) v

$$\tan(\alpha + \gamma) = \frac{\Delta_y}{\Delta_d}, \tag{3.4}$$

saj je opaženi odmik za kot γ zamaknjen glede na dejanski kot premika α. Vpliv nagiba kamere je prikazan na Sliki 2.13. Oceno kota α nato dobimo z enačbo

$$\alpha = \tan^{-1}\left(\frac{\Delta_y}{\Delta_d}\right) - \gamma. \tag{3.5}$$

Pri tej formulaciji upoštevamo spremembe, ki jih kot med kamero in objektom povzroči na vizualnih značilnicah. Končne vrednosti, uporabljene za premik kvadrokopterja, pa morajo biti enake, kot če bi bil kot γ enak 0. To pomeni,

da napake izračunamo z naslednjimi formulami:

$$\begin{aligned} \varepsilon_\psi &= \Delta_x, \\ \varepsilon_\varphi &= 0, \\ \varepsilon_z &= \sin\alpha \ \mid \Delta_y \mid, \\ \varepsilon_\vartheta &= \cos\alpha \ \mid \Delta_d \mid . \end{aligned} \tag{3.6}$$

Tukaj vrednosti, ki jih izračunamo iz ocene premika objekta v sliki, dodatno obtežimo z velikostjo napake, ki smo jo opazili s kamero. Če torej opazimo premik pod kotom 0°, pomeni, da se je objekt premaknil vzporedno s horizontalno ravnino, velikost potrebnega premika pa nam določi velikost izmerjene napake Δ_d. Enako velja za ostale premike glede na kot α.

Zaradi enostavnosti spremembe kota φ sploh ne nastavljamo, marveč celoten premik slike objekta po osi y popravimo z zasukom okrog osi z. Spremenili smo tudi formulacijo spremembe globine, ki jo računamo po enačbi

$$\Delta_d = \sqrt{\frac{w_r \cdot h_r}{w \cdot h}}, \tag{3.7}$$

kjer so oznake enake kot v enačbi (3.2), le da smo obrnili ulomek, da so vrednosti značilke nekoliko bolj intuitivne (ta sedaj meri delež slike, ki ga zajema sledeni objekt). Kvadratni koren pa služi temu, da se vrednosti ne spreminjajo kvadratno, kar poenostavi generiranje kontrolnih vrednosti (vse značilke se spreminjajo linearno s spremembo v realnem prostoru).

Za dobro delovanje sistema moramo upoštevati tudi direktno interakcijo regulatorjev obeh delov sistema. Torej ni dovolj, da iz napak po oseh x in y generiramo kontrolne vrednosti za kamero in kvadrokopter, saj se v tem primeru lahko zgodi, da se objekt sicer nahaja v centru slike, da pa je hkrati kamera v skrajni legi. V tem primeru bo lahko majhen premik kvadrokopterja povzročil veliko napako, ki je regulator kamere sam ne bo mogel popraviti. Precej bolj smiseln način delovanja je torej, da sistem najprej s kamero sledi objektu, hkrati pa skuša poravnati kvadrokopter tako, da je kamera kar se da blizu svoji izhodiščni poziciji (odmik po obeh oseh naj bo enak 0).

Enačbe sistema je torej treba nekoliko spremeniti. Izračunati moramo razliko med izhodiščno in trenutno pozicijo kamere in to upoštevati pri vodenju kvadrokopterja. Tako dobimo naslednji enačbi:

$$\begin{aligned} \varepsilon_x &= \frac{(c_x - r_x)}{x_m}, \\ \varepsilon_y &= \frac{(c_y - r_y)}{y_m}. \end{aligned} \tag{3.8}$$

Tukaj c_x in c_y predstavljata trenutni poziciji kamere, r_x in r_y pa referenčni poziciji virtualne kamere (v našem primeru sta obe enaki 0). Vrednosti x_m in y_m tukaj predstavljata maksimalen kot odmika kamere po obeh oseh.

Kadar je kot γ enak 0, moramo upoštevati odmik po obeh oseh:

$$\begin{aligned} \varepsilon_\psi &= \Delta_x - \varepsilon_x, \\ \varepsilon_z &= \Delta_y - \varepsilon_y. \end{aligned} \tag{3.9}$$

V primeru, da fiksiramo kot γ na vrednost, večjo od 0, pa popravek upoštevamo le po osi x, ker se v tem primeru kot kamere ne sme regulirati.

3.2.1 Upoštevanje naklona kamere

Kot je opisano v Poglavju 1.2, se sledenje s kvadrokopterji večinoma izvaja v relativno enostavnem okolju (veliki, odprti in ravni prostori). Pri testiranju metod, opisanih v članku [4], smo opazili, da je sledenje objektom (oz. ljudem), kjer objekt in kvadrokopter ležita v isti ravnini, lahko problematično, če je prostor bolj raznolik (recimo prisotnost ovir, stopnic ipd.). Poleg tega, kadar sledimo ljudem, lahko kvadrokopter, ki človeku leti v višini obraza, povzroči neprijeten občutek. Za razširitev uporabnosti je torej primerno povečati višino, na kateri leti kvadrokopter, in temu ustrezno prilagoditi regulator. Ker ne poznamo ali predpostavimo višine sledenega objekta, bi bilo nesmiselno direktno nastavljati višino, marveč je boljši pristop, če določimo kot med objektom in kamero ter pustimo sistemu, da sam prilagodi višino kvadrokopterja. Prvi korak pri tem je fiksiranje kota kamere po osi y, imenujmo ga γ, takojšnja posledica tega pa je, da postane za centriranje objekta

odgovoren le regulator za višino kvadrokopterja. Če torej nastavimo smer pogleda kamere pod ravnino kvadrokopterja (konkretno: $c_y < 0$), bo slika objekta nastala v spodnji polovici slike kamere, za centriranje objekta pa se bo kvadrokopter moral premakniti navzgor. Ko sprožimo fiksiranje kota, se kamera v nekaj korakih premakne na ustrezno pozicijo (da se izognemo prehitrim premikom), nato pa regulator kvadrokopterja le-tega ustrezno dvigne, da se center objekta spet znajde v centru slike. Ob tem moramo kot γ posredovati tudi regulatorju kvadrokopterja, saj je njegovo delovanje odvisno od kota nagiba, kot je opisano v Poglavju 2.4.1 in Poglavju 3.2.

3.3 Integracija sledilnika

Sledilnik, ki ga uporabljamo v sistemu, se imenuje LDP (Layered Deformable Parts) in je opisan v delu [28]. Sledilnik je integiran v knjižnici Legit[1]. V knjižnico je najprej vključena implementacija sledilnika KCF, ki deluje kot osnova za lokalizacijo objekta, njegova lokacija pa je nato precizirana, kot je opisano v Poglavju 2.6.2. Zaradi želje po delovanju v realnem času je KCF implementiran v programskem jeziku C++ (za razliko od originalnega sledilnika v okolju MATLAB), iz istega razloga se uporablja tudi deskriptor FHOG, implementiran z SSE ukazi (implementiral P. Dollar [2]).

3.3.1 Inicializacija sledilnika

Sledilnike po navadi inicializiramo z uporabo pravokotnika (s klikom določimo eno od oglišč, s potegom miške pa še njemu diagonalno oglišče). A ker včasih želimo sledilnik zagnati na premikajočem se objektu, oz. ker je določena mera premikanja v nekaterih sistemih prisotna že zaradi njihove narave (leteči roboti), ta pristop ni vedno ustrezen. V našem sistemu lahko zato sledilnik zaženemo na tri različne načine: s pravokotnikom, z enim samim klikom, ki sledilnik inicializira v kvadratu okrog izbrane točke, ali z avtomatsko inicia-

[1] https://github.com/vicoslab/legit
[2] https://github.com/pdollar/toolbox

lizacijo na obrazu, ki ga zazna detektor obrazov. Razmeroma enostavno bi bilo vključiti tudi poljuben detektor za sledenje drugih objektov.

3.3.2 Detekcija odpovedi

Kot je opisano v Poglavju 2.6.2, se kvaliteta sledenja v določenem časovnem koraku meri z mero PSR. V članku [26] je opisano, da padec te mere pod določen prag označuje delno zakrivanje (angl. occlusion) oziroma izgubo objekta. Eksperimentalno smo ugotovili, da padec mere PSR pod določen prag ne sovpade vedno z odpovedjo sledilnika, marveč lahko to povzročijo tudi nenadni premiki ali zakrivanja, sledenje pa se nato brez težav nadaljuje. Za praktično uporabo te mere smo tako poleg pragovne vrednosti nastavili še število zaporednih detekcij, katerih mera PSR mora biti pod pragom, da sistem zazna odpoved sledilnika. V primeru, da sistem zazna detekcijo, ali da ta pade izven vidnega polja kamere, se vsa regulacija kvadrokopterja takoj zaustavi, da ne pride do nepredvidljivih premikov. Tudi zaradi tega je natančna detekcija odpovedi pomemben del celotnega sistema. V primeru, da bi v sistem vključili tudi metodo redetekcije, bi bilo smiselno implementirati še metode za preiskovanje prostora, kot jih omenijo v članku [3].

3.3.3 Predikcija iskalnega okna

Sledilni algoritmi večinoma niso dovolj hitri, da bi preiskali celotno sliko in našli njen predel, ki najbolj ustreza sledenemu objektu, marveč to počnejo na manjšem območju, določenem s predhodnimi pozicijami objekta. To lahko povzroči probleme pri nenadnih premikih v sliki, saj lahko objekt med dvema zaporednima slikama spremeni svojo pozicijo bolj kot to predvideva sledilnik. Kot je opisano v Poglavju 2.6.2, sledilnik LDP izvede ujemanje s korelacijskim filtrom na delu slike, ki obkroža lokacijo prejšnje regije. Sledilnik lahko torej pravilno sledi objektu le, če njegova slika ne pade izven tega dela slike (ta del ustreza 1,5 krat povečani velikosti regije). Če je premik objekta večji od tega, bo sledilnik skoraj gotovo odpovedal. Ker imamo v sistemu nekaj

informacij o bodočih premikih v kameri (kontrola iz sistema in joypada, vzlet, pristanek ipd.), lahko le-te uporabimo za grobo predikcijo pozicije objekta v sliki. To naredimo z direktno manipulacijo notranjega stanja sledilnika, ki mu nastavimo prejšnjo pozicijo (tisto, v katere okolici išče naslednjo pozicijo) relativno na predviden premik objekta. To naredimo tako, da izračunamo velikost predvidenega premika v pikslih. Če premik kamere sprožimo direktno s pošiljanjem ukaza gonilniku kvadrokopterja, lahko za pretvorbo uporabimo enačbo (2.8), sicer moramo premik izračunati iz izhodnih vrednosti regulatorja, ki se generirajo v našem sistemu. Koordinati objekta po premiku kamere tako dobimo po enačbah

$$\begin{aligned} x_p &= x_c - d_x, \\ y_p &= y_c - (-d_y), \end{aligned} \tag{3.10}$$

kjer sta d_x in d_y izračunani velikosti premika po posamezni osi, x_c in y_c pa trenutni koordinati objekta. Ker hočemo vpliv premika negirati, moramo velikosti premika odšteti od trenutne pozicije. V primeru koordinate y moramo vrednost premika prej še negirati, saj se vrednosti koordinat y v sliki povečujejo v smeri navzdol. V eksperimentu iz Poglavja 5.1.2 je prikazano, kako to sledilniku omogoči prenesti večje nenadne premike, ne da bi izgubil sledeni objekt. Sicer je res, da je zaporedje slik, ki ga dobimo iz kamere kvadrokopterja, dokaj zvezno (če ne upoštevamo izpadlih slik) in da bi problem velikih premikov lahko rešili tudi z uporabo dolgoročnega sledilnika, a naš pristop kljub temu ponuja boljšo rešitev, saj ob večjih nenadnih premikih lahko pride do zabrisanja zaradi premika (angl. motion blur) oziroma do večjih sprememb v izgledu objekta, kar bi lahko povzročilo težave pri redetekciji objekta. Naš pristop tukaj pripomore k izogibu odpovedi sledilnika, ki se lahko spremembi izgleda objekta takoj prilagodi.

Poglavje 4

Implementacija sistema

V tem poglavju bomo opisali konkretno platformo, na kateri smo implementirali in testirali predlagan sistem, ogrodje ROS (Robot Operating System)[1], znotraj katerega smo sistem integrirali, ter strukturo in potek delovanja našega sistema.

4.1 Kvadrokopter Parrot Bebop 1

Sistem smo implementirali na kvadrokopterju Bebop Drone podjetja Parrot. Na platformi se nahaja dvojedrni procesor ARM Cortex A9, na katerem teče operacijski sistem Linux. Na kvadrokopterju se nahajajo naslednji senzorji:

- pospeškometer,
- žiroskop,
- ultrazvočni senzor,
- kamera za optični tok,
- glavna kamera.

Pospeškometer in žiroskop sta standardna oprema na dronih in služita stabilizaciji, ultrazvočni senzor in kamera za optični tok se nahajata na spodnjem delu naprave in se uporabljata za oceno hitrosti in pozicije kvadrokopterja (kamera za optični tok ocenjuje premike glede na tla, ultrazvočni senzor

[1] http://www.ros.org/

pa višino). Zaradi uporabe optičnega toka je Bebop zmožen t.i. lebdenja (angl. hover), kadar mu ne podamo nobenih ukazov. Z uporabo optičnega toka zazna spremembe v teksturi tal in jih z vgrajenim regulatorjem skuša negirati, z ultrazvočnim senzorjem pa ohranja višino.

Ena opaznejših posebnosti tega kvadrokopterja je širokokotna glavna kamera, ki ima vidni kot približno 180°, kot je vidno na Sliki 4.1. Z uporabo grafične procesne enote na čipu se iz vsake širokokotne slike izreže okno v razmerju 16:9, kar omogoča simulacijo premične kamere (uporabnik lahko spreminja smer, v katero kaže izrezana slika), poleg tega pa se lahko z uporabo informacij iz žiroskopa negira vpliv nagnjenosti naprave in s tem doseže programsko stabilizacijo slike. Torej, tudi če je kvadrokopter nagnjen (po kotih φ ali ϑ), se iz širokokotne slike izreže del slike, ki še vedno gleda naravnost. Tako se slika uspešno stabilizira tudi pri skoraj navpičnih nagibih. Med letenjem se po brezžičnem omrežju prenaša video v velikosti 640×368, hkrati pa se v interni pomnilnik shranjuje video z resolucijo 1080p. Kvadrokopter ob zagonu ustvari svojo brezžično dostopno točko, na katero se povežemo z računalnikom, nato pa po omrežju pošilja podatke o svojem stanju (stanje baterije, orientacija, hitrost ipd.) in tok slik iz glavne kamere. Video teče s 30 slikami na sekundo, frekvenca drugih informacij pa je omejena na 5Hz.

4.2 ROS

Naš sistem je implementiran v ogrodju ROS. To je široko razširjen sistem za robotske aplikacije, saj omogoča enostavno pošiljanje sporočil med različnimi programi (v ROSu je program, ki opravlja točno določeno nalogo in komunicira z drugimi programi, imenovan vozlišče - angl. node) in vključuje množico knjižnic in gonilnikov, namenjenih robotiki. Posamezna vozlišča med seboj komunicirajo preko t.i. tem (angl. topic). Teme delujejo tako, da vozlišče, ki bere oz. generira podatke, objavlja sporočila v temo, vozlišča, ki te podatke potrebujejo, pa se lahko na to temo naročijo (angl. subscribe). ROS posamezna vozlišča privzeto poganja paralelno, po potrebi pa si lahko vo-

Slika 4.1: Surova slika iz kamere.

zlišča obsežnejše vire (recimo slike) tudi delijo. To zelo poenostavi velik del aktivnosti, zahtevanih pri delu z roboti, kot je branje podatkov z različnih senzorjev, pošiljanje ukazov robotu in medprocesna komunikacija.

ROS vključuje standardne tipe sporočil, kot so cela števila, števila s plavajočo vejico, nizi znakov ipd., hkrati pa omogoča, da napišemo svoje lastne tipe sporočil, ki jih nato uporabimo pri komunikaciji med vozlišči. Tako lahko na primer napišemo svoje sporočilo, ki shrani vse podatke za inicializacijo sledilnika (izbrana regija in prva slika).

V ROS je vključen tudi sistem zajemanja sporočil, imenovan `rosbag`. Z njim lahko na disk shranimo vsa sporočila, ki so se v nekem časovnem obdobju prenesla preko ogrodja. To nam lahko zelo poenostavi delo z mobilnimi roboti, saj lahko shranjena sporočila kasneje predvajamo in analiziramo precej enostavneje kot med samim letenjem oz. vožnjo.

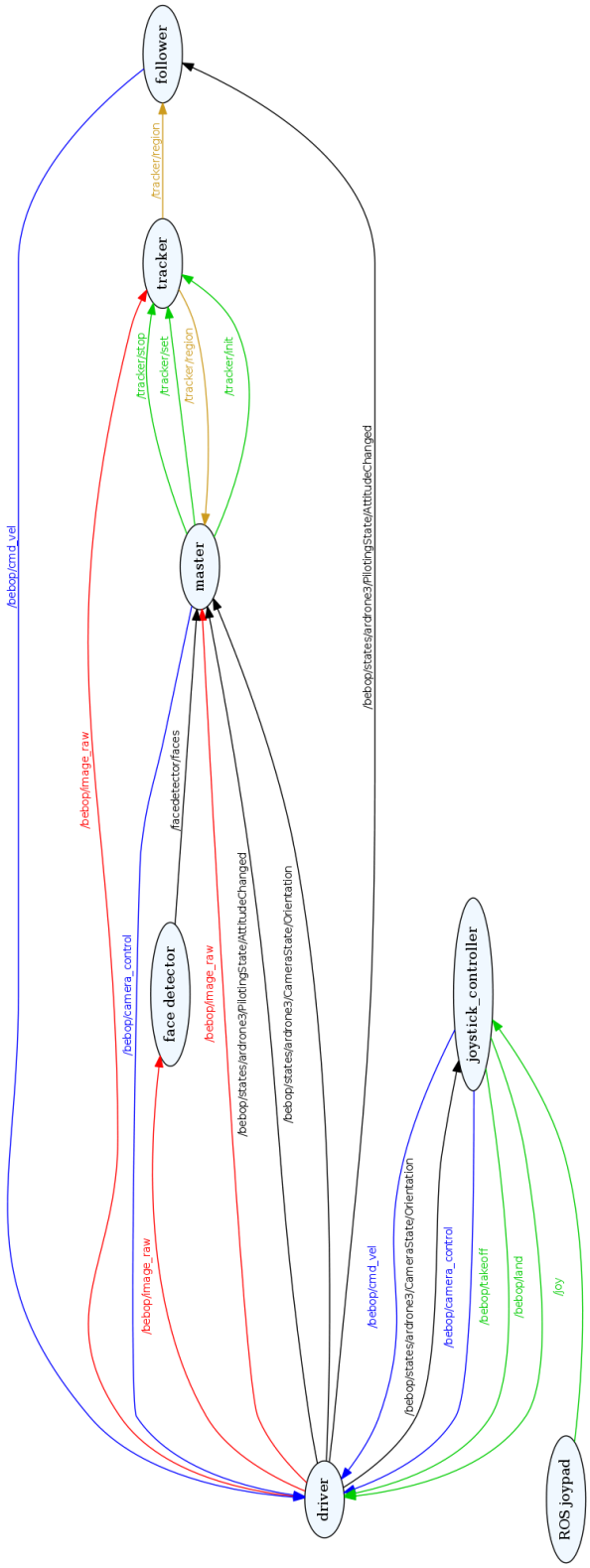
4.3 Integracija sistema v ROSu

ROS zaradi večje preglednosti in modularnosti poudarja ločevanje delov sistema na manjše enote, zato je naš sistem implementiran z večimi sestavnimi deli (vozlišči). Ti so:

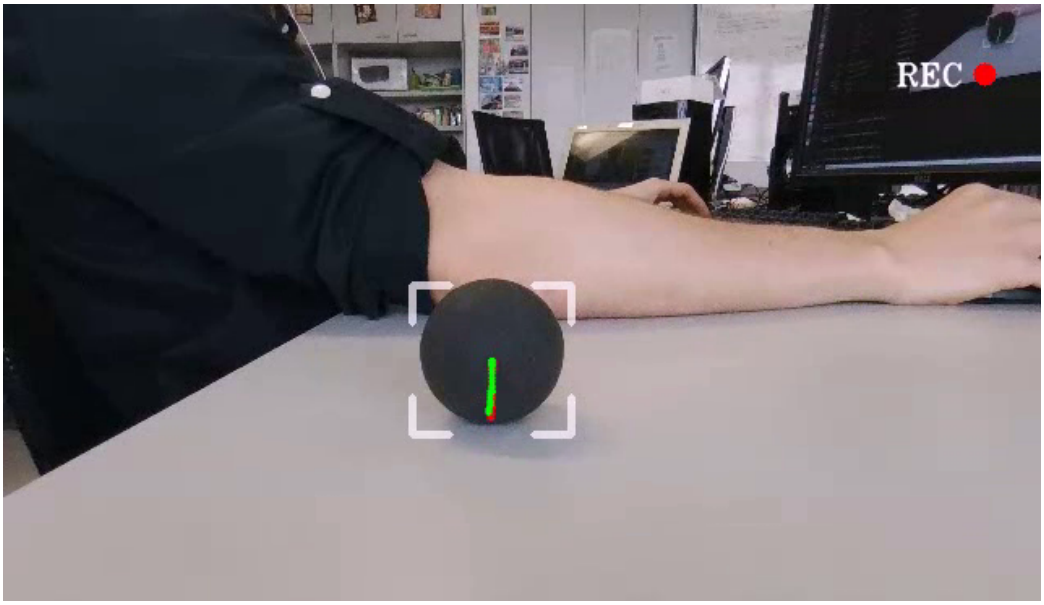
- gonilnik (`driver`),
- glavno vozlišče (`master`),
- vozlišče sledilnika (`tracker`),
- vozlišče sledilca/zasledovalca (`follower`),
- detektor obrazov (`facedetector`),
- kontroler igralnega ploščka (`joy`).

Vozlišča so med sabo povezana, kot prikazuje Slika 4.2. Sistem deluje tako, da po vklopu kvadrokopterja in vzpostavitvi brezžične povezave poženemo gonilnik[2], ki nastavi povratne funkcije (angl. callback) in začne brati podatke, ki jih pošilja kvadrokopter, in jih posredovati v ustrezne ROS teme. Tok slik iz kvadrokopterja berejo glavno vozlišče, detektor obrazov in vozlišče

[2]https://github.com/AutonomyLab/bebop_autonomy



Slika 4.2: Graf ROS vozlišč.



Slika 4.3: Prikaz vmesnika med sledenjem.

sledilnika (na grafu označeno z rdečo barvo). Uporabnik s sistemom komunicira preko igralnega ploščka, miške in tipkovnice (označeno z zeleno barvo). Prikaz delovanja sledilnika (regija objekta, trajektorija) je na Sliki 4.3.

Ko sistem zaženemo, ta začne prikazovati sliko iz kvadrokopterja, hkrati pa se zaženejo tudi vsa druga vozlišča, ki nato čakajo na uporabnikov vnos. Ko uporabnik inicializira sledilnik, se kreira sporočilo tipa `Im_reg`, ki vsebuje sliko s časovno oznako in vgnezdeno sporočilo tipa `Region`, ki hrani informacije o poziciji in velikosti regije za inicializacijo. Vozlišče sledilnika nato s pomočjo označene regije inicializira sledilnik, ki začne brati zaporedne slike direktno iz kvadrokopterja in vsako uporabi za posodobitev regije sledenega objekta. Regija, ki opisuje objekt, se nato pošlje nazaj glavnemu vozlišču, ki skrbi za prikaz in vodenje kamere, ter vozlišču sledilca, ki iz regije izračuna potreben premik kvadrokopterja (v grafu z rumeno barvo). Nadzor kamere in kvadrokopterja se izvajata preko sporočila tipa `geometry_msgs::Twist`, ki vsebuje dva vektorja s po tremi komponentami, ki opisujeta premik po vseh šestih prostostnih stopnjah (linearni in kotni premik po oseh x, y in z).

Premik kamere se nastavlja s kotnim zasukom okrog osi y in z, premik kvadrokopterja pa z linearnim premikom po oseh x, y in z (ti ustrezajo kotoma ϑ, φ ter spremembi višine) ter s kotnim zasukom okrog osi z (ustreza spremembi kota ψ). Prenos sporočil za vodenje je na Sliki 4.2 označen z modro barvo. Sistem smo uporabljali na dveh različnih platformah. Prva je osebni računalnik s procesorjem Intel Xeon X5450 in 16GB delovnega pomnilnika, druga pa prenosni računalnik s procesorjem Intel i3 in 4GB delovnega spomina. Kljub občutni razliki med sistemoma, nismo opazili spremembe v hitrosti delovanja. Slovenska agencija za civilno letalstvo je leta 2016 izdala direktivo o varnosti glede uporabe brezpilotnih zrakoplovov, ki omejuje uporabo kvadrokopterjev in podobnih naprav, na območja, kjer ni ljudi in kjer se nahajajo le objekti, ki niso namenjeni bivanju.

Poglavje 5

Eksperimentalna evalvacija

Za konkreten prikaz prednosti našega pristopa smo predlagani sistem evalvirali z različnimi eksperimenti, ki so namenjeni ločenemu testiranju njegovih različnih funkcionalnosti. Ločeno smo nastavili parametre in intenzivno testirali najprej regulator kamere, nato pa ločeno še regulator kvadrokopterja samega. Množico testov smo zaključili s tremi eksperimenti, ki preizkusijo skrajne meje delovanja predlaganega sistema in jih primerjajo s sistemom s fiksno kamero.

5.1 Evalvacija regulatorja kamere

Eksperimenti v tem poglavju testirajo zmožnosti sledilnika in virtualne kamere za ohranjanje objekta čim bolj v centru vidnega polja v raznolikih situacijah.

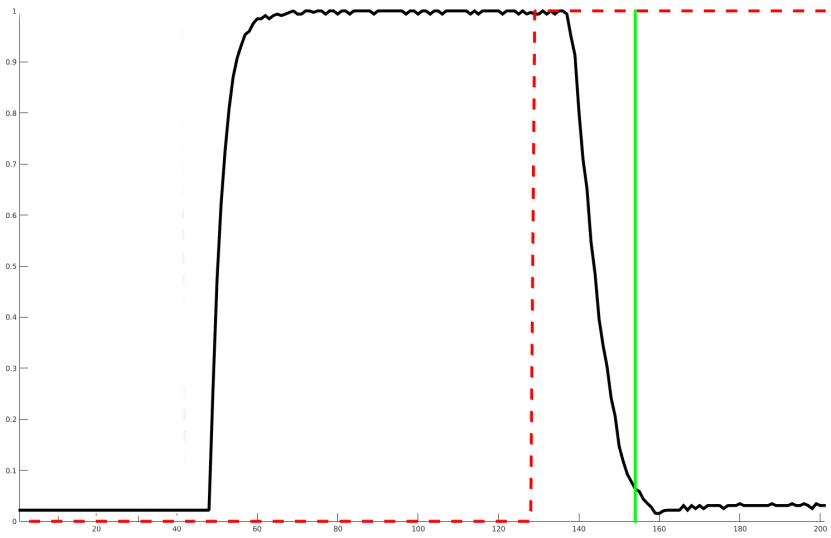
5.1.1 Kalibracija PID regulatorja

Ta eksperiment je namenjen pridobitvi optimalnih vrednosti parametrov za PID regulator kamere (eden za vsako os slike). Zaradi dinamike sistema je fokus usmerjen v odzivnost regulatorja, ki se mora na nenadne premike v sliki hitro odzvati, hkrati pa s svojim delovanjem ne sme povzročiti dodatnih motenj v sliki (oscilacije).

Najprej moramo določiti pravilno razmerje med vrednostjo napake, ki jo izračunamo iz regije objekta, in vrednostmi za kontrolo kamere. Nastavljanje ustreznih parametrov PID regulatorja začnemo s proporcionalnim delom (K_p). Naša prva izbira je bila vrednost 1, kar se je izkazalo za izrazito premajhno, saj ima gonilnik kvadrokopterja spodnjo mejo za premik kamere. Za hitro pridobitev ustreznih vrednosti smo uporabili bisekcijo, torej smo vrednosti podvajali oziroma prepolavljali, dokler nismo dosegli optimalnih rezultatov. Te smo merili s časom, ki je pretekel med zagonom regulatorja in ustalitvijo v končnem stanju. Poleg tega smo opazovali morebitno prisotnost oscilacij. Oscilacije nastanejo, kadar ima eden od parametrov preveliko vrednost, kar povzroči, da se stanje sistema približa želenemu stanju, nato pa ga prekorači in se mora zatem popraviti v nasprotno smer. Po teoriji PID regulatorjev je idealen odziv sistema na vzbuditev z enotinim impulzom enak enotski stopnici, torej kar se da hiter premik v končno stanje, brez prekoračitve.

Eksperiment za posamezen nabor vrednosti parametrov smo izvedli tako, da smo na mirujočem predmetu inicializirali sledilnik, nato pa avtomatizirano premaknili virtualno kamero, da smo inducirali premik objekta v sliki oz. napako med centrom objekta in centrom slike. Ko je kamera dosegla želeno končno pozicijo, smo vklopili regulator in počakali, da se sistem ustali v končnem stanju, torej izniči vso vnešeno napako. S preslikavo med spremembo kota kamere in spremembo, ki jo ta povzroči v sliki (po enačbi (2.8), smo doesgli, da je bil premik kamere v vsaki iteraciji enak. Kot začetni trenutek ustalitve smo obravnavali regijo, katere center je ležal znotraj majhne okolice centra slike (v naših eksperimentih je bil to krog s polmerom 10px). Poleg tega smo po začetni ustalitvi počakali 80 naslednjih detekcij, da smo se prepričali o stabilnosti končnega stanja. To nam je hkrati omogočilo tudi detekcijo prekoračitev oz. oscilacij sistema. Če je po začetni ustalitvi center regije kdaj padel izven zahtevanega območja, smo to zaznali kot oscilacijo in ustrezno prilagodili izmerjeni čas do ustalitve.

Izmerjena vrednost t_s v Tabeli 5.1 je čas med vklopom regulatorja kamere

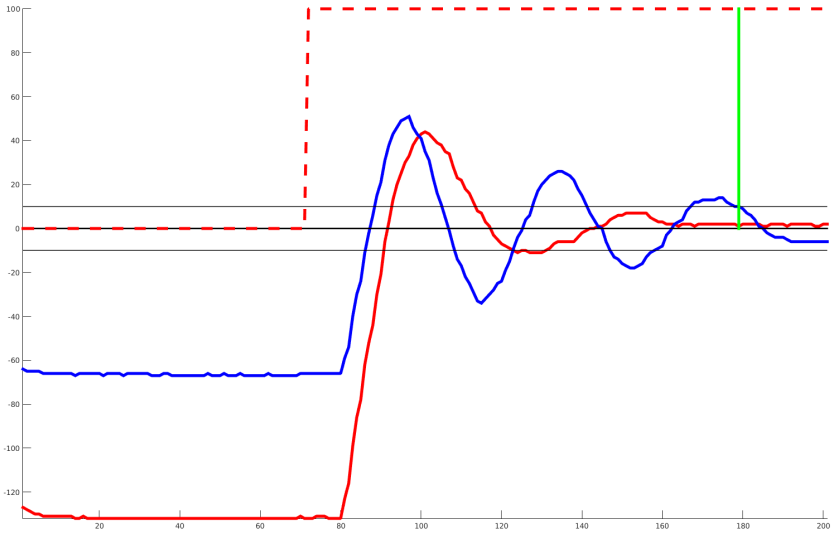


Slika 5.1: Odmik centra objekta od centra slike v odvisnosti od časa. Rdeča črtkana črta označuje aktivnost regulatorja, zelena vertikala pa označuje trenutek ustalitve v končnem stanju.

in ustalitvijo sistema (manjše vrednosti kažejo na hitrejšo ustalitev in pomenijo kvalitetnejši regulator). Na Sliki 5.2 je prikazan primer oscilacije, ki se lahko pojavi pri prevelikih vrednostih parametrov regulatorja. Za vsako konfiguracijo parametrov smo izvedli vsaj 15 premikov v velikosti 250 pisklov, nato pa izračunali povprečen čas do ustalitve (upoštevajoč prekoračitve in oscilacije). Ugotovili smo, da je pri optimalnih paremetrih povprečen čas do ustalitve v končnem stanju 1,017s. Izvedba eksperimenta z optimalnimi parametri je prikazana na Sliki 5.3.

5.1.2 Evalvacija predikcije regije iskanja objekta

Eksperiment, opisan v tem razdelku, služi za evalvacijo predikcije za sledilnik, kot je opisano v Poglavju 3.3.3. Izveden je bil z večjimi premiki virtualne kamere, ki so sledeni objekt 'zanesli' izven regije, ki jo sledilnik navadno preišče,



Slika 5.2: Oscilacija pri preveliki vrednosti parametra K_p. Rdeča in modra krivulja predstavljata odmik centra sledenega objekta od centra slike, rdeča črtkana črta prikazuje aktivnost regulatorja kamere, zelena vertikala pa trenutek ustalitve v končnem stanju.

x			y					
K_p	K_i	K_d	K_p	K_i	K_d	t_s[s]	prekoračitev	oscilacija
2	0	0	2	0	0	27,357	0	0
4	0	0	4	0	0	15,179	0	0
8	0	0	8	0	0	12,429	0	0
16	0	0	16	0	0	10,714	0	0
32	0	0	32	0	0	1,500	y	0
64	0	0	64	0	0	3,785	x,y	1
32	0	0	24	0	0	1,250	0	0
32	0	2	24	0	2	1,057	0	0
32	0	4	24	0	4	1,05	0	0
32	0	8	24	0	8	1,250	0	0
32	2	8	24	2	8	1,303	x,y	0
32	1	8	24	1	8	1,341	0	0
32	1	8	24	0,5	8	1,143	y	0
32	1	8	24	0,25	8	1,107	y	0
32	1	8	24	0,125	8	1,071	0	0
32	1	16	24	0,125	16	1,214	0	0
32	1	32	24	0,125	32	1,017	0	0
32	1	64	24	0,125	64	1,250	y	0

Tabela 5.1: Parametri za PID regulatorja kamere.

zato je prišlo do odpovedi (PSR pod določenim pragom za več zaporednih slik). S tem, da smo v trenutku premika kamere nastavili regijo sledilnika na predvideno lokacijo, smo omogočili sledilniku, da preišče drugo območje slike kot bi ga sicer in se s tem izogne odpovedi.

Najprej moramo najti približno velikost odmika, ki zagotovo povzroči izgubo objekta. To naredimo na podoben način kot v prejšnjem eksperimentu, torej s premikom za določeno število pikslov v naključni smeri. Nato iterativno povečujemo razdaljo premika, dokler ne pridemo do razdalje, ki je sledilnik ne more več premagati. Zatem lahko začnemo uporabljati predikcijo, da v trenutku premika sledilniku nastavimo novo regijo, ki jo preiskuje in ga tako usmerimo v pravi del slike, kamor se bo objekt vizualno premaknil. Zaradi efekta, opisanega v Poglavju 3.1, je ta predikcija manj natančna, ko se bližamo robovom slike, a ker sledilnik preišče razmeroma veliko področje v okolici prejšnje detekcije, to ne vpliva bistveno na učinkovitost predikcij.

Eksperiment smo implementirali avtomatizirano, tako da se ob njegovem zagonu najprej kamera centrira na objekt, nato pa se shranijo začetna pozicija kamere in podatki za inicializacijo sledilnika. Regulator kamere se potem izklopi, da ne povzroča motenj pri premikih, za tem pa se izmenično izvajajo premiki v naključno smer in centriranje kamere, da slika objekta ne obtiči ob robu vidnega polja kamere. V primeru odpovedi sledilnika se le-ta reinicializira iz shranjenih podatkov, sproti pa merimo število odpovedi za posamezno velikost premika.

Eksperiment smo izvedli s tremi različnimi velikostmi objekta v sliki (60×60, 40×40 in 30×30 pisklov). Velikost odmika smo povečevali od 150 do 300 pikslov. Uporabili smo sistem brez predikcije, kjer je lokalizacijo izvajal samo sledilnik in sistem s predikcijo, ki je aktivno nastavil izhodišče preiskovane regije sledilnika, kot je opisano v Poglavju 3.3.3.

Opisani eksperiment smo najprej izvedli z objektom velikosti 60×60 pikslov in ugotovili, da samo s premikom virtualne kamere ne moremo povzročiti odpovedi sledilnika. Razlog za to je, da je velikost preiskovanje regije odvisna od velikosti objekta. Prve odpovedi smo zaznali pri velikosti objekta

Slika 5.3: Prikaz delovanja predikcije. Z zeleno je označena predvidena pozicija objekta po premiku kamere.

velikost objekta	velikost premika [px]	uspešnost brez predikcije[%]	uspešnost s predikcijo[%]
60x60	150	100	100
	200	100	100
	250	100	100
	300	100	100
40x40	150	100	100
	200	70	100
	250	20	100
	300	0	100
30x30	150	100	100
	200	50	100
	250	0	100
	300	0	100

Tabela 5.2: Uspešnost predikcije pod različnimi pogoji.

40×40 pikslov, kjer je sledilnik začel odpovedovati pri premikih velikosti 200 pikslov. Z uporabo predikcije smo se uspešno izognili odpovedi sledilnika tudi za precej velike premike kamere pri razmeroma majhnih velikostih sledenega objekta (če je velikost objekta 30×30 pisklov, to predstavlja le 0,38 odstotka celotne površine slike). Premikov večjih od 300 pisklov nismo izvajali, saj bi ponesli objekt izven vidnega polja kamere. Le-to je približno 40° po osi x in približno 25° po osi y (izračunano po enačbi (2.8)). Rezultati so podani v Tabeli 5.2.

5.2 Evalvacija regulatorja kvadrokopterja

Razvili smo tuid eksperimente, ki testirajo uspešnost kvadrokopterjevih premikov glede na vizualne značilke. Cilj eksperimentov je določiti primerne parametre za gladko odzivanje kvadrokopterja na spremembe v sliki, preizkusiti sledenje z nagibom kamere in hkrati zaznati morebitne težave, ki nastanejo zaradi nekaterih predpostavk v našem sistemu (gledamo odmik objekta v sliki, ne pa tudi sprememb v ozadju, omejitve strojne opreme, omejitve sledilnika ipd.).

5.2.1 Kalibracija PID regulatorja

S tem eksperimentom smo določili vrednosti PID regulatorjev, ki nadzorujejo premike kvadrokopterja. Za določitev primernih vrednosti smo izolirali vsako od treh smeri nadzora, ki jih uporabljamo. Za vsako od njih smo najprej nastavili parametra K_i in K_d na 0, parameter K_p pa na 1. Z opazovanjem sistema med delovanjem in analizo grafov napake smo iterativno določili vrednosti parametrov za stabilno delovanje. Ker so lahko premiki med letenjem nepredvidljivi, smo se osredotočili predvsem na gladke odzive in stabilno delovanje, ne toliko na samo hitrost odzivov, ki bi v robnih primerih lahko povzročili nestabilno obnašanje kvadrokopterja. Predvsem pri premikih naprej in nazaj smo parametre nastavili precej konservativno, saj se sledilnik spremembi skale včasih ne prilagodi popolnoma pravilno, kar lahko

θ			ψ			z		
K_p	K_i	K_d	K_p	K_i	K_d	K_p	K_i	K_d
1	0,25	0	1	0	0,25	1,5	0,125	0

Tabela 5.3: Parametri za PID regulatorje kvadrokopterja.

povzroči oscilacije in zmanjša stabilnost.

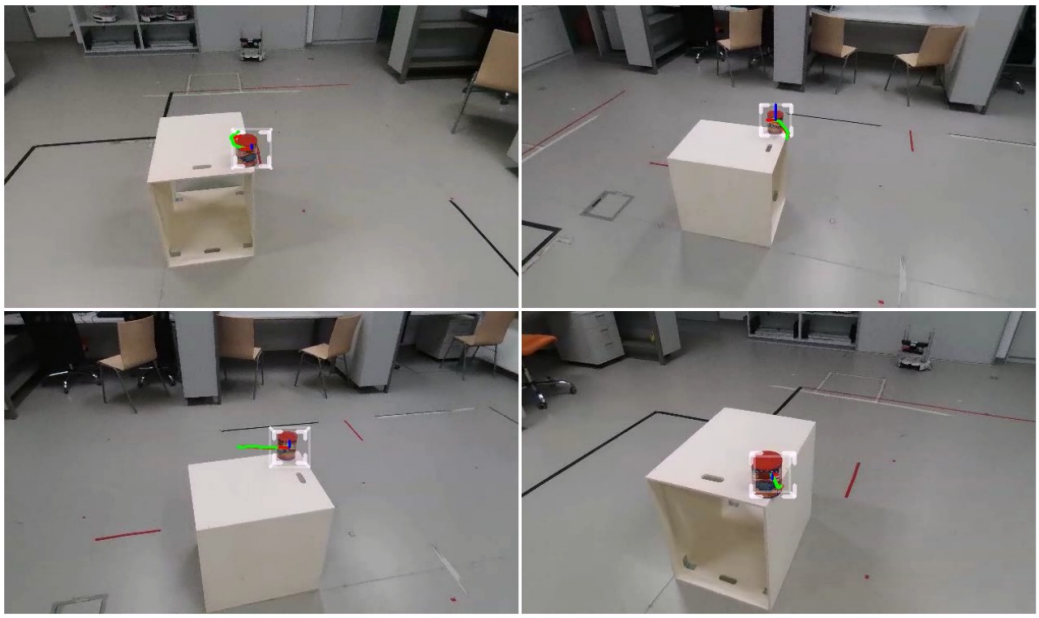
5.2.2 Regulacija pozicije ob naklonu kamere

Eksperiment preizkusi delovanje predlagane sheme za nadzor kvadrokopterja, kadar ta leti višje od sledenega objekta. Namen poskusa je preveriti uspešnost sistema, da kvadrokopter zavzame in ohranja določen kot glede na objekt (nastavi ga uporabnik), kot je opisano v Poglavju 3.2.1. Opazujemo, kako se spreminja višina, na kateri leti kvadrokopter, glede na želeni kot in kvaliteto ocene kota, ki jo pridobimo iz vizualnih značilk sledilnika. Eksperiment smo izvedli tako, da smo med letom inicializirali sledilnik na statičnem objektu v vidnem polju kamere, nato pa ročno določili kot med kamero in objektom. Potek eksperimenta je prikazan na Sliki 5.4.

Opazimo lahko, da naša rešitev, ki določitev ustrezne višine posredno določi preko fiksiranja kota kamere po osi y, deluje pravilno. Opazimo pa lahko, da začne kvadrokopter krožiti okrog statičnega objekta, saj sistem deluje le na podlagi odmika centra objekta od sredine slike. Ker pri tem ne postavimo nobenih drugih omejitev, se zaradi nepopolne stabilizacije kvadrokopter začne premikati vstran. Sistem sicer popravi njegov zasuk okrog osi z in ohranja razdaljo do objekta, a v trenutnem stanju ne more izničiti napak tega tipa.

5.2.3 Dinamično sledenje ob naklonu kamere

Eksperiment iz poglavja 5.2.2 smo nato ponovili še z dinamičnim objektom. Sledilnik smo inicializirali med letom, nastavili kot γ na vrednost -35°, nato pa objekt premikali po prostoru in opazovali odziv sistema. Zanimalo nas je

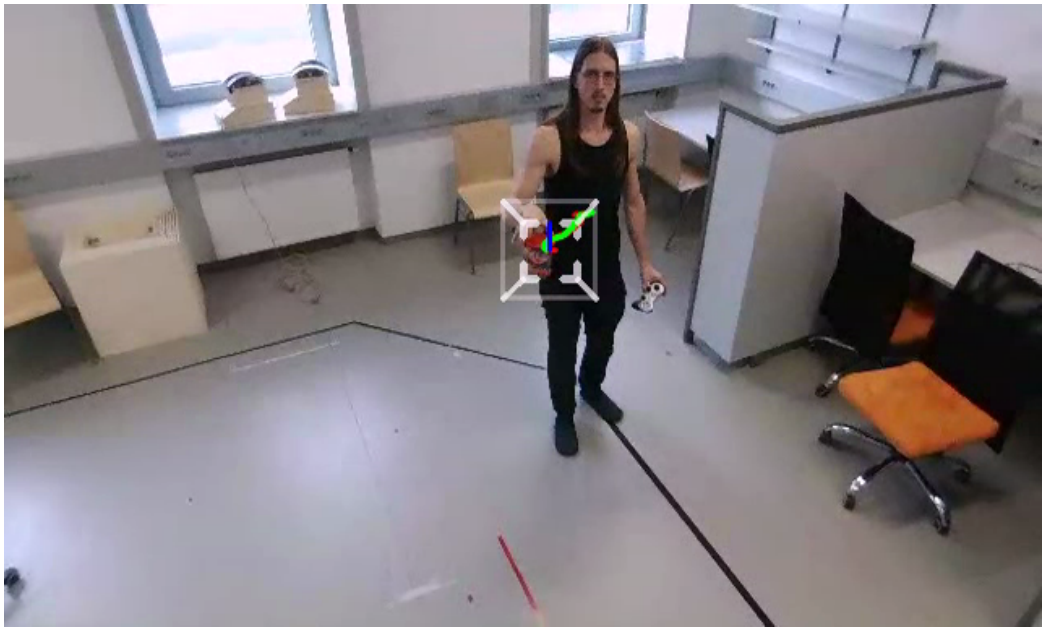


Slika 5.4: Sledenje statičnega objekta z naklonom kamere.

predvsem, če kvadrokopter uspešno ohranja zadani kot med kamero in objektom ter če se primerno prilagodi povečanju oziroma zmanjšanju razdalje med njima. Opazili smo, da se problem kroženja okrog objekta tu ne pojavi, če se objekt konstantno premika. Ker smo omejeni z resolucijo kamere in natančnostjo ocene skale, ki jo izvaja sledilnik, morajo biti za pravilno sledenje objekti v sliki dovolj veliki (pomaga tudi, če se dovolj razlikujejo od ozadja, a to velja za vsakršno avtomatsko vizualno sledenje). Če namreč začnemo z objektom, ki zajema premajhen del celotne slike, bo sledilnik težko zaznal potencialno zmanjšanje objekta, kar lahko povzroči napake in odpoved sledenja. Prikaz sistema med delovanjem je na sliki 5.5.

5.3 Evalvacija regulacije zasledovanja

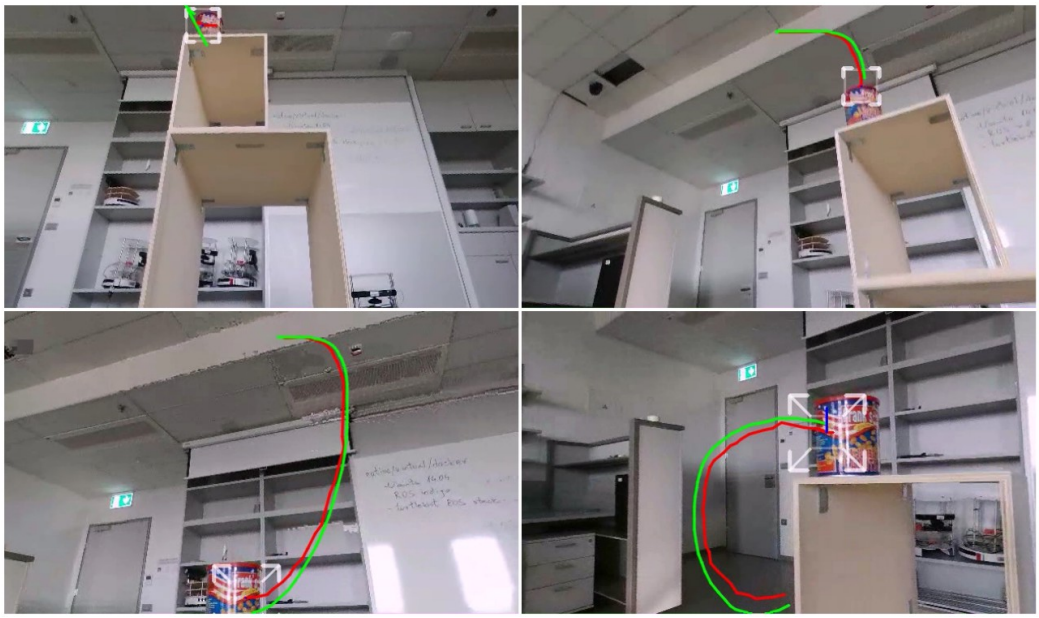
V tem razdelku bomo predstavili tri eksperimente, ki smo jih nastavili za intenziven preizkus sposobnosti našega sistema in zajemajo raznolike okoliščine, v katerih naš sistem še posebej izstopa.



Slika 5.5: Sledenje premikajočega se objekta z naklonom kamere.

5.3.1 Uspešnost vzleta

Prispevek našega sistema v primerjavi z drugimi, ki uporabljajo fiksne kamere, je v tem, da lahko sledilnik inicializiramo med tem, ko je kvadrokopter še na tleh, in šele nato zaženemo proces vzleta. Premična kamera nam tu omogoča pogled navzgor proti objektu, ki smer pogleda samodejno prilagodi, ko se po vzletu višini kvadrokopterja in objekta izravnata. Statične kamere so tu omejene po svojem vidnem polju oziroma po razdalji med objektom in kamero. Torej, če je objekt dovolj daleč, dovolj majhnen, ali če ima kamera dovolj velik zorni kot, lahko objekt ostane znotraj slikovne ravnine tudi med vzletom, sicer tak manever ni mogoč. Za pravilen vzlet moramo le poskrbeti, da se napaka zaradi odmika kamere ne začne upoštevati, preden se kvadrokopter v zraku stabilizira, sicer lahko to v začetku povzroči nekoliko oscilacij. Za izvedbo eksperimenta smo sledeni objekt dvignili na višino 96cm in opazovali sposobnost vzleta brez odpovedi sledilnika. Uporabili smo dve verziji sistema, eno, kjer smo fiksirali kamero (s tem smo simulirali kvadro-



Slika 5.6: Potek vzleta s premično kamero.

kopter s fiksno kamero, kakršnega so uporabili v primerljivih člankih [4, 2, 7]) in drugo, kjer je sistem s kamero aktivno sledil objektu. Za izhodišče smo določili razdaljo, kjer je bil objekt ravno še v vidnem polju kamere, ko ta gleda naravnost, kar je 2,5m. Oba sistema sta na tej razdalji vzletela brez težav. Ker je to za fiksno kamero spodnja meja razdalje, smo nadaljnje poskuse izvedli le s premično kamero. Zmanjševali smo razdaljo in ugotovili, da tak sistem uspešno izvede vzlet tudi pri svoji minimalni razdalji, ki je 80cm. Potek vzleta pri minimalni razdalji je prikazan na sliki 5.6. Tudi to razdaljo bi se morda z uporabo predikcije še dalo zmanjšati (torej, da bi glede na dinamiko vzleta kamero vnaprej premaknili navzdol), a je po naših opažanjih dvig kvadrokopterja hitrejši od premika virtualne kamere, poleg tega pa je sledenje objektom na še manjši razdalji redkokdaj smiselno.

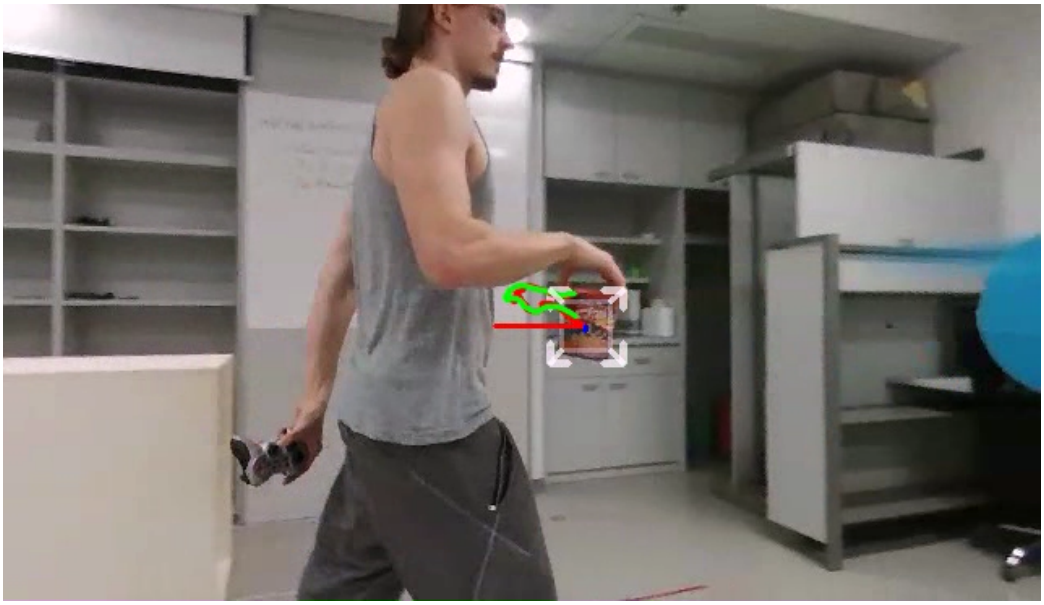
5.3.2 Zasledovanje pri hitrih premikih

Prednost tega, da objektu vzporedno sledimo s premično kamero in s kvadrokopterjem je med drugim tudi to, da se sistem lahko hitreje odzove na premike. To lastnost smo preizkusili tako, da smo med letom inicializirali sledilnik na objektu, nato pa ga premikali v krogu okrog kvadrokopterja. Kvadrokopter je zato napako minimiziral samo z zasukom okrog osi z. Za eksperiment smo izklopili regulator za ohranjanje razdalje, da je kvadrokopter ostal kar se da blizu centra kroga. Oseba je med izvedbo eksperimenta dvignila objekt in se z njim premikala po obodu kroga. Eksperiment smo najprej izvedli s fiksno kamero, nato pa še s premično. Opazovali smo, kolikšna je maksimalna hitrost premikanja objekta, ki ji sistem še lahko sledi. Za testiranje smo uporabili krog s polmerom 2m. Med sledenjem smo merili čas, ki ga je oseba porabila za en obhod kroga in iz tega ocenili hitrost premikanja objekta. Podobno kot pri eksperimentu z vzletom smo najprej preizkusili meje sistema s fiksno kamero in ugotovili, da je maksimalna hitrost, pri kateri se objekt lahko premika okrog kvadrokopterja, enaka 1,4m/s (čas za obhod je bil 9,1s). Pri sistemu s premično kamero smo ugotovili, da lahko to hitrost povečamo do približno 2,3m/s preden sledilnik odpove. Eksperiment smo pri vsaki hitrosti ponovili večkrat in dobili konsistentne rezultate. Slika 5.7 prikazuje stanje sistema med izvedbo eksperimenta.

Želeli smo pokazati tudi, da dinamična kamera pripomore k hitrim odzivom na spremembe v sliki. Ker se kamera na opažene spremembe lahko odzove hitreje kot celoten kvadrokopter, se tako lahko objekt okrog kvadrokopterja premika z večjo hitrostjo (oz. na manjši razdalji).

5.3.3 Zasledovanje po klančini

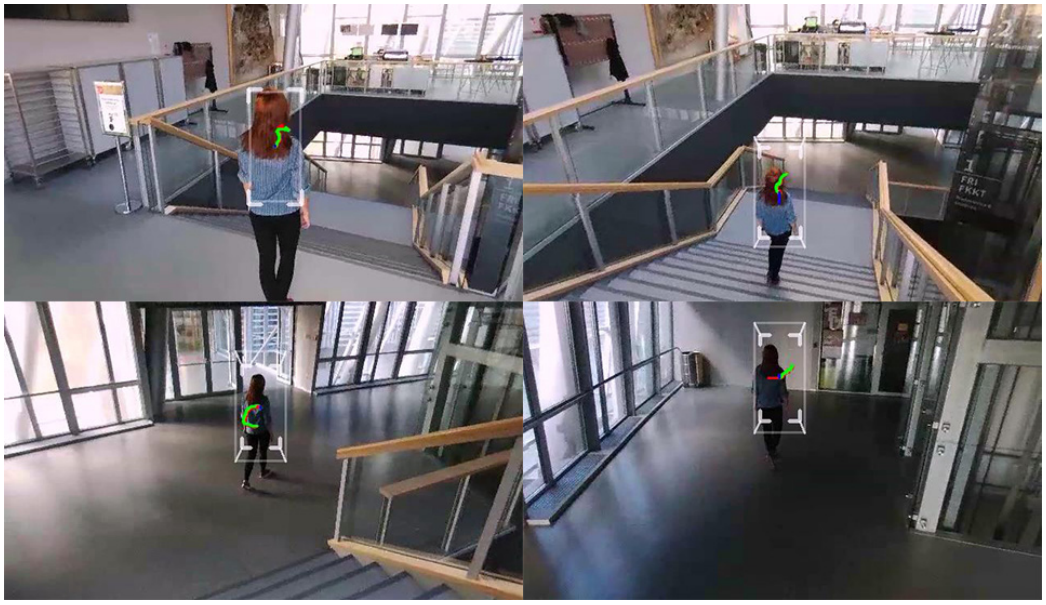
Eden od problemov sistemov s fiksnimi kamerami je, da ne omogočajo dolgoročnega zasledovanja objektov, ki se premikajo po neravnih prostorih, denimo po klančinah. Če se objekt premakne izven ravnine kvadrokopterja, ta minimizira nastalo napako s spremembo svoje višine, torej kot med kamero



Slika 5.7: Prikaz kroženja sledenega objekta okrog kvadrokopterja za evalvacijo maksimalne hitrosti sledenja.

in objektom predstavlja zgolj napako v meritvah. Sledenje pod kotom je sicer mogoče s fiksno kamero, ki je usmerjena navzdol, a je zaradi spremenjenega geometričnega odnosa med kamero in objektom treba ustrezno spremeniti regulator kvadrokopterja.

Delovanje našega sistema, ki tovrstno sledenje omogoča, smo tako preizkusili s sledenjem osebe med hojo po stopnicah. Sledilnik smo inicializirali med letenjem, nato pa nastavili kot γ na 35°, da se je kvadrokopter dvignil nad osebo. Oseba se je nato spustila po stopnicah. Opazili smo lahko želen odziv sistema, torej, da se je ob zmanjšanju tarče kvadrokopter premaknil naprej, zaradi spusta sledene regije pa tudi zmanjšal svojo višino, kar je prikazano na sliki 5.8. Ker je bil že v začetku približno na višini dveh metrov, mu spust ni povzročil težav. Eksperiment smo ponovili večkrat in preverili, da vodenje kvadrokopterja deluje konsistentno, pod pogojem, da je inicializacija sledilnika dobra in da se le-ta pravilno prilagodi spremembi velikosti tarče, kar je ključnega pomena. Verzija našega sistema s fiksno kamero uspešne



Slika 5.8: Sledenje osebe med hojo po stopnicah.

izvedbe tega eksperimenta ne omogoča, saj se pri razmeroma strmem spustu kvadrokopter prilagodi višini tarče in ob tem zaleti v tla (sprememba velikosti je majhna v primerjavi s spremembo višine).

Poglavje 6

Sklepne ugotovitve

V delu smo predstavili sistem, ki regulira premično kamero, da ohranja objekt v centru vidnega polja kamere, hkrati pa formulirali regulator kvadrokopterja, ki omogoča ohranjanje objekta blizu centra slike tudi med letenjem. Prispevali smo izvirno formulacijo regulatorja kvadrokopterja, ki omogoča sledenje pod kotom, sproti poravnava kamero in kvadrokopter, da ohranja kar največjo fleksibilnost sistema in omogoča vzlet in pristanek brez izgube objekta. Poleg tega smo razširili možnosti uporabe podobnih sistemov na območja, ki niso vezana na ravnino in dodali še nekatere praktične detajle, vezane na sledenje s kvadrokopterjem. Sestavili in izvedli smo tudi več vrst eksperimentov, ki so primerni za evalvacijo podobnih sistemov. Pokazali smo, da lahko z našim sistemom kvadrokopter vzleti, ne da bi izgubil objekt tudi pri zelo majhnih razdaljah. Na premike objekta po oseh x in y se lahko naš sistem odziva s precejšno hitrostjo, poleg tega pa omogoča tudi daljše sledenje tarč z naklonom kamere (do 35°) in občutnimi spremembami velikosti tarče. Sposobnost sledenja z naklonom kamere sistemu omogoča tudi sledenje ob premikih objekta, ki niso vezani na ravnino tal.

6.1 Nadaljnje delo

Trenutni problem našega sistema je kroženje kvadrokopterja okrog statičnega objekta, saj se sistem odziva le na odmik centra objekta od centra slike. Če zaradi nepopolne stabilizacije kvadrokopter zdrsne nekoliko vstran, se ta napaka popravi s spremembo zasuka po osi z. To povzroči, da sistem dojame napako za odpravljeno, čeprav se je smer njegovega pogleda na objekt spremenila. Napake tega tipa bi se dalo nadgraditi z metodami, ki bi upoštevale spremembe v predelu slike, ki ne predstavlja objekta in s tem skušale oceniti lasten premik kamere. To bi lahko naredili z uporabo detektorja vogalov (na podoben način kot Engel et al. v članku [33]). Morebitne odpovedi sledilnika bi prav tako lahko naslovili z eksplicitno uporabo metod za ponovno detekcijo, ki jih uporabljajo sorodni sistemi [2, 9, 1]. Mogoča bi bila tudi ocena položaja in orientacije kamere s katero od direktnih metod za rekonstrukcijo prostora, kot sta DTAM [34] ali LSD-SLAM [35].

Literatura

[1] H. Lim, S. N. Sinha, Monocular localization of a moving person onboard a quadrotor mav, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 2182–2189.

[2] K. Haag, S. Dotenco, F. Gallwitz, Correlation filter based visual trackers for person pursuit using a low-cost quadrotor, in: 15th International Conference on Innovations for Community Services, I4CS 2015, Nuremberg, Germany, July 8-10, 2015, 2015, pp. 1–8. doi:10.1109/I4CS.2015.7294481.
URL http://dx.doi.org/10.1109/I4CS.2015.7294481

[3] M. Danelljan, F. S. Khan, M. Felsberg, K. Granström, F. Heintz, P. Rudol, M. Wzorek, J. Kvarnström, P. Doherty, A low-level active vision framework for collaborative unmanned aircraft systems, in: Workshop at the European Conference on Computer Vision, Springer, 2014, pp. 223–237.

[4] J. Pestana, J. L. Sánchez-López, S. Saripalli, P. Campoy, Computer vision based general object following for gps-denied multirotor unmanned vehicles, in: American Control Conference, ACC 2014, Portland, OR, USA, June 4-6, 2014, 2014, pp. 1886–1891. doi:10.1109/ACC.2014.6858831.
URL http://dx.doi.org/10.1109/ACC.2014.6858831

[5] J. Pestana, I. Mellado-Bataller, J. L. Sánchez-López, C. Fu, I. F. Mondragón, P. Campoy, A general purpose configurable controller for indo-

ors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles, Journal of Intelligent and Robotic Systems 73 (1-4) (2014) 387–400. doi:10.1007/s10846-013-9953-0.
URL http://dx.doi.org/10.1007/s10846-013-9953-0

[6] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE transactions on pattern analysis and machine intelligence 34 (7) (2012) 1409–1422.

[7] M. Danelljan, G. Häger, F. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: British Machine Vision Conference, Nottingham, September 1-5, 2014, BMVA Press, 2014.

[8] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596.

[9] M. Danelljan, F. Shahbaz Khan, M. Felsberg, J. Van de Weijer, Adaptive color attributes for real-time visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1090–1097.

[10] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, IEEE, 2005, pp. 886–893.

[11] M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for uav tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 445–461.

[12] J. Papon, M. Schoeler, Semantic pose using deep networks trained on synthetic rgb-d, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 774–782.

[13] H. Hattori, V. Naresh Boddeti, K. M. Kitani, T. Kanade, Learning scene-specific pedestrian detectors without real data, in: Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3819–3827.

[14] M. Mueller, G. Sharma, N. Smith, B. Ghanem, Persistent aerial tracking system for uavs, in: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, IEEE, 2016, pp. 1562–1569.

[15] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, P. H. Torr, Struck: Structured output tracking with kernels, IEEE transactions on pattern analysis and machine intelligence 38 (10) (2016) 2096–2109.

[16] C. Harris, M. Stephens, A combined corner and edge detector., in: Alvey vision conference, Vol. 15, Citeseer, 1988, pp. 10–5244.

[17] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: Binary robust independent elementary features, in: European conference on computer vision, Springer, 2010, pp. 778–792.

[18] Y. Freund, R. E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: European conference on computational learning theory, Springer, 1995, pp. 23–37.

[19] J.-L. Blanco, A tutorial on se (3) transformation parameterizations and on-manifold optimization, University of Malaga, Tech. Rep 3.

[20] F. Chaumette, S. Hutchinson, Visual servo control. i. basic approaches, IEEE Robotics & Automation Magazine 13 (4) (2006) 82–90.

[21] H. T. Nguyen, M. Worring, R. Van Den Boomgaard, Occlusion robust adaptive template tracking, in: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, Vol. 1, IEEE, 2001, pp. 678–683.

[22] S. S. Nejhum, J. Ho, M.-H. Yang, Online visual tracking with histograms and articulating blocks, Computer Vision and Image Understanding 114 (8) (2010) 901–914.

[23] G. Shu, A. Dehghan, O. Oreifej, E. Hand, M. Shah, Part-based multiple-person tracking with partial occlusion handling, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1815–1821.

[24] C.-H. Kuo, C. Huang, R. Nevatia, Multi-target tracking by on-line learned discriminative appearance models, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 685–692.

[25] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, D. G. Lowe, A boosted particle filter: Multitarget detection and tracking, in: European Conference on Computer Vision, Springer, 2004, pp. 28–39.

[26] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2544–2550.

[27] A. Lukežič, Izboljšan robusten model z regijami za vizualno sledenje objektov, Ph.D. thesis, Univerza v Ljubljani (2015).

[28] A. Lukežič, L. Čehovin, M. Kristan, Deformable parts correlation filters for robust visual tracking, arXiv preprint arXiv:1605.03720.

[29] P. Naidu, Improved optical character recognition by matched filtering, Optics Communications 12 (3) (1974) 287–289.

[30] D. S. Bolme, B. A. Draper, J. R. Beveridge, Average of synthetic exact filters, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2105–2112.

[31] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, et al., The Visual Object Tracking VOT2014 Challenge Results, Springer International Publishing, Cham, 2015, pp. 191–217. doi:10.1007/978-3-319-16181-5_14.
URL http://dx.doi.org/10.1007/978-3-319-16181-5_14

[32] M. Kristan, J. Perš, V. Sulič, S. Kovačič, A graphical model for rapid obstacle image-map estimation from unmanned surface vehicles, in: Asian Conference on Computer Vision, Springer, 2014, pp. 391–406.

[33] J. Engel, J. Sturm, D. Cremers, Scale-aware navigation of a low-cost quadrocopter with a monocular camera, Robotics and Autonomous Systems 62 (11) (2014) 1646–1656.

[34] R. A. Newcombe, S. J. Lovegrove, A. J. Davison, Dtam: Dense tracking and mapping in real-time, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2320–2327.

[35] J. Engel, T. Schöps, D. Cremers, Lsd-slam: Large-scale direct monocular slam, in: European Conference on Computer Vision, Springer, 2014, pp. 834–849.