Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Pedro Damian Kostelec

Mozaičenje s kvadrokopterjem AR.Drone

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

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

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

Št. naloge: 00067/2013
Datum: 02.04.2013

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

Kandidat: **PEDRO DAMIAN KOSTELEC**

Naslov: **MOZAIČENJE SLIK S KVADROKOPTERJEM AR.DRONE**
IMAGE MOSAICING WITH QUADRICOPTER AR.DRONE

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Mozaičenje slik je proces, ki iz množice slik sestavi večjo sliko in sicer tako, da slike registrira med seboj in jih postavi v skupni koordinatni sistem. V diplomski nalogi izdelajte postopek za mozaičenje zaporedja slik posnetega z nizkocenovnim kvadrokopterjem AR.Drone. Postopek zasnujte na algoritmih računalniškega vida, ki naj s pomočjo detekcije značilnih točk na zaporednih slikah le-te paroma registrirajo med seboj. Za mozaičenje poizkusite uporabiti tudi podatke o hitrosti leta in orientaciji kvadrokopterja med letom ter dobljene rezultate primerjajte s tistimi dobljenimi na osnovi slikovne informacije.

Mentor:
doc. dr. Danijel Skočaj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Pedro Damian Kostelec, z vpisno številko **63100213**, sem avtor diplomskega dela z naslovom:

Mozaičenje s kvadrokopterjem AR.Drone

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 13. september 2013 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

CSV ang. Comma-separated values, vrednosti ločene z vejico

JPEG ang. Joint Photographic Experts Group

PNG ang. Portable Network Graphics

RANSAC ang. RANdom SAmple Consensus

SDK ang. Software development kit, orodja za razvoj programske opreme

SIFT ang. Scale-invariant feature transform

USB ang. Universal Serial Bus

Povzetek

V tem diplomskem delu smo razvili sistem, ki omogoča avtomatsko gradnjo mozaika večjega števila slik pridobljenih z mobilno platformo Parrot AR.Drone. Zaporedne pare slik smo združili na podlagi izločenih parov korespondenčnih točk na slikah. Korespondenčne pare smo filtrirali z algoritmom RANSAC, z nelinearno optimizacijsko metodo Levenberg-Marquardt pa smo ocenjeno homografijo optimizirali. Predstavili smo tudi nekaj pravil za detekcijo slabo ocenjenih homografij. Zgrajene mozaike smo ovrednotili in razložili zakaj nekatere slike delujejo bolje kot druge. Poskusili smo tudi rekonstruirati let kvadrokopterja na podlagi navigacijskih podatkov, ki jih prejemamo med letom. Ugotovili smo, da so prejeti navigacijski podatki preveč nenatančni in šumni za natančno rekonstrukcijo, ki bi jo potrebovali za gradnjo mozaika.

Ključne besede: mozaičenje, AR.Drone, RANSAC, SIFT, Levenberg--Marquardt, homografija, radialna distorzija, kvadrokopter, ukrivljanje slik, projektivna geometrija

Abstract

In this bachelor thesis we developed a system for the automatic mosaicing of video frames obtained from the mobile platform Parrot AR.Drone. Sequential images were collated by extracting putative matching keypoints. Matching pairs were filtered with the RANSAC algorithm and the estimated homography was optimized using the Levenberg-Marquardt non-linear optimization method. We also present a set of rules to detect badly estimated homographies. We evaluate the generated mosaics and explain why some images are better suited for mosaicing than others. Moreover, we attempted to reconstruct the flight path using navigation data received from the quadricopter during flight. We concluded that the measured navigation data is too imprecise and noisy to use it for mosaicing.

Keywords: mosaicing, AR.Drone, RANSAC, SIFT, Levenberg-Marquardt, homography, radial distortion, quadricopter, image warping, projective geometry

Poglavje 1

Uvod

Kvalitetni posnetki Zemeljske površine so pomembni za navigacijo, gradbeništvo, arheologijo, kmetijstvo, planiranje mest, itd. Vendar je pridobitev visoko ločljivih posnetkov drago. V tem diplomskem delu predstavimo rešitev, ki združuje slike v mozaik s pomočjo nizkocenovnega kvadrokopterja.

To poglavje smo razdelili na štiri dele. V prvem motiviramo problem, ki ga bomo reševali in predstavimo nekaj primerov uporabe. V drugem določimo cilje diplomske naloge. V tretjem prestavimo sorodna dela in v zadnjem razložimo zgradbo diplomske naloge.

1.1 Motivacija

Mapiranje je človeška dejavnost, pri kateri zapisujemo našo percepcijo sveta na grafični način. Prvi znani zemljevidi segajo dalje v preteklost kot katerikoli drug način pisne komunikacije [1]. Ljudje smo ustvarili zemljevide, da bi lažji opisali svet in planirali poti. Čeprav imamo zemljevide skoraj vseh obljudenih predelov Zemlje, niso ti zemljevidi tako natančni kot bi si včasih želeli. Na primer, lastnik plantaže bi v vsakem trenutku rad imel pregled nad svojo plantažo, da bi čimprej odkril izumirajoče rastline, še preden okužijo druge rastline v okolici. Vodja gradbišča bi rad imel aktualen tloris gradbišča, da bi lažje meril napredek gradnje in načrtoval naslednjo akcijo.

Urbanist bi z aktualnim zemljevidom mesta lažje načrtoval gradnjo nove podzemeljske železnice. Vozniki pa bi radi na svojih navigacijskih napravah videli zemljevide visoke ločljivosti, da bi se lažje orientirali po okolici.

Iz opisanih primerov razberemo dve ključne lastnosti zemljevidov, ki jih v današnjih zemljevidih pogosto pogrešamo: aktualnost in visoka ločljivost. Slika 1.1a prikazuje zemljevid neke lokacije iz aplikacije Google Maps[1], kjer opazimo pomanjkanje aktualnosti in visoke ločljivosti. Na Sliki 1.1b je prikazan aktualen zemljevid iste lokacije, ki je višje ločljivosti. Ta zemljevid smo zgradili v sklopu te diplomske naloge.

Da bi lažje razumeli zakaj javno dostopnim zemljevidom, kot so zemljevidi v navigacijskih napravah in aplikacijah, kot je Google Maps, manjkajo opisane lastnosti, moramo poznati vire slikovnega materiala, ki se uporablja za gradnjo teh zemljevidov. Pogosti vir slikovnega materiala za izdelavo zemljevidov so posebej opremljeni sateliti, ki letijo v nizki orbiti. Ti sateliti so opremljeni z dragimi kamerami visoke ločljivosti, vendar je zaradi oddaljenosti satelita od Zemlje ločljivost na njeni površini samo med 0.6 do 10 metrov na slikovni element (ang. pixel) [2] [3]. Uporaba takšnih satelitov je omejena na sončne dni. Za pridobitev slik z višjo ločljivostjo se uporabljajo posebej opremljena letala, ki slikajo površino Zemlje z ločljivostjo več slikovnih elementov na meter [4]. Če želimo pridobiti slike z višjo ločljivostjo, se moramo spustiti nižje, ali uporabiti boljše kamere. V tem diplomskem delu si bomo ogledali prvi pristop. Namesto satelitov in letal bomo uporabili kvadrokopterje.

Kvadrokopterji so leteča plovila, ki za vzgon in premikanje uporabljajo štiri propelerje. Kontrola leta je dosežena s spreminjanjem hitrosti vrtenja enega ali več propelerjev. AR.Drone je kvadrokopter, ki je zaradi nizke cene in enostavnosti uporabe dosegljiv širši javnosti. Opremljen je z dvema kamerama za zajem video posnetkov in slik.

Vgrajeni kameri služita razvedrilu, z dodatno programsko opremo pa ju lahko izkoristimo za resnejše namene. En tak namen, ki ga bomo predstavili

[1] `https://www.google.com/maps/`

(a) (b)

Slika 1.1: Levo je prikazana slika iz Google Maps v najvišji ponujeni ločljivosti za neko lokacijo. Desno je odsek mozaika iste lokacije, ki smo ga izdelali v tem diplomskem delu. Z belim okvirjem smo na levi sliki označili približno lokacijo desne slike. Celoten mozaik prikažemo v Poglavju 6.4.

v tem diplomskem delu, je sestavljanje mozaikov. Mozaik je skupek delno prekrivajočih se slik, ki skupaj tvorijo večjo sliko. Če združimo dovolj veliko število zaporednih slik zajetih s kamero na kvadrokopterju, dobimo zemljevid, ki prikazuje pot, ki jo je preletel kvadrokopter. Slika 1.1 primerja ločljivost slike iz aplikacije Google Maps in mozaik iste lokacije, ki smo ga izdelali v tem diplomskem delu.

Z rešitvijo, ki jo predstavimo v tej diplomski nalogi, bomo zadostili potrebam lastnika plantaže, ki bo lahko vsako jutro s kvadokopterjem preletel svoje zemljišče, zgradil mozaik, in preveril, ali ni na novem zemljevidu vidno kakšno večje odstopanje od zemljevida preteklega dneva, tedna ali meseca, kar bi lahko pomenilo, da je škodljiv osebek napadel plantažo. Na podoben način zadostimo željam vodje gradbišča, ki bo lahko redno spremljal napredek gradnje, in urbanista, ki si bo pred načrtovanjem nove podzemeljske železnice lahko hitro izdelal aktualen zemljevid.

1.2 Cilji diplomske naloge

Cilj diplomske naloge je sestaviti mozaik iz zaporednih slik pridobljenih s kvadrokopterjem AR.Drone. Naloge se bomo lotili na dva načina.

Najprej bomo mozaik sestavili z uporabo tehnik računalniškega vida. Iz dveh zaporednih slik bomo najprej pridobili opisnike značilnih točk, ter z njihovo pomočjo eno od slik rektificirali, da bo njena ravnina preslikana v ravnino prve slike. Postopek bomo ponovili na vseh zaporednih parih slik, tako da bodo na koncu vse slike rektificirane na ravnino prve slike.

Nato bomo poskusili sestaviti mozaik s pomočjo navigacijskih podatkov pridobljenih iz kvadrokopterja. Ti podatki so hitrost, smer, naklon, itd. S pomočjo teh podatkov bomo rekonstruirali preleteno pot kvadrokopterja in, če bodo podatki dovolj natančni, izdelali mozaik.

Oba pristopa bomo nato ovrednotili in primerjali.

1.3 Sorodna dela

Mozaičenje je ena izmed prvih stvari, na katere pomislimo ob besedi računalniški vid. Zato je o različnih pristopih k mozaičenju napisanih veliko število člankov ([5], [6], [7], [8]). Zanimiva nadgradnja mozaičenja so stereo panorame. Članek [9] opisuje alogoritem za izgradnjo stereo panorame z eno samo kamero (v splošnem za stereo potrebujemo par kalibriranih kamer). Iz zaporedja slik, ki ga uporabimo za izdelavo mozaika, lahko izvlečemo tudi informacijo o globini [5] in sestavimo delni 3-D model scene [10]. Pri gradnji mozaikov ni pomemben samo končni rezultat, vendar tudi učinkovitosti algoritma. Članek [11] predstavi, kako lahko v realnem času zgradimo mozaik na mobilnih napravah, ki imajo omejeno procesorsko moč.

Tudi o kvadrokopterju AR.Drone je napisanih več člankov. [12] predstavlja kvadrokopterja AR.Drone kot platformo za raziskave in izobraževanje ter razpravlja o težavah, ki jih lahko srečamo pri uporabi kvadrokopterja kot robotsko platformo. Članek [13] predstavi enostaven a robusten navigacijski sistem na podlagi slik kamere za avtonomni kvadrokopter. Naprednejše navigacijske metode za kvadrokopter AR.Drone so opisane v [14], [15] in [16]. Zadnja izmed njih ([16]) tudi predstavi robustno metodo za izračun translacije in rotacije med posameznimi slikami in zgrajenim zemljevidom.

1.4 Zgradba diplomske naloge

Diplomska naloga je razdeljena na sedem poglavij.

V Poglavju 1 smo predstavili problem, ki ga bomo reševali, opisali cilje diplomske naloge, sorodna dela in strukturo diplomske naloge.

Poglavje 2 predstavlja mobilno platformo AR.Drone. Opisani so sestavni deli kvadrokopterja, ter njihove specifikacije. Nato predstavimo še uradno aplikacijo za mobilne naprave za upravljanje kvadrokopterja AR.Drone. Na koncu opišemo uradno programsko podporo, ki skrbi za upravljanje kvadrokopterja iz računalnika, ter ogrodje YADrone, ki omogoča kontrolo kvadrokopterja s programskim jezikom Java.

Poglavje 3 opisuje teoretično zasnovo, ki smo jo uporabili za sestavo mozaika. V tem poglavju bralec pridobi vso potrebno informacijo za dobro razumevanje gradnje zemljevida s tehnikam računalniškega vida.

Poglavje 4 je dvodelno. V prvem delu opišemo, kako smo zajemali video posnetek preletene poti s pomočjo mobilne aplikacije za upravljanje kvadrokopterja. V drugem delu postopoma opišemo, kako smo iz tako pridobljenega video posnetka sestavili mozaik.

Poglavje 5 je prav tako dvodelno. V prvem delu opišemo pridobivanje slikovnega materiala in navigacijskih podatkov s pomočjo ogrodja YADrone. Nato prikažemo, kako smo te podatke uporabili za rekonstrukcijo preletene poti.

V poglavju 6 ovrednotimo oba načina gradnje mozaikov in ju primerjamo. Razložili bomo tudi s kakšnimi slikami lahko zgradimo natančen mozaik in kakšne slike prispevajo k večjim napakam.

V poglavju 7 povzamemo naše ugotovitve in opišemo nekaj smernic za nadaljnje delo.

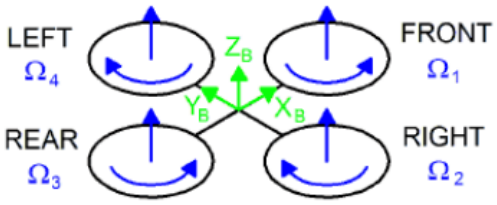
Poglavje 2

Opis mobilne platforme AR.Drone

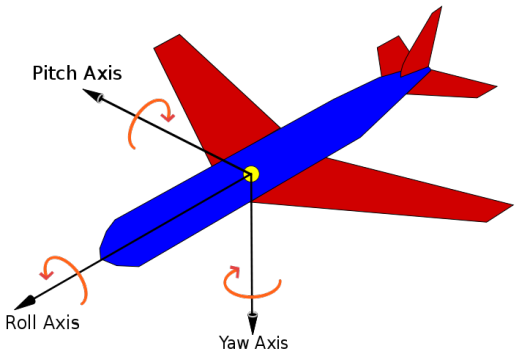
V tem poglavju bomo opisali platformo AR.Drone. V prvem delu predstavimo kvadrokopter AR.Drona s tehničnega vidika. Opisali bomo osnove delovanja ter specifikacije strojne opreme. V drugem delu bomo opisali uradno programsko podporo za programiranje aplikacij za kvadrokopter AR.Drone ter ogrodje YADrone, ki predstavlja dodatno abstrakcijo nad uradno programsko podporo. Nazadnje bomo opisali uradno mobilno aplikacijo za mobilne naprave z operacijskem sistemom Android s katero lahko upravljamo kvadrokopter in snemamo let z integriranimi kamerami.

2.1 Tehnične specifikacije

Kvadrokopterji so leteča plovila, ki za kontrolo leta spreminjajo kotno hitrost vrtenja štirih propelerjev (glej Sliko 2.1). Vsak par nasprotujočih si propelerjev (pari REAR, FRONT in LEFT, RIGHT na sliki) se vrti v isto smer. En par se vrti v smeri urnega kazalca, drugi pa v nasprotno smer. Propelerji povzročijo silo vzgona, ki mora biti vsaj tako veliko kot je sila teže kvadrokopterja. V nasprotnem primeru kvadrokopter ne more lebdeti v zraku. S spreminjanjem kotne hitrosti propelerjev se lahko plovilo obrne



Slika 2.1: Shema kvadrokopterja. Vsak par nasprotujočih si propelerjev se vrti v drugo smer. Vir: [17].



Slika 2.2: Prikaz rotacij v 3-D: naklon (ang. pitch), nagib (ang. roll) in odklon (ang. yaw). Vir: [18].

v treh dimenzijah relativno na svoj masni center. Te tri rotacije imenujemo naklon (ang. pitch), nagib (ang. roll) in odklon (ang. yaw). Model letala, z označenimi rotacijami, prikazuje Slika 2.2. Kvadrokopter se premakne vedno, ko kot naklona ali nagiba ni enak nič. S spreminjanjem kotne hitrosti levega in desnega propelerja v nasprotnih smereh se kvadrokopter nagne levo oz. desno. S spreminjanjem hitrosti sprednjega in zadnjega propelerja se kvadrokopter nagne naprej oz. nazaj. S spreminjanjem kotne hitrosti parov propelerjev v nasprotnih smereh se kvadrokopter odkloni iz trenutne smeri.

V tem diplomskem delu bomo uporabili kvadrokopter AR.Drone 2.0, ki

Slika 2.3: Levo je prikazan kvadrokopter AR.Drone z zaščitnim ogrodjem za let v notranjih prostorih. Desno je prikazan kvadrokopter z ogrodjem za let po zunanjih prostorih. Vir: [17].

ga izdeluje podjetje Parrot[1]. Za nalogo je primeren, ker je opremljen s kamerami in omogoča dokaj enostavno kontrolo leta. Kvadrokopter je prikazan na Sliki 2.3. Brez zaščitnega ogrodja za let v notranjih prostorih meri 451 milimetrov v dolžino in širino. Zaščitno ogrodje za let po notranjih prostorih doda 66 milimetrov tako v dolžino kot širino. Izdelano je iz vzdržljivega in lahkega stiropora in namenjeno za zaščito kvadrokopterja pred poškodbami zaradi padcev. Skelet, ki povezuje propelerje, je izdelan iz karbonskih vlaken. Skupna teža kvadrokopterja z zaščitnim ogrodjem za let po notranjih prostorih je 420 gramov, z ogrodjem za let po zunanjih prostorih pa 380 gramov. Propelerje poganjajo štiri 14.5W motorji. Energijo pridobijo iz polnilne litionske baterije s kapaciteto 1000mAh, ki zadošča za približno 12 do 15 minut leta.

Kvadrokopter AR.Drone ima integrirano logiko, ki skrbi za avtomatsko stabilizacijo leta. Zato ima integriran računalnik z 1Ghz procesorjem ARM Cortex A8, ki poganja operacijski sistem Linux 2.6.32. Pomnilnik DDR2 RAM je velikosti 1Gbit. AR.Drone ima integrirano kartico Wi-Fi, ki omogoča povezavo z napravo za nadzorovanje leta (na primer prenosni računalnik ali mobilni telefon). Vključen je tudi konektor USB 2.0, na katerega lahko

[1] `http://www.parrot.com/`

povežemo USB ključ.

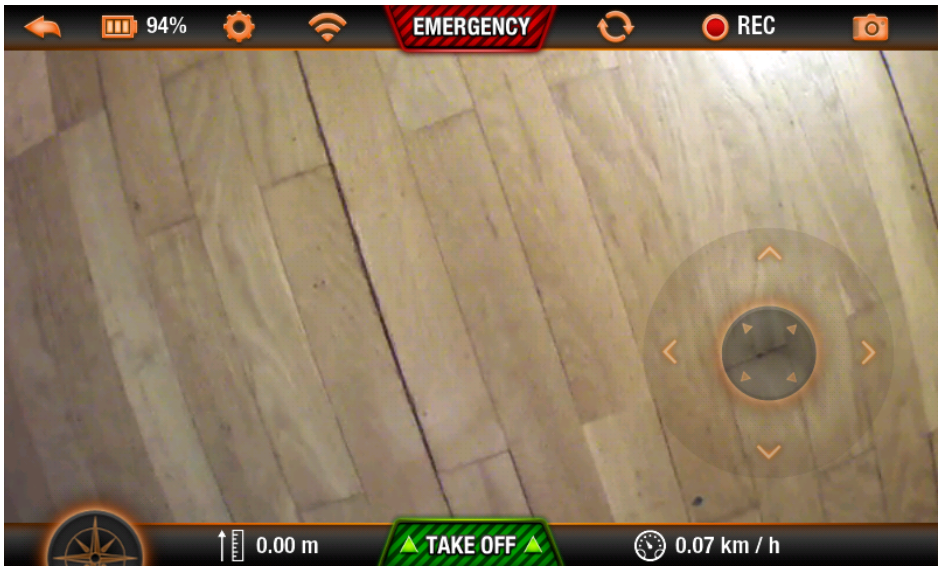
Za stabilizacijo leta AR.Drone uporablja informacijo o trenutni orientaciji v prostoru, ki jo dobi iz triosnega žiroskopa in pospeškometra. Za pomoč pri pristajanju kvadrokopter izkorišča podatke o višini, ki jih pridobi iz ultrazvočnega senzorja. Senzor je na spodnjem delu kvadrokopterja in je usmerjen navzdol. Ultrazvočni senzor deluje tako, da oddaja zvočno valovanje proti tlom, ki valove odbije nazaj k senzorju. Na podlagi časa potovanja valovanja sistem izmeri razdaljo do tal. Območje delovanja senzorja je med 20cm in 6m.

V kvadrokopter sta vgrajeni dve CMOS kameri. Prva je usmerjena navzdol, druga pa naprej. Slike kamere, ki je usmerjena navzdol, so namenjene meritvi hitrosti kvadrokopterja. Zato je hitrost zajema slik 60 slik na sekund, kar zmanjša učinek zamegljenosti (ang. motion blur). Ta kamera ima ločljivost 320×240 slikovnih elementov in vidni kot 64°. Prednja kamera ima ločljivost 1280×720 slikovnih elementov in hitrost zajemanja 30 slik na sekundo. Ima široki vidni kot 92°. Posnetki so zakodirani v H264 [19], slike pa formatu JPEG [20]. Posnetke lahko prenesemo preko povezave Wi-Fi na napravo, ki upravlja kvadrokopter, ali pa jih shranjujemo na USB ključ.

2.2 Mobilna aplikacija za upravljanje kvadrokopterja

AR.Drone nima namenskega daljinskega upravljalnika. Upravljanje leta kvadrokopterja je možno preko aplikacije za pametne mobilne naprave. Podjetje Parrot je razvilo aplikacije za naprave z operacijskim sistemom iOS in Android. V tem poglavju bomo opisali funkcionalnosti uradne mobilne aplikacije za naprave bazirane na operacijskem sistemu Android.

Najpomembnejša funkcionalnost aplikacije je upravljanje kvadrokopterja. Slika 2.4 prikazuje zaslonski posnetek pogleda med letom. Uporabniški vmesnik ima indikatorje za višino, hitrost, stanje baterije, povezavo Wi-Fi, gumb za slikanje in začetek snemanja. V nastavitvah lahko določimo, da se posnetki shranjujejo na USB ključ, določimo lahko maksimalno višino leta, vertikalno



Slika 2.4: Zaslonski posnetek aplikacije AR.FreeFlight na Android mobilni napravi. Aplikacija omogoča enostavno kontrolo leta kvadrokopterja, zajemanje slik in snemanje posnetkov s pomočjo kamer vgrajenih na AR.Drone.

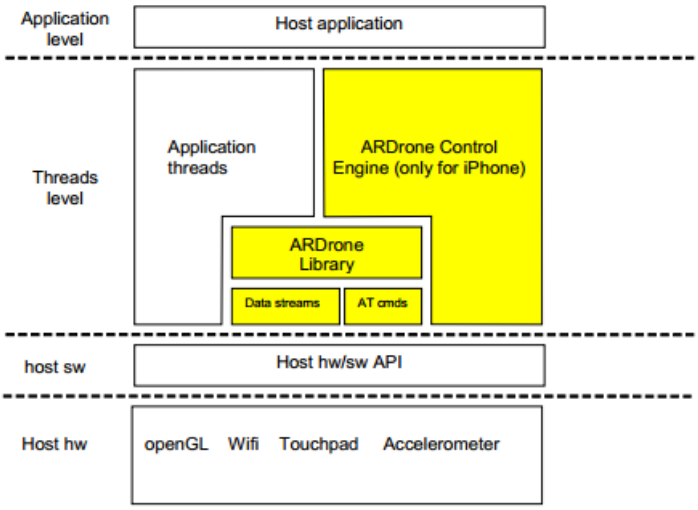
hitrost, hitrost obračanja in maksimalen kot naklona.

Druge funkcionalnosti so igre, ki temeljijo na razširjeni resničnosti (ang. augmented reality), galerija uporabniških posnetkov in AR.Drone Academy[2], ki ponuja shranjevanje letov na zemljevid in dodatne statistike leta.

2.3 Ogrodje YADrone

Podjetje, ki prodaja AR.Drone, ponuja uradno programsko podporo, s katero lahko pišemo aplikacije, ki komunicirajo s kvadrokopterjem. Te aplikacije imajo dostop do navigacijskih podatkov in slik iz kamer. Aplikacije, ki uporabljajo razvojno ogrodje AR.Drone SDK (ang. Software Development Kit), se izvajajo na prenosnem računalniku. Komunikacija s kvadrokopterjem poteka preko povezave Wi-Fi. Slika 2.5 prikazuje zgradbo aplikacije, zgrajena z razvojnim ogrodjem AR.Drone SDK. Najvišja plast predstavlja

[2] `http://ardrone2.parrot.com/ar-drone-academy/`



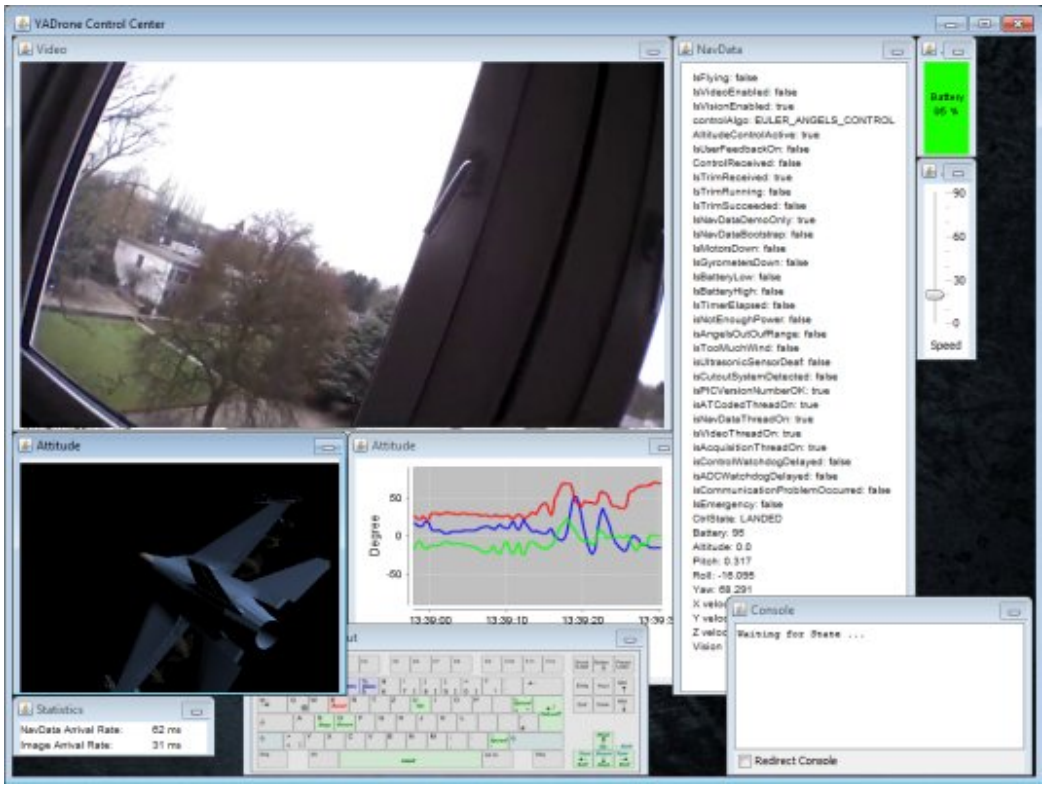
Slika 2.5: Arhitektura aplikacije za AR.Drone po plasteh. Vir: [17].

uporabniško aplikacijo, druga programe, ki so vključeni v SDK, tretja programsko opremo, ki je nameščena na kvadrokopterju, zadnja pa strojno plast.

Slabost AR.Drone SDK je pomanjkanje dobre dokumentacije. Zaradi tega je veliko razvijalcev razvilo knjižnice in ogrodja, ki olajšajo pisanje aplikacij za AR.Drone. Med njimi so ARDrone-Control-.NET[3], ki prinaša .Net SDK za Windows, gonilnik AR.Drone za ROS[4], AR.Drone orodje za Lab_VIEW[5], Windows AR Drone Control software[6], ARbDrone[7], ki omogoča programiranje za kvadrokopter AR.Drone v programskem jeziku Ruby, in še veliko drugih.

Za to diplomsko delo smo izbrali ogrodje YADrone[8], ki je napisano v programskem jeziku Java. Pomembne lastnosti tega ogrodja so, da ponuja enostavno kontrolo kvadrokopterja, podporo za branje navigacijskih podatkov in prejemanje slik iz kamer ter aplikacijo Control Center, ki vizualizira meritve

[3] https://github.com/shtejv/ARDrone-Control-.NET
[4] https://github.com/AutonomyLab/ardrone_autonomy
[5] http://sine.ni.com/nips/cds/view/p/lang/sl/nid/211837
[6] http://www.winardrone.com/
[7] https://github.com/bklang/ARbDrone
[8] http://vsisls1.informatik.uni-hamburg.de/projects/yadrone/



Slika 2.6: Zaslonski posnetek aplikacije Control Center (YADrone), ki omogoča upravljanje kvadrokopterja s tipkovnice in na vizualen način prikazuje meritve senzorjev. Vir: [21].

senzorjev in omogoča upravljanje AR.Drona s tipkovnice. Slika 2.6 prikazuje zaslonski posnetek aplikacije Control Center.

Poglavje 3

Teoretična podlaga

Opis gradnje mozaika smo razdelili na dva poglavja. V tem poglavju bomo predstavili teoretično podlago in algoritme, ki smo jih uporabili pri sestavljanju mozaikov. V Poglavju 4 predstavimo podrobnosti implementacije.

3.1 Mozaičenje

Proces gradnje mozaika lahko opišemo kot združevanje več zaporednih slik, da tvorijo logično celoto. Na grobo opišimo postopek za avtomatsko združevanje dveh slik:

1. **Odstranitev radialne distorzije**: S pomočjo notranjih parametrov kamere iz slik odstranimo radialno distorzijo (poglavje 3.2).

2. **Značilne točke**: Izračunamo značilne točke in njihove opisnike v obeh slikah (Poglavje 3.3).

3. **Korespondenčni pari značilnih točk** Na podlagi opisnikov značilnih točk poiščemo korespondenčne pare točk (Poglavje 3.4).

4. **Ocena homografije** Ocenimo transformacijo, ki preslika eno sliko v prostor druge (Poglavje 3.5).

5. **Robustna ocena** S pomočjo algoritma RANSAC filtriramo dobre korespondenčne pare točk (Poglavje 3.6).

6. **Optimizirana ocena** Ponovno ocenimo homografijo tako, da minimiziramo reprojekcijsko napako s pomočjo algoritma Levenberg-Marquardt (Poglavje 3.7).

7. **Ukrivimo slike** Na podlagi ocenjene homografije projeciramo prvo slike v prostor druge slike, da tvorimo brezhiben mozaik dveh slik (Poglavje 3.8).

Zgornji postopek ponovimo za vsak zaporedni par slik in tako dobimo brezhibni mozaik večjega števila slik.

Ko gledamo poljubno sliko zajeto s kamero opazimo, da liki v 3-D sceni ne ohranijo svojih lastnosti. Črte ostanejo ravne, ampak vzporedne črte se ne ohranijo. Dolžine, koti in razmerja dolžin se prav tako ne ohranijo. Transformaciji, ki preslika objekte iz scene na sliko, pravimo projektivna transformacija.

V evklidskih koordinatah ne moremo opisati točke v neskončnosti. Zato, ko imamo opravka s projektivnimi transformacijami raje uporabimo homogeni koordinatni sistem, kjer lahko predstavimo točke v neskončnosti. Homogene koordinate so sestavljene iz ene koordinate več kot evklidske. Da spremenimo evklidsko koordinato (x, y, z) v homogeno ji dodano 1 na zadnje mesto $(x, y, z, 1)$. Za homogene koordinate velja, da so točke, ki se razlikujejo preko skupnega faktorja, enakovredne. Na primer $(x, y, z, 1)$ in $(3x, 3y, 3z, 3)$ sta enakovredni, ker se razlikujeta samo preko skupnega faktorja 3. Točke v neskončnosti v tem koordinatnem sistemu predstavimo z ničlo na zadnjem mestu $(x, y, z, 0)$. Točko v homogenem sistemu lahko spremenimo v evklidsko tako, da delimo z zadnjim elementom. Na primer $(5, 3, 1, 4)$ v homogenem zapišemo kot $(\frac{5}{4}, \frac{3}{4}, \frac{1}{4})$ v evklidskem koordinatnem sistemu.

Projektivno transformacijo scene predstavimo z linearno transformacijo homogenih koordinat:

$$\mathbf{X}' = \mathbf{H}\mathbf{X} \,, \tag{3.1}$$

kjer je $\mathbf{X}$ vektor koordinat na sceni, $\mathbf{H}$ ne-singularna matrika in $\mathbf{X}'$ vektor s transformiranimi koordinatami.

3.2 Odstranitev radialne distorzije

V modelu idealne luknjičaste kamere (ang. pinhole camera model) velja, da so črte v 3-D slikane kot črte na sliki. Za realne kamere ta predpostavka ne drži. Črte v 3-D se pogosto slikajo v krivulje na sliki. Ta pojav je prikazan na sliki 3.1a. Temu pojavu pravimo radialna distorzija. Če za sestavo mozaika uporabimo slike z izrazito radialno distorzijo, bo končen mozaik zelo nenatančen, kot je prikazano na sliki 4.3.

Da bi lahko sestavili natančen mozaik, moramo slikam najprej odstraniti radialno distorzijo, ki jo povzročajo širokokotne leče, med njimi tista v sprednji kameri na kvadrokopterju AR.Drone. Za odstranitev radialne distorzije potrebujemo parametre, ki opisujejo kako se svetloba ukrivi, ko potuje skozi lečo kamere. Proces, s katerim pridobimo te parametre, imenujemo kalibracija kamere.

Cilj kalibracije kamere je izračun parametrov, ki opišejo preslikavo med referenčnimi 3-D koordinatami in 2-D koordinatami slikovne ravnine. Poznamo notranje (ang. intrinsic) in zunanje (ang. extrinsic) parametre. Zunanji parametri so rotacija in translacija kamere relativno na svetovni koordinatni sistem. Notranji parametri so goriščna razdalja (ang. focal length), koordinate glavne točke (ang. principal point), koeficienti poševnosti (ang. skew coefficients) in parametri distorzije. Gorišče kamere je točka, v kateri se zberejo svetlobni žarki, ko so ukrivljeni zaradi leče v kameri. Goriščna razdalja je razmak med to točko in lečo (označeno s črko **f** na sliki 3.3). Glavna točka je točka, v kateri se glavna os (ang. principal axis) seka s slikovno ravnino (označena s črko **p** na sliki 3.3). V idealni kameri pozicija središča kamere sovpada s centrom slikovne ravnine. Koeficienti radialne distorzije opisujejo ukrivljenost črt, ki so v realni ravnini ravne. Notranje parametre bomo uporabili za odstranitev radialne distorzije iz slik zajetih s sprednjo

(a) (b)

Slika 3.1: Slika (a) je popačena z radialno distorzijo. Slika (b) prikazuje isto sliko potem, ko smo radialno distorzijo odpravili.

Slika 3.2: Mozaik dveh slik z izrazito radialno distorzijo. Zaradi radialne distorzije ne moremo pravilno združiti slik, kar se opazi pri odstopanjih na gorah in poljih.

kamero AR.Drona.

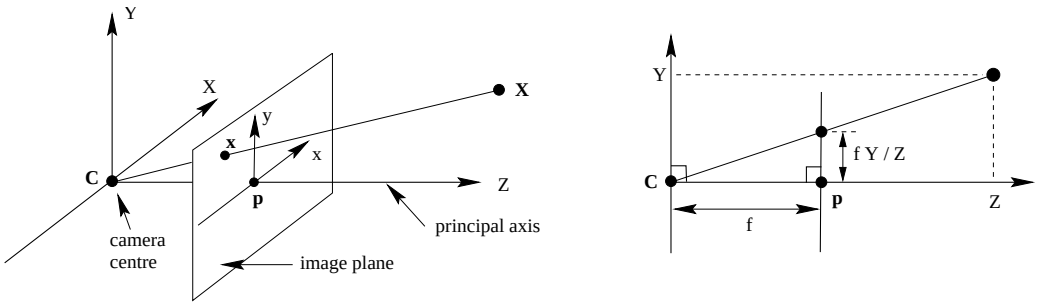
Podroben proces kalibracije kamere si lahko bralec prebere v [22].

Radialno distorzijo lahko odpravimo tako, kot je prikazano na shemi 3.4. Označimo koordinate točke na sliki, ki je zajeta z idealno lunknjičasto kamero, ki ne povzroča distorzije, z $(\tilde{x}, \tilde{y})$ merjeno v enotah goriščne razdalje. Točka na sliki, zajeti z realno kamero, s koordinatama (x_d, y_d), je povezana z idealno točko preko radialnega premika. Radialno distorzijo lahko modeliramo kot:

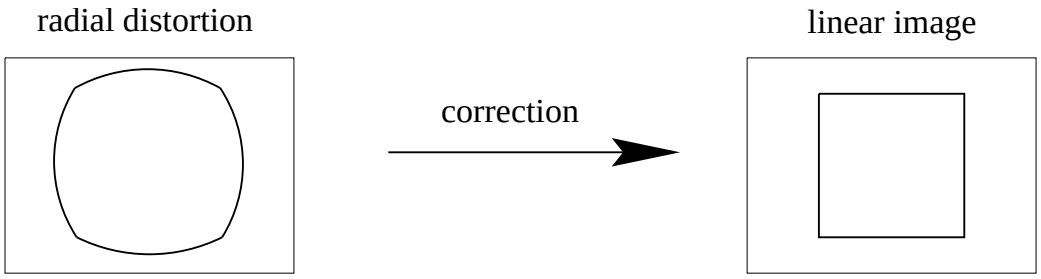
$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} , \tag{3.2}$$

kjer je $\tilde{r}$ radialna razdalja $\sqrt{\tilde{x}^2 + \tilde{y}^2}$ od centra radialne distorzije, $L(\tilde{r})$ pa funkcija radija, znana kot faktor distorzije.

Približek arbitrarni funkciji $L(\tilde{r})$ je lahko podan s Taylorjevo vrsto $L(\tilde{r}) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + \dots$ [23]. Koeficienti za odstranitev radialne distorzije $k_1, k_2, k_3, ..., x_c, y_c$ so del notranje parametrov kamere, in smo jih pridobili s



Slika 3.3: Geometrija luknjičaste kamere: **C** je center kamere, **p** pa glavna točka. **f** označuje goriščno razdaljo.



Slika 3.4: Slika kvadrata z izrazito radialno distorzijo je popravljena v sliko, ki bi bila zajeta s kamero z idealnimi linearnimi lečami. Povzeto po [23].

kalibracijo kamere.

Na podlagi funkcije $L(\tilde{r})$ lahko preslikamo slikovne elemente iz popačene slike v novo sliko brez radialne distorzije. Vendar se zaradi kvantiziranosti slike v končno število slikovnih elementov ne bodo vsi slikovni elementi popačene slike preslikali v točno določen slikovni element v novi sliki. Potrebno bo torej razdeliti barve med več slikovnih elementov, saj bi drugače dobili luknje v sliki. Pogosto boljša rešitev je opraviti obratno operacijo. Slikovne elemente za novo sliko pridobimo iz popačene slike preko inverzne funkcije $L^{-1}(\tilde{r})$. Če sedaj želimo pridobiti eno vrednost izmed več slikovnih elementov v popačeni sliki, lahko interpoliramo barvno vrednost izmed sosedov. V tem diplomskem delu smo uporabili drugi pristop in interpolirali barve z linearno interpolacijo [24]. Slika 3.1a po odstranitvi radialne distorzije je prikazana na Sliki 3.1b.

3.3 Značilne točke

Da bi lahko združili zaporedje slik v mozaik, moramo najprej biti sposobni pravilno združiti dve zaporedni sliki. V računalniškem vidu to naredimo tako, da na slikah poiščemo lokalne značilne točke (ang. keypoints), nato pa izberemo tiste, ki se pojavijo na obeh slikah. Ko najdemo dovolj takih točk, lahko izračunamo transformacijo (homografijo), ki preslika koordinate prve slike v prostor druge slike. Procesu izločanja značilnih točk na slikah pravimo ekstrakcija značilnih točk (ang. feature extraction). Značilne točke so razpoznavne strukture slikovnih elementov na slikah, ki jih opišemo matematično. Nizko nivojske značilne točke so primitivni geometrični elementi (npr. točke, črte, koti), višje nivojske pa lahko tudi predmeti (npr. stol, kamen). Izbira primernih značilnih točk je kritična za izdelavo mozaika.

Lokalna značilna točka je majhen del slike, ki se razlikuje od svoje okolice v barvi, teksturi ali intenziteti. Pogoste značilne točke so robovi in koti. Najpomembnejša lastnost detektorja značilnih točk je ponovljivost, kar pomeni da mora biti zmožen zaznati iste značilne točke v več slikah iste scene, ki se

razlikujejo po pogledu (ang. viewport) in osvetlitvi [16]. Druga zelo pomembna lastnost detektorja značilnih točk je različnost, kar pomeni da mora biti informacija shranjena v značilnih točkah čimbolj unikatna. Dobra značilna točka je invariantna, kar pomeni da razlike v pogledu kamere, osvetlitvi, rotaciji in merilu ne vplivajo na njen opis.

Scale-invariant feature transform (SIFT) [25] je verjetno najpogosteje uporabljen opisnik značilnih točk. Opisniki SIFT so invariantni na merilo, rotacijo, osvetlitev in pogled. Algoritem lahko orišemo tako. Najprej izračunamo notranjo predstavitev slike, da omogočimo invariantnost na merilo. Nato z aproksimacijo funkcije Laplace Gaussa najdemo značilne točke na sliki. Na razliki Gaussov (ang. Difference of Guassians) značilnih točk poiščemo tiste, ki so na maksimumih ali minimumih funkcije. Nato odstranimo slabe točke (npr. tiste na robovih in regijah z nizkim kontrastom). Za vsako značilno točko izračunamo orientacijo. Vsak nadaljnji izračun je relativen na to orientacijo, kar omogoča rotacijsko invariantnost. Končno se izračuna višje nivojski opisnik, ki unikatno identificira značilno točko. Ta opisnik je vektor 128 elementov sestavljen in histogramov orientacij gradientov.

3.4 Korespondenčni pari značilnih točk

Da bi lahko preslikali eno sliko na prostor druge slike s pomočjo opisnikov značilnih točk, moramo vedeti, katere značilne točke prve slike sovpadajo z značilnimi točkami druge slike. Ker je opisnik SIFT značilnih točk vektor, lahko najdemo korespondenčne točke z izračunom razdalje med dvema opisnikoma z L_2 normo, ki je definirana kot:

$$L_2(a,b) = \sqrt{\sum_{i=1}^{N} |a_i - b_i|^2}\,, \tag{3.3}$$

kjer je N demenzionalnost opisnika (v primeru opisnikov SIFT 128).

Pogosto se za iskanje korespondenčnih opisnikov uporablja metoda grobe

sile (ang. Brute-force deskriptor matcher) med dvema naboroma opisnikov. Za vsak opisnik v prvem naboru poiščemo najbližji opisnik v drugem naboru tako, da izračunamo L_2 normo med vsakim parom. Zaradi časnovne kompleksnosti $O(n^2)$ je ta način iskanja korespondenčnih parov počasen za velike nabore opisnikov [16].

V tem diplomskem delu uporabljamo približni algoritem Best-Bin-First (BBF) [26]. Približnost je v tem, da algoritem vrne najbližjega soseda z veliko verjetnostjo. Opisnik sovpada z drugim opisnikom samo, če produkt razdalje med njima in nekim pragom ni večji od razdalje prvega opisnika do vseh drugih opisnikov.

3.5 Ocena homografije

S korespondenčnimi pari točk lahko sedaj ocenimo transformacijo, ki preslika eno sliko v prostor druge. Za točko $\mathbf{x}$ v prvi sliki in korespondenčno točko $\mathbf{x}'$ v drugi sliki lahko napišemo transformacijo:

$$\begin{aligned} \mathbf{x}' &= \mathbf{H}\mathbf{x} \\ \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} &= \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \end{aligned} \tag{3.4}$$

kjer je $\mathbf{H}$ matrika dimenzij 3×3, ki združuje 8 parametrov, med njimi rotacijo, merilo, poševnost (ang. skew), translacijo in perspektivno transformacijo [27]. Točke $\mathbf{x}$ in $\mathbf{x}'$ smo zapisali v homogenih koordinatah.

Pomembna lastnost homografije $\mathbf{H}$ (in vsake linearne transformacije) je, da lahko združimo več transformacij s pred-množenjem. Inverz $\mathbf{H}^{-1}$ povzroči, da se izvede obratna transformacije (npr. rotacija s smeri urnega kazalca namesto v nasprotni smeri urnega kazalca).

Na podlagi korespondenčnih parov točk bi radi ocenili vseh 8 parametrov matrike H. Iz enačbe (3.4) dobimo sistem linearnih enačb z osmimi nez-

nankami:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1} \tag{3.5}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1} \tag{3.6}$$

Obe enačbi množimo z imenovalcem in ju preuredimo:

$$h_{11}x + h_{12}y + h_{13} - x'h_{31}x - x'h_{32}y - x' = 0 \tag{3.7}$$

$$h_{21}x + h_{22}y + h_{23} - y'h_{31}x - y'h_{32}y - y' = 0 \tag{3.8}$$

S 4 pari korespondenčnih točk lahko ocenimo vseh 8 parametrov homografije. Natančnost homografije pa lahko povečamo, če uporabimo večje število korespondenčnih točk. S tem dobimo predeterminiran sistem enačb:

$$\mathbf{Ah} = \mathbf{0};\ \mathbf{h} \neq 0$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \mathbf{0}\ , \tag{3.9}$$

ki ga lahko učinkovito rešimo preko dekompozicije po singularnih vrednosti (SVD) $\mathbf{A} \overset{\text{svd}}{=} \mathbf{UDV}^T$. Rešitev je enotski lastni vektor, ki pripada najmanjši lastni vrednosti $\mathbf{A}$ [27]. Elemente homografije dobimo iz zadnjega stolpca matrike lastnih vektorjev $\mathbf{V}$, in ker zahtevamo $h_{33} = 1$, normaliziramo z zadnjo vrednostjo.

3.6 Robustna ocena homografije

Pri izbiri korespondenčnih parov bomo pogosto dobili določen delež slabih parov, tj. nekatere značilne točke na eni sliki bodo bile združene s točkami v drugi sliki, s katerimi v resnici ne sovpadajo. Če homografijo ocenimo s slabimi pari, pa čeprav je delež le teh majhen, bo homografija neuporabna. Zato je zelo pomembno, da homografijo ocenimo samo na parih točk, ki v resnici sovpadajo. RANSAC (RAndom SAmple Consensus) je algoritem, s katerim lahko lokaliziramo te pare točk. Spodnja razlaga algoritma je povzeta po [28] in [22].

Cilj algoritma RANSAC je robustno prilagoditi nek model podatkovni zbirki S, ki vsebuje izstopajoče točke (ang. outliers). To so tiste točke, ki se modelu ne prilegajo, ne-izstopajoče (ang. inliers) pa tiste, ki se. Algoritem opišemo v petih korakih:

1. Ocenimo model z naborom s točk iz S, ki jih naključno vzorčimo.
2. Določimo nabor točk S_i, ki so v razdalji t od modela. Ta nabor sovpada z modelom, in predstavlja ne-izstopajoče točke.
3. Če je število ne-izstopajočih točk (nabor S_i) večje od praga T, ponovno ocenimo model, tokrat z vsemi točkami v S_i, in končamo.
4. Če je število točk v S_i manjše od T, ponovimo od začetka.
5. Po N ponovitvah izberemo največji nabor ne-izstopajočih točk S_i in z njim ocenimo model.

Minimalno število točk za ocenitev homografije je 4, zato je velikost nabora s enaka štiri. Število ponovitev ocenimo tako, da je z verjetnostjo $p = 0.995$ vsaj eden od naključnih vzorčenj brez izstopajočih točk. Naj bo w verjetnost, da je neka točka ne-izstopajoča. Vsaj N vzorčenj je potrebnih, da zadostimo enačbi $(1 - w^s)^N = 1 - p$. Iz enačbe izrazimo N:

$$N = log(1 - p)/log(1 - w^s) \tag{3.10}$$

Na primer za $w = 0.88$, $p = 0.995$ in $s = 4$ dobimo število ponovitev $N = 6$.

Ker se število ne-izstopajočih točk razlikuje od slike do slike, je bolje, da izračunamo število ponovitev adaptivno na podlagi deleža ne-izstopajočih točk, tako [22]:

1. $N = \infty$, število_vzorčenj = 0.
2. **Dokler** $N >$ število_vzorčenj **ponovi**
   (a) Vzorči in izračunaj število ne-izstopajočih točk.
   (b) $w = \frac{\text{število ne-izstopajočih točk}}{\text{skupno število točk}}$
   (c) Izračunaj novi N po enačbi (3.10) z verjetnostjo $p = 0.995$
   (d) Povečaj število_vzorčenj za 1.
3. Končaj

Razdalja t je merilo uspešnosti modela. V primeru homografije smo jo definirali tako:

$$t = d(\mathbf{X}_1, \mathbf{H}\mathbf{X}_2) + d(\mathbf{X}_2, \mathbf{H}^{-1}\mathbf{X}_1) \tag{3.11}$$

kjer je $\mathbf{X}_1$ vektor točk prve slike $\mathbf{X}_2$ vektor točk druge slike, $\mathbf{H}$ ocenjena homografija, $\mathbf{H}^{-1}$ njen inverz, d pa funkcija, ki izračuna vsoto kvadratov razlik podanih vektorjev.

Algoritem RANSAC ne samo da odstrani slabe korespondenčne točke, vendar tudi omogoča, da pravilno ocenimo homografijo, če slika ni povsem ravninska. Na primer, zamislimo si dve tlorisni sliki parkirišča, kjer so ob robovih posajena drevesa. Čeprav bi našli same pravilne korespondence med obema slikama, ne moremo pravilno oceniti homografije, ker je premik vrhov dreves, zaradi paralakse, večji od premika ceste. Algoritem RANSAC bo v tem primeru uspešno odstranil pare korespondenčnih točk, ki so na drevesih, tako da bodo ostale samo korespondenčne točke na cestišču. S takšnimi točkami pa lahko dobro ocenimo homografijo.

3.7 Optimizirana ocena homografije

Ocenjena homografija po principu najmanjših kvadratov, kot smo razložili v Poglavju 3.5, ni vedno optimalna. Za izboljšavo ocenjene homografije z nelinearno minimizacijo najmanjših kvadratov (ang non-linear least-squares minimization) uporabimo metodo Levenberg-Marquardt (LM)[29]. Za M ne-izstopajočih točk pridobljenih z algoritmom RANSAC lahko izračunamo izboljšano homografijo, ki minimizira enačbo (3.12) s pomočjo algoritma LM:

$$\sum_{i=1}^{M} |\mathbf{x}_i' - \mathbf{H}\mathbf{x}_i|^2 \ . \tag{3.12}$$

Algoritem LM interpolira med Gauss-Newtnovo metodo in Gradientno metodo. LM je bolj robusten kot Gauss-Newtnova metoda, kar pomeni da v večini primerov najde rešitev tudi, če je začetni približek daleč od končnega lokalnega minimuma.

Podrobno razlago algoritma Levenberg-Marquardt lahko bralec najde v [29].

Ko ocenjeno homografijo izboljšamo s tem algoritmom, jo lahko uporabimo za transformacijo ene slike v prostor druge, kot bomo opisali v naslednjem poglavju.

3.8 Ukrivljanje slik

Ko imamo ocenjeno homografije, moramo samo še poravnati drugo sliko v prostor prve slike. Ukrivljanje slik (ang. image warping) je transformacija, ki spremeni prostorsko konfiguracijo slike. To lahko formalno zapišemo:

$$\mathbf{I}' = \mathbf{f}(\mathbf{I}), \mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2 \ , \tag{3.13}$$

kjer je $\mathbf{I}$ slika, ki jo ukrivljamo, $\mathbf{I}'$ pa ukrivljena slika.

V tem poglavju bomo opisali, kako z ocenjeno homografijo ukrivimo sliko iz konfiguracijskega prostora ene slike v prostor druge slike [30].

Naj $\mathbf{X} = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \\ 1 & \dots & 1 \end{bmatrix}$ predstavlja pozicije točk v sliki $\mathbf{I}$, in $\mathbf{X}' = \begin{bmatrix} x'_1 & \dots & x'_n \\ y'_1 & \dots & y'_n \\ 1 & \dots & 1 \end{bmatrix}$ korespondenčne pozicije točk v ukrivljeni sliki $\mathbf{I}'$. Pozicije so podane v homogenih koordinatah. Tedaj enačba (3.13) postane:

$$\mathbf{X}' = \mathbf{H}\mathbf{X} \,, \tag{3.14}$$

kjer je z $\mathbf{H}$ označena matrika transformacije (homografija). Zaradi diskretnega značaja rasterskih slik nimamo zagotovila, da se vsak slikovni element preslika v en sam izhodni slikovni element. Zaradi tega ukrivljanje pogosto opravimo v obratni smer, t.j. iz izhodne slike v vhodno sliko. Ker je T matrika velikosti 3×3 in ima poln rang, lahko izračunamo inverzno transformacijo tako:

$$\mathbf{X} = \mathbf{H}^{-1}\mathbf{X}' \,. \tag{3.15}$$

Če se kateri izhodni slikovni element ne preslika v en sam vhodni element, njegovo vrednost interpoliramo iz vrednosti sosednjih slikovnih elementov. Pri ukrivljanju slik z ocenjeno homografijo smo uporabili bilinearno interpolacijo. Ta način interpolacije zahteva štiri sosedne vrednosti za koordinato, ki jo interpoliramo [24].

Poglavje 4

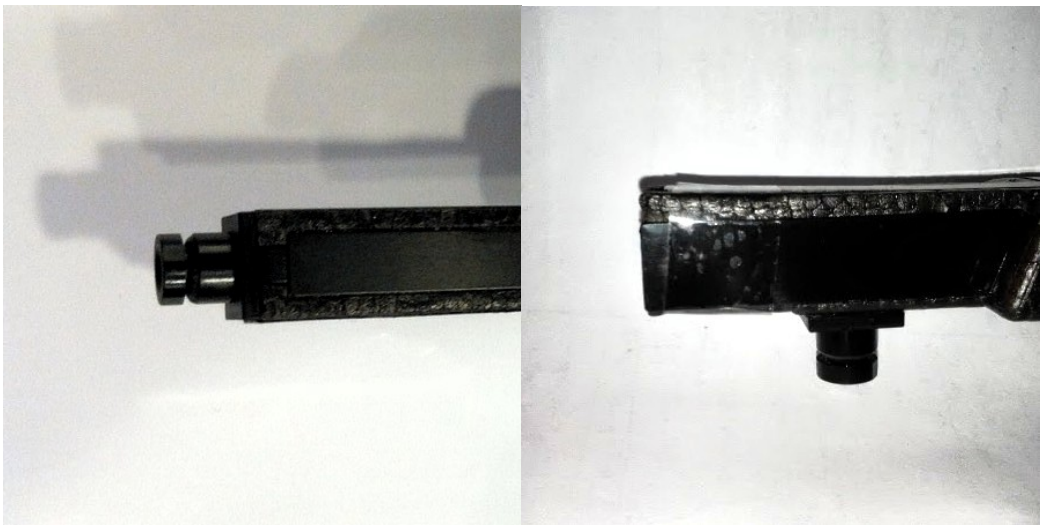
Gradnja mozaika

V tem poglavju bomo opisali postopek, ki smo ga uporabili za gradnjo mozaika s pomočjo tehnik računalniškega vida. Naprej bomo opisali zajem slik, ki smo ga izvedli s pomočjo uradne mobilne aplikacije AR.FreeFlight[1] za mobilne naprave z operacijskim sistemom Android, nato pa kako smo slike procesirali in zgradili mozaik.

4.1 Zajem in priprava slik

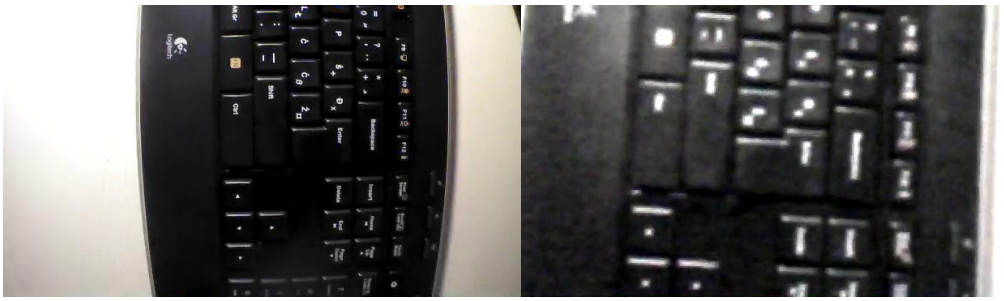
AR.Drone ima dve kameri. Prva je usmerjena navzdol in ima ločljivost 320×240 slikovnih elementov ter vidni kot 64°. Druga, ki je obrnjena vodoravno v smeri sprednjega dela kvadrokopterja, ima ločljivost 1280×720 slikovnih elementov in širši vidni kot 92°. Da bi izkoristili višjo ločljivost in vidni kot sprednje kamere za izdelavo mozaika, smo najprej predelali kvadrokopter in sprednjo kamero obrnili navzdol. Slika 4.1 prikazuje postavitev sprednje kamere pred in po predelavi. Slika 4.2 primerja vidni kot in ločljivost obeh kamer, pri čemer sta bile obe sliki zajeti iz iste višine (približno 40cm). Iz nje je jasno, da bomo s prednjo kamero zajeli več terena in slike bodo boljše ločljivosti.

[1] `https://play.google.com/store/apps/details?id=com.parrot.freeflight`



(a) (b)

Slika 4.1: Slika (a) prikazuje originalno postavitev sprednje kamere na kvadrokopterju AR.Drone. Da bi pridobili slike tal višje ločljivosti in širšega zornega kota, smo kamero obrnili navzdol (b).



(a) (b)

Slika 4.2: Primerjava kamer na kvadrokopterju AR.Drone: slika (a) je zajeta s sprednjo kamero ločljivosti 1280×720 in vidnim kotom 92°. Slika (b) je zajeta s spodnjo kamero kvadrokopterja z ločljivostjo 320×240 in vidnim kotom 64°. Sliki sta zajeti iz iste višine (približno 40cm).

S pomočjo uradne mobilne aplikacije AR.Drone, ki smo jo opisali v Poglavju 2.2, smo zajeli posnetek kamere kvadrokopterja. Da ne bi imeli težav s prenosom posnetka na mobilno napravo preko brezžične povezave Wi-Fi, smo na kvadrokopter priključili USB pomnilniški ključ in v Android aplikaciji omogočili shranjevanje posnetkov nanj.

Pridobljen posnetek je zakodiran v formatu H264 [19]. Ker naše nadaljnje delo poteka na posameznih slikah ne pa na celem posnetku, smo posnetek pretvorili v zaporedje slik. To smo storili s programom *avconv*[2]. *avconv* je hiter pretvornik video in avdio vsebin. Natančen opis vseh njegovih funkcionalnosti je na voljo na spletni strani programa. V nadaljnjem opisu se bomo omejili samo na funkcionalnost, ki smo jo neposredno uporabili za pretvorbo video posnetka v zaporedje slik.

Listing 4.1: Ukaz za pretvorbo video posnetka v zaporedje slik

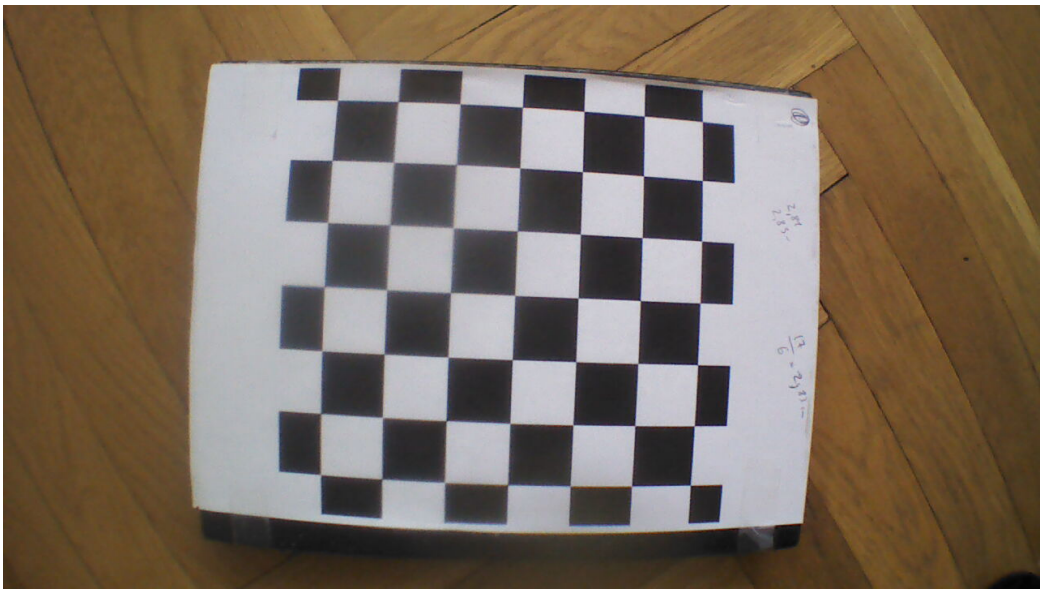
```
avconv -i video.mp4 -r 6 -vsync 1 -q 1 -an -y video/%5d.png
```

Z ukazom 4.1 pretvorimo video posnetek imenovan video.mp4 v zaporedje slik v formatu PNG (Portable Network Graphics). Slike bodo shranjene v mapi imenovani *video*; prva slika bo imela ime 00001.png, druga 00002.png, itd. Format PNG smo izbrali zato, ker uporablja postopek stiskanja podatkov brez izgub [31].

Ime vhodnega video posnetka podamo za mnemonikom `-i`. Mnemonik `-r` določa vzorčevalni čas (ang. frame rate) oziroma število slik na sekundo video posnetka. Pri izbiri vrednosti tega paramera je pomembno, da se zaporedne slike prekrivajo vsaj za približno 30%. V kombinaciji z opcijo `-q` uporabimo `-vsync`, ki določi metodo video sinhronizacije (ang. video sync method). Opcija `-q` določa kvaliteto slik. Vrednost se lahko nahaja med 1 (najboljša kvaliteta) in 21 (najslabša kvaliteta). Opcija `-an` izključi zvok, ki je za nas nepomemben. Opcija `-y` pove naj izhodne slike prepišejo morebitne obstoječe datoteke z istim imenom.

Zaradi slabe leče sprednje kamere na kvadrokopterju AR.Drone so slike

[2] `http://libav.org/avconv.html`



Slika 4.3: Sprednja kamera na kvadrokopterju AR.Dronu povzroča veliko radialne distorzije na slikah. Ukrivljenost ravnih črt je jasno vidna na šahovnici.

delno ukrivljene. To se vidi na Sliki 4.3 v ukrivljenih črtah. Temu pojavu pravimo radialna distorzija. Da bomo lahko uspešno sestavili mozaik iz slik kvadrokopterja, moramo najprej odpraviti radialno distorzijo iz slik. Za to potrebujemo notranje parametre kamere.

4.2 Kalibracija kamere

Za kalibracijo kamere smo uporabili Matlab Calibration Toolbox[3] (orodja za kalibracijo Matlab). Z njim lahko na podlagi vhodnih slik izračunamo notranje in zunanje parametre kamere. Program prejme več slik, ki prikazujejo šahovnico slikano z več zornih kotov in razdalj. Nato smo za vsako sliko ročno označili kote šahovnice in določili število ter velikost kvadratkov na šahovnici. Matlab Calibration Toolbox nato izračuna vse parametre kamere. Slika 4.4 prikazuje 32 slik, ki smo jih uporabili za kalibracijo. Te slike smo pridobili

[3] http://www.vision.caltech.edu/bouguetj/calib_doc/

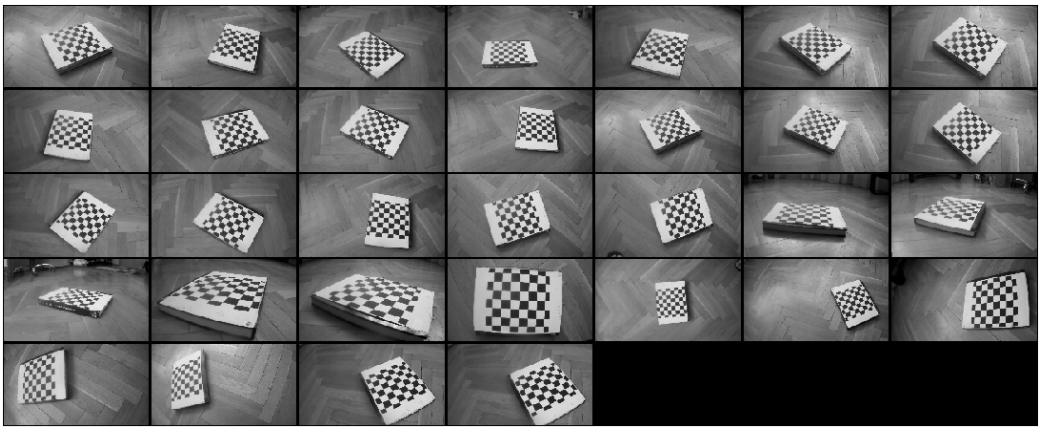
Parameter	Vrednost
Goriščna razdalja f_x	1121.04774 ± 3.41195
Goriščna razdalja f_y	1122.10740 ± 3.31091
Glavna točka c_x	658.22148 ± 6.64335
Glavna točka c_y	330.55202 ± 5.85748
Koeficient poševnosti x	0.00000
Koeficient poševnosti y	0.00000
Distorzija k_1	-0.52885 ± 0.00844
Distorzija k_2	0.32422 ± 0.03388
Distorzija k_3	0.00018 ± 0.00179
Distorzija k_4	-0.00151 ± 0.00112
Distorzija k_5	0.00000

Tabela 4.1: Notranji parametri sprednje kamere kvadrokopterja pridobljeni z orodjem Matlab Calibration Toolbox.

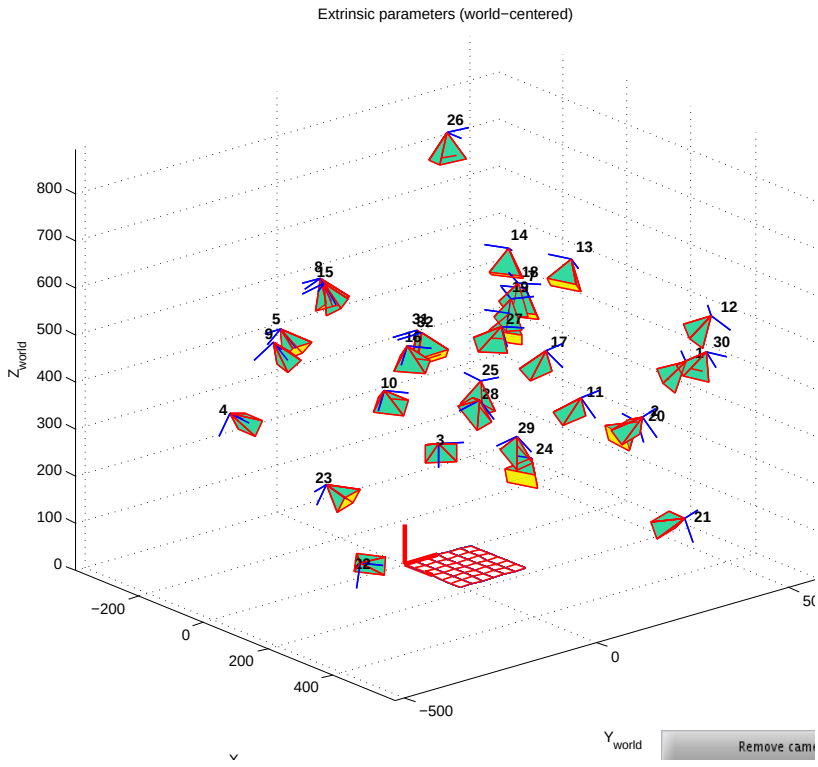
iz sprednje kamere kvadrokopterja AR.Drona, ki povzroča izrazito radialno distorzijo. Tabela 4.1 prikazuje izračunane notranje parametre. Slika 4.5 grafično prikazuje zunanje parametre postavitev kamer.

4.3 Odstranitev radialne distorzije

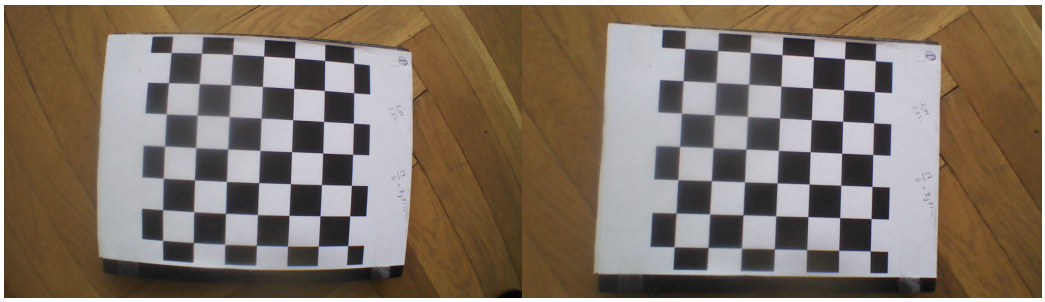
Ko imamo izračunane notranje parametre kamere, lahko odstranimo radialno distorzijo iz slik tako, kot smo opisali v Poglavju 3.2. Slika 4.6a prikazuje sliko, kateri želimo odstraniti radialno distorzijo. Slika 4.6b pa prikazuje isto sliko potem, ko smo distorzijo odpravili. Slikovne elemente, ki so se pri odpravi distorzije preslikali v območje zunaj originalnih dimenzij slike, smo odstranili. Posledica odprave radialne distorzije je, da se robovi slike raztegnejo. Da ne bi vključili tako raztegnjenih robov v končni mozaik, smo vsem slikam odrezali nekaj slikovnih elementov ob robovih.



Slika 4.4: 32 slik, ki smo jih uporabili za kalibracijo kamere. Slike so pridobljene s sprednjo kamero kvadrokopterja AR.Drone.



Slika 4.5: 3-D prikaz zunanjih parametrov kamere za 32 slik, ki smo jih uporabili za kalibracijo kamere.



(a) (b)

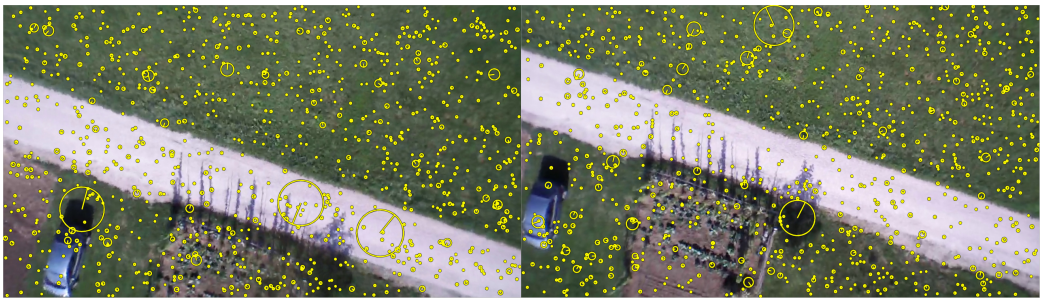
Slika 4.6: Slika (a) prikazuje originalno sliko zajeto s sprednjo kamero kvadrokopterja. Šahovnica na sliki je ukrivljena. Slika (b) prikazuje isto sliko po odstranitvi radialne distorzije. Stranice šahovnice niso več ukrivljene.

4.4 Detekcija korespondenčnih točk

Na popravljenih slikah lahko sedaj poiščemo značilne točke in njihove opisnike. Uporabili smo implementacijo opisnikov SIFT knjižnice VLFleat[4], ki implementira priljubljene algoritme računalniškega vida. Za večjo učinkovitost in kompatibilnost je napisana v programskem jeziku C in ponuja vmesnik za Matlab, v katerem je napisan programski del diplomskega dela. Proces izločanja značilnih točk in izračun opisnikov smo opisali v Poglavju 3.3. Sliki 4.7 prikazujeta naključnih 1000 izločenih značilnih točk na dveh zaporednih slikah zajetih s sprednjo kamero kvadrokopterja. Zaradi teksture na slikah so te točke razpršene po celi sliki. Število vseh zaznanih značilnih točk na Sliki 4.7a je 6613 in 6914 na Sliki 4.7b.

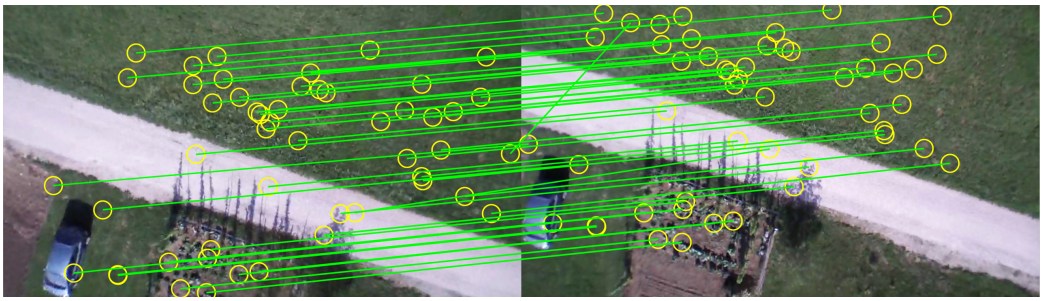
Sedaj želimo poiskati tiste točke na prvi sliki, ki sovpadajo s točkami na drugi sliki. Algoritem za izbiro korespondenčnih parov točk smo opisali v Poglavju 3.4. Od skoraj 7000 zaznanih značilnih točk na vsaki sliki smo skupaj dobili 551 korespondenčnih parov. Slika 4.8 prikazuje 50 povezanih korespondenčnih točk. Iz slike je razvidno, da smo zajeli tudi nekaj slabih korespondenčnih parov. Homografija ocenjena s takšnimi pari bi bila neu-

[4] `http://www.vlfeat.org/`



(a) (b)

Slika 4.7: Prikazanih je naključnih 1000 zaznanih značilnih točk na dveh zaporednih slikah.

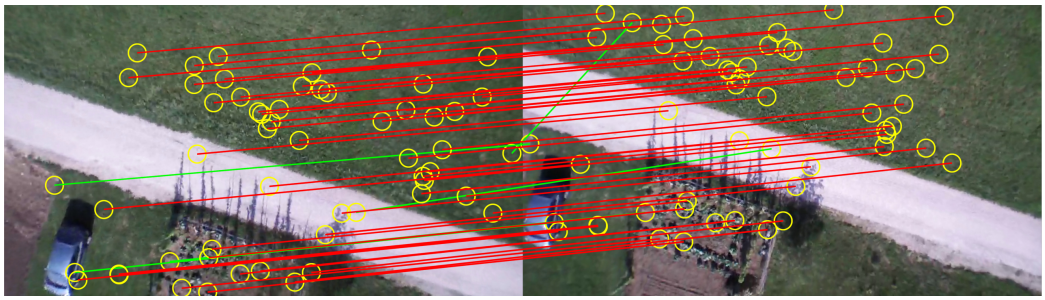


Slika 4.8: Prikazanih je 50 naključno izbranih korespondenčnih parov točk pred filtriranjem z algoritmom RANSAC.

porabna. Zato moramo pare dodatno filtrirati, kar lahko naredimo z algoritmom RANSAC, ki smo ga opisali v Poglavju 3.6.

Korespondenčne pare točk filtrirane z algoritmom RANSAC prikazuje Slika 4.9. Z zeleno so narisani izstopajoči, z rdečo pa ne-izstopajoči pari. Od vseh 551 korespondenčnih parov smo jih po filtriranju obdržali 448. Delež ne-izstopajočih parov na teh dveh slikah je 88.566%. Kljub 11.4 % deležem izstopajočih točk, je algoritem RANSAC z verjetnostjo $p = 0.995$ ločil ne-izstopajoče od izstopajočih. Adaptivni algoritem RANSAC je izvedel samo 14 vzorčenj.

Iz Slike 4.9 je razvidno, da je zaznal tudi nekaj dobrih korespondenčnih parov kot izstopajoče. Ker imamo dovolj dobrih korespondenčnih parov točk



Slika 4.9: Prikazanih je 50 naključno izbranih korespondenčnih parov točk po filtriranjem z algoritmom RANSAC. Z rdečo barvo so prikazani dobri korespondenčni pari (ne-izstopajoči), z zeleno pa tisti, ki jih je algoritem RANSAC ocenil za slabe (izstopajoči).

za izračun homografije, nas nekaj napačnih negativov ne moti. Veliko bolje je zavreči nekaj dobrih korespondenc, kot pa ohraniti eno samo slabo korespondenco.

4.5 Ukrivljanje slik

Na podlagi nabora korespondenčnih parov, ki jih je algoritem RANSAC določil za ne-izstopajoče, izračunamo homografijo, kot smo opisali v Poglavju 3.5. Ocenjeno homografijo uporabimo kot vhodni približek za algoritem Levenberg-Marquardt, ki oceno homografije izboljša.

Uporabili smo implementacijo algoritma Levenberg-Marquardt, ki je vključena v Matlab. Klic metode zgleda tako:

Listing 4.2: Primer klica algoritma Levenberg-Marquardt

```
h=lsqcurvefit(@fun, h0 ,pts1, pts2, lb, ub, option)
```

Parametri, ki jih funkcija zahteva so naslednji:

1. fun: funkcija, ki določi povezavo med originalnimi in transformiranimi koordinatami $\mathbf{X}$ in $\mathbf{X}' = \mathbf{HX}$. Kot vhodne parametre sprejme dva vektorja. Prvi je homografija v stolpični obliki, drugi pa vektor koordinat

značilnih točk. V funkciji vrnemo koordinate, transformirane s podano homografijo.

2. h0: začetna ocena homografije v stolpični obliki, ki smo jo ocenili po principu najmanjših kvadratov.

3. pts1: vektor ne-izstopajočih značilnih točk prve slike.

4. pts2: vektor ne-izstopajočih značilnih točk druge slike.

5. lb: Spodnja meja za ocenjeno homografijo. Mi podamo prazno matriko, kar pomeni, da ne določimo meje.

6. ub: Zgornja meja za ocenjeno homografijo. Mi podamo prazno matriko, kar pomeni, da ne določimo meje.

7. options: V opcijah podamo ime algoritma, ki ga želimo uporabiti: optimset('Algorithm','Levenberg−Marquardt');

Ko imamo ocenjeno homografijo, moramo z njo ukriviti sliko. V Poglavju 3.8 smo opisali matematični vidik problema. Sedaj bomo razložili nekatere podrobnosti implementacije. Pri ukrivljanju slike smo se odločili, da ne bi radi odrezali delov slike, ki segajo izven velikosti originalne slike. Zato smo se soočili z dvema problemoma. Prvi je, kako vnaprej izračunati dimenzije ukrivljene slike, drugi pa za koliko bomo morali nadoknaditi pozicijo slike, ko jo dodamo v mozaik.

Prvi problem smo rešili tako, da smo najprej izračunali točke, v katere se preslikajo vogali slike. Koordinate vogalov na ukrivljeni sliki izračunamo tako, da z njimi pomnožimo homografijo $\mathbf{X}' = \mathbf{HX}$, kjer je $\mathbf{X}$ matrika dimenzij 3×4, ki vsebuje koordinate vogalov v homogenem formatu. Nato smo homogene koordinate pretvorili nazaj v evklidske, z deljenjem z zadnjim elementom koordinate. Na primer, slike, ki smo jih uporabili v prejšnjih primerih, so velike 1280×720 slikovnih elementov in njihove koordinate vogalov v evklidskem koordinatnem sistemu so (1, 1), (1, 720), (1280, 1), (1280, 720). Te koordinate se z ocenjeno homografijo preslikajo na (-134, -97), (-92,

605), (1116, -91) in (1127, 579). Dimenzije matrike, ki bo hranila ukrivljeno sliko, definiramo tako:

Listing 4.3: Izračun velikosti ukrivljene slike na podlagi vogalov originalne slike.

```
% w in h sta sirina in visina originalne slike.
corners = [ 1 1 w w;
       1 h 1 h;
       1 1 1 1];

% Transformacija koordinat z ocenjeno homografijo H.
cp = H * corners;

% Izracunane koordinate pretvorimo v evklidski koordinatni
   sistem.
cp = round(cp./repmat(cp(3,:),3,1));

% Poiscemo najvecjo in najmanjso vrednost koordinat.
minX = min(cp(1, :));
maxX = max(cp(1, :));
minY = min(cp(2, :));
maxY = max(cp(2, :));

% Sestavimo matriko za ukrivljeno sliko.
Xpr = minX : maxX;
Ypr = minY : maxY;
[Yp, Xp] = ndgrid(Ypr, Xpr);
```

Ker dimenzije originalne in ukrivljene sliki niso enake, bomo morali ukrivljeno sliko premakniti za primerno razdaljo, ki je za vsako dimenzijo (x in y) enaka najmanjši vrednosti izmed koordinat ukrivljenih vogalov. Za interpolacijo barv smo uporabili bilinearno interpolacijo, ki poskrbi da so barve porazdeljene med sosednjimi slikovnimi elementi. Ker poznamo velikost in

Slika 4.10: Mozaik dveh zaporednih slik. Z ocenjeno homografijo smo drugo sliko ukrivili v prostor prve slike.

pozicijo slike, jo lahko sedaj vključimo na mozaik. Slika 4.10 prikaže dobljeni mozaik.

Nadgradnja postopka, da podpira mozaičenje več kot dveh slik, je dokaj enostavna. Ker lahko homografije združimo s pred-množenjem, lahko globalno homografijo, ki definira transformacijo med prvo in tretjo sliko v zaporedju, izračunamo $\mathbf{H}_{global} = \mathbf{H}_{i-2}\mathbf{H}_{i-1}\mathbf{H}_i$, kjer podpisano število predstavlja indeks predhodne lokalne homografije glede na lokalno homografijo zadnjega para slik. Problem lahko posplošimo na poljubno sliko v zaporedju:

$$\mathbf{H}_{global} = \mathbf{H}_{i-n}\mathbf{H}_{i-n+1}\ldots\mathbf{H}_{i-1}\mathbf{H}_i = \prod_{j=-n}^{0}\mathbf{H}_{i+j}\ . \tag{4.1}$$

Ker je produkt matrik asociativen, ni potrebno da hranimo vso zgodovino lokalnih homografij ampak samo eno globalno, ki je produkt vseh predhodnih. Vsakič, ko zaporedju dodamo sliko, je dovolj, da izračunano lokalno homografijo pred-množimo z globalno. Opisani postopek nam omogoča, da

(a) (b)

Slika 4.11: Na levi je mozaik štirih slik.Vsaka naslednja slika je ukrivljena v prostor prve slike v zaporedju. Zaradi nizke kvalitete slike ob robovih mozaik ni idealen, kar se vidi v okolici avtomobila (b).

je časovna kompleksnost dodajanja slik na mozaik konstantna. Slika 4.11 predstavlja mozaik sestavljen iz zaporedja štirih slik.

Zaradi nizke kvalitete slik in vinjetenja lahko na mozaikih pogosto opazimo robove posameznik slik. Slaba kvaliteta slik prispeva tudi k slabše ocenjeni homografiji. Oboje je jasno vidno na Sliki 4.11b kjer je avto nekoliko zamaknjen. Mozaike lahko izboljšamo, če pred izračunom značilnih točk slikam odstranimo nekaj slikovnih elementov ob robovih. Slika 4.12 prikazuje mozaik istih štirih slik, kjer smo slikam ob robovih odrezali pas širok 50 slikovnih elementov. S tem smo tudi delno odstranili rahlo vinjetenje.

4.6 Zaznava slabih homografij

Kvaliteta posnetka, zajetega s kamero kvadrokopterja AR.Drone, se občasno zelo poslabša. Slika 4.13a prikazuje posnetek, ki je normalne kvalitete, 4.13b pa nižje kvalitete. Obe sliki sta del video posnetka zajetega s sprednjo kamero kvadrokopterja. V video posnetku sta se pojavili manj kot pol sekunde

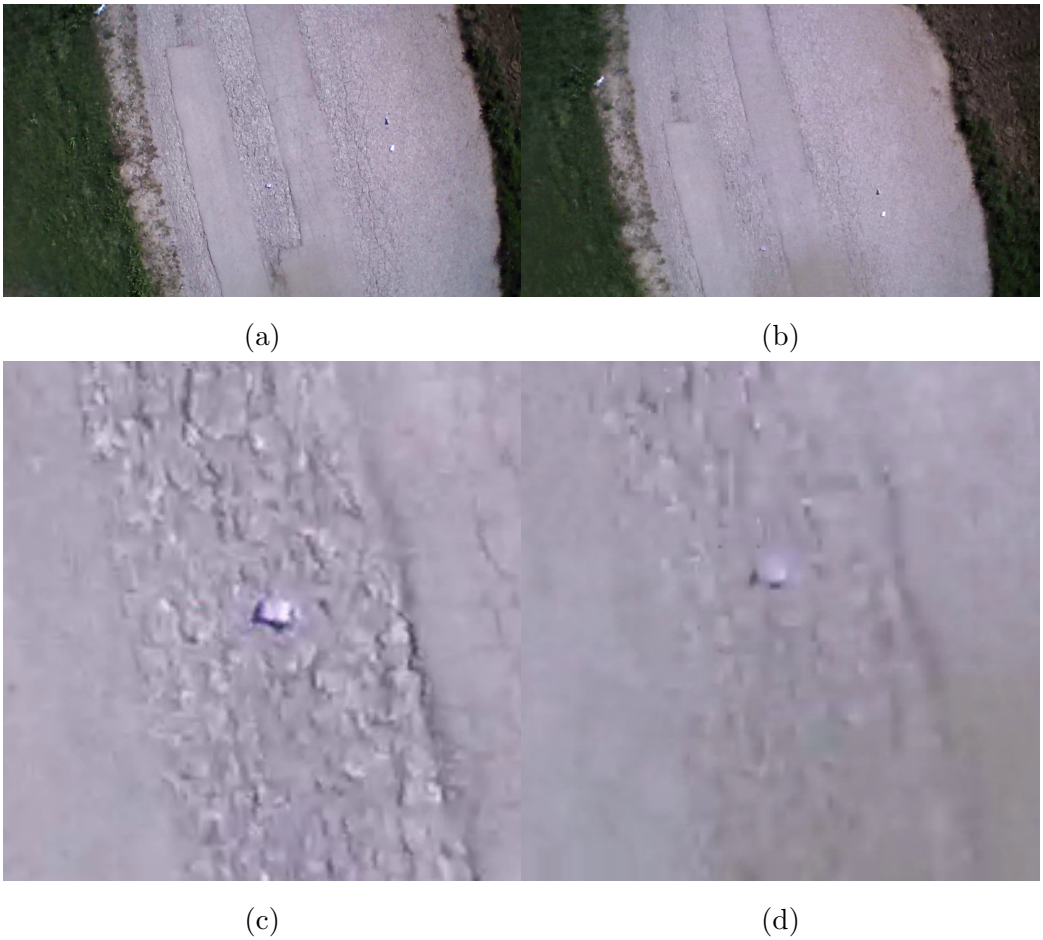
Slika 4.12: S tem, da smo slikam odstranili 50 slikovnih elementov roba smo izboljšali izdelan mozaik. Avto je sedaj pravilno sestavljen, robovi slik pa niso več tako jasno opazni.

narazen. Zaradi spremenljive kvalitete slik, so lahko izračunani korespondenčni pari slabi. Algoritem RANSAC slabe pare odstrani, vendar je včasih delež izstopajočih parov tako velik, da nam ostane zelo malo parov točk, s katerimi lahko ocenimo homografijo. Posledično je lahko homografija slabše ocenjena. Kot smo opisali v Poglavju 4.5 lokalne homografije združujemo, da dobimo globalno transformacijo trenutne slike v zaporedju. Lokalne napake se tako prenesejo v vse naslednje slike. Ena sama slabo ocenjena homografija v zaporedju pokvari mozaik. Da bi to preprečili, smo empirično poiskali nekaj pravil, s katerimi zaznamo slabo ocenjene homografije in pripadajoče slike preskočimo.

Prvo takšno pravilo je, da odstranimo slike pri katerih je delež dobrih korespondenčnih parov točk (ne-izstopajoči pari) manjši od nekega praga γ_1. Empirično smo ta prag postavili na 0.6. Tako ohranimo vse slike, kjer je delež ne-izstopajočih točk velik (delež ne-izstopajočih točk je pri "dobrih" zaporednih slikah skoraj vedno nad 0.85).

Druga metrika, ki smo jo uporabili da bi preprečili apliciranje slabo ocenjene homografije na mozaik, je absolutno odstopanje elementov homografije. Na podlagi pomena posameznih elementov homografije in empiričnega testiranja smo določili nekaj pravil, ki jih prikažemo v tabeli 4.2. Ta pravila veljajo, ko je ocenjena homografija normalizirana, tako da je $h_{33} = 1$. Slike, ki povzročijo lokalno transformacijo, ki ni v skladu s temi pravili, preskočimo.

Zgornja pravila v velikem število primerov pripomorejo k boljšemu mozaiku. Pravila delujejo, ker je med zaporednimi slikami sprememba v translaciji, rotaciji in merilu majhna.



(a) (b)

(c) (d)

Slika 4.13: Variabilna kvaliteta posnetka: nepredelani sliki (a) in (b) sta bile zajeti s sprednjo kamero kvadrokopterja AR.Drone z locljivostjo 1280×720. Sliki (c) in (d) sta izreza zgornjih posnetkov velikosti 200×200 slikovnih elementov na katerih se vidi cigaretna škatlica. Posnetka (a) in (b) sta bila zajeta manj kot pol sekunde narazen vendar je kvaliteta posnetka (a) mnogo boljša. Sliki (b) in (d) sta tako zamegljeni, da je cigaretna škatlica komaj prepoznavna.

Tabela 4.2: Seznam pravil na podlagi absolutnega odstopanja elementov homografije **H**. Prvo število v oklepajih predstavlja vrstico, drugo pa stolpec matrike **H**.

Pravilo	Dosežek
$0.4 < \mathbf{H}(1,1) < 1.4$	Omejitev spremembe v merilu
$0.4 < \mathbf{H}(2,2) < 1.4$	Omejitev spremembe v merilu
$0 < \mathbf{H}(1,3) < 1000$	Omejitev spremembe v translaciji
$0 < \mathbf{H}(2,3) < 1000$	Omejitev spremembe v translaciji
$0 < \mathbf{H}(3,1) < 0.0004$	Omejitev spremembe v projektivni transformaciji
$0 < \mathbf{H}(3,2) < 0.0004$	Omejitev spremembe v projektivni transformaciji
$0 < \mathbf{H}(1,2) < 0.2$	Omejitev poševnosti (ang. skew)
$0 < \mathbf{H}(2,1) < 0.2$	Omejitev poševnosti

Poglavje 5

Rekonstrukcija poti iz navigacijskih podatkov

V tem poglavju bomo opisali poskus gradnje mozaika samo z uporabo navigacijskih podatkov, ki jih pridobimo iz kvadrokopterja. To so podatki o hitrosti, orientaciji, itd. Najprej bomo opisali program, ki temelji na ogrodju YADrone, s katerim smo pridobili navigacijske podatke iz kvadrokopterja. V drugem delu pa bomo opisali, kako lahko iz teh podatkov izračunamo pozicijo posameznih slik za gradnjo mozaika.

5.1 Zajem slik in navigacijskih podatkov

Ko smo gradili mozaik s tehnikam računalniškega vida, smo potrebovali samo zaporedje slik. Zaradi enostavnosti in lažje kontrole kvadrokopterja v letu smo za shranjevanje posnetkov uporabili uradno Android mobilno aplikacijo. Ker hočemo sedaj raziskati še možnost izdelave mozaika brez uporabe tehnik računalniškega vida za iskanje korespondenčnih točk med slikami, smo razvili program s katerim lahko upravljamo kvadrokopter, hkrati pa poleg slik shranjujemo navigacijske podatke.

Program je napisan v ogrodju YADrone, ki smo ga opisali v Poglavju 2.3. Podatki, ki smo jih pridobili iz kvadrokopterja, so slika iz kamere, naklon,

nagib, odklon, višina, tri komponente vektorja hitrosti kvadrokopterja in stanje baterije.

Zajemanje podatkov je asinhrono, kar pomeni, da ne sprejmemo hkrati slike in pripadajoče kontrolne podatke. Ker prejeti podatki ne vsebujejo časa, ko so se podatki dejansko izmerili oz. slika zajela, se pojavi problem korespondence med meritvami in slikami, ki ga ne moremo uspešno razrešiti.

Ogrodje YADrone pošilja posnetek kot zaporedje slik. Prejete slike zakodiramo v format JPEG [20] in shranimo na disk. Prenos slik preko povezave Wi-Fi, kodiranje v format JPEG in zapis slike na trdi disk je časovno potraten proces, zaradi katerega smo imeli težave, če smo hoteli shraniti vse prejete slike. Zato smo se odločili, da shranjujemo samo vsako n-to sliko (n se je gibal okrog 3, odvisno od hitrosti leta).

Navigacijske podatke in ime slike na trdem disku smo shranjevali v datoteko v formatu CSV [32]. Ker je prejem podatkov asinhron, smo vsak prejeti podatek najprej shranili v vrsto. Uporabili smo vrste za slike, podatke o višini, stanju baterije in hitrosti. Ko se je v vsaki od vrst za navigacijske podatke pojavil podatek, smo podatke v vrstah zapisali v datoteko. Shranili smo tudi čas zapisa podatkov. Vsakič, ko smo prejeli sliko, smo jo shranili na disk, njeno ime pa vpisali v datoteko s podatki. Ker smo prejemali več navigacijskih podatkov kot slik je nekaj polj za ime slike praznih. Tabela 5.1 prikazuje primer vsebine podatkovne datoteke v tabelarični obliki.

5.2 Izvedba

S pomočjo navigacijskih podatkov sestavimo mozaik tako, da posamezno sliko zamaknemo za relativni premik med trenutno in predhodno sliko. Prvo sliko tako postavimo na poljubno mesto, ostale pa relativno na predhodno sliko. Za izračun relativnega premika med dvema slikama moramo shranjene podatke nekoliko preurediti.

Ker podatke in slike prejemamo asinhrono in ker nismo shranjevali vsake prejete slike vendar približno vsako tretjo, imamo v podatkovni datoteki več

Tabela 5.1: Izsek CSV datoteke, ki prikazuje čas shranjevanja podatkov (timestamp), ime shranjene slike (image), nagib (pitch), nagib (roll), odklon (yaw), višina (h), tri komponente vektorja hitrosti (vx, vy, vz) in stanje baterije (b). Orientacije so v stopinjah, višina v milimetrih, hitrosti v mm/s, stanje baterije pa v odstotkih.

timestamp	image	pitch	roll	yaw	h	vx	vy	vz	b
1368455215444	00023.jpg	-2.192	2.697	157.164	677	276.2801	-163.57538	0.0	22
1368455215558	null	-2.073	1.797	157.219	695	294.02267	-164.61394	0.0	22
1368455215570	null	-2.03	1.341	157.294	699	312.55347	-149.35143	0.0	22
1368455215636	00024.jpg	-1.803	1.306	157.33	690	325.48895	-104.244225	0.0	22
1368455215702	null	-1.703	1.143	157.329	696	337.43307	-78.517944	0.0	22
1368455215765	00025.jpg	-1.507	0.896	157.367	699	351.4464	-66.25562	0.0	22
1368455215832	00026.jpg	-1.292	0.604	157.357	702	374.2386	-42.755016	0.0	22
1368455215898	null	-0.791	0.515	157.339	707	417.53867	-47.558777	0.0	22
1368455215961	null	-0.194	0.735	157.361	723	415.9827	-29.987808	0.0	22
1368455216035	00027.jpg	0.528	1.0	157.333	756	410.45563	-18.38236	0.0	22
1368455216094	null	1.207	1.133	157.26	776	343.6403	6.261978	0.0	22
1368455216159	null	1.725	1.129	157.14	789	317.93796	19.273197	0.0	22
1368455216224	00028.jpg	1.907	0.997	157.116	803	285.52066	30.649017	0.0	22
1368455216290	null	1.947	0.612	157.072	815	257.00858	36.37931	0.0	22
1368455216361	null	1.969	0.468	157.05	830	228.80917	42.15938	0.0	22
1368455216419	null	1.958	0.579	157.099	856	195.08156	47.463383	0.0	22
1368455216491	null	1.97	0.785	157.154	886	144.94698	58.872692	0.0	22
1368455216555	null	1.987	1.005	157.231	916	89.57386	68.15773	0.0	22
1368455216614	00029.jpg	1.872	1.067	157.35	958	58.051895	77.80594	0.0	22
1368455216679	null	1.742	1.21	157.39	974	20.771671	96.2457	0.0	22
1368455216747	null	1.288	1.427	157.367	999	2.7179322	116.36378	0.0	22
1368455216810	null	1.142	1.524	157.276	1027	-14.588915	132.42563	0.0	22
1368455216883	null	1.069	1.602	157.204	1048	-24.25906	150.1332	0.0	22
1368455216940	null	0.911	1.582	157.163	1068	-35.69627	183.76715	0.0	22
1368455217006	00030.jpg	0.899	1.245	157.132	1073	-54.665035	242.08432	0.0	22
1368455217075	null	0.653	0.73	157.135	1086	-87.096375	269.5268	0.0	22
1368455217138	null	-0.007	0.37	157.097	1085	-95.238655	254.54927	0.0	22
1368455217202	null	-0.715	-0.186	157.079	1092	-87.933334	253.30496	0.0	22
1368455217270	null	-1.307	-0.779	157.039	1101	-74.25615	213.80408	0.0	22
1368455217340	00031.jpg	-1.663	-1.053	156.984	1101	-86.22327	251.06305	0.0	22
1368455217404	null	-1.862	-1.149	157.001	1103	-83.58077	213.90364	0.0	22
1368455217463	null	-2.11	-1.029	157.01	1106	-70.26667	183.00346	0.0	22
1368455217530	00032.jpg	-2.325	-0.841	157.061	1105	-41.416393	182.55177	0.0	22

Tabela 5.2: Izsek tabele 5.1, ki prikazuje navigacijske podatke med prvo in drugo sliko.

timestamp	image	pitch	roll	yaw	h	vx	vy	vz	b
1368455215558	null	-2.073	1.797	157.219	695	294.02267	-164.61394	0.0	22
1368455215570	null	-2.03	1.341	157.294	699	312.55347	-149.35143	0.0	22
1368455215636	00024.jpg	-1.803	1.306	157.33	690	325.48895	-104.244225	0.0	22

vrstic s samimi navigacijskimi podatki, kot vrstic, ki tudi vsebujejo sliko. To je vidno v podatkovni tabeli 5.1. Tabela 5.2 prikaže tri vrstice navigacijskih podatkov pridobljenih med prvo in drugo sliko. Te podatke moramo združiti tako, da bo vsaka vrstica s sliko vsebovala povzetek teh podatkov.

Ker je čas med posameznimi meritvami kratek, lahko med njimi predpostavimo enakomerno gibanje. Zato lahko pot, ki jo je kvadrokopter preletel med zajemom prve in druge slike izračunamo po enačbi:

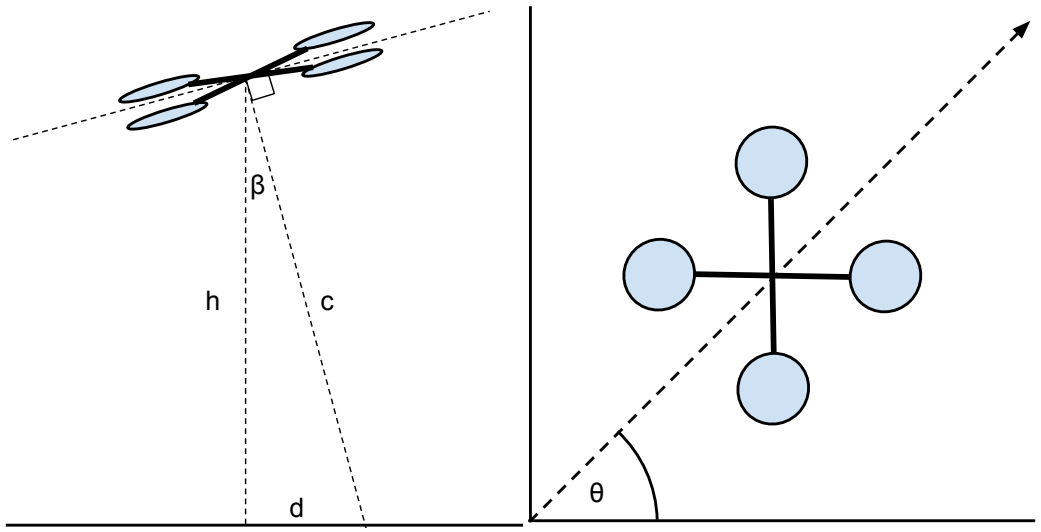
$$\vec{s} = \sum_{i=1}^{n} \vec{v}_i \cdot \Delta t_i \ , \tag{5.1}$$

kjer je $\vec{v}_i$ vektor treh komponent hitrosti, Δt pa trajanje odseka. Vektor $\vec{s}$ je vsota premikov v n odsekih.

Iz podatkovnih tabel je razvidno, da nam ogrodje YADrone za z-komponento hitrosti vedno vrne napačno vrednost (0.0), čeprav se je višina leta kvadrokopterja spreminjala. Zato tega podatka ne moremo uporabiti v izračunih.

Z relativnim premikom med dvema zaporednima slikama lahko izračunamo pozicijo kvadrokopterja, ko je zajel slike. Vendar zaradi orientacije kvadrokopterja v času zajema slike, slika ne prikazuje tla točno pod kvadrokopterjem. Zamik slike glede na pozicijo kvadrokopterja lahko izračunamo na podlagi izmerjenih podatkov višine in orientacije (naklon, nagib in odklon).

Slika 5.1a prikazuje stransko sliko kvadrokopterja, ki je nagnjen (npr. zaradi vožnje naprej). S črko h smo označili višino kvadrokopterja, s c pa smer, v katero je obrnjena kamera. Kamera je postavljena pod pravim kotom glede na kvadrokopter. Z d smo označili razdaljo odklona pozicije zajete slike



(a) (b)

Slika 5.1: Diagram (a) prikazuje kvadrokopterja iz strani nagnjen za kot β. Diagram (b) prikazuje isto sliko slikami od zgoraj, kjer se vidi da kvadrokopter leti diagonalno pod kotom θ.

kamere iz navpične pozicije kvadrokopterja. Za kot naklona kvadrokopterja β lahko d lahko izračunamo tako:

$$d = h \times \tan \beta \ . \tag{5.2}$$

Ker se kvadrokopter lahko na enak način nagiba naprej-nazaj (naklon) in levo-desno (nagib), enačba (5.2) velja za odmik točke centra slike v obeh dimenzijah. V primeru nagiba bi namesto kota naklona β uporabili kot nagiba α:

$$d_{nagib} = h \times \tan \alpha \ . \tag{5.3}$$

Sedaj si poglejmo še Sliko 5.1b, ki prikazuje let kvadrokopterja, ko leti poševno. Kot leta glede na vodoravnico θ lahko izračunamo iz x in y-komponente hitrosti (v_x in v_y) tako:

$$\theta = \arctan \frac{v_y}{v_x} \,. \tag{5.4}$$

Vendar tako ne moremo ločiti med kotom v I. in III., ter II. in IV. kvadrantu koordinatnega sistema. Ker želimo ločiti med koti v različnih kvadrantih moramo primerjati predznake v_x in v_y. Če sta oba predznaka pozitivna, je kvadrokopter usmerjen v kvadrant I, če sta oba negativna v kvadrantu III, če je v_x pozitiven, v_y pa negativen, je kvadrokopter usmerjen v kvadrant IV, če je v_x negativen, v_y pa pozitiven pa v kvadrant II. Uporabili smo Matlabovo funkcijo atan2(vy, vx), ki vrne pravilni kot, ne glede na smer leta.

Z enačbami (5.2) in (5.3) smo izračunali odmik nazaj/naprej in levo/desno centra slike relativno na smer leta kvadrokopterja. Vendar pri sestavljanju mozaika koordinate niso določene relativno na trenutno smer leta kvadrokopterja, ampak na koordinatni sistem platna/matrike na katero rišemo slike. Zato moramo izračunani odmik razdeliti na komponenti x in y.

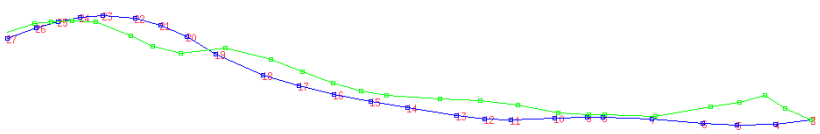
Ker smo predhodno izračunali smer leta, lahko x in y-komponente odmika izračunamo tako:

$$\begin{aligned} d_x &= d \cos\theta \\ d_y &= d \sin\theta \end{aligned} \tag{5.5}$$

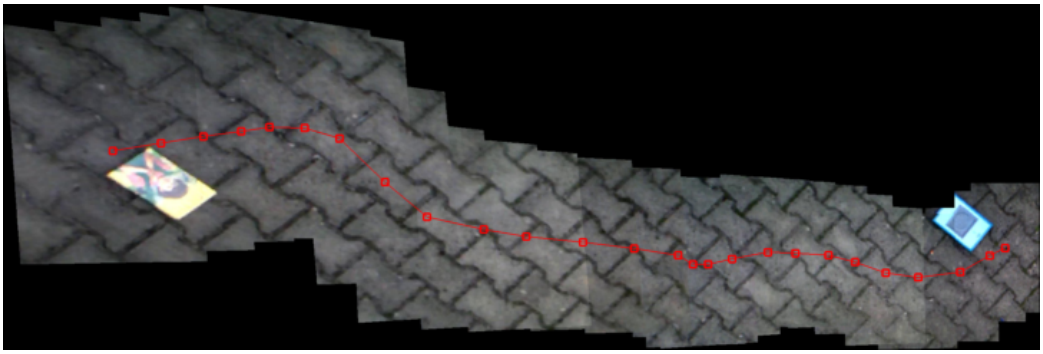
Zgornji izračun ponovimo za odmik zaradi nagiba in naklona kvadrokopterja. Pozicijo centra slike lahko izračunamo tako, da seštejemo koordinate kvadrokopterja in odmike po komponentah.

Slika 5.2a prikazuje rekonstruirano pot leta kvadrokopterja z modro črto, pozicije centrov zajetih slik pa z zeleno barvo. Za primerjavo si oglejmo mozaik na Sliki 5.2b, sestavljen s pomočjo tehnik računalniškega vida, kot smo opisali v prejšnjem poglavju. Na njem smo z rdečo črto povezali centre slik, da jih lahko primerjamo s potjo ocenjeno iz navigacijskih podatkov. Iz slike je razvidno, da so odstopanja velika. Če bi na izračunani poti prikazali slike v pravi orientaciji in perspektivi, bi bil izdelan mozaik zelo slab. Na podlagi podobnih primerjav smo ugotovili, da iz samih navigacijskih podatkov ne bomo mogli zgraditi dobrega mozaika. To velja za kvadrokopter AR.Drone,

ki nima natančnih senzorjev. Zato smo nadaljnji razvoj v tej smeri opustili.



(a)



(b)

Slika 5.2: Slika (a) prikazuje pot, kot smo je rekonstruirali iz samih navigacijskih podatkov. Modra črta prikazuje rekonstruirano pot kvadrokopterja, zelena pa lokacije zajetih slik, ki smo jih izračunali na podlagi izmerjenih rotacij kvadrokopterja. Slika (b) predstavlja mozaik zgrajen s tehnikam računalniškega vida. Rdeča črta povezuje centre slik. Ta rdeča črta in zelena iz slike (a) preveč odstopata. Iz samih navigacijskih podatkov torej ne bomo mogli zgraditi dobrega mozaika.

Poglavje 6

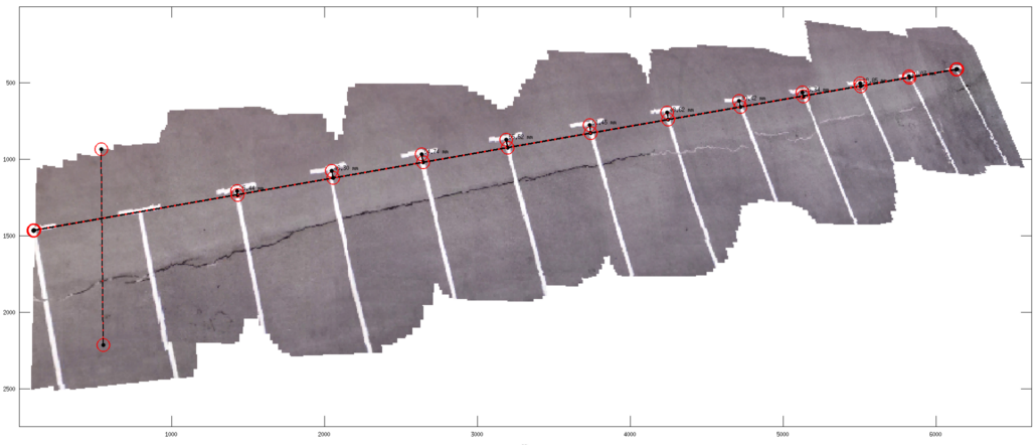
Eksperimentalni rezultati

V tem poglavju bomo prikazali primere zgrajenih mozaikov in jih ovrednotili. V prvem delu bomo ovrednotili mozaike zgrajene s pomočjo tehnik računalniškega vida. Opisali bomo tudi nekaj primerov slik, s katerimi lahko uspešno zgradimo mozaik in nekaj primerov s katerimi dobrega mozaika ne moremo zgraditi. V drugem delu ovrednotimo še rekonstruirane poti leta kvadrokopterja na podlagi navigacijskih podatkov. Na koncu prikažemo še nekaj primerov zgrajenih mozaikov.

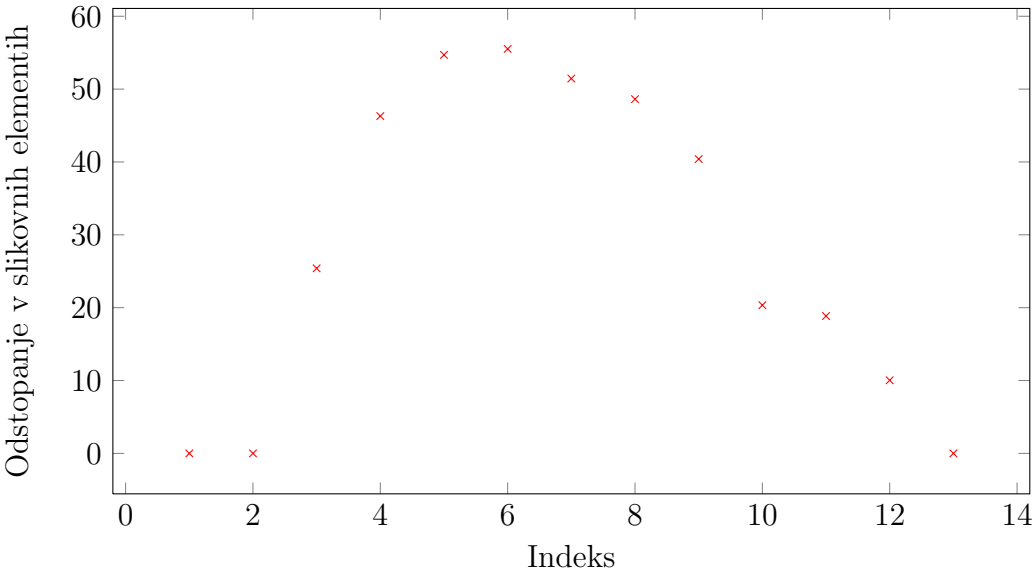
6.1 Mozaiki zgrajeni s tehnikam računalniškega vida

Prvi test, ki ga bomo naredili je izmerili odstopanja zgrajenega mozaika glede na pričakovano pozicijo slik. Na Sliki 6.1 je prikazan mozaik parkirišča sestavljen iz 300 zaporednih slik. Slike so bile zajete s prednjo kamero kvadrokopterja. Čez mozaik smo narisali premico, ki povezuje prvi in zadnji lik oblike črke T. Nato smo za vsako talno označbo izmerili razdaljo vrha označbe do premice. Slika 6.2 prikazuje graf izmerjene vrednosti v enotah slikovnih elementov. Povprečna vrednost meritev je 28.58 slikovnih elementov, standardni odklon pa 21.89.

Za drugi test smo izmerili odstopanje neke korespondenčne točke, ko jo



Slika 6.1: Mozaik sestavljen iz 300 slik. Odstopanja smo merili v razdalji pravokotno od najdaljše črte. Navpična črta na levem robu predstavlja višino zajetih slik (1280 slikovnih elementov) na podlagi katere smo kalibrirali ostale meritve.



Slika 6.2: Podatki predstavljajo odstopanja od črte, ki povezuje vrhove prve in zadnje talne označbe.

preletimo drugič. Ker slike združujemo samo na podlagi prehodne slike zaporedja, se napaka ocenjenih homografij prenaša v vse naslednje slike. Zato, ko kvadrokopter drugič preleti isto lokacijo pričakujemo odstopanje. Za kontrolno točko s pomočjo katere računamo odstopanje smo uporabili vogal knjige, ki smo jo položili na tla. S kvadrokopterjem smo leteli iz knjige naravnost, nato pa se po približno isti poti vrnili nazaj do knjige. Let smo ponovili dvakrat, enkrat smo snemali s prednjo kamero, enkrat pa s spodnjo kamero z nižjo ločljivostjo. Mozaika sta vidna na Sliki 6.3. Da bi lahko izmerili odstopanje korespondenčne točke, ko jo drugič obiščemo, smo slike narisali napol prozorne. Zato je slika na mozaikih prikazana dvakrat.

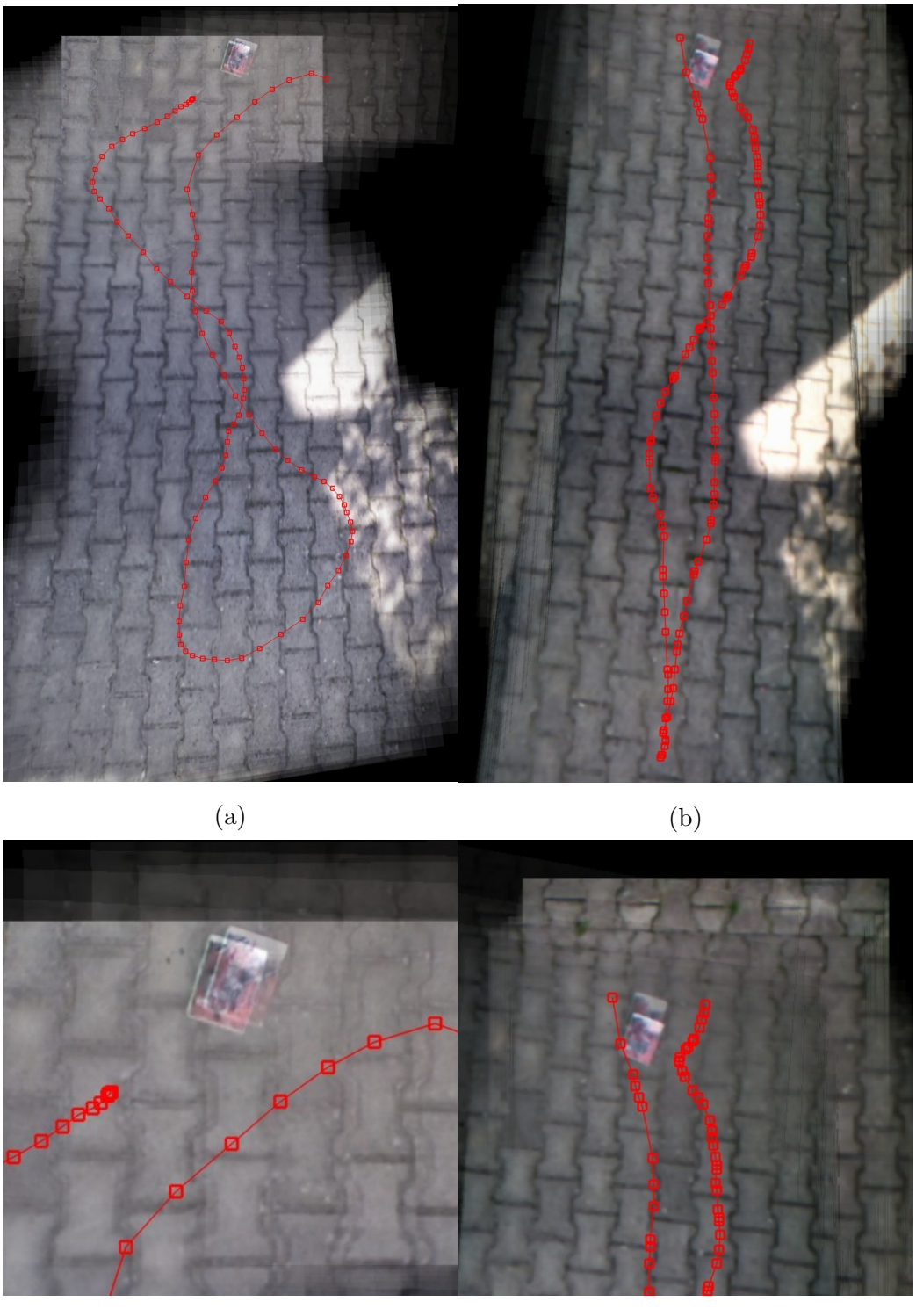
Izmerili smo razdalje od kota knjige, ki smo jo uporabili kot referenčno točko. Pri kameri z nizko ločljivostjo smo izmerili odstopanje 14 slikovnih elementov, pri kameri z visoko ločljivosti pa odstopanje 42 slikovnih elementov. Čeprav na slikah zgleda, da je odstopanje večje pri kameri z nizko ločljivostjo, relativno na zajeto velikost slike (320 namesto 1280 slikovnih elementov), je natančnost mozaika nižje ločljivosti boljša, kar lahko pripišemo manjši zamegljenosti, zaradi hitrejšega zajema slik kamere (60fps namesto 30fps).

6.2 Pomembni faktorji pri gradnji mozaika

Med gradnjo mozaikov smo opazili, da je kvaliteta mozaika zelo odvisna od slik, ki ga sestavljajo. Tukaj bomo opisali in prikazali nekaj faktorjev, ki vplivajo na kvaliteto mozaika.

Tekstura. Slika 6.4a prikazuje zid, ki ima precej malo teksture. S takšno sliko ne morem uspešno zgraditi mozaika, ker ne moremo izločiti distinktivnih značilnih točk. Izločene značilne točke si bodo precej podobne in ne bomo vedeli, s katerimi značilnimi točkami v naslednji sliki sovpadajo. Slika 6.4b prikazuje primer slike, ki ima dovolj teksture, da bi lahko uspešno izločili značilne točke. Podobno težavo imamo, če slike zajemamo ob večernih urah, ko je malo svetlobe in je nastala slika večinoma črna.

Relief. Ker homografije ocenijo kako se celotna slika preslika v prostor



(a) (b)

(c) (d)

Slika 6.3: Sliki primerjata natančnost izdelanega mozaika s primerjanjem pozicije korespondenčne točke (kot knjige), ko jo drugič preletimo. Sliki (a) in (c) sta zajeti s prednjo kamero, (b) in (d) pa s spodnjo. Dolžina letov je približno ista. Mozaik (a) vsebuje 90 slik, (b) pa 110.

(a) (b)

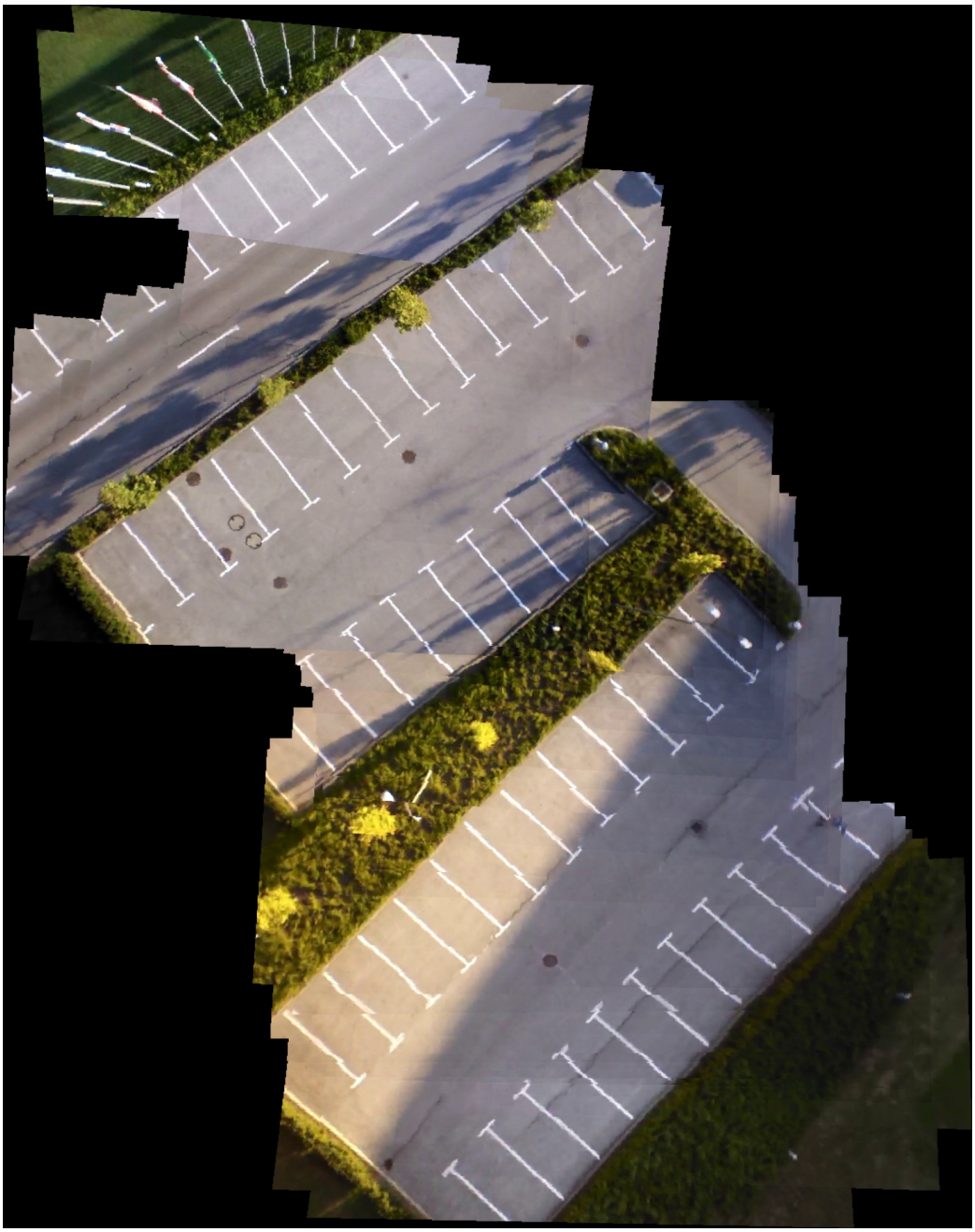
Slika 6.4: Slika (a) ima premalo teksture, da bi lahko izločili distinktivne značilne točke. Slika (a) tudi prikazuje vinjetenje. Slika (b) ima ravno dovolj teksture za gradnjo mozaika.

druge, mora biti višina v 3-D vseh slikanih točk približno enaka. Zaradi algoritma RANSAC pa lahko gradimo mozaika tudi s slikami, kjer določen delež predmetov štrli iz ravnine. Slika 6.5 prikazuje mozaik zgrajen iz zaporedja slik parkirišča, iz katerega izstopajo drevesa in zastave.

Spremenljiva kvaliteta zaporednih slik. V Poglavju 4.6 smo povedali, da se kvaliteta posnetka, zajetega s kvadrokopterjem AR.Drone, občasno močno spremeni. Na Sliki 4.13 smo pokazali primera dobre in slabe kvalitete posnetka. Iz takšnih slik težko zgradimo mozaik, ker ne moremo najti korespondenčnih točk na obeh slikah. Podobna težava se pojavi, če kvadrokopter leti prehitro. Takrat bodo slike, zaradi hitrosti gibanja, zamegljene. Zato smo omejili hitrost kvadrokopterja tako, da smo določili maksimalni kot naklona 8°.

Valoviti posnetek. Pri sprednji kameri kvadrokopterja smo opazili, da se na video posnetkih pojavi neke vrste valovanje. Zgleda, kot da bi se nek zračni val premikal vertikalno čez sliko. Učinek je posledica tresenja kvadrokopterja med letom zaradi neporavnanih propelerjev. Zaradi tega se črte nekoliko deformirajo. Ta pojav je viden na Sliki 6.6. Posledica je, da izdelani mozaiki niso tako natančni, kot bi lahko bili. Primera takšnih mozaikov sta vidna na Slikah 6.1 in 6.5.

Prekrivanje slik. Da bi lahko dobro ocenili homografijo med dvema za-



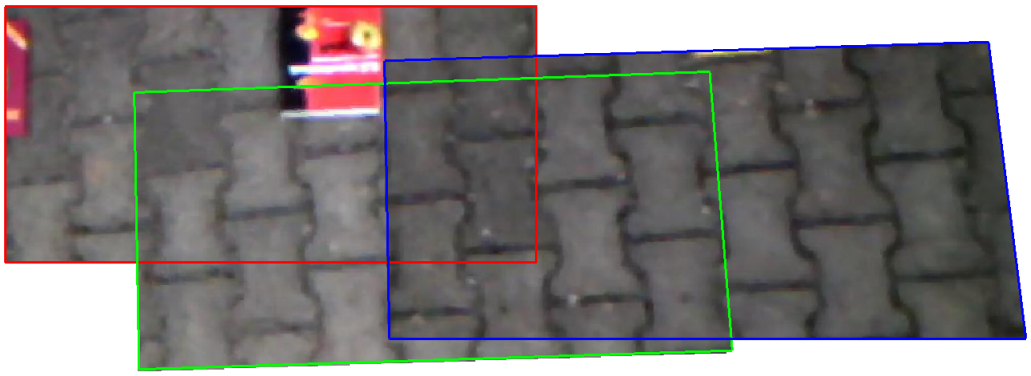
Slika 6.5: Mozaik 120 slik, zgrajen iz slik, kjer niso vsi predmeti na isti višini. Iz največje ravnine (ceste) odstopajo drevesa in zastave na stebrih.

Slika 6.6: Snemanje s prednjo kamero kvadrokopterja povzroči deformacijo, ki je na tej sliki vidna kot zamegljenost črt.

porednima slikama se morajo slike dovolj prekrivati. Na sliki 6.7 prikažemo mozaik, kjer smo slikam jasno označili robove. Če bi bile vse slike dobre ločljivosti in kvalitete, bi bilo dovolj, če bi se slike prekrivale vsaj 20%. Ampak, ker so zaporedne slike spremenljive kvalitete, moramo občasno katero od slik preskočiti. Uspešno smo zgradili mozaike iz slik, ki so se prekrivale približno 50%. Večina mozaikov, ki smo jih zgradili pa je sestavljenih iz slik, ki se prekrivajo vsaj 70%. S tem smo dosegli večjo mero redundantnosti, saj lahko preskočimo tudi po več zaporednih slik iz zaporedja, pa še vedno lahko zgradimo mozaik.

Vinjetenje. Vinjetenje je pojav, ko so robovi slik temnejši, oziroma imajo barve ob robovih nižjo nasičenost kot center slike. Vinjetenje je opazno na sliki 6.4a. Posledica je, da se na mozaikih jasno vidi vsaka posamezna slika. Da bi pojav omilili, smo slikam odrezali nekaj slikovnih elementov roba. Takšen mozaik zgleda bolj enovit. Mozaike bi lahko še bolj izboljšali s postprocesiranjem, kar pa ni bil cilj diplomske naloge.

Radialna distorzija. S slikami, ki so popačene z radialno distorzijo, ne



Slika 6.7: Na tem mozaiku smo z barvnimi črtami označili robove slik.

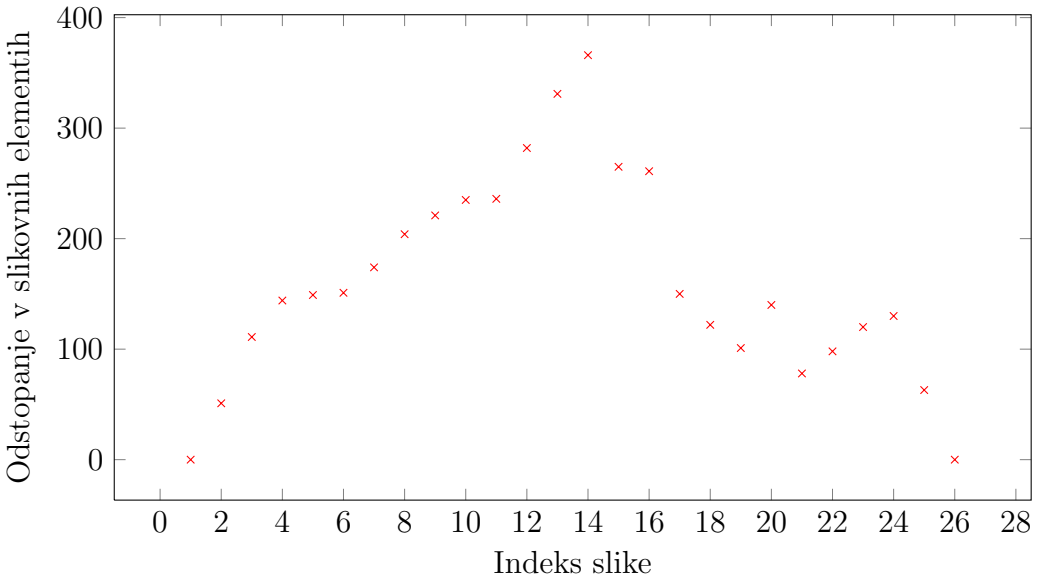
moremo zgraditi mozaika. Primer mozaika dveh slik z radialno distorzijo je prikazan na Sliki 3.2. Za uspešen mozaik moramo radialno distorzijo odpraviti, kar smo opisali v Poglavju 3.2. Po odstranitvi radialne distorzije iz slik smo uspešno gradili mozaike.

6.3 Rekonstrukcija preletene poti iz navigacijskih podatkov

V Poglavju 5.2 smo pokazali primer rekonstruirane poti s pomočjo navigacijskih podatkov in mozaik izdelan s pomočjo tehnik računalniškega vida. Malo natančnejši pogled na slike je pokazal, da se rekonstruirana pot iz navigacijskih podatkov in pot, ki smo jo ocenili na podlagi izdelanega mozaika vidno razlikujeta. Sedaj bomo odstopanja še izmerili. Slika 6.9 prikazuje rekonstruirano pot, ki smo jo postavili nad izdelan mozaik tako, da začetek in konec poti sovpadata. Ker smo poti obarvali z različnimi barvami, bomo najprej razložili kaj vsaka barva predstavlja:

Modra črta Rekonstruirana pot kvadrokopterja iz navigacijskih podatkov

Zelena črta Izračunane pozicije centrov slik, na podlagi rotacije in izračunane pozicije kvadrokopterja.



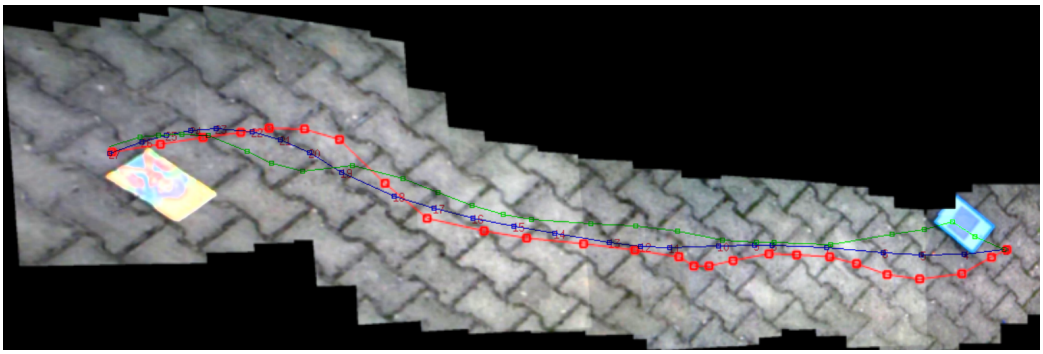
Slika 6.8: Podatki predstavljajo odstopanja ocenjenih pozicij slik od pozicij slik na mozaiku sestavljenem s tehnikam računalniškega vida.

Rdeča črta Z rdečo črto smo povezali centre slik mozaika, ki smo ga zgradili s pomočjo tehnik računalniškega vida.

Slika 6.8 prikazuje graf izmerjenih odstopanj izračunanih pozicij slik (zelena črta) od pozicije slik pridobljenih iz mozaika (rdeča črta). Povprečno odstopanje je 160, standardni odklon pa 93 slikovnih elementov. Iz velikega povprečnega odstopanja sklepamo, da iz samih navigacijskih podatkov ne bomo uspešni pri gradnji zemljevida.

Drugi test, ki smo ga opravili je primerjava razmerja dolžin posameznik odsekov (razmakov med slikami). Za ta test smo uporabili rekonstruirano pot prikazano na sliki 6.11a in mozaik na sliki 6.11b. Pomen barv je enak kot v prvem testu. Najprej smo po enačbi (6.1) izračunali razdaljo med pozicijami slik za pot rekonstruirano iz navigacijskih podatkov ($\mathbf{d}_{data}$) in pot pridobljeno iz mozaika $\mathbf{d}_{cv}$:

$$\begin{aligned} \mathbf{d}_{cv} &= \sqrt{\mathbf{x}_{cv}^2 + \mathbf{y}_{cv}^2} \\ \mathbf{d}_{data} &= \sqrt{\mathbf{x}_{data}^2 + \mathbf{y}_{data}^2}\ , \end{aligned} \tag{6.1}$$



Slika 6.9: Ocenjene pozicije fotografij (zeleno) smo prikazali na mozaiku sestavljenem s pomočjo tehnik računalniškega vida, kjer smo dejanske pozicije slik prikazali z rdečo črto.

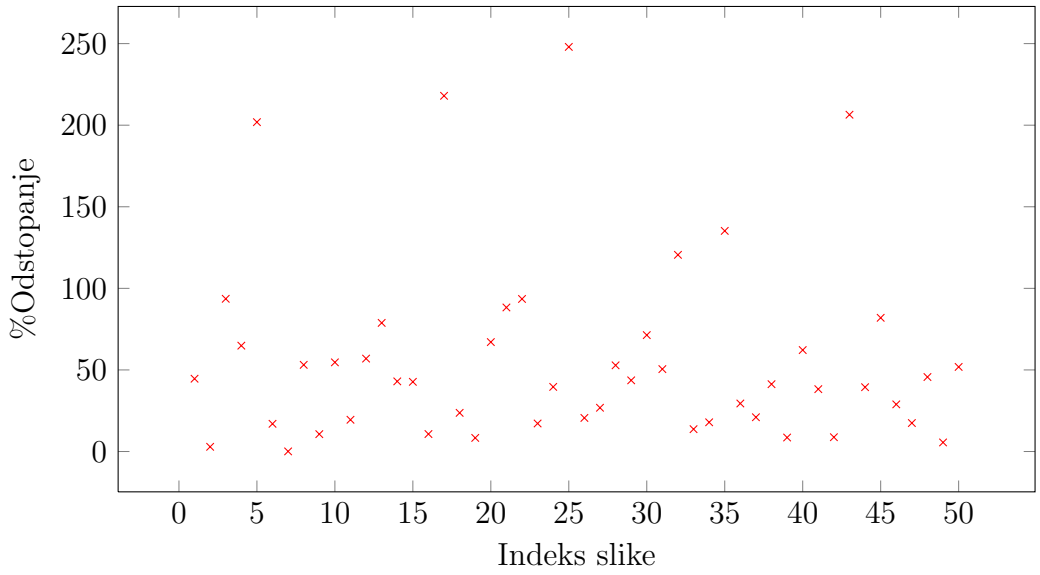
kjer je $\mathbf{x}$ vektor x-komponente koordinat slik, $\mathbf{y}$ pa vektor y-komponente koordinat slik. Ker niso enote obeh vektorjev razdalj enake, smo vse meritve normalizirali, tako da smo delili z vsoto razdalj:

$$\begin{aligned} \hat{\mathbf{d}}_{cv} &= \frac{\mathbf{d}_{cv}}{\sum_{i=1}^{N} \mathbf{d}_{cv,i}} \\ \hat{\mathbf{d}}_{data} &= \frac{\mathbf{d}_{data}}{\sum_{i=1}^{N} \mathbf{d}_{data,i}} \, , \end{aligned} \tag{6.2}$$

kjer je N število podatkov. Sedaj je vsota vseh razdalj za oba nabora podatkov enaka 1, in ju lahko primerjamo. Ker nas zanima povprečno odstopanje podatkov, smo izračunali relativno spremembo podatkov v odstotkih:

$$\%Odstopanje = \frac{|\hat{\mathbf{d}}_{data} - \hat{\mathbf{d}}_{cv}|}{|\hat{\mathbf{d}}_{cv}|} \times 100 \, . \tag{6.3}$$

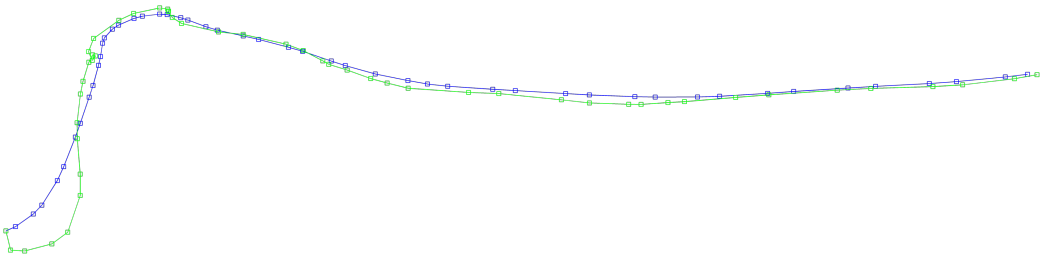
Graf, ki prikazuje izmerjena odstopanja je prikazan na Sliki 6.10. Na vodoravni osi je prikazan zaporedni indeks slike, na navpični pa izračunano odstopanje v odstotkih. Povprečna vrednost odstopanj normaliziranih dolžin odsekov v odstotkih je 56.77%, standardni odklon pa 56.98%. Izmerjena odstopanja so prevelika, da bi lahko s takšnimi podatki uspešno sestavili mozaik. Z boljšim kvadrokopterjem, ki bi bil opremljen z boljšimi senzorji, bi bila odstopanja manjša.



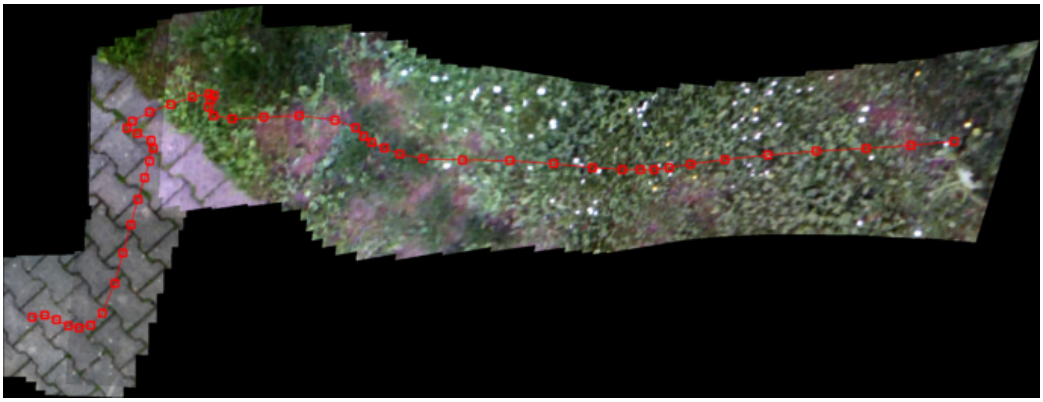
Slika 6.10: Graf prikazuje izračunana odstopanja normaliziranih dolžin odsekov, ki so vidni na Sliki 6.11.

6.4 Še nekaj primerov

V tem poglavju prikažemo še nekaj primerov izdelanih mozaikov. Slike 6.12, 6.13 in 6.14 so zajete na isti lokaciji. Razlikujejo se v višini leta kvadrokopterja in deležem prekrivanja slik. Slika 6.15 prikazuje mozaik zgrajen brez pravil za odkrivanje slabih homografij. Zato so se nekatere slike slabo transformirale. Sliki 6.16 in 6.17 primerjata mozaike zgrajene s sprednjo in spodnjo kamero kvadrokopterja AR.Drone.

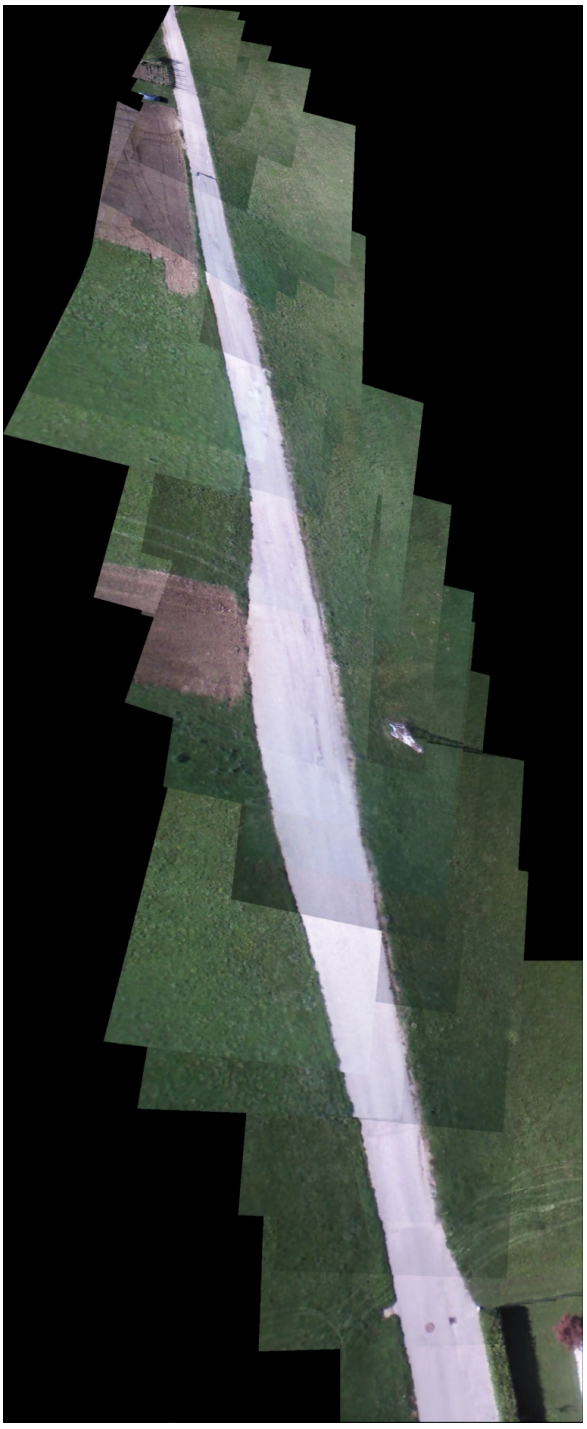


(a)



(b)

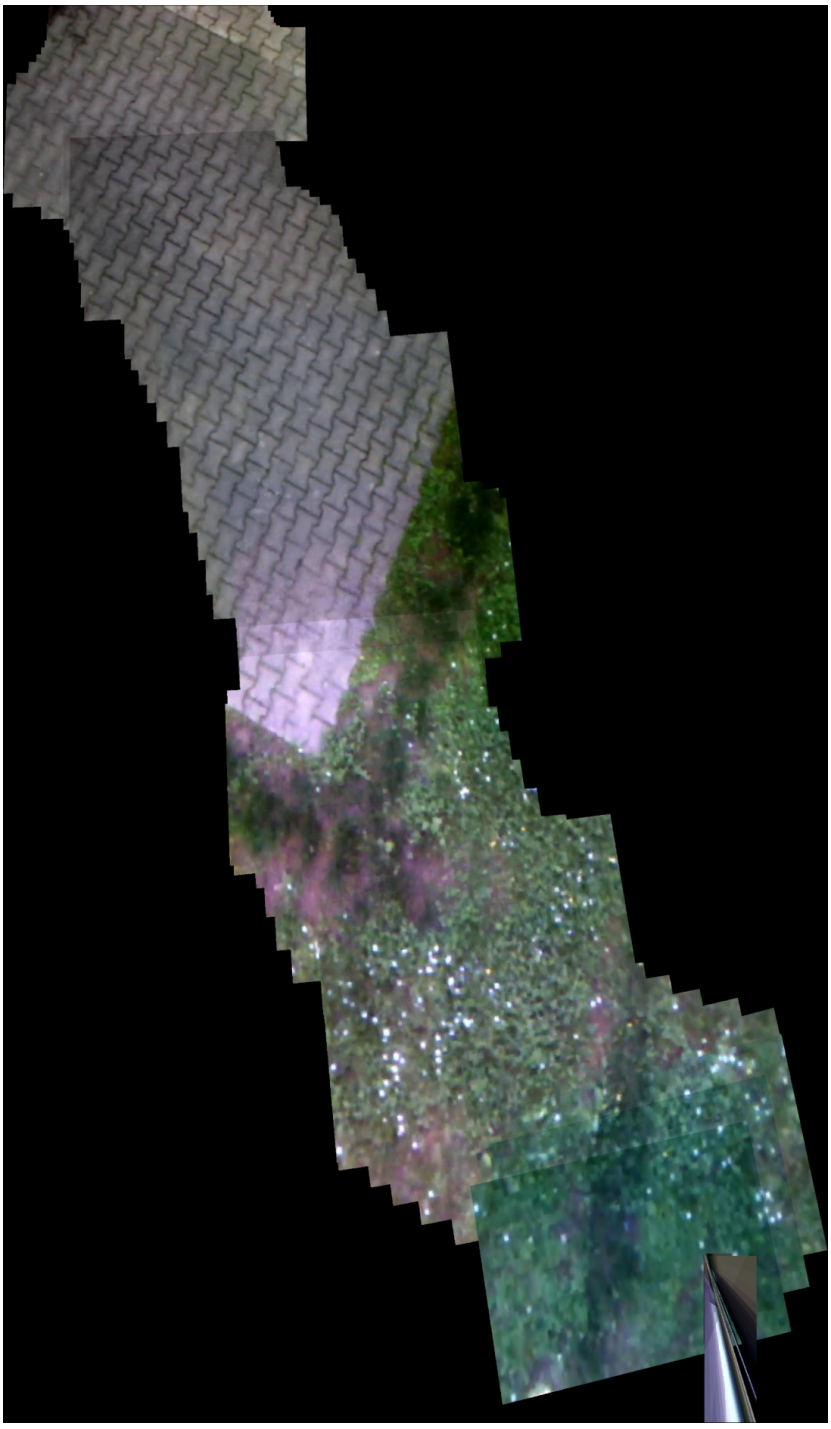
Slika 6.11: Zelena pot na sliki (a) predstavlja pozicije slik, ki smo jih izračunali iz navigacijskih podatkov. Rdeča pot na sliki (b) predstavlja pozicije centrov slik na mozaiku izdelanem s tehnikami računalniškega vida.



Slika 6.12: Mozaik 35 slik zajetih iz približno 30 metrov višine. Čeprav se pri tem mozaiku slike prekrivajo dosti manj kot na Sliki 6.13, smo prav tako uspešno sestavili mozaik.

Slika 6.13: Mozaik 351 slik zajetih iz približno 20 metrih višine. Zaradi slabše ocenjene homografije na sredini slike, je ena polovica mozaika nekoliko zamaknjena.

Slika 6.14: Mozaik 500 slik zajetih iz približno 10 metrov višine. Pokritost je približno polovica toliko kot mozaik na Sliki 6.12. Zaradi nekaterih slabo ocenjenih homografij na dnu mozaika se pot predčasno zoži in deformira.



Slika 6.15: Slike za ta mozaik so bile zajete s spodnjo kamero kvadrokopterja AR.Drone. Mozaik je bil izdelan brez pravil za zaznavo slabih homografij, ki smo jih opisali v Poglavju 4.6. Zaradi tega so se slike na dnu slabo ukrivile.

Slika 6.16: Ta mozaik je bil izdelan iz slik sprednje kamere kvadrokopterja.

Slika 6.17: Ta mozaik je bil izdelan iz slik spodnje kamere kvadrokopterja.

Poglavje 7

Sklepne ugotovitve

7.1 Sklep

V tem diplomskem delu smo sestavljali mozaike iz zaporedja slik pridobljenih s kvadrokopterjem AR.Drone. Nalogo smo uspešno rešili s pomočjo algoritmov računalniškega vida. Dodatno smo poskusili rekonstruirati pot kvadrokopterja s pomočjo navigacijskih podatkov, ki nam jih je kvadrokopter pošiljal med letom. Izkazalo se je, da lahko navigacijske podatke uporabimo le za grobo oceno preletene poti in sestavljanje mozaika na podlagi teh meritev ne bi bilo uspešno.

Z izdelanim programom smo uspešno združili do 1000 slik velikosti 1280×720. Zatem so se pojavile težave z velikostjo pomnilnika računalnika. Izdelane mozaike smo ovrednotili in izmerili napako. Primerjali smo mozaike zgrajene s sprednjo in spodnjo kamero na kvadrokopterju AR.Drone. Razlika je, da ima sprednja kamera boljšo ločljivost, a tudi povzroča izrazito radialno distorzijo. Spodnja kamera ima nižjo ločljivost, slike pa niso popačene.

Čeprav smo naš program namenili mozaičenju slik pridobljenih s kvadrokopterjem AR.Drone, ga lahko uporabimo tudi za mozaičenje slik, zajetih z drugimi napravami. Slika 7.1 prikazuje mozaik sestavljen iz slik zajetih z mobilnim telefonom Samsung Galaxy Ace 2. Ker se v tem primeru kamera vrti okoli svoje osi, slike pa projiciramo na ravnino, ki jo določa prva slika zaporedja,

Slika 7.1: Primer mozaika treh slik, ki so bile zajete s kamero mobilnega telefona Samsung Galaxy Ace 2.

bo vsaka naslednja slika večja in bolj raztegnjena. Da bi nekoliko omejili razteg slik bi lahko projecirali slike na ravnino, ko jo določa slika na sredini zaporedja. Tako bi dobili sliko, ki je na straneh višja kot na sredini.

7.2 Možnosti za izboljšavo

Avtomatska stabilizacija leta mobilne platforme AR.Drone močno olajša upravljanje kvadrokopterja, vendar je še vedno težko pilotirati kvadrokopter po vnaprej določeni poti. Zato je težko sestaviti mozaik, ki bi pokril večje območje. Junija 2013 je podjetje Parrot razvilo GPS napravo Parrot GPS Flight Recorder, ki jo lahko priključimo na konektor USB kvadrokopterja

AR.Drone. Razvili so tudi pripadajočo programsko rešitev, v kateri lahko na zemljevidu označimo točke, ki jih mora kvadrokopter preleteti. Kvadrokopter nato avtonomno vzleti, obišče vse označene točke in snema let. Če bi pravilno določili točke, ki naj jih kvadrokopter preleti, bi lahko s programsko opremo, izdelani v tej diplomski nalogi, sestavili mozaik, ki pokrije večje območje.

Na mozaikih, ki smo jih izdelali, se pogosto da razpoznati robove posameznih slik. S tem, da bi bolje zlili (ang. blend) robove slik bi bil mozaik bolj prijeten na pogled. Poznamo več metod zlivanja. Preprosta metoda je linearna interpolacija prekrivajočih se slikovnih elementov. Naprednejše metode so Poissonovo zlivanje [33] in več pasovno zlivanje (ang. multi-band blending) [34].

V Poglavju 6.1 smo izmerili natančnost mozaikov tako, da smo izmerili odstopanje neke korespondenčne točke, ko jo dvakrat preletimo. Napako bi zmanjšali, če bi uporabili globalno optimizacijsko metodo, kot je tista opisana v [35].

Natančnost mozaikov bi izboljšali, če bi kvadrokopter imel boljše kamere. Originalna različica kvadrokopterja AR.Drone je imela ločljivosti sprednje kamere 640×480 slikovnih elementov. Različica 2.0, ki smo jo uporabili v tem diplomskem delu, je imela kamero ločljivosti 1280×720. Na podlagi tega napredka lahko za naslednjo različico kvadrokopterja AR.Drone pričakujemo boljšo kamero, s katero bomo lahko izdelali bolj natančen mozaik. Čeprav je to diplomsko delo obravnavalo mozaičenje iz slik kvadrokopterja AR.Drone, lahko izdelani program brez sprememb uporabimo na slikah zajetih z drugimi letečimi plovili. Z boljšo opremo bi lahko zajeli boljše slike in posledično izdelali boljše mozaike.

Literatura

[1] "History of cartography (Wikipedia)," http://en.wikipedia.org/wiki/History_of_cartography, dostop: 10/09/2013.

[2] "Spot image (Wikipedia)," http://en.wikipedia.org/wiki/Spot_Image, dostop: 01/09/2013.

[3] S. I. Corporation, "Google Earth: Identifying high-resolution satellite target locations," http://www.satimagingcorp.com/google_earth.html, dostop: 7/09/2013.

[4] "Google Earth (wikipedia)," http://en.wikipedia.org/wiki/Google_Earth, dostop: 10/09/2013.

[5] R. Szeliski, "Video mosaics for virtual environments," *IEEE Comput. Graph. Appl.*, vol. 16, št. 2, str. 22–30, Mar. 1996.

[6] H.-Y. Shum in R. Szeliski, "Panoramic image mosaics," Teh. Rep., 1997.

[7] R. Szeliski in H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," v *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, str. 251–258.

[8] S. Peleg in J. Herman, "Panoramic mosaics by manifold projection," v *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997, str. 338–343.

[9] S. Peleg in M. Ben-Ezra, "Stereo panorama with a single camera," v *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, 1999, str. –401 Vol. 1.

[10] R. Szeliski, "Image mosaicing for tele-reality applications," v *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, 1994, str. 44–53.

[11] Y. Xiong in K. Pulli, "Fast and high-quality image blending on mobile phones," v *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, 2010, str. 1–5.

[12] T. Krajník, V. Vonásek, D. Fišer, in J. Faigl, "AR-Drone as a Platform for Robotic Research and Education," v *Research and Education in Robotics: EUROBOT 2011.* Heidelberg: Springer, 2011.

[13] T. Krajník, M. Nitsche, S. Pedre, L. Přeučil, in M. Mejail, "A Simple Visual Navigation System for an UAV," v *International Multi-Conference on Systems, Signals and Devices.* Piscataway: IEEE, 2012, str. 34.

[14] N. Dijkshoorn in A. Visser, "Integrating sensor and motion models to localize an autonomous AR.Drone," *International Journal of Micro Air Vehicles*, vol. 3, št. 4, str. 183–200, December 2011.

[15] A. Visser in R. J. Nick Dijkshoorn, M. van der Veen, "Closing the gap between simulation and reality in the sensor and motion models of an autonomous AR.Drone," v *Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV11)*, 2011.

[16] N. Dijkshoorn, "Simultaneous localization and mapping with the AR.Drone," Magistrska naloga, Universiteit van Amsterdam, July 2012.

[17] S. Piskorski, N. Brulez, P. Eline, in F. D'Haeyer, "AR.Drone developer guide," http://www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf, dostop: 3/07/2013.

[18] "Aircraft principal axes (wikipedia)," http://en.wikipedia.org/wiki/Aircraft_principal_axes, dostop: 10/09/2013.

[19] T. Wiegand, G. J. Sullivan, G. Bjontegaard, in A. Luthra, "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, št. 7, str. 560–576, Jul. 2003.

[20] G. Wallace, "The jpeg still picture compression standard," *Consumer Electronics, IEEE Transactions on*, vol. 38, št. 1, str. xviii–xxxiv, 1992.

[21] YADrone, "YADrone - yet another AR.Drone framework," http://vsisls1.informatik.uni-hamburg.de/projects/yadrone/, dostop: 5/09/2013.

[22] J. Heikkila in O. Silven, "A four-step camera calibration procedure with implicit image correction," v *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997, str. 1106–1112.

[23] R. I. Hartley in A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[24] J. A. Parker, R. V. Kenyon, in D. Troxel, "Comparison of interpolating methods for image resampling," *Medical Imaging, IEEE Transactions on*, vol. 2, št. 1, str. 31–39, 1983.

[25] D. Lowe, "Object recognition from local scale-invariant features," v *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, str. 1150–1157 vol.2.

[26] J. S. Beis in D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," v *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, ser. CVPR '97. Washington, DC, USA: IEEE Computer Society, 1997, str. 1000–.

[27] E. Dubrofsky, "Homography estimation," Magistrska naloga, The university of British Columbia, 2007, master's essay.

[28] M. A. Fischler in R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, št. 6, str. 381–395, 1981.

[29] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematic*, vol. 11, št. 2, str. 431–441, Jun. 1963.

[30] C. A. Glasbey in K. V. Mardia, "A review of image warping methods," *Journal of Applied Statistics*, vol. 25, str. 155–171, 1998.

[31] W3C, "Portable Network Graphics (PNG) specification (second edition)," http://www.w3.org/TR/PNG/, dostop: 13/08/2013.

[32] "Comma-separated values (Wikipedia)," http://en.wikipedia.org/wiki/Comma-separated_values, dostop: 16/08/2013.

[33] P. Pérez, M. Gangnet, in A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, št. 3, str. 313–318, Jul. 2003.

[34] M. Brown in D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, št. 1, str. 59–73, Avg. 2007.

[35] R. Unnikrishnan in A. Kelly, "A constrained optimization approach to globally consistent mapping," v *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, 2002, str. 564–569 vol.1.