Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Saje

Učinkovit postopek izračuna časa do trka s pomočjo kamere

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in matematika

Mentor: doc. dr. Matej Kristan

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: 00032/2013
Datum: 11.04.2013

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

Kandidat: **NEJC SAJE**

Naslov: **UČINKOVIT POSTOPEK IZRAČUNA ČASA DO TRKA S POMOČJO KAMERE**

AN EFFICIENT CAMERA-BASED APPROACH FOR ESTIMATING TIME-TO-COLLISION

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

V nalogi naslovite problem izračuna časa do trka z objektom, ki ga opazujemo s kamero. Najprej izpeljite izračun časa do trka s pomočjo vektorjev gibanja izračunanih iz zaporedja slik. Nato analizirajte možne pristope za hiter izračun vektorjev gibanja ter pristope za izboljšavo robustnosti izračuna. Pri izpeljavi robustne metode obravnavajte zaporedne slike kot sistem več pogledov kamere ter vpeljite omejitve, ki jih v takem primeru vsiljuje epipolarna geometrija. Razvito metodo ovrednotite na realnih primerih.

Mentor:

doc. dr. Matej Kristan

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič

UNIVERZA V LJUBLJANI • Fakulteta za matematiko in fiziko • 1

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Saje, z vpisno številko **63100351**, sem avtor diplomskega dela z naslovom:

Učinkovit postopek izračuna časa do trka s pomočjo kamere

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 16. septembra 2013 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Mateju Kristanu za obsežno pomoč, številne ideje in predloge pri izdelavi algoritma ter diplomske naloge. Zahvalil bi se doc. dr. Janezu Peršu za pomoč pri zajemu posnetkov na linearnem pogonu. Hvala tudi staršem in Sanji za pomoč in potrpežljivost v času nastajanja diplomske naloge.

Kazalo

Seznam uporabljenih kratic in simbolov

- BRIEF - Binary Robust Independent Elementary Features
- FAST - Features from Accelerated Segment Test
- FOE - Focus of expansion (fokus ekspanzije)
- MAD - Median absolute deviation (mediana absolutnega odklona od mediane)
- RANSAC - Random Sample Consensus
- SIFT - Scale-invariant feature transform
- SURF - Speeded Up Robust Features
- SVD - Singular value decomposition (razcep po singularnih vrednostih)
- TTC - Time to contact (čas do trka)

Povzetek

Cilj diplomske naloge je bila implementacija algoritma, ki iz zaporedja slik izračuna čas do trka z objektom, proti kateremu se premika kamera. Čas do trka je izračunan s pomočjo informacije o spremembi velikosti slike objekta skozi čas. Ta informacija je pridobljena iz redkega polja gibanja, za izračun katerega sta uporabljena hiter detektor značilnih točk in hiter opisnik regij. Korespondence so filtrirane z uporabo epipolarne geometrije, zaradi šumnih podatkov pa je fokus ekspanzije izračunan kot presek premic, ki jih določajo vektorji polja gibanja. Glede na količino točk je izbrana velikost okolice fokusa ekspanzije ter na tej okolici izračunan globalni čas do trka. Za vizualizacijo rezultatov algoritma je bila slika razdeljena v regije ter za vsako regijo izračunana verjetnost trka z uporabo izračunanega časa do trka, variance izračunov znotraj regije ter števila točk v regiji. Algoritem je bil testiran na slikovnih zaporedjih, ki so bila pridobljena v kontroliranih pogojih na linearnem pogonu ter z uporabo električnega motorja za premikanje kamere.

Ključne besede: čas do trka, redko polje gibanja, značilne točke, epipolarna geometrija, računalniški vid, kamera

Abstract

The primary objective of this thesis was the implementation of an algorithm for calculating time to contact with an object the camera is approaching. Time to contact is calculated using information about the change of object's size over time. This information is extracted from a sparse motion field which is calculated using a fast keypoint detector and an efficient keypoint descriptor. Correspondences are filtered using epipolar geometry. Because the data is noisy, the focus of expansion is calculated as the intersection of lines that run through the vectors of the motion field. Depending on the amount of keypoints, the size of the region surrounding the focus of expansion is chosen and the global time to contact is calculated using the points inside that region. The image is divided in regions and the probability of collision is calculated using time to contact calculations, the variance of calculations inside the region and the number of points in the region. The implemented algorithm was tested on image sequences acquired in controlled environments.

Keywords: time to contact, sparse motion field, keypoints, epipolar geometry, computer vision, camera

Poglavje 1

Uvod

Čas do trka (angl. time to collision, TTC) poimenujemo informacijo, ki nam pove, čez koliko časa bomo ob ohranitvi trenutne hitrosti in smeri prišli v stik z oviro. Informacijo o času do trka lahko uporabimo za izogibanje oviram ali opozarjanje na nevarnost trka.

Določevanje časa do trka z uporabo optičnih informacij so najprej začeli preučevati v psihologiji [18]. Človeški možgani namreč za zaznavanje trka ne potrebujejo informacije o hitrosti premikanja, razdalje ali velikosti ovire, ampak je dovolj že podatek o spremembi velikosti slike ovire skozi čas. Potreba po le optični informaciji nas usmeri k uporabi računalniškega vida. Če smo zmožni izračunati, kako hitro se spreminja velikost slike ovire, lahko brez dodatnih podatkov izračunamo čas do trka z oviro.

Za izračun hitrosti spreminjanja velikosti slike ovire imamo na voljo veliko različnih načinov. Lahko detektiramo konkretne objekte v sliki in potem spremljamo spreminjanje njihove velikosti skozi čas [26]. Detekcija objektov je v računalniškem vidu precej raziskano področje [35, 30, 10], lahko pa spremljamo le vnaprej določene tipe ovir.

Če želimo zaznavanje trka posplošiti na vse ovire, lahko koncept ovire kot objekta izpustimo in opazujemo posamezne slikovne elemente ali slikovne regije. Če predpostavimo, da je vsak slikovni element ali slikovna regija del nekega objekta, lahko hitrost spreminjanja velikosti tega objekta opišemo s

hitrostjo spreminjanja razdalje med opazovanim elementom ter fokusom ekspanzije, tj. točke na sliki, od katere se vse ostale točke navidezno odmikajo.

Informacijo o času do trka lahko uporabimo tudi v prometu. V [18] Lee ugotavlja, da voznik avtomobila v situaciji, ko vozilo pred njim začne zavirati, zavira v dveh fazah. V prvi fazi le prepozna potrebo po zaviranju, šele nato se odloči o moči, s katero bo zaviral. Vozniki namreč navadno ne začnejo takoj zavirati s polno močjo zaradi možnosti naleta za njimi vozečih vozil. Zavorne luči vozniku pomagajo le pri prvi fazi, medtem ko mora za drugo fazo informacijo o količini potrebnega zaviranja zbrati sam s pomočjo optičnih informacij.

Kamera in procesorska enota, nameščeni v vozilu, bi lahko v taki situaciji vozniku v realnem času s pomočjo npr. jakosti opozorilnega zvoka posredovali informacijo o hitrosti približevanja ovire pred njim. Ta informacija bi vozniku pomagala do hitrejše odločitve o moči zaviranja in posledično skrajšala njegov odzivni čas.

1.1 Sorodna dela

Za izračun časa do trka potrebujemo informacijo o neki optični količini, kot je npr. velikost objekta, kot ali lokacija, in hitrost spreminjanja te količine. Na področju računanja časa do trka s pomočjo računalniškega vida je bilo objavljenih že več različnih pristopov.

V članku [7] je Camus uporabil optični tok. Da je uspel optični tok izračunati v realnem času, je uporabil algoritem, ki namesto preiskovanja velike okolice slikovnega elementa pregleda le osem-soseščine v slikah ob časih $t-1$ do $t-n$. Dobljen optični tok je izračunan za vsak slikovni element. Povprečje komponent enotskih vektorjev optičnega toka predstavlja odmik fokusa ekspanzije od središča slike. Ob predpostavki, da se kamera premakne naravnost naprej, iz vektorjev optičnega toka in izračunanega fokusa ekspanzije izračuna odmik ter hitrost odmikanja točk od fokusa ekspanzije. Nato izračuna čas do trka. Ta metoda zahteva izračun gostega optičnega toka.

Cohen in Bryne[8] uporabita značilne točke SIFT [23], a imata zaradi specifične domene (majhna zračna plovila) na voljo tudi podatke o vztrajnosti (z uporabo senzorjev premikanja in rotacije). Fokus ekspanzije izračunata s pomočjo podatkov o vztrajnosti in notranjih parametrov kamere. Na isti način ocenita tudi bistveno matriko (angl. essential matrix) med položajema kamere ob času $t-1$ ter t. Dobljeno bistveno matriko potem izboljšata z uporabo algoritma RANSAC [9], ki iz ujemanj značilnih točk izračuna bistveno matrko, ki se najbolj sklada z ujemanji. Premike značilnih točk nato projicirata na epipolarne premice in s projekcijami izračunata čas do trka. Natančnost te metode je odvisna od natančnosti podatkov o vztrajnosti.

Tretji način izračuna je sledenje objektom in uporaba hitrosti spreminjanja njihove velikosti. Negre et al. [24] na sliki poiščejo značilne točke, nato pa s pomočjo Laplaceve piramide izračunajo "skalo" teh točk. Točkam skozi čas sledijo z uporabo metode opisane v članku [22]. Iz hitrosti spreminjanja skale značilne točke potem izračunajo čas do trka. Raziskovalci v članku [26] uporabijo klasifikacijo za detekcijo vozil na sliki. Ko najdejo isto vozilo na zaporednih slikah, lahko s pomočjo informacije o spremembi velikosti vozila na sliki izračunajo čas do trka.

V članku [15] raziskovalci predstavijo direktno metodo za izračun časa do trka brez uporabe optičnega toka, značilnih točk ali detekcije objektov. Ob predpostavki, da se svetlost točke skozi čas ne spreminja, z analizo sprememb gradientov slike skozi čas izračunajo čas do trka.

1.2 Cilji in prispevki

Veliko načinov za izračun časa do trka s pomočjo računalniškega vida že obstaja, vendar so pogosto računsko zahtevne. V diplomski nalogi smo želeli pri ocenjevanju časa do trka uporabiti novejše, hitre detektorje značilnih točk in opisnike regij ter tako implementirati računsko učinkovito robustno metodo za detektiranje časa do trka brez uporabe dragih senzorjev in velikih količin računske moči. Implementirana metoda je zaradi uporabe značilnih

točk robustna na homogene regije in zaradi uporabe geometrijskih omejitev robustna na šum.

Za namen eksperimentalnega vrednotenja naše metode smo zajeli tudi zbirko posnetkov, v katerih se kamera premika naravnost naprej proti oviri. Za vsako oviro smo posneli več posnetkov, tako da je mogoče izračunati varianco metode.

1.3 Sestava diplomske naloge

V Poglavju 2 so opisane teoretične osnove konceptov, ki smo jih uporabili pri izdelavi diplomske naloge, kot je polje gibanja, detektorji in opisniki značilnih točk, epipolarna geometrija ter izračun časa do trka. Poglavje 3 vsebuje podroben opis implementacije našega algoritma za izračun časa do trka ter pripadajoče vizualizacije. V Poglavju 4 eksperimentalno pokažemo odstopanje epipola od fokusa ekspanzije pri šumnih podatkih ter opišemo zajem posnetkov, na katerih smo algoritem testirali. Predstavimo in interpretiramo zbrane meritve in izračune. Poglavje 5 vsebuje zaključke, do katerih smo prišli, ter opisuje možnosti za izboljšave.

Poglavje 2

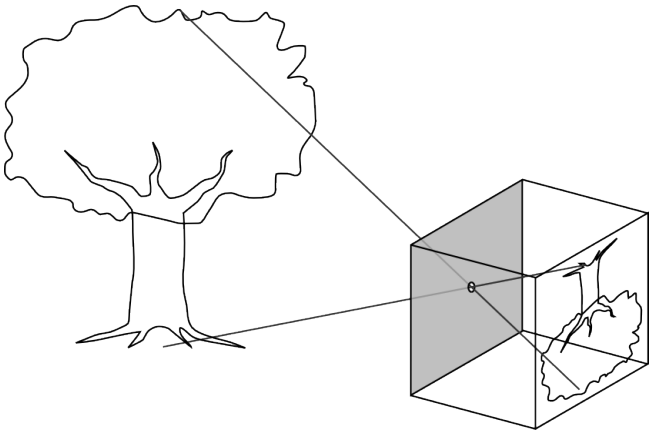
Teoretični okvir

V nadaljnjih poglavjih bomo opisali teoretične osnove konceptov, ki smo jih uporabili pri ocenjevanju časa do trka. Najprej opišemo centralno projekcijski model kamere (Poglavje 2.1) in izpeljemo enačbo za izračun časa do trka (Poglavje 2.2). Redko polje gibanja (Poglavje 2.3) smo izračunali z uporabo značilnih točk, detektorje in opisnike katerih smo opisali v Poglavju 2.4. Ujemanja smo dodatno filtrirali z uporabo epipolarne geometrije (Poglavje 2.5).

2.1 Centralno projekcijski model kamere

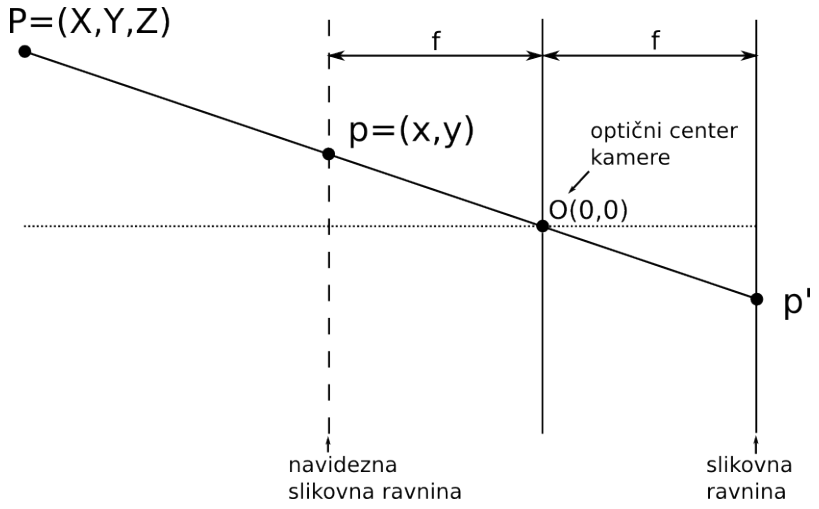
Pri računanju časa do trka smo uporabili centralno projekcijski model kamere (angl. pinhole camera). Če imamo škatlo, ki ima na sprednji strani majhno odprtino, in pred njo postavimo osvetljen predmet, se na notranji zadnji strani škatle pojavi obrnjena slika predmeta. Sliko ustvarijo žarki, ki se odbijejo od predmeta in skozi odprtino osvetlijo zadnji del škatle (glej Sliko 2.1). Odprtina škatle ima vlogo *optičnega centra kamere*, zadnja stran škatle pa predstavlja *slikovno ravnino*. Razdaljo med optičnim centrom kamere in slikovno ravnino imenujemo *goriščna razdalja* in jo označimo z f.

Opazujemo točko $\mathbf{P} = (X, Y, Z)$ v prostoru s koordinatnim sistemom z izhodiščem v optičnem centru kamere. Žarek potuje iz točke $\mathbf{P}$ skozi optični center kamere in na slikovni ravnini ustvari *sliko* točke $\mathbf{P}$, ki jo označimo s $\mathbf{p}'$.



Slika 2.1: Centralno projekcijski model kamere

Tako pridobljena slika predmeta je narobe obrnjena, zato za računanje raje uporabimo *navidezno* slikovno ravnino, ki je enaka slikovni ravnini zrcaljeni preko optičnega centra kamere. Na navidezni slikovni ravnini se točka $\mathbf{P}$ projicira v sliko $\mathbf{p} = (x, y)$ (glej Sliko 2.2).



Slika 2.2: Projekcija točke na slikovno ravnino

Z uporabo podobnih trikotnikov lahko zapišemo zvezi

$$\frac{y}{f} = \frac{Y}{Z} \tag{2.1}$$

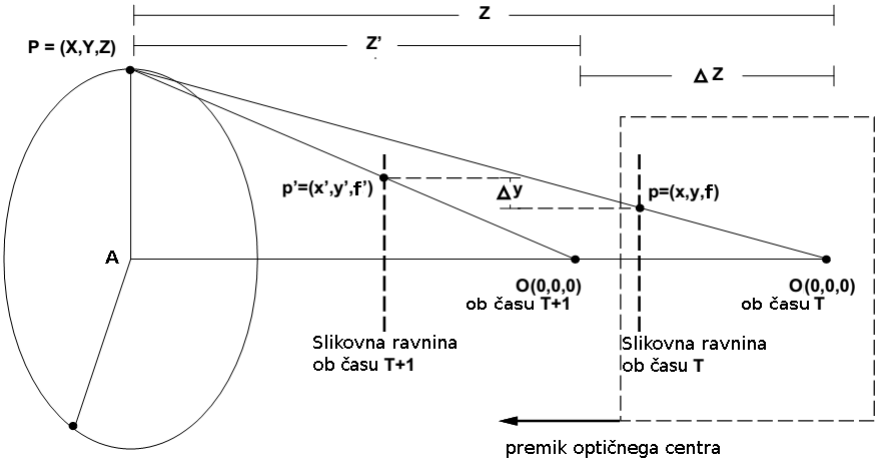
in

$$\frac{x}{f} = \frac{X}{Z}. \tag{2.2}$$

2.2 Čas do trka

Tudi če ne poznamo niti hitrosti premikanja kamere niti oddaljenosti od ovire, lahko izračunamo čas, ob katerem bo kamera prišla v stik z oviro.

Opazujemo točko **P** s koordinatami (X, Y, Z) (glej Sliko 2.3). **P** je fiksirana v prostoru, medtem ko se središče kamere, ki je tudi izhodišče koordinatnega sistema, premika proti neki drugi točki s hitrostjo $\dot{Z}$. V Poglavju 2.3 smo sliko **e** te točke poimenovali fokus ekspanzije (angl. focus of expansion, FOE), v opisanem koordinatnem sistemu pa se nahaja na koordinatah $(0, 0)$. Za lažje računanje oddaljenosti točke **P** od fokusa ekspanzije zavrtimo koordinatni sistem okoli z osi tako, da je x koordinata točke **P** enaka 0. Tako lahko njeno oddaljenost od fokusa ekspanzije izrazimo le s koordinato y. Slikovna ravnina je za goriščno razdaljo f oddaljena od središča kamere.



Slika 2.3: Geometrija časa do trka. Slika je povzeta po [7].

Točki **P** na slikovni ravnini pripada slika **p**. Med približevanjem kamere točki **P** se slika **p** spreminja. S pomočjo podobnih trikotnikov lahko zapišemo:

$$\frac{y}{f} = \frac{Y}{Z}. \tag{2.3}$$

Če dobljeno enačbo odvajamo po času, dobimo:

$$\frac{\dot{y}}{f} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2}. \tag{2.4}$$

Ker je točka **P** fiksirana v prostoru, velja $\dot{Y} = 0$. Če točka **P** ni fiksirana v smeri koordinatne osi z, jo lahko gledamo kot stacionarno in njeno hitrost prištejemo hitrosti kamere. Če pa **P** ni fiksirana v smeri koordinatnih osi x ali y, pa brez informacije o $\dot{X}$ in $\dot{Y}$ ni mogoče natančno izračunati časa do trka. Uporabimo še zvezo $Y = yZ$, iz česar sledi

$$\frac{\dot{y}}{f} = \frac{-y\dot{Z}}{Z}. \tag{2.5}$$

Delimo z y in vzamemo obratne vrednosti:

$$\frac{yf}{\dot{y}} = -\frac{Z}{\dot{Z}} = \tau. \tag{2.6}$$

Dobljena količina τ je znana kot čas do trka (angl. time to contact, TTC) [7] in nam pove razmerje med razdaljo od ovire in hitrostjo približevanja kamere oviri. V splošnem ne ležijo vse točke na ravnini yz, fokus ekspanzije pa ne leži vedno v točki $(0, 0)$. Pri računanju oddaljenosti točke od fokusa ekspanzije uporabimo evklidsko razdaljo, kar nam da enačbo

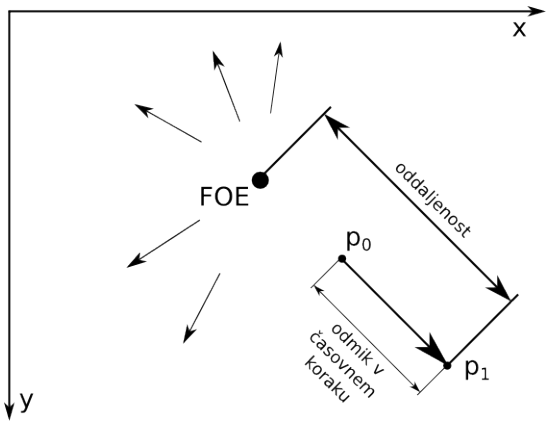
$$\frac{f \cdot d(p, FOE)}{\frac{d}{dt} d(p, FOE)} = \tau. \tag{2.7}$$

Imamo kamero, ki se premika v smeri fokusa ekspanzije, sliki ob času t in $t + \Delta t$ ter lokacijo projekcij točke **P** v obeh slikah (glej Sliko 2.4). Hitrost odmikanja točke od fokusa ekspanzije aproksimiramo z diferenčnim kvocientom:

$$\frac{d(p_1, \mathrm{FOE}) - d(p_0, \mathrm{FOE})}{\Delta t} \tag{2.8}$$

in tako dobimo enačbo za čas do trka s točko P:

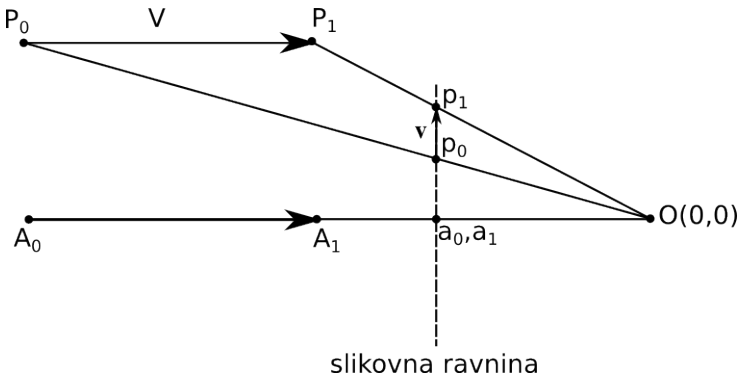
$$\frac{f \cdot \Delta t \cdot d(p_1, \mathrm{FOE})}{d(p_1, \mathrm{FOE}) - d(p_0, \mathrm{FOE})} = \tau. \tag{2.9}$$



Slika 2.4: Razdalje, ki jih uporabimo pri računanju časa do trka.

2.3 Polje gibanja

Ko se predmet, ki ga opazujemo s kamero, premika, se spreminja tudi njegova slika. Za vsak vektor premika **V** točke **P** na predmetu imamo na slikovni ravnini vektor **v**, ki predstavlja premik slike **p** (glej Sliko 2.5). Polje vseh vektorjev premika na slikovni ravnini imenujemo *polje gibanja* [34, 25].



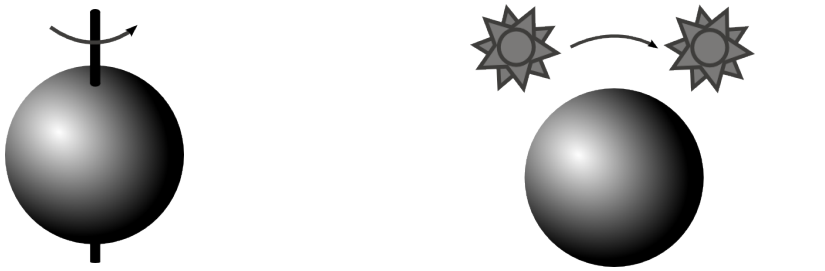
Slika 2.5: Premik predmeta na slikovni ravnini ustvari polje gibanja.

Če se kamera premika v smeri neke točke **A**, se slika te točke na slikovni ravnini ne spreminja (Slika 2.5). Ob pogledu na polje gibanja se zdi, da so vsi vektorji premika usmerjeni stran od te točke. Sliko točke, proti kateri se kamera premika, zato imenujemo *fokus ekspanzije* in označimo z **e**.

2.3.1 Direkten izračun polja gibanja

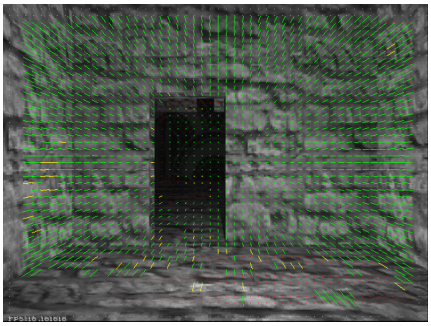
Za oceno polja gibanja lahko uporabimo direktne metode, ki so podrobneje opisane v [17]. Najpogostejša med njimi, *optični tok* (angl. optical flow), je porazdelitev navideznega gibanja svetlobnih vzorcev v sliki [16].

V idealnem primeru je optični tok enak polju gibanja, v splošnem pa je to res le ob določenih pogojih [34]. Primer, ko optični tok ni enak polju gibanja, je vrteča se krogla, saj se svetlobni vzorci ob vrtenju ne spreminjajo (Slika 2.6a). Če spremenimo položaj osvetlitve krogle, pa dobimo navidezno gibanje, čeprav krogla miruje [16] (Slika 2.6b). Izračunamo lahko gosti optični tok, ki izračuna premik vsakega slikovnega elementa, ali pa redki optični tok, kjer najprej poiščemo zanimive točke in potem izračunamo premike le-teh točk. Primer gostega optičnega toka je prikazan na Sliki 2.7.



(a) Nezaznano premikanje (b) Napačno zaznano premikanje

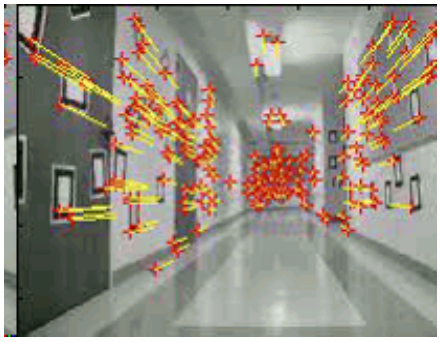
Slika 2.6: Napake optičnega toka



Slika 2.7: Primer gostega optičnega toka. Slika je povzeta po [32].

2.3.2 Izračun polja gibanja z značilnimi točkami

Medtem ko direktne metode za oceno polja gibanja uporabijo vse slikovne elemente in nam dajo gosto polje vektorjev, pa se slabše odrežejo pri večjih premikih in homogenih regijah, kjer jih je nemogoče oceniti brez dodatne obdelave. Za velike premike in posnetke, ki vsebujejo homogene regije, so bolj primerne metode z značilnimi točkami [33]. Te v sliki najprej poiščejo značilne točke in njihove opisnike, nato pa sledijo tem točkam skozi zaporedje slik. Dobljeno polje gibanja je bolj redko kot polje, pridobljeno z direktnimi metodami, lahko pa postopek naredimo bolj robusten. Zaradi šuma so namreč prisotna napačna ujemanja, kar nakazuje potrebo po dodatnih omejitvah. Za filtriranje napačnih ujemanj uporabimo geometrijske omejitve dveh pogledov.



Slika 2.8: Polje gibanja pridobljeno z uporabo značilnih točk. Slika je povzeta po [4].

2.4 Značilne točke in opisniki

Če želimo povezati vsebino ene slike z vsebino druge, moramo v obeh slikah najti skupne točke. To naredimo tako, da najprej v vsaki sliki neodvisno poiščemo t.i. značilne točke. To so točke, ki imajo razpoznavno okolico, po kateri jih lahko ponovljivo razpoznamo. Primer takih točk so koti. Preprost Harrisov detektor [12] kote zazna z analizo gradientov okolice dane točke.

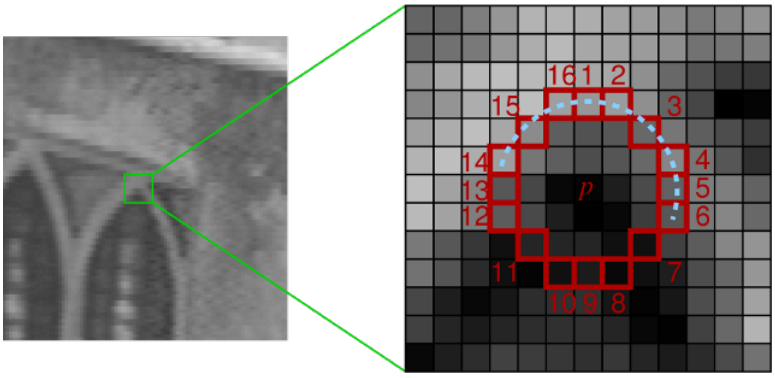
Značilne točke iz obeh slik moramo znati primerjati med sabo, da lahko najdemo ujemajoče se pare. Pri primerjanju točk uporabimo njihovo okolico, ki jo lahko opišemo na različne načine z uporabo opisnikov. Najpreprostejši opisnik je spisek intenzitet slikovnih elementov v dani okolici, ampak je zelo občutljiv že na majhne premike in fotometrične spremembe, zato se ga v praksi ne uporablja. Naprednejši detektorji so lahko invariantni na skalo in orientacijo točke in njene okolice, kot je npr. popularen detektor in opisnik SIFT[23] ali hitrejši SURF[2]. Nekateri algoritmi za detekcijo in opis se osredotočijo na hitrost in učinkovitost. Med njimi sta tudi hiter detektor značilnih točk FAST[27] in opisnik regij BRIEF[6].

2.4.1 FAST

Algoritem FAST (angl. Features from Accelerated Segment Test) je bil ustvarjen za realnočasno zaznavanje kotov v sliki [27]. Za vsak slikovni element p pregleda 16 slikovnih elementov, ki ležijo na diskretizirani krožnici s središčem v p in polmerom 4 slikovne elemente. Če ima na krožnici vsaj $n = 12$ zaporednih elementov ali nižjo ali višjo intenziteto kot p, je p prepoznan kot značilna točka (glej Sliko 2.9).

Postopek je dodatno optimiziran tako, da med oštevilčenimi slikovnimi elementi na krožnici (glej Sliko 2.9) najprej preverimo slikovne elemente 1, 9, 5 in 13. Tako lahko hitro zavržemo slikovne elemente, ki zagotovo niso značilne točke. V primeru značilne točke morajo imeti namreč vsaj trije od štirih preverjenih elementov intenziteto ali višjo ali nižjo kot p.

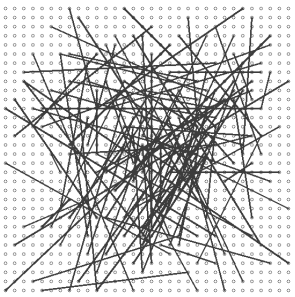
Kote lahko detektiramo še hitreje in za $n < 12$ [28], če za odločanje o prisotnosti kota na slikovnem elementu p uporabimo odločitveno drevo, strojno naučeno na množici kotov detektiranih z algoritmom FAST.



Slika 2.9: Iskanje kotov z algoritmom FAST. Na povečavi okolice potencialne značilne točke je prikazana diskretizirana krožnica okoli p ter oštevilčeni elementi, ki ležijo na krožnici. Prekinjena črta označuje 12 zaporednih elementov, ki imajo v tem primeru višjo intenziteto kot p. Slika je povzeta po [27].

2.4.2 BRIEF

Opisnik BRIEF (angl. Binary Robust Independent Elementary Features) okolico značilne točke opiše z nizom bitov dolžine $n_d \in \{128, 256, 512\}$. V okolici značilne točke, zglajeni z Gaussovim filtrom, med seboj primerja intenzitete n_d parov slikovnih elementov. Pari točk za primerjanje so vzorčeni iz normalne porazdelitve s sredino v središču okolice značilne točke (Slika 2.10).



Slika 2.10: Izbira $n_d = 128$ parov slikovnih elementov za gradnjo opisnika z vzorčenjem iz normalne porazdelitve. Slika je povzeta po [6].

Na okolici $\mathbf{p}$ značilne točke p je definiran test τ [6]

$$\tau(\mathbf{p}; \mathbf{x_1}, \mathbf{x_2}) := \begin{cases} 1; & \mathbf{p}(\mathbf{x_1}) < \mathbf{p}(\mathbf{x_2}) \\ 0; & \text{sicer} \end{cases}, \tag{2.10}$$

kjer je $\mathbf{p}(\mathbf{x})$ intenziteta slikovnega elementa $\mathbf{x} = (x, y)$ v zglajeni okolici značilne točke p. Z izbiro jedra glajenja ter množice n_d $(\mathbf{x}_1, \mathbf{x}_2)$ parov je opisnik enolično določen in definiran z

$$\mathbf{b}_{n_d}(\mathbf{p}) = \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_{1i}, \mathbf{x}_{2i}). \tag{2.11}$$

Opisnik BRIEF ni rotacijsko invarianten, je pa rotacijsko invarianten opisnik ORB (angl. Oriented BRIEF) [29], ki je razširitev opisnika BRIEF.

Za iskanje ujemajočih se parov opisnikov moramo znati izračunati razdaljo med njimi. Pri opisniku BRIEF je razdalja med opisnikoma definirana kot Hammingova razdalja med njima. Izračun Hammingove razdalje je na večini procesorjev zelo učinkovit, saj nad opisnikoma uporabimo le XOR operacijo in preštejemo bite, ki so v rezultatu postavljeni na 1.

2.4.3 Filtriranje ujemanj

Ko za opisnik iz prve slike najdemo opisnik iz druge slike, ki mu je najbližji po razdalji, zaradi šuma še ni nujno, da opisnika res opisujeta isto značilno točko. Da bi se znebili čim več napačnih ujemanj, ujemanja filtriramo po različnih kriterijih. Najpopularnejši je Lowejev [23] kriterij, ki zavrže ujemanja, pri katerih je razmerje med oddaljenostjo prvega najbližjega opisnika in oddaljenostjo drugega najbližjega opisnika večja kot 0.8.

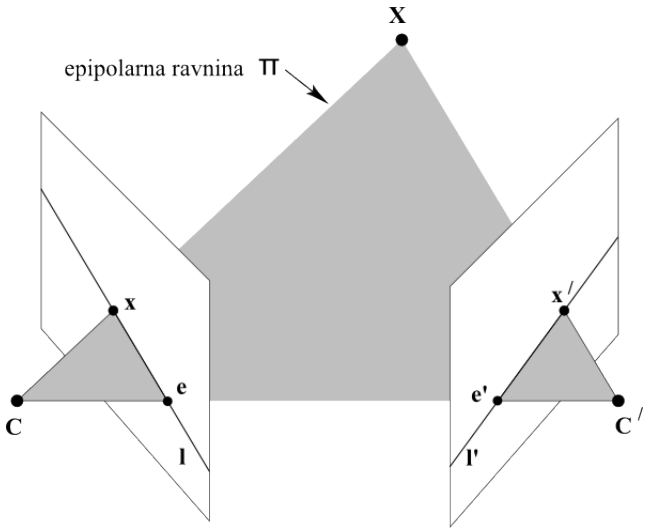
2.5 Epipolarna geometrija

Denimo, da na točko $\mathbf{X}$ v trirazsežnem prostoru gledamo skozi dve različni kameri s središčima $\mathbf{C}$ in $\mathbf{C}'$ (Slika 2.11). Sliko točke na slikovni ravnini prve kamere označimo z $\mathbf{x}$, sliko točke na slikovni ravnini druge kamere pa z $\mathbf{x}'$.

Potem točke $\mathbf{C}$, $\mathbf{C}'$, $\mathbf{x}$, $\mathbf{x}'$ in $\mathbf{X}$ ležijo na isti *epipolarni ravnini*, ki jo označimo s π.

Če poznamo središči kamer $\mathbf{C}$ in $\mathbf{C}'$ ter sliko $\mathbf{x}$, ne pa tudi slike $\mathbf{x}'$, lahko koplanarnost zgoraj omenjenih točk uporabimo za zmanjšanje preiskovalnega prostora za lokacijo slike $\mathbf{x}'$ na slikovni ravnini druge kamere. Epipolarna ravnina je namreč določena z *osnovnico* (angl. baseline) $\overrightarrow{\mathbf{CC}'}$ ter vektorjem $\overrightarrow{\mathbf{Cx}}$ in ker točka $\mathbf{x}'$ leži na epipolarni ravnini, je njena lokacija omejena le na presečišče epipolarne ravnine in slikovne ravnine druge kamere. Dobljeno premico označimo z $\mathbf{l}'$ in jo imenujeno *epipolarna premica*.

Različnim točkam v prostoru pripadajo različne epipolarne ravnine, vse pa se sekajo v osnovnici. Pripadajoče epipolarne premice na vsaki od slikovnih ravnin se sekajo v točkah $\mathbf{e}$ in $\mathbf{e}'$, ki ju imenujemo *epipola*.



Slika 2.11: Epipolarna geometrija. Slika je povzeta po [13].

2.5.1 Fundamentalna matrika

Če imamo podani dve sliki, potem za vsako točko $\mathbf{x}$ s prve slike obstaja pripadajoča epipolarna premica $\mathbf{l}'$ na drugi sliki. Točka $\mathbf{x}'$ z druge slike, ki pripada točki $\mathbf{x}$, mora ležati na premici $\mathbf{l}'$. Ta epipolarna premica je projekcija

vektorja $\overrightarrow{\mathbf{Cx}}$ na drugo sliko, torej obstaja preslikava $\mathbf{x} \mapsto \mathbf{l}'$ iz točke na prvi sliki v epipolarno premico na drugi sliki (glej Sliko 2.11).

Denimo, da imamo v prostoru ravnino π, ki ne seka nobenega izmed središč kamer. Ker sta tako množica točk $\mathbf{x}_i$ kot množica točk $\mathbf{x}'_i$ projekciji množice točk $\mathbf{X_i}$ na ravnini π, obstaja med množicama 2D homografija H_π.

$$\mathbf{x}'_i = \mathbf{H}_\pi \mathbf{x}_i \tag{2.12}$$

Točka $\mathbf{x}'$ leži na epipolarni premici $\mathbf{l}'$, $\mathbf{l}'$ pa poteka skozi epipol $\mathbf{e}'$. V homogenem koordinatnem sistemu je premica določena kot $\mathbf{l} = (a, b, c)^\top$ in zadošča enačbi $ax + by + cz = 0$. Če torej želimo najti koeficiente enačbe premice $\mathbf{l}'$, moramo zadostiti enačbama

$$\mathbf{l'}^\top \mathbf{e}' = \mathbf{0} \tag{2.13}$$

in

$$\mathbf{l'}^\top \mathbf{x}' = \mathbf{0}, \tag{2.14}$$

rešitev pa najlažje dobimo kot vektorski produkt

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}', \tag{2.15}$$

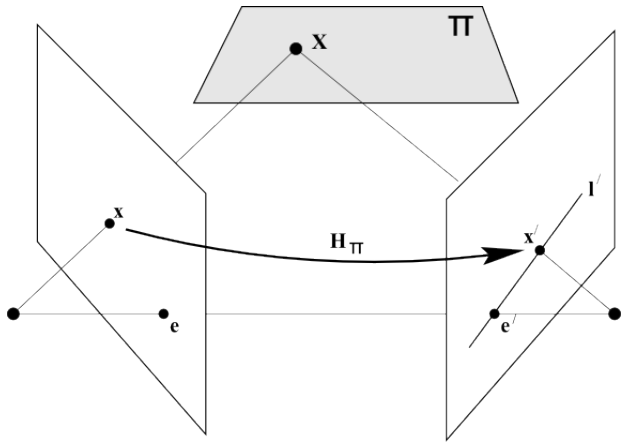
kjer je $[\mathbf{e}']_\times \mathbf{x}'$ matrična oblika vektorskega produkta [20]. Ker lahko $\mathbf{x}'$ zapišemo kot $\mathbf{x}' = \mathbf{H}_\pi \mathbf{x}$, dobimo

$$\mathbf{l}' = [\mathbf{e}']_\times \mathbf{H}_\pi \mathbf{x} = \mathbf{F}\mathbf{x}, \tag{2.16}$$

torej velja

$$\mathbf{F} = [\mathbf{e}']_\times \mathbf{H}_\pi. \tag{2.17}$$

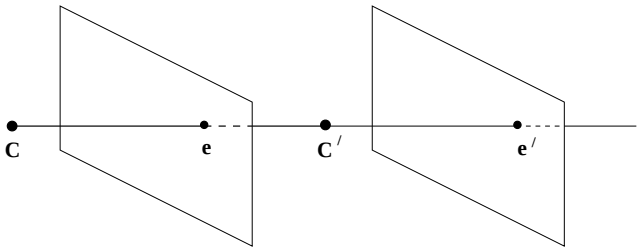
Izkaže se, da je preslikava (2.17) pravzaprav projektivna preslikava iz točk v premice, ki jo izrazimo z matriko $\mathbf{F}$ [13]. Matrika $\mathbf{F}$, imenovana *fundamentalna matrika*, je algebraična predstavitev epipolarne geometrije.



Slika 2.12: Preslikava točke $\mathbf{x}$ v točko $\mathbf{x}'$. Slika je povzeta po [13].

2.5.2 Lokacija epipola ob premiku naprej

Če opazujemo premik kamere naprej, kot je prikazano na Sliki 2.13, opazimo, da je lokacija epipola v obeh slikah enaka. Epipola v obeh slikah sovpadata s sliko točke, proti kateri se kamera premika. Sliko te točke smo v Poglavju 2.3 poimenovali fokus ekspanzije.



Slika 2.13: Epipolarna geometrija ob premiku kamere naprej. Slika je povzeta po [13].

2.5.3 Izračun fundamentalne matrike

Fundamentalno matriko lahko enostavno izračunamo, če poznamo dovolj veliko množico točk na prvi slikovni ravnini ter njihove pripadajoče točke na drugi slikovni ravnini [21, 14]. Opisali bomo osemtočkovni algoritem za izračun fundamentalne matrike.

Iz enačb (2.14) in (2.16) sledi

$$\mathbf{x'}^{\top}\mathbf{F}\mathbf{x} = \mathbf{0} \tag{2.18}$$

za vse točke $\mathbf{x}$ na prvi slikovni ravnini in pripadajoče točke $\mathbf{x'}$ na drugi slikovni ravnini. Če pišemo $\mathbf{x} = (u, v, 1)^{\top}$ in $\mathbf{x'} = (u', v', 1)^{\top}$ ter napišemo enačbo (2.18) kot

$$\begin{pmatrix} u', & v', & 1 \end{pmatrix} \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{0} \tag{2.19}$$

in jo razpišemo, dobimo za vsak par ujemajočih se točk po eno linearno enačbo oblike

$$\begin{pmatrix} uu', uv', u, vu', vv', v, u', v', 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}. \tag{2.20}$$

Te enačbe zložimo v linearni sistem

$$\mathbf{A}\mathbf{f}_F = \mathbf{0}, \tag{2.21}$$

kjer vsaka vrstica matrike $\mathbf{A}$ predstavlja koeficiente enačbe (2.20) za par ujemajočih se točk, $\mathbf{f_F}$ pa vektor devetih elementov fundamentalne matrike. Ker sta matrika $\mathbf{F}$ in posledično tudi $\mathbf{f}_F$ definirana le do skale natančno in imata le osem prostostnih stopenj, pri reševanju sistema uporabimo še omejitev

$$\|\mathbf{f}_F\| = 1. \tag{2.22}$$

Dobljen sistem enačb je slabo pogojen numerični problem [14]. Zato v praksi podatke najprej transformiramo tako, da so centrirani in je srednja kvadratna oddaljenost podatkov od centra enaka dva slikovna elementa, šele nato uporabimo osemtočkovni algoritem. Želimo, da se epipolarne premice sekajo v eni točki, zato fundamentalni matriki vsilimo rang 2. Izračunano fundamentalno matriko $\mathbf{F}_t$ razcepimo po singularnih vrednostih (angl. singular value decomposition, SVD):

$$\mathbf{F}_t \overset{\mathbf{SVD}}{=} \mathbf{UDV}^\top, \tag{2.23}$$

kjer sta matriki $\mathbf{U}$ in $\mathbf{V}$ matriki levih in desnih singularnih vektorjev, $\mathbf{D} = diag(d_{11}, d_{22}, d_{33})$ pa diagonalna matrika singularnih vrednosti. Najmanjšo singularno vrednost postavimo na 0 in dobimo $\mathbf{D}' = diag(d_{11}, d_{22}, 0)$ ter nazaj konstruiramo $\mathbf{F}'_\mathbf{t} = \mathbf{UD}'\mathbf{V}^\top$. Če sta $\mathbf{T}$ in $\mathbf{T}'$ transformaciji, s katerima smo transformirali podatke, je matrika transformirana v originalne enote enaka

$$\mathbf{F} = \mathbf{T}^\top \mathbf{F}'_\mathbf{t} \mathbf{T}'. \tag{2.24}$$

2.5.4 Algoritem RANSAC

Če imamo šumne podatke, osemtočkovni algoritem običajno uporabimo v kombinaciji z algoritmom RANSAC. RANSAC [9] je algoritem, ki nam omogoča prileganje modela tudi na zelo šumne podatke. To naredi tako, da iterativno iz množice podatkov izbere najmanjšo skupino točk, na katero lahko prilega model. Na to skupino točk prilega model in glede na prednastavljen kriterij prešteje točke, ki ta model podpirajo. Po koncu zanke vzame model, ki ga podpira največ točk, in na tistih točkah, ki ga podpirajo, izračuna končen model. V Algoritmu 1 je opisan algoritem za izračun fundamentalne matrike iz šumnih podatkov.

Reprojekcijsko napako za par točk $(\mathbf{x}, \mathbf{x}')$ izračunamo kot povprečje razdalje prve točke od epipolarne premice v prvi sliki in razdalje druge točke od epipolarne premice v drugi sliki:

$$\text{reproj\_napaka}(\mathbf{x}, \mathbf{x}', \mathbf{F}) = \frac{d(\mathbf{x}, \mathbf{F}\mathbf{x}') + d(\mathbf{x}', \mathbf{F}^\top \mathbf{x})}{2}. \tag{2.25}$$

Algoritem 1 Algoritem RANSAC za izračun fundamentalne matrike.

Vhod: pari ujemajočih se točk $(\mathbf{x}, \mathbf{x}')$, ϵ in *st_ponovitev*

Izhod: fundamentalna matrika $\mathbf{F}$

1: $i \leftarrow 1$
2: **ponavljaj**
3: naključno izberi osem parov ujemajočih se točk
4: z izbranimi točkami izračunaj fundamentalno matriko $\mathbf{F}_c$
5: preštej pare, za katere je reprojekcijska napaka manjša od ϵ
6: $i \leftarrow i + 1$
7: **dokler** $i \leq$ *st_ponovitev*
8: $\mathbf{F}_b \leftarrow$ fundamentalna matrika, ki jo podpira največ parov
9: *podporniki* $\leftarrow$ pari, ki podpirajo $\mathbf{F}_b$
10: **vrni** fundamentalno matriko $\mathbf{F}$ izračunano na točkah *podporniki*

Poglavje 3

Izvedba izračuna časa do trka

V tem poglavju bomo predstavili naš algoritem za izračun časa do trka. Algoritem za izračun potrebuje dve sliki, pred in po premiku, ter čas, ki je pretekel med zajemom prve in zajemom druge slike. Na obeh slikah poišče značilne točke in poveže tiste značilne točke, ki se ujemajo, v redko polje gibanja. Iz polja gibanja izračuna fokus ekspanzije z izračunom približnega presečišča premic, ki potekajo skozi vektorje polja gibanja (Poglavje 3.1). Iz polja gibanja izračuna lokacijo fokusa ekspanzije (Poglavje 3.2). S temi podatki izračuna čas do trka za posamezne značilne točke (Poglavje 3.3) in s pomočjo hevristik oceni čas do trka za celotno sliko (Poglavje 3.4). V Poglavju 3.5 povzamemo celoten algoritem za izračun časa do trka, v Poglavju 3.6 pa predstavimo tudi izračun in vizualizacijo verjetnosti trka v posameznem predelu slike.

3.1 Polje gibanja

Polje gibanja ocenjujemo iz dveh slik, posnetih na kameri ob času t in $t + \Delta t$. Za ocenitev polja gibanja smo uporabili metodo z značilnimi točkami. Uporaba značilnih točk nam da manjšo množico točk, za katere moramo ugotoviti lokacijo na drugi sliki, kot bi jo imeli pri uporabi direktnih metod. Primer direktne metode, optični tok, opazuje vsak slikovni element slike in

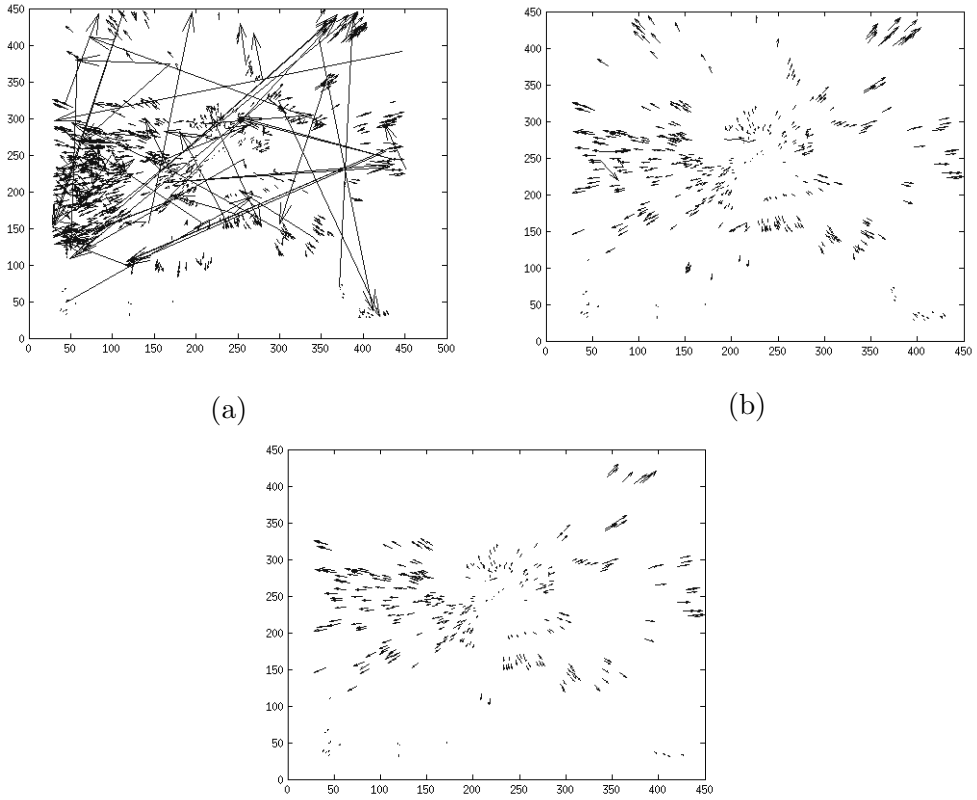
je tako algoritem računsko bolj zahteven.

Najprej potrebujemo lokacije in opisnike značilnih točk v obeh slikah. Značilne točke smo detektirali s pomočjo algoritma FAST [28], hitrim detektorjem značilnih točk, ki kote v sliki zazna z uporabo klasifikacije delčkov slike s strojno naučenim odločitvenim drevesom. FAST smo podrobneje opisali v Poglavju 2.4.1.

Opisnike dobljenih značilnih točk smo izračunali z uporabo algoritma BRIEF [6], ki vsako značilno točko opiše z nizom bitov. Podrobneje smo algoritem opisali v Poglavju 2.4.2. S pomočjo opisnikov poskusimo točkam v prvi sliki določiti pripadajoče točke v drugi sliki tako, da vsakemu opisniku značilne točke v drugi sliki najdemo najbližji opisnik značilne točke v prvi sliki glede na Hammingovo razdaljo [11]. Primer ujemanj je prikazan na Sliki 3.1a. Ujemanja nato filtriramo z Lowejevim kriterijem [23], ki je opisan v Poglavju 2.4.3 (glej Sliko 3.1b).

3.1.1 Filtriranje ujemanj z geometrijo

Sliki kamere ob času t ter ob času $t + \Delta t$ lahko obravnavamo kot sliki dveh kamer z različnima pogledoma. Brez dodatnih senzorjev ne moremo vedeti točnih parametrov premika kamere, lahko pa model kamere in geometrijo dveh pogledov uporabimo za filtriranje ujemanj. Iz pridobljenih korespondenc poskusimo izračunati fundamentalno matriko, ki se najbolj prilega danim podatkom. Za robusten izračun uporabimo algoritem RANSAC[9] v kombinaciji z osemtočkovnim algoritmom za izračun fundamentalne matrike, ki smo ga podrobneje opisali v Poglavju 2.5.4. Algoritem RANSAC je sposoben tudi iz zelo šumnih korespondenc izračunati fundamentalno matriko, ki se dobro prilega podatkom.

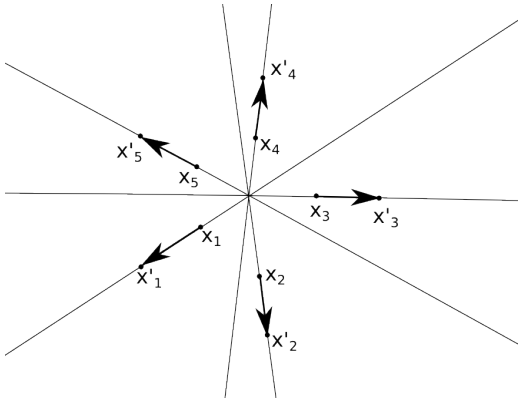


(a) (b)

(c)

Slika 3.1: (a) Začetna ujemanja, (b) ujemanja po filtriranju po razdalji, (c) ujemanja po filtriranju z algoritmom RANSAC.

Rezultat algoritma RANSAC sta fundamentalna matrika ter množica korespondenčnih parov, ki se najbolj skladajo s to matriko. Na Sliki 3.1 so prikazana začetna ujemanja značilnih točk, ujemanja po filtriranju glede na razdaljo med opisnikoma, ter ujemanja po filtriranju z algoritmom RANSAC (glej Sliko 3.1c). Dobljeno množico korespondenčnih točk, ki ustrezajo epipolarni geometriji, uporabimo za nadaljnje računanje časa do trka.



Slika 3.2: Izračun fokusa ekspanzije s pomočjo polja gibanja.

3.2 Fokus ekspanzije

Za izračun časa do trka potrebujemo fokus ekspanzije. Ena možnost za izračun je uporaba fundamentalne matrike, saj se po teoriji epipolarne geometrije pri premiku kamere naravnost naprej epipol na obeh slikah nahaja v fokusu ekspanzije. V praksi se je izkazalo, da je osemtočkovni algoritem za izračun fundamentalne matrike zelo občutljiv na šum, kar smo opisali v Poglavju 4.1. Zato smo se odločili za izračun fokusa ekspanzije s pomočjo pridobljenega polja gibanja. V idealnem primeru se premice, ki potekajo skozi vektorje polja gibanja, sekajo v eni točki. Ker pa so naši podatki šumni, moramo poiskati približno rešitev. Če so točke $\mathbf{x}_i$ značilne točke v prvi sliki ter $\mathbf{x}'_i$ pripadajoče točke v drugi sliki (glej Sliko 3.2), lahko premico skozi posamezen par ujemajočih se točk predstavimo kot $(\mathbf{x}_i, \mathbf{n}_i)$, kjer je

$$\mathbf{n}_i := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (\mathbf{x}'_i - \mathbf{x}_i)/\|\mathbf{x}'_i - \mathbf{x}_i\| \tag{3.1}$$

normala premice, ki jo dobimo tako, da enotski vektor v smeri $\mathbf{x}'_i - \mathbf{x}_i$ rotiramo za 90°. Oddaljenost potencialnega fokusa ekspanzije $\mathbf{e}$ od posamezne premice izračunamo kot projekcijo vektorja od izhodišča premice do točke $\mathbf{e}$ na enotski vektor normale:

$$d(\mathbf{e}, (\mathbf{x}_i, \mathbf{n}_i)) = \|(\mathbf{e} - \mathbf{x}_i)^\top \mathbf{n}_i\|. \tag{3.2}$$

Funkcijo napake potencialnega epipola določimo kot vsoto kvadratov razdalj od epipola do vseh premic

$$f(\mathbf{e}) = \sum_{i=1}^{n} d(\mathbf{e}, (\mathbf{x}_i, \mathbf{n}_i))^2 = (\mathbf{e} - \mathbf{x}_i)^\top \mathbf{n}_i \mathbf{n}_i^\top (\mathbf{e} - \mathbf{x}_i) \tag{3.3}$$

in jo razpišemo kot

$$f(\mathbf{e}) = \sum_{i=1}^{n} \mathbf{e}^\top \mathbf{n}_i \mathbf{n}_i^\top \mathbf{e} - \mathbf{e}^\top \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i - \mathbf{x}_i \mathbf{n}_i \mathbf{n}_i^\top \mathbf{e} + \mathbf{x}_i^\top \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i \tag{3.4}$$

$$= \mathbf{e}^\top \left(\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^\top \right) \mathbf{e} - 2\mathbf{e}^\top \left(\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i \right) + \sum_{i=1}^{n} \mathbf{x}_i^\top \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i. \tag{3.5}$$

Za iskanje minimuma odvajamo po $\mathbf{e}$:

$$\frac{\partial f(\mathbf{e})}{\partial \mathbf{e}} = 2 \left(\sum_i \mathbf{n}_i \mathbf{n}_i^\top \right) \mathbf{e} - 2 \left(\sum_i \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i \right) := \mathbf{0}, \tag{3.6}$$

kar nam da

$$\left(\sum_i \mathbf{n}_i \mathbf{n}_i^\top \right) \mathbf{e} = \left(\sum_i \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i \right). \tag{3.7}$$

Tako na koncu rešujemo linearni sistem enačb $\mathbf{A}\mathbf{e} = \mathbf{b}$, kjer je

$$\mathbf{A} = \sum_i \mathbf{n}_i \mathbf{n}_i^\top \tag{3.8}$$

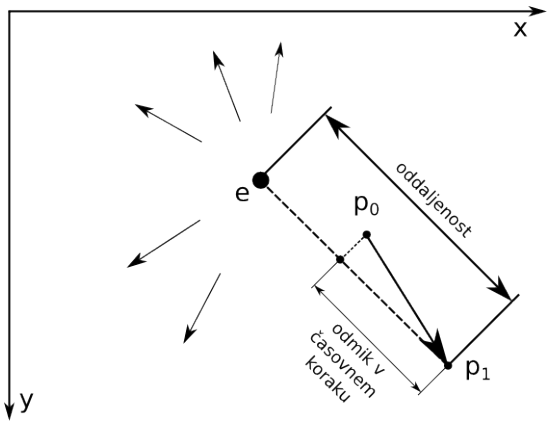
in

$$\mathbf{b} = \sum_i \mathbf{n}_i \mathbf{n}_i^\top \mathbf{x}_i. \tag{3.9}$$

Dobljen sistem linearnih enačb ima dve enačbi in dve neznanki, zato ga lahko rešimo s pomočjo Gaussove eliminacije. Rešitev sistema so koordinate fokusa ekspanzije $\mathbf{e}$, ki je v smislu kvadratov razdalj najbližji premicam, ki potekajo skozi vektorje polja gibanja.

3.3 Lokalni čas do trka

Izračunano polje gibanja je množica parov točk $(\mathbf{p}_{0i}, \mathbf{p}_{1i})$, kjer so $\mathbf{p}_{0i}$ točke iz prve, $\mathbf{p}_{1i}$ pa točke iz druge slike. To polje nam opisuje premik značilnih



Slika 3.3: Izračun odmika točke od fokusa ekspanzije.

točk $\mathbf{p}_i$ od časa $t - \Delta t$ do časa t. Po enačbi (2.9) izračunamo čas do trka za neko točko tako, da njeno oddaljenost od fokusa ekspanzije delimo z njenim odmikom od fokusa ekspanzije v določenem časovnem koraku. Čas do trka računamo ob času t, torej je

$$d(\mathbf{p}_i, \mathbf{e}) = \|\mathbf{e} - \mathbf{p}_{1i}\|. \tag{3.10}$$

Ker je izračunano polje gibanja šumno, moramo za odmik točke od fokusa ekspanzije vzeti projekcijo vektorja polja gibanja na premico skozi fokus ekspanzije in točko $\mathbf{p}_1$ (glej Sliko 3.3):

$$\Delta d(\mathbf{p}_i, \mathbf{e}) = \frac{(\mathbf{p}_{1i} - \mathbf{p}_{0i})^\top (\mathbf{e} - \mathbf{p}_{1i})}{\|\mathbf{e} - \mathbf{p}_{1i}\|} \tag{3.11}$$

in na koncu za vsako točko p_i izračunamo čas do trka

$$\tau_i = \frac{\Delta t \; d(\mathbf{p}_i, \mathbf{e})}{\Delta d(\mathbf{p}_i, \mathbf{e})}. \tag{3.12}$$

3.4 Globalni čas do trka

V primeru premikanja proti ravni površini, ki pokriva celotno sliko, bi imele vse točke enak τ, v našem primeru pa moramo poleg šuma upoštevati tudi predmete v okolici in okolico samo. Najbolje bi bilo, če bi pri računanju

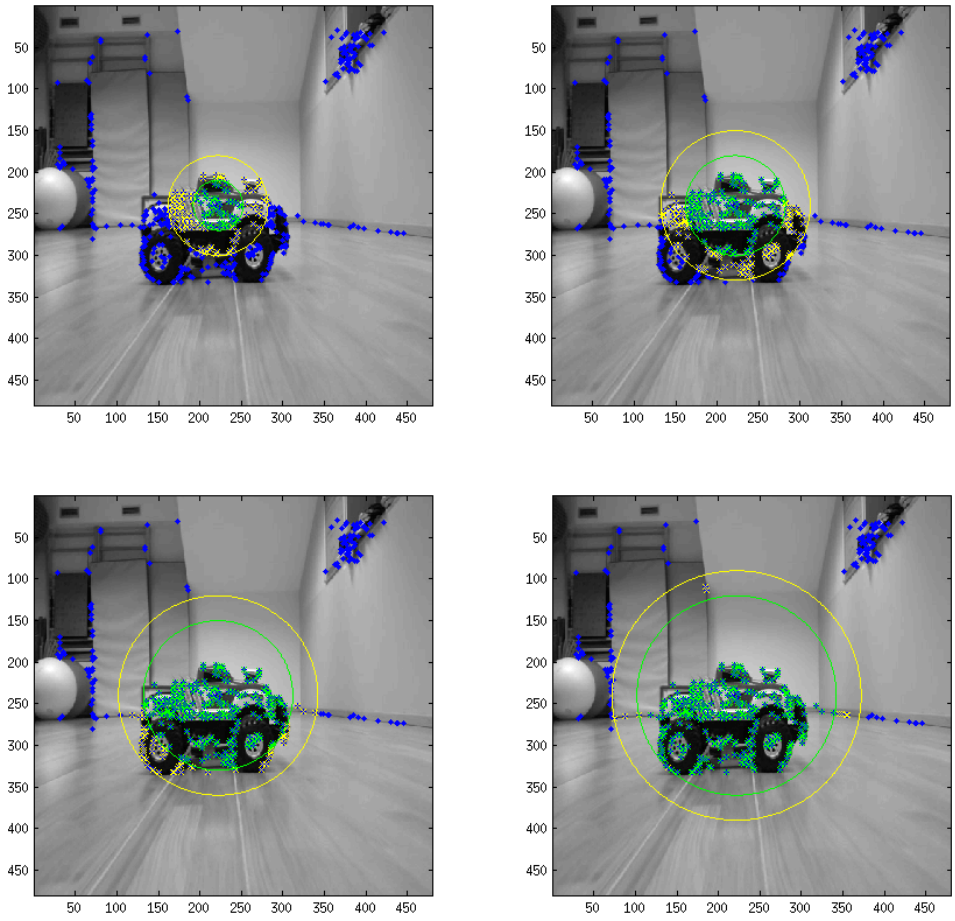
globalnega časa do trka upoštevali le točke, ki pripadajo ciljni oviri. V našem algoritmu poskušamo zagotoviti, da je takih točk vsaj več kot polovico. Zato uporabimo postopek določitve območja zanimanja.

Predpostavimo, da smo fokus ekspanzije izračunali pravilno. Za začetno območje zanimanja nastavimo krog s središčem v fokusu ekspanzije in radijem velikosti 1/16 širine slike. Krog povečujemo za 1/16 širine slike in vsakič preštejemo, koliko novih točk se nahaja znotraj kroga. Če se število točk poveča za več kot polovico povečanja prejšnje iteracije, se postopek nadaljuje, drugače pa se povečanje kroga zavrne in postopek se zaključi. Postopek je povzet v Algoritmu 2, primer korakov pri določitvi območja zanimanja pa prikazan na Sliki 3.4.

Algoritem 2 Določitev območja zanimanja

```
1: polmer ← 1/16 širine slike
2: tock_prej ← st. tock znotraj kroga K(FOE, polmer)
3: dodanih_tock_prejsnjic ← 0
4: ponavljaj
5:    polmer_nov ← polmer + 1/16 širine slike
6:    tock_potem ← st. tock znotraj kroga K(FOE, polmer_nov)
7:    dodanih_tock ← tock_potem - tock_prej
8:    če dodanih_tock > 0.5· dodanih_tock_prejsnjic potem
9:       polmer ← polmer_nov
10:      tock_prej ← tock_potem
11:      dodanih_tock_prejsnjic ← dodanih_tock
12:   drugače
13:      prekini ponavljanje
14:   končaj če
15: končaj ponavljaj
```

Iz točk, ki se nahajajo na območju zanimanja, poskušamo čim bolj robustno oceniti čas do trka za celotno sliko. Ker so podatki še vedno šumni, čase do trka za točke iz območja zanimanja najprej uredimo po velikosti.



Slika 3.4: Določitev območja zanimanja

Spodnjih in zgornjih 20% vrednosti odrežemo, na preostanku pa izračunamo povprečje (glej Algoritem 3).

Algoritem 3 Izračun globalnega časa do trka

Vhod: območje zanimanja, množica τ_i

Izhod: globalni čas do trka T

1: časi $\leftarrow \tau_i$, za $i \in \{i \mid \mathbf{x}'_i \in$ območje zanimanja$\}$
2: časi $\leftarrow sortiraj$(časi)
3: časi $\leftarrow$ odreži prvih 20% in zadnjih 20% elementov
4: $T \leftarrow povprecje$(časi)
5: **vrni** T

3.5 Povzetek algoritma

Celoten algoritem za izračun časa do trka je povzet v Algoritmu 4. Koraki 1–6 so opisani v Poglavju 3.1, korak 7 v Poglavju 3.2, korak 9 v Poglavju 3.3 in korak 10 v Poglavju 3.4.

Algoritem 4 Izračun časa do trka

Vhod: sliki ob času $t - \Delta t$ in t

Izhod: čas do trka

1: izračunaj značilne točke v obeh slikah
2: izračunaj opisnike značilnih točk v obeh slikah
3: poišči ujemanja med značilnimi točkami v obeh slikah
4: filtriraj ujemanja glede na njihovo oddaljenost
5: filtriraj ujemanja glede na epipolarno geometrijo
6: iz ujemanj konstruiraj polje gibanja
7: iz polja gibanja izračunaj fokus ekspanzije
8: določi območje zanimanja
9: za značilne točke v drugi sliki izračunaj čas do trka
10: izračunaj čas do trka za celotno sliko

3.6 Porazdelitev v regije

Poleg globalnega časa do trka smo za grafično vizualizacijo izračunali še oceno verjetnosti trka v posameznem delu slike. Sliko najprej razdelimo na $k \times k$ regij, kjer je k poljubno celo število. Verjetnost trka na posamezni regiji slike smo modelirali z

$$P_{trk} = P_{TTC} \cdot P_{konsenz} \cdot P_{st.p.}. \tag{3.13}$$

Člen P_{TTC} je verjetnost trka ocenjena iz izračunanih časov do trka za točke v regiji. Manjši kot je čas do trka, večja je verjetnost trka. Parameter $\sigma_{TTC} = 50$ izbran tako, da je 60 slik pred trkom, kar je pri zajemanju s hitrostjo 30 slik na sekundo dve sekundi, približno 50% verjetnost trka.

$$P_{TTC} = \exp\left(-\frac{1}{2\sigma_{TTC}^2} med(\{\tau_i \in regija\})^2\right) \tag{3.14}$$

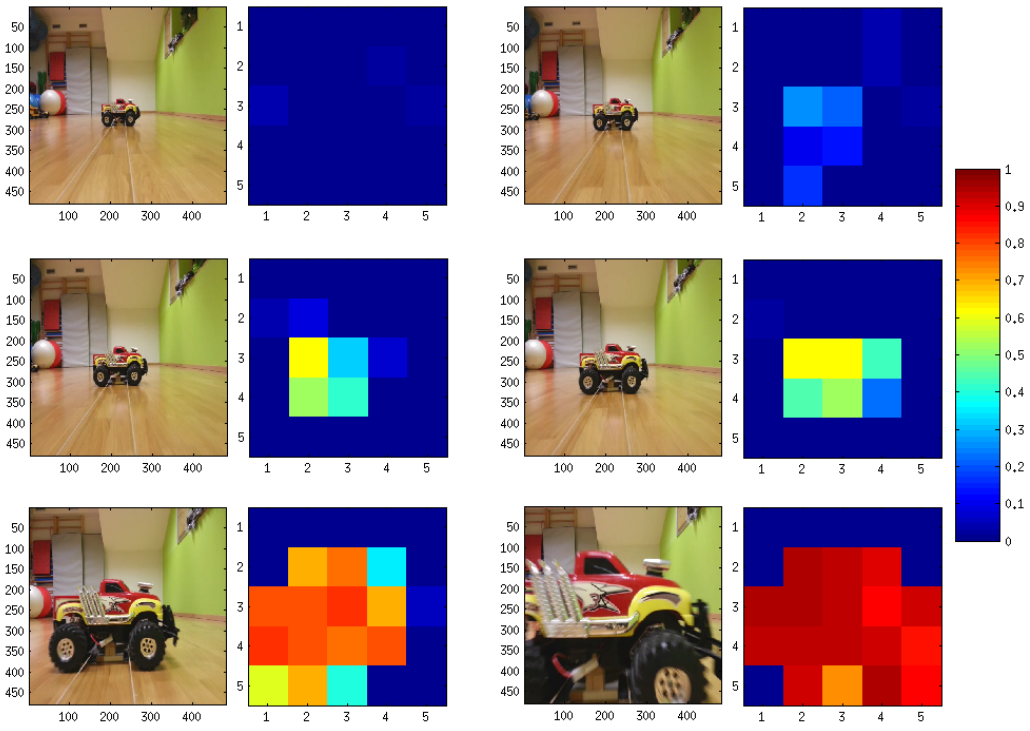
Vendar pa verjetnost trka, izračunana le iz izračunanih časov do trka posameznih točk, ni robustna. Regija lahko vsebuje veliko šumnih točk, kar pomeni, da meritev ni zanesljiva. Nezanesljivim meritvam znižamo težo z uporabo člena $P_{konsenz}$, ki izračuna verjetnost meritve glede na mediano odmika od mediane (angl. median absolute deviation, MAD) izračunanih časov do trka za točke v regiji. Parameter $\sigma_{konsenz} = 20$ je izbran tako, da ima regija z 20 slikami do trka MAD le 60% teže, regija z 70 slikami MAD pa že praktično 0%.

$$P_{konsenz} = \exp\left(-\frac{1}{2\sigma_{konsenz}^2} MAD(\{\tau_i \in regija\})^2\right) \tag{3.15}$$

Regija lahko vsebuje tudi premalo točk za zanesljiv izračun. Člen $P_{st.p.}$ določa verjetnost meritve glede na količino točk, ki to meritev podpira. Parameter $\sigma_{st.p.} = 2$ je nastavljen tako, da ima regija z le eno točko samo 20% teže, regija s štirimi točkami pa že 80% teže.

$$P_{st.p.} = \frac{n^2}{\sigma_{st.p.} n^2} \tag{3.16}$$

Parameter n je število točk z izračunanim časom do trka v regiji. Na Sliki 3.5 je prikazana sekvenca slik ter pripadajoča vizualizacija z regijami.



Slika 3.5: Vizualizacija s porazdelitvijo v regije.

Poglavje 4

Eksperimentalno vrednotenje

V tem poglavju bomo najprej predstavili eksperiment, s katerim smo raziskali vpliv šuma na lokacijo epipola, izračunanega iz fundamentalne matrike (Poglavje 4.1). Predstavili bomo zajem posnetkov, na katerih smo testirali naš algoritem (Poglavje 4.2). Predstavili bomo meritve in jih analizirali (Poglavje 4.3). Vsi izračuni so bili opravljeni na računalniku Lenovo X1 Carbon, s procesorjem Intel Core i7-3667U 2.0GHz in 8GB pomnilnika.

4.1 Vpliv šuma na lokacijo epipola

Za računanje fokusa ekspanzije smo najprej poskusili uporabiti fundamentalno matriko, ki smo jo pridobili iz poiskanih korespondenc s pomočjo osemtočkovnega algoritma. V teoriji se v primeru premika kamere naprej epipol v drugi sliki nahaja v fokusu ekspanzije (glej Poglavje 2.5.2), v praksi pa se je izkazalo da izračunan epipol ne sovpada z fokusom ekspanzije. Izračunan epipol ni optimalen zato, ker pri računanju fundamentalne matrike z osemtočkovnim algoritmom matriki naknadno vsilimo rang. Raziskovalci v članku [36] predlagajo izboljšano različico osemtočkovnega algoritma, ki omejitev ranga upošteva že med minimizacijo.

Opravili smo poskus, s katerim smo izmerili odstopanja epipola od dejanskega fokusa ekspanzije ter odstopanje presečišča premic, ki potekajo skozi

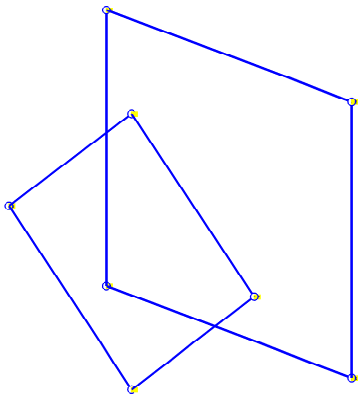
vektorje polja gibanja, od dejanskega fokusa ekspanzije. V programskem okolju MATLAB® smo konstruirali dva kvadrata in na vsakem določili štiri značilne točke (glej Sliko 4.1). S simulacijo premika kamere [31] smo posneli dve sliki, eno pred in eno po premiku kamere naprej. Polje gibanja, ki ga je sestavljalo šestnajst vektorjev, smo pošumili s šumom, vzorčenim iz standardne normalne porazdelitve s standardnim odklonom σ. Na dobljenem polju gibanja smo izračunali fundamentalno matriko in iz nje izrazili epipol v drugi sliki. Izračunali smo še presek premic, ki potekajo skozi vektorje polja gibanja, po postopku, opisanem v Poglavju 3.2. Primer pošumljenega polja in izračunanega epipola ter preseka premic je prikazan na Sliki 4.2.

Za intenzitete šuma $\sigma \in \{0, 1, 2, 3, 4\}$ smo opravili po 1000 poskusov ter izračunali povprečno oddaljenost epipola in preseka premic od pravega fokusa ekspanzije. Rezultati so zbrani v Tabeli 4.1 in prikazani na Sliki 4.3.

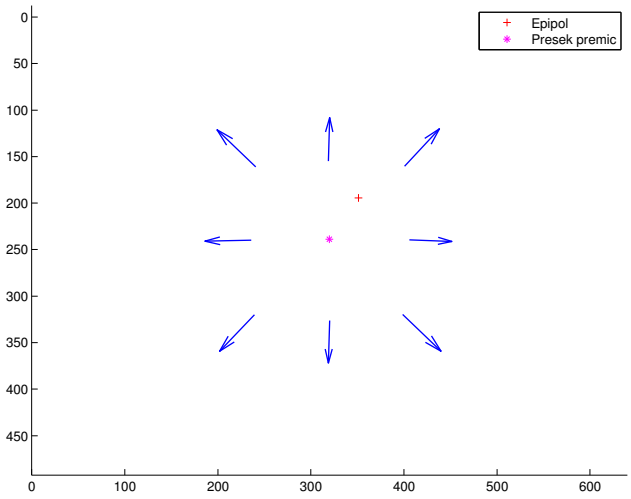
	Epipol [piksli]		Presek premic [piksli]	
σ	Povprečna oddaljenost	Standardni odklon	Povprečna oddaljenost	Standardni odklon
1	92.33	219.69	2.17	1.11
2	1045.11	13062.99	4.37	2.21
3	817.61	5873.41	6.38	3.47
4	738.86	3444.81	8.58	4.53
5	18150.24	54387.47	11.15	5.74

Tabela 4.1: Odstopanje epipola in preseka premic polja gibanja od pravega fokusa ekspanzije za različne intenzitete šuma.

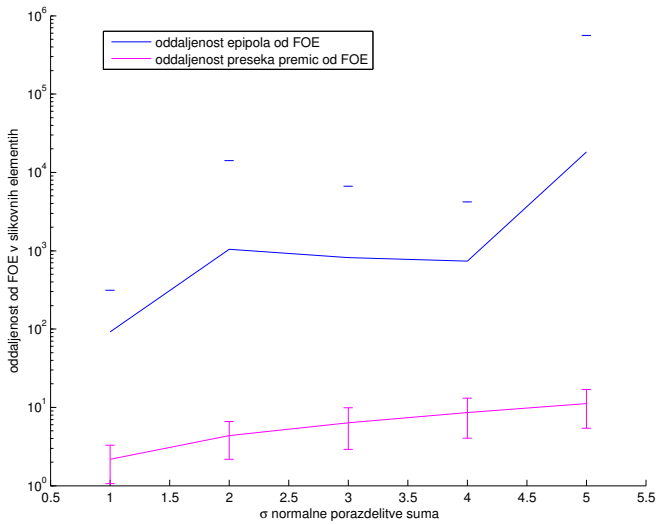
Vidimo, da je odstopanje epipola od dejanskega fokusa ekspanzije že pri majhnem šumu precej veliko, z večanjem šuma pa strmo narašča. Nasprotno pa je presek premic, ki potekajo skozi polje gibanja, tudi ob večanju šuma še vedno blizu dejanskega fokusa ekspanzije.



Slika 4.1: Konstruirana kvadrata v prostoru.



Slika 4.2: Konstruirano polje gibanja in izračunani fokusi ekspanzije.



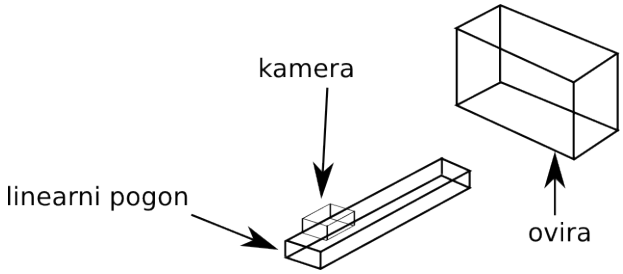
Slika 4.3: Odstopanje izračunanih fokusov ekspanzije od dejanskega fokusa ekspanzije za različne količine šuma.

4.2 Zajem zbirke posnetkov

Za namen eksperimentalnega vrednotenja našega algoritma smo zajeli zbirko posnetkov, v katerih se kamera premika naravnost naprej proti oviri s konstantno hitrostjo. Za vsako oviro smo posneli več posnetkov, tako da je mogoče izračunati varianco metode, ki jo ocenjujemo. Posnetke smo ročno obrezali, da vsi predstavljajo enak premik kamere.

4.2.1 Zajem na linearnem pogonu

Meritve časa do trka smo izvedli na linearnem pogonu, saj smo potrebovali konstantno hitrost. Kamera je bila pritrjena na linearni pogon, obrnjena proti ciljni oviri in od nje oddaljena 60 centimetrov (glej Sliko 4.4). V približno 3.5 sekunde se je kamera premaknila 40 centimetrov v smeri ovire. Zajemali smo slike v velikosti 640x480. Kamera je snemala s hitrostjo 30 slik na sekundo, tako da posnetek enega premika sestavlja 105 slik. Ker linearni pogon ne doseže konstantne hitrosti hipoma, niti se hipoma ne ustavi, smo meritve algoritma opravili le na slikah od 15 do 95. Za vsako oviro na linearnem



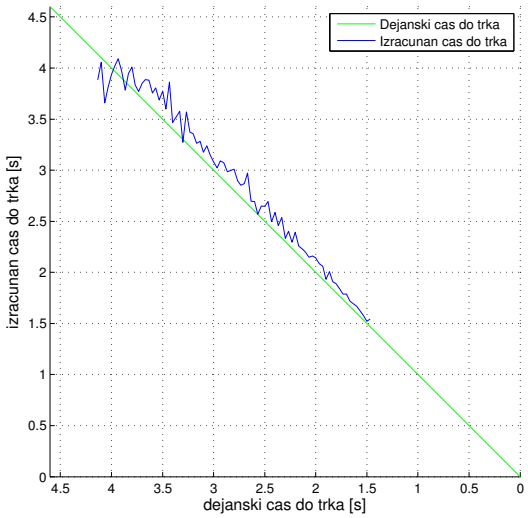
Slika 4.4: Skica sistema z linearnim pogonom

Slika 4.5: Sekvenca slik posnetih na linearnem pogonu

pogonu smo posneli 10 poskusov. Slika 4.5 prikazuje sekvenco slik posnetih na linearnem pogonu, na Sliki 4.6 pa so prikazane izračunane vrednosti časa do trka za en poskus.

Ekstrapolacija dejanskega časa trka

Ker do dejanskega trka kamere z oviro ni prišlo, smo morali čas trka ekstrapolirati. Linearni pogon pospešuje prvih 10 slik in v tem času kamero premakne za 2 centimetra. Za ustavitev potrebuje zadnjih 8 slik, medtem ko se kamera premakne za 1 centimeter. V 88 slikah, ko linearni pogon kamero premika s konstantno hitrostjo, se torej kamera premakne za 37 centimetrov. V trenutku, preden začne linearni pogon upočasnjevati, je kamera od ovire oddaljena 21 centimetrov. Če za 37 centimetrov potrebuje 88 slik, potrebuje

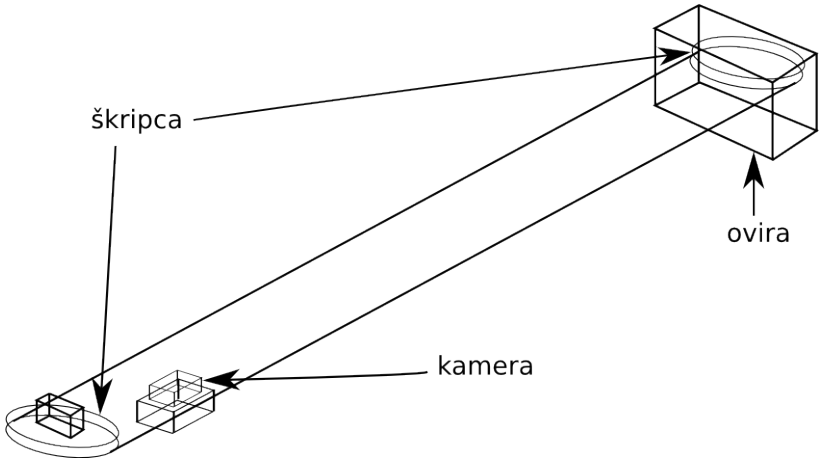


Slika 4.6: Graf ene meritve na linearnem pogonu

za 1 centimeter približno 2.38 slike ter za $37 + 21 = 58$ centimetrov približno 138 slik. V nadaljnjih meritvah bomo čas 4.6 sekunde, to je čas, ob katerem bi posneli 138. sliko, uporabili za čas trka kamere z oviro.

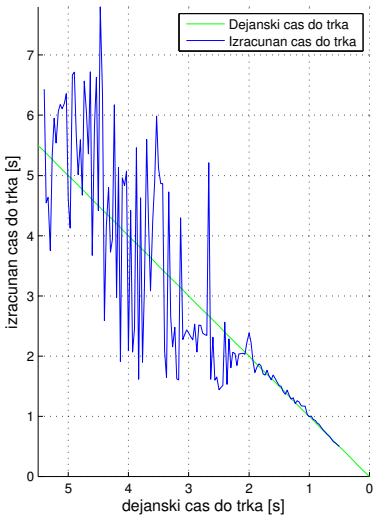
4.2.2 Zajem na škripcih

Ker lahko linearni pogon kamero premakne le za 40 centimetrov, smo konstruirali še poskus, ki nam je omogočil zajem posnetkov večjih premikov in pri višji hitrosti. Izdelali smo dva škripca in ju povezali s sklenjeno vrvjo. V vrv smo vpeli čolniček z gladkim dnom, na katerem je bila fiksirana kamera. Na enega izmed škripcev smo namestili električni motor, pred drugega pa smo postavili ciljno oviro (glej Sliko 4.7). Razpon med škripcema je znašal 5 metrov, ki jih je s konstantno hitrostjo kamera prepotovala v 5 sekundah in pri hitrosti snemanja 30 slik na sekundo posnela 150 slik. Dejanski čas trka smo določili eksperimentalno z mehko oviro. Do trka pride ob času 5.5 sekunde. Zajemali smo slike v velikosti 640x480, za vsako oviro pa smo posneli 5 poskusov. Slika 4.8 prikazuje sekvenco slik posnetih na linearnem pogonu, na Sliki 4.9 pa so prikazane izračunane vrednosti časa do trka za en poskus.



Slika 4.7: Skica sistema s škripcema

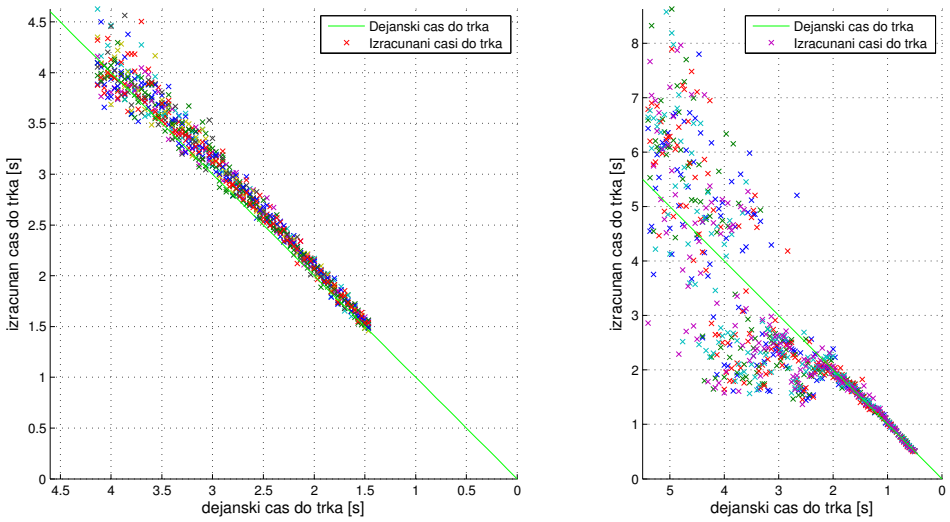
Slika 4.8: Sekvenca slik posnetih na škripcih



Slika 4.9: Graf ene meritve na škripcih

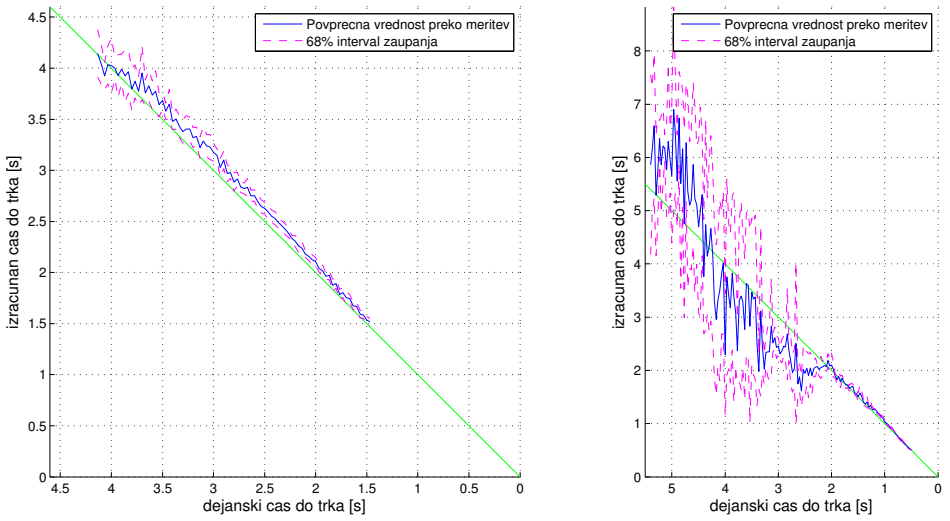
4.3 Rezultati

Rezultati izračunanih časov do trka za vsako sliko in vsak poskus so prikazani na Sliki 4.10, povprečje in standardni odklon izračunanih časov do trka preko vseh meritev ob določenem času pa je prikazano na Sliki 4.11. Vidimo, da s približevanjem oviri standardni odklon meritev pada, kar pomeni, da natančnost algoritma narašča obratno sorazmerno z oddaljenostjo od ciljne ovire. Do tega pride zato, ker se slika ovire na veliki razdalji spreminja počasi in so zaznani premiki zelo majhni in ne vsebujejo dovolj informacije za natančen izračun časa do trka. Še posebej pri poskusu s škripci jasno vidimo, da algoritem pri veliki oddaljenosti od ovire ne daje realnih rezultatov. Šele približno 2 sekundi pred trkom se izračunan čas zares približa dejanskemu času do trka.



(a) Poskus na linearnem pogonu (b) Poskus s škripci

Slika 4.10: Prikaz vseh meritev



(a) Poskus na linearnem pogonu (b) Poskus s škripci

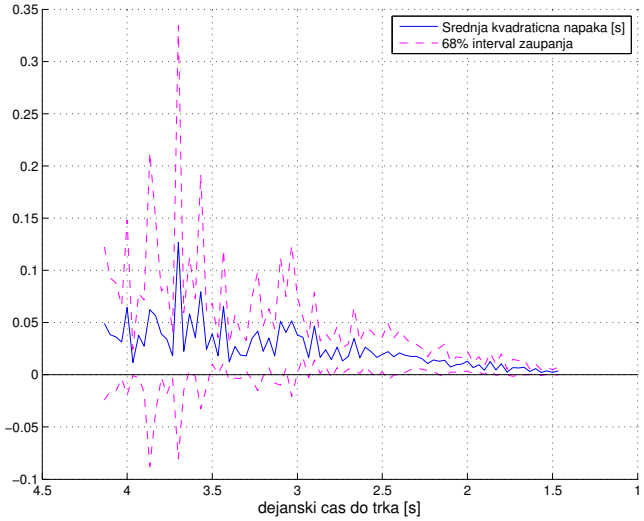
Slika 4.11: Povprečje in standardni odklon izračunanih časov do trka

4.3.1 Odstopanje od dejanskega časa do trka

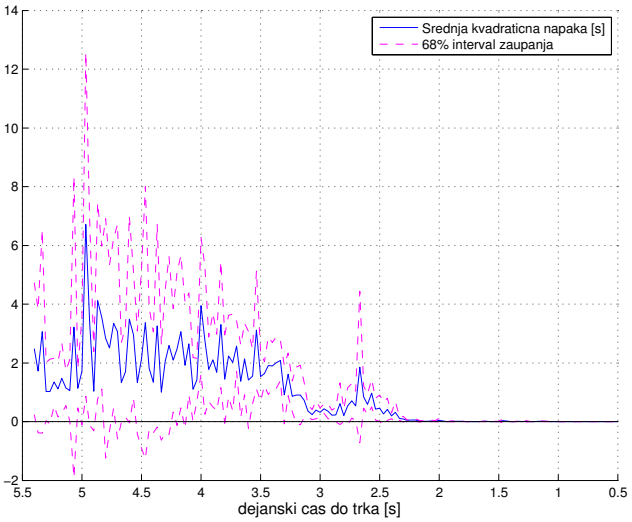
Sliki 4.12a in 4.12b prikazujeta povprečje in standardni odklon kvadratov odstopanj izračunanih časov do trka od dejanskega časa do trka ob določenem času. Ko kamera pride dovolj blizu ovire, je odstopanje izračunanega časa do trka od dejanskega časa do trka majhno. Časovni interval poskusa smo razdelili v skupine po 0.5 sekunde. V vsaki skupini smo izračunali povprečje in standardni odklon kvadratov odstopanj izračunanih časov do trka od dejanskega časa do trka preko vseh poskusov in vseh slik v skupini. V Tabelah 4.2 in 4.3 vidimo, da je povprečno odstopanje na začetku posnetkov veliko, v obeh primerih pa se na neki točki izračun stabilizira. Pri poskusu z nižjo hitrostjo se stabilizira prej, že okoli 4 sekunde pred trkom, pri poskusu z višjo hitrostjo pa približno 2.5 sekunde pred trkom. Podatki iz tabel so prikazani še v Slikah 4.13a in 4.13b.

Čas do trka [s]	Kvadrat odstopanja od dejanskega TTC	Standardni odklon kvadrata odstopanja
4.3–3.8	0.042	0.078
3.8–3.3	0.030	0.048
3.3–2.8	0.027	0.038
2.8–2.3	0.028	0.028
2.3–1.8	0.028	0.028

Tabela 4.2: Poskus na linearnem pogonu

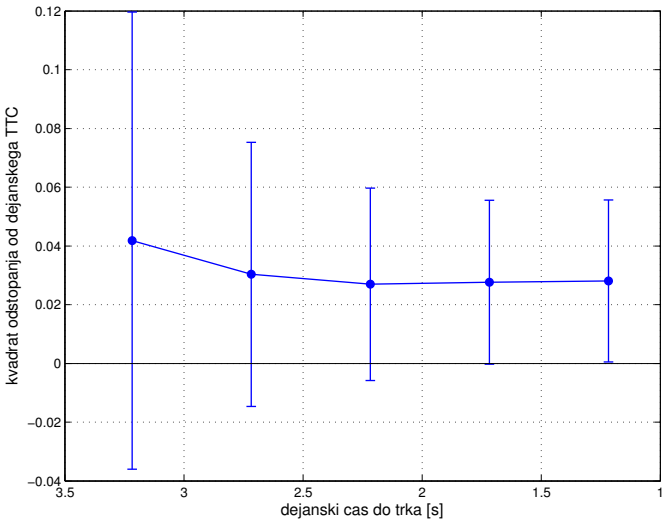


(a) Poskus na linearnem pogonu

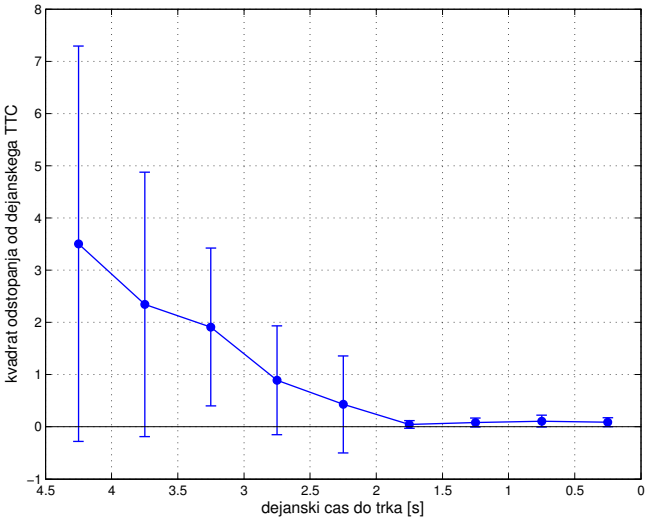


(b) Poskus s škripci

Slika 4.12: Povprečje in standardni odklon odstopanj izračunanega od dejanskega časa do trka.



(a) Poskus na linearnem pogonu



(b) Poskus s škripci

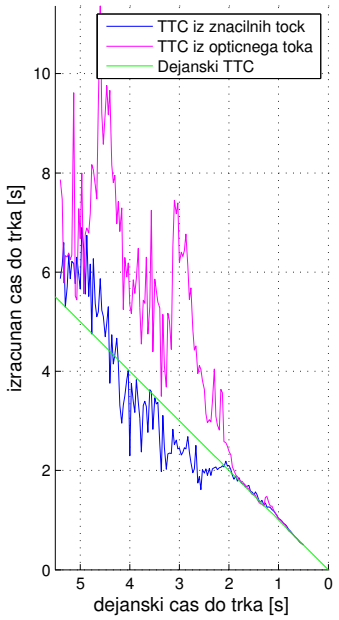
Slika 4.13: Povprečje in standardni odklon odstopanj izračunanega od dejanskega časa do trka za časovne skupine

Čas do trka [s]	Kvadrat odstopanja od dejanskega TTC	Standardni odklon kvadrata odstopanja
5.0–4.5	3.506	3.787
4.5–4.0	2.344	2.537
4.0–3.5	1.911	1.517
3.5–3.0	0.891	1.047
3.0–2.5	0.429	0.927
2.5–2.0	0.045	0.077
2.0–1.5	0.083	0.087
1.5–1.0	0.108	0.117
1.0–0.5	0.087	0.087

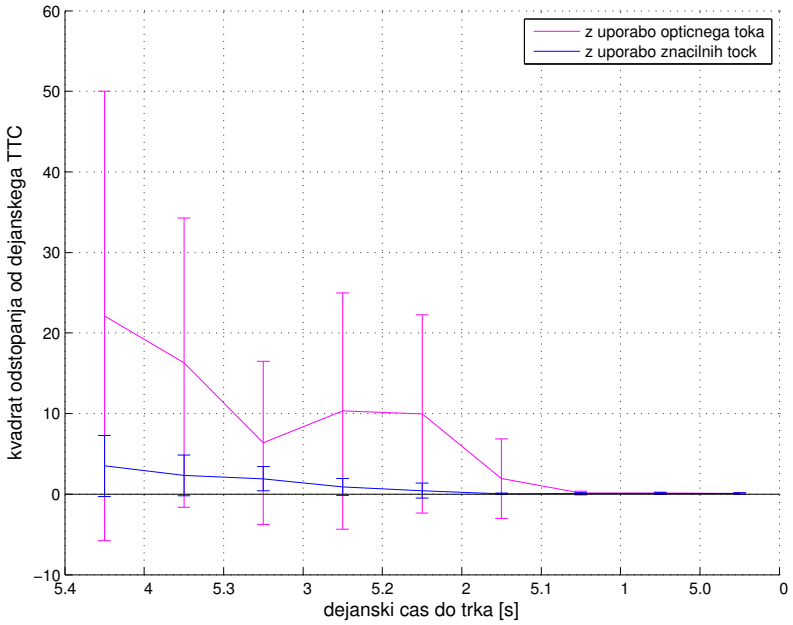
Tabela 4.3: Poskus na škripcih

4.3.2 Primerjava z uporabo optičnega toka

V naši metodi smo za ocenitev polja gibanja uporabili značilne točke. Testirali smo tudi enega izmed alternativnih načinov določitve polja gibanja, redek optični tok. Korake 1–6 v Algoritmu 4, ki ocenijo polje gibanja, smo zamenjali z izračunom redkega optičnega toka z uporabo piramidne implementacije algoritma Lucas-Kanade [3]. Povprečne vrednosti meritev za obe metodi so prikazane na Sliki 4.14, kvadratično odstopanje od dejanskega časa do trka pa na Sliki 4.15. Vidimo, da pri veliki oddaljenosti od ovire nobena od metod ne dosega velike natančnosti, čeprav ima metoda z značilnimi točkami manjšo varianco. Ko se oviri dovolj približamo, pa dajeta metodi primerljive rezultate. Odstopanje izračunanega od dejanskega časa do trka za metodo z optičnim tokom je prikazano v Tabeli 4.4.



Slika 4.14: Primerjava povprečnega izračunanega časa do trka z uporabo značilnih točk in z uporabo optičnega toka.



Slika 4.15: Primerjava kvadratov odstopanj časa do trka z uporabo značilnih točk in z uporabo optičnega toka.

Čas do trka [s]	Kvadrat odstopanja od dejanskega TTC	Standardni odklon kvadrata odstopanja
5.0–4.5	22.115	27.908
4.5–4.0	16.320	17.968
4.0–3.5	6.371	10.128
3.5–3.0	10.317	14.648
3.0–2.5	9.962	12.308
2.5–2.0	1.922	4.918
2.0–1.5	0.137	0.228
1.5–1.0	0.119	0.128
1.0–0.5	0.082	0.088

Tabela 4.4: Odstopanje od dejanskega TTC za metodo z uporabo optičnega toka.

4.3.3 Hitrost izračuna

Izmerili smo hitrost izračuna časa do trka za našo metodo z uporabo značilnih točk in metodo z uporabo optičnega toka. Meritve smo opravili za dve velikosti slik, rezultati pa so prikazani v Tabeli 4.5. Metoda z značilnimi točkami je hitrejša pri manjši velikosti slike, pri večji velikosti slike pa počasnejša. Direktna primerjava časov ni mogoča, saj je večina metode z značilnimi točkami implementirana v okolju MATLAB®, večina metode z uporabo optičnega toka pa v C++ knjižnico OpenCV [5]. Hitrost izračuna je manjša pri večji

Velikost slike [piksel]	Hitrost metode z značilnimi točkami [slik/sekundo]	Hitrost metode z optičnim tokom [slik/sekundo]
320x240	67.66	53.18
640x480	21.29	24.29

Tabela 4.5: Hitrost izračuna časa do trka.

velikosti slike, zato bi morali pri realni aplikaciji najti kompromis med hitrostjo izračuna in oddaljenostjo od ovire, na kateri želimo natančen izračun časa do trka.

Poglavje 5

Sklep

V diplomski nalogi smo izdelali algoritem za izračun časa do trka z objektom, ki ga opazujemo s kamero. Za izračun časa do trka smo uporabili vektorje polja gibanja. Polje gibanja smo izračunali z uporabo ujemanj značilnih točk iz zaporednih slik. Za detektor značilnih točk smo izbrali FAST [28], opisnike točk pa smo pridobili z algoritmom BRIEF [6]. Za povečanje robustnosti smo zaporedne slike obravnavali kot sistem dveh kamer, kar nam je omogočalo vpeljavo omejitev, ki jih takemu sistemu vsili epipolarna geometrija. Za robusten izračun epipolarne geomerije smo uporabili algoritem RANSAC [9]. Epipol, izračunan iz pridobljene fundamentalne matrike, se je zaradi šumnih podatkov izkazal za neprimeren način računanja fokusa ekspanzije. Zato smo fokus ekspanzije izračunali kot presek premic, ki potekajo skozi vektorje polja gibanja. Predstavili smo postopek, kako iz podatkov o času do trka za posamezne točke sklepamo na čas do trka za celotno sliko.

V sklopu diplomske naloge je bila zajeta zbirka posnetkov, na katerih se kamera s konstantno hitrostjo približuje objektu. Za vsak objekt je bilo zajeto več posnetkov in s tem omogočena analiza variance algoritma.

Zajeta zbirka je bila uporabljena za eksperimentalno vrednotenje izdelanega algoritma. Analizirali smo odstopanje izračunanega časa do trka od dejanskega časa do trka ter varianco tega odstopanja. Ugotovili smo, da ima algoritem pri veliki oddaljenosti od objekta visoko varianco, ko pa se

objektu dovolj približamo, da so spremembe velikosti dovolj vidne na sliki, pa je izračunan čas zelo blizu dejanskemu času do trka.

Algoritem smo primerjali tudi z modificiranim algoritmom, kjer smo za izračun polja gibanja uporabili optični tok. Izkazalo se je, da ima algoritem z uporabo značilnih točk manjšo varianco na veliki oddaljenosti od objekta, v bližini objekta pa so izračuni obeh algoritmov zelo podobni.

5.1 Smernice za nadaljnji razvoj

Algoritem je sestavljen iz več ločenih delov, zato je možnosti za izboljšave precej. Lahko bi eksperimentalno preizkusili še druge hitre detektorje značilnih točk, kot je ORB [29], in opisnike, kot sta na primer FREAK [1] in BRISK [19]. Če bi imeli dovolj dobra ujemanja, bi se morda lahko znebili izračuna fundamentalne matrike in s tem pohitrili algoritem.

Ko je kamera daleč od objekta, se slika objekta zelo malo spreminja. Natančnost na večjih razdaljah bi morda izboljšali tako, da bi v neki majhni okolici fokusa ekspanzije izračunali gosto polje gibanja in potem iz tega polja izračunali čas do trka.

Isti problem bi lahko naslovili tako, da bi slike izpuščali, dokler ne bi bili detektirani premiki dovolj veliki za natančnejši izračun časa do trka.

Ko imamo izračunan lokalni čas do trka za posamezne točke, bi jih lahko grupirali glede na evklidsko razdaljo med njimi in glede na podoben čas do trka. Na ta način bi lahko zaznali več objektov v sceni z različnimi časi do trka.

Trenutno je algoritem implementiran v programskem okolju MATLAB®. Algoritem bi lahko implementirali v nižjenivojskem jeziku, kot je na primer C, ga optimizirali in preizkusili na manj zmogljivi strojni opremi.

Literatura

[1] Alexandre Alahi, Raphael Ortiz, in Pierre Vandergheynst. Freak: Fast retina keypoint. V *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012.

[2] Herbert Bay, Tinne Tuytelaars, in Luc Van Gool. Surf: Speeded up robust features. V *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[3] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

[4] Jean-Yves Bouguet in Pietro Perona. Visual navigation. `http://www.vision.caltech.edu/bouguetj/Motion/navigation.html`. Dostopano: 1.9.2013.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, in Pascal Fua. Brief: Binary robust independent elementary features. V *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[7] Ted Camus. Calculating time-to-contact using real-time quantized optical flow. *National Institute of Standards and Technology NISTIR*, 5609, 1995.

[8] Benjamin Cohen in Jeffrey Byrne. Inertial aided sift for time to collision estimation. V *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1613–1614. IEEE, 2009.

[9] Martin A Fischler in Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Dariu M Gavrila in Vasanth Philomin. Real-time object detection for "smart" vehicles. V *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 87–93. IEEE, 1999.

[11] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[12] Chris Harris in Mike Stephens. A combined corner and edge detector. V *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[13] Richard Hartley in Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[14] Richard I Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, 1997.

[15] Berthold KP Horn, Yajun Fang, in Ichiro Masaki. Time to contact relative to a planar surface. V *Intelligent Vehicles Symposium, 2007 IEEE*, pages 68–74. IEEE, 2007.

[16] Berthold KP Horn in Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981.

[17] Michal Irani in P Anandan. About direct methods. *Vision Algorithms: Theory and Practice*, pages 267–277, 2000.

[18] David N Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5:437–459, 1976.

[19] Stefan Leutenegger, Margarita Chli, in Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. V *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[20] Shuangzhe Liu in Gõtz Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *Int. J. Inform. Syst. Sci*, 4(1):160–177, 2008.

[21] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62, 1987.

[22] David G Lowe. Object recognition from local scale-invariant features. V *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[23] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[24] Amaury Nègre, Christophe Braillon, James L Crowley, in Christian Laugier. Real-time time-to-collision from variation of intrinsic scale. V *Experimental Robotics*, pages 75–84. Springer, 2008.

[25] Robyn Owens. The motion field. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html`. Dostopano: 2.8.2013.

[26] Eric Raphael, Raymond Kiefer, Pini Reisman, in Gaby Hayon. Development of a camera-based forward collision alert system. *SAE International*, 4(1):467–478, 2011.

[27] Edward Rosten in Tom Drummond. Fusing points and lines for high performance tracking. V *Computer Vision, 2005. ICCV 2005. Tenth*

IEEE International Conference on, volume 2, pages 1508–1515. IEEE, 2005.

[28] Edward Rosten in Tom Drummond. Machine learning for high-speed corner detection. V *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[29] Ethan Rublee, Vincent Rabaud, Kurt Konolige, in Gary Bradski. Orb: an efficient alternative to sift or surf. V *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[30] Henry Schneiderman in Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. V *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 746–751. IEEE, 2000.

[31] Milan Sonka, Vaclav Hlavac, Roger Boyle, et al. Image processing, analysis, and machine vision. 1999.

[32] Selim Temizer. Optical flow based robot navigation. `http://people.csail.mit.edu/lpk/mars/temizer_2001/Optical_Flow/`. Dostopano: 1.9.2013.

[33] Philip HS Torr in Andrew Zisserman. Feature based methods for structure and motion estimation. *Lecture notes in computer science*, pages 278–294, 2000.

[34] Alessandro Verri in Tomaso Poggio. Motion field and optical flow: Qualitative properties. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(5):490–498, 1989.

[35] Paul Viola in Michael Jones. Rapid object detection using a boosted cascade of simple features. V *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[36] Yinqiang Zheng, Shigeki Sugimoto, in Masatoshi Okutomi. A practical rank-constrained eight-point algorithm for fundamental matrix estimation.